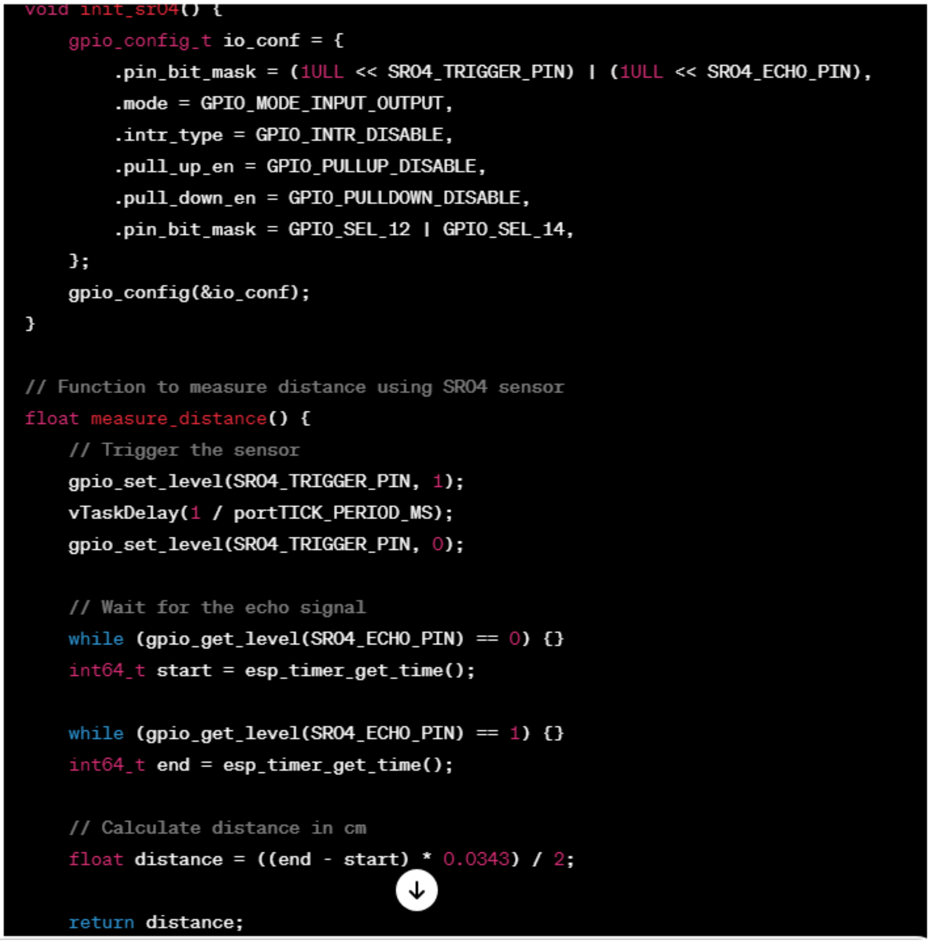
CSE 121: Lab 5 Report

By Daniel Beltran

11/21/2023

Lab 5: Part 1

For lab 5 the main objective was to use the US-100/SR04 to measure distance and using the SHTC3 sensor for the ESP32C3 for adjusting the speed of sound based on temperature. Fortunately, measuring temperature and humidity was the second part of lab 2 which allowed the recycling of code for lab 5. However, there were some alterations that needed to be made such as removing humidity as it was not required and creating some sort of function that allowed us to measure distance, which I had asked ChatGPT. My search query consisted of the following prompt “For the ESP32C3, you will make use of the SR04 compatible devices to measure distance and to adjust the speed of sound based on temperature.” With that, I also provided it with my previous code that initializes the I2C and read temperature.

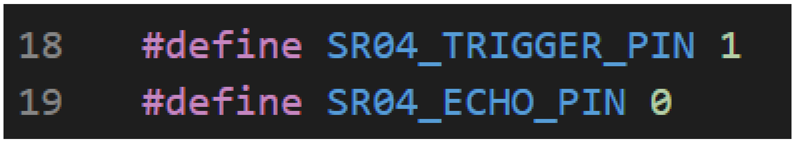


```
void init_sr04() {
    gpio_config_t io_conf = {
        .pin_bit_mask = (1ULL << SR04_TRIGGER_PIN) | (1ULL << SR04_ECHO_PIN),
        .mode = GPIO_MODE_INPUT_OUTPUT,
        .intr_type = GPIO_INTR_DISABLE,
        .pull_up_en = GPIO_PULLUP_DISABLE,
        .pull_down_en = GPIO_PULLDOWN_DISABLE,
        .pin_bit_mask = GPIO_SEL_12 | GPIO_SEL_14,
    };
    gpio_config(&io_conf);
}

// Function to measure distance using SR04 sensor
float measure_distance() {
    // Trigger the sensor
    gpio_set_level(SR04_TRIGGER_PIN, 1);
    vTaskDelay(1 / portTICK_PERIOD_MS);
    gpio_set_level(SR04_TRIGGER_PIN, 0);

    // Wait for the echo signal
    while (gpio_get_level(SR04_ECHO_PIN) == 0) {}
    int64_t start = esp_timer_get_time();

    while (gpio_get_level(SR04_ECHO_PIN) == 1) {}
    int64_t end = esp_timer_get_time();

    // Calculate distance in cm
    float distance = ((end - start) * 0.0343) / 2;

    return distance;
```

The following screenshot is what ChatGPT had provided for me which consisted of two functions that initialized the SR04 GPIO PINS and one that measured distance. Instead of creating another for measuring distance, handling it in **app_main()** was suitable as it was fairly small lines of code.

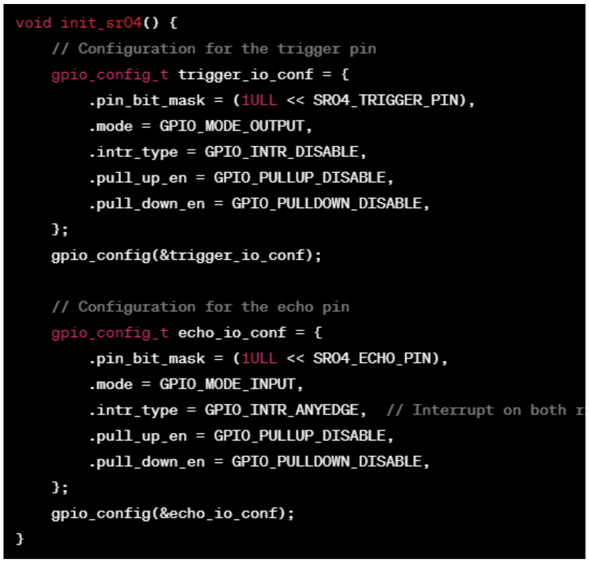
Before anything else, I had looked through the manual on the US-100 and learned about Trigger/Echo pulses that are to be defined in the program to connect the ESP32C3 to the Ultrasound Sensor. Essentially, the trigger allows for the ultrasonic sound to transmit from the sensor and the echo remains HIGH whilst the trigger is one to measure the distance. There will be a round-trip (i.e. to and from said object) so measuring the distance will require the time it takes for that trip multiplied by the speed of sound (in microseconds) and divided by 2.

The following constraints will be used for connecting the ESP32C GPIO pins to the SR04 to which it can be initialized. The pins on the board correspond to the definitions made below and must be connected on a breadboard with the sensor.



```
#define SR04_TRIGGER_PIN 1
#define SR04_ECHO_PIN 0
```

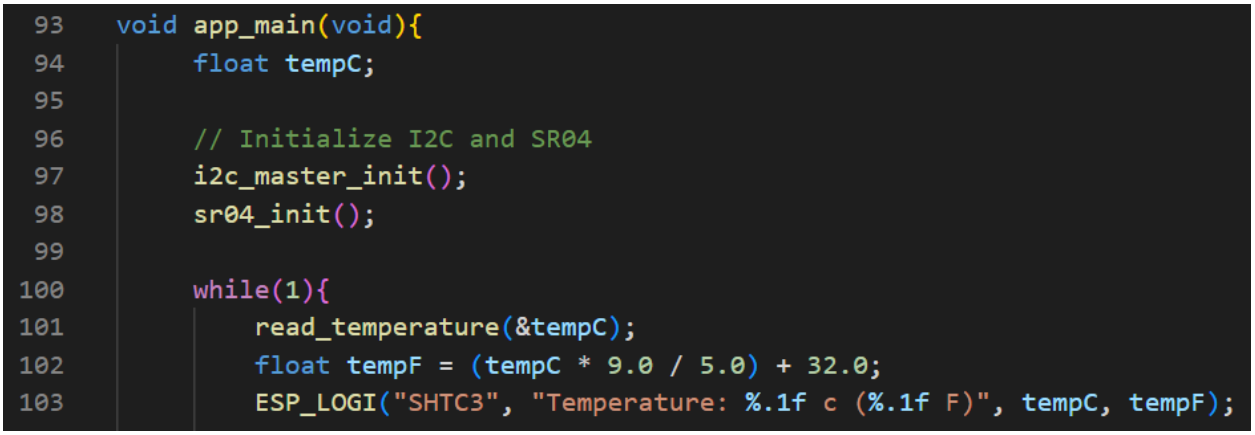
Now for the measurement of the distance required me to use the TRIGGER/ECHO pins but first I had to initialize them. Since ChatGPT provided one big configuration of the TRIGGER/ECHO, I had asked it again but specifically with initializing both pins, resulting in:



```
void init_sr04() {
    // Configuration for the trigger pin
    gpio_config_t trigger_io_conf = {
        .pin_bit_mask = (1ULL << SR04_TRIGGER_PIN),
        .mode = GPIO_MODE_OUTPUT,
        .intr_type = GPIO_INTR_DISABLE,
        .pull_up_en = GPIO_PULLUP_DISABLE,
        .pull_down_en = GPIO_PULLDOWN_DISABLE,
    };
    gpio_config(&trigger_io_conf);

    // Configuration for the echo pin
    gpio_config_t echo_io_conf = {
        .pin_bit_mask = (1ULL << SR04_ECHO_PIN),
        .mode = GPIO_MODE_INPUT,
        .intr_type = GPIO_INTR_ANYEDGE,  // Interrupt on both r
        .pull_up_en = GPIO_PULLUP_DISABLE,
        .pull_down_en = GPIO_PULLDOWN_DISABLE,
    };
    gpio_config(&echo_io_conf);
}
```

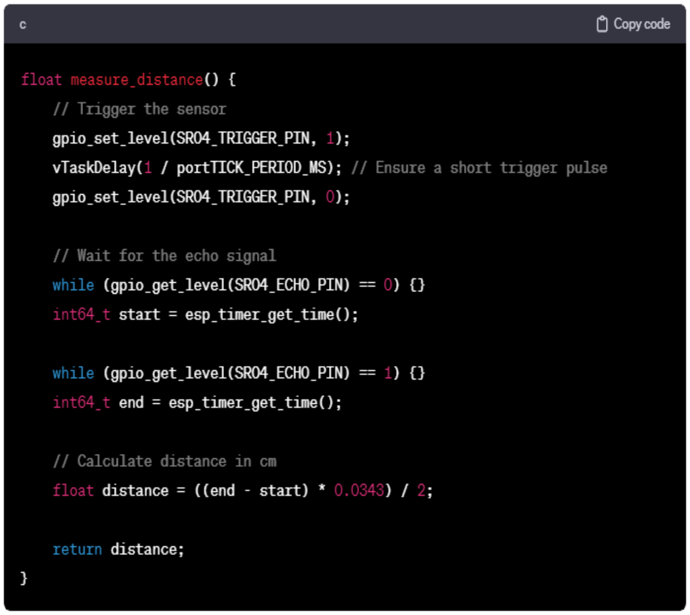
With **sr04_init()** being provided, the majority of the function was used with the .pull_up_en and .pull_down_en not necessary for the program to initialize SR04. One thing to note is that for configuration of the Echo pin in .intr_type, instead of using **GPIO_INTR_ANYEDGE** (i.e. which handles the rising/falling edge), the use of **GPIO_INTR_POSEDGE** primarily accounts for the rising edge, which is particularly needed in the program to remain HIGH when the trigger is set off.

So now that I had initialized the SR04, it was time to build **app_main()** and do all the necessary calls for functions. For starters, initializing the I2C and SR04 before entering the loop condition was always expected. Since lab2 part 2 worked with temperature, simply did the same thing with the function call and printed out the temperature from the sensor.



```c
void app_main(void){
    float tempC;

    // Initialize I2C and SR04
    i2c_master_init();
    sr04_init();

    while(1){
        read_temperature(&tempC);
        float tempF = (tempC * 9.0 / 5.0) + 32.0;
        ESP_LOGI("SHTC3", "Temperature: %.1f c (%.1f F)", tempC, tempF);
```

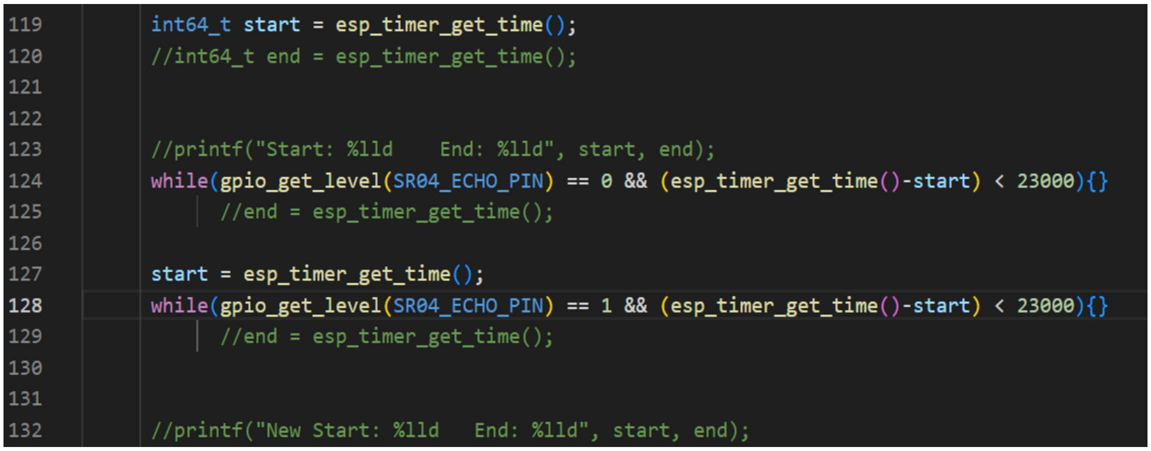
All that’s left is to measure the distance and also adjust the speed of sound depending on the temperature. Here is the logic template provided by ChatGPT:



```c
float measure_distance() {
    // Trigger the sensor
    gpio_set_level(SR04_TRIGGER_PIN, 1);
    vTaskDelay(1 / portTICK_PERIOD_MS); // Ensure a short trigger pulse
    gpio_set_level(SR04_TRIGGER_PIN, 0);

    // Wait for the echo signal
    while (gpio_get_level(SR04_ECHO_PIN) == 0) {}
    int64_t start = esp_timer_get_time();

    while (gpio_get_level(SR04_ECHO_PIN) == 1) {}
    int64_t end = esp_timer_get_time();

    // Calculate distance in cm
    float distance = ((end - start) * 0.0343) / 2;

    return distance;
}
```

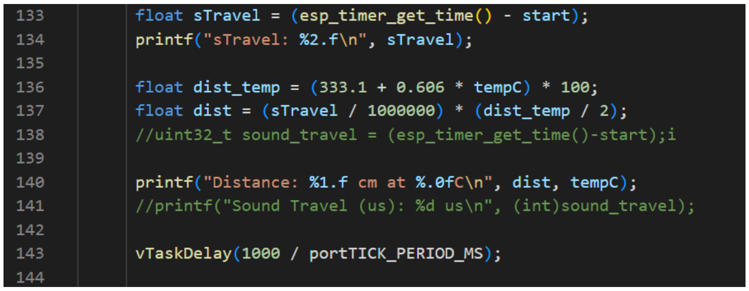
Setting the levels of the GPIO was done in the manner to mimic how TRIGGER/ECHO works, as mentioned before. Using **gpio_set_level()** we use this function to set the SR04 trigger to 1 (or active) to indicate the TRIGGER is high and we delay by 10 microseconds, though setting the level to 0 beforehand to initialize. Instead of vTaskDelay(), using **esp_rom_delay_us()** to clearly indicate the use of microseconds and for readability to indicate that here is when we allow the TRIGGER to stay high. After the 10 microsecond delay, we set the level back to 0.

The next part provides us with the majority of the logic that we need, two loop conditions that check for the GPIO set level to be literally 1 or 0. Setting up two variables for start and end (in my program, I used **esp_timer_get_time()** directly as my "end" variable) to calculate the echo pulse from the start to the end of the receiver. Aside from checking the level of the GPIO, we need to specify boundary conditions of detection as the maximum for the US-100 is roughly 2 cm to 450 cm. From here, using the maximum distance we can create a timeout value that corresponds to the end of the program once the start-to-end distance exceeds a certain timeout. This threshold can vary with testing but using such a large number ensures the program will run smoothly as giving less amount of time can stop the program suddenly.



```
    int64_t start = esp_timer_get_time();
    //int64_t end = esp_timer_get_time();


    //printf("Start: %lld    End: %lld", start, end);
    while(gpio_get_level(SR04_ECHO_PIN) == 0 && (esp_timer_get_time()-start) < 23000){}
        //end = esp_timer_get_time();

    start = esp_timer_get_time();
    while(gpio_get_level(SR04_ECHO_PIN) == 1 && (esp_timer_get_time()-start) < 23000){}
        //end = esp_timer_get_time();


    //printf("New Start: %lld   End: %lld", start, end);
```

From the provided code of ChatGPT, using start before each loop condition to restart timer was more than enough and calling on esp_timer_get_time() directly worked just fine. As you see from the print statements, it was a lot of figuring out what the values mean. Since a lot of my errors occurred with either: there was no detection of the sensor (needed to use male-female jumper wires not male-male), insanely large values, and values that incremented, so printing of each variable allowed me to understand how the programming was operating as it ran.



```
    float sTravel = (esp_timer_get_time() - start);
    printf("sTravel: %2.f\n", sTravel);

    float dist_temp = (333.1 + 0.606 * tempC) * 100;
    float dist = (sTravel / 1000000) * (dist_temp / 2);
    //uint32_t sound_travel = (esp_timer_get_time()-start);i

    printf("Distance: %1.f cm at %.0fC\n", dist, tempC);
    //printf("Sound Travel (us): %d us\n", (int)sound_travel);

    vTaskDelay(1000 / portTICK_PERIOD_MS);

```

Storing a float variable **sTravel** to indicate the round-trip time that it took from the start to end of the receiver. Since we must adjust the speed of sound to temperature, the formula provided by ChatGPT isn't sufficient. When searching for information on the topic, the first page gave a header with a summary of what I had searched for. The formula was provided with the description of units (meters per second) which can be seen on line 136, link provided below. Passing in our **tempC**, handling temperature from the ESP sensor, while also multiplying the whole formula by 100 to properly convert to centimeters per second. Line 137 does the calculation of distance by properly dividing the temperature distance by 2 (as mentioned before) while multiplying it by the total distance traveled divided by 1,000,000 as it converts the final value to microseconds.

Overall the objective of this lab was fairly simple but there was a lot of trial and error when testing the data values. Reading through conceptual review of trigger/echo/distance immensely helped with the thinking process and allowed me to piece everything together. Though ensuring the sensor was picking up consistent values was the real difficulty in the lab as it was hard to debug.

US-100 Datasheet:

https://media.digikey.com/pdf/Data%20Sheets/Adafruit%20PDFs/4019_Web.pdf

Trigger, Echo, and Distance:

https://lastminuteengineers.com/arduino-sr04-ultrasonic-sensor-tutorial/

Formula for Sound + Temperature:

https://study.com/learn/lesson/speed-sound-overview-equation.html#:~:text=The%20formula%20is%20v%20%3D%20331,be%20343.2%20meters%20per%20second.