Lab 3
CSE121
Xavier Thompson

ChatGPT Prompts

Prompt:
write a c program to send a string from an esp32c3 to an arduino LCD display over i2c

ChatGPT

To send a string from an ESP32-C3 to an Arduino LCD display over I2C, you'll need to use the Wire library on both devices for I2C communication. You should have the appropriate connections between the ESP32-C3 and the Arduino (with an I2C LCD module) in place. Ensure you have the Wire library installed on both the ESP32-C3 and the Arduino.

Here's a sample C program for the ESP32-C3 to send a string to the Arduino LCD display over I2C:

c
Copy code

```
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/i2c.h"

// I2C configurations
#define I2C_SCL_PIN 22 // Replace with your actual GPIO pin
#define I2C_SDA_PIN 21 // Replace with your actual GPIO pin
#define I2C_MASTER_FREQ_HZ 100000
#define I2C_LCD_ADDR 0x3F // Replace with the LCD I2C address

// LCD control commands
#define LCD_COMMAND 0x00
#define LCD_DATA 0x40

// Function to send data to I2C LCD
void sendStringToLCD(const char* data) {
    i2c_cmd_handle_t cmd = i2c_cmd_link_create();
    // Write control byte for data transmission
    i2c_master_start(cmd);
    i2c_master_write_byte(cmd, (I2C_LCD_ADDR << 1) | I2C_MASTER_WRITE, true);

    // Send data command
    i2c_master_write_byte(cmd, LCD_DATA, true);

    // Send the string data
    i2c_master_write(cmd, (uint8_t*)data, strlen(data), true);

    i2c_master_stop(cmd);

    i2c_master_cmd_begin(I2C_NUM_0, cmd, pdMS_TO_TICKS(1000));
    i2c_cmd_link_delete(cmd);
}

void app_main() {
    i2c_config_t conf;
    conf.mode = I2C_MODE_MASTER;
    conf.sda_io_num = I2C_SDA_PIN;
    conf.sda_pullup_en = GPIO_PULLUP_ENABLE;
    conf.scl_io_num = I2C_SCL_PIN;
    conf.scl_pullup_en = GPIO_PULLUP_ENABLE;
    conf.master.clk_speed = I2C_MASTER_FREQ_HZ;
    i2c_param_config(I2C_NUM_0, &conf);
    i2c_driver_install(I2C_NUM_0, I2C_MODE_MASTER, 0, 0, 0);

    const char* dataToSend = "Hello, Arduino!";

    sendStringToLCD(dataToSend);

    // You can add additional logic or tasks here.
}
```

Make sure to replace I2C_SCL_PIN, I2C_SDA_PIN, and I2C_LCD_ADDR with the actual pins and LCD I2C address you're using. This code initializes the I2C communication, sends the data to the Arduino LCD, and can be extended with additional functionality as needed.

On the Arduino side, you should have code to receive and display the data sent over I2C. The Arduino code would depend on the specific LCD module you're using, as there are different libraries and implementations for different LCD displays. You need to adapt the Arduino code to your particular LCD module.

Prompt 2: write a simple c program to replace wire.h LCD init function for espressif esp32c3

To replace the Wire library's LCD initialization function with a custom I2C LCD initialization function for an ESP32-C3, you would need to interact directly with the I2C bus and your specific LCD module. Here's a simplified example of how you can create a custom initialization function for a generic I2C LCD using the ESP-IDF framework:

ChatGPT

```
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/i2c.h"

#define I2C_SCL_PIN 22 // Replace with your actual GPIO pin
#define I2C_SDA_PIN 21 // Replace with your actual GPIO pin
#define I2C_LCD_ADDR 0x3F // Replace with your LCD I2C address

#define LCD_BACKLIGHT 0x08
#define LCD_ENABLE 0x04
#define LCD_COMMAND 0x00
#define LCD_DATA 0x01

void i2c_master_init() {
    i2c_config_t conf;
    conf.mode = I2C_MODE_MASTER;
    conf.sda_io_num = I2C_SDA_PIN;
    conf.sda_pullup_en = GPIO_PULLUP_ENABLE;
    conf.scl_io_num = I2C_SCL_PIN;
    conf.scl_pullup_en = GPIO_PULLUP_ENABLE;
    conf.master.clk_speed = 100000; // I2C speed
    i2c_param_config(I2C_NUM_0, &conf);
    i2c_driver_install(I2C_NUM_0, I2C_MODE_MASTER, 0, 0, 0);
}

void lcd_send_command(uint8_t value) {
    i2c_cmd_handle_t cmd = i2c_cmd_link_create();
    i2c_master_start(cmd);
    i2c_master_write_byte(cmd, (I2C_LCD_ADDR << 1) | I2C_MASTER_WRITE, true);
    i2c_master_write_byte(cmd, LCD_COMMAND, true);
    i2c_master_write_byte(cmd, value, true);
    i2c_master_stop(cmd);

    i2c_master_cmd_begin(I2C_NUM_0, cmd, 1000 / portTICK_RATE_MS);
    i2c_cmd_link_delete(cmd);
}

void lcd_init() {
    // LCD initialization commands
    vTaskDelay(50 / portTICK_RATE_MS); // Power-on delay
    lcd_send_command(0x33);
    lcd_send_command(0x32); // Set to 4-bit mode
    lcd_send_command(0x28); // Function set: 4-bit mode, 2 lines, 5x8 font
    lcd_send_command(0x0C); // Display on, cursor off, blink off
    lcd_send_command(0x06); // Entry mode set: increment, no shift
    lcd_send_command(0x01); // Clear display
    vTaskDelay(2 / portTICK_RATE_MS);
}

void app_main() {
    i2c_master_init();
    lcd_init();

    // Your code here

    while (1) {
        // Your application logic here
    }
}
```