University of Michigan
Introduction to Embedded System Design
(EECS373)

Project Final Report
Team: AI Liar's Dice

Yuchen Wu (wuyc@umich.edu)
Xiwen Wei (xiwenwei@umich.edu)
Yi Wang (umwangyi@umich.edu)

April 22, 2022

1 Customer

Those who are interested in playing dice games against artificial intelligence with real dice.

2 Value

The AI Liar's Dice provides an interactive game experience with real dice and rollers. The players can interact with the AI through visual displays and vocal announcements. Plus, AI bluffing features are added as a part of its algorithm, which can further improve gaming experiences and bring fun to the players by letting them decide whether the AI lies or not.

3 Approach

Our project is to implement an AI player in the Liar's dice game. Here's a description regarding the game rules. The goal is to let AI act as one of the players and do all the things that a player needs to do, such as rolling the dice, reading the dice faces, and making optimal bids to win against a human player.

At the beginning of each game, a high-speed spin disk driven by a DC motor and H bridge will act as a dice roller and generate a collection of five random dice faces. These dice are rolled inside a PLA cup and will not be revealed until the game ends when either the AI or the human player challenges a bid.

Inside the dice roller cup, an OV2640 camera is used to capture the image of the dice faces. The camera will transmit the image back to the microcontroller, which will further send the image back to the host PC for Open CV image processing.

After the host PC sends the recognition result back to the microcontroller, the MCU will run a probability calculation algorithm to determine the best bid to make. Then the AI will announce its decision to the player using a speaker and LCD display.

The player will then input his/her bids through a keypad interfaced to the NUCLEO board. After receiving the input, the MCU will make a new bid based on the probability calculation result.

When a challenge is made, the program inside the NUCLEO board will decide who's the winner. Then it is ready for the next round of the game.

Major functions of the project:

1. Dice rolling mechanism
   (a) 5 dice are placed on a disk and when the disk is spun by the DC motor beneath, dice will be rolled. We choose a DC motor instead of a servo because we need enough speed and power to spin and roll all the dice. The DC motor is controlled by the H bridge where the H bridge is controlled by the PWM signal sent from the NUCLEO board.
2. Dice face detection
   (a) An OV2640 camera is used to capture the dice faces. A resolution of 320x240 pixels is used to reduce the size of the image for faster transmission. The camera features a FIFO buffer and supports the SPI interfaces for data transmission. The camera can be configured using the Serial Camera Control Bus (SCCB), which is a 2-line serial protocol similar to I2C.
   (b) After the clock has been passed to the camera module and parameters such as resolution have been correctly configured, the image data will be sent to the Arduino once the slave select line for the camera module goes high. With the help of Direct Memory Access (DMA) controller embedded in the Arduino, the data is stored in the Arduino's memory and sent back to the host PC via the USART serial protocol.
   (c) After the image data is sent to the host PC, the OpenCV algorithms are run on the image to detect blobs, which are further merged into individual dice by performing density-based clustering. After the completion of dice face recognition, the results are sent back to the STM32 microcontroller using LPUART. Then the information is stored inside the NUCLEO board and decisions are made based on the dice faces.
3. Interaction with player
   (a) The 4×4 keypad communicates with the NUCLEO board through 8 GPIO pins. The player will use the keypad to input their bids for each round in the game. This is the information given to the probability calculation program for the AI to make its decision.
   (b) The display is a 2.8-inch TFT LCD. It's controlled by the NUCLEO board through SPI. This LCD uses ILI9341 as a single-chip SOC driver for a display with a resolution of 240×320. We'll use the LCD to display characters (the bids of players) and images (challenge, winner, etc.).
   (c) The speaker receives the NUCLEO board DAC output (handled by DMA). We'll store .wav files as unsigned integer array on board in advance. During the game we'll use the speaker to announce the AI's decision, the human-player's bid, and the winner.

4. Probability calculation and decision making

   (a) The algorithm of AI is based on probability calculation results. Denote the probability of exactly k dice have a particular face value in n dice as P. Then from the formula:

   $$P = \frac{n!}{k!(n-k)!}(\frac{1}{6})^{k(5/6)^{n-k}}$$

   we can derive the probability.

   (b) The AI will first check whether the player is bluffing or not. If the probability of the player's bid being true is lower than a threshold value, the AI will challenge it. Otherwise, the AI will compare a series of probabilities and make its bid following the game rules.

4 Components

1. Processors
   (a) STM32 NUCLEO L4R5ZI-P board (lab stock)
   (b) Arduino Uno board (ordered)
2. Dice rolling mechanism
   (a) Spin disk (3D printed)
   (b) Dice cup (3D printed)
   (c) DC motor and H bridge (lab stock)
3. Image processing
   (a) OV2640 camera module (ordered)
4. Player Interaction
   (a) Keypad (lab stock)
   (b) Speaker (lab stock)
   (c) LCD display: 2.8 inch SPI Module ILI9341 (ordered)
5. Miscellaneous
   (a) 9V Batteries (ordered)
   (b) LED Light Strips (ordered)

5 Claimed Difficulty

Component	Complexity	Final status
Numeric Keypad	2/3	completed
DC Motors with H-Bridge	2/3	completed
Graphics display	2/3	completed
Camera Module	2/3	completed
Speaker	2/3	completed
Open CV and probability algorithms	1/3	completed

The total complexity is 3.67.

6 Team Member Contributions

6.1 Member Task Breakdown

- Xiwen Wei
  1. Implemented the control of the DC motor through PWM and H-bridge.
  2. Configured ILI9431 LCD and wrote functions to display instructions and dice images.
  3. Designed and implemented the AI's algorithm.
- Yuchen Wu
  1. Implemented the camera interfacing codes on both STM32 and Arduino Uno.
  2. Designed and 3D-printed the dice roller model.
  3. Revised and debugged the OpenCV dice face detection algorithm.
- Yi Wang
  1. Implemented the OpenCV dice face detection algorithm.
  2. Completed the reading of the keypad input.
  3. Generated audio files, implemented and optimized audio output via the speaker.

6.2 System Integration

Yi Wang and Xiwen Wei were key to integrating the components on the NUCLEO board (keypad, motor, speaker, and LCD display) by designing a finite state machine model according to the process of the game. They were also responsible for sending the OpenCV result from PC back to the board through UART.
Yuchen Wu was responsible for properly installing the camera module to the dice roller mechanism. He managed to set up the serial communication between the Arduino and host PC and debugged the OpenCV algorithm to correctly recognize dice face as well.

6.3 Weighted Member Contribution

EECS 373 Contribution Weights for AI Liar's Dice Team:

- Yi Wang: 33.3 %
- Xiwen Wei: 33.3 %
- Yuchen Wu: 33.3 %

7 Technical Challenges and Solutions

7.1 Speaker

Audio outputs are used for better interaction with the players. We first generated all the audio we wanted as 16-bit, mono, 22050 Hz .wav files. Then we transformed the sound data into 8-bit unsigned integer arrays and stored them on the board. In this way, we could use DMA to transfer the data to the DAC register for reducing the CPU load.

However, when we were testing the speaker output, there were significant noises. To solve this problem, we first found out that the 16-bit .wav files are stored as 2's complement numbers, while the DAC register stored 12-bit unsigned integers. Therefore, we changed the DAC register alignment to discard the 4 LSBs of sound data, as the original file couldn't fit in the DAC register. Then, we added all the remaining data with 2^{11} to compensate the negative values. Finally, we managed to have very clear speaker outputs.

7.2 Camera

When first proposing this project, our team originally decide to interface the camera with STM32 controllers. However, moving a large amount of contiguous data from one peripheral to the MCU and eventually to the host PC again will not be a trivial task. To be more specific, the main challenge of interfacing cameras is composed of configuring DMAs and their corresponding interrupts. To set up the DMA controllers for use, one needs to be very familiar with the complicated DMA interfaces and registers, such as the DMA streams

and DMA Channel Memory Address Registers(CMAR), and managed to enable them with correct values and proper NVIC interrupts.

There are available third-party libraries that managed to successfully interfacing those cameras using STM32. Nevertheless, most of them, if not all, are developed more than 5 years ago and they are based on depreciated libraries such as Standard Peripheral Libraries, rather than the most recent HAL libraries. More than that, these libraries may be developed based on different boards, which feature different DMA interfaces as well. For example, one of the libraries is based on STM32 F1 series, whose DMAs are relatively simple and do not have the streams and DMA multiplexers as the L4 series do.

Given the time frame and the overall progress of this project, our team has decided to switch from STM32 controllers to Arduino Uno for camera interfacing. The primary advantage of utilizing Arduino boards is that one does not needs to set up the DMAs manually. Despite our prior efforts in implementing camera interfacing in STM32, the rest of the camera interfacing task become an SPI interfacing task with the Arduino. As a result of that, with several times of debugging to correctly store serial image on host PC, we managed to receive clear image captured by the camera module.

8 Final Design

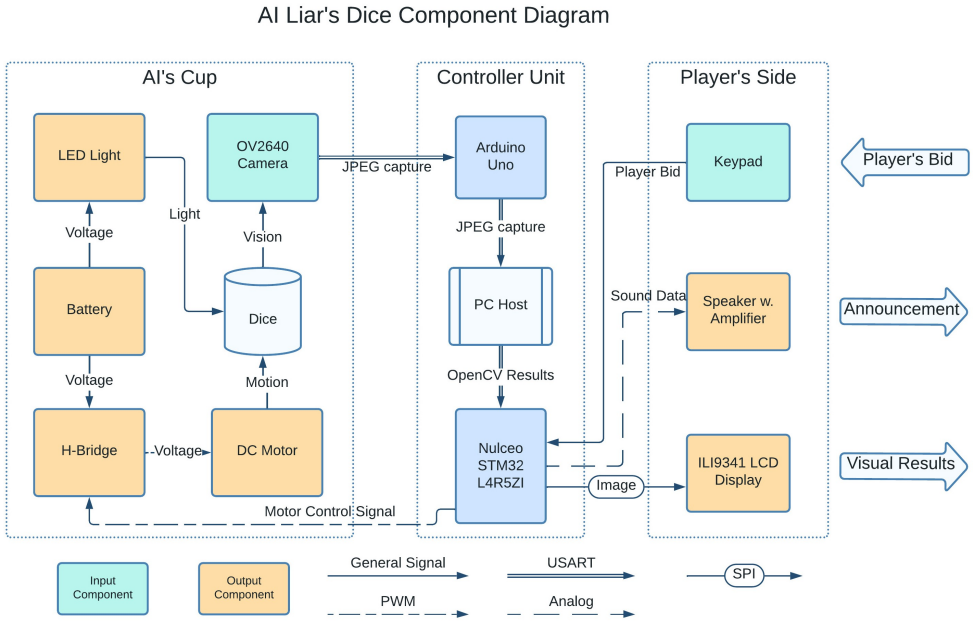

Figure 1: Component Diagram

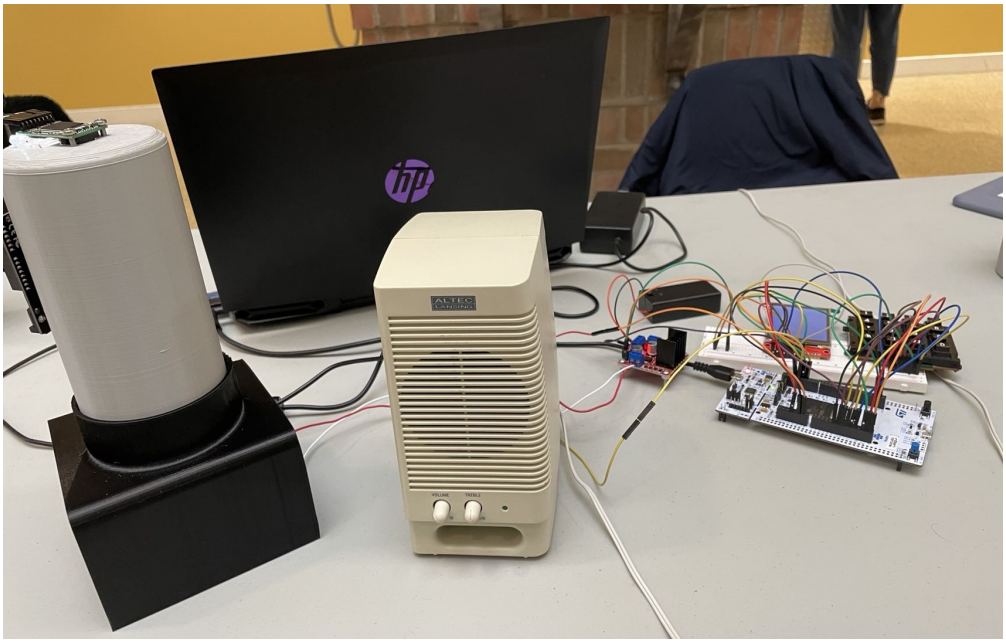

Figure 2: System Overview