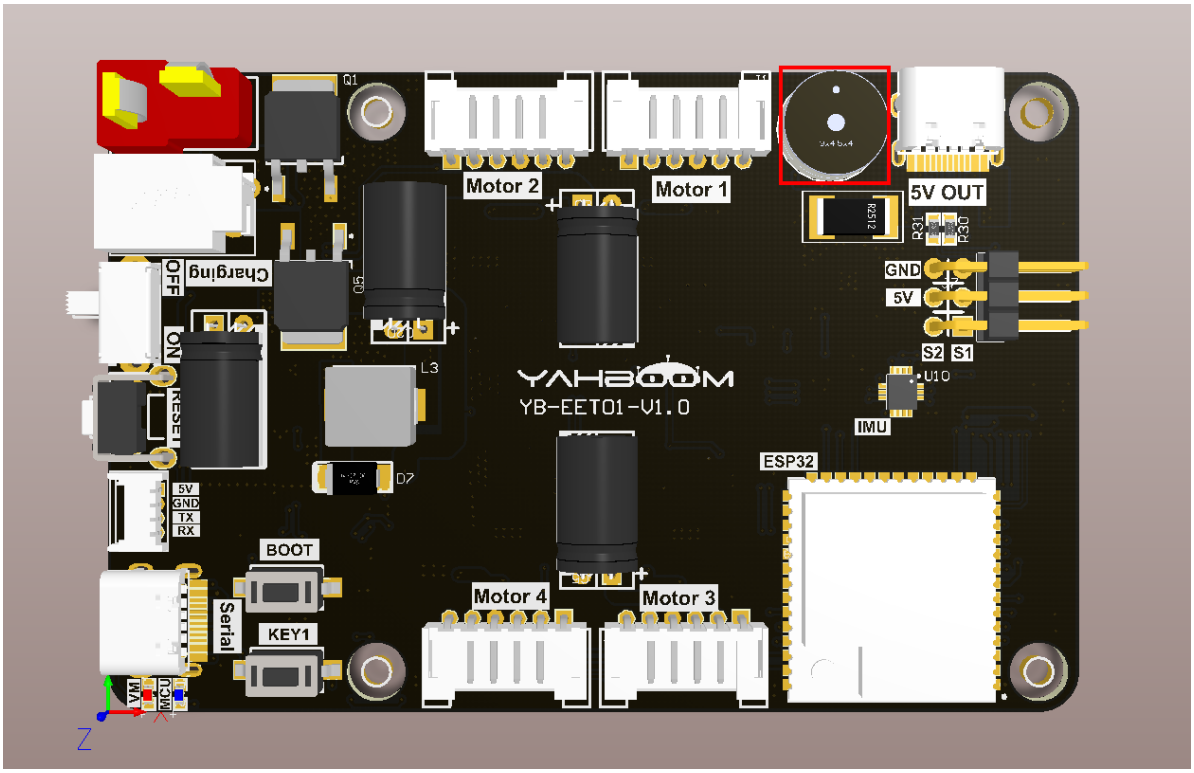
Drive the buzzer

1. Experimental purpose

Drive the active buzzer on the microROS control board to sound every 500 milliseconds.

2. Hardware connection

As shown in the figure below, the buzzer is an onboard component, so there is no need to connect other external devices. You only need to connect the type-C data cable to the computer and the microROS control board as a firmware flash function.



The buzzer on the microROS control board is an active buzzer. When the GPIO pin level is high level, the buzzer is turned on, and when the GPIO pin level is low level, the buzzer is turned off.

3. Core code analysis

The virtual machine path corresponding to the program source code is as follows

```
~/esp/Samples/esp32_samples/beep
```

Initialize the buzzer's GPIO, where BEEP_GPIO corresponds to GPIO46 of the hardware circuit, and the GPIO mode is set to output mode.

```
static void Beep_GPIO_Init(void)
{
    // zero-initialize the config structure.
    gpio_config_t io_conf = {};
    //disable interrupt 禁用中断
    io_conf.intr_type = GPIO_INTR_DISABLE;
    //set as output mode 设置为输出模式
    io_conf.mode = GPIO_MODE_OUTPUT;
    //bit mask of the pins that you want to set 引脚编号设置
    io_conf.pin_bit_mask = (1ULL<<BEEP_GPIO);
    //disable pull-down mode 禁用下拉
    io_conf.pull_down_en = 0;
    //disable pull-up mode 禁用上拉
    io_conf.pull_up_en = 0;
    //configure GPIO with the given settings 配置GPIO口
    gpio_config(&io_conf);
    // 关闭蜂鸣器
    Beep_Off();
}
```

Turn on the buzzer and the buzzer will sound continuously.

```
void Beep_On(void)
{
    beep_state = BEEP_STATE_ON_ALWAYS;
    beep_on_time = 0;
    gpio_set_level(BEEP_GPIO, BEEP_ACTIVE_LEVEL);
}
```

Turn off the buzzer.

```
void Beep_Off(void)
{
    beep_state = BEEP_STATE_OFF;
    beep_on_time = 0;
    gpio_set_level(BEEP_GPIO, !BEEP_ACTIVE_LEVEL);
}
```

In order to automatically turn off the buzzer when the buzzer times out, you need to start the buzzer task when initializing the buzzer to manage the buzzer whistle time.

```
static void Beep_Task(void *arg)
{
    ESP_LOGI(TAG, "Start Beep_Task with core:%d", xPortGetCoreID());
    while (1)
    {
        Beep_Handle();

        vTaskDelay(pdMS_TO_TICKS(10));
    }

    vTaskDelete(NULL);
}
```

The main task of the Beep_Handle function is to automatically reduce the value of beep_on_time in the BEEP_STATE_ON_DELAY state until it equals 0 and automatically turn off the buzzer.

```
void Beep_Handle(void)
{
    if (beep_state == BEEP_STATE_ON_DELAY)
    {
        if (beep_on_time > 0)
        {
            beep_on_time--;
        }
        else
        {
            Beep_Off();
            beep_state = BEEP_STATE_OFF;
        }
    }
}
```

Set the buzzer turn-on time. It will turn off when time=0. It will always sound when time=1. If time>=10, it will turn off automatically after a delay of xx milliseconds.

```
void Beep_On_Time(uint16_t time)
{
    if (time == 1)
    {
        beep_state = BEEP_STATE_ON_ALWAYS;
        beep_on_time = 0;
        Beep_On();
    }
    else if (time == 0)
    {
        beep_state = BEEP_STATE_OFF;
        beep_on_time = 0;
        Beep_Off();
    }
    else
    {
        if (time < 10) time = 10;
        if (time > 10000) time = 10000;
        beep_state = BEEP_STATE_ON_DELAY;
        beep_on_time = (time / 10);
        gpio_set_level(BEEP_GPIO, BEEP_ACTIVE_LEVEL);
```

```
    }
}
```

Call the Beep_Init function in app_main to initialize the buzzer, and set the buzzer to sound for 500 milliseconds and then automatically turn off in the loop (every 1000 milliseconds).

```
void app_main(void)
{
    printf("hello yahboom\n");
    ESP_LOGI(TAG, "Nice to meet you!");
    Beep_Init();
    vTaskDelay(pdMS_TO_TICKS(1000));

    // 每1秒响一次蜂鸣器 The buzzer sounds every 1 second
    while(1)
    {
        Beep_On_Time(500);
        vTaskDelay(pdMS_TO_TICKS(1000));
    }
}
```

4. Compile, download and flash firmware

Use a Type-C data cable to connect the virtual machine/computer and the microROS control board. If the system pops up, choose to connect to the virtual machine.

Activate the ESP-IDF development environment. Note that every time you open a new terminal, you need to activate the ESP-IDF development environment before compiling the firmware.

```
source ~/esp/esp-idf/export.sh
```

Enter the project directory

```
cd ~/esp/Samples/esp32_samples/beep
```

Compile, flash, and open the serial port simulator

```
idf.py build flash monitor
```

If you need to exit the serial port simulator, press **Ctrl+]**.

5. Experimental results

The serial port simulator prints the greeting "hello yahboom". At this time, we can hear a beep every 500 milliseconds.

```
I (313) main_task: Calling app_main()
hello yahboom
I (318) MAIN: Nice to meet you!
I (322) gpio: GPIO[46]| InputEn: 0| OutputEn: 1| OpenDrain: 0| Pullup: 0| Pulldown: 0|
 Intr:0
I (331) BEEP: Start Beep_Task with core:1
```