Software Overview

Year: 2023 Semester: Fall Team: 5 Project: Smart Air Hockey Table

Creation Date: September 4, 2023 Last Modified: October 9, 2023

Author: Alan A. Chung Ma Email: achungma@purdue.edu

1 Software Overview

The software functionalities of the Smart Air Hockey Table (SAHT) are meticulously crafted to offer an unparalleled and enjoyable gaming experience. For this product, all operations will be executed on an embedded system; thus, any mention of "software" actually pertains to the firmware operating on the table's microcontroller. This firmware will handle reading sensor data, determining the puck's precise position in real-time, monitoring the game's progression, producing on-the-fly graphics, and driving a grid of individually addressable RGB LEDs. An overview of the system can be seen in Appendix 2, State Machine 1. A more detailed flowchart denoting the overall flow of the system is available in Appendix 1, Flowchart 2.

The primary software challenge lies in ensuring the graphics displayed on the RGB LED grid accurately follows the game puck's movement. The system's computational speed floor is governed more by the puck's velocity than the commonly accepted human-perceivable refresh rate of 75 Hz [1]. As per several online references, an air hockey puck can reach peak speeds of around 80 mph [2][3], equating to approximately 35 m/s. Given this top speed and assuming it's achieved immediately over the table's shorter 0.8m span, with three distinct readings (start, middle, and end), the display update and data retrieval rate should exceed 87.5 Hz.

*1.1 Reading Sensor Data*

The SAHT incorporates two analog sensors: photoresistors for goal detection and a grid of digital hall effect sensors to track the game puck's position. Additionally, the hall effect sensor data is compressed externally using logic gates. The method on how this is achieved is outside the scope of this report. This approach offers the system near-instantaneous update rates when read, ensuring performance requirements are met. The signal from the photoresistors will be used to trigger game state updates, as discussed in section 1.3. The frequency at which the hall effect sensor data is read will be determined by the timer, as this will also trigger additional functions. A brief outline about the data from the hall effect sensor can be viewed in section 2.1.

*1.2 Determining the Puck Position*

The microcontroller will interpret the hall effect sensor data, as detailed in section 1.1, to determine the puck's position using the algorithm from section 2.1. This identified position will inform the graphics generation (refer to section 1.4), which will subsequently be projected onto the RGB LED grid, as described in section 1.5.

*1.3 Keeping Game State*

A game-state management system has been designed to provide a seamless monitoring of the SAHT and game progress. A comprehensive state machine denoting the processes which will be described can be viewed in Appendix 2, State Machine 1.

Upon initiating the system, a series of checks and boot-up sequences are activated. The microcontroller conducts an initialization phase where each sensor undergoes a check to ensure optimal performance. Simultaneously, the onboard grid of lights are also verified for their operational status. This rigorous startup routine ensures that the underlying electronic components are functional and ensures a consistent playing experience.

Once the system's integrity checks are cleared, the system transitions to what we call a 'game-ready' phase. Here, players are presented with a distinctive graphic on the LED grid. This serves as an intuitive indicator that the table is ready for the puck to be placed at the designated center area.

After the puck hovers on the designated area for a brief period of time, it's game-on, and the system transitions to an idle state. A background timer is initialized and is continuously at work, driving a routine to precisely calculate the puck's position. This continuous positional data is then used to refresh the LED grid's graphics, offering real-time visual feedback on puck movements.

The system is designed to respond instantaneously when a goal is scored. An interrupt is triggered, which not only updates the scoreboard but also sets the table back to its 'game-ready' state, eagerly waiting for the next round to commence. A goal graphic is displayed every time a goal is scored.

A score tracker, part of the game-state management system, logs each score. Each game will run until a player scores seven goals (this value may be changed with a dip switch on the main board). The moment a player hits their seventh score the winning graphics – which are distinctly different from the regular goal graphics – light up the table. The score tracker resets to the initial state, preparing the table for another match. A flowchart denoting the score tracker can be found in Appendix 1, Flowchart 1.

*1.4 Producing Graphics*

There will be two types of graphics which will be displayed on SAHT’s LED grid: predefined animations and dynamic graphics. The software will need to be able to handle these two seamlessly with the aid of the game-state management system introduced in section 1.3.

Five predefined animated graphics are included: graphics for each player's score, graphics for each player's victory, and a 'game-ready' animation, as elaborated in section 1.3. Due to space constraints, directly storing all animation frames in full resolution is not feasible. As a solution, a data structure has been designed for efficient storage; a breakdown is available in section 3.2.

Beyond these set animations, a dynamic graphic will be showcased during active gameplay. This graphic evolves in real-time, influenced by the puck's position and trajectory. The mechanics of this dynamic graphic are driven by a drawing algorithm, the intricacies of which are detailed in section 2.2.

*1.5 Driving a Grid of RGB LEDs*

For a comprehensive understanding of the chosen LEDs' specifications and attributes, one can refer to the datasheet provided in reference [4]. The SAHT will have these LEDs organized in a two-dimensional grid pattern under the playing surface, which allows for a systematic addressing based on rows. This design choice ensures streamlined operations, making it more straightforward to drive the LEDs according to the desired graphics.

One of the software's core functionalities is to translate the compressed graphic representations (as elaborated in section 3.2) into directives for these LEDs. Harnessing the protocol from the datasheet is crucial to guarantee the LEDs' efficient and synchronized response. Though the exact implementation details are yet to be finalized, our current trajectory involves a strategy capitalizing on parallel output. The tentative plan suggests delegating distinct row groups to individual timers. By integrating multiple timers, the system could potentially drive LED rows concurrently, optimizing visual response times and elevating the game's visual experience.

2 Description of Algorithms

*2.1 Puck position*

The algorithm derives its input from two bit arrays procured from the hall effect sensor grid: one representing the rows and the other the columns. Each entry in these arrays signifies whether the puck is detected in that specific row or column. This system is analogous to using line break sensors to trace an object across two dimensions.

To deduce the puck's exact position, the algorithm first examines the "lines" obstructed in each vector. By correlating the obstructed lines from both the row and column vectors, a two-dimensional coordinate of the puck on the air hockey table emerges. In cases where multiple lines are simultaneously blocked, indicating the puck spans multiple rows or columns, the algorithm calculates the mean of the blocked lines. This provides a central representation of the puck's position, ensuring precision in tracking its movement.

*2.2 Drawing dynamic graphics*

The dynamic graphics drawing algorithm combines the previously displayed graphic with the puck’s current position to create the new graphic for the SAHT. Consider the prior graphic as $A\_{t}$and the next graphic as $A\_{t+1}$, with the puck’s current position denoted by coordinates $<x,y>$. The algorithm operates as follows: $A\_{t+1}=max(decrement(A\_{t}), glow(<x,y>))$. In this equation, the “decrement” function reduces the intensity of the $A\_{t}$ matrix, dimming the prior graphic’s luminance. Concurrently, the “glow” function crafts a radiant halo around the puck’s present location on the matrix. By taking the maximum of the two matrices, such that the maximum value of each element between the two matrices is present in $A\_{t+1}$. The resulting graphic portrays the puck in brilliant illumination while leaving a faint trail of its recent trajectory.

3 Description of Data Structures

The SAHT leans on three data structures: the bit vectors representing the readings for the hall effect sensor grid, the storage structure of the preset animated graphics, and the encoding for the RGB LED lights.

*3.1 Hall Effect Sensor Grid*

The hall effect sensor grid, integral to the SAHT’s ability to track the air hockey table puck, employs a pair of bit arrays to adeptly capture and represent the grid’s reading in real-time. Each of these arrays corresponds to a dimension: one for the rows and another for the columns of the grid. Within these arrays, a bit value indicates whether or not the puck is detected along that specific line in the corresponding dimension. This system effectively simulates line break sensors in two dimensions, offering a precise mechanism to track the puck's movement across the table.

*3.2 Preset Animation Graphic*

A lot of options were considered when designing the protocol that will store the SAHT’s preset animations. The main finding was realizing that the team will not fully utilize all 24 bits designated for color per pixel. This insight paved the way for a more efficient storage structure for each individual pixel. The second insight was that the animations the team will create are going to be very basic and will involve a lot of repeated frames. The resulting data structure for storing animations will consist of two arrays.

The first array captures all the frames, while the second delineates the sequence in which these frames should be displayed. Each frame is structured as a two-dimensional array. Here, every element is encoded with an 8-bit value, which signifies both color and brightness. The two most significant bits represent brightness, offering the system four discernible brightness levels, with an additional 'off' state. The remaining six bits capture color nuances. When these color values are zeroed out, the LED will be put to the ‘off’ state.

*3.3 RGB LED Encoding*

Controlling the RGB LEDs is no trivial task. Each LED row is orchestrated through a bespoke protocol as laid out in its datasheet[4]. The WS2812B, which the SAHT incorporates, uses a distinctive pulse-width encoding system. This encoding aids in conveying color values with high throughput and precision, ensuring that large series of LEDs display their intended hue and intensity.

4 Sources Cited

[1] Sierra Vidaure, *How Many Frames Per Second Can We Actually See In?* Available: <https://azretina.sites.arizona.edu/node/837>

[2] Prince, *How fast does a hockey puck travel?* Available: <https://www.hockeyever.com/how-fast-does-a-hockey-puck-travel/>

[3] Matt Robbs, *How Fast Does An Air Hockey Puck Travel?* Available: <https://retroonly.com/how-fast-does-an-air-hockey-puck-travel/>

[4] Worldsemi, *WS2812B Datasheet.* Available: <https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf>

Appendix 1: Program Flowcharts

*Flowchart 1: Score Tracker*



*Flowchart 2: Overall System Flowchart*

**

Appendix 2: State Machine Diagrams

*State Machine 1: High level overview of the system*

