Question 2

The open-loop forward kinematics works under an assumption that wheel encoder measurements will always be error-free. This method applies standard values to convert each encoder step into a standardized distance and maintains that wheelbase remains consistent. The robot's wheels face potential slipping conditions while operating on surfaces that could be uneven so each wheel step leads to slightly different movements in real-world situations. The calculated position usually diverges from the robot's actual position because slight wheel grip differences or small robot part degradation happens.

This system lacks built-in self-correction capabilities during the execution process. Without sensor feedback the robot performs an open-loop operation where it follows the previously calculated path. Repeated small errors in the sequence of steps result in an increased distinction between theoretical and actual positional data which creates drift.

Limitation:
Computers use floating-point numbers to execute these calculations but such numbers sometimes introduce small rounding inaccuracies during the process. Rounding errors in the computer system tend to be insignificant individually but they can compound when numerous movements occur.

The methodology functions well as a positioning predictor but its effectiveness depends on perfect conditions because it does not receive feedback and encounters errors through step-size measurement and numeric rounding.

Question 3

A robot's position estimation depends on measuring wheel rotation distances through odometry systems. The method enables easy and inexpensive movement measurement yet produces several errors and limitations.

Error accumulation represents a main drawback for this system. The small wheel rotation inaccuracies from each cycle result in cumulated positioning errors. Tiny errors in knowing the wheel radius or slight misalignments during the robot construction process will lead to an accumulating drift between the actual position and the estimated position. The continuous growth of such errors accumulates and gets termed “drift,” resulting in large positional discrepancies when traveling extended distances or executing numerous turns.

The movement of wheels relative to the surface so that they do not move forward as intended constitutes wheel slippage as an error source. During movement on slippery or uneven ground the wheels tend to spin without achieving the expected robot advancement distance. The sensors end up recording movements that fail to occur which results in deteriorating accuracy. Sensor noise coupling with wheel encoder resolution limits introduces random errors because these errors accumulate throughout time.

The robots incorporate systematic errors which result from poor manufacturing quality of their components. The robot experiences incorrect calculations because its wheels are misaligned or their dimensions slightly differ from each other. Physical adjustments together with calibration are required to rectify these errors.

The combination of sensors through fusion stands as an effective way to enhance odometry performance. The success of this method requires integrating odometry results with additional sensor data including inertial measurement units (IMUs) together with GPS receivers and laser scanners. Correct wheel calibration combined with well-designed robot construction will help decrease both systematic errors and wheel misalignments.

The utility of odometry for basic navigation exists yet its accuracy remains confined by drift as well as wheel slippage and sensor noise and mechanical imperfectness. When extra sensors are combined with proper calibration systems the overall reliability of performance improves.

Question 4

When supplying values beyond hardware or mathematical capabilities to the `**diff_drive_inverse_kin**` to `**move_steps**` sequence disruptions occur. When commanded to move long distances or achieve speeds beyond motor capabilities the computed wheel speeds expressed in steps per second may produce values that exceed hardware limits. The resulting commands may surpass the robot motors' maximum limitations of speed and torque capabilities. Extremely high turning angles (omega_rad) result in swift rotation motion of the robot. The inverse kinematic algorithm computes wheel step commands that could result in either unrealistically high values or completely undefined calculations because of small numbers that appear in the calculations.

The application of negative values presents a possibility of system failure. The implementation of our code supports negative speed values when moving backward together with negative distance values or negative angular measurements to specify direction yet requires precise sign management rules. The incorrect handling of wheel movements may produce wheel speed calculations that fail to match actual wheel movements particularly when the robot executes in-place turns. Small floating-point numbers can encounter rounding errors that create significant issues which force the robot to deviate from its expected motion or produce unintended movements.

Where should it be checked?
The system requires multiple checks and fixes throughout its points to prevent issues from arising. The diff_drive_inverse_kin function needs to perform input validation checks which ensure speed and distance values stay within robot specifications while also validating possible unreasonable turning rates. The function would either set invalid values to maximum allowable ranges or trigger an error. Before running move_steps, both control loops and higher-level planning codes need to verify that generated step commands stay within hardware limits. Such validation checks guarantee that the motors receive operational commands within their capabilities to avoid robotic mishaps or malfunction.

Early evaluation and correction of possible extreme values protects the robot from unsafe or inaccurate motions while guaranteeing its operation according to the issued commands.

Question 5

Part 2

The precise speed at which the controller operates through Hz directly influences the accuracy of odometry calculations. Odometry functions by calculating wheel distance measurements which it uses to determine the robot’s total position through integration across multiple time periods. The robot divides its motion into smaller and more precise time segments when running control loops at higher frequencies. The size of errors within each time interval remains small because integration results in more precise overall movement control. The system maintains high response speed to motion changes and sensor noise gets minimized because correction algorithms operate at increased frequency.

Each update duration becomes extended when the control loop operates at lower frequencies (lower Hz). Big movements in the controller introduce magnified consequences from wheel measurement mistakes and movement computation inaccuracy. Integration during odometry accumulates multiple errors as time advances thereby producing bigger cumulative errors. The odometry system has reduced accuracy in

tracking wheel speed changes when the robot travels far between software updates because it cannot adjust deviations swiftly. The true robot position becomes less precise because of this occurrence.

Running the system at higher Hz values enables better control response since the system uses real-time sensor feedback to provide smooth control adjustments. High frequency operation of running algorithms in robots may create additional processing obstacles and greater sensor noise sensitivity. A practical control loop function exists between fast enough operation to reduce integration errors and system performance ruined by sensor noise.

The accuracy of odometry improves through higher controller Hz because smaller movement increments result in less accumulation of error during updates.

Part3

Robot speed essentially determines how accurately odometry works. The position calculations with odometry depend on exact wheel rotation measurements to establish robot movement throughout time. Each wheel position update gives information about a minimal distance at low robot speeds thus enhancing accuracy. The system performs better in calculating odometry results due to the increased precision of measuring small distance increments. Each measurement error remains small enough to lose significance during the accumulation of numerous small wheel movements.

The rising robot speed extends the distance which builds up between successive control loop updates. A longer travel distance will amplify errors in wheel rotation measurement along with any slight slips between wheels. A robot operating at higher speeds suffers more slippage that often creates unpredictable movement behavior when wheels lose traction contact. Because the encoders lack precision in recording this phenomenon the calculated and actual positions show increased mismatch.

When operating at higher speeds the control system fails to get enough samples of wheel rotations to record rapid motion changes effectively. The coarse data resolution produces bigger segment approximations of the robot's path which leads to greater probabilities of large integration errors. Because the odometry system requires guessing the path during longer intervals it produces errors that lead to position drift.

The accuracy of odometry measurements improves when speeds decrease because the distance for measurements between updates becomes shorter and wheel slip minimizes and measurement precision increases. Faster vehicle speeds result in bigger integration errors and measurement inaccuracies because updates are less frequent and wheel slip increases along with sensor noise becoming more pronounced.

Part 4

The following list shows the major error factors which receive two-sentence descriptions along with their corresponding explanation and correction approach:

Wheel Slippage
Inaccurate distance measurement occurs when the robot wheels slip because actual movement becomes shorter than their calculated wheel rotations.
Using high-traction surfaces combined with optimized wheel materials or implementing slip detection sensors enables better odometry calculation through adjusted measurements.

Environment and Human error
If the surface is uneven there might be deviation from the original path as well as
When placing the robot to the starting position human error might lead to the robot being misaligned which can deviate the robot from the expected path.
Using better wheel or improving surface interaction can help fix this.

Calibration Errors in Physical Parameters
The odometry calculation produces inaccurate results because the measurement of wheel diameter along with wheelbase and steps-to-millimeter conversion contains systematic errors.
Regular mechanical property checks alongside precise measurement equipment can lower the amount of errors present in the robot system.

Integration Errors Due to Controller Frequency
When control loop sampling occurs at low frequencies the integrated motion increments become larger which results in increased error accumulation throughout the course of measurement.
The controller performs better when its update frequency is higher because it detects smaller motions while minimizing accumulation errors.

Floating-Point Arithmetic and Rounding Errors
Over repeated computations with floating-point arithmetic rounding errors add up even though they remain small individually thus producing inaccuracies in position calculations.
By applying more precise numerical computations or error correction methods these errors in the control system can be minimized.

Question 6

Part a

The implemented code demonstrates that odometry functions well when performing multiple movement sequences yet shows its restricted performance since positional errors tend to accumulate as the sequence extends. This robot implementation requires odometry accuracy for movements which start with forward motion followed by turning motions and ending with path retracement. The positioning estimate becomes more erroneous after the forward movement because of wheel measuring errors or wheel slippage which then worsens when the robot executes a 180-degree turn before trying to get back to its initial spot.

The testing sequence reveals that odometry operates as an open-loop system because it has no built-in self-correction capability after sending control commands. Complex behaviors including obstacle navigation or following set paths experience problem accumulation that results in notable position discrepancy between planned and actual trajectories. Additional sensors or feedback mechanisms including GPS and vision systems together with inertial measurement units become necessary to periodically update the robot's position because of accumulated errors.

Part b

The robot's immediate stopping functionality requires sensor monitoring inside the movement loops which can be integrated into my current program structure. The move_straight function should contain an operational check during control cycles to read front sensor data until the threshold drops too low before immediately triggering a stop function (epuckcomm.stop_all()) and exiting the loop.

The implementation faces difficulties because my current program functions through open-loop control by sending commands one after another without instantaneous feedback. The addition of continuous sensor checks would require me to redesign code sections to operate either as event-driven applications or multi-threaded implementations for sensor data processing alongside movement instructions. The proper filtering of sensor data becomes important since improper management could lead to unnecessary vehicle stops because of false triggers. An emergency stop becomes implementable but it reveals design weaknesses in my open-loop system that requires me to develop better methods for real-time decision handling during asynchronous events.