

ECEN 3753 Final Project: Labyrinth (FreeRTOS)

Ben Mackert

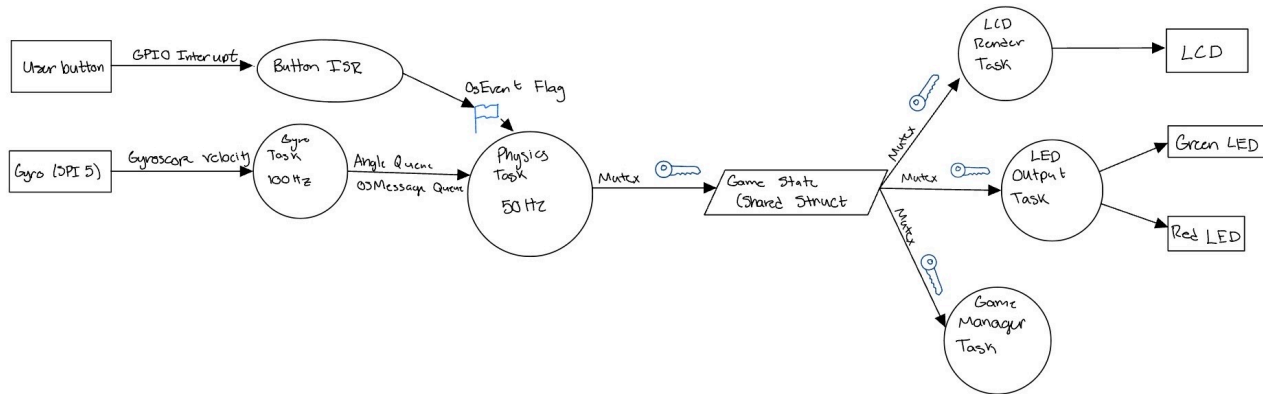


Figure 1: Task diagram: data flow, inter-task communication and mutex

Final Functional Test Status

#	Test Name	Procedure	Expected Result	Status
1	Tilt moves drone +X	Tilt board in +X direction	Drone moves right on LCD	Pass
2	Tilt moves drone -X	Tilt board in -X direction	Drone moves left on LCD	Pass
3	Tilt moves drone +Y	Tilt board in +Y direction	Drone moves down on LCD	Pass
4	Tilt moves drone -Y	Tilt board in -Y direction	Drone moves up on LCD	Pass
5	Wall collision stops drone	Tilt drone into a wall	Drone stops at wall, velocity zeroed, does not pass through	Pass
6	Hole collision ends game	Tilt drone over a hole	Game over screen with "Fell in hole!" message	Pass
7	Waypoint collection (in-order)	Tilt drone onto active waypoint	Waypoint turns grey, next turns green, WP counter increments	Pass
8	Waypoint out-of-order ignored	Tilt drone onto waypoint 3 before reaching waypoint 2	No credit awarded, counter unchanged	Pass
9	Disruptor activation	Press and hold user button while energy ≥ 6000 mJ	Ball turns magenta, drone tunnels walls and ignores holes	Pass
10	Disruptor below minimum energy	Try to activate with energy < 6000 mJ	Disruptor does not activate	Pass
11	Energy recharge	Release button after disruptor use	Energy bar refills; green LED brightens; red LED flashes until 6000 mJ reached	Pass
12	Game completion (win)	Collect all waypoints within time limit	Win screen with final score (elapsed ms)	Pass
13	Timeout game over	Let timer expire without collecting all waypoints	Game over screen with "Time's up!" message	Pass

#	Test Name	Procedure	Expected Result	Status
14	Excessive tilt game over	Rotate board past ~90 degrees on either axis (AngleGain=500 gives 45000 stored mdeg limit)	Game over screen with "Excessive tilt!" message	Pass
15	HUD updates during play	Play game normally	Time bar shrinks, energy bar changes, WP counter increments	Pass
16	Tilt indicator	Tilt board in various directions	Red dot in bubble-level indicator tracks tilt direction and magnitude	Pass
17	Config validation - PinAtCenter	Set PinAtCenter=MAZE in config	Game refuses to start, displays "PinAtCenter=MAZE not supported!"	Pass
18	LED1 (green) brightness	Observe green LED during play	Brightness proportional to energy level via software PWM	Pass
19	LED2 (red) flash rate	Observe red LED while energy < 6000 mJ	Flash period = 1/10 of recharge time remaining; turns off when charged	Pass

Project Status Summary

The game is fully playable and meets all required MVP deliverables. The drone moves under gravity physics driven by the gyroscope, collides inelastically with randomly-generated walls, falls into holes, and must reach three randomized waypoints in order within 30 seconds. The quantum disruptor is fully functional: it activates only when energy ≥ 6000 mJ, drains at 10 W, bypasses all wall and hole collisions while active, and recharges at 1 W when inactive. Both LEDs operate correctly. Green brightness tracks energy via software PWM and red flash rate tracks recharge progress. A HUD shows time remaining, energy level, and waypoint progress. A bubble-level tilt indicator between the maze grid and HUD shows the current tilt direction and magnitude in real time. Win/lose screens display with appropriate messages and scores. Config validation halts the game with an explanation if unsupported configuration values are detected.

The maze is randomly generated each run using a gyro thermal-noise PRNG seed, so no two games are identical. Waypoint positions are also randomized. The only notable MVP item not fully implemented is the "quantum-tunnel forward to clear a wall" push-out behavior when the disruptor deactivates mid-wall. When that occurs, the regular wall-collision code resolves the overlap on the next physics tick, which is adequate for gameplay but not strictly spec-compliant for that edge case.

Summary Effort & Estimates

I have completed 89% of my estimated work (40 hours estimated for the work completed thus far, divided by 45 hr total estimate) in 83% of the budgeted total-project time (37.5 hr time spent, of 45 hr total estimate). For the work that has been completed, I took 0.94x (37.5/40) as much time as I estimated.

Single Work Item Estimate Evaluation

The item with the largest discrepancy was **Physics Engine - acceleration/velocity/position**, estimated at 1 hour but taking 2.5 hours (2.5x over).

The reason was an integer precision failure that was not obvious at first. The initial implementation tracked position and velocity in millimeters as integers. At small tilt angles, gravity-induced acceleration per 20 ms tick is on the order of 0.1-0.3 mm, which truncates to zero in integer arithmetic and leaves the drone completely stationary. The bug presented as "physics not working" rather than "precision too low," because tilting the board firmly produced zero motion with no error. The fix required redesigning all position and velocity storage to use micrometers (mm x 1000), which recovered three decimal places of precision and made the physics immediately responsive. This pattern of an integer truncation masking itself as a logic failure cost a full extra hour of diagnosis time and is a common pitfall in fixed-point embedded systems that I did not anticipate when writing the estimate.

Analysis of Solution

RT Task Analysis

Four FreeRTOS tasks run concurrently with the following design parameters:

Task	Priority	Period	Stack	Role
GyroTask	High (osPriorityHigh)	10 ms (100 Hz)	512 B	SPI gyro read + angle integration
PhysicsTask	AboveNormal	20 ms (50 Hz)	1024 B	Physics update, wall/hole checks, disruptor
LCDTask	Normal	30 ms (~33 Hz)	4096 B	Incremental LCD redraw, tilt indicator, HUD
LEDTask	BelowNormal	10 ms (PWM cycle)	512 B	Green LED PWM + red LED flash rate

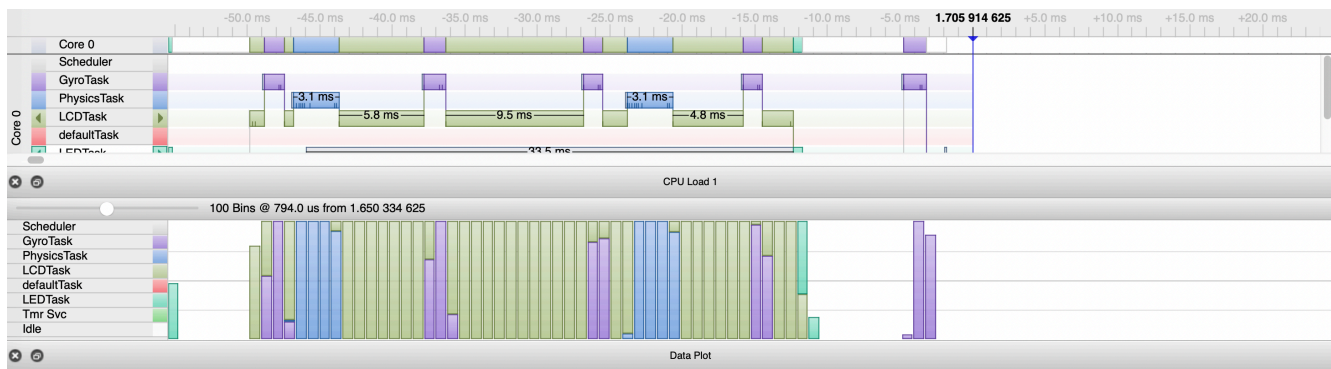


Figure 2: SEGGER SystemView trace of the four tasks

Execution time estimates:

- **GyroTask:** approximately 0.5-1 ms per tick (two SPI register reads at 1 MHz plus three integer multiplies for angle integration). Well within the 10 ms deadline with approximately 9 ms slack.
- **PhysicsTask:** approximately 0.5-1 ms per tick (sin lookup, two multiplies, velocity clamp, 3x3 wall neighborhood check of approximately 18 wall tests, hole scan). Well within the 20 ms deadline.
- **LCDTask:** most variable. A full frame erases the old ball, restores nearby walls/holes/waypoints (approximately 10-20 pixel operations), draws the ball, updates the tilt indicator (two circle fills plus 7 line segments), and every 5th frame updates the HUD (approximately 3 bar clears plus redraws). Estimated 5-15 ms per frame against a 30 ms budget; the worst case with HUD update and many nearby walls to restore fits comfortably.
- **LEDTask:** less than 0.1 ms per tick (two GPIO writes and integer arithmetic); runs at the lowest priority and has essentially unlimited slack.

Inter-task coordination: GyroTask sends to PhysicsTask via an 8-slot AngleMsg_t message queue (non-blocking put, drain-to-latest get). PhysicsTask and LCDTask share the Game_t struct via gameStateMutex. The mutex critical sections are short. LCDTask holds it for a single struct copy (approximately microseconds) and PhysicsTask holds it for one physics update cycle (approximately 500 us). No priority inversion was observed; CMSIS-RTOS2 mutexes use priority inheritance by default.

Conflicts observed: The main conflict discovered during testing was the disruptor false-activation bug. osEventFlagsWait with timeout=0 returns osFlagsErrorResource = 0xFFFFFFFFD when no flags are pending,

and `0xFFFFFFFF & BTN_PRESSED_FLAG(0x01) = 1` caused `Game_DisruptorActivate` to be called every physics tick. Fixed by guarding the flag check with `!(flags & 0x80000000U)` to reject CMSIS-RTOS2 error codes before processing.

Code Space Analysis

text	data	bss	dec
60152	112	187416	247680

- **Flash usage:** text + data = 60,264 bytes of 2,097,152 bytes (2 MB), which is **2.9% of flash used**. The majority of text is the LCD driver and FreeRTOS kernel; game logic (`GameState.c`, `PhysicsEngine.c`, `ApplicationCode.c`) is a small fraction.
- **RAM usage:** data + bss = 187,528 bytes of 262,144 bytes (256 KB), which is **71.5% of RAM used**. The BSS section (185,880 bytes) is dominated by two large allocations: the LCD framebuffer at $240 \times 320 \times 2 = 153,600$ bytes (RGB565, direct-mapped to the LTDC peripheral), and the FreeRTOS heap at **24,576 bytes** (configured in `FreeRTOSConfig.h`), which contains all task stacks and RTOS objects. The `Game_t` struct itself is approximately 1,500 bytes (walls, cells, config, drone state). The remaining approximately 6,000 bytes of BSS are HAL driver state, linker-reserved heap/stack region, and other globals.

The STM32F429I-DISC1 also has 64 KB of CCM SRAM and 8 MB of external SDRAM connected via FMC. The LCD framebuffer could be moved to external SDRAM to recover 153,600 bytes of internal RAM, reducing RAM usage from 71.5% to approximately 13%, at the cost of slightly slower pixel writes over the external bus versus AHB.

Flash is extremely underutilized at 2.9%. There is ample room for additional features such as a solvable maze generator, a score leaderboard, or a touch-screen config menu.

Physics Update Approach

Design: The physics engine uses Riemann-sum (forward Euler) integration at 50 Hz. The gyroscope measures angular rate in degrees/second; each tick integrates: $\text{angle} += \text{rate} \times \text{dt}$. The tilt angle is then used to compute gravitational acceleration on the drone: $a_x = g \times \sin(\text{angle}_y / \text{AngleGain})$ and $a_y = g \times \sin(\text{angle}_x / \text{AngleGain})$, where $g = 980 \text{ cm/s}^2$, approximated with a 90-entry integer sine lookup table (0-90 degrees, scaled $\times 1000$). Velocity and position are accumulated via $v += a \times \text{dt}$ and $x += v \times \text{dt}$. All values are stored in micrometers to preserve precision through integer arithmetic.

Rationale for grouping: Angle integration runs in a separate `GyroTask` at 100 Hz to keep it time-accurate. Position and velocity update runs in `PhysicsTask` at 50 Hz since the LCD only refreshes at approximately 33 Hz anyway. Wall collision is checked inside `Game_Update` (called from `PhysicsTask`) after the position update, so the position is always consistent when collisions are resolved with no race between movement and collision. Hole and out-of-bounds checks run in the same `Game_Update` call for the same reason.

Wall collision design: Rather than checking only the cell the drone occupies, the code checks a 3x3 neighborhood of wall segments (rows $[\text{row}, \text{row}+1]$ x cols $[\text{col}-1, \text{col}+1]$ for horizontal walls; rows $[\text{row}-1, \text{row}+1]$ x cols $[\text{col}, \text{col}+1]$ for vertical walls). This is necessary because the drone's radius (5 mm) can overlap a wall in an adjacent cell without the drone center being in that cell. Velocity direction is used to select the correct push direction, which prevents the drone from being pushed through the wall it just collided with.

Limitations not obvious at first:

1. **Integer truncation at small angles** (described above in the estimate evaluation). Required switching all positions/velocities from mm to μm .
2. **Drone teleportation through walls** before damping was applied. Without the 10%/tick friction and 20 mm/s velocity cap, the drone accelerated indefinitely and moved many cell-widths per physics tick, passing entirely through walls between checks. Damping was essential to make collision detection reliable, not just cosmetically pleasing.
3. **Ball eraser artifact.** Erasing the drone's old pixel position by drawing a white circle also erased holes, wall segments, and waypoint markers underneath. A post-erase restore pass was required that checks each static element against the erased area, essentially performing a partial redraw of the maze every frame.

Configuration Data Analysis

AngleGain = 500: This proved to be the most gameplay-critical parameter. At AngleGain = 1000, a 90 degree board rotation produces a 90 degree simulated maze tilt and the drone accelerates violently and is nearly uncontrollable. At 500, a 90 degree board rotation produces 45 degrees of simulated tilt, which is still fast but manageable. Values below 300 felt sluggish. The drone barely moved with typical tilt angles. 500 is a comfortable sweet spot.

ObstacleProbability.Wall = 100/1000 (10%): At 200/1000, the maze was so dense with walls that the drone was frequently trapped without the disruptor, making gameplay frustrating. At 100/1000, walls create interesting obstacles while still leaving navigable paths. The original spec default of 100 proved well-calibrated.

ObstacleProbability.Hole = 50/1000 (5%): The default was originally 200/1000, which made the board nearly unplayable because holes were everywhere and the drone fell in within seconds of the disruptor running out. Reducing to 50/1000 created a genuine but manageable threat level.

TimeToComplete_ms = 30000: Thirty seconds is tight for collecting 3 randomized waypoints across a 180 mm x 180 mm maze. For first-time players, 45-60 seconds would be more appropriate; 30 seconds suits experienced play.

Waypoints.Diameter_mm = 14: Reduced from the spec default of 90 mm. At 90 mm, waypoints were trivially easy to hit and the drone barely needed to be aimed. At 14 mm, the player must deliberately navigate to each waypoint, which is the intended mechanic. 10 mm would require pixel-perfect control and becomes frustrating.

Configuration items found to be unnecessary: - `Waypoints.Location`: since all waypoint positions are randomized at `Game_Init` anyway, the hardcoded defaults in the config have no effect in normal gameplay. - `Disruptor.MaxTime_ms`: at 10,000 mW drain and 15,000 mJ max energy, the battery runs out in 1.5 seconds, well before the 1,000 ms `MaxTime_ms` cap is ever reached. The energy limit is the binding constraint, making `MaxTime_ms` redundant at the default values.

What Would I Do With 2 More Weeks?

1. Solvable maze generation (highest priority). The current random maze has no guarantee that any waypoint is reachable without the disruptor. A flood-fill pass after generation, checking that a path exists from the drone's starting cell to each waypoint cell without crossing walls, would ensure the game is always beatable by skill alone. The spec notes this as a bonus item, but it has a direct impact on playability because an unsolvable maze makes the game feel unfair rather than challenging.

2. Hardware TRNG seed. The current PRNG uses gyro thermal noise as entropy, which works but is slower (8 reads x 3 ms = 24 ms at startup) and less random than a hardware source. The STM32F429 has a hardware RNG peripheral, but it requires PLL48CK to be configured in `SystemClock_Config`. Two weeks would allow time to rework the clock tree to enable PLL48CK alongside the existing 168 MHz system clock, replacing the gyro-noise seed with a true 32-bit hardware random value per game.

3. Touch-screen configuration menu. The STM32F429I has a resistive touch controller on the display. A simple pre-game menu would allow adjusting AngleGain, TimeToComplete_ms, and obstacle probabilities before each run, making the game self-demonstrating on lab day without recompiling. The instructions say this is a bonus feature that "wows the customer."

4. Disruptor wall push-out. When the disruptor deactivates while the drone is partially inside a wall, the spec requires the drone to quantum-tunnel instantaneously forward on its current trajectory until it is no longer intersecting with any walls. The current code relies on the next wall-collision tick to push the drone out, which works in practice but is not spec-compliant. A proper implementation would advance the drone position along its velocity vector in `Game_DisruptorDeactivate` until all wall intersections are resolved.

5. Per-game score display and retry. Currently the win/lose screen halts the program and the board must be reset to play again. Adding a "Press button to restart" handler that re-runs `Game_Init` with a new seed, and optionally displays a high-score list stored in the STM32's internal flash (last 4 KB of the 2 MB flash bank), would make the game much more engaging for multiple consecutive runs.

Work Item Table (Final Update)

Work Item	Status	Estimate	Time Spent	Required?
Read and digest project specification	Complete	2 hr	1.5 hr	Yes
Design task diagram	Complete	1.5 hr	1 hr	Yes
Identify unit tests and cuts	Complete	1 hr	0.5 hr	Yes
Week 1 report and risk register	Complete	1 hr	0.5 hr	Yes
Config data structure	Complete	2 hr	1.5 hr	Yes
Gyro driver - add X-axis reading	Complete	0.5 hr	0.5 hr	Yes
Physics engine - acceleration/velocity/position	Complete	1 hr	2.5 hr	Yes
Button input handling (ISR + event flags)	Complete	0.5 hr	1 hr	Yes
Game state machine (init/play/win/lose)	Complete	3 hr	3 hr	Yes
Physics engine - hole collision detection	Complete	1 hr	0 hr	Yes
Week 2 report	Complete	1 hr	1 hr	Yes
Random maze generation (walls, holes, RNG)	Complete	2 hr	1.5 hr	Yes
LCD rendering - maze, drone, waypoints	Complete	3 hr	3.5 hr	Yes
Green LED PWM (energy level)	Complete	1 hr	1 hr	Yes
Red LED flash (recharge status)	Complete	1 hr	0.5 hr	Yes
Boundary condition handling (HardEdged)	Complete	1 hr	1 hr	Yes
Waypoint tracking and sequencing	Complete	2 hr	1.5 hr	Yes
Disruptor logic (energy store, drain, recharge)	Complete	1 hr	1 hr	Yes
Game completion screen and scoring	Complete	2 hr	1.5 hr	Yes
Week 3 report	Complete	1 hr	0.5 hr	Yes
Physics engine - wall collision (inelastic)	Complete	1 hr	2 hr	Yes
LCD rendering - HUD (timer, energy, score)	Complete	2 hr	1.5 hr	Yes
Config validation (version, PinAtCenter)	Complete	0.5 hr	0.5 hr	Yes
LCD ball trail fix (redraw nearby elements)	Complete	1 hr	1 hr	Yes

Work Item	Status	Estimate	Time Spent	Required?
Stack overflow / NULL pointer crash fixes	Complete	0 hr	1.5 hr	Yes
Week 4 report	Complete	1 hr	0.5 hr	Yes
Wall system redesign (h_walls/v_walls arrays)	Complete	1 hr	1 hr	Yes
Excessive tilt game-over condition	Complete	2 hr	1 hr	Yes
LCD tilt direction indicator (bubble level)	Complete	1 hr	0.5 hr	Yes
Disruptor false-activation bug fix	Complete	0 hr	0.5 hr	Yes
Functional testing and debugging	Complete	10 hr	8 hr	Yes
Final week report	Complete	1 hr	1.5 hr	Yes
Disruptor wall push-out on deactivate	Not Complete	1 hr	0 hr	No
Hardware TRNG for maze generation	Not Complete	1 hr	0 hr	No
Touch-screen config menu	Not Complete	3 hr	0 hr	No

Risk Register (Final)

See [risks.md](#)

Risk categories: Green: < 100 risk score Yellow: > 100 and < 1000 Red: > 1000

Item	Prob	Impact	Risk	Recognized	Mitigated/Resolved	ROAM	How
Unfamiliar with integer-only physics math	40	30	1200	2026/03/24	2026/03/31	R	Implemented sin lookup table and integer unit conversions
Time management, many features in limited weeks	30	50	1500	2026/03/24		A	Prioritized MVP features; followed spec's suggested order

Item	Prob	Impact	Risk	Recognized	Mitigated/Resolved	ROAM	How
Integer precision causing zero movement	50	40	2000	2026/04/07	2026/04/07	R	Switched all position/velocity units to micrometers
FreeRTOS 60 heap/stack overflow causing hard faults		50	3000	2026/04/07	2026/04/07	R	Increased heap to 24 KB; tuned per-task stack sizes
LCD frame-buffer out-of-bounds writes causing freeze	40	50	2000	2026/04/07	2026/04/07	R	Added bounds checks to LCD_Draw_Pixel
Stack overflow from large local structs in tasks	50	50	2500	2026/04/14	2026/04/14	R	Replaced Game_t copy on LCD stack with scalar params; increased LCD stack to 4 KB
NULL pointer dereference in LCD font rendering	30	50	1500	2026/04/14	2026/04/14	R	Added LCD_SetFont call at LCD-Task startup

Item	Prob	Impact	Risk	Recognized	Mitigated/Resolved	ROAM	How
Drone too fast for wall collision detection	40	40	1600	2026/04/14	2026/04/14	R	Restored 10% friction damping and 20 mm/s velocity cap
CMSIS-RTOS2 error code matching button event flag	30	40	1200	2026/04/19	2026/04/19	R	Added !(flags & 0x80000000U) guard before flag processing
Hardware RNG unavailable (PLL48CK not configured)	50	20	1000	2026/04/14	2026/04/14	R	Replaced with gyro thermal noise PRNG seed