```
// Including the Arduino library for the LCD display
#include <LiquidCrystal.h>

// These digital ports will receive the input for each RGB color
const int red = 8;
const int green = 9;
const int blue = 10;

// Assigning port 13 to the buzzer to play sounds
const int buzzer = 13;

// These are the ports associated with the LCD
const int RS = 12;
const int E = 11;
const int DB4 = 5;
const int DB5 = 4;
const int DB6 = 3;
const int DB7 = 2;

LiquidCrystal lcd(RS, E, DB4, DB5, DB6, DB7);

// Initializing the setup for the RGB inputs, buzzer output, and LCD
void setup() {
  pinMode(red, INPUT);
  pinMode(green, INPUT);
  pinMode(blue, INPUT);
  pinMode(buzzer, OUTPUT);
  lcd.begin(16, 2);
  Serial.begin(9600);
}

// The loop checks which color we have based on the inputs we read from the RGB
void loop() {
  int redValue = digitalRead(red);
  int greenValue = digitalRead(green);
  int blueValue = digitalRead(blue);

  if (redValue && !greenValue && !blueValue) {
    lcd.print("red color");
    // Generate a tone on the buzzer
```

```
    tone(buzzer, 1000);  // Frequency of 1000 Hz

    // Wait for 1 second
    delay(1000);

    // Stop the tone
    noTone(buzzer);

    // Wait for 1 second
    delay(1000);
  }
  if (!redValue && greenValue && !blueValue) {
    lcd.print("color verde");
    playSong();
  }
  if (!redValue && !greenValue && blueValue) {
    lcd.print("color azul");
    // Play the Morse code for "BLUE"
    playMorse("BLUE", 100);
    delay(2000); // Wait for 2 seconds between each repetition
  }
  if (redValue && greenValue && !blueValue) {
    lcd.print("color amarillo");
  }
  if (!redValue && greenValue && blueValue) {
    lcd.print("color cyan");
  }
  if (redValue && !greenValue && blueValue) {
    lcd.print("color magenta");
  }
  if (redValue && greenValue && blueValue) {

    lcd.print("color blanco ");
  }
  if (!redValue && !greenValue && !blueValue) {
    lcd.print("LED apagado");
  }
  delay(100);
  lcd.clear();
}
```

```
void playTone(int pin, int frequency) {
  // Calculate the period of the tone
  long period = 1000000L / frequency;

  // Calculate half of the period
  long halfPeriod = period / 2;

  // Generate the tone
  for (int i = 0; i < 100; i++) {
    digitalWrite(pin, HIGH);
    delayMicroseconds(halfPeriod);
    digitalWrite(pin, LOW);
    delayMicroseconds(halfPeriod);
  }
}

void playMorse(const char* word, int dotDuration) {
  const int unitTime = dotDuration;
  const int dashDuration = unitTime * 3;
  const int silenceDuration = unitTime;

  for (int i = 0; i < strlen(word); i++) {
    char letter = toUpperCase(word[i]);

    switch (letter) {
      case 'A':
        playDot();
        playDash();
        break;
      case 'Z':
        playDash();
        playDot();
        playDot();
        playDot();
        break;
      case 'U':
        playDot();
        playDot();
        playDash();
```

```
        break;
      case 'L':
        playDot();
        playDash();
        playDot();
        playDot();
        break;
    }

    // Add a silence after each letter
    delay(silenceDuration);
  }
}

void playDot() {
  tone(buzzer, 1000); // Frequency of 1000 Hz
  delay(200); // Duration of a dot
  noTone(buzzer);
  delay(200); // Pause between dots
}

void playDash() {
  tone(buzzer, 1000); // Frequency of 1000 Hz
  delay(600); // Duration of a dash
  noTone(buzzer);
  delay(200); // Pause between dashes
}

void playSong() {
  // Define the notes of the song
  int notes[] = {262, 294, 330, 349, 392, 440, 494, 523};

  // Define the duration of each note
  int durations[] = {200, 200, 200, 200, 200, 200, 200, 200};

  // Play each note of the song
  for (int i = 0; i < sizeof(notes) / sizeof(notes[0]); i++) {
    int note = notes[i];
    int duration = durations[i];
```

```
    playNote(note, duration);

    // Pause between each note
    delay(100);
  }
}

void playNote(int note, int duration) {
  tone(buzzer, note);
  delay(duration);
  noTone(buzzer);
}
```