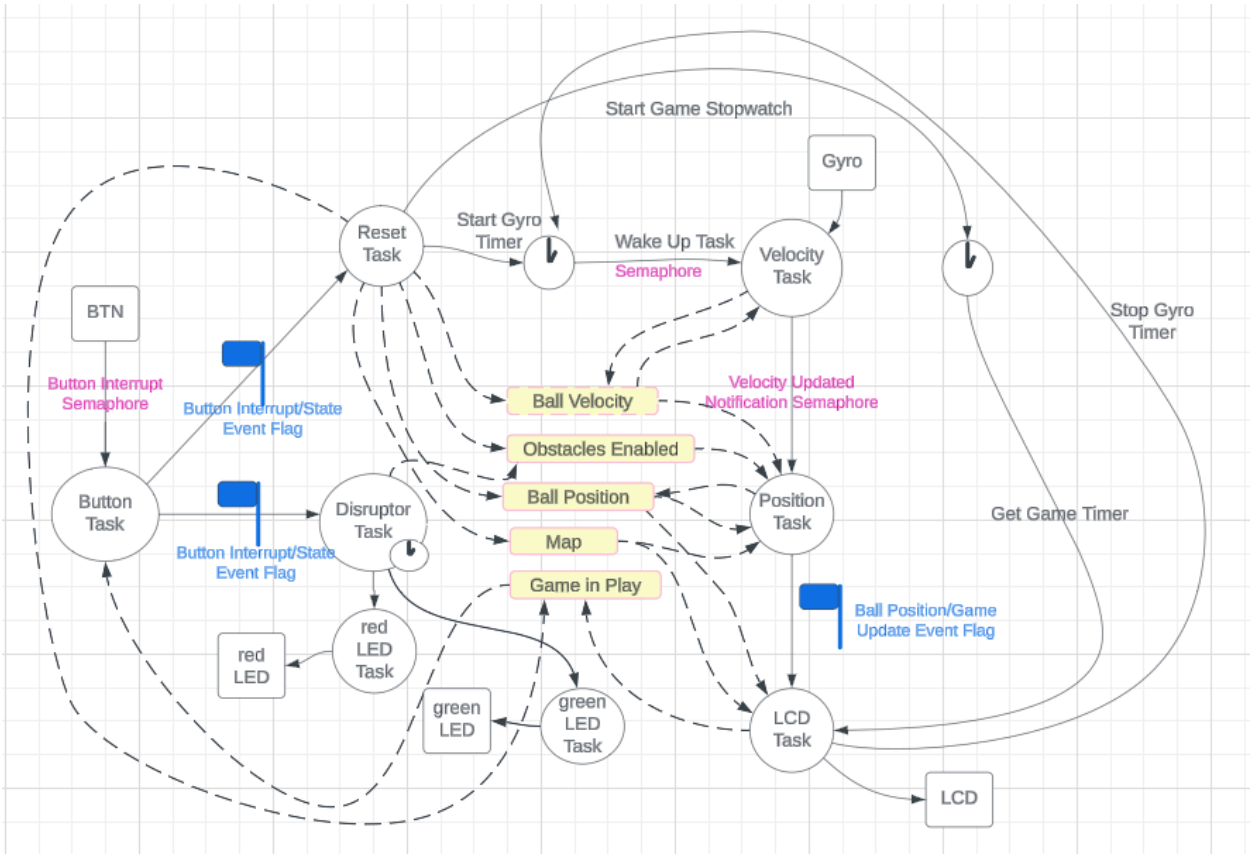

Figure 1: Task diagram. Note that dashed arrows represent mutexes.

Tasks

1. Task to drive green LED
    a. Does: Drives the LED depending on the energy status
    b. Needs: Current disruptor energy level
    c. Waits: Update frequency timer
    d. How: Green uses energy level % of max as pwm.
    e. Sends: Nothing
2. Task to drive red LED
    a. Does: Drives the LED depending on the energy status
    b. Needs: Current disruptor energy level
    c. Waits: Update frequency timer
    d. How: Red does time left to recharge, which should be (min_activation-current)/recharge_rate bounded at 0
    e. Sends: Nothing
3. Task to deal with button input
    a. Does: Determines what a button press does depending on if the game is currently in play.
    b. Needs: Currently playing
    c. Waits: Button ISR semaphore
    d. Sends: Event flag to either disruptor or reset task and disruptor (in latter case, to let disruptor know to reset its local datum energy)

    e. How: Sends to one event flag group and each task waits for only 1 flag
4. Disruptor task
    a. Does: Keeps track of current disruptor energy
    b. Needs: Initial disruptor energy, depletion rate, max time, recharge rate, button status
    c. Waits: One time tick OR button task event flag (implemented using event flag timeout)
    d. Sends: Obstacles enabled or not to position task via mutex (async), current energy level to LED task via message queue
    e. How: If button not pressed use timer to increment energy at required rate, if button pressed use timer to decrement, and also check conditions
5. Task to calculate velocity (physics!)
    a. Does: Calculates current angle of board and from there velocity
    b. Needs: Data from gyro, initial angles and velocities
    c. Waits: Timer (100 ms)
    d. Sends: Current x and y velocity to position task via mutex (and then notifies with semaphore)
    e. How: Math. Also, check each iteration if the timer is running. If not, reset angles to 0.
6. Position task
    a. Does: Update ball's position and monitors interaction with map
    b. Needs: Map knowledge (waypoints, holes, walls), velocity from velocity task, obstacles enabled from disruptor task
    c. Waits: Velocity semaphore
    d. Sends: Updated ball position to LCD task via mutex, and then notifies of this update or game win/loss with event flag
    e. How: Calculates ball's next position from velocity. Checks if wall (calculation changed) or hole/edge (game loss) or waypoint
7. Reset Task
    a. Does: Generates map and begins game. Starts gyro timer and game timer.
    b. Needs: Access to data
    c. Waits: Button task event flag
    d. Sends: Updated data. Position, velocity, obstacles enabled, map all through mutex to global variables
    e. How: Randomly make a map and update data structure for walls and holes and waypoints. Set angles and velocities to 0 (this assumes the board starts flat), enable obstacles, set currently playing to True, start timers
8. LCD Task
    a. Does: Displays map with ball or game end screen
    b. Needs: Map, ball, game end stats (win/loss/waypoints/time)
    c. Waits: Position task event flag
    d. Sends: Currently playing to button task via mutex
    e. How: Displays board most of the time. Uses a stopwatch timer to time game. At game end shows stats instead and tells the user to press the button to play again. Also stops the gyro timer so that other tasks do not run and just wait. Then just waits.

Global Variables (and which tasks will need to access)

1. Map (5, 6, 7)
    a. Holes
    b. Walls
    c. Waypoints
    d. Edge (where ball can fall off)
2. Ball position (x, y) (5, 6, 7)
3. Ball velocity (x,y) (4, 5, 6)
4. Obstacles enabled bool (3, 5, 6)
5. Game in play bool (2, 6, 7)

    *Note that disruptor energy level and x/y angles can probably be local to their respective tasks.
    *The angles, game time, and energy have also been made global variables.

Timers

1. Disruptor timer to adjust energy level at rate specified by configuration (1-shot or periodic)
2. Gyro timer to run gyro task every 100 ms (1-shot)
3. Game stopwatch to time game

ITC

1. 3 semaphores
    a. 1 from button ISR to button task
    b. 1 from velocity task to position task
    c. 1 from velocity timer callback to velocity task
2. 2 event flag groups
    a. 1 from button task to disruptor task and reset task
    b. 1 from position task to LCD task