The RC Circuit
6.111 Final Project Proposal: First Draft

Gian Delfin, Vivian Huang, Luis Terrones-Verastegui

1. Overview

The RC Circuit is a maze-solving system that takes in a user-defined maze and quickly displays a path through the maze. The user can make a maze by laying out paper "walls" in front of the camera setup. From here, we capture an image of the maze and process it using the Nexys4 in order to determine where the walls of the map are. Next, we use the Nexys4 to run the processed image through a maze-solving algorithm in order to generate a solution to the maze. Finally, we project the maze and an image of a RC car following the solved path on top of the input maze. Ideally, we do this entire process as quickly as possible in order to approach real-time maze-solving/obstacle detection.

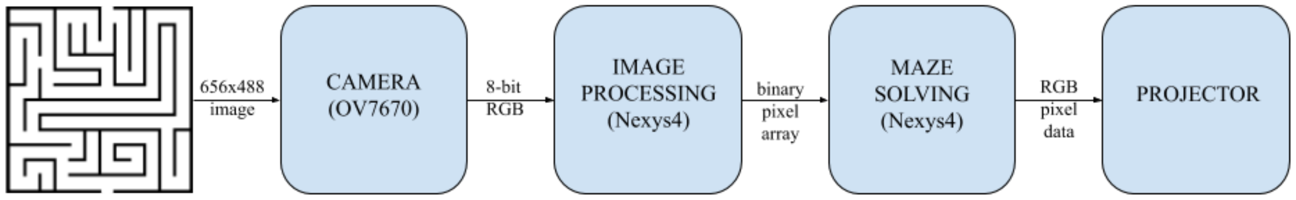
2. Goals

2.1. The minimum goal is to send an image of the maze from the camera to the FPGA, process the image, and determine the correct path through the maze using a maze-solving algorithm.

2.2. The next goal is to project onto the ground an image of a small car traversing through the correct path determined by the maze-solving algorithm.

2.3. The stretch goal is to perform real-time maze-solving as the user changes the physical walls of the maze.

3. Overall Block Diagram



4. Main Components

    4.1. Nexys 4 DDR
    4.2. OV7670 Camera Sensor
    4.3. Overhead Projector
    4.4. Tripod (to support camera and overhead projector)

5. Modules

    5.1. Camera

    The OV7670 camera sensor will capture images of the maze. The FPGA will configure the control registers of the camera via I2C protocol and receive pixel bytes serially. These pixels will be represented as a 2D array in the FPGA and will be processed in the next module.

    5.2. Image Processing

    We will perform image segmentation on the maze and transform the 2D array of pixel values into a 2D array of 1's and 0's. The 0's will represent the presence of a wall, while the 1's will represent the absence of a wall. This 2D binary array is passed to the maze solving algorithm.

    5.3. Maze Solving Algorithm

    There are several different maze solving algorithms. One common approach is to use the wall follower detailed below; this will be the first method that we will attempt. However, because the computer has knowledge of the entire maze in this case, we will also attempt to implement a shortest-path algorithm to determine the optimal path from start to finish.

    - Wall Following

      If a solution exists, this method works by moving the traveler along one wall of the maze until it reaches an exit; if the maze is unsolvable, the traveler will return to the entrance having traveled along every corridor next to that connected section of walls at least once.

- Lee Algorithm
  This is based on the breadth-first search algorithm and would operate on the 2D binary array representing the maze. The pixels would be treated as nodes in an undirected graph and there would exist edges between neighboring pixels. For example, the pixel at location (i, j) would have neighbors (i + 1, j), (i - 1, j), (i, j - 1) and (i, j + 1). This algorithm would find an optimal solution to the maze if one exists, but is more computationally expensive and may require a lot of memory. We will implement this algorithm after the Wall Following one.

6. Potential Performance and Hardware Limitations:

- The OV7670 camera has a 640x480 resolution that could limit the resolution of the maze.
- The Nexys 4 DDR FGPA's BRAM write depth is limited to 1048576, which is insufficient to support 1080p resolution (1920x1080 = 2073600), so the maze resolution is also limited by the FPGA board.
- Although popular, the Lee Algorithm is computationally expensive. For single layer maze routing, expansion is $O(d^2)$ for a connection of distance d, while cleanup is $O(N^2)$ for an NxN grid. If we use this algorithm, hardware can accelerate this process, but we will have to choose an approach which can be implemented on the FPGA.