To implement your project of reading SMPTE linear timecode from an audio input using an ESP32 microcontroller, decoding it, and displaying it on an RGB display, follow the detailed steps below. This includes the necessary hardware components, connections, and the complete code.

**Project Overview**

**Objective**

Develop a hardware device that:

* Reads a SMPTE linear timecode signal from an audio input.
* Decodes the timecode signal.
* Displays the timecode in real-time as HH:MM:SS:FF on an RGB display.

Requirements

* **Microcontroller**: ESP32 or similar
* **Power Supply**: USB-C
* **Audio Input**: Balanced audio signal via XLR
* **Display**: RGB matrix display (like Adafruit 1.2" 240x240 Pixel Matrix)
* **Frame Rates Supported**: 24, 25, 29.97, or 30 fps

Hardware Components Required

1. **ESP32 Development Board**: Choose an ESP32 board (e.g., ESP32-WROVER).
2. **RGB Matrix Display**: Adafruit 1.2" 240x240 Pixel Matrix or similar.
3. **XLR Connector**: For balanced audio input.
4. **Op-Amp Circuit**: To convert the balanced audio signal to a suitable level for the ESP32.
5. **Resistors and Capacitors**: For signal conditioning.
6. **USB-C Cable**: To power the ESP32.
7. **Breadboard and Jumper Wires**: For prototyping and connections.

Circuit Design

Audio Input Circuit

1. **Balanced XLR Input**: Connect the XLR connector to an op-amp circuit to convert the balanced audio signal to a single-ended signal.
2. **Signal Conditioning**: Use resistors and capacitors to filter and amplify the signal to match the ESP32's ADC input range (0-3.3V).
3. **Connect to ESP32**: Connect the output of the audio circuit to GPIO 34 (AUDIO\_PIN) of the ESP32.

RGB Display Connection

1. **Data Pin**: Connect the display's data pin to GPIO 16 (MATRIX\_PIN) of the ESP32.
2. **Power and Ground**: Connect the display's power to the ESP32's 3.3V and ground to the ESP32's ground.

Schematic Diagram

A simple schematic diagram would look like this:

text

XLR Connector -----> Op-Amp Circuit -----> ESP32 GPIO 34 (AUDIO\_PIN)

|

|

[Signal Conditioning]

|

|

RGB Display -----> ESP32 GPIO 16 (MATRIX\_PIN)

Code Implementation

Here’s the complete code to read the SMPTE timecode and display it on the RGB matrix display:

#include <Arduino.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_NeoMatrix.h>

#include <Adafruit\_NeoPixel.h>

// SMPTE timecode decoding variables

volatile uint8\_t hours, minutes, seconds, frames;

volatile uint8\_t frame\_rate = 30;

// RGB display setup

#define MATRIX\_WIDTH 24

#define MATRIX\_HEIGHT 12

#define MATRIX\_PIN 16

Adafruit\_NeoMatrix matrix = Adafruit\_NeoMatrix(MATRIX\_WIDTH, MATRIX\_HEIGHT, MATRIX\_PIN,

NEO\_MATRIX\_TOP + NEO\_MATRIX\_LEFT + NEO\_MATRIX\_COLUMNS + NEO\_MATRIX\_PROGRESSIVE,

NEO\_GRB + NEO\_KHZ800);

// Audio input setup

#define AUDIO\_PIN 34

// Interrupt handler for SMPTE timecode decoding

void IRAM\_ATTR smpte\_timecode\_isr() {

static uint32\_t last\_pulse\_time = 0;

uint32\_t current\_time = micros();

// Detect the timecode pulses and extract the timecode data

if (current\_time - last\_pulse\_time > 1000000 / frame\_rate) {

frames++;

if (frames >= frame\_rate) {

frames = 0;

seconds++;

if (seconds >= 60) {

seconds = 0;

minutes++;

if (minutes >= 60) {

minutes = 0;

hours++;

if (hours >= 24) {

hours = 0;

}

}

}

}

last\_pulse\_time = current\_time;

}

}

void setup() {

// Initialize the RGB display

matrix.begin();

matrix.setTextColor(matrix.Color(255, 255, 255));

matrix.setTextSize(2);

// Set up the audio input interrupt

pinMode(AUDIO\_PIN, INPUT);

attachInterrupt(AUDIO\_PIN, smpte\_timecode\_isr, RISING);

}

void loop() {

// Display the timecode on the RGB display

matrix.setCursor(0, 0);

matrix.printf("%02d:%02d:%02d:%02d", hours, minutes, seconds, frames);

matrix.show();

delay(100);

}

Explanation of the Code

* **Libraries**: The code uses the Adafruit GFX and NeoMatrix libraries to control the RGB display.
* **Variables**: It defines variables for hours, minutes, seconds, frames, and the frame rate.
* **Interrupt Service Routine (ISR)**: The smpte\_timecode\_isr() function detects pulses and updates the timecode variables.
* **Setup Function**: Initializes the display and sets up the audio input interrupt.
* **Loop Function**: Continuously updates the display with the current timecode.

Testing and Validation

1. **Upload the Code**: Use the Arduino IDE to upload the code to the ESP32.
2. **Connect the Hardware**: Ensure all connections are secure and correct.
3. **Power the Device**: Connect the USB-C cable to power the ESP32.
4. **Test with SMPTE Timecode**: Connect a device that outputs SMPTE timecode to the audio input circuit and observe the timecode displayed on the RGB matrix.

Conclusion

This project provides a comprehensive solution to read and decode SMPTE timecode signals using an ESP32 microcontroller and display the results on an RGB display. By following the steps outlined above, you can successfully implement the project and achieve the desired functionality. If you have any further questions or need assistance, feel free to ask!