```
////////////////////////////////////////////////////////////////////
 ////////////
// Drive task
// Interprets user input & auton commands and sends to drive motors
// To Do: Comments, tidy up

void run_drive(void* params) {

    double currentTime = 0;
    double leftPower = 0;
    double rightPower = 0;
    double leftSpeed = 0;
    double rightSpeed = 0;

    double lastAngle = 0;
    double turnPulse = 0;

    double slewRate = 2;

    int turnGoodCount = 0;

    while (true) {


        if (usingGyro) {
            direction = gyroDirection/10;  // gyroDirection is updated
             by gyro code, direction is used by drive code
        }
        else {
            // maybe using compass/encoders?
            // direction = compassDirection
        }

        // This is where the fun begins

        double forward = 0;
        double turn = 0;

        // Calculate useful information
        currentTime = pros::millis();           // current time to
         determine if timed out

        // find where encoders are right now
        double currentDistLeft = getLeftEnc();
        double currentDistRight = getRightEnc();
        currentDist = (currentDistRight + currentDistLeft)/2;

        if (controller.get_digital(BTN_ABORT)) {    // if user wants
         to abort, stop auton move
            autoMode = DRIVEMODE_USER;
        }

        // auto functions
        if (autoMode != DRIVEMODE_USER) {   // If auton is asking for
         drive to move

            if (drivingToPos) {         // keep calculating new angle
             & distance to stay on-target
                // Must write position tracking algorythm first
                driveTo(targetS, targetX, targetY);
            }

            forward = autoSpeed;        // autoSpeed is speed asked
             for, forward will be sent to drive motors

            if (autoMode == DRIVEMODE_TURN)  {  // if we are only
             turning, make translational speed 0
                forward = 0;
                autoSpeed = 0;
            }

            if (autoMode == DRIVEMODE_DIST) {   // If auto move should
             end with a distance
                double slowDown = (targetDistance - currentDist) /
                 (0.75 * ticksPerTile);

                forward *= slowDown;

                if (autoSpeed > 0 && forward < minForward) forward =
                 minForward;
                if (autoSpeed < 0 && forward > minForward) forward = -
                 minForward;

                if (forward > 127) forward = 127;   // Cap max and min
                 speed
                if (forward < -127) forward = -127;

                // Terminate contition for distance
                if (autoSpeed > 0) {
                    if (currentDist > targetDistance) autonComplete =
                     true;
                }
                else {
                    if (currentDist < targetDistance) autonComplete =
                     true;
                }
            }

            if (currentTime > autoTimeOut + recordedTime &&
             autoTimeOut > 0) {     // If auton move has timed out,
             stop driving
```

```
        autonComplete = true;
        std::cout << "Time Out - ";
    }

    // Turn code
    double driveMag = autoSpeed;
    double seek = targetDirection;
    double angle = 0;

    if (turnMode == TURNMODE_GYRO) {
        angle = seek - direction;
    }
    else if (turnMode == TURNMODE_ENCODER) {
        angle = (recordedDistRight - recordedDistLeft)/2;
        angle -= (currentDistRight - currentDistLeft)/2;
        angle /= ticksPerDegree;
    }

    if (angle < 0) angle += 360;
    if (angle > 180) angle -= 360;

    angle /= (2 * turnRate);
    angle *= 127;
    if (driveMag < minSpeed) {
        angle *= 2;
    }

    if (angle < -maxTurn) angle = maxTurn;
    if (angle > maxTurn) angle = maxTurn;

    if (driveMag > minSpeed) {
        if (angle < 0) {
            if (angle > -2) {
                angle = 0;
            }
            else if (angle > -4) {
                angle = -4;
            }
        }
        else {
            if (angle < 2) {
                angle = 0;
            }
            else if (angle < 4) {
                angle = 4;
            }
        }
    }
    else {
        turn = angle;
        angle = abs(angle);
            if (angle < minSpeed) {
                if (((lastAngle > 0) && (turn < 0)) || ((lastAngle
                 < 0) && (turn > 0))) {
                    angle = 0;
                }
                else {
                    if (angle > minSpeed/5) {
                        angle = minSpeed;
                    }
                    else {
                        turnPulse++;
                        if (turnPulse < pulseTime) {
                            angle = minSpeed;
                        }
                        else {
                            angle = 1;
                            if (turnPulse > pulsePause) {
                                turnPulse = 0;
                            }
                        }
                    }
                }
            }
            if (turn < 0) angle *= -1;
        }

        turn = angle;

        if (autoSpeed == 0 || autoMode == DRIVEMODE_TURN) {
            if (abs(direction - targetDirection) < turnAccepted) {
                turnGoodCount++;
                if (turnGoodCount > 3)
                    autonComplete = true;
            }
            else {
                turnGoodCount = 0;
            }
        }

        lastAngle = angle;

    }
    // Auto-move is complete, so stop moving
    if (autonComplete) {
        autonComplete = false;
        autoMode = DRIVEMODE_USER;
        forward = 0;
        turn = 0;
        autoSpeed = 0;
        drivingToPos = false;
        nextCommand = true;
```

```
            std::cout << "Drive Move Done: " << currentTime <<
             std::endl;
        }

        // User controls
        if (autoMode == DRIVEMODE_USER) {

            // Tank controls
            leftSpeed = controller.get_analog(ANALOG_LEFT_Y);
            rightSpeed = controller.get_analog(ANALOG_RIGHT_Y);

            if (abs(leftSpeed) < deadZone) leftSpeed = 0;
            if (abs(rightSpeed) < deadZone) rightSpeed = 0;
        }
        else {
            leftSpeed = forward - turn;
            rightSpeed = forward + turn;
        }

        // Constant-speed override
        if (speedOverride) {
            leftSpeed = leftRunSpeed;
            rightSpeed = rightRunSpeed;
        }

        // dampen motors so they don't spike current
        rightPower = rightPower + ( (rightSpeed - rightPower) /
         slewRate );
        leftPower = leftPower + ( (leftSpeed - leftPower) /
         slewRate );

        // std::cout << "gyro: " << gyroDirection << std::endl;

        // Send speeds to drive motors
        drive_left_1.move_voltage(leftPower * 12000 / 127);
        drive_left_2.move_voltage(leftPower * 12000 / 127);
        drive_left_3.move_voltage(leftPower * 12000 / 127);
        drive_right_1.move_voltage(rightPower * 12000 / 127);
        drive_right_2.move_voltage(rightPower * 12000 / 127);
        drive_right_3.move_voltage(rightPower * 12000 / 127);

        pros::delay(10);   // don't hog cpu
    }
}
```