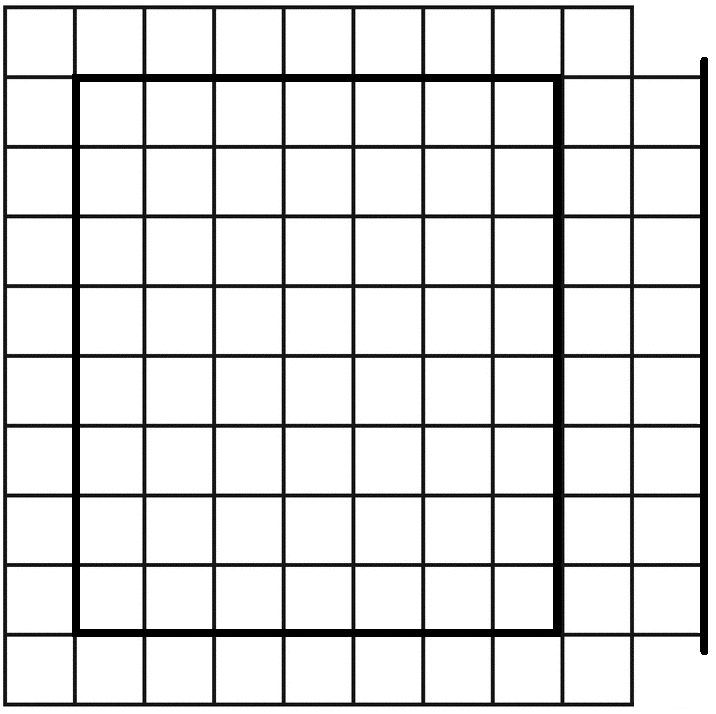
**Snake Game**

**Game Overview**

We will, by the end of February, be creating a hardware implementation of the game Snake. The game will follow the same rules as the classic game. A user will control a robot (the snake’s brain) and collect food around a game board, represented by other robots. It will differ from the classic game since we do not have unlimited robots to use, we will be using 8 robots. The game ends when the 7 robots that are food are collected, when a user hits the edge of the game board, or the user hits the snake’s tail.

**The Game Board**

The game will be played on a white board (plywood) with black lines spray painted on it. The boards dimensions will be approx. 40 inches X 40 inches and the black lines will be spaced every 4 inches. The border of the playing area will contain thicker black lines to represent the borders. To the right of the board, there will be a docking space for the inactive robots.



**Figure 1:** the game board.

Playing area is inside the thick black square. Docking area is on the right side. The robots will be docked in numerical order of their ID from top to bottom.

**Setup & Game Logic**

Before the game starts, all 8 robots will be docked in the docking space. For this document, I will be referring to each robot by a variable (R0-R7). When the user hits start on the application, the robots with ID R0 and R1 will move to starting locations on the board. The user will then have control of R0 and will navigate the board to find R1.

|  |  |
| --- | --- |
| **If the user collides with food(R1):**  The game will pause momentarily. Two actions will happen simultaneously.   1. R1 will orient itself to continue moving in the same direction R0 was moving in. 2. R2 will move onto a random location on the game board | **If the user collides with a wall:**  The game will end and all the robots will change into a cleanup state. The application will display that the user has lost. |

Assuming the game continues, the user is now in control of R1, with R0 following behind it.

|  |  |
| --- | --- |
| **If the user collides with food(R2):**  Follow the same steps as collision with food above (substituting R2 for R1 and R3 for R2). | **If the user collides with a wall or tail:**  Follow same steps as collision with wall above. |

These steps will continue until the game is over or the user wins by colliding with R7 (the last food). At this point, the application will display that the user has won, and the robots will enter the cleanup state.

**Cleanup**

After the game ends, if there is food on the board, it will move up on the board avoiding the snake. It will the follow the outermost black line of the grid counterclockwise and go to its docking position. The snake then follows in a similar fashion to fill the rest of the docking space from bottom to top.

**The Robots**

We will be using Zumo Robots by Polulu equipped with both an Arduino Uno and a nodeMCU microcontroller. The Zumo robots are good for this project since the come with an onboard accelerometer, gyroscope, magnetometer, and IR floor sensors. We are equipping it with an Arduino so we can program the robots to adhere to the game logic. The nodeMCU will connect to the Arduino to give it WiFi connectivity. We need the WiFi so all the robots can be connected to a server that can enable communication(will be discussed later) between the robots. Polulu has how-to’s and drivers for the Zumo robot’s various sensors on their website that we will need to learn to use(<https://www.pololu.com/docs/0J57/all#5>). One sensor we will need to add to the robots is some sort of proximity based sensor to detect collision with other robots.

**Communication**

The nodeMCU’s will be communicating with each other and the android app over WiFi using the MQTT protocol. MQTT works on a publish and subscribe model. Basically, a device can publish a message (a string) about a certain topic to the server. The server checks to see what devices are subscribed to that topic. Any device subscribed to the topic will then receive that message. The nodeMCU’s and the android application will be connected to a MQTT server that I have set up at the following link (73.32.15.220). The nodeMCU will communicate with the arduino via UART to relay messages from the server. The arduino will be directly controlling the zumo bot, so all the main game logic and cleanup logic will be stored on it. An alternative solution to having the nodeMCU + arduino is to equip each zumo bot with a Raspberry Pi Zero W.