CSE121 Lab2 report

Professor Renau
Jinsung Park
4/19/2023

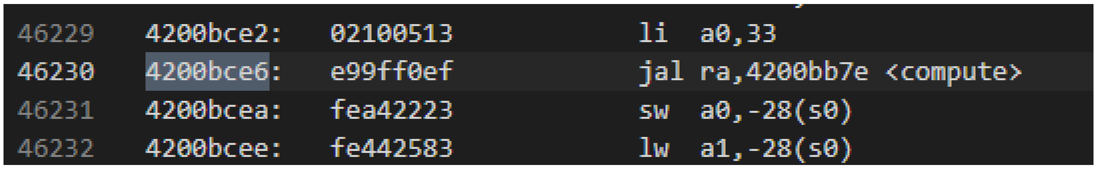
2.1

Configure openocd
In the case that openocd doesn't run because of permission: `wget https://raw.githubusercontent.com/espressif/openocd-esp32/master/contrib/60-openocd.rules`
Copy over file: `sudo mv 60-openocd.rules /etc/udev/rules.d`
Then apply change: `sudo udevadm trigger`
In a separate window of the project folder, let run the openocd: `idf.py openocd`

Download gdb
Use link to download risv gdb for arms: `wget https://github.com/espressif/binutils-gdb/releases/download/esp-gdb-v12.1_20221002/riscv32-esp-elf-gdb-12.1_20221002-aarch64-linux-gnu.tar.gz`
Unzip the file: `tar -xvzf risv32-esp-elf-gdb.tar.gz`
Run gdb: `./riscv32-esp-elf-gdb/bin/riscv32-esp-elf-gdb`

Run gdb
Create empty project: `idf.py create-project lab2_1`
Go inside new project: `cd lab2_1`
Set target: `idf.py set-target esp32c3`
Build project: `idf.py build`
Replace .elf file with the new one: `cp ~/esp/lab2_image.elf ./build/lab2_1.elf`
Flash project: `idf.py flash`
Run gdb on the .elf file: `riscv-esp-elf-gdb build/lab2_1.elf`

In a separate terminal, open assembly to find location of compute function
Disassemble .elf file: `riscv-esp-elf-objdump -d build/lab2_1.elf > lab2_asm.txt`
Open lab2_asm.txt and find the line where it says compute. Write down the memory address. There's actually two addresses. The write one you should look for is 0x4200bce6



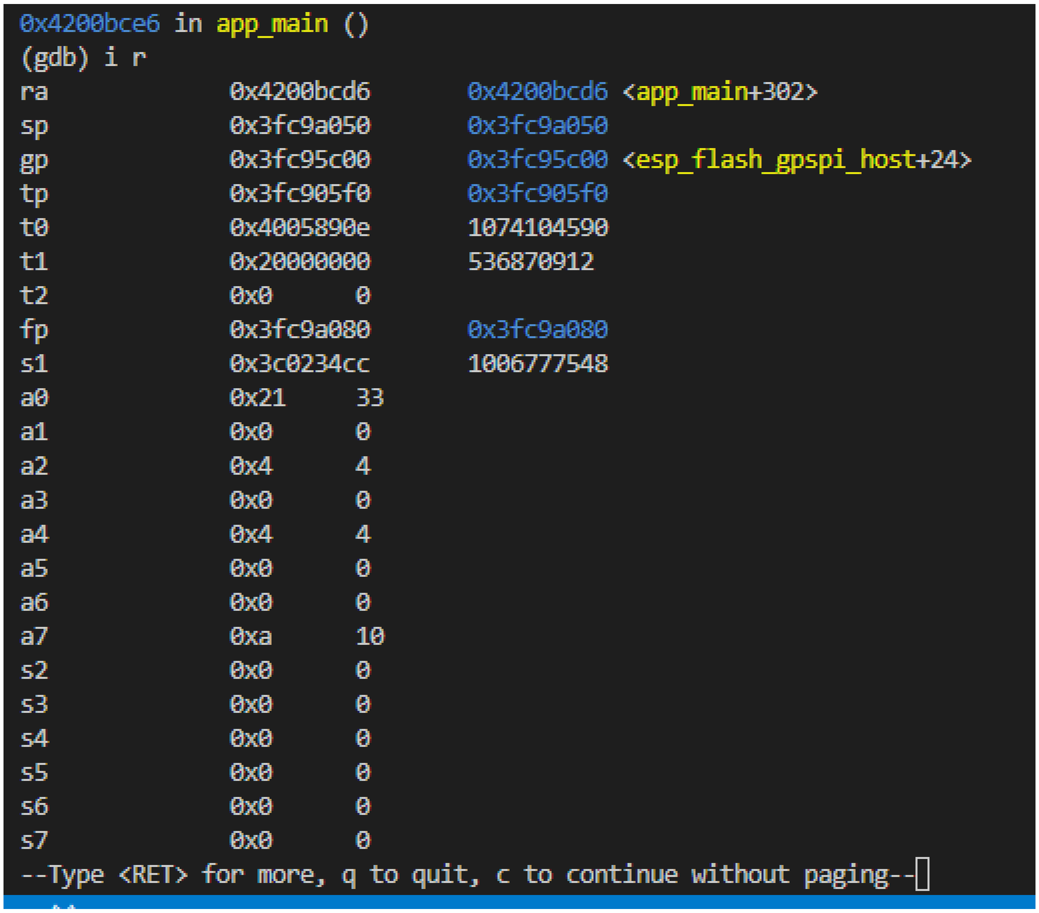
```
4200bce2:   02100513        li  a0,33
4200bce6:   e99ff0ef        jal ra,4200bb7e <compute>
4200bcea:   fea42223        sw  a0,-28(s0)
4200bcee:   fe442583        lw  a1,-28(s0)
```

Find variables in register
Inside gdb write gdbinit code,

```
target remote :3333
set remote hardware-watchpoint-limit 2
mon reset halt
```

flushregs
b compute
c
Go find the memory address of the compute function, keep typing “ni” until you get to that address. You should press “ni” approximately 30 times.

Once you are at that address, Check register: info registers
$a0, $a1, and $a2 are the 3 compute values



```
0x4200bce6 in app_main ()
(gdb) i r
ra             0x4200bcd6       0x4200bcd6 <app_main+302>
sp             0x3fc9a050       0x3fc9a050
gp             0x3fc95c00       0x3fc95c00 <esp_flash_gpspi_host+24>
tp             0x3fc905f0       0x3fc905f0
t0             0x4005890e       1074104590
t1             0x20000000       536870912
t2             0x0      0
fp             0x3fc9a080       0x3fc9a080
s1             0x3c0234cc       1006777548
a0             0x21     33
a1             0x0      0
a2             0x4      4
a3             0x0      0
a4             0x4      4
a5             0x0      0
a6             0x0      0
a7             0xa      10
s2             0x0      0
s3             0x0      0
s4             0x0      0
s5             0x0      0
s6             0x0      0
s7             0x0      0
--Type <RET> for more, q to quit, c to continue without paging--
```

Answer:
Compute 1st argument is 33
Compute 2nd argument is 0
Compute 3rd argument is 4

Links

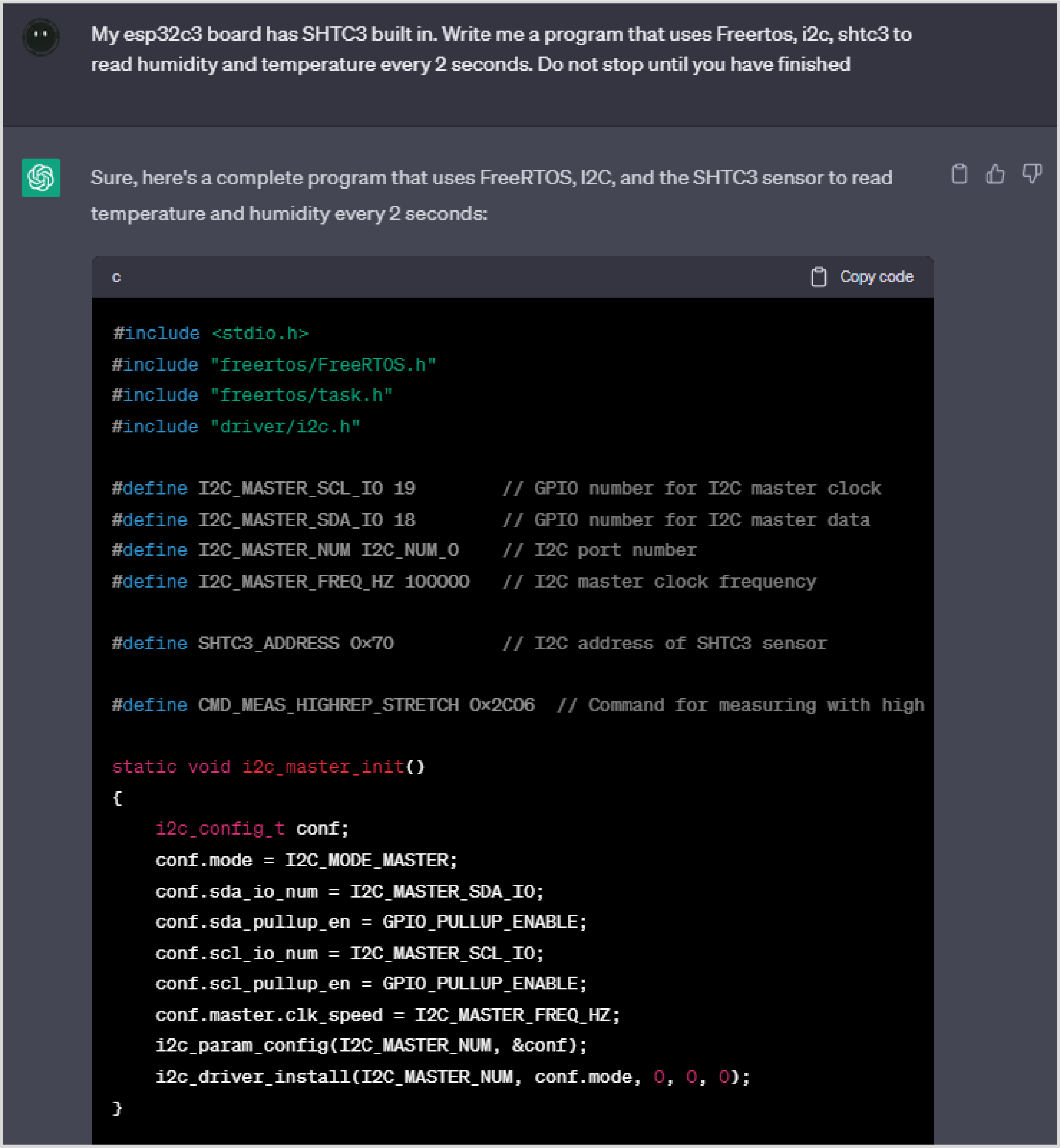
- https://docs.espressif.com/projects/esp-idf/en/latest/esp32c3/api-guides/tools/idf-tools.html?highlight=gdb
- https://web.cecs.pdx.edu/~apt/cs491/gdb.pdf
- https://riscv.org/wp-content/uploads/2015/01/riscv-calling.pdf

2.2

Professor’s code:

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

static const char *TAG = "lec2";

#define I2C_MASTER_SCL_IO 8 /*!< GPIO number for I2C master SCL */
#define I2C_MASTER_SDA_IO 10 /*!< GPIO number for I2C master SDA */
#define I2C_MASTER_NUM I2C_NUM_0
#define I2C_MASTER_TX_BUF_DISABLE 0
#define I2C_MASTER_RX_BUF_DISABLE 0
#define I2C_MASTER_FREQ_HZ 100000
#define SHTC3_SENSOR_ADDR 0x70 /*!< SHTC3 12C address */

#define SHTC3_CMD_READ_ID OxEFCS8
//#define SHTC3_CMD_MEASURE 0x7866
#define SHTC3_CMD_MEASURE 0x7CA2

static esp_err_t i2c_master_init()
{
    i2c_config_t conf = {
        .mode = I2C_MODE_MASTER,
        .sda_io_num = I2C_MASTER_SDA_IO,
        .sda_pullup_en = GPIO_PULLUP_ENABLE,
        .scl_io_num = I2C_MASTER_SCL_IO,
        .scl_pullup_en = GPIO_PULLUP_ENABLE,
        .master.clk_speed = I2C_MASTER_FREQ_HZ,
};

esp_err_t err = i2c_param_config(I2C_MASTER_NUM, &conf);
if (err != ESP_OK) {
    ESP_LOGE(TAG, "Failed to i2c_param_config %d", err);
    return err;
}

err = i2c_driver_install(I2C_MASTER_NUM, conf.mode, I2C_MASTER_RX_BUF_DISABLE,
I2C_MASTER_TX_BUF_DISABLE,0);
if(err != ESP_OK){
    ESP_LOGE(TAG, "Failed to i2c_driver_install %d", err);
    return err;
}
return err;
}

static esp_err_t shtc3_read(uint16_t command, uint8_t *data, size_t size){
    i2c_cmd_handle_t cmd = i2c_cmd_link_create();

    esp_err_t err;
#if 1
    i2c_master_start(cmd);
    i2c_master_write_byte(cmd, (SHTC3_SENSOR_ADDR <<1) | I2C_MASTER_WRITE, true);
    i2c_master_write_byte(cmd, command >> 8, true);
    i2c_master_write_byte(cmd, command & 0xFF, true);
    i2c_master_stop(cmd);
    err = i2c_master_cmd_begin(I2C_MASTER_NUM, cmd, pdMS_TO_TICKS(1000));
    if(err != ESP_OK){
        ESP_LOGE(TAG, "Failed to 1st write %d", err);

        i2c_cmd_link_delete(cmd);
        return err;
    }

    vTaskDelay(pdMS_TO_TICKS(20));

    #endif
    i2c_cmd_link_delete(cmd);

    cmd = i2c_cmd_link_create();
    i2c_master_start(cmd);
    i2c_master_write_byte(cmd, (SHTC3_SENSOR_ADDR << 1) | I2C_MASTER_READ, true);
    i2c_master_read(cmd, data, size, I2C_MASTER_LAST_NACK);
    i2c_master_stop(cmd);
    err = i2c_master_cmd_begin(I2C_MASTER_NUM, cmd, pdMS_TO_TICKS(1000));
    if(err != ESP_OK){
        ESP_LOGE(TAG, "failed to 2nd read %d", err);
    }

    i2c_cmd_link_delete(cmd);
    return err;
}

static float calculate_humidity(uint16_t raw_humidity)
{
    return 100.0 * (float)raw_humidity / 65535.0;
```

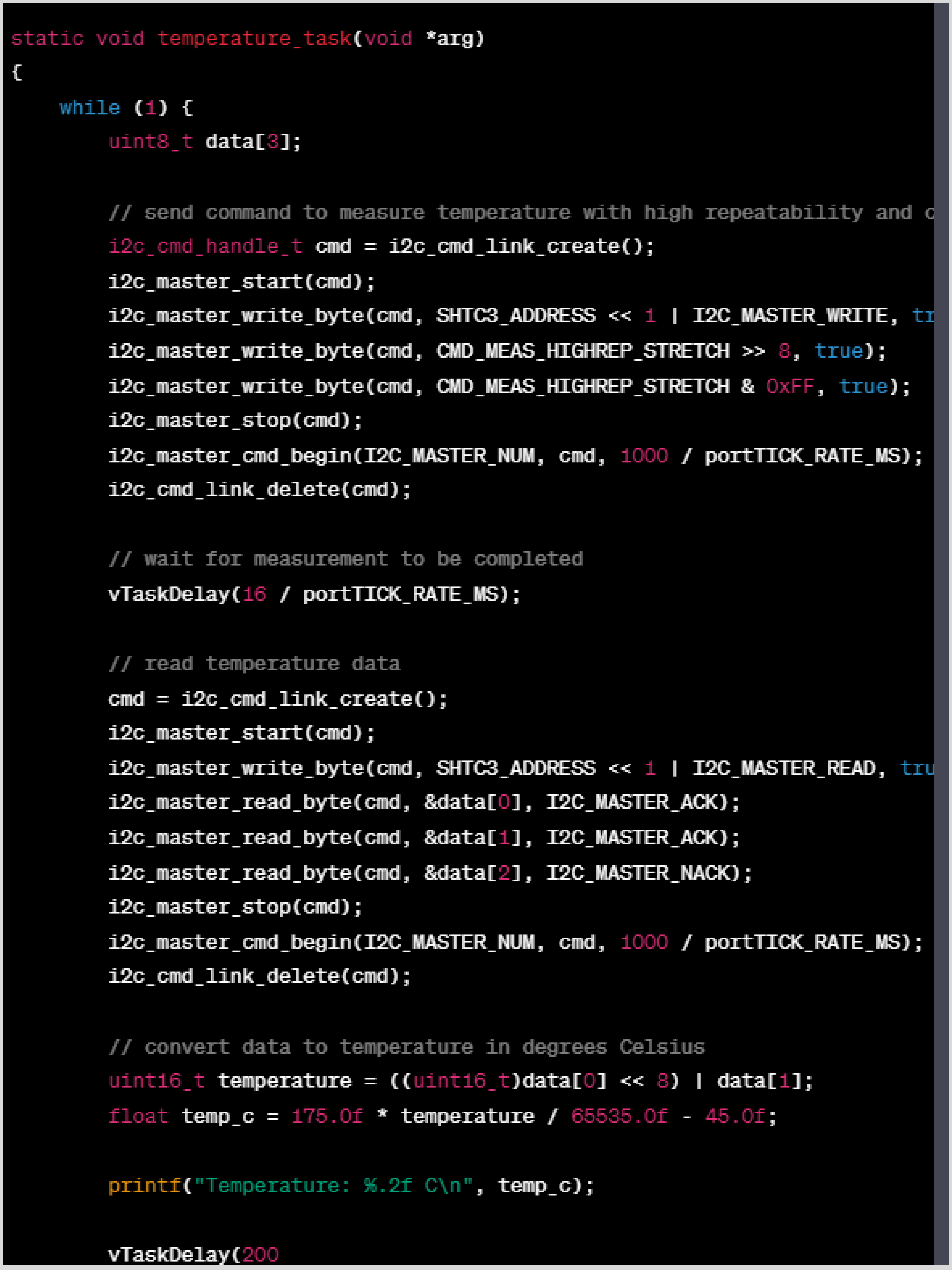
```
}

void shtc3_task(){
    while(1){

        uint8_t data[6] = {0,};
        uint16_t raw_humidity=0;

        esp_err_t err = shtc3_read(SHTC3_CMD_MEASURE, data, 6);
        if(err == ESP_OK){
            raw_humidity = (data[3] << 8) | data[4];
            float humidity = calculate_humidity(raw_humidity);
            ESP_LOGI(TAG, "Humidity: %.2f %%", humidity);
        } else {
            ESP_LOGI(TAG, "Failed to read data from SHTC3 sensor %d", err);
        }

        vTaskDelay(pdMS_TO_TICKS(2000));
    }
}
void app_main(void){
    esp_err_t err = i2c_master_init();
    if(err != ESP_OK) {
        ESP_LOGE(TAG, "Failed to initialize I2C master");
        return;
    }

    shtc3_task();
}
```



My esp32c3 board has SHTC3 built in. Write me a program that uses Freertos, i2c, shtc3 to read humidity and temperature every 2 seconds. Do not stop until you have finished

Sure, here's a complete program that uses FreeRTOS, I2C, and the SHTC3 sensor to read temperature and humidity every 2 seconds:

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

#define I2C_MASTER_SCL_IO 19        // GPIO number for I2C master clock
#define I2C_MASTER_SDA_IO 18        // GPIO number for I2C master data
#define I2C_MASTER_NUM I2C_NUM_0    // I2C port number
#define I2C_MASTER_FREQ_HZ 100000   // I2C master clock frequency

#define SHTC3_ADDRESS 0x70          // I2C address of SHTC3 sensor

#define CMD_MEAS_HIGHREP_STRETCH 0x2C06  // Command for measuring with high

static void i2c_master_init()
{
    i2c_config_t conf;
    conf.mode = I2C_MODE_MASTER;
    conf.sda_io_num = I2C_MASTER_SDA_IO;
    conf.sda_pullup_en = GPIO_PULLUP_ENABLE;
    conf.scl_io_num = I2C_MASTER_SCL_IO;
    conf.scl_pullup_en = GPIO_PULLUP_ENABLE;
    conf.master.clk_speed = I2C_MASTER_FREQ_HZ;
    i2c_param_config(I2C_MASTER_NUM, &conf);
    i2c_driver_install(I2C_MASTER_NUM, conf.mode, 0, 0, 0);
}
```



```
static void temperature_task(void *arg)
{
    while (1) {
        uint8_t data[3];

        // send command to measure temperature with high repeatability and c
        i2c_cmd_handle_t cmd = i2c_cmd_link_create();
        i2c_master_start(cmd);
        i2c_master_write_byte(cmd, SHTC3_ADDRESS << 1 | I2C_MASTER_WRITE, tr
        i2c_master_write_byte(cmd, CMD_MEAS_HIGHREP_STRETCH >> 8, true);
        i2c_master_write_byte(cmd, CMD_MEAS_HIGHREP_STRETCH & 0xFF, true);
        i2c_master_stop(cmd);
        i2c_master_cmd_begin(I2C_MASTER_NUM, cmd, 1000 / portTICK_RATE_MS);
        i2c_cmd_link_delete(cmd);

        // wait for measurement to be completed
        vTaskDelay(16 / portTICK_RATE_MS);

        // read temperature data
        cmd = i2c_cmd_link_create();
        i2c_master_start(cmd);
        i2c_master_write_byte(cmd, SHTC3_ADDRESS << 1 | I2C_MASTER_READ, tru
        i2c_master_read_byte(cmd, &data[0], I2C_MASTER_ACK);
        i2c_master_read_byte(cmd, &data[1], I2C_MASTER_ACK);
        i2c_master_read_byte(cmd, &data[2], I2C_MASTER_NACK);
        i2c_master_stop(cmd);
        i2c_master_cmd_begin(I2C_MASTER_NUM, cmd, 1000 / portTICK_RATE_MS);
        i2c_cmd_link_delete(cmd);

        // convert data to temperature in degrees Celsius
        uint16_t temperature = ((uint16_t)data[0] << 8) | data[1];
        float temp_c = 175.0f * temperature / 65535.0f - 45.0f;

        printf("Temperature: %.2f C\n", temp_c);

        vTaskDelay(200
```

Links

- https://www.mouser.com/datasheet/2/682/Sensirion_04202018_HT_DS_SHTC3_Preliminiary_D2-1323493.pdf
- https://github.com/esp-rs/esp-rust-board