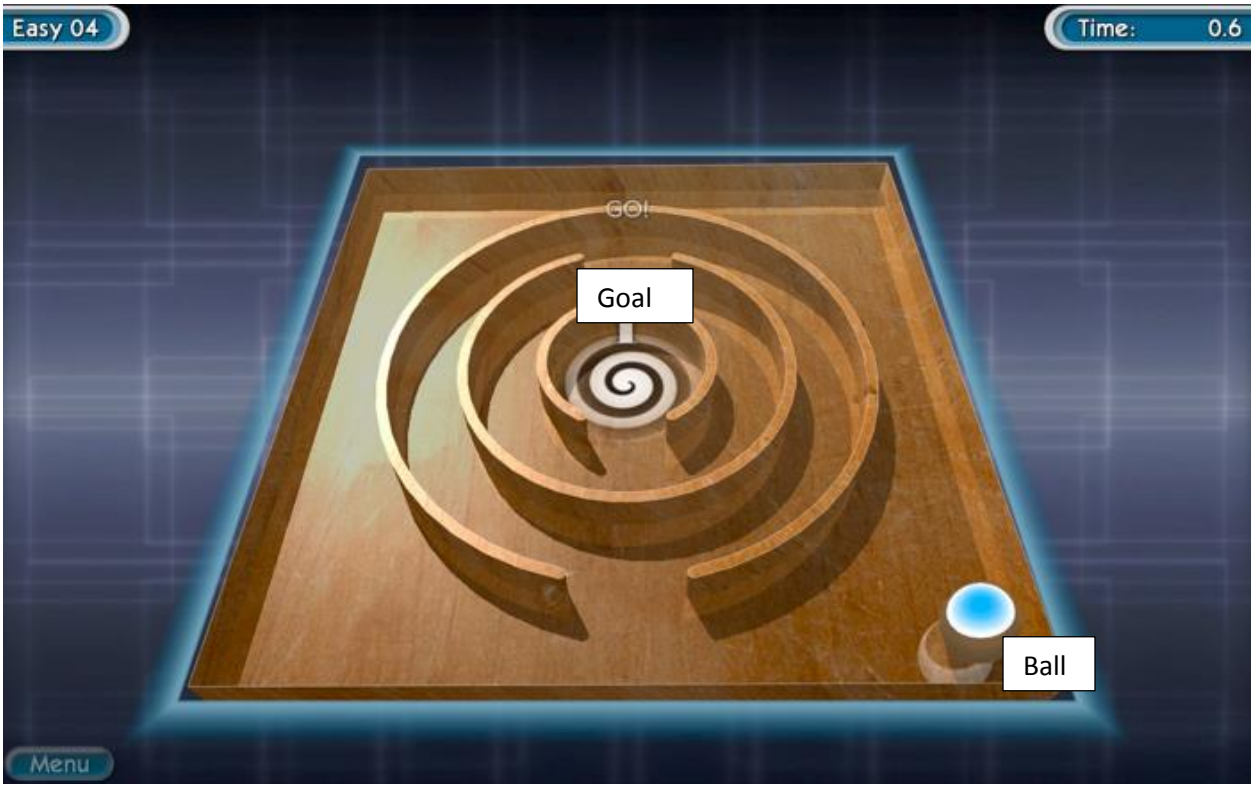
**CSE438 – Project 4**

Our task in Project 4 is to implement a simple game on the Galileo using the 8x8 LED Grid and the MPU-6050 accelerometer and gyroscope sensor module. We will also analyze the real-time behavior of the game.

The game we will be implementing is a simple simulation of a ball balance game:




In this type of game, you tilt the board in different directions in order to cause the ball to roll toward the goal, usually around walls and obstacles. In most versions, there is an outside wall that prevents the ball from rolling off the edge of the board.

In our game, the LED grid will be the "board". A single lit LED will represent the ball. We will detect tilting of the board by reading from the MPU-6050 module. The game should behave as if the sensor module is connected to the back of the board. The goal of the game is to get the ball to the center of the board and balance it there for 3 seconds. If the ball rolls off of any edge of the board, you lose and the game restarts.

At the start of the game, the 4 center LEDs flash 5 times. This is the "center" of the board. The initial position of the ball on the board should be random, but not placed on the outer edge of the board, and not placed in the center 4 LEDs of the board.

As the player tilts the sensor (and the board) the ball should accelerate in the direction of tilt. This acceleration should be proportional to the degree of tilt. If the player keeps the ball within the center 4 LEDs for 3 seconds, the player wins and the game restarts.

If a player wins, a Firework-style exploding animation should be displayed (one expanding ring of LEDs is sufficient). If a player loses, a flashing X should be displayed for several seconds.

**Program Internal Structure Requirements**

1. Write a kernel module to Interface with the MPU-6050 module using I2C, and posting its data as joystick events using the Linux input subsystem.
2. In user space, interface with the 8x8 LED grid using SPI.
3. In user space, read from the joystick event and design and Implement the game as described above.

**Real-Time Analysis**

As part of your assignment, you will turn in a report containing Real-Time Analysis of your project. The report should contain the following:

1. Identify and describe a list of at least 3 tasks that your project completes in both User Space and Kernel Space.
2. For each task, measure and record the execution time.
3. Your project must update the screen at a rate of 100Hz. Determine a valid schedule for your 3 tasks. Include a drawing of your schedule. (You do not have to implement your schedule in your project).
4. Describe any conditions that could occur to cause any task to miss its deadline.

**Bonus:**

+20 Extra Credit for implementing multiple levels with increasing difficulty. As each level starts, the level number should be displayed before the goal is flashed. Increase in level should be related to increase in acceleration due to gravity. Example: Level 1 – Balance on the moon, Level 10 – Balance on Jupiter.