

# 自行车码表II

---

- 自行车码表II
  - 实验目的
  - 实验器材
    - 硬件
    - 软件
  - 实验原理
  - 实验步骤
    - 通过面包板在PA11和PA12各连接一个按钮开关到地;
    - 编写中断驱动模式程序, 以中断处理时钟定时器和两个一个按钮 (模拟钢圈计数), 主程序循环中不做事情。
  - 实验心得

## 实验目的

- 熟练掌握中断驱动程序编程方法。

## 实验器材

### 硬件

- STM32F103核心板1块;
- ST-Link 1个;
- 杜邦线(孔-孔) 4根;
- 杜邦面包线 (孔-针) 3根;
- 面包线 (针-针) 若干;
- 按钮2个;
- 面包板1块。

### 软件

- STM32CubeIDE;
- Serial Port Utility(友善串口调试助手)

## 实验原理

## 自行车码表功能

### 启动

```
配置定时器、
输入和
输出

cur_time = 0;
rotations = 0;
tenth_miles = 0;

while (1) {
    sleep;
}
```

### ISR 1: 钢圈旋转

```
rotations++;
if (rotations >
    R_PER_MILE/10) {
    tenth_miles++;
    rotations = 0;
}
speed =
    circumference/
    (cur_time - prev_time);
计算avg_speed;
prev_time = cur_time;
从中断返回
```

### ISR 2: 模式按钮

```
mode++;
mode = mode %
    NUM_MODES;
从中断返回
```

### ISR 3: 时间定时器

```
cur_time++;
lcd_refresh--;
if (lcd_refresh == 0) {
    转换tenth_miles
    并显示
    转换speed
    并显示
    if (mode == 0)
        转换cur_time
        并显示
    else
        转换avg_speed
        并显示
    lcd_refresh =
        LCD_REF_PERIOD
}
```

ARM University Program  
Copyright © ARM Ltd 2013

THE ARCHITECTURE FOR THE DIGITAL WORLD™



ARM

## 自行车码表功能

MVC

### 启动

```
配置定时器、
输入和
输出
mode = 0
cur_time = 0;
rotations = 0;
tenth_miles = 0;

while (1) {
    sleep;
}
```

### ISR 1: 钢圈旋转

```
rotations++;
if (rotations >
    R_PER_MILE/10) {
    tenth_miles++;
    rotations = 0;
}
speed =
    circumference/
    (cur_time - prev_time);
计算avg_speed;
prev_time = cur_time;
从中断返回
```

### ISR 2: 模式按钮

```
mode++;
mode = mode %
    NUM_MODES;
从中断返回
```

### ISR 3: 时间定时器

```
cur_time++;
lcd_refresh--;
if (lcd_refresh == 0) {
    转换tenth_miles
    并显示
    转换speed
    并显示
    if (mode == 0)
        转换cur_time
        并显示
    else
        转换avg_speed
        并显示
    lcd_refresh =
        LCD_REF_PERIOD
}
```

64 {  
V + timer

32  
mode = 0

0 1 3 2  
5  
0/4  
80x

LCD  
显示

ARM University Program  
Copyright © ARM Ltd 2013

THE ARCHITECTURE FOR THE DIGITAL WORLD™

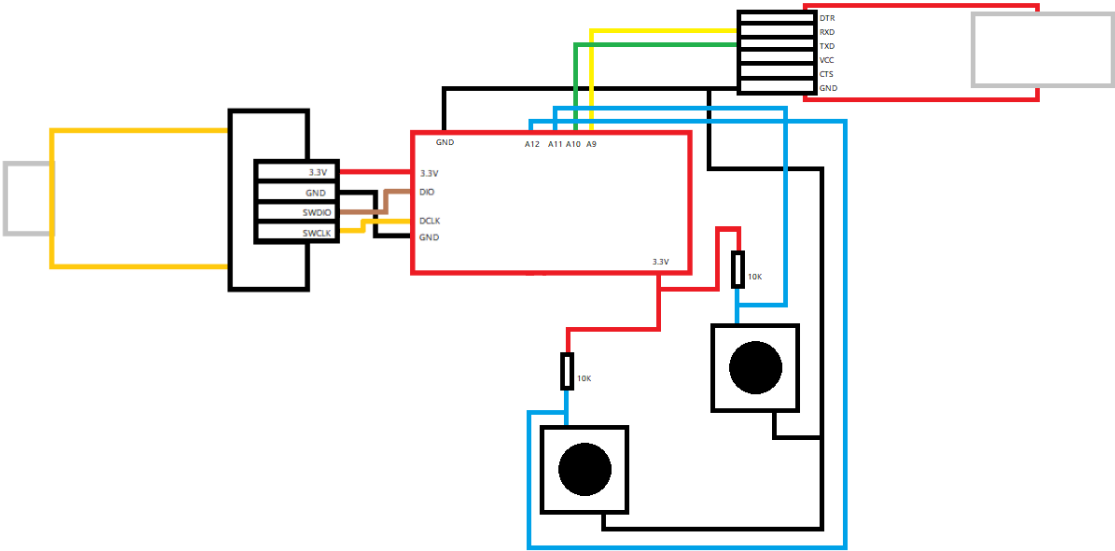


ARM

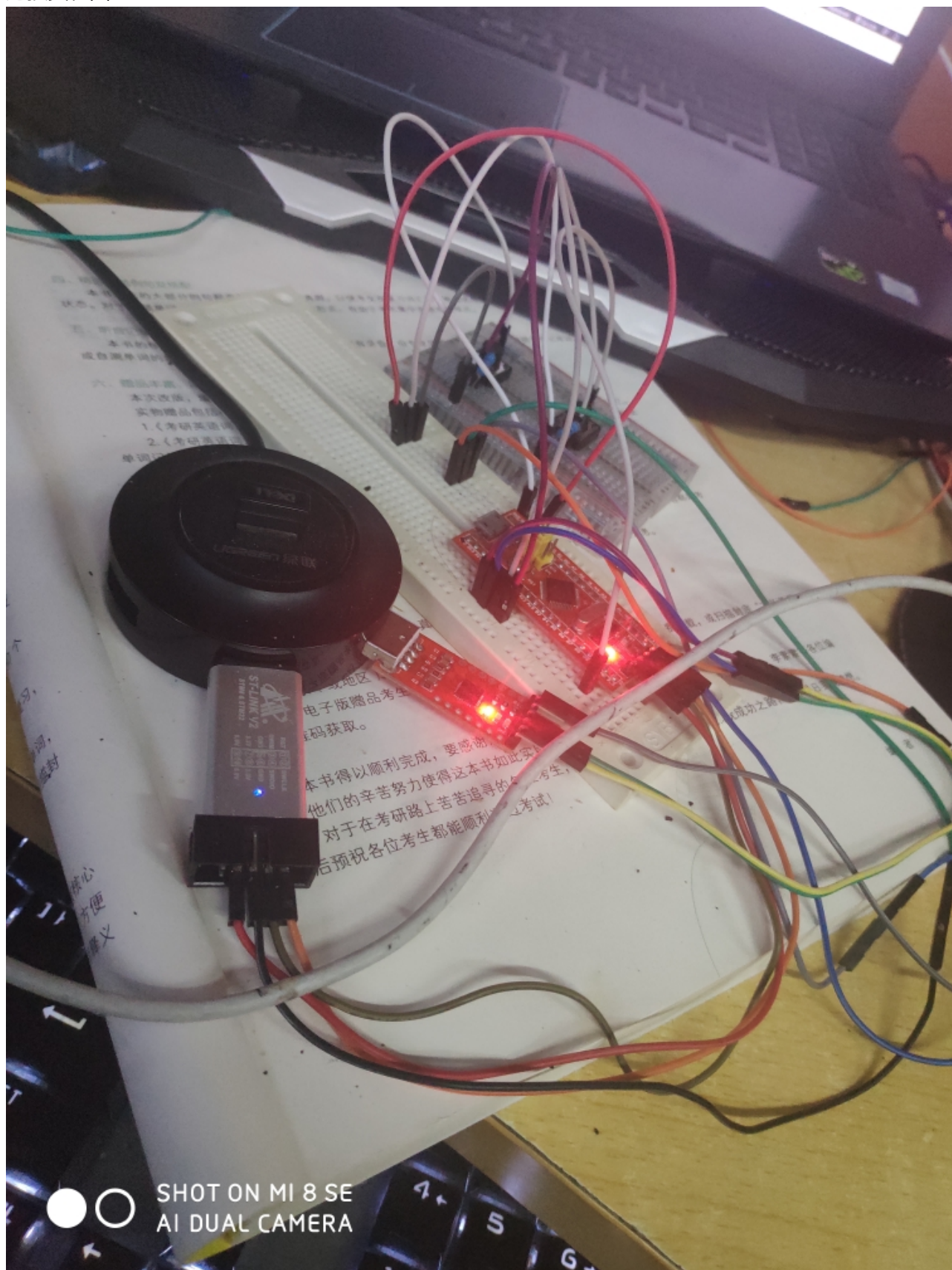
## 实验步骤

通过面包板在PA11和PA12各连接一个按钮开关到地;

- 连接原理图



- 连接实物图



编写中断驱动模式程序，以中断处理时钟定时器和两个一个按钮（模拟钢圈计数），主程序循环中不做事情。

- 工程配置

- 设置两个按钮对应引脚PA11和PA12;

PA11 Configuration :

GPIO mode

Input mode

GPIO Pull-up/Pull-down

No pull-up and no pull-down

User Label

Mode

PA12 Configuration :

GPIO mode

External Interrupt Mode with Falling edge trigger

GPIO Pull-up/Pull-down

Pull-up

User Label

Data

- 定时器配置(1ms触发一次);

### TIM4 Mode and Configuration

Mode

Slave Mode Disable v

Trigger Source Disable v

☒ Internal Clock

Channel1 Disable v

Channel2 Disable v

Channel3 Disable v

Channel4 Disable v

Combined Channels Disable v

☐ XOR activation

Configuration

Reset Configuration

<input checked="" type="checkbox"/> NVIC Settings	<input checked="" type="checkbox"/> DMA Settings
<input checked="" type="checkbox"/> Parameter Settings	<input checked="" type="checkbox"/> User Constants

Configure the below parameters :

🔍 Search (Ctrl+F)
↶ ↷
i

▼ Counter Settings

Prescaler (PSC - 16 bits v.. 7199

Counter Mode                      Up

Counter Period (AutoRelo...9

Internal Clock Division (C... No Division

- 中断优先级设置(优先级比较是先比较组优先级，如果相同则比较子优先级，越小优先级越高);  
NVIC Mode and Configuration

Configuration

NVIC

Code generation

Priority Group 

2 bi...

☐ Sort by Preemption Priority and Sub Priority

☐ Sort by interrupts names

Search 

☒ Show only enabled interrupts

☒ Force DMA channels Interrupt

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	<input checked="" type="checkbox"/>	0	0
Hard fault interrupt	<input checked="" type="checkbox"/>	0	0
Memory management fault	<input checked="" type="checkbox"/>	0	0
Prefetch fault, memory access fault	<input checked="" type="checkbox"/>	0	0
Undefined instruction or illegal state	<input checked="" type="checkbox"/>	0	0
System service call via SWI instruction	<input checked="" type="checkbox"/>	0	0
Debug monitor	<input checked="" type="checkbox"/>	0	0
Pendable request for system service	<input checked="" type="checkbox"/>	0	0
Time base: System tick timer	<input checked="" type="checkbox"/>	0	0
DMA1 channel4 global interrupt	<input checked="" type="checkbox"/>	0	0
DMA1 channel5 global interrupt	<input checked="" type="checkbox"/>	0	0
EXTI line[9:5] interrupts	<input checked="" type="checkbox"/>	0	2
TIM1 capture compare interrupt	<input checked="" type="checkbox"/>	1	0
TIM4 global interrupt	<input checked="" type="checkbox"/>	0	1
USART1 global interrupt	<input checked="" type="checkbox"/>	0	0
EXTI line[15:10] interrupts	<input checked="" type="checkbox"/>	0	2

☒ Enabled

Preemption Priority 

0

Sub Priority 

2

- 设置定时器中断优先级高于DMA中断高于外部中断

• 代码

```
/* stm32f1xx_it.c中利用SysTick对PA11按钮去抖动 */
extern uint8_t BtnState;           // 按键当前状态
extern uint8_t BtnChangeFlag;      // 按键状态是否发生改变
extern uint8_t bike_watch_mode;    // 码表模式切换
void Key_Scan(void)
{
    if(HAL_GPIO_ReadPin(Mode_GPIO_Port,Mode_Pin) == GPIO_PIN_RESET)
//判断有没有按
    {
        char buf[64];
        if(btnCount == 0)
        {
            BtnState=1;              //设置按键标志
            if(pushFlag == 0)
            {
                last = HAL_GetTick();
                pushFlag = 1;
            }
            btnCount = 0;
        }
    }
}
```



```

    }
}
else
{
    btnCount = 0;
    if(BtnState == 1)
    {
        now = HAL_GetTick();
        BtnChangeFlag = 1;
        pushFlag = 0;
        HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);
        bike_watch_mode = (bike_watch_mode + 1) % 4; // 4种模式
    }
    else
    {
        BtnChangeFlag = 0;
    }
    BtnState = 0;
}
}

void SysTick_Handler(void)
{
    /* USER CODE BEGIN SysTick_IRQn 0 */
    Key_Scan();
    /* USER CODE END SysTick_IRQn 0 */
    HAL_IncTick();
    /* USER CODE BEGIN SysTick_IRQn 1 */
    HAL_SYSTICK_IRQHandler();
    /* USER CODE END SysTick_IRQn 1 */
}

/* main.c中定义全局变量 */
uint8_t BtnState = 0; // 按键当前状态
uint8_t BtnChangeFlag = 0; // 按键是否改变状态
uint16_t bike_watch_counter = 0, bike_watch_last_counter = 0; // 码表计数器, 加1表示转动1圈, 假定转动1圈前进2.56m
uint8_t bike_watch_mode = 0, bike_time_flag = 0; // 码表模式控制和码表刷新频率设置
uint16_t bike_watch_time = 0, bike_watch_time_last = 0; // 码表计时器
double distance = 0, bike_speed = 0, bike_average_speed = 0; // 路程、速度、平均速度
/* main.c中定义引脚PA12对应按钮的外部中断回调函数 */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    if(GPIO_Pin == GPIO_PIN_12)
    {
        bike_watch_counter++; // 计数器
        distance += 2.56 * (bike_watch_counter - bike_watch_last_counter); // 计算路程
        if(bike_watch_time != bike_watch_time_last) // 计

```



算速度, 防止除0

```

        bike_speed = 2.56 * 1000 * (bike_watch_counter -
bike_watch_last_counter) / (bike_watch_time - bike_watch_time_last);
    else
        bike_speed = 0;
        bike_average_speed = 2.56 * 1000 * (bike_watch_counter) /
(bike_watch_time);    // 计算平均速度
        bike_watch_time_last = bike_watch_time;
        if(bike_watch_counter == 65536)
// 防止计数器溢出
        {
            bike_watch_counter = 0;
        }
    }
}
/* main.c中定义TIM4定时器中断回调函数 */
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    if (htim->Instance == TIM4)
    {
        bike_watch_time ++;           // 计时器
        bike_time_flag ++;           // 刷新计数器
        if(bike_time_flag == 100)    // 大约100ms刷新一次
        {
            bike_time_flag = 0;
            char buf[100];
            uint16_t length;
            if(bike_watch_mode == 0)    // 模式1: 显示路程
            {
                length = sprintf(buf, "Distance: %.3fm \r\n", distance);
            }
            else if(bike_watch_mode == 1) // 模式2: 显示时间
            {
                length = sprintf(buf, "Time: %.3fs\r\n", (float)(1.0 *
bike_watch_time / 1000));
            }
            else if(bike_watch_mode == 2) // 模式3: 显示速度
            {
                if(bike_watch_time != bike_watch_time_last)
                {
                    length = sprintf(buf, "Speed: %.3f m/s\r\n", bike_speed);
                }
            }
            else if(bike_watch_mode == 3) // 模式4: 显示平均速度
            {
                length = sprintf(buf, "Average Speed: %.3f m/s\r\n",
bike_average_speed);
            }
            HAL_UART_Transmit_DMA(&huart1, (uint8_t *)buf, length); // 串口输出
        }
        bike_watch_last_counter = bike_watch_counter;
    }
}

```

/\* 扩展内容: main.c中main函数内while循环可以加入以下代码完成休眠操作, 不过由于会关闭Sys

```

tick中断, 所以需要将引脚PA11对应的按钮修改成引脚中断触发 */
HAL_SuspendTick();
__HAL_RCC_PWR_CLK_ENABLE();
HAL_PWR_EnterSLEEPMode(PWR_MAINREGULATOR_ON, PWR_SLEEPENTRY_WFI);
HAL_ResumeTick();
/* 注: 最后实验没有采用实现休眠操作, 因为在实验过程中发现以上代码编译下载到板子上后, 后续
烧录操作都会失败, 需要根据实验1提供的实验指南对跳帽操作才能再次编译。 */

```

## • 结果

### ◦ 时间

```

[20:17:01.807] Time: 50.900s
[20:17:01.907] Time: 51.000s
[20:17:02.007] Time: 51.100s
[20:17:02.107] Time: 51.200s
[20:17:02.207] Time: 51.300s
[20:17:02.307] Time: 51.400s
[20:17:02.407] Time: 51.500s
[20:17:02.507] Time: 51.600s
[20:17:02.607] Time: 51.700s
[20:17:02.707] Time: 51.800s
[20:17:02.807] Time: 51.900s
[20:17:02.907] Time: 52.000s
[20:17:03.007] Time: 52.100s
[20:17:03.107] Time: 52.200s
[20:17:03.207] Time: 52.300s
[20:17:03.307] Time: 52.400s
[20:17:03.407] Time: 52.500s

```

### ◦ 路程

```

[20:16:53.907] Speed: 0.629 m/s
[20:16:54.007] Speed: 0.000 m/s
[20:16:54.107] Speed: 0.000 m/s
[20:16:54.207] Speed: 0.000 m/s
[20:16:54.307] Speed: 9.734 m/s
[20:16:54.407] Speed: 17.902 m/s
[20:16:54.507] Speed: 17.902 m/s
[20:16:54.607] Speed: 17.902 m/s
[20:16:54.707] Speed: 10.079 m/s
[20:16:54.808] Speed: 10.079 m/s
[20:16:54.907] Speed: 10.079 m/s
[20:16:55.007] Speed: 9.922 m/s
[20:16:55.107] Speed: 9.922 m/s
[20:16:55.207] Speed: 10.894 m/s
[20:16:55.307] Speed: 10.894 m/s
[20:16:55.407] Speed: 10.079 m/s
[20:16:55.507] Speed: 0.000 m/s
[20:16:55.607] Speed: 21.880 m/s
[20:16:55.707] Speed: 21.880 m/s
[20:16:55.807] Speed: 13.264 m/s
[20:16:55.907] Speed: 13.264 m/s
[20:16:56.007] Speed: 14.545 m/s
[20:16:56.107] Speed: 14.545 m/s
[20:16:56.207] Speed: 15.610 m/s
[20:16:56.307] Speed: 0.000 m/s
[20:16:56.407] Speed: 18.824 m/s
[20:16:56.507] Speed: 18.824 m/s
[20:16:56.607] Speed: 12.995 m/s
[20:16:56.707] Speed: 29.767 m/s
[20:16:56.807] Speed: 18.824 m/s
[20:16:56.907] Speed: 43.390 m/s
[20:16:57.007] Speed: 43.390 m/s
[20:16:57.107] Speed: 43.390 m/s

```

## ○ 速度

---

[20:16:57.607]	Average Speed: 4.233 m/s
[20:16:57.707]	Average Speed: 4.233 m/s
[20:16:57.807]	Average Speed: 4.368 m/s
[20:16:57.907]	Average Speed: 4.368 m/s
[20:16:58.007]	Average Speed: 4.409 m/s
[20:16:58.108]	Average Speed: 4.503 m/s
[20:16:58.207]	Average Speed: 4.503 m/s
[20:16:58.307]	Average Speed: 4.540 m/s
[20:16:58.406]	Average Speed: 4.641 m/s
[20:16:58.507]	Average Speed: 4.684 m/s
[20:16:58.607]	Average Speed: 4.892 m/s
[20:16:58.707]	Average Speed: 4.981 m/s
[20:16:58.807]	Average Speed: 4.981 m/s
[20:16:58.907]	Average Speed: 5.070 m/s
[20:16:59.007]	Average Speed: 5.112 m/s
[20:16:59.107]	Average Speed: 5.157 m/s
[20:16:59.207]	Average Speed: 5.198 m/s
[20:16:59.308]	Average Speed: 5.455 m/s
[20:16:59.407]	Average Speed: 5.496 m/s
[20:16:59.507]	Average Speed: 5.496 m/s
[20:16:59.607]	Average Speed: 5.528 m/s
[20:16:59.707]	Average Speed: 5.562 m/s
[20:16:59.807]	Average Speed: 5.654 m/s
[20:16:59.907]	Average Speed: 5.696 m/s
[20:17:00.007]	Average Speed: 5.696 m/s
[20:17:00.107]	Average Speed: 5.727 m/s
[20:17:00.207]	Average Speed: 5.727 m/s
[20:17:00.307]	Average Speed: 5.757 m/s
[20:17:00.407]	Average Speed: 5.801 m/s
[20:17:00.507]	Average Speed: 5.838 m/s
[20:17:00.607]	Average Speed: 5.934 m/s
[20:17:00.707]	Average Speed: 6.117 m/s

- 平均速度

```
[20:17:05.907] Distance: 670.720m
[20:17:06.007] Distance: 673.280m
[20:17:06.107] Distance: 675.840m
[20:17:06.207] Distance: 675.840m
[20:17:06.307] Distance: 678.400m
[20:17:06.407] Distance: 680.960m
[20:17:06.507] Distance: 683.520m
[20:17:06.607] Distance: 686.080m
[20:17:06.707] Distance: 686.080m
[20:17:06.807] Distance: 691.200m
[20:17:06.907] Distance: 693.760m
[20:17:07.007] Distance: 693.760m
[20:17:07.107] Distance: 711.680m
[20:17:07.207] Distance: 714.240m
[20:17:07.307] Distance: 714.240m
[20:17:07.407] Distance: 716.800m
[20:17:07.507] Distance: 721.920m
[20:17:07.607] Distance: 721.920m
[20:17:07.707] Distance: 724.480m
[20:17:07.808] Distance: 727.040m
[20:17:07.907] Distance: 734.720m
[20:17:08.007] Distance: 737.280m
[20:17:08.107] Distance: 742.400m
[20:17:08.207] Distance: 742.400m
[20:17:08.307] Distance: 742.400m
[20:17:08.407] Distance: 752.640m
[20:17:08.507] Distance: 755.200m
[20:17:08.607] Distance: 755.200m
[20:17:08.707] Distance: 765.440m
[20:17:08.807] Distance: 765.440m
[20:17:08.907] Distance: 765.440m
[20:17:09.007] Distance: 765.440m
```

- 综合

```
[20:17:03.007] Time: 52.100s
[20:17:03.107] Time: 52.200s
[20:17:03.207] Time: 52.300s
[20:17:03.307] Time: 52.400s
[20:17:03.407] Time: 52.500s
[20:17:03.508] Speed: 14.798 m/s
[20:17:03.607] Speed: 14.798 m/s
[20:17:03.707] Speed: 14.798 m/s
[20:17:03.807] Speed: 4.531 m/s
[20:17:03.907] Speed: 2560.000 m/s
[20:17:04.007] Speed: 25.098 m/s
[20:17:04.107] Speed: 25.098 m/s
[20:17:04.207] Speed: 16.732 m/s
[20:17:04.307] Speed: 16.732 m/s
[20:17:04.407] Speed: 16.732 m/s
[20:17:04.507] Average Speed: 7.646 m/s
[20:17:04.607] Average Speed: 7.646 m/s
[20:17:04.707] Average Speed: 7.621 m/s
[20:17:04.807] Average Speed: 7.621 m/s
[20:17:04.907] Average Speed: 7.640 m/s
[20:17:05.007] Average Speed: 7.726 m/s
[20:17:05.107] Average Speed: 7.750 m/s
[20:17:05.207] Average Speed: 7.750 m/s
[20:17:05.307] Average Speed: 7.768 m/s
[20:17:05.407] Average Speed: 7.768 m/s
[20:17:05.507] Average Speed: 7.768 m/s
[20:17:05.607] Distance: 665.600m
[20:17:05.707] Distance: 668.160m
[20:17:05.808] Distance: 668.160m
[20:17:05.907] Distance: 670.720m
[20:17:06.007] Distance: 673.280m
[20:17:06.107] Distance: 675.840m
```

## 实验心得

- 本次实验其实非常简单，一是只是将实验4的实现方式由前后台模式变为中断触发模式。算法、代码变动都不是很大。二是在课堂上老师都已经给我们详细讲解过了。
- 个人认为本次实验比较需要注意的点是：为不同的中断设置不同的优先级。实验中还遇到了一个小问题，就是传输输出字符串的字符数组开的太小了，导致PC查看到的串口输出存在乱码情况。所以如果没有特别情况需要非常节省内存空间的话，尽量还是把数组开的大一点。