# 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 will be a recreation of the game Reverse-A-Bomb from the video game Mario Party. In this game, bombs walk towards two players on opposite sides down lanes, which the players can reverse the direction of by pressing a button in front of each lane. The project will use 2 Arduino Nano 33 IOTs for the button press gesture recognition sent over MQTT, OpenCV for positional tracking to tell which lane the players want to switch, a third Arduino Nano 33 IOT for controlling LED strips to represent the bombs, and a laptop to run the game logic via PyGame.**



***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? Well our version of the Reverse-A-Bomb game will encompass all these factors through four key components. Speech recognition, localization, remote operation, and hand gestures in order to be triumphant in our game. This proposal aims to expand exactly how each of these components will be integrated through hardware and software implementation, and how each aspect of the game will be using these components.

## II. Game Instructions

The rules to the game are you must be able to reverse bombs that are coming towards your direction and flip them in the opposite direction towards your opponent. There are a total of six rows you will be multi-tasking between before the bomb can reach your side. During the game there will be power ups you will receive in 30 second increments that can be activated in order to help you maintain the bombs going in the other direction. The player loses when five bombs blow up on your side or the timer of three minutes runs out and whoever has the most amount of bombs blown up on their side loses.

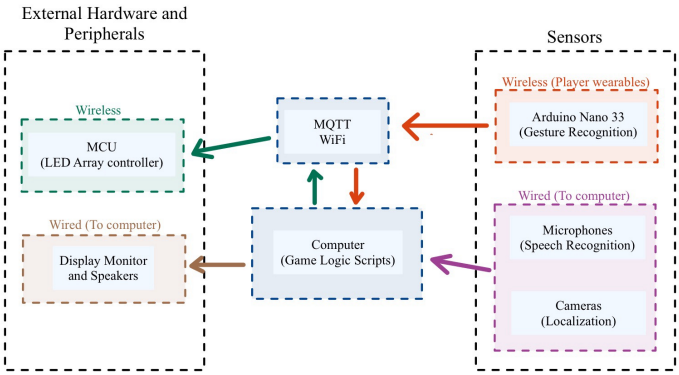


Fig. 2. Revers-a-Bomb Data Flow Diagram

### A. Speech Recognition

Speech Recognition in our game will take form in starting/stopping the game as well as letting the server know when the player wants to activate the power ups. There are two main power ups that will be available in the game. First, is freezing a row, stopping the bombs from coming towards you. The second is killing off a bomb so that the player doesn't have to worry about that row for five seconds. We'll be using our laptop's built in microphone and Google Cloud Speech-to-Text API to transcribe the player's audio into speech and then allowing the server to handle what power up should be enabled.

### B. Localization

The localization component of our project focuses on accurately determining the positions of the players within the game environment. This information is essential for coordinating the

players' actions with the virtual game elements and ensuring a seamless gaming experience. To achieve localization, we employ OpenCV, a powerful computer vision library, for positional tracking.

A camera mounted above the gaming area will capture video footage of the players. We plan to use a wide-angle camera to ensure adequate coverage of the entire playing field. Each player will wear a wristband equipped with an Arduino Nano 33 IOT and an LED strip. The Arduino will collect motion data from the built-in accelerometer and gyroscope, while the LED strip will provide visual feedback for debugging and testing purposes.

### C. Remote Operation

The game will be remotely operated via wireless technology such as MQTT for the Arduinos and speech recognition for starting/stopping the game, as well as for the power ups. In this way, the users will 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 their accelerometer and gyroscopic data to a topic via MQTT. Then, the main game program running on the laptop will subscribe to this topic in order to receive the newest data for that Arduino. When the game state is updated, the laptop will publish the current LED state desired to another topic, which the Arduino controlling the LEDs will have subscribed to. That Arduino will update its own state accordingly so that the LEDs lit reflect the current game state.

### D. Gestures

The Arduino Nano 33 MCUs, embedded with built-in IMUs, are conveniently placed in wristbands that are worn by players. These IMUs are programmed to capture sharp acceleration gestures made by players when striking their designated buttons. The code, inspired by Lab 4 in the design class, will incorporate MQTT and IoT principles to efficiently transmit IMU readings.

Fig. 3. Arduino Nano embedded into wearable wristband

In the upcoming stages of development, the gesture detection algorithm will evolve to include decision trees and time dependencies to better manage variations in strikes. Initially we will have a simple static decision algorithm that will be based off of constant conditions, i.e. same accelerations each press. Once this simple algorithm is established we can then incorporate decision trees which will provide a structured approach for classifying the sharp accelerations detected by the IMUs. The algorithm will consider factors such as the magnitude, duration, and frequency of the spikes to differentiate between different levels of button strikes, accommodating the variation in player intensity.

Moreover, time dependencies will be introduced to the algorithm, considering the evolving dynamics of the game. As the game progresses, the speed of the bombs increases, impacting the force with which players strike the buttons. The algorithm will adapt its thresholds and decision-making criteria dynamically, ensuring accuracy in gesture recognition across varying game states. This dynamic adjustment will be crucial for maintaining responsiveness and precision, particularly as the gameplay becomes more challenging.

To refine and optimize the gesture detection algorithm, confusion matrices will be employed during the training phase. These matrices will help analyze and fine-tune the algorithm's performance by providing insights into the classification accuracy for different strike intensities. The combination of decision trees, time dependencies, and confusion matrices forms a comprehensive approach to gesture detection, enhancing the overall realism and engagement of the Reverse-a-Bomb game. The algorithm's ability to adapt to changing conditions ensures a seamless and enjoyable gaming experience as players face increased speed and complexity throughout the gameplay.

## III. Testing

### A. Speech Recognition - Testing

Testing the speech recognition part of our game we will need to be able to test for accuracy of the player's words. We need to ensure accurate detection of the words to then let the MQTT server know which power up should be enabled and when the player wants to start the game or stop it. We will be testing the accuracy of Google's Speech-to-Text API with different people's voices. We plan to adjust parameters if the API keeps detecting a different word other than what the person is actually saying to ensure complete recognition of the word and no problems in our game. We will also test for noise and what is an acceptable distance that the laptop's microphone will still be able to recognize the player's speech. Along with this we must test how loud the person should be saying the words and what is considered too loud or too quite. Our main concern will be to getting the speech recognition component to work with the voice of anyone who plays our game.

Below is an example of the speech recognition working from Lab 2 in class:

### B. Localization - Testing

We will begin by placing markers or wristbands at predefined points on the playing field and capturing video footage.

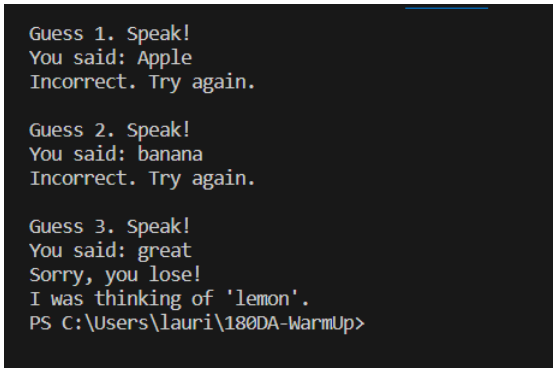


Fig. 4. Speech Recognition from Lab 2

Comparing the tracked positions reported by OpenCV with the actual positions will assess accuracy under various lighting conditions and player orientations. Dynamic movement testing will involve players wearing wristbands and performing controlled actions like walking, running, and stopping. By comparing the tracked positions with their actual locations, we can evaluate tracking accuracy, smoothness, and response time.

### 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 will use basic hand gestures such as thumbs up and thumbs down. If the test is successful, we will add more hand gestures such as numbers 1 to 5 on each hand. If the subsequent test is successful, we will add additional hand gestures for power-ups.

## IV. Work Distribution

### A. Game logic - Jacob

Jacob will be primarily responsible for the main game logic. This will include integration of the subsystems as well.

### B. LED Controller and wristband IMUs - William

William will be responsible for developing the wristbands with Arduino Nano 33 IOTs attached, as well as writing the code for and setting up the LED controlling Arduino.

### C. Localization - Joseph

Joseph will be responsible for the OpenCV positional tracking of players. This includes deciding how to track them (LEDs, hats, ect) and also how to find out where they are relative to the rows in game.

### D. Speech Recognition - Laura

Laura will be responsible for the speech recognition module, designing a robust and easily interchangeable portion of code to recognize keywords for power ups.

## V. Timeline

### A. Winter Week 6

By the end of Winter week 6, all materials should be planned for and ordered. Each section of the project as outlined in the work distribution section should be underway with the materials we currently have.

### B. Winter Week 8

By the end of Winter week 8, each section of the project should be individually functional. We should be able to receive IMU data from 2 Arduino Nano 33 IOTs and correctly classify the motion on the laptop, the position tracking should be able to give us the x and y coordinates of the center of each player, the speech recognition should be able to output when any of the key words specified have been used, and the Arduino controlling the LEDs should 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 should be mostly complete, just missing integration of the other components.

### C. Winter Week 10

By the end of 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 should 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

By Spring week 3, the LEDs should be fully working to reflect the current game state. Also, the power ups via speech recognition should 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

By Spring week 5, our game should be fully working, although perhaps not will a proper start/end state. The power up availability should also be indicated by the LEDs on the Arduino Nano 33 IOTs on the wrist or by some other LEDs. This is when our Midterm presentation is as well. We will plan to add the polish needed for a complete game.

### G. Spring Week 8

By now, the project should be complete. We should work now to stress test the system and polish the user experience.

### H. Spring Week 10

Project demo and open house.

### I. Spring Week 11

Final presentation.

## 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."