

# **BPHero UWB 用户手册**

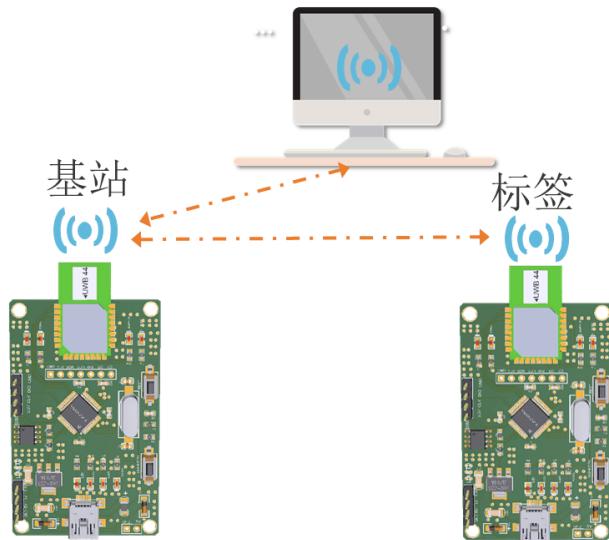
# 内容

场景介绍.....	3
硬件介绍.....	6
单模块介绍.....	6
其它器件介绍.....	7
资料介绍.....	8
开箱检查.....	10
发货数量.....	10
检查模块运行状态.....	10
安装软件.....	11
安装STM32 ST-LINK Utility.....	11
安装Stlink V2驱动.....	12
安装串口驱动.....	12
安装MDK.....	13
下载现有HEX.....	16
自行编译HEX.....	17
查看串口信息.....	19
放置基站.....	23
打开定位软件.....	23
代码介绍.....	24
标签基站的选择.....	24
跟踪小车.....	25
测距与定位的串口数据.....	30
均值滤波.....	30
售后与保修政策.....	31

# 场景介绍

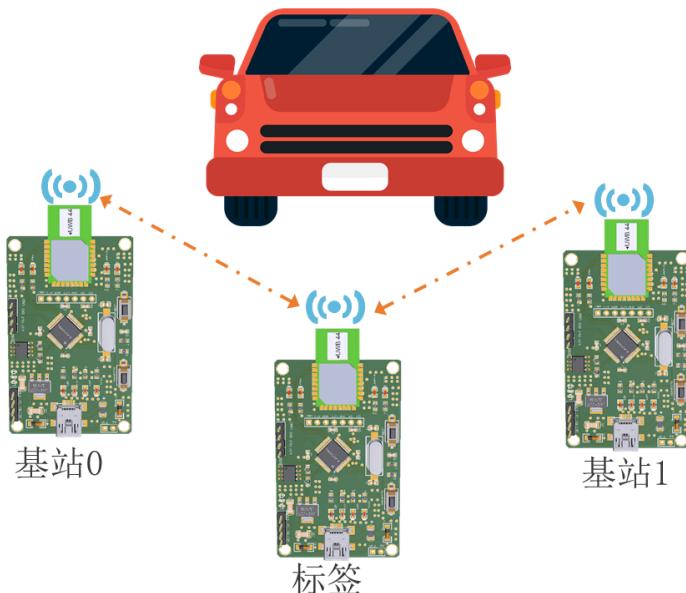
## 测距场景

基站0与标签通过收、发信息之间的传播时间计算距离。



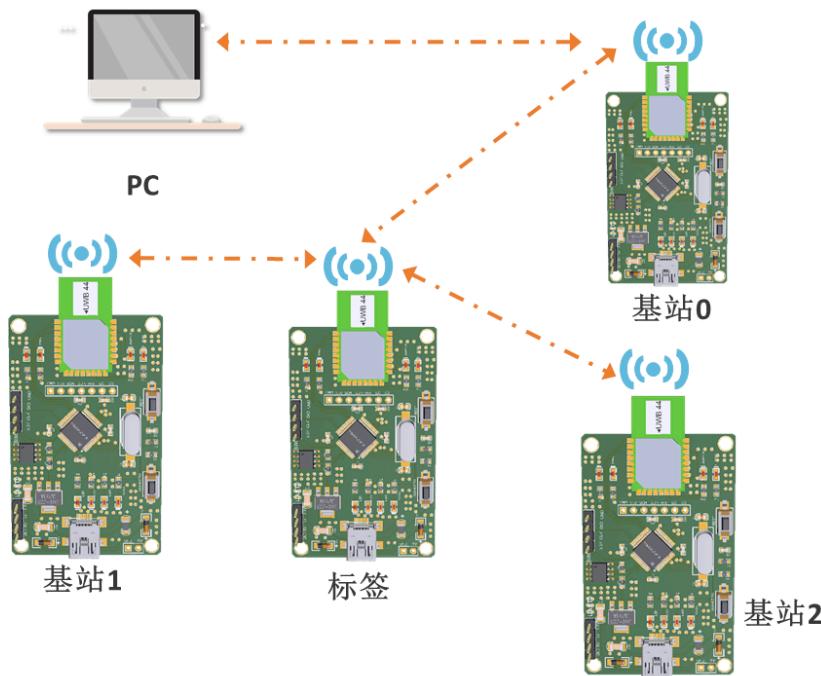
## 跟随小车场景

基站0与基站1分别放到小车头部的两端，由标签牵引小车前进的方向。可设置小车的启动距离，当大于该距离时，小车启动；小于该距离，则停止。行动过程中，采用余弦定理，计算基站0与标签的内角度，根据该角度和基站0与标签的距离控制小车移动。



## 典型定位场景

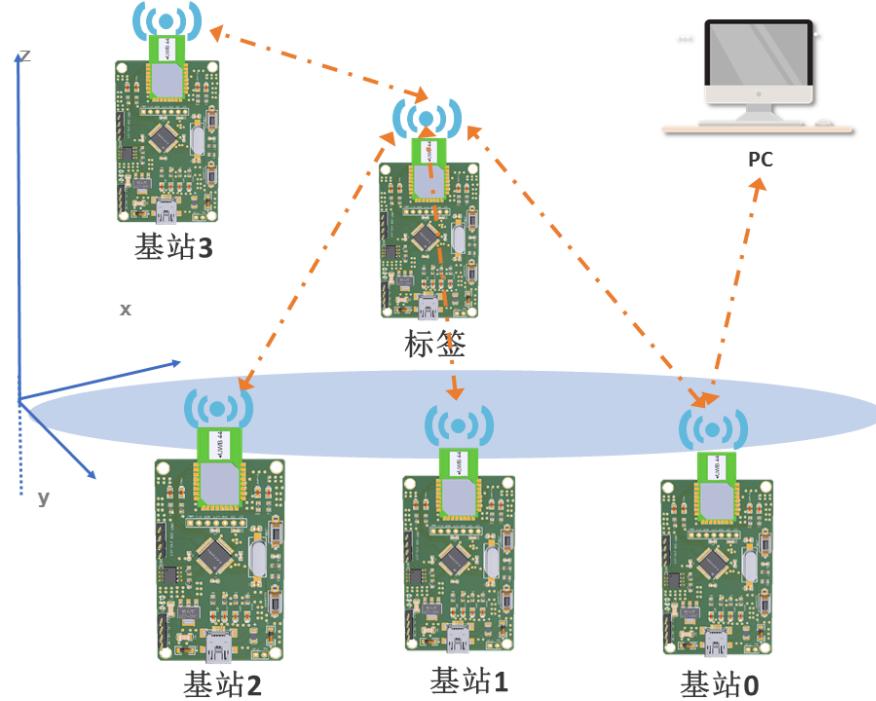
基站0、基站1、基站2分别放置到同一平面，且都与标签通信。



### 其他定位场景

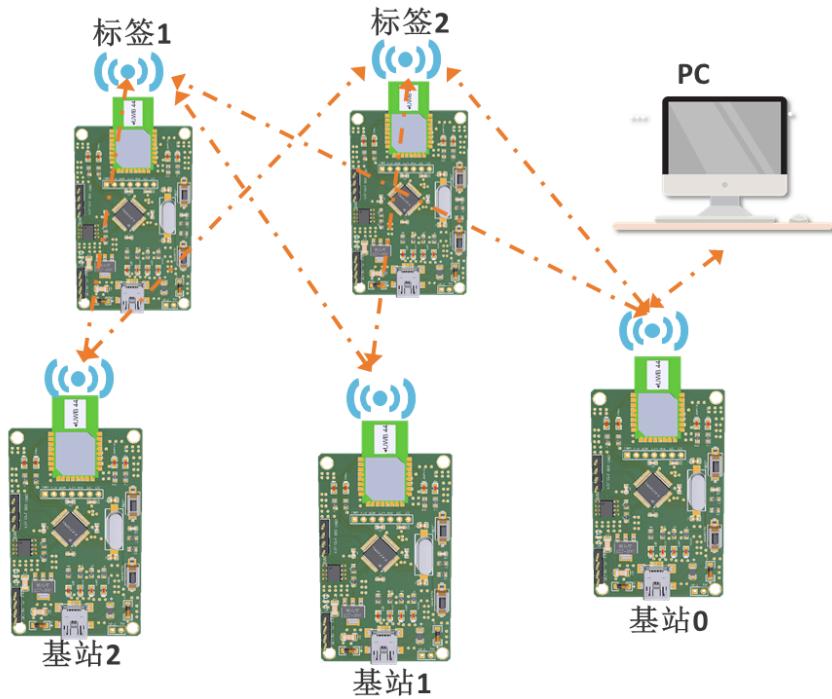
#### 三维实例

基站0、基站1、基站2分别放置到同一平面，基站3放置Z轴，所有基站与标签通信。



#### 多标签实例

多个标签同时与基站通信，基站可二维、三维。

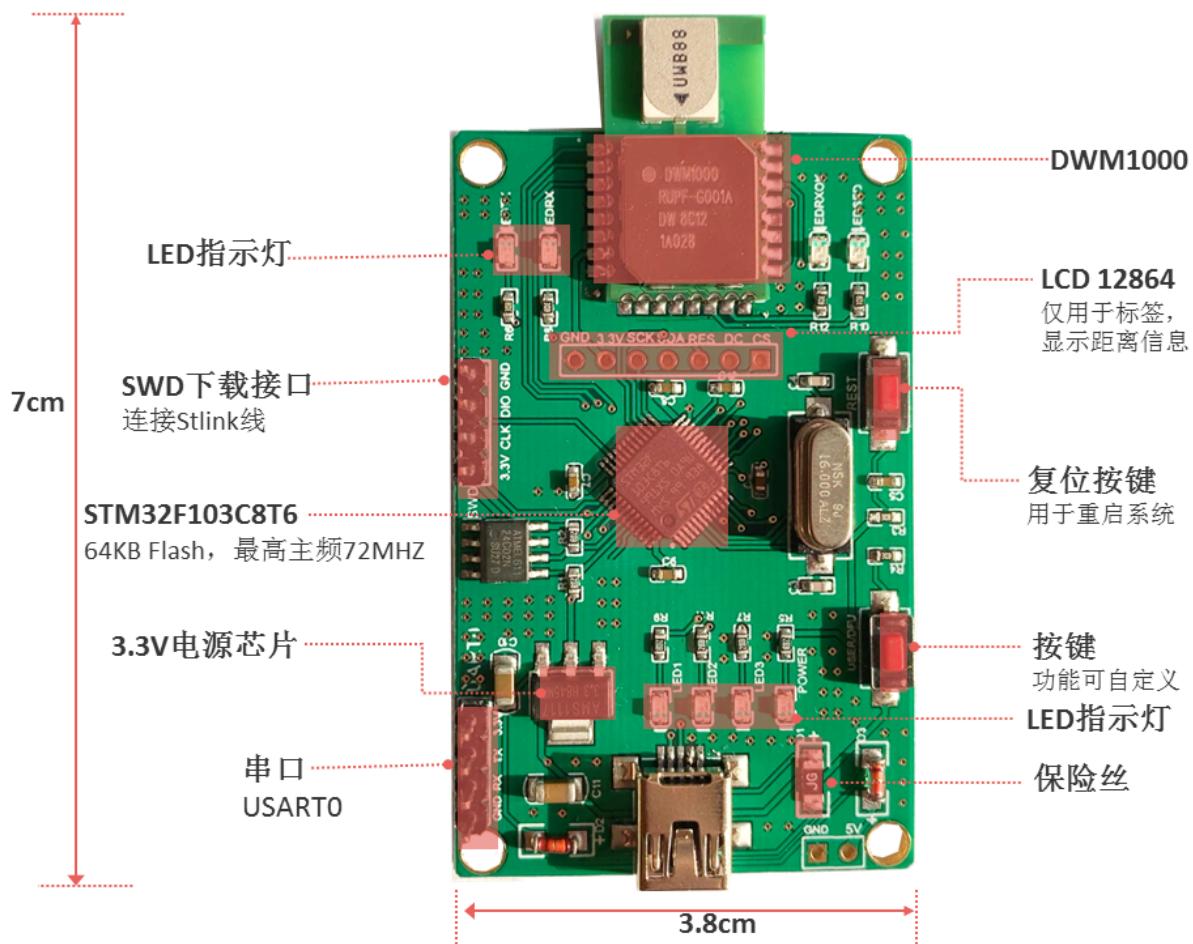


以上所有套件都是标签显示与各基站的距离信息，基站0将距离信息传达到PC中，PC中的上位机计算坐标信息。

# 硬件介绍

## 单模块介绍

### 外观



### 性能

分类	参数
电源	5v
额定功率	500mW
工作温度	-20°C ~ 80°C
工作频率	3hz ~ 4hz , 最高40hz
处理器的外部时钟源	16Mhz

分类	参数
测距精度	5cm
二维定位精度	10cm
Z轴定位精度	20cm
通信距离	50m
标签最大个数	6个
基站最大个数	4个
定位代码开发软件	MDK5
定位代码	C
上位机语言	C++

## 其它器件介绍



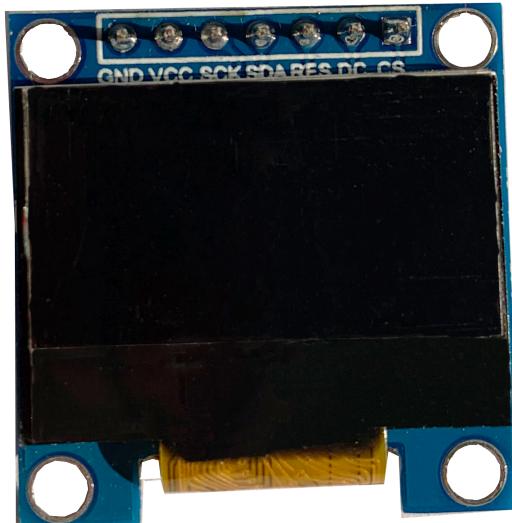
StlinkV2 , 颜色随机



USB转串口线



USB转接头



LCD 12864

## 资料介绍

---

不提供资料光碟，默认所有资料通过百度云共享方式分享。若不能使用百度云，可单独购买U盘(16GB Kingston USB3.0)，售价50 RMB。



资料目录	说明
上位机软件以及源码	官方上位机源码，在官方基础上稍作修改，已编译好，可用于显示定位。
定位源码-STM32	BPhero-UWB定位C源码，可用MDK打开修改代码。该文件会持续更新修正，请关注 <a href="#">论坛</a> 。
其它代码收集	官方DWM1000 API实例，以及官方定位源码(CoIED工程)。
开发板原理图	BPhero-UWB模块原理图。
STM32 基础资料	STM32芯片数据手册。
测试HEX文件	包含标签和基站的HEX。
Keil MDK以及其他工具	MDK安装包、Stlink驱动、串口驱动以及串口助手。
官方资料汇总	官方DWM1000资料汇总。

# 开箱检查

---

## 发货数量

收到快递时，若发现电路板损坏或者缺件，请拒收。如后期发现问题请联系客服。

名称	数量	说明
BPHero-DWM 1000模块	<ul style="list-style-type: none"> <li>• 测距：2个</li> <li>• 跟随小车：3个</li> <li>• 定位：4个</li> <li>• 其他为实际购买情况</li> </ul>	-
USB转接头	与BPHero-DWM 1000模块数量一致	一个模块一个，用于为模块充电。
LCD 12864	1个	显示定位距离，插在标签上。
USB转串口线	1个	把距离信息通过串口传输到电脑上。
StlinkV2	1个	为STM32F03C8T6主控芯片下载程序。
定位资料	1份	-

## 检查模块运行状态

设备发货前，已下载HEX程序，可使用5V通过USB接口为设备供电，请不要通过串口为模块充电。

检查以下内容是否正常，如果异常，请参考[查看串口信息](#) on page 19，查看串口信息是否正常，并联系客服。

### 仅标签上单

检查项	说明
标签指示灯	通信指示灯：左上角（UWB模块左侧）2个指示灯闪烁。 电源指示灯：右下角（靠近按键）1个指示灯闪烁。
标签显示屏	51UWB Node： MASTER TAG www.51uwb.cn

### 仅基站上单

检查项	说明

基站指示灯

通信指示灯：左上角（UWB模块左侧）1个指示灯闪烁。

电源指示灯：右下角（靠近按键）1个指示灯闪烁。

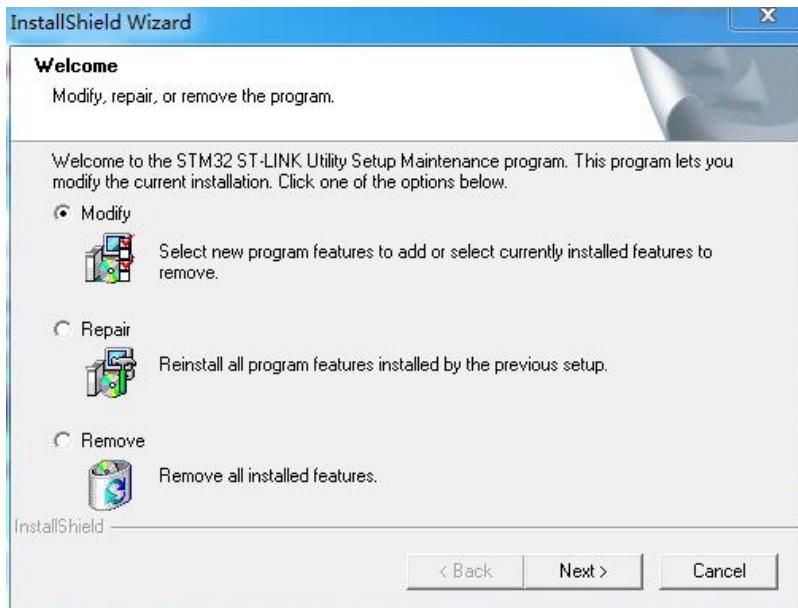
基站、标签都上单

检查项	说明
基站、标签指示灯	通信指示灯：左上角（UWB模块左侧）2个指示灯闪烁。 电源指示灯：右下角（靠近按键）1个指示灯闪烁。
标签显示屏	51UWB Node： an0: mm an1: mm ..... m表示实际的距离，显示个数与基站个数有关。

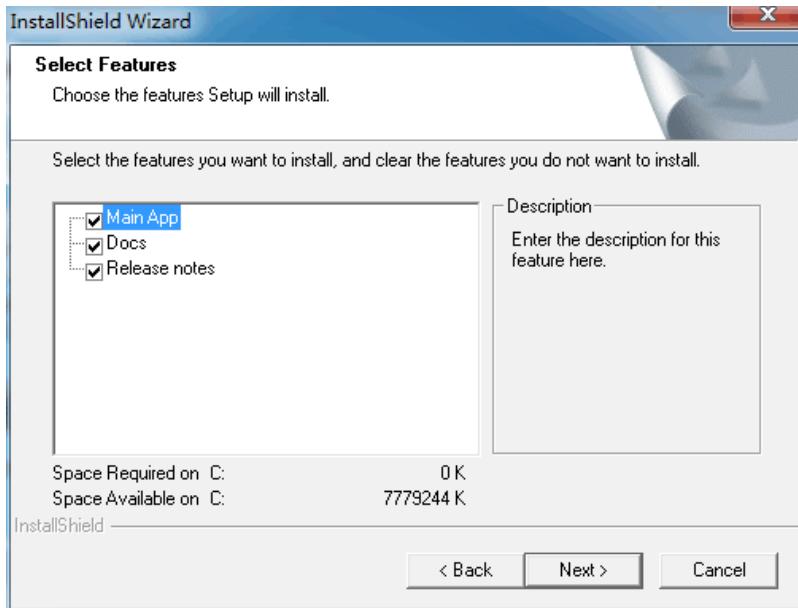
## 安装软件

### 安装STM32 ST-LINK Utility

- 在“Keil MDK以及其它工具”目录下，解压STM32 ST-LINK Utility，单击解压后的setup文件。
- 选择“Modify”，单击“Next”。



3. 按照界面默认设置，单击“Next”，按照界面提示进行安装。



4. 安装完成之后，桌面图标如下所示。



## 安装Stlink V2驱动

1. 在如下目录中，解压红框文件，获取安装文件。

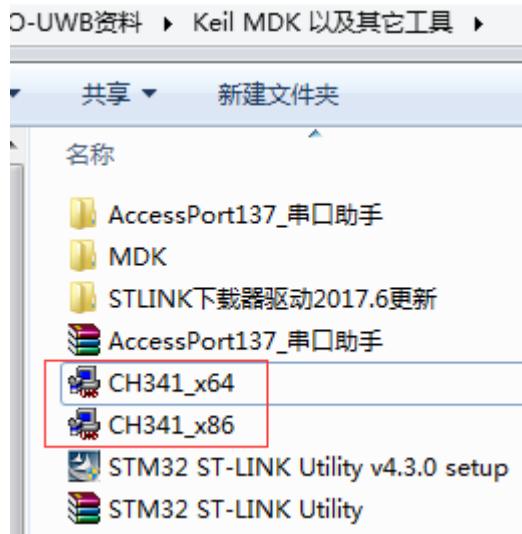
< > □ 我的网盘 > ... DWM1000 > Keil MDK 以及其它工具 > STLINK下载器驱动2017.6更新 >

□ 文件名	修改时间	大小
□  ST_Link驱动en.stsw-link009.zip	2020-04-12 21:50	5.07MB
□  STM32 ST-LINK Utility v3.9.0.zip	2020-04-12 21:45	18.22MB
□  STlink驱动官网链接.txt	2020-04-12 21:49	59B
□  en.DM00236638.pdf	2018-11-12 22:51	111KB
□  en.DM00217720.pdf	2018-11-12 22:51	99KB

2. 按照界面提示进行安装。

## 安装串口驱动

在如下路径，根据电脑位数选择驱动，按照界面提示进行安装。



## 安装MDK

---

1. 在如下路径，使用管理员权限打开红框内容。

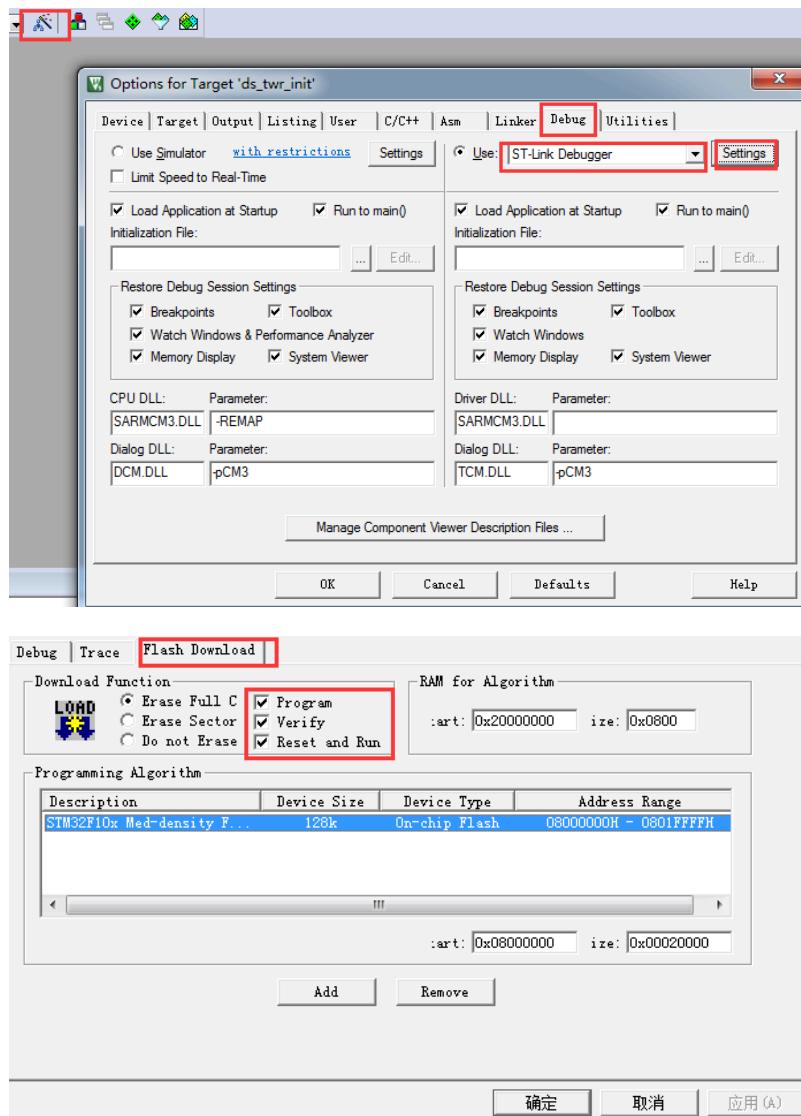


2. 按照界面提示进行安装，安装完成之后桌面图标如下所示，使用管理员权限打开MDK。



3. 在MDK中，按照下图配置MDK。

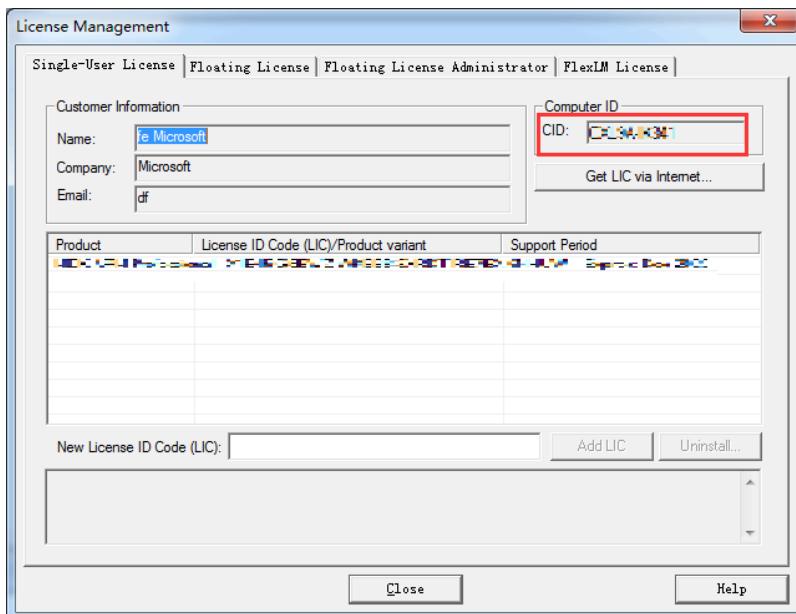
MDK有多种方法下载程序，这里仅介绍STLink方式。



根据芯片类型设置，STM32F103C8T6属于Medium，如果需要修改，请按界面提示“Remove”或“Add”。

#### 4. 破解。

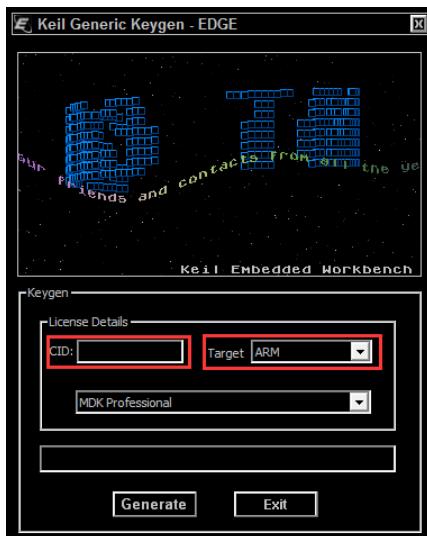
a) 在MDK中，单击File > License Management，拷贝CID的内容。



b) 在如下路径，使用管理员权限打开破解软件。



c) 在CID中粘贴上述拷贝的CID内容，Target选择“ARM”，单击“Generate”。



- d) 将生成的密钥拷贝到上述MDK中的“New License ID Code”，并单击“Add LIC”，按照界面提示加载License。

## 下载现有HEX

1. 如下所示，用Stlink线将BPhero-UWB模块通过Stlink接口与PC相连。

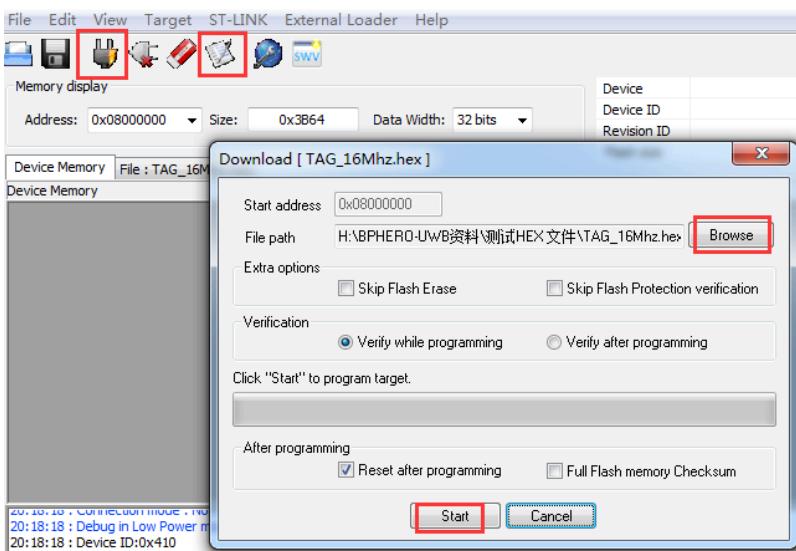


注意：

请参考截图或[单模块介绍](#) on page 6，确认Stlink接口的位置，不要连接到串口接口上，串口无法为设备供电。



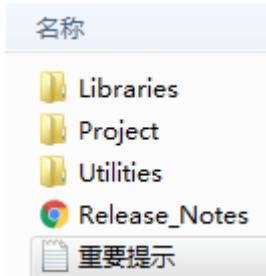
2. 打开STM32 ST-Link Utility，按如下所示，选择带TAG的HEX。



3. 请参考[检查模块运行状态](#) on page 10，确认标签状态。
4. 重复上述方法分别给基站烧录带ANTHOR的HEX，参考[检查模块运行状态](#) on page 10，确认基站指示灯状态。

# 自行编译HEX

1. 使用前，建议备份“定位源码-STM32”。
2. 解压“定位源码-STM32”，解压后如下所示。



3. 在“Project\ds\_twr\_init\MDK-ARM”目录下，使用MDK打开Project项目。

名称	修改日期	类型	大小
DebugConfig	2018/10/28 14:39	文件夹	
RTE	2018/11/11 16:00	文件夹	
STM3210E-EVAL	2019/1/12 11:35	文件夹	
note	2011/4/4 19:03	文本文档	3 KB
Project.uvgui Administrator	2016/2/1 17:15	ADMINISTRATO...	159 KB
Project.uvguix.DF	2019/1/12 11:36	DF 文件	89 KB
Project.uvopt	2016/2/1 17:15	UVOPT 文件	85 KB
Project.uvoptx	2019/1/12 11:36	UVOPTX 文件	78 KB
Project.uvproj.saved_uv4	2016/1/24 8:45	SAVED UV4 文件	370 KB
Project	2018/12/15 22:29	Microvision5 Project	408 KB
Project_cont_frame.dep	2016/1/24 8:44	DEP 文件	131 KB
Project_cont_wave.dep	2016/1/24 8:33	DEP 文件	131 KB
Project_ds_twr_init.dep	2016/2/1 17:12	DEP 文件	131 KB
Project_rx_send_resp.dep	2016/1/24 8:28	DEP 文件	131 KB
Project_simple_tx.dep	2016/1/23 23:23	DEP 文件	131 KB
Project_STM3210E-EVAL.dep	2016/1/23 1:12	DEP 文件	128 KB
Project_STM32100E-EVAL.dep	2016/1/22 22:30	DEP 文件	109 KB
Project_tx_sleep.dep	2016/1/24 0:00	DEP 文件	131 KB
readme	2011/4/4 19:03	文本文档	5 KB

4. 工程目录如下所示。



提示：

三边定位代码，卡尔曼滤波以及AT24CXX代码未使用，仅供参考。

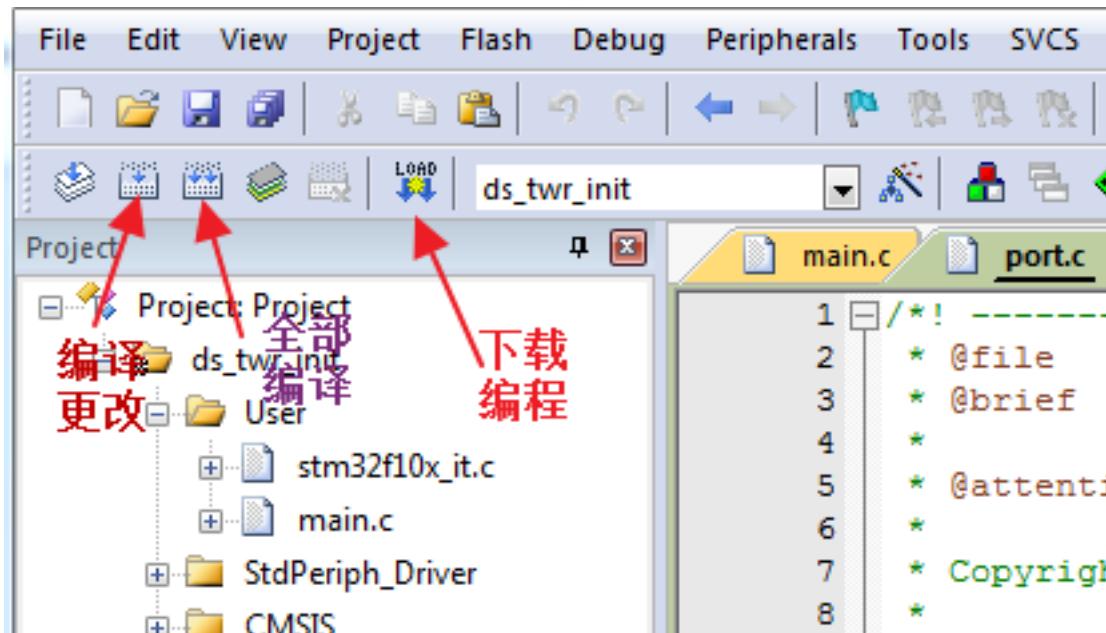
5. 修改代码。

- 修改标签与基站，可参考[标签基站的选择](#) on page 24。
- 修改为跟踪小车功能，可参考[跟踪小车](#) on page 25。

6. 编译代码，当编译后没有错误，即可下载程序到模块中。

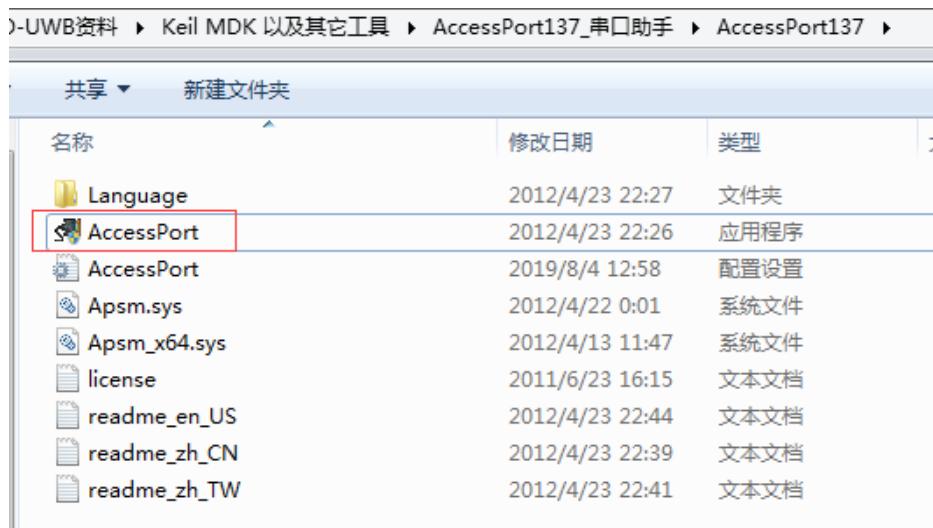
MDK常用的编译方法有“编译更改”、“全部编译”。

当第一次编译时可选择“全部编译”，后面编译可选择“编译更改”，只编译后面修改的部分可以加快编译时间。



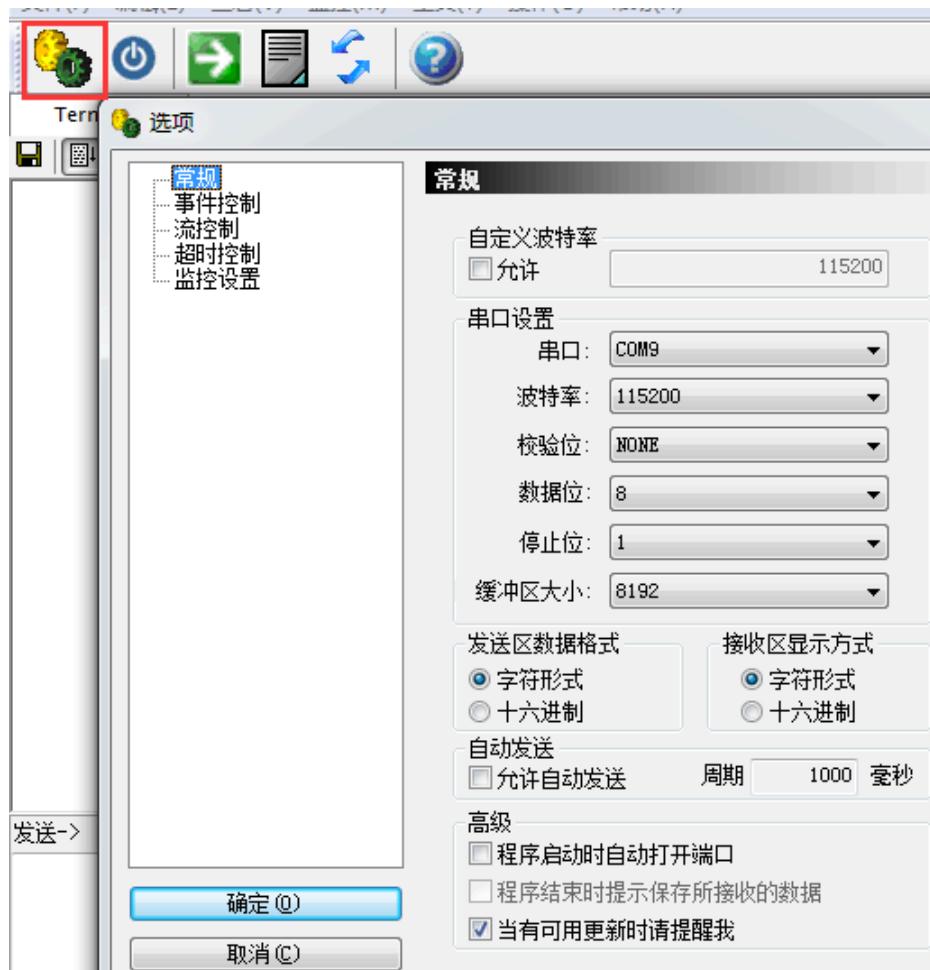
## 查看串口信息

1. 在PC中的如下路径，单击红框，打开串口助手。

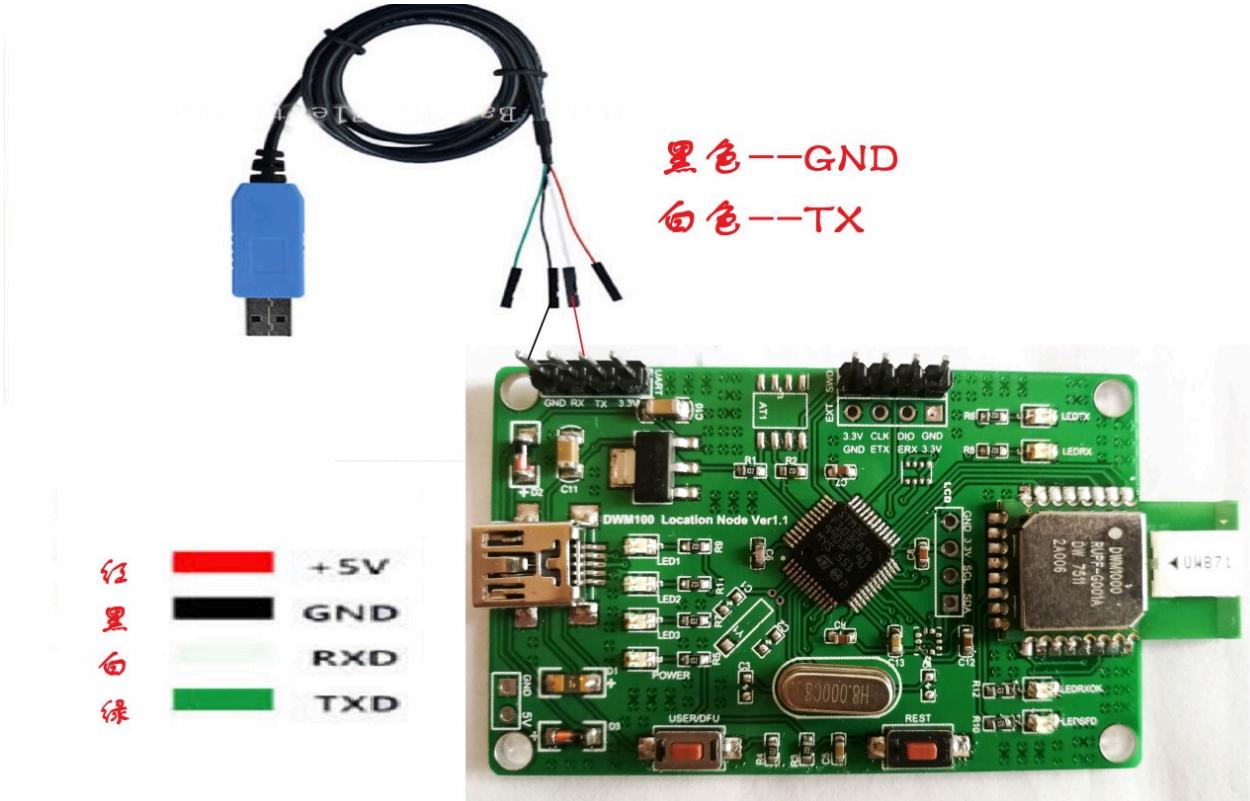


2. 如下所示，单击按钮，进行串口配置，串口波特率为115200。

可参考<https://jingyan.baidu.com/article/647f0115e398b97f2148a8f3.html>，确认PC连接的实际串口号。



3. 如下图所示，串口的黑色连接基站0的GND，串口的白色连接基站0的TX，串口另一端连接到电脑上。



4. ，串口软件界面显示如下，为上电模块初始化信息。

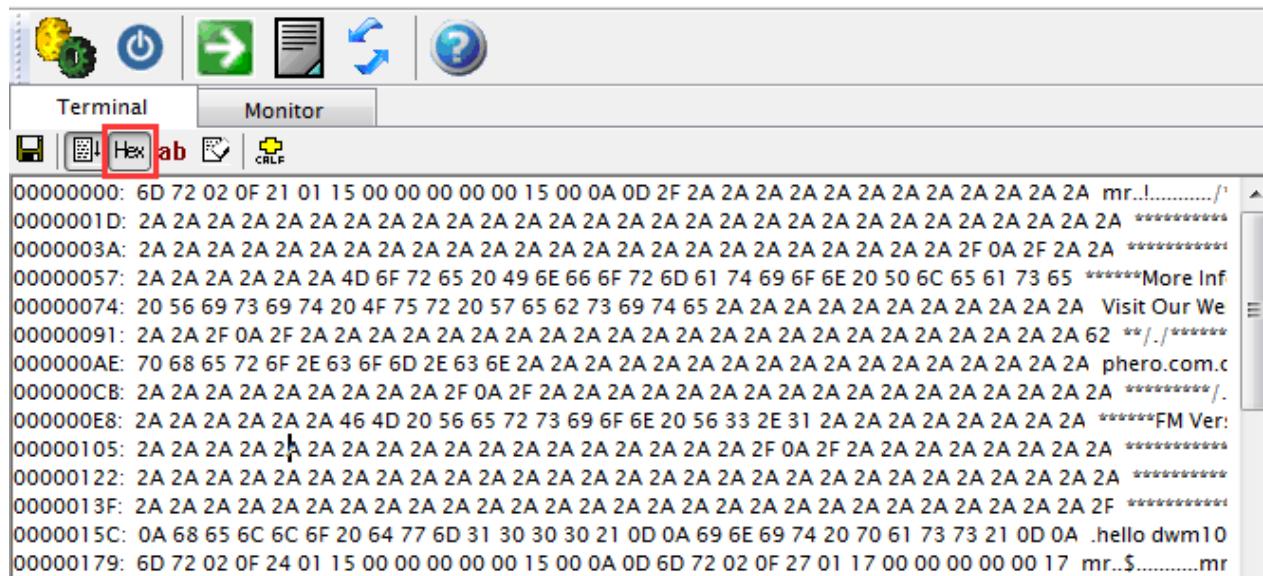
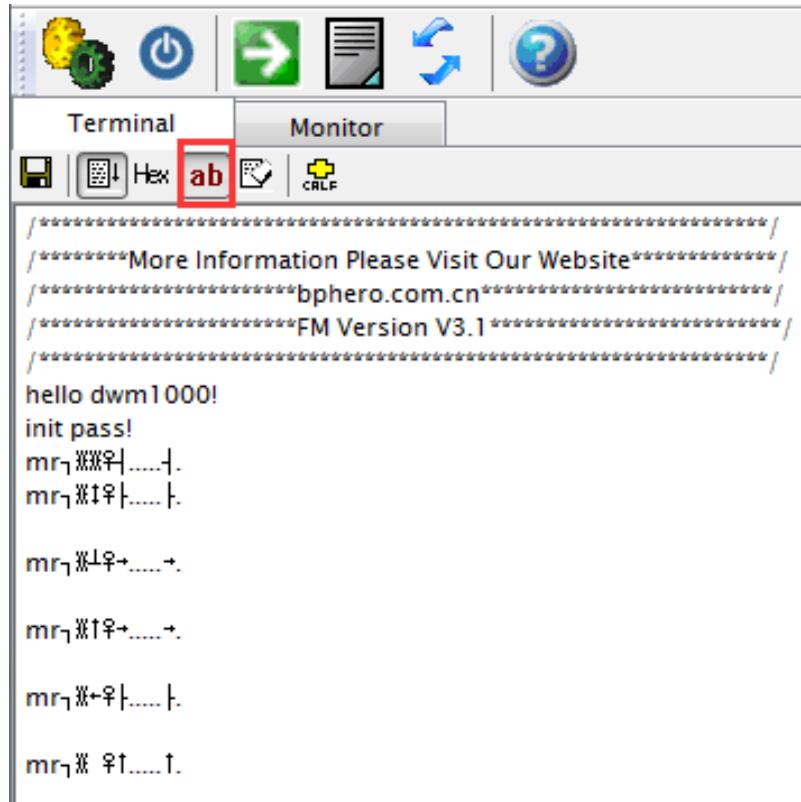
如果显示init fail，则模块异常，请联系客服。

```
*****
*****More Information Please Visit Our Website*****
*****bphero.com.cn*****
*****FM Version V3.1*****
*****
hello dwm1000!
init pass!
```

5. 重复上述方法分别连接基站，查看基站串口信息。

在标签未上电时，基站的串口显示数据和标签相同。

在标签上电时，除了基站0的其他基站串口数据与标签相同，基站0仅在上电瞬间或复位瞬间时显示初始化信息，如下第一张图所示。后续则传输距离信息，选择Hex，请参考[测距与定位的串口数据](#) on page 30理解串口信息。



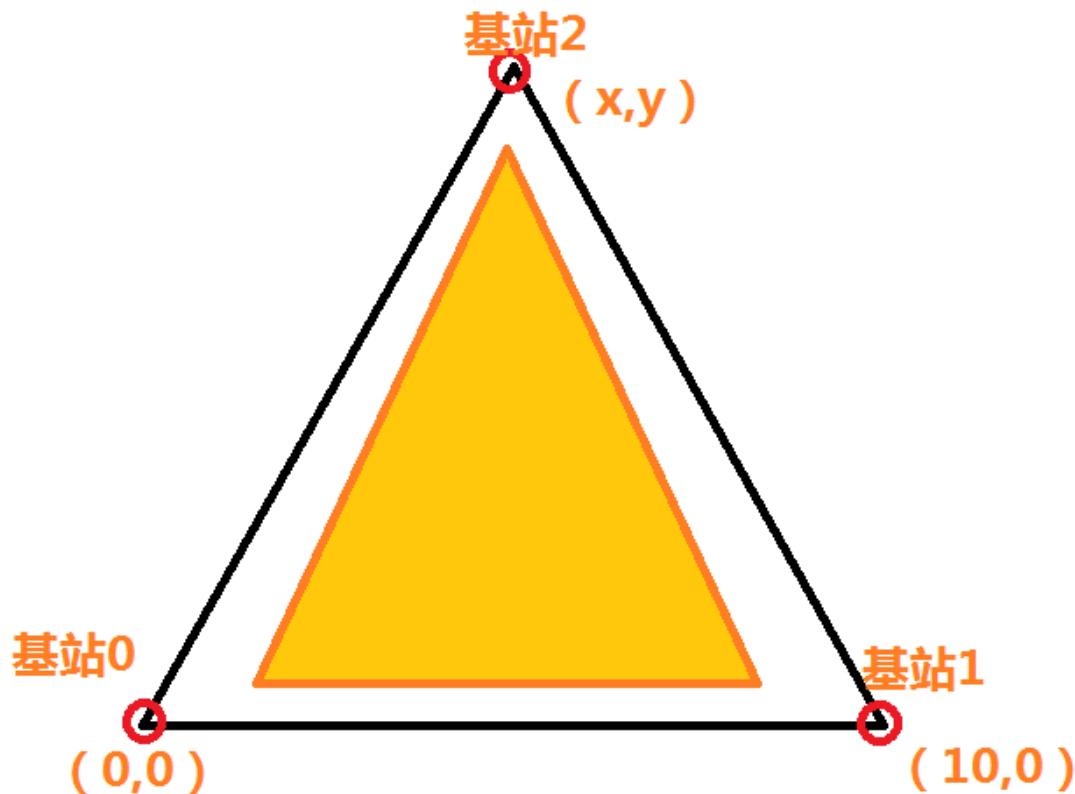
## 放置基站

将烧录好程序的基站放到指定位置，若三个基站，则放置为三角形，建议等边三角形。四个基站，则放置为矩形，建议正方形。

对于3基站1标签定位，只能实现2D定位，即所有模块需要在一个平面上，推荐使用等高的移动电源供电，所有模块都垂直放置到同一水平面。模块之间不要放置障碍物，障碍物对于定位误差影响很大。

基站0和基站1在x轴上，基站2在y轴上。举例如下图所示，基站0放置的坐标是(0,0)，基站1放置在x轴上，基站2放置在(x,y)，可以在(5,8)处，5是中点，8是自定义值。

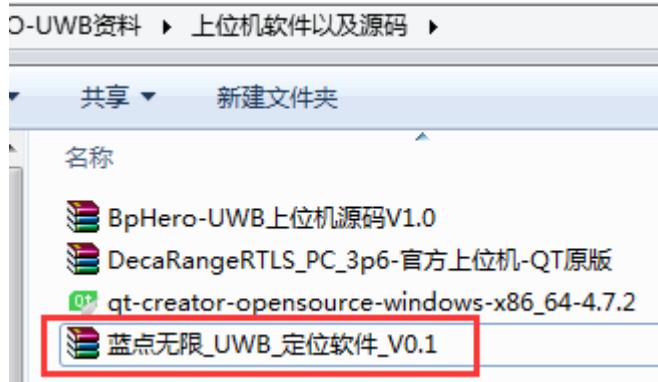
由于通信定位计算中有一定的误差，标签活动范围需要在上图的实心黄色部分。



## 打开定位软件

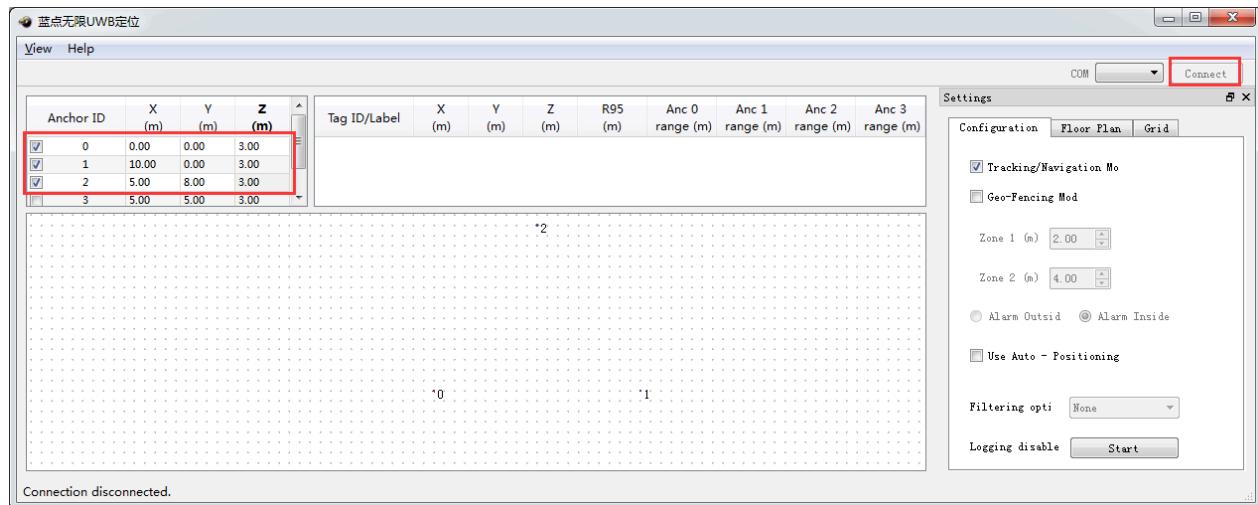
1. 基站0连接串口，详细操作请参考[#unique\\_15/unique\\_15\\_Connect\\_42\\_step4](#)。

2. 如图所示，解压以下文件，并单击解压后的文件。



3. 按照实际坐标与距离进行填写，单击“Connect”。

标签信息会自动显示在界面上，建议标签在基站包围的范围内运行。



## 代码介绍

### 标签基站的选择

```

00183: // #define TAG
00184: #define TAG_ID 0x0F
00185: #define MASTER_TAG 0x0F
00186: #define MAX_SLAVE_TAG 0x02
00187: #define SLAVE_TAG_START_INDEX 0x01
00188:
00189: #define ANTHOR
00190: #define ANCHOR_MAX_NUM 3
00191: #define ANCHOR_IND 1 // 0 1 2

```

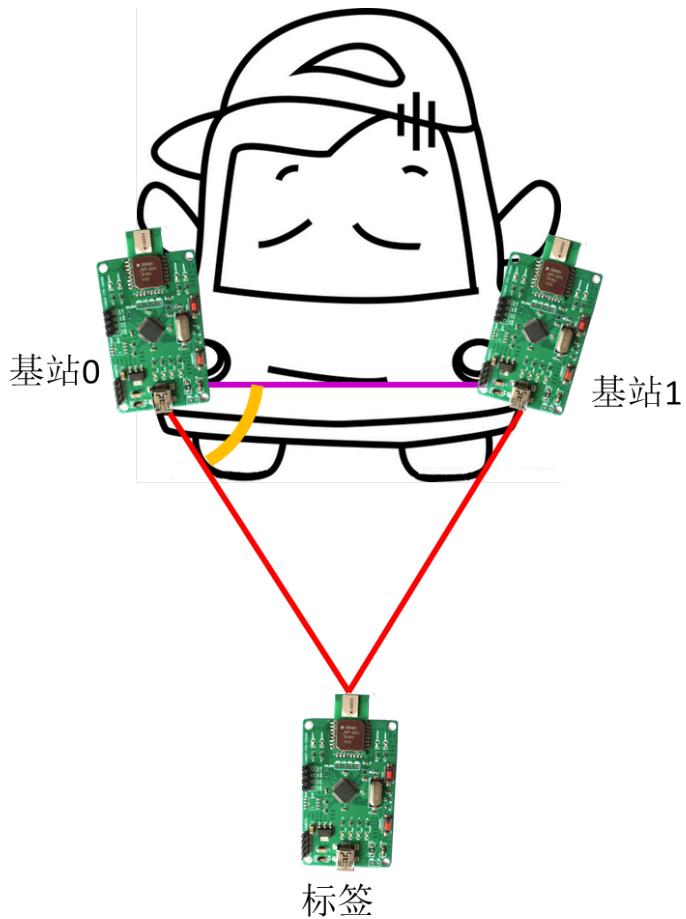
参数	说明
TAG	<ul style="list-style-type: none"> <li>TAG为标签开关，标签分为MASTER TAG和SLAVE TAG，MASTER TAG负责调度SLAVE TAG分配的测距任务。</li> </ul>
ANTHOR	<ul style="list-style-type: none"> <li>ANTHOR为基站开关。</li> <li>TAG与ANTHOR只能选择其中一个，使用时需删除前面的“//”。</li> </ul>
TAG_ID	<p>定义SLAVE TAG。</p> <ul style="list-style-type: none"> <li>一个标签时，该TAG值需与MASTER_TAG的值一样。</li> <li>多个标签时，除了第一个标签与MASTER_TAG的值一样，其它标签时，该TAG值可自定义，该值不能与其他值重复。</li> </ul>
MASTER TAG	定义MASTER TAG，系统中必须有MASTER TAG，MASTER_TAG值无需修改。
MAX_SLAVE_TAG	定义系统中的SLAVE TAG个数。
SALVE_TAG_START_INDEX	无需修改该值。
ANTHOR_MAX_NUM	定义基站总数。
ANTHOR_IND	基站ID，从0开始，若基站共有3个，分别修改分别编译基站0、1、2。

## 跟踪小车

---

### 概述

如下图所示，手持模块为MASTER TAG，车上的两个模块分别为基站0和基站1，两条红线为手持模块与基站0、基站1的测量距离。根据余弦定理可计算黄色角度，根据该角度和一条红线长度可控制小车移动。

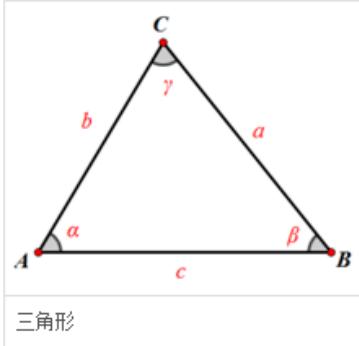


### 余弦定理

如下图所示为余弦定理计算角度，更多跟踪小车原理说明请参见网站 [bphero.com.cn](http://bphero.com.cn)。

对于任意三角形，任何一边的平方等于其他两边平方的和减去这两边与它们夹角的余弦的积的两倍。 [1]

若三边为 $a$ ,  $b$ ,  $c$  三角为 $A$  ( $\alpha$ ),  $B$  ( $\beta$ ),  $C$  ( $\gamma$ ) , 则如下图所示, 在 $\triangle ABC$ 中,



### 余弦定理表达式1

$$c^2 = a^2 + b^2 - 2ab \cos \gamma$$

同理, 也可描述为:

$$b^2 = c^2 + a^2 - 2ac \cos \beta$$

$$a^2 = b^2 + c^2 - 2bc \cos \alpha$$

勾股定理是余弦定理的特例, 当  $\gamma$  为 $90^\circ$ 时,  $\cos(\gamma) = 0$  , 余弦定理可简化为  $c^2 = a^2 + b^2$  , 即勾股定理。

### 余弦定理表达式2

$$\cos \alpha = \frac{b^2 + c^2 - a^2}{2bc}$$

$$\cos \beta = \frac{c^2 + a^2 - b^2}{2ca}$$

$$\cos \gamma = \frac{a^2 + b^2 - c^2}{2ab}$$

### 跟踪开关

BPhero-UWB代码除了可用于测距定位, 还可以用于简易跟踪小车。通过修改如下代码可用于跟踪小车。

```

.27: #define DISTANCE3 0.27
.28:
.29: //*****
.30: //distance1 anthor0 <--> TAG mm
.31: //distance2 anthor1 <--> TAG mm
.32: //distance3 anthor2 <--> TAG mm
.33: //*****
.34: static void compute_angle_send_to_anthor0(int distance1, int distance2,int distance3)
.35: {
.36:     static int framenum = 0 ;
.37:
.38: #if 1 //compute angle for smartcar
.39:     float dis3_constans = DISTANCE3;
.40:     float cos = 0;
.41:     float angle = 0 ;
.42:     float dis1 = (float)distance1/1000; //m
.43:     float dis2 = (float)distance2/1000; //m
.44:

```

参数	说明
if	总开关。 • 1时为跟踪小车功能。 0时为测距定位功能。
define DISTANCE3	小车上两个基站之间的实际距离，请根据实际距离进行修改。

### 基站0与基站1的距离

定义基站0与基站1之间的距离，并将距离单位换算成米。该距离建议尽量远（建议0.3m以上），两个模块越远算出的角度越精确。

```

01135: static void compute_angle_send_to_anthor0(int distance1, int distance2,int distance3)
01136: {
01137:     static int framenum = 0 ;
01138:
01139: #if 1 //compute angle for smartcar
01140:     float dis3_consts = DISTANCE3;
01141:     float cos = 0;
01142:     float angle = 0 ;
01143:     float dis1 = (float)distance1/1000; //m
01144:     float dis2 = (float)distance2/1000; //m
01145:
01146:     if(dis1 + dis3_consts < dis2 || dis2+dis3_consts < dis1)
01147:     {
01148:         //printf("ERROR!\r\n");
01149:         //return;
01150:     }

```

### 余弦定理计算角度

通过余弦定理计算出弧度（cos），再转换为角度（angle），最后将角度信息显示在液晶上，便于调试。

```

1:     cos = (dis1*dis1 + dis3_consts* dis3_consts - dis2*dis2)/(2*dis1*dis3_consts);
2:     angle = acos(cos)*180/3.1415926;
3:     printf("cos = %f, arccos = %f\r\n",cos,angle);
4:     sprintf(dist_str, "angle: %3.2f m", angle);
5:     OLED_ShowString(0, 6,"");
6:     OLED_ShowString(0, 6,dist_str);
7:

```

### 设置启动距离与角度

if(dis1>1)为设置小车启动距离，当小车与被跟踪模块大于1时距离时启动，小于这个距离时，小车停止不动。

if(angle >110 , else if angle <75)为设置小车的左、右转向。当角度小于75度时，右转；但角度大于110度时，左转。

```

if(dis1 > 1)
{
    if(angle > 110)
    {
        printf("turn right\r\n");
        angle_msg[10] = 'R';
    }
    else if(angle < 75)
    {
        printf("turn left\r\n");
        angle_msg[10] = 'L';
    }
    else
    {
        printf("forward\r\n");
        angle_msg[10] = 'F';
    }
}
else
{
    printf("stay here\r\n");
    angle_msg[10] = 'S';
}
angle_msg[LOCATION_FLAG_IDX] = 0;

```

### 收发信息

系统将角度与距离信息放置在rx\_buffer[10]中，MASTER TAG通过射频将该信息发送给基站0。基站0的串口TX与小车主控MCU的RX相连，基站0通过串口将发送数据给主控MCU，主控MCU负责控制小车前后左右行驶。

```

else if (memcmp(rx_buffer, angle_msg, ALL_MSG_COMMON_LEN) == 0 && ANCHOR_IND == 0)
{
    if(rx_buffer[LOCATION_FLAG_IDX] == 1)//location infomartion
    {
        rx_buffer[ALL_MSG_TAG_IDX] = tag_index;
        USART_puts(&rx_buffer[LOCATION_INFO_START_IDX], rx_buffer[LOCATION_INFO_LEN_IDX]);
    }
    else //follow car
    {
        putchar(rx_buffer[10]);
    }
}

```

下表为串口数据格式，用户需自行准备MCU，BPhero-UWB只会通过串口发送给MCU特定指令，用户需要根据这些指令自行设置MCU。

字符	说明
R	右转
L	左转
F	前进
S	停止

## 测距与定位的串口数据

```
'm'    'r'    0x02  TAG_IDframe_Frame_Dis_0_Dis_0_Dis_1_Dis_1_Dis_2_Dis_2_Dis_X_Dis_X_H'  '\n'
```

- ‘m’ 和‘r’为自定义数据头。
- 0x02为版本号。
- TAG\_ID为标签ID，多标签时可通过这个ID区别。
- Frame\_1、Frame\_2 为帧序列号。
- Dis\_x\_L、Dis\_x\_H为基站与标签的距离，其中Dis\_x\_L表示低8位，Dis\_x\_H表示高8位。
  - 3个基站时，最后的两个为基站0到标签的距离，与前面的Dis\_0\_L、Dis\_0\_H相同。
  - 4个基站时，最后的两个为第四个基站与标签的距离。
- ‘\r’和‘\n’为自定义数据结束，换行。

## 均值滤波

均值滤波为减少定位过程中的误差。

### 均值

标签和基站多次测距后，将多次距离取平均值，然后再传递给后面的代码。

ANCHOR\_REFRESH\_COUNT为计算次数，已提前宏定义，可根据实际情况修改。如果需要提高刷新频率，可将该值修改小，反之亦然。目前测试三基站一标签最快刷新频率为40HZ。

```

00457:         int Anchor_Index = 0;
00458:         while(Anchor_Index < ANCHOR_MAX_NUM)
00459:         {
00460:             if(Anhordistance_count[Anchor_Index] >= ANCHOR_REFRESH_COUNT )
00461:             {
00462:                 distance_mange();
00463:                 Anchor_Index = 0;
00464:                 //clear all
00465:                 while(Anchor_Index < ANCHOR_MAX_NUM)
00466:                 {
00467:                     Anhordistance_count[Anchor_Index] = 0;
00468:                     Anhordistance[Anchor_Index] = 0;
00469:                     Anchor_Index++;
00470:                 }
00471:                 break;
00472:             }
00473:             Anchor_Index++;
00474:         }
00475:
```

### 滤波

滤波可防止较大波动出现，本代码中使用了较容易实现的中值滤波，代码简单，执行速度快。

```

01035: #define Filter_N 5 //max filter use in this system
01036: #define Filter_D 5 //each filter contain "Filter_D" data
01037: int Value_Buf[Filter_N][Filter_D] = {0};
01038: int filter_index[Filter_N] = {0};
01039: int filter(int input, int fliter_idx)
01040: {
01041:     char count = 0;
01042:     int sum = 0;
01043:     if(input > 0)
01044:     {
01045:         Value_Buf[fliter_idx][filter_index[fliter_idx]++] = input;
01046:         if(filter_index[fliter_idx] == Filter_D) filter_index[fliter_idx] = 0;
01047:
01048:         for(count = 0; count < Filter_D; count++)
01049:         {
01050:             sum += Value_Buf[fliter_idx][count];
01051:         }
01052:         return (int)(sum/Filter_D);
01053:     }
01054:     else
01055:     {
01056:         for(count = 0; count < Filter_D; count++)
01057:         {
01058:             sum += Value_Buf[fliter_idx][count];
01059:         }
01060:         return (int)(sum/Filter_D);
01061:     }
01062:
01063: } ? end filter ?

```

滤波中有两个重要的宏定义，Filter\_N和Filter\_D。

- Filter\_N表示滤波器数量，每个基站需要一个独立的滤波器，目前代码中定义了5个滤波器。Filter\_D越大，平滑性越好，但是反应会越慢，则计算的坐标点跟不上实际移动速度。
- Filter\_D表示一个滤波器内的数据条数，目前代码中定义是5，则滤波器将会对5条数据求中值。Filter\_D和均值宏定义ANCHOR\_REFRESH\_COUNT两者选取需要均衡，而不是两者同时选择较大值。

## 售后与保修政策

---

- 按照快递签收时间算起7日内发现质量问题，可包邮保修或更换（需在线和客服说明原因并拍好清晰照片当场确认是否人为损坏，人为损坏不保修、不退换）。
- 7日后15日内免费保修，运费买卖双方各自负责（只限于产品本身质量问题，人为损坏的买家承担来回运费并支付维修器件成本费用）。
- 15日后1年内免人工维修费，酌情收取配件更换成本费，来回运费买家负责，无法维修的将原样返回给买家。