

Lab 6 report


6.1:

Working on this project probably took longer than it should have. I spent about three hours trying to get the ultrasound sensor working on I2C, thinking that that was the only way we could communicate to it. I had the SCL hooked up to the trigger and SDA to the echo, but I think the SCL was too fast and would prevent the ultrasound from waiting for the echo. I spent a lot of time adjusting and tweaking commands thinking that was the issue since I would get ESP_ERR_TIMEOUT whenever I would try to write a pulse. It was after I saw the professor's lecture that I realized we could use GPIO, which made things easier. I [generated some code](#) which didn't need any adjustments. Then I referenced [this website](#) for the formula of the speed of sound with temperature, which is $v=311+(0.61*T)$. After that I copied over a file from lab 3 which had all of my temperature measurement functions. It needed some adjustments where I re-enabled a function to initialize the I2C. Then it was simple as adding the function to the distance measurement program.

I have a SR04 ultrasound sensor connected to a ESP32C3 using gpio 0 for trigger and gpio 1 for echo. Can you write the code for it using esp-idf?

Sure! Below is an example of how you can use the ESP-IDF framework to read distance measurements from the HC-SR04 ultrasonic sensor using an ESP32-C3 microcontroller. This code uses GPIO 0 for the trigger pin and GPIO 1 for the echo pin.

c

 Copy code

```
#include <stdio.h>
#include <string.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/gpio.h"
#include "esp_log.h"
#include "esp_timer.h"

#define TRIGGER_PIN GPIO_NUM_0
#define ECHO_PIN GPIO_NUM_1
```



```
static const char *TAG = "SR04";

void sr04_init(void)
{
    gpio_pad_select_gpio(TRIGGER_PIN);
    gpio_set_direction(TRIGGER_PIN, GPIO_MODE_OUTPUT);
    gpio_pad_select_gpio(ECHO_PIN);
    gpio_set_direction(ECHO_PIN, GPIO_MODE_INPUT);
}

int64_t measure_distance()
{
    // Send a 10us pulse to trigger the sensor
    gpio_set_level(TRIGGER_PIN, 0);
    vTaskDelay(pdMS_TO_TICKS(2)); // Wait for 2 ms
    gpio_set_level(TRIGGER_PIN, 1);
    ets_delay_us(10); // Wait for 10 us
    gpio_set_level(TRIGGER_PIN, 0);

    // Wait for the echo pin to go high and measure the time
    int64_t start_time = 0;
    int64_t end_time = 0;
    while (gpio_get_level(ECHO_PIN) == 0)
```

```

{
    start_time = esp_timer_get_time();
}
while (gpio_get_level(ECHO_PIN) == 1)
{
    end_time = esp_timer_get_time();
}

// Calculate the distance in cm
int64_t duration = end_time - start_time;
float distance = (duration / 2.0) * 0.0343;

ESP_LOGI(TAG, "Measured distance: %.2f cm", distance);
return distance;
}

void app_main(void)
{
    sr04_init();
    while (1)
    {
        measure_distance();
        vTaskDelay(pdMS_TO_TICKS(1000)); // Delay for 1 second
    }
}

```

```
}  
}
```

Explanation:

1. **Initialization** (``sr04_init``): The trigger pin is configured as an output and the echo pin as an input.
2. **Measurement** (``measure_distance``):
 - The trigger pin is set low, then high for 10 microseconds, and then low again to generate the trigger pulse.
 - The code then waits for the echo pin to go high, records the start time, and waits for it to go low again to record the end time.
 - The distance is calculated using the time difference between the start and end times, taking into account the speed of sound (0.0343 cm/us).
3. **Main Loop** (``app_main``): The main loop continuously calls ``measure_distance`` and logs the result every second.

Notes:



- This code assumes the use of the ESP-IDF framework, so make sure you have the ESP-IDF installed and properly set up.
- The ``ets_delay_us`` function is used for microsecond delays, which is available in the ESP-IDF.
- The ``esp_timer_get_time`` function is used to get the current time in microseconds.
- The distance is calculated assuming the speed of sound is 343 meters per second (or 0.0343 cm/us).

Make sure to include this code in your main application file and configure your ``CMakeLists.txt`` and other build files as required by ESP-IDF.