Software Overview

Year: 2023 Semester: Spring Team: 8 Project: Engineer’s Chess

Creation Date: January 26, 2023 Last Modified: January 26, 2023

Author: Tyson Kline Email: kline62@purdue.edu

Assignment Evaluation:

| **Item** | **Score (0-5)** | **Weight** | **Points** | **Notes** |
| --- | --- | --- | --- | --- |
| **Assignment-Specific Items** | | | | |
| **Software Overview** |  | x2 |  |  |
| **Description of Algorithms** |  | x2 |  |  |
| **Description of Data Structures** |  | x2 |  |  |
| **Program Flowcharts** |  | x3 |  |  |
| **State Machine Diagrams** |  | x3 |  |  |
| **Writing-Specific Items** | | | | |
| **Spelling and Grammar** |  | x2 |  |  |
| **Formatting and Citations** |  | x1 |  |  |
| **Figures and Graphs** |  | x2 |  |  |
| **Technical Writing Style** |  | x3 |  |  |
| **Total Score** |  | | |  |

5: Excellent 4: Good 3: Acceptable 2: Poor 1: Very Poor 0: Not attempted

General Comments:

1.0 Software Overview

Engineer’s Chess will perform a multitude of functions in order to achieve the goal of a voice-controlled chess set. Many of these functions will require a large amount of firmware and software to be written in order to be accomplished.

The first major function accomplished through firmware is the game display, which will be made using a 64x64 LED matrix. The LED matrix is a separately sourced piece of hardware with its own microcontroller. Firmware will be written to control the matrix by sending commands from the main microcontroller to the microcontroller on the LED matrix through GPIO. By sending signals to each input pin of the LED matrix, we will be able to control each individual pixel of the display.

Firmware will also be necessary to control the two feedback displays. These displays will be two LCD screens controlled on the same line so that the displays are mirrored. We will accomplish this by writing code to utilize SPI to send signals to the LCD text displays.

We will also write software to control various aspects of Engineer’s Chess. The first of these is the software to identify voice commands given by the user. While we plan on using predefined libraries for the majority of this task, we will write software to interface with the rest of the project. One library we are considering using is Picovoice (1). This library has detailed documentation on working with the Jetson Nano we are using, so the integration of this software into our project should be simple.

We will also need software to control the game logic. Many other open source chess projects have been created in the past, and we have found one named PythonChess (2) that we plan on utilizing and focusing our design efforts on other portions of the project. Once again, we will need to write additional software to interface the inputs and outputs of the game logic with the rest of the project.

2.0 Description of Algorithms

We will use an algorithm to control the LED matrix display. The display functions by sending the colors for the top half of the display and the bottom half of the display at the same time. The pixels are sent one row at a time, so each row of signals must be sent quickly and repeatedly to ensure a smooth display. The algorithm we plan on using is detailed in the pseudocode below. There are eight rows of spaces on the board, and each of these spaces will take up an eight pixel by eight pixel section of the LED matrix display.

for (# of spaces / 2)

for (horizontal pixels per space)

increment row

for (# of spaces)

for (vertical pixels per space)

send top color

send bottom color

3.0 Description of Data Structures

We will be using data structures to store and modify the current game state. The first structure we will use will be square information. This structure will hold several pieces of information. First, it will store what color the background color of the square is. This color may be black or white, or any other choice between two colors that we may choose. The next piece of information that will be stored in this structure will be what piece is currently on that particular square. This could be a pawn, rook, knight, bishop, king, or queen. Finally, the data structure will store which player the piece belongs to. This structure will be used as an input to the game logic, as well as to determine what color each pixel should take on the LED screen.

Another data structure our project will utilize will be the packets sent from the speech processing computer to the main microcontroller. The data sent by the processing computer will be one of eight origin x-coordinates, one of eight origin y-coordinates, one of eight destination x-coordinates, and one of eight destination y-coordinates. This maps very neatly into four three-bit sections combined into one packet, along with a few signal bits. Our current plan for the organization of this packet can be seen below.

A picture containing text, clock

Description automatically generated

Figure 1: Packet format for move data sent from the speech processing computer to the microcontroller

4.0 Sources Cited:

1. “Picovoice platform: Nvidia Jetson Quick start,” Picovoice. [Online]. Available: https://picovoice.ai/docs/quick-start/picovoice-jetson/. [Accessed: 25-Feb-2023].
2. “Chess: A chess library for python¶,” python. [Online]. Available: https://python-chess.readthedocs.io/en/latest/index.html. [Accessed: 25-Feb-2023].

Appendix 1: Program Flowcharts

*Diagram

Description automatically generated*

Figure 1: Command Execution Flowchart

Appendix 2: State Machine Diagrams

*Diagram

Description automatically generated*

Figure 2: Microcontroller State Machine