PIA PROJECT

5x5 LED Emotion Matrix

Professor: Alexandru Guzu

Student: Rasid Esra , 411G

1. **Introduction**

This code is designed to create a dynamic LED display using an ESP32 microcontroller, representing different emotional expressions with LEDs. The LEDs are connected to pins on the ESP32, and each emotional expression is associated with a set of specific pins. The emotions include a smiley face, a sad face, an amazed face, and a poker face.

The circuit features two eyes LEDs (connected to pins 17 and 16), which remain lit throughout the operation, simulating the eyes of the displayed emotion. The emotional expressions are triggered using a button connected to pin 27. When the button is pressed, the code switches to the next emotion in a sequence. There's a debounce mechanism in place to prevent accidental multiple presses of the button.

Components Used:

• ESP32 microcontroller

• LEDs (for smiley, sad, amazed, and poker faces)

• Two LEDs (for eyes)

• Button (for changing emotions)

Here's a brief overview of the key components and functionalities in the code:

**ESP32 Microcontroller**: The ESP32 is a powerful microcontroller based on the Xtensa LX6 CPU architecture. It integrates Wi-Fi and Bluetooth connectivity, making it suitable for IoT projects. In this code, the ESP32 controls the LEDs to display different emotional expressions and responds to button presses.

**LED Pins**: Light Emitting Diodes (LEDs) are semiconductor devices that emit light when a current flows through them. In this project, LEDs are used to visually represent different emotions. The specific pins connected to the LEDs determine which ones light up to form the desired expression. Pins 4, 15, 25, 26, and 0 are used to represent different segments of the smiley face. Pins 39, 13, 15, 26, and 25 represent segments of the sad face. Pins 32, 33, and 15 represent segments of the amazed face. Pins 25 and 15 represent segments of the poker face.

**Eyes Pins**: Pins 17 and 16 are dedicated to representing the eyes of the displayed emotion. They are turned on and remain lit for all emotions.

**Button**: A button connected to pin 27 serves as a trigger to switch to the next emotion when pressed.

**Emotion Sequencing**: The code cycles through the emotions in sequence (smiley, sad, amazed, and poker) each time the button is pressed.

**Debounce Mechanism**: To prevent unintended button presses due to contact bouncing, a debounce delay of 200 milliseconds is introduced. This ensures that a single button press is recognized.

*The Arduino IDE (Integrated Development Environment) is a software application used for writing, compiling, and uploading code to Arduino boards. It provides a user-friendly interface for programming Arduino microcontrollers. The IDE supports multiple programming languages, including C and C++, and offers a set of built-in functions and libraries specific to Arduino boards. It allows developers to write code, manage libraries, and communicate with Arduino hardware effectively.*

In summary, this code showcases how to use an ESP32 microcontroller to create an interactive LED display of different emotional expressions. The eyes LEDs are always on, while the other LEDs change to represent the current emotion each time the button is pressed. This can be a fun and creative project to visually convey various emotions using LEDs.

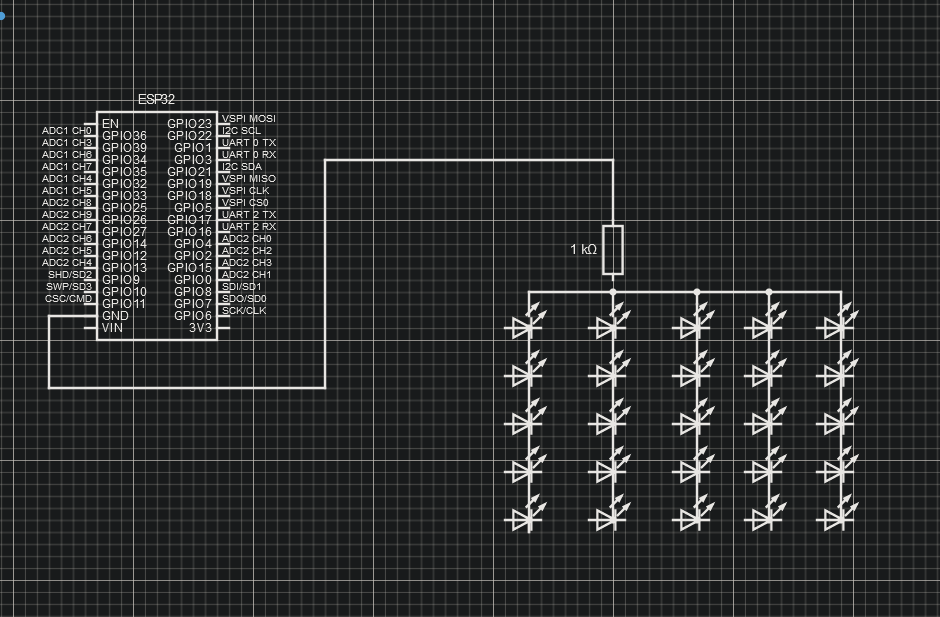
1. **Hardware Implementation:**
2. **ESP32 Pin Configuration:**
   * Connect LEDs' anodes (longer leads) to specific GPIO pins (specified in code).
   * Connect LEDs' cathodes (shorter leads) to the ground (GND).
3. **LED Connections:**
   * Connect smiley, sad, amazed, and poker face LEDs to respective GPIO pins.
   * Connect eye LEDs to GPIO pins 17 and 16.
4. **Button Connection:**
   * Connect one button terminal to GPIO pin 27.
   * Connect the other button terminal to ground (GND).
5. **Power Supply:**
   * Power the ESP32 using appropriate source (USB, battery).
6. **Circuit Setup:**
   * Arrange components on a breadboard or custom PCB.
   * Use jumper wires for connections based on code's pin configuration.
7. **Ground and Power:**
   * Connect GND pins of all components to common ground rail.
   * Provide appropriate voltage (3.3V/5V) to power rail.
8. **Debouncing:**
   * Implement button debouncing using capacitors or software to prevent noise.
9. **Testing and Troubleshooting:**
   * Upload code to ESP32 using Arduino IDE.
   * Test circuit by pressing button to observe LED changes.
   * Debug or adjust as needed.
10. **Software Implementation**

* **Include Libraries:** The code begins with an #include statement that includes the necessary Arduino library for the code to work properly.
* **Variable Declarations:**
  + Various constants are declared to represent pin numbers for the eyes, emotional expressions, and buttons.
  + currentEmotion keeps track of the currently displayed emotion.
  + numEmotions indicates the total number of emotions.
* **Setup Function:**
  + setup() is the function that runs once when the microcontroller starts up.
  + It sets up the pins for eyes and emotions as outputs.
  + Initializes the eyes to be on (HIGH) using digitalWrite().
* **Functions for Controlling LEDs:**
  + turnOnPins() turns on a set of LEDs by setting their respective pins to HIGH.
  + turnOffPins() turns off a set of LEDs by setting their respective pins to LOW.
  + turnOnEyes() and turnOffEyes() control the eye LEDs by manipulating their pins.
* **Function to Show Emotion:**
  + showEmotion() takes an emotion index and:
    - * Turns off all emotion-related LEDs using turnOffPins().
      * Turns on the specific set of LEDs for the emotion using turnOnPins().
* **Loop Function:**
  + loop() is the main function that runs repeatedly after setup().
  + Reads the state of the button using digitalRead() and checks if it's pressed (LOW).
  + Implements debounce mechanism by checking the time elapsed since last press.
  + If the button was pressed and debounce time has passed, it advances to the next emotion using showEmotion().
* **Overall Operation:**
  + The code continuously loops, checking for button presses.
  + When the button is pressed and the debounce time has passed, it changes to the next emotion using showEmotion().
  1. The eyes are always on, while the specific emotion-related LEDs are turned on based on the current emotion index.

The software implementation coordinates the interaction between the button and the LEDs to display different emotional expressions. It ensures that the LEDs representing the current emotion light up while maintaining the eyes' constant illumination. The debounce mechanism prevents unintended multiple button presses. The main loop is responsible for continuously checking the button state and updating the displayed emotion accordingly.

1. **Circuit Schematic**

The circuit schematic involves an ESP32 microcontroller, LEDs, and a button. The ESP32 is the central component controlling the display of different emotional expressions. The LEDs, including those for eyes and various emotions (smiley, sad, amazed, poker), are connected to specific GPIO pins on the ESP32. The eyes LEDs (pins 17 and 16) remain continuously illuminated, while the emotional expression LEDs change based on user input. A button connected to pin 27 allows users to switch between different emotions. Proper connections, such as resistors and ground, are established to ensure smooth operation and prevent damage to components. The ESP32 processes the button input, activates corresponding LEDs, and produces a dynamic visual representation of the chosen emotional expression.

****

1. **Code**

**#include <Arduino.h>**

**const int eyesPin1 = 17;**

**const int eyesPin2 = 16;**

**const int smileyPins[] = {4, 15, 25, 26, 0};**

**const int sadPins[] = {39, 13, 15, 26, 25};**

**const int amazedPins[] = {32, 33, 15, 26};**

**const int pokerPins[] = {25, 15, 26};**

**const int buttonPin = 27;**

**int currentEmotion = 0;**

**int numEmotions = 4;**

**unsigned long lastDebounceTime = 0;**

**unsigned long debounceDelay = 200; // Increase debounce delay for better response**

**void setup() {**

**pinMode(eyesPin1, OUTPUT);**

**pinMode(eyesPin2, OUTPUT);**

**digitalWrite(eyesPin1, HIGH);**

**digitalWrite(eyesPin2, HIGH);**

**for (int i = 0; i < sizeof(smileyPins) / sizeof(smileyPins[0]); i++) {**

**pinMode(smileyPins[i], OUTPUT);**

**pinMode(sadPins[i], OUTPUT);**

**}**

**for (int i = 0; i < sizeof(amazedPins) / sizeof(amazedPins[0]); i++) {**

**pinMode(amazedPins[i], OUTPUT);**

**}**

**for (int i = 0; i < sizeof(pokerPins) / sizeof(pokerPins[0]); i++) {**

**pinMode(pokerPins[i], OUTPUT);**

**}**

**pinMode(buttonPin, INPUT\_PULLUP);**

**}**

**void turnOnPins(const int pins[], int count) {**

**for (int i = 0; i < count; i++) {**

**digitalWrite(pins[i], HIGH);**

**}**

**}**

**void turnOffPins(const int pins[], int count) {**

**for (int i = 0; i < count; i++) {**

**digitalWrite(pins[i], LOW);**

**}**

**}**

**void showEmotion(int emotionIndex) {**

**currentEmotion = emotionIndex;**

**turnOffPins(smileyPins, sizeof(smileyPins) / sizeof(smileyPins[0]));**

**turnOffPins(sadPins, sizeof(sadPins) / sizeof(sadPins[0]));**

**turnOffPins(amazedPins, sizeof(amazedPins) / sizeof(amazedPins[0]));**

**turnOffPins(pokerPins, sizeof(pokerPins) / sizeof(pokerPins[0]));**

**switch (currentEmotion) {**

**case 0:**

**turnOnPins(smileyPins, sizeof(smileyPins) / sizeof(smileyPins[0]));**

**break;**

**case 1:**

**turnOnPins(sadPins, sizeof(sadPins) / sizeof(sadPins[0]));**

**break;**

**case 2:**

**turnOnPins(amazedPins, sizeof(amazedPins) / sizeof(amazedPins[0]));**

**break;**

**case 3:**

**turnOnPins(pokerPins, sizeof(pokerPins) / sizeof(pokerPins[0]));**

**break;**

**}**

**}**

**void loop() {**

**int buttonState = digitalRead(buttonPin);**

**if (buttonState == LOW && millis() - lastDebounceTime > debounceDelay) {**

**showEmotion((currentEmotion + 1) % numEmotions);**

**lastDebounceTime = millis();**

**}**

**}**