Reverse-A-Bomb

1st Jacob Levinson
dept. ECE
UCLA
Los Angeles, USA
jrlevinson@g.ucla.edu

2nd Joseph Kwon
dept. ECE
UCLA
Los Angeles, USA
jkwon23@g.ucla.edu

3rd Laura Gonzalez
dept. ECE
UCLA
Los Angeles, USA
laurita412@g.ucla.edu

4th William Escobar
dept. ECE
UCLA
Los Angeles, CA
wescobar96@g.ucla.edu

***Abstract*—This project aims to recreate the game Reverse-A-Bomb from Mario Party into a real-life version. Reverse-A-Bomb is a captivating game that translates seamlessly into a physical setup. In this game, bombs move towards two players positioned on opposite sides along lanes. The players have the ability to reverse the bombs' direction by pressing buttons located in front of each lane. To achieve this, the project will utilize 2 Arduino Nano 33 IOTs integrated into wristbands for recognizing button press gestures, with data transmitted via MQTT for simplicity. OpenCV will be employed for positional tracking to determine which lane the players intend to switch. Another Arduino Nano 33 IOT will control LED strips, representing the movement of the bombs. A laptop running PyGame will handle the game logic. The integration of spatial tracking and gesture recognition will deliver a fully immersive experience, engaging players in a whole-body gaming adventure.**

***Index Terms*—Reverse-A-Bomb, MQTT, Arduino Nano 33 IOT, OpenCV**

Fig. 1. Original Revers-a-Bomb Mario Party Game

I. Introduction

Have you ever wanted to step into a game where your voice wields power, your gestures defy danger, and every move you do is an adrenaline rush? Welcome to our real-world spin on Reverse-A-Bomb, a challenging mini-game from the Mario Party video game. The game is simple: two players face each other on opposite sides of a table. There are 6 rows of "bombs", which will walk either towards or away from each player's side. To reverse the direction of a bomb, a player must press a button in front of the respective row. If a bomb reaches a player's side, they lose a life! The first player to lose 3 lives loses. Additionally, there are power ups that can be activated with voice commands to freeze bombs, slow bombs, and more!

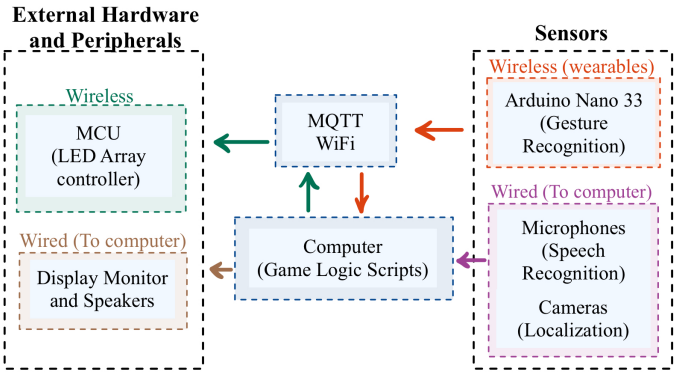

Fig. 2. Revers-a-Bomb Data Flow Diagram

II. Design

The game is comprised of 4 main components: speech recognition for power-ups, position tracking for determining which row a player wishes to reverse when a button is pressed, wireless communications for sending data between our wristbands and the game server as well as from the game server to the LED controller, and gesture recognition to determine when a player wishes to reverse a bomb.

A. Speech Recognition

Speech recognition will start and stop the game as well as notify the server when the player wants to activate the power ups. The three main power-ups that are being developed are "freeze", to freeze the bombs, "slow", to slow the bombs, and "destroy", to temporarily destroy one random bomb moving towards the player. We will use wireless microphones in order to place the microphones as close as possible to each player's mouth for optimal clarity, and Google Cloud Speech-to-Text API to transcribe the player's audio into speech. Then, the game allows the server to handle what power up will be enabled.

B. Localization

Our project's localization aspect focuses on accurately pinpointing players' positions within the game environment. This data will be crucial in conjunction with gesture recognition to identify the row a player is in when defusing a bomb. For localization, we utilize OpenCV, a computer vision library, for positional tracking. To facilitate this, each player will wear a hat in a bright, easily recognizable color like neon green.

A camera positioned above the gaming area will record footage of the players. By marking indicators at the playing field's edges, we can determine each player's relative position in the game.

C. Remote Operation

The game will be remotely operated via MQTT for the Arduino wristbands and speech recognition for starting/stopping the game and activating power-ups. This way, the players will be able to freely move around the play space and never have to physically interact with the laptop running the game.

The Arduinos on the players' wrists will connect to a WiFi network and publish any recognized gestures to a topic via MQTT. Then, the main game program running on the laptop will subscribe to this topic in order to receive all gestures associated with each respective Arduino. When the game state is updated, the laptop will publish the current desired LED state to another topic, to which the LED-controlling Arduino is the subscriber of. This central Arduino will update its own state accordingly so that the LEDs lit reflect the current game state.

D. Gestures

The Arduino Nano 33 IOTs, embedded with built-in IMUs, will be conveniently placed in wristbands that are worn by players. These Arduinos will be programmed to analyze the IMU data continuously in order to classify the data as one of a set of gestures, or none of them. Gestures will be classified with simple decision trees based on heuristics as our gestures will be very distinct from each other. When a gesture is recognized, the Arduino will send a message indicating as such to the main game server via MQTT.

III. Testing

A. Speech Recognition - Testing

Our primary focus is ensuring that the speech recognition component functions seamlessly with the voices of all players, even in noisy environments, to enhance the system's reliability. To achieve this goal, we will conduct extensive testing with various individuals under simulated background noise conditions. Following the classification results, we will fine-tune several aspects: adjusting microphone gain, tweaking API parameters, and updating our word cloud to encompass commonly mistaken words. Additionally, we will determine the optimal distance between microphones and players' mouths, along with the appropriate speaking volume for players.

Fig. 3. Arduino Nano embedded into wearable wristband

B. Localization - Testing

We will start by putting markers at specific spots on the field and recording video. Then, we will compare the positions tracked by OpenCV to the real ones to gauge accuracy in different lighting and player orientations. For dynamic movement tests, players will wear wristbands and do controlled actions like walking, running, and stopping. We will compare the tracked positions to their real locations to evaluate accuracy, smoothness, and response time. This will help us adjust parameters to better track each player's bounding box.

C. Remote Operation - Testing

Testing the remote operation can be done initially by simply reading each Arduino's IMU data over MQTT on our laptop to ensure that section is working. In order to to test the LED-controlling Arduino before the game is completed, we can send test LED patterns over MQTT to ensure that the data is being received and the LED strips have been set up correctly.

D. Gestures - Testing

We have already completed basic testing on the button slap gesture. By recording the IMU data when performing a slap, we were able to analyze this data to see the minimum peak acceleration in the Z-axis when a slap occurs over many trials. We used this to inform our splits in the decision tree algorithm and currently have a robust slap detection algorithm that is not triggered by other movements.

IV. Work Distribution

Jacob is responsible for the overall game logic and design integration.

William is responsible for the code development of the LED-controlling Arduino and the Arduinos on players' wrists, as well as creating the physical wristbands and designing the interface between the main game and the LED-controlling Arduino.

Joseph is responsible for the OpenCV positional tracking of players, tracking objects like LEDs and hats to determine their relative positions to the LED strips.

Laura is responsible for managing the speech recognition module and setting up wireless microphones. She will develop a robust code segment that recognizes keywords for power-up activation, which seamlessly integrates into the main Pygame game loop.

V. Timeline

A. Winter Week 6

All materials will be planned for and ordered. Each section of the project as outlined in the work distribution section will be underway with the materials we currently have.

B. Winter Week 8

Each section of the project will be individually functional. We will be able to receive IMU data from 2 Arduino Nano 33 IOTs and correctly classify the motion on the laptop, the position tracking will be able to give us the x and y coordinates of the center of each player, the speech recognition will be able to output when any of the key words specified have been used, and the Arduino controlling the LEDs will be able to receive MQTT data and at least change its internal state and output pins (if the LED strips have not yet arrived). The game logic will be mostly complete, just missing integration of the other components.

C. Winter Week 10

All of the systems will be integrated into the one game program. We will not focus on power ups here, rather we will have a terminal output when one would have been activated to show that the recognition at least works. The game state will be output to the terminal every second if we do not have the LED logic fully working by then as well.

D. Winter Week 11

For our final presentation of Winter, we will show the systems integrated with the game logic successfully running by showing the game state from the terminal. At this point, power ups will not be integrated.

E. Spring Week 3

The LEDs will be fully working to reflect the current game state. Also, the power ups via speech recognition will be working as well. All that is left to add is additional power ups and perhaps new gestures to indicate when the microphone should listen.

F. Spring Week 5

Our game will be fully working, albeit perhaps not with a proper start/end state. The power up availability will be indicated by the LEDs on the Arduino Nano 33 IOTs on the wrist or by some other LEDs. We will have our Midterm Presentation and add the polish that are needed for a complete game.

G. Spring Week 8

The project will be complete. We will stress test the system and polish the user experience.

H. Spring Week 10

Project demo and open house.

I. Spring Week 11

Final presentation.

VI. Conclusion

Our Reverse-A-Bomb game will be designed to be a fun and immersive experience accessible to all players, regardless of technical expertise. We will achieve this through seamless integration of various technologies: speech recognition for power-ups, MQTT for wireless communication, and OpenCV camera tracking for positional data. With these features in place, players can fully immerse themselves in the game without needing to worry about its technical intricacies.

References

[1] S. M. Wiki, "Revers-a-Bomb - Super Mario Wiki, the Mario encyclopedia," Super Mario Wiki, Jan. 08, 2024. https://www.mariowiki.com/Revers-a-Bomb

[2] Arduino, "Nano 33 IoT," Arduino Documentation. https://docs.arduino.cc/hardware/nano-33-iot/ (accessed Feb. 16, 2024).

[3] MQTT Python Client, https://github.com/eclipse/paho.mqtt.python

[4] ECE 180DA. "Lab 4: The Hardware Tutorial."