

Lab4 码表

- Lab4 码表
 - 实验目的
 - 实验器材
 - 硬件
 - 软件
 - 实验原理
 - 连接电路图
 - TIM定时器
 - 码表
 - 实验步骤
 - 通过面包板在PA11和PA12各连接一个按钮开关到地;
 - 配置GPIO
 - 配置定时器
 - 编写前后台程序，以中断处理时钟定时器和其中一个按钮（模拟钢圈计数），以主循环读取按钮的变化值并根据定时的值做输出，第二个按钮（模拟模式切换）由主程序轮询判断，用以改变输出的数据（速度还是里程）。程序通过串口输出。
 - 扩展内容
 - 用定时器的外部触发来测量时间。
 - 实验心得

实验目的

- 理解MCU上电启动过程;
- 了解嵌入式裸机编程模型的一般情况;
- 熟练掌握前后台编程模型;
- 熟练掌握引脚中断响应程序编程方法;
- 了解中断驱动编程模型。

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。

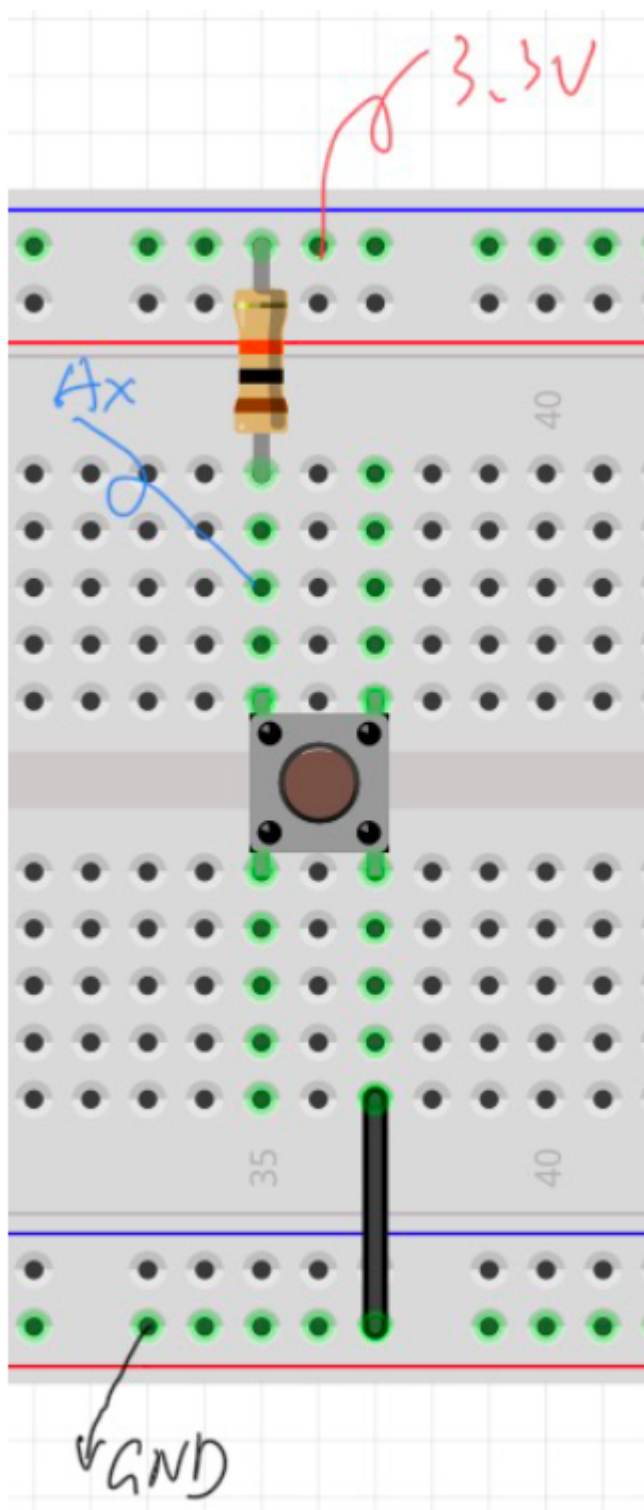
软件

- STM32CubeIDE;
- Putty

- Serial Port Utility(友善串口调试助手)

实验原理

连接电路图



- 按钮连接

TIM定时器

- STM32F1 系列中，除了互联型的产品，共有8个定时器，分为基本定时器，通用定时器和高级定时器。基本定时器TIM6和TIM7是一个16位的只能向上计数的定时器，只能定时，没有外部IO。通用定时器TIM2/3/4/5是一个16位的可以向上/下计数的定时器，可以定时，可以输出比较，可以输入捕捉，每个定

定时器有四个外部IO。高级定时器TIM1/8是一个16位的可以向上/下计数的定时器，可以定时，可以输出比较，可以输入捕捉，还可以有三相电机互补输出信号，每个定时器有8个外部IO。

- 基本定时器的核心是时基，通用计时器和高级定时器也有。
- 时钟源
 - 定时器时钟TIMxCLK，即内部时钟CK_INT，经APB1预分频器后分频提供，如果APB1预分频系数等于1，则频率不变，否则频率乘以2，库函数中APB1预分频的系数是2，即 $PCLK1=36M$ ，所以定时器时钟 $TIM_xCLK=36*2=72M$ 。
- 计数器时钟
 - 定时器时钟经过PSC预分频器之后，即CK_CNT，用来驱动计数器计数。PSC是一个16位的预分频器，可以对定时器时钟TIMxCLK进行1~65536 之间的任何一个数进行分频。具体计算方式为： $CK_CNT=TIM_xCLK/(PSC+1)$ 。
- 计数器
 - 计数器 CNT 是一个 16 位的计数器，只能往上计数，最大计数值为 65535。当计数达到自动重载寄存器的时候产生更新事件，并清零从头开始计数。
- 自动重载寄存器
 - 自动重载寄存器ARR是一个16位的寄存器，这里面装着计数器能计数的最大数值。当计数到这个值的时候，如果使能了中断的话，定时器就产生溢出中断。
- 定时时间的计算
 - 定时器的定时时间等于计数器的中断周期乘以中断的次数。计数器在 CK_CNT的驱动下，
 - 计一个数的时间则是CK_CLK的倒数，等于： $1/(TIM_xCLK/(PSC+1))$ ，
 - 产生一次中断的时间则等于： $1/(CK_CLK * ARR)$ 。如果在中断服务程序里面设置一个变量time，用来记录中断的次数，
 - 那么就可以计算出我们需要的定时时间等于： $1/CK_CLK *(ARR+1)*time$ 。

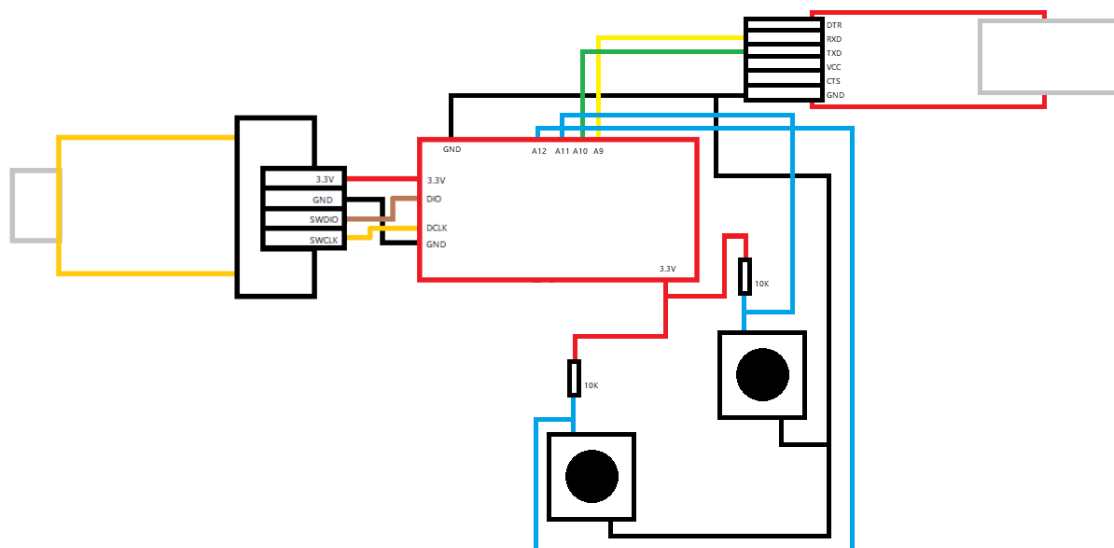
码表

- PA12连接的按钮表示车轮转的圈数，通过设置计数器可以得到里程，通过定时器中断得到的时间可以计算出速度；PA11连接的按钮切换模式，模式一在串口输出里程，模式二在串口输出速度。

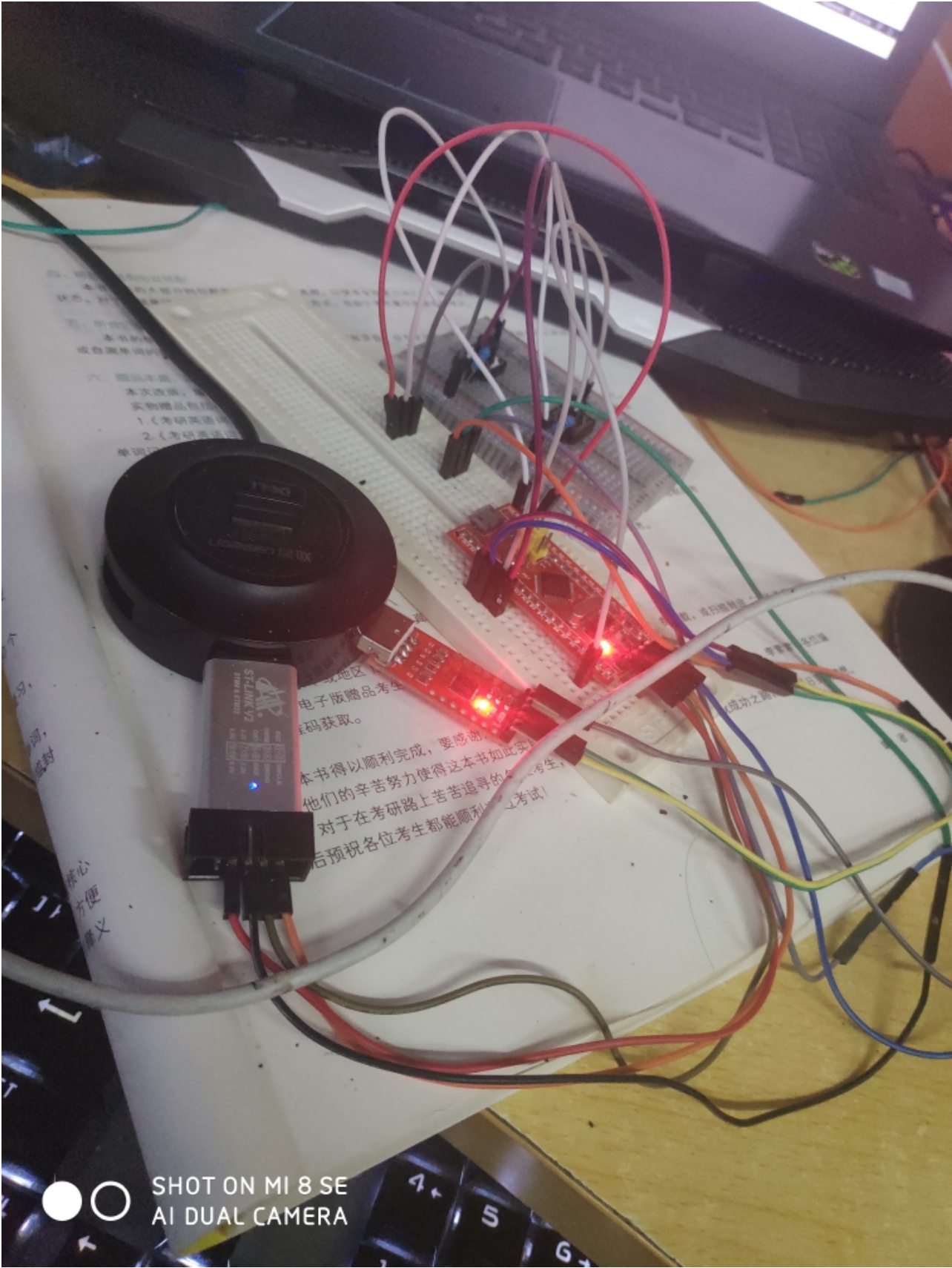
实验步骤

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



配置GPIO

PA11	n/a	n/a	Input m...	Pull-up	n/a	Mode	<input checked="" type="checkbox"/>
PA12	n/a	n/a	Extern...	Pull-up	n/a	Data	<input checked="" type="checkbox"/>

配置定时器

TIM4 Mode and Configuration

Mode

Slave Mode Disable

Trigger Source Disable

☒ Internal Clock

Channel1 Disable

Channel2 Disable

Channel3 Disable

Channel4 Disable

Combined Channels Disable

Configuration

Reset Configuration

☒ NVIC Settings

☒ DMA Settings

☒ Parameter Settings

☒ User Constants

Configure the below parameters :

↶
↷

i

▼ Counter Settings

Prescaler (PSC - 16 bits.. 7200

Counter Mode Up

Counter Period (AutoRel.. 1999

Internal Clock Division (... No Division

Auto Reload Disable

Reset Configuration

☒ NVIC Settings

☒ DMA Settings

☒ Parameter Settings

☒ User Constants

NVIC Interrupt Table	Enabled	Preemption Priority	Sub
TIM4 global interrupt	<input checked="" type="checkbox"/>	0	0

- Prescaler为7200，Counter Mode为1999，所以定时器是大约每0.2s中断一次。

编写前后台程序，以中断处理时钟定时器和其中一个按钮（模拟钢圈计数），以主循环读取按钮的变化值并根据定时的值做输出，第二个按钮（模拟模式切换）由主程序轮询判断，用以改变输出的数据（速度还是里程）。程序通过串口输出。

- 思路
 - 每0.2s算一次速度，用当前计数器的值减去上一个0.2s的计数器的值，乘以车轮长度除以0.2s即可算出即时速度，车轮长度按2.56m计算。
- 代码

```

/* stm32f1xx_it.c中插入对PA11按钮去抖动 */
void Key_Scan(void)
{
    if(HAL_GPIO_ReadPin(Mode_GPIO_Port,Mode_Pin) == GPIO_PIN_RESET)
    //判断有没有按下PA11按钮
    {
        char buf[64];
        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; // 按钮状态翻转
            pushFlag = 0;
        }
        else
        {
            BtnChangeFlag = 0;
        }
        BtnState = 0;
    }
}

/* main.c 定义用于计数和模式切换的全局变量 */
uint16_t bike_watch_counter = 0, bike_watch_last_counter = 0;
uint8_t bike_watch_flag = 0, bike_watch_mode = 0, bike_time_flag = 0;
/* main.c的main函数中 */
HAL_TIM_Base_Start_IT(&htim4); // 启动定时器TIM4
char buf[64];
uint16_t length;
if(BtnChangeFlag == 1) // 模式切换
{
    HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);
    bike_watch_mode ^= 1;
    length = sprintf(buf, "Mode Change\n\r", 13);
    HAL_UART_Transmit_DMA(&huart1, (uint8_t *)buf, length);
    HAL_Delay(1);
}

```

```

if(bike_watch_flag == 1 && bike_time_flag == 1)           // 码表里程和速度计算和显示
{
    bike_watch_flag = 0;
    bike_time_flag = 0;
    if(bike_watch_mode == 1)           // 里程
    {
        length = sprintf(buf, "Distance: %f m\r\n", (float)bike_watch_counter*2.56);
        HAL_UART_Transmit_DMA(&huart1, (uint8_t *)buf, length);
    }
    else           // 速度
    {
        if(bike_watch_time != bike_watch_time_last)
        {
            length = sprintf(buf, "Speed: %f m/s\r\n", (bike_watch_counter -
bike_watch_last_counter)*2.56*10/0.2);
            HAL_UART_Transmit_DMA(&huart1, (uint8_t *)buf, length);
        }
    }
    bike_watch_last_counter = bike_watch_counter;
}
/* main.c的定时器回调函数 */
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    if (htim->Instance == TIM2)
    {
        bike_time_flag = 1;
    }
}
/* main.c的引脚中断回调函数 */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    if(GPIO_Pin == GPIO_PIN_12)
    {
        bike_watch_counter ++;
        bike_watch_flag = 1;
    }
}

```

- 结果

```
[22:20:18.407] Distance: 5.120000 m
[22:20:18.539] Distance: 7.680000 m
[22:20:18.739] Distance: 10.240000 m
[22:20:18.939] Distance: 12.800000 m
[22:20:19.138] Distance: 15.360000 m
[22:20:19.339] Distance: 20.480000 m
[22:20:19.539] Distance: 23.040000 m
[22:20:19.739] Distance: 28.160000 m
[22:20:19.939] Distance: 33.280000 m
[22:20:20.139] Distance: 35.840000 m
[22:20:20.339] Distance: 38.400000 m
[22:20:20.539] Distance: 40.960000 m
[22:20:20.776] Mode Change
[22:20:21.077] Speed: 12.800000 m/s
[22:20:21.271] Speed: 12.800000 m/s
[22:20:21.428] Speed: 12.800000 m/s
[22:20:21.594] Speed: 12.800000 m/s
[22:20:21.760] Speed: 12.800000 m/s
[22:20:21.939] Speed: 25.600000 m/s
[22:20:22.139] Speed: 12.800000 m/s
[22:20:22.339] Speed: 25.600000 m/s
[22:20:22.540] Speed: 25.600000 m/s
[22:20:22.739] Speed: 25.600000 m/s
[22:20:22.939] Speed: 12.800000 m/s
[22:20:23.139] Speed: 38.400000 m/s
[22:20:23.339] Speed: 12.800000 m/s
[22:20:23.812] Mode Change
[22:20:23.859] Distance: 92.160000 m
[22:20:24.236] Distance: 94.720000 m
```

扩展内容

用定时器的外部触发来测量时间。

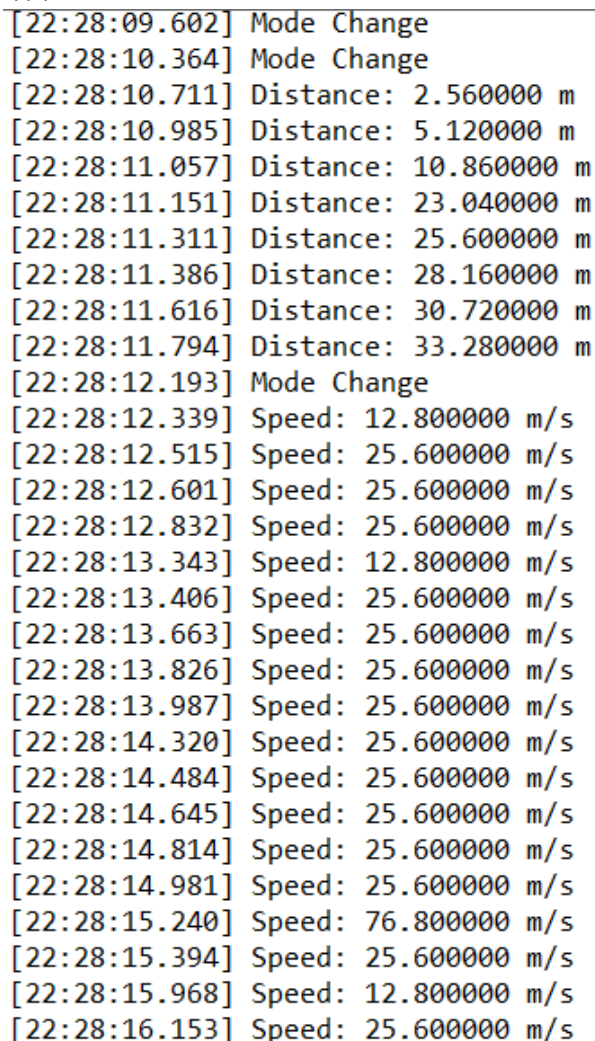
- 代码

```
if(bike_watch_flag == 1)          // 码表里程和速度计算和显示
{
    bike_watch_flag = 0;
    if(bike_watch_mode == 1)      // 里程
    {
        length = sprintf(buf, "Distance: %f m\r\n", (float)bike_watch_counter*2.56);
        HAL_UART_Transmit_DMA(&huart1, (uint8_t *)buf, length);
    }
    else                          // 速度
    {
        if(bike_watch_time != bike_watch_time_last)
        {
            length = sprintf(buf, "Speed: %f m/s\r\n", (bike_watch_counter -
bike_watch_last_counter)*2.56*10/(bike_watch_time - bike_watch_time_last));    //
测量时间
            HAL_UART_Transmit_DMA(&huart1, (uint8_t *)buf, length);
        }
    }
}
```



```
    }  
    }  
    bike_watch_last_counter = bike_watch_counter;  
}  
/* main.c的定时器回调函数 */  
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)  
{  
    if (htim->Instance == TIM4)  
    {  
        bike_watch_time += 1;    // 测量时间  
    }  
}
```

- 结果



```
[22:28:09.602] Mode Change  
[22:28:10.364] Mode Change  
[22:28:10.711] Distance: 2.560000 m  
[22:28:10.985] Distance: 5.120000 m  
[22:28:11.057] Distance: 10.860000 m  
[22:28:11.151] Distance: 23.040000 m  
[22:28:11.311] Distance: 25.600000 m  
[22:28:11.386] Distance: 28.160000 m  
[22:28:11.616] Distance: 30.720000 m  
[22:28:11.794] Distance: 33.280000 m  
[22:28:12.193] Mode Change  
[22:28:12.339] Speed: 12.800000 m/s  
[22:28:12.515] Speed: 25.600000 m/s  
[22:28:12.601] Speed: 25.600000 m/s  
[22:28:12.832] Speed: 25.600000 m/s  
[22:28:13.343] Speed: 12.800000 m/s  
[22:28:13.406] Speed: 25.600000 m/s  
[22:28:13.663] Speed: 25.600000 m/s  
[22:28:13.826] Speed: 25.600000 m/s  
[22:28:13.987] Speed: 25.600000 m/s  
[22:28:14.320] Speed: 25.600000 m/s  
[22:28:14.484] Speed: 25.600000 m/s  
[22:28:14.645] Speed: 25.600000 m/s  
[22:28:14.814] Speed: 25.600000 m/s  
[22:28:14.981] Speed: 25.600000 m/s  
[22:28:15.240] Speed: 76.800000 m/s  
[22:28:15.394] Speed: 25.600000 m/s  
[22:28:15.968] Speed: 12.800000 m/s  
[22:28:16.153] Speed: 25.600000 m/s
```

实验心得

- 本次实验较为简单，主要延续了实验3中用到的引脚中断和定时器中断。
- 实验中需要注意的有：
 - 定时器配置
 - 引脚中断配置