读取 DHT11 温湿度传感器数据

淘宝店铺：

PC 端：

http://n-xytrt8gqu585po94mwj5atokcyd4.taobao.com/index.htm

手机端：

https://shop.m.taobao.com/shop/shop_index.htm?sellerId=755668508&shopId=104493595&inShopPageId=423890608&pathInfo=shop/index2

资料下载地址：

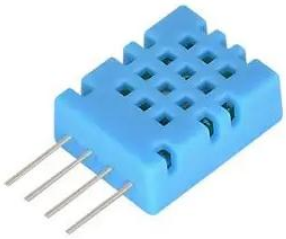
链接：https://pan.baidu.com/s/1kCjD8yktZECSGmHomx_veg?pwd=q8er
提取码：q8er

源码下载地址：

https://gitee.com/vi-iot/esp32-board.git

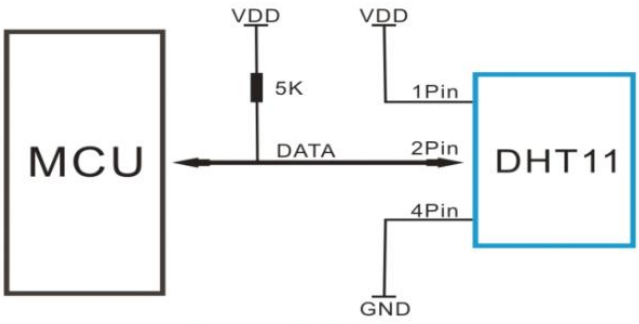
一、DHT11 介绍

DHT11 是一款常用的数字温湿度传感器，适合于需要测量温度和湿度的基本应用场景。其测量范围为湿度为 5%到 95%RH，温度范围为-20℃到 60℃，湿度测量精度为±5%RH，温度测量精度为±2℃,采用单总线通信协议，通过一个数据引脚完成输入输出双向传输。



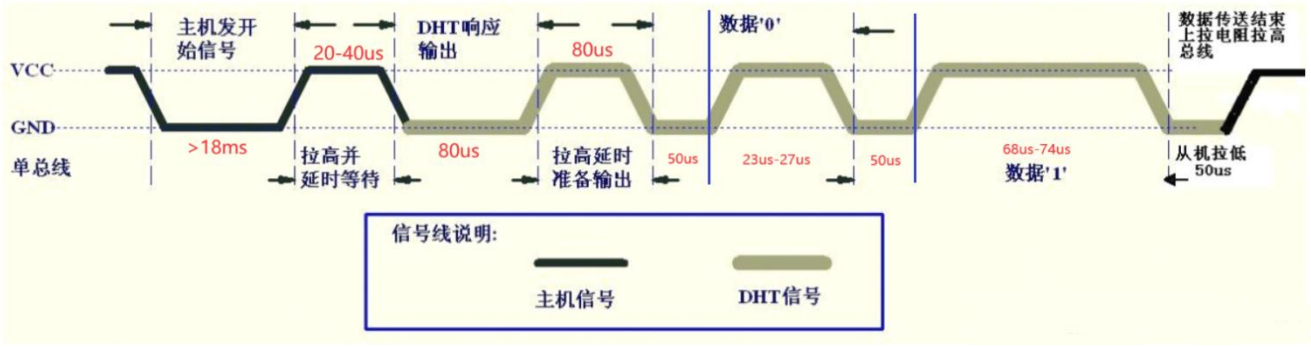
DHT11 实物图

DHT11 传感器有 4 个管脚，其中有一个管脚不用接，另外就是 VCC，SIG，GND，可能大家自行买的 DHT11 是带了转接板的，因此需要仔细确认管脚顺序，DHT11 硬件连接简单，与 MCU 连接如下所示，只需要在信号脚上接一个 5K 的上拉电阻即可。供电范围是 3-5.5V，超过这个范围可能会烧毁。



典型应用电路

DHT11 的通信包数据长度为 40 个字节。组成如下：8bit 湿度整数数据+8bit 湿度小数数据+8bit 温度整数数据+8bit 温度小数数据+8bit 校验和，湿度小数数据是 0 不用关注，数据传输格式为高位在前，低位在后，校验和是前 4 个字节相加。对于 DHT11 的数据传输时序非常严格，并非是低电平为 0，高电平为 1，和上一课 WS2812 类似，对于逻辑 0 和逻辑 1，均是通过不同时间的高电平占比来表示的。具体时序逻辑如下图



大致流程就是，主机发起开始信号（>18ms 的低电平），拉高 40us 等待。然后 DHT11 会拉低 80us 表示响应信号，然后拉高 80us 表示开始传输数据。每一位数据前都有 50us 的低电平，紧接着如果是数据 0，高电平持续时间大概在 23-27us，如果是数据 1，高电平大概是 68us-74us，全部数据传输完成后，DHT11 拉低 50us，表示传输完毕，然后主机拉高表示

通信结束。通过梳理这些时序，我们发现使用 ESP32 的 RMT 模块是可以拿来驱动 DHT11 的，不过这次我们要用到的是 RMT 模块的接收。具体代码请看下节。

二、ESP32 驱动 DHT11

在本课程配套的开发板上，DHT11 的信号脚是接到 GPIO25 上，大家也可以接到其他 GPIO 上，不过要改一下代码中的 GPIO 脚号。代码在 esp32-board/dht11 中，我们先来分析几个关键函数。

```
// DHT11 初始化
void DHT11_Init(uint8_t dht11_pin)
{
    DHT11_PIN = dht11_pin;

    rmt_rx_channel_config_t rx_chan_config = {
        .clk_src = RMT_CLK_SRC_APB,   // 选择时钟源
        .resolution_hz = 1000 * 1000,     // 1 MHz 滴答分辨率，即 1 滴答 = 1
μs
        .mem_block_symbols = 64,          // 内存块大小，即 64 * 4 = 256 字
节
        .gpio_num = dht11_pin,            // GPIO 编号
        .flags.invert_in = false,         // 不反转输入信号
        .flags.with_dma = false,          // 不需要 DMA 后端
    };
    //创建 rmt 接收通道
    ESP_ERROR_CHECK(rmt_new_rx_channel(&rx_chan_config,
&rx_chan_handle));

    //新建接收数据队列
    rx_receive_queue = xQueueCreate(20,
sizeof(rmt_rx_done_event_data_t));
    assert(rx_receive_queue);

    //注册接收完成回调函数
     ESP_LOGI(TAG, "register RX done callback");
    rmt_rx_event_callbacks_t cbs = {
        .on_recv_done = example_rmt_rx_done_callback,
    };
    ESP_ERROR_CHECK(rmt_rx_register_event_callbacks(rx_chan_handle, &cbs,
rx_receive_queue));
    //使能 RMT 接收通道
    ESP_ERROR_CHECK(rmt_enable(rx_chan_handle));
}
```

初始化中，首先是需要定义 rmt_rx_channel_config_t 内容，大致就是选择时钟源、时钟分辨率、底层内存块大小（一般设置为 64）、GPIO 编号等内容，时钟频率这里设定为 1M，刚好是 1us，因此低于 1us 的脉冲无法分辨。而 DHT11 时序中没有低于 10us 的脉冲，因此

绰绰有余。

然后创建一个接收队列，在 main 函数主循环中实时接收数据。队列数据内容是 rmt_rx_done_event_data_t，里面包含了 RMT 数据（RMT 符号）信息。

另外一个重要的点是设置 RMT 接收完成回调函数，当 ESP32 检测到 RMT 接收完成后，就会调用这个接收函数，通知我们数据接收完毕，我们看一下这个回调函数的实现。

```
//接收完成回调函数
static bool example_rmt_rx_done_callback(rmt_channel_handle_t channel,
const rmt_rx_done_event_data_t *edata, void *user_data)
{
    BaseType_t high_task_wakeup = pdFALSE;
    QueueHandle_t rx_receive_queue = (QueueHandle_t)user_data;
    // send the received RMT symbols to the parser task
    xQueueSendFromISR(rx_receive_queue, edata, &high_task_wakeup);
    return high_task_wakeup == pdTRUE;
}
```

回调函数中 edata 参数包含了 RMT 数据信息，我们把这个参数发到刚才创建的队列中，然后就返回了，xQueueSendFromISR 是 xQueueSend 的中断函数版本，只能在中断函数中调用。

然后看一下，我们是如何获取 DHT11 数据的，请看如下代码

```
// 使用 RMT 接收 DHT11 数据
int DHT11_StartGet(int *temp_x10, int *humidity)
{
    //发送 20ms 开始信号脉冲启动 DHT11 单总线
    gpio_set_level(DHT11_PIN, 1);
    gpio_set_direction(DHT11_PIN, GPIO_MODE_OUTPUT);
    ets_delay_us(1000);
    gpio_set_level(DHT11_PIN, 0);
    ets_delay_us(20000);
    //拉高 20us
    gpio_set_level(DHT11_PIN, 1);
    ets_delay_us(20);
    //信号线设置为输入准备接收数据
    gpio_set_direction(DHT11_PIN, GPIO_MODE_INPUT);
    gpio_set_pull_mode(DHT11_PIN,GPIO_PULLUP_ONLY);

    //启动 RMT 接收器以获取数据
    rmt_receive_config_t receive_config = {
        .signal_range_min_ns = 1000,     //最小脉冲宽度(1us),信号长度小于这
个值，视为干扰
        .signal_range_max_ns = 200*1000,    //最大脉冲宽度(200us)，信号长度
大于这个值，视为结束信号
    };

    rmt_symbol_word_t raw_symbols[64];  //接收缓存
```

```
    rmt_rx_done_event_data_t rx_data;   //实际接收到的数据
    ESP_ERROR_CHECK(rmt_receive(rx_chan_handle, raw_symbols,
sizeof(raw_symbols), &receive_config));
    // wait for RX done signal
    if (xQueueReceive(rx_receive_queue, &rx_data, pdMS_TO_TICKS(1000)) ==
pdTRUE) {
        // parse the receive symbols and print the result
        return parse_items(rx_data.received_symbols,
rx_data.num_symbols,humidity, temp_x10);
    }
    return 0;
}
```

前面的启动时序代码就不做过多说明了，大家对照着时序图和注释看就行，有几个点需要说明一下，rmt_receive_config_t 这个结构体定义了两种脉冲，第一种是最小脉冲信号，低于这个脉冲宽度，RMT 模块认为是干扰信号，第二种是最大脉宽，大于此脉宽的视为 RMT 接收结束信号，在这里分别设置 1us 以及 200us。

调用 rmt_receive 函数，把过滤脉冲配置、接收的 RMT 符号缓存传入，在底层即开始接收到，然后我们从队列中接收从回调函数发来的数据即可。

parse_items 函数是解析 RMT 符号，把它转化为我们程序中可以识别的数据。接下来看下这个函数的实现

```
// 将 RMT 读取到的脉冲数据处理为温度和湿度(rmt_symbol_word_t 成为 RMT 符号)
static int parse_items(rmt_symbol_word_t *item, int item_num, int *humidity,
int *temp_x10)
{
    int i = 0;
    unsigned int rh = 0, temp = 0, checksum = 0;
    if (item_num < 42){                 // 检查是否有足够的脉冲数
        ESP_LOGI(TAG, "item_num < 42  %d",item_num);
        return 0;
    }
    item++;                             // 跳过开始信号脉冲

    for (i = 0; i < 16; i++, item++)    // 提取湿度数据
    {
        uint16_t duration = 0;
        if(item->level0)
            duration = item->duration0;
        else
            duration = item->duration1;
        rh = (rh << 1) + (duration < 35 ? 0 : 1);
    }
    for (i = 0; i < 16; i++, item++)    // 提取温度数据
    {
        uint16_t duration = 0;
```

```
        if(item->level0)
            duration = item->duration0;
        else
            duration = item->duration1;
        temp = (temp << 1) + (duration < 35 ? 0 : 1);
    }

    for (i = 0; i < 8; i++, item++){    // 提取校验数据

        uint16_t duration = 0;
        if(item->level0)
            duration = item->duration0;
        else
            duration = item->duration1;
        checksum = (checksum << 1) + (duration < 35 ? 0 : 1);
    }
    // 检查校验
    if ((((temp >> 8) + temp + (rh >> 8) + rh) & 0xFF) != checksum){
        ESP_LOGI(TAG, "Checksum failure %4X %4X %2X\n", temp, rh, checksum);
        return 0;
    }
    // 返回数据
    *humidity = rh >> 8;
    *temp_x10 = (temp >> 8) * 10 + (temp & 0xFF);
    return 1;
}
```

上一课在介绍驱动 WS2812 时，我们提到了 RMT 符号，这个是 RMT 模块用于描述 RMT 数据的符号。其定义如下

```
/**
 * @brief The layout of RMT symbol stored in memory, which is decided by
the hardware design
 */
typedef union {
    struct {
        uint16_t duration0 : 15; /*!< Duration of level0 */
        uint16_t level0 : 1;     /*!< Level of the first part */
        uint16_t duration1 : 15; /*!< Duration of level1 */
        uint16_t level1 : 1;     /*!< Level of the second part */
    };
    uint32_t val; /*!< Equivalent unsigned value for the RMT symbol */
} rmt_symbol_word_t;
```

由上可以知道，本质上 RMT 符号是一种描述，它描述了一个数据位高电平时间和低电平时间分别占用是多少，duration 成员就是电平时间，level 表示电平。DHT11 总共包含 40 个数据位，再加上起始符号和结束符号，一共有 42 个 RMT 符号。所以在 parse_items 函数

中我们首先就判断了 RMT 符号个数，然后通过 item++跳过起始符号，然后开始的是湿度数据，湿度数据整数+小数一共 16 位。通过 DHT11 时序图知道，我们只需要判断高电平的持续时间，就能分辨出这个 RMT 符号表示的是数据 0 还是 1。在 for 循环中，我们先判断 RMT 符号中的结构体 level 成员，看看 level0 还是 level1 是 1，如果 level0 是 1，我们就取 duration0 作为高电平的持续时间，这样最保险，然后我判断持续时间是否大于 35us，大于 35us 就是数据 1，否则就是数据 0，经过循环和左移操作后，我们就能把数据组合起来，温度值、校验也是同样处理方式。最后是校验，我们把前四位数据加起来，看看是否与校验位相等即可（只取低 8 位）

在 app_main()中，我们循环调用 DHT11_StartGet 函数，可以不断的获取到温湿度数据。再次提醒代码在 esp32-board/dht11 中，如果大家有什么疑问，建议显示反复的查看这份代码，加深理解，esp-idf 的世界非常丰富。