

# Lab 4 Report – Ryan King

## Lab 4 Part 2:

For part 2 and 3 I got help from ChatGPT in converting from Arduino.

<https://chatgpt.com/share/f58af14e-3f6b-4e63-9ef9-cb70b3d373fd>

(Although I gave up on the accelerometer approach for part 3 and just used Yuja and my previous code)

## Lab 4 Part 3:

I did ask ChatGPT as stated above, but ended up not using it's code.

“Given an accelerometer (esp-idf) that measures x and y acceleration, how can i get x and y velocity to turn it into a mouse?”

```
#include "accelerometer_driver.h" // Your accelerometer driver

#define ACCEL_X_ADDR 0x00 // Replace with actual register address
#define ACCEL_Y_ADDR 0x01 // Replace with actual register address

static const char *TAG = "MouseExample";
float velocity_x = 0.0;
float velocity_y = 0.0;
float previous_accel_x = 0.0;
float previous_accel_y = 0.0;
float dt = 0.01; // Assume 100 Hz sampling rate (10 ms)

void app_main() {
    // Initialize I2C and accelerometer
    i2c_master_init();
    accelerometer_init();

    while (1) {
        // Read acceleration data
        float accel_x = accelerometer_read(ACCEL_X_ADDR) * 9.81; // Convert to g
        float accel_y = accelerometer_read(ACCEL_Y_ADDR) * 9.81; // Convert to g

        // Integrate acceleration to get velocity
        velocity_x += 0.5 * (previous_accel_x + accel_x) * dt;
        velocity_y += 0.5 * (previous_accel_y + accel_y) * dt;

        // Store current acceleration for next iteration
        previous_accel_x = accel_x;
        previous_accel_y = accel_y;

        // Apply drift correction (simple example)
        if (abs(accel_x) < 0.1) velocity_x *= 0.99;
        if (abs(accel_y) < 0.1) velocity_y *= 0.99;

        // Map velocity to mouse movement
        int mouse_dx = (int)(velocity_x * SCALE_FACTOR);
        int mouse_dy = (int)(velocity_y * SCALE_FACTOR);

        // Move mouse (you need a function to send this data to the OS)
        move_mouse(mouse_dx, mouse_dy);

        // Log data for debugging
        ESP_LOGI(TAG, "Accel X: %.2f, Accel Y: %.2f, Vel X: %.2f, Vel Y: %.2f",
            accel_x, accel_y, velocity_x, velocity_y);

        // Delay for the next sample
        vTaskDelay(pdMS_TO_TICKS(10)); // 10 ms delay
    }
}
```

“The accelerometer outputs int16\_t, please account for this”

```
float previous_accel_x = 0.0;
float previous_accel_y = 0.0;
float dt = 0.01; // Assume 100 Hz sampling rate (10 ms)

void app_main() {
    // Initialize I2C and accelerometer
    i2c_master_init();
    accelerometer_init();

    while (1) {
        // Read raw acceleration data (int16_t)
        int16_t raw_accel_x = accelerometer_read_raw(ACCEL_X_ADDR);
        int16_t raw_accel_y = accelerometer_read_raw(ACCEL_Y_ADDR);

        // Convert raw data to acceleration in g
        float accel_x_g = raw_accel_x * ACCEL_SENSITIVITY;
        float accel_y_g = raw_accel_y * ACCEL_SENSITIVITY;

        // Convert acceleration in g to m/s²
        float accel_x = accel_x_g * GRAVITY;
        float accel_y = accel_y_g * GRAVITY;

        // Integrate acceleration to get velocity
        velocity_x += 0.5 * (previous_accel_x + accel_x) * dt;
        velocity_y += 0.5 * (previous_accel_y + accel_y) * dt;

        // Store current acceleration for next iteration
        previous_accel_x = accel_x;
        previous_accel_y = accel_y;

        // Apply drift correction (simple example)
        if (abs(accel_x) < 0.1) velocity_x *= 0.99;
        if (abs(accel_y) < 0.1) velocity_y *= 0.99;

        // Map velocity to mouse movement
        int mouse_dx = (int)(velocity_x * SCALE_FACTOR);
        int mouse_dy = (int)(velocity_y * SCALE_FACTOR);

        // Move mouse (you need a function to send this data to the OS)
        move_mouse(mouse_dx, mouse_dy);

        // Log data for debugging
        ESP_LOGI(TAG, "Raw Accel X: %d, Raw Accel Y: %d, Accel X: %.2f,
                    raw_accel_x, raw_accel_y, accel_x, accel_y, velocity_x

        // Delay for the next sample
        vTaskDelay(pdMS_TO_TICKS(10)); // 10 ms delay
    }
}
```

“how do I stop the velocity from going the other way when I stop the mouse due to inertia?”

```
void app_main() {
    // Initialize I2C and accelerometer
    i2c_master_init();
    accelerometer_init();

    while (1) {
        // Read raw acceleration data (int16_t)
        int16_t raw_accel_x = accelerometer_read_raw(ACCEL_X_ADDR);
        int16_t raw_accel_y = accelerometer_read_raw(ACCEL_Y_ADDR);

        // Convert raw data to acceleration in g
        float accel_x_g = raw_accel_x * ACCEL_SENSITIVITY;
        float accel_y_g = raw_accel_y * ACCEL_SENSITIVITY;

        // Convert acceleration in g to m/s²
        float accel_x = accel_x_g * GRAVITY;
        float accel_y = accel_y_g * GRAVITY;

        // Apply deadzone
        if (fabs(accel_x) < DEADZONE_THRESHOLD) accel_x = 0.0;
        if (fabs(accel_y) < DEADZONE_THRESHOLD) accel_y = 0.0;

        // Integrate acceleration to get velocity
        velocity_x += 0.5 * (previous_accel_x + accel_x) * dt;
        velocity_y += 0.5 * (previous_accel_y + accel_y) * dt;

        // Apply damping to reduce velocity over time
        velocity_x *= DAMPING_FACTOR;
        velocity_y *= DAMPING_FACTOR;

        // Store current acceleration for next iteration
        previous_accel_x = accel_x;
        previous_accel_y = accel_y;

        // Map velocity to mouse movement
        int mouse_dx = (int)(velocity_x * SCALE_FACTOR);
        int mouse_dy = (int)(velocity_y * SCALE_FACTOR);

        // Move mouse (you need a function to send this data to the OS)
        move_mouse(mouse_dx, mouse_dy);

        // Log data for debugging
        ESP_LOGI(TAG, "Raw Accel X: %d, Raw Accel Y: %d, Accel X: %.2f,
                    raw_accel_x, raw_accel_y, accel_x, accel_y, velocity_x

        // Delay for the next sample
        vTaskDelay(pdMS_TO_TICKS(10)); // 10 ms delay
```

“The problem is that if I move the board left, it picks up on that acceleration, but as the board decelerates, it records that as rightwards movement. I want it to give me its velocity which is still leftwards”

```
i2c_master_init();
accelerometer_init();

while (1) {
    // Read raw acceleration data (int16_t)
    int16_t raw_accel_x = accelerometer_read_raw(ACCEL_X_ADDR);
    int16_t raw_accel_y = accelerometer_read_raw(ACCEL_Y_ADDR);

    // Convert raw data to acceleration in g
    float accel_x_g = raw_accel_x * ACCEL_SENSITIVITY;
    float accel_y_g = raw_accel_y * ACCEL_SENSITIVITY;

    // Convert acceleration in g to m/s²
    float accel_x = accel_x_g * GRAVITY;
    float accel_y = accel_y_g * GRAVITY;

    // Apply deadzone
    if (fabs(accel_x) < DEADZONE_THRESHOLD) accel_x = 0.0;
    if (fabs(accel_y) < DEADZONE_THRESHOLD) accel_y = 0.0;

    // Integrate acceleration to get velocity
    velocity_x += accel_x * dt;
    velocity_y += accel_y * dt;

    // Apply damping to reduce velocity over time
    velocity_x *= DAMPING_FACTOR;
    velocity_y *= DAMPING_FACTOR;

    // Ensure velocity direction is consistent
    if ((velocity_x > 0 && accel_x < 0) || (velocity_x < 0 && accel_x > 0))
        velocity_x += accel_x * dt;
    if ((velocity_y > 0 && accel_y < 0) || (velocity_y < 0 && accel_y > 0))
        velocity_y += accel_y * dt;

    // Map velocity to mouse movement
    int mouse_dx = (int)(velocity_x * SCALE_FACTOR);
    int mouse_dy = (int)(velocity_y * SCALE_FACTOR);

    // Move mouse (you need a function to send this data to the OS)
    move_mouse(mouse_dx, mouse_dy);

    // Log data for debugging
    ESP_LOGI(TAG, "Raw Accel X: %d, Raw Accel Y: %d, Accel X: %.2f, Accel Y: %.2f, velocity_x: %.2f, velocity_y: %.2f",
              raw_accel_x, raw_accel_y, accel_x, accel_y, velocity_x, velocity_y);
}
```

\*pasted an error to ChatGPT\*

The stack dump indicates that there was a call to `ESP\_ERROR\_CHECK` which failed, causing the program to abort. The error code `0xffffffff` (which corresponds to `ESP\_FAIL`) was returned by the `read\_gyro` function, leading to the failure. The relevant part of your code where the error occurred is:

```
ESP_ERROR_CHECK(read_gyro(&x, &y, &z));
```

“it seems it could be this line? why is that esp\_err\_t ret = i2c\_master\_cmd\_begin(I2C\_MASTER\_NUM, cmd, 1000 / portTICK\_PERIOD\_MS);”

```
static const char *TAG = "I2C_Example";

esp_err_t i2c_master_init(void) {
    i2c_config_t conf = {
        .mode = I2C_MODE_MASTER,
        .sda_io_num = I2C_MASTER_SDA_IO,
        .scl_io_num = I2C_MASTER_SCL_IO,
        .sda_pullup_en = GPIO_PULLUP_ENABLE,
        .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) {
        return err;
    }
    return i2c_driver_install(I2C_MASTER_NUM, conf.mode, I2C_MASTER_RX_BUF_DISABLE, I2C_MASTER_TX_BUF_DISABLE, 0);
}

esp_err_t read_gyro(int16_t *x, int16_t *y, int16_t *z) {
    esp_err_t ret;
    uint8_t data[6];

    i2c_cmd_handle_t cmd = i2c_cmd_link_create();
    i2c_master_start(cmd);
    i2c_master_write_byte(cmd, (GYRO_SENSOR_ADDR << 1) | I2C_MASTER_WRITE, true);
    i2c_master_write_byte(cmd, GYRO_REG_X, true);
    i2c_master_start(cmd);
    i2c_master_write_byte(cmd, (GYRO_SENSOR_ADDR << 1) | I2C_MASTER_READ, true);
    i2c_master_read(cmd, data, sizeof(data), I2C_MASTER_LAST_NACK);
    i2c_master_stop(cmd);

    ret = i2c_master_cmd_begin(I2C_MASTER_NUM, cmd, 1000 / portTICK_PERIOD_MS);
    i2c_cmd_link_delete(cmd);

    if (ret != ESP_OK) {
        ESP_LOGE(TAG, "I2C command failed: %s", esp_err_to_name(ret));
        return ret;
    }

    *x = (data[0] << 8) | data[1];
    *y = (data[2] << 8) | data[3];
    *z = (data[4] << 8) | data[5];

    return ESP_OK;
}

void app_main() {
    ESP_ERROR_CHECK(i2c_master_init());

    int16_t x, y, z;

    while (1) {
        esp_err_t err = read_gyro(&x, &y, &z);
        if (err != ESP_OK) {
            ESP_LOGE(TAG, "Failed to read gyro data: %s", esp_err_to_name(err));
        } else {
            ESP_LOGI(TAG, "Gyro X: %d, Gyro Y: %d, Gyro Z: %d", x, y, z);
        }

        vTaskDelay(pdMS_TO_TICKS(1000)); // 1 second delay
    }
}
```

“here is read\_gyro do you see any issues?”

```
i2c_master_start(cmd);
i2c_master_write_byte(cmd, IMU_SENSOR_ADDR << 1 | WRITE_BIT, ACK_CHECK_EN);
i2c_master_write_byte(cmd, 0x0B, ACK_CHECK_EN);
i2c_master_start(cmd);
i2c_master_write_byte(cmd, IMU_SENSOR_ADDR << 1 | READ_BIT, ACK_CHECK_EN);
i2c_master_read(cmd, data, sizeof(data), I2C_MASTER_LAST_NACK);
i2c_master_stop(cmd);

ret = i2c_master_cmd_begin(I2C_MASTER_NUM, cmd, 1000 / portTICK_PERIOD_MS);
i2c_cmd_link_delete(cmd);

if (ret != ESP_OK) {
    ESP_LOGE(TAG, "I2C command failed with error: %s", esp_err_to_name(ret));
    return ret;
}

*x = (int16_t)((data[0] << 8) | data[1]);
*y = (int16_t)((data[2] << 8) | data[3]);
*z = (int16_t)((data[4] << 8) | data[5]);

ESP_LOGI(TAG, "Read gyro values: x=%d, y=%d, z=%d", *x, *y, *z);
return ESP_OK;
}

void app_main() {
    ESP_ERROR_CHECK(i2c_master_init());

    int16_t x, y, z;

    while (1) {
        esp_err_t err = read_gyro(&x, &y, &z);
        if (err != ESP_OK) {
            ESP_LOGE(TAG, "Failed to read gyro data: %s", esp_err_to_name(err));
        }

        vTaskDelay(pdMS_TO_TICKS(1000)); // 1 second delay
    }
}

esp_err_t i2c_master_init() {
    i2c_config_t conf = {
        .mode = I2C_MODE_MASTER,
        .sda_io_num = I2C_MASTER_SDA_IO,
        .scl_io_num = I2C_MASTER_SCL_IO,
        .sda_pullup_en = GPIO_PULLUP_ENABLE,
        .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, "I2C param config failed with error: %s", esp_err_to_name(err));
        return err;
    }

    err = i2c_driver_install(I2C_MASTER_NUM, conf.mode, I2C_MASTER_RX_BUF_DISABLE, I2C_MAST
    if (err != ESP_OK) {
        ESP_LOGE(TAG, "I2C driver install failed with error: %s", esp_err_to_name(err));
    }

    return err;
}
```

“how can i measure how long its been since the last loop iteration”

### Using `esp\_timer\_get\_time`

If you need more precision, you can use `esp\_timer\_get\_time` which returns the time in microseconds:

```
c Copy code

#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_timer.h"
#include "esp_log.h"

#define TAG "TimeMeasure"

void app_main() {
    int64_t last_time = esp_timer_get_time();
    int64_t current_time;
    int64_t elapsed_time;

    while (1) {
        // Do your work here
        ESP_LOGI(TAG, "Doing work...");

        // Measure the elapsed time
        current_time = esp_timer_get_time();
        elapsed_time = current_time - last_time;
        last_time = current_time;

        // Convert microseconds to milliseconds
        int64_t elapsed_time_ms = elapsed_time / 1000;

        ESP_LOGI(TAG, "Time since last loop: %lld ms", elapsed_time_ms);

        // Sleep for 1 second
        vTaskDelay(pdMS_TO_TICKS(1000));
    }
}
```