**Reaction Arduino Developer Document**

**Table of Contents**

1. Overview
2. System Requirements
3. Hardware Setup
4. Game Flow
5. Key Features
6. Code Structure
7. Module Details
8. Mathematical Details

**1. Overview**

This game challenges players to respond quickly to color changes. Players must clap when the color is green and remain silent during other colors. The game includes multiple difficulty levels and a high-score system.

**2. System Requirements**

**Hardware Requirements**

* Arduino board (e.g., Arduino Uno)
* RGB LED (Red, Green, Blue)
* Piezo buzzer
* Button
* Sound sensor (e.g., microphone module)
* Three output pins for RGB LED control
* EEPROM (built into the Arduino)

**Software Requirements**

* Arduino IDE
* EEPROM library

**3. Hardware Setup**

1. **RGB LED**:
   * Blue connected to PIN 9
   * Green connected to PIN 10
   * Red connected to PIN 11
2. **Buzzer**:
   * Connected to PIN 6
3. **Button**:
   * Connected to PIN 7, using an internal pull-up resistor (INPUT\_PULLUP)
4. **Sound Sensor**:
   * Analog output connected to A0
5. **Indicator LEDs**:
   * Three LEDs connected to PINs 2, 3, and 4

**4. Game Flow**

1. **Initialization**: Display the current level and high score.
2. **Game Start**: Press the button to start the game and enter the waiting state.
3. **Color Display**: Randomly show Red, Green, or Blue on the RGB LED.
4. **Player Reaction**:
   * Green: Player must clap.
   * Red/Blue: Player must remain silent.
5. **Rewards and Penalties**:
   * Correct reactions score points; incorrect reactions end the game.
6. **Level Up**: Automatically advance levels after reaching the required score threshold, increasing difficulty.
7. **Game Over**: Display the final score and update the high score if necessary before resetting.

**5. Key Features**

**5.1 Random Color Display**

Randomly choose and display a color (Red, Green, or Blue) using the RGB LED.

void setRandomColor() {

int color = random(0, 3);

switch(color) {

case 0: // Blue

digitalWrite(rgbPins[0], HIGH);

break;

case 1: // Green

digitalWrite(rgbPins[1], HIGH);

break;

case 2: // Red

digitalWrite(rgbPins[2], HIGH);

break;

}

}

**5.2 Clap Detection**

Detect a clap based on the output value from the sound sensor.

bool currentSoundState = analogRead(soundSensorPin) > 500;

if (currentSoundState && !previousSoundState) {

// Clap detected

}

**5.3 Scoring and Levels**

Increase score for correct reactions and advance levels upon reaching the score threshold.

if (score >= pointsPerLevel \* level) {

level++;

updateLevel();

}

**5.4 High Score Storage**

Save and retrieve the high score using EEPROM.

EEPROM.get(highScoreAddress, highScore);

EEPROM.put(highScoreAddress, highScore);

**6. Code Structure**

**Global Variables**

* **Button and LED Pins**: Define all input/output pins.
* **Game State**: Enum to track the current game state.
* **Timing Variables**: Track color display and reaction timing.
* **Score and Level**: Store the player’s score and current level.

**Functions**

* setup(): Initialize hardware and game parameters.
* loop(): Main loop to manage game logic.
* setRandomColor(): Set a random color on the RGB LED.
* giveReward(): Reward the player based on reaction time.
* gameOver(): End the game.
* resetGame(): Reset the game state.

**7. Module Details**

**7.1 Long Press Detection**

Reset the game if the button is pressed for more than 5 seconds.

if (currentTime - buttonPressStartTime >= longPressDuration) {

resetGame();

}

**7.2 Difficulty Adjustment**

Reduce reaction and display times as levels increase.

reactionTime = max(baseReactionTime - (level - 1) \* reactionTimeReductionPerLevel, minReactionTime);

**8. Mathematical Details**

**8.1 Random Color Selection**

Generate a random number using random(0, 3) to select a color:

* random(0, 3) returns integers 0 to 2, corresponding to:
  + 0: Blue
  + 1: Green
  + 2: Red

**8.2 Scoring and Level Calculation**

Determine if the player has scored enough points to level up:

* Formula:
* if (score >= pointsPerLevel \* level) {
* level++;
* }
  + pointsPerLevel: Points required per level.
  + level: Current level.

**8.3 Reaction and Display Time Adjustment**

Decrease display and reaction times with increasing levels:

* **Reaction Time Formula**:
* reactionTime = baseReactionTime - (level - 1) \* reactionTimeReductionPerLevel;
* reactionTime = max(reactionTime, minReactionTime);
  + baseReactionTime: Initial reaction time.
  + reactionTimeReductionPerLevel: Reduction per level.
  + minReactionTime: Minimum reaction time.
* **Color Display Time Formula**:
* currentColorDisplayTime = baseColorDisplayTime - (level - 1) \* colorDisplayReductionPerLevel;
* currentColorDisplayTime = max(currentColorDisplayTime, minColorDisplayTime);
  + baseColorDisplayTime: Initial color display time.
  + colorDisplayReductionPerLevel: Reduction per level.
  + minColorDisplayTime: Minimum display time.

**8.4 Clap Reaction Time**

Calculate the time difference between the displayed color and the detected clap:

* Formula:
* userReactionTime = clapTime - lastStateChange;
  + clapTime: Timestamp of the detected clap.
  + lastStateChange: Timestamp of the color display start.

**8.5 Reward Logic**

Award points based on reaction speed:

* Reward Conditions:
* if (reactionTimeUser < 1000) {
* score += 3;
* } else if (reactionTimeUser < 2000) {
* score += 2;
* } else {
* score += 1;
* }
  + Fast reaction: 3 points.
  + Medium reaction: 2 points.
  + Slow reaction: 1 point.