Smart Hardware Design

Conception de matériel intelligent

Diseño de hardware inteligente

スマートハードウェア設計

智能硬件设计

第六章 智能硬件的输出控制

تصميم الأجهزة الذكية

Slimme hardwareontwerpen

Σχεδίαση έξυπνου υλικού

大连理工大学-朱明

Progettazione di hardware intelligente

스마트 하드웨어 설계

Smart-Hardware-Design

Проектирование умного оборудования

6.0 思考回顾

- [6.0.0] 智能硬件的硬件
  - 智能硬件的六大硬件组成

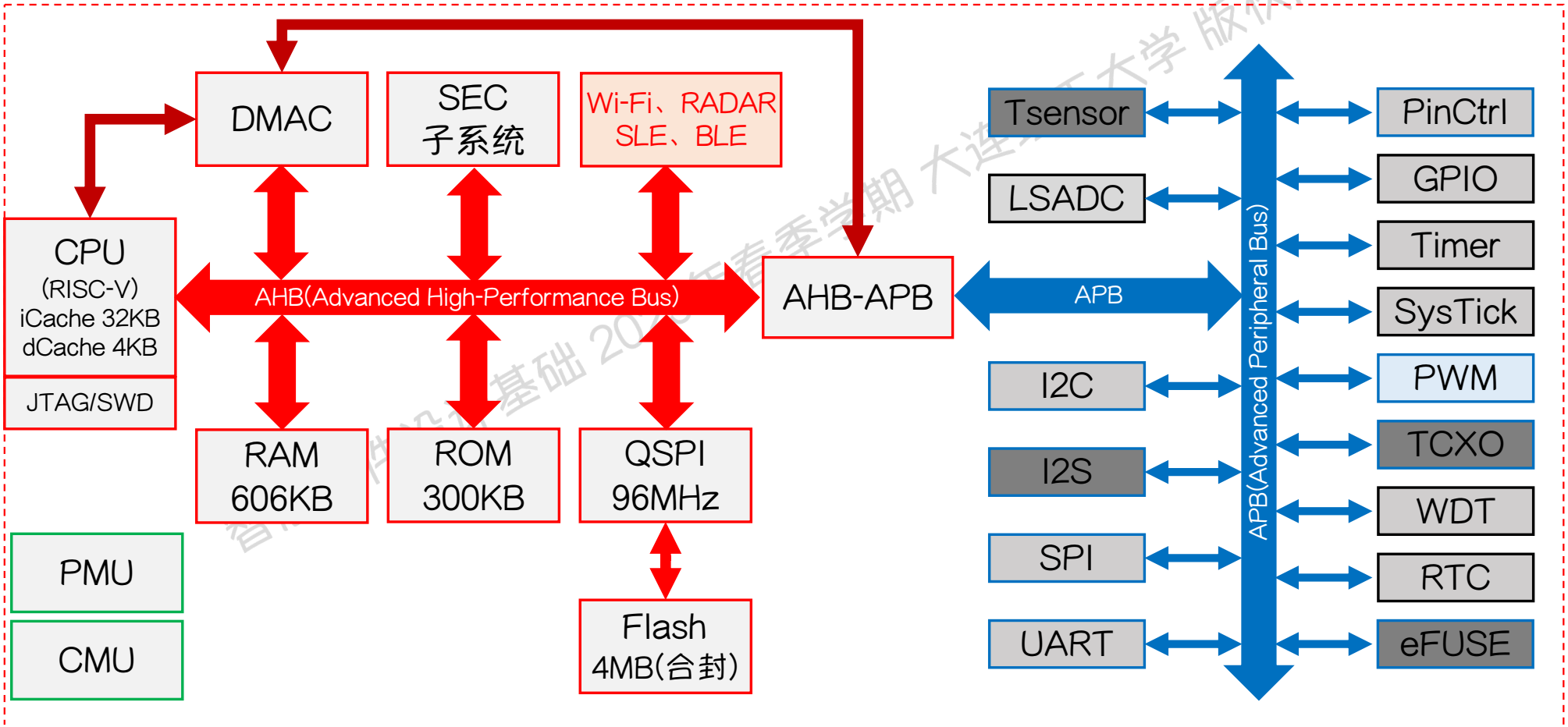
输入/感知	感知外部/内部状态，产生感知数据
通信	智能硬件之间或与外部系统的联网和数据通信
存储	保存操作系统、应用程序、数据和配置等内容
计算	进行运算和决策，控制智能硬件系统工作
输出/执行	输出计算结果，控制外部设备等

电源系统

智能硬件大多是电池供电设备，(电池)电源系统为所有系统设备供电

6.0 思考回顾

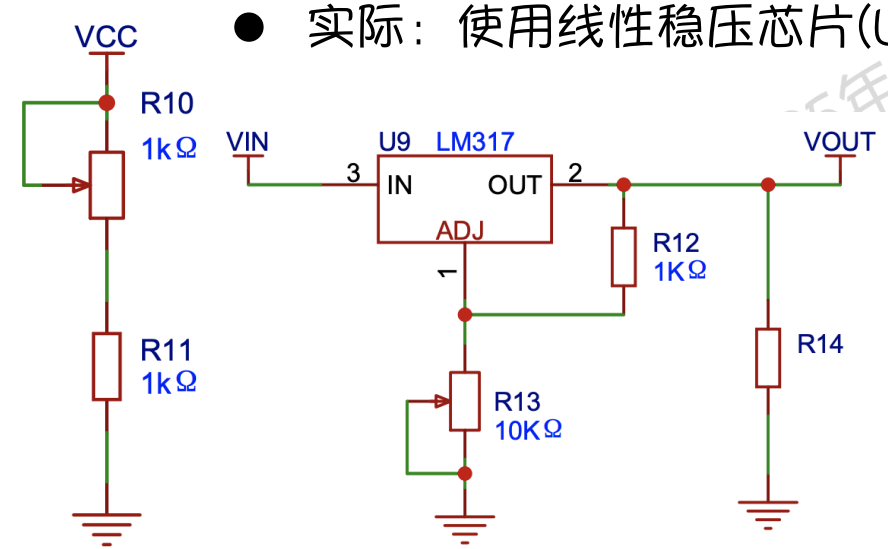
- [6.0.0] 智能硬件的SoC内部结构(华为海思WS63)



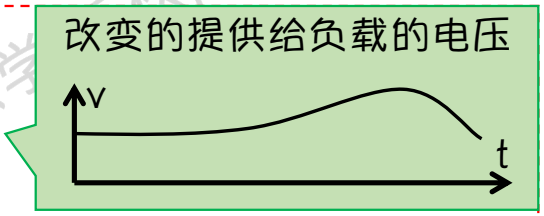
6.1 信息系统的功率控制

●[6.1.1] 线性电压调节功率控制

- 基于线性电压调节器的功率控制
  - 调节原理：通过等效电阻机制改变负载供电电压
    - 理论：调节R10的阻值，就可以改变R11的功率
    - 实际：使用线性稳压芯片(U9)电路(R12/R13)完成负载(R14)的电压调节



分压电阻调节　　三端稳压芯片调节



目前主要使用的线性稳压器是LDO(Low Dropout Regulator，低压差线性稳压器)，具有较低的压降(Dropout Voltage，即左图中的V_{IN}-V_{OUT})

优点：①提供非常稳定的输出电压，适用于精密仪器等对电压要求严格的设备②线性调节器具有较低的电磁干扰(EMI)，适用于噪声敏感设备③设计相对简单，组件少，成本低

缺点：①通过分压形式来实现电压调整，压差越大，效率越低②效率低则意味着有部分能量转化为热能，需要额外的散热措施来将设备温度控制在合理范围内，防止过热损坏

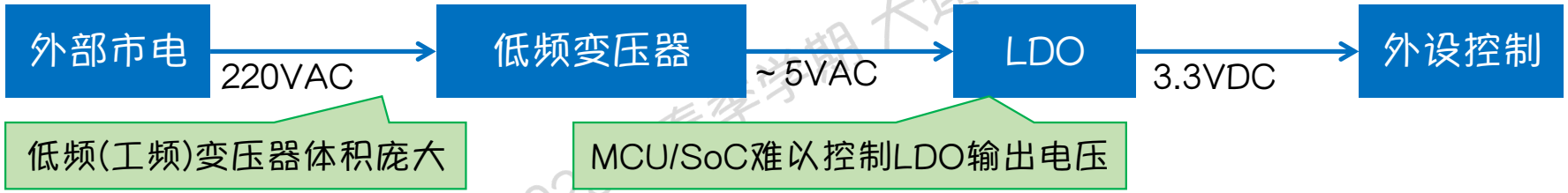
6.1 信息系统的功率控制

●[6.1.1] 线性电压调节功率控制

- 基于线性电压调节器的功率控制

该过程实际上也是智能硬件系统的供电系统

  - 从市电到智能硬件系统的低压直流(3.3V)的变换过程复杂



  - 线性电压调节在非阻性负载的控制方面表现不佳：反应速度慢
    - 电感性负载：如电动机、变压器和电感元件等，它们的电流是随着时间变化(具有惯性)的，电流变化较慢，且会产生高频噪声和电压尖峰
    - 容性负载：如电池、储能电容器等，在充放电过程中具有瞬时大电流的特性，特别是在电容充电开始时，电流需求较大
    - 非线性负载：如LED或光敏电阻等，负载电流和电压之间并非简单的线性关系，导致线性调节器在调节时可能产生较大的误差或不稳定

6.1 信息系统的功率控制

●[6.1.2] 脉冲宽度功率控制

- 目前最常见的功率控制方法(没有之一)
  - 广泛应用于电源转换、电机驱动、LED调光等领域

各类开关电源(如变压器、充电器)

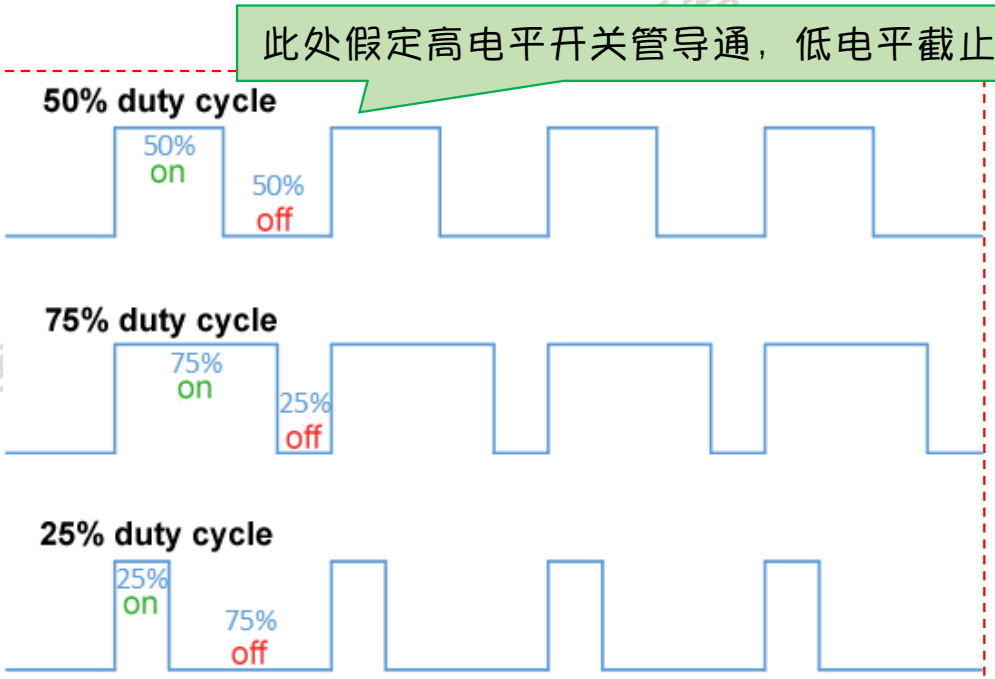
高速吹风机、手机振动电机等

各类LED照明设备(含不可调光)

- 核心思想：控制开关器件的通断时间，调节等效输出功率或电压
  - 使用一种信号(开关信号)控制另一种信号(电压)的形态：脉冲宽度调制
  - 脉冲宽度调制(PWM，Pulse Width Modulation)的主要参数
    - 波形：只有一种类型，方波信号(高低电平控制开关器件的通断)
    - 频率： PWM信号频率，表征开关器件的开关速度，受开关器件速度限制
    - 占空比(Duty Cycle)：PWM信号中高电平持续的时间比例，其直接决定了平均电压和功率输出，占空比越大，输出功率或电压越高

6.1 信息系统的功率控制

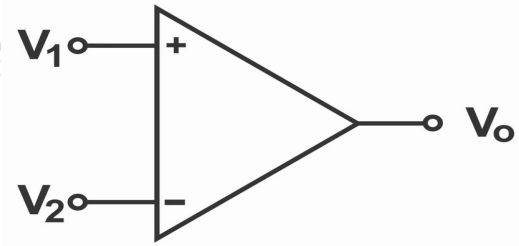
- [6.1.2] 脉冲宽度功率控制
  - PWM的占空比(Duty Cycle)
    - $Duty\ Cycle = \frac{T_{on}}{T_{total}} \times 100\%$
      - 范围：[0, 100]%
    - $T_{total} = T_{on} + T_{off}$
    - 此处：$T_{on} = T_{high}$
      - 会因为电路而有所不同
  - 占空比是比值，与频率无关
    - 频率1KHz，PWM占空比是10%，求高电平时间和低电平时间
    - 提升频率到2KHz，高电平时间和低电平时间分别是多少

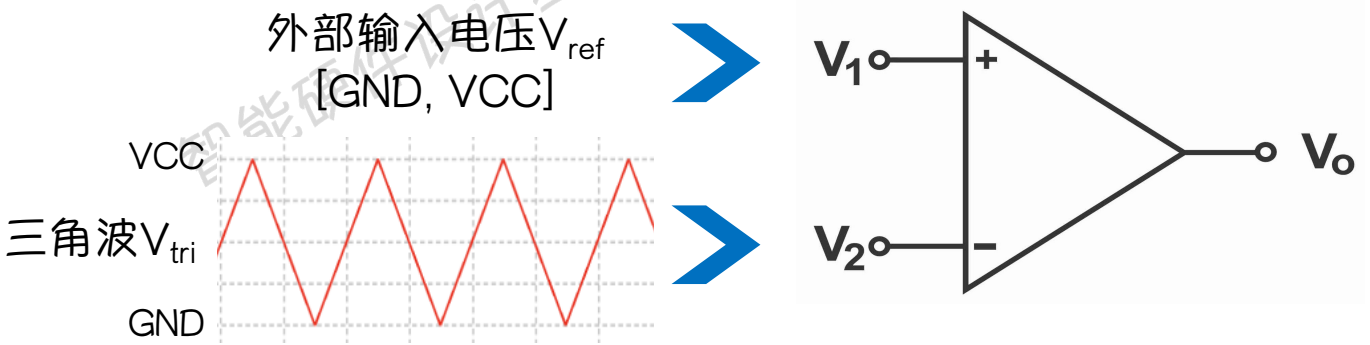


6.2 PWM信号发生机制

●[6.2.1] PWM发生电路

- 基于电压比较器的PWM发生电路
  - 电压比较器应用基础：对输入电压进行比较的元件
    - VCC=逻辑高电平“H”，GND=逻辑低电平”L”
    - 若V1>V2，Vo=VCC，逻辑高电平H
    - 若V1<V2，Vo=GND，逻辑低电平L
  - 三角波发生电路(略去，但电路很简单)

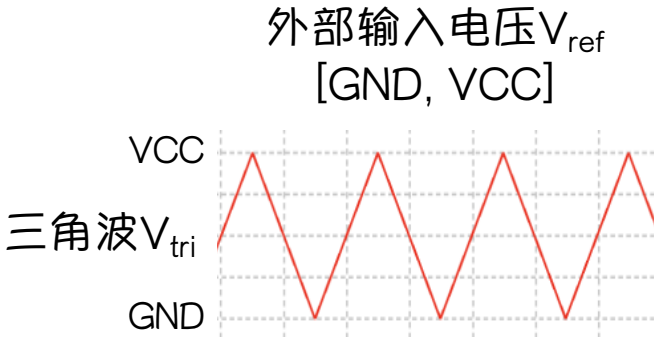


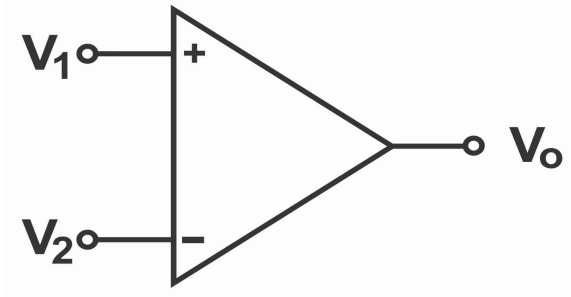


讨论
输出波形形态

6.2 PWM信号发生机制

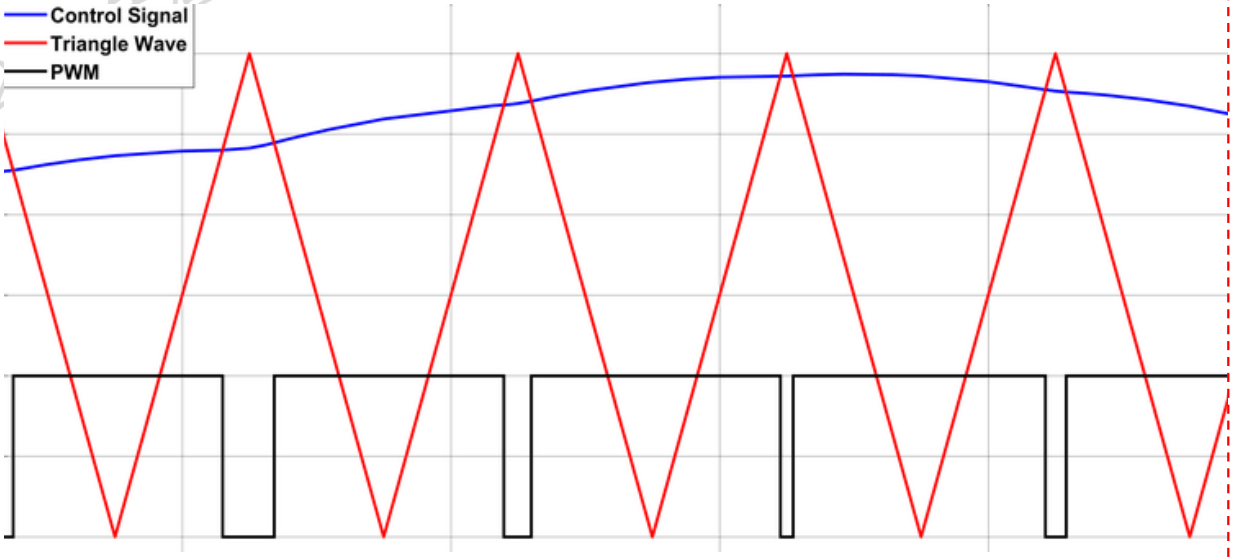
●[6.2.1] PWM发生电路





讨论
输出波形形态

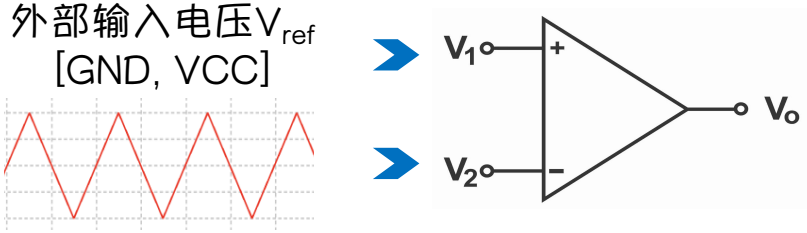
- $V_{ref}>V_{tri}$， Vo=H
- $V_{ref}<V_{tri}$， Vo=L
- $F_{tri}=F_{PWM}$
- V_{ref}↑， Duty Cycle↑
- V_{ref}↓， Duty Cycle↓



6.2 PWM信号发生机制

[6.2.1] PWM发生电路

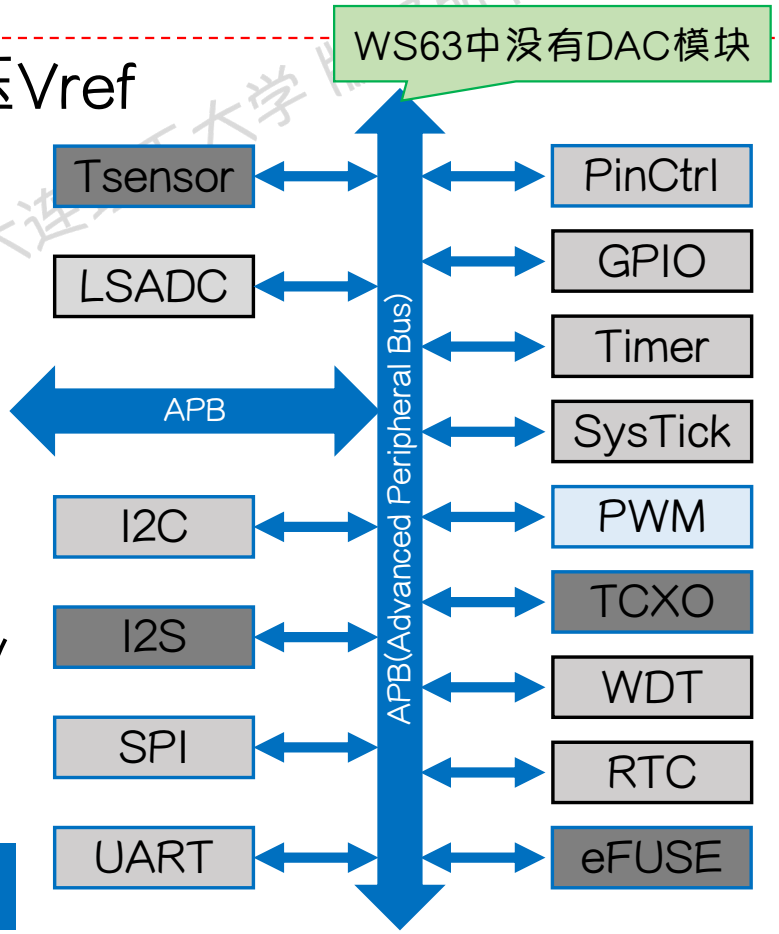
- 基于电压比较器的PWM发生电路
  - 优点：纯硬件，不需要额外控制，抗干扰性强
    - 不涉及MCU/SoC控制，系统简单，全部硬件电路实现，抗干扰性强
    - 可以实现极高的控制精度(完全连续，不存在调节级别的问题)
    - 可输出高频率的PWM信号，并保持极高的占空比精度
  - 缺点：纯硬件，人工可控程度有限
    - 无法实现程序控制的PWM调整、无法实现远程控制、无线遥控等控制功能
  - 改进方法：MCU/SoC可控制PWM的输出
    - 直接方案：引入软件进行调节



6.2 PWM信号发生机制

- [6.2.2] PWM的软硬混合发生机制
  - 如何利用MCU/SoC产生外部输入电压Vref
    - WS63的外设中不直接具备该功能
    - ADC - DAC(数字模拟转换器)
    - 回顾ADC的主要参数
      - 分辨率 ➢ 分辨率
      - 转换速度 ➢ 转换速度
      - 量程 ➢ 输出范围
    - 尝试计算DAC的输出，N=12，V_{ref}=3.3V
      - 输入数值2345，求输出电压

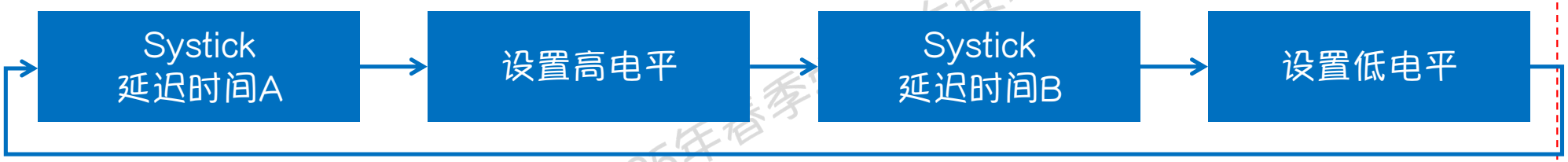
适合硬件PWM进行扩展，体积大、系统复杂



6.3 数字化PWM生成

●[6.3.1] 基于延迟的生成方法

- 利用定时器计数值+用户判断实现PWM信号输出
  - 例如，获取Systick的时间值，或者利用Systick延迟实现时间控制



- 高电平时间：A or B的时间
- 低电平时间：A or B的时间
- 周期是多少？
- 频率是多少？

可行性分析
合理性分析
性能的分析

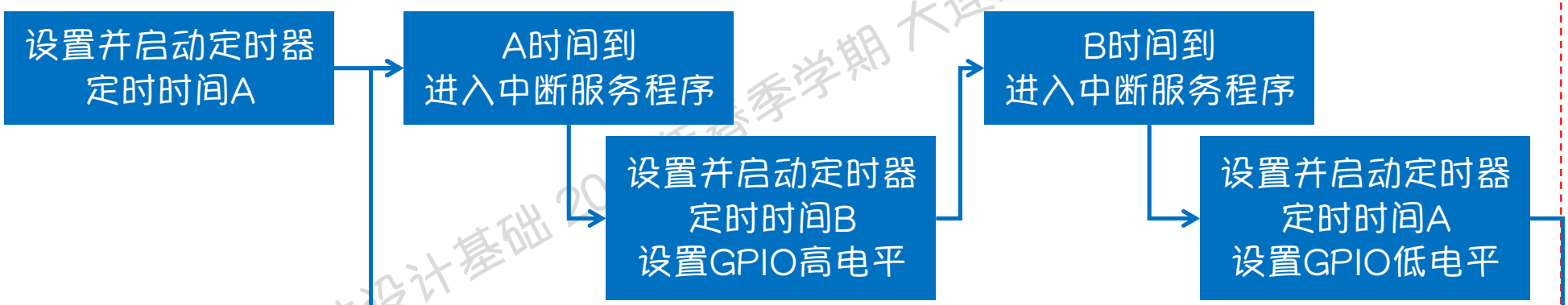
可行、不合理，没性能
智能硬件开发领域的耻辱

绝不能让宝贵的系统资源浪费在无意义的延时工作上

6.3 数字化PWM生成

[6.3.2] 基于用户控制的生成方法

- 利用定时器+中断实现PWM信号输出
  - 最直接：设置两个定时器，分别表示低电平和高电平时间



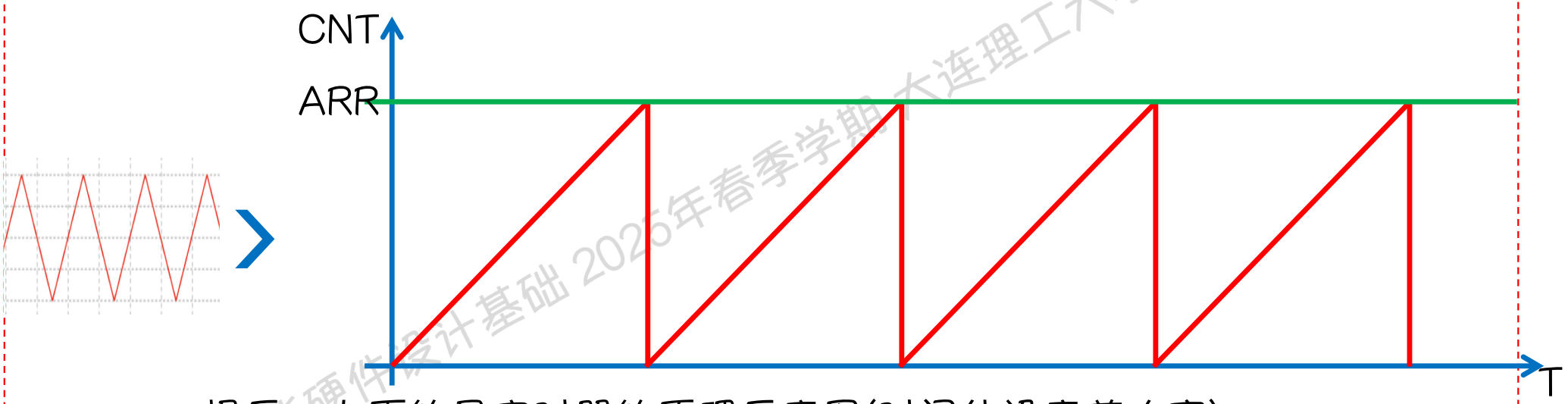
- 高电平时间：定时器A or B的时间
- 低电平时间：定时器A or B的时间
- 周期是多少？频率是多少？

可行性分析
合理性分析
性能的分析

可行
也合理
性能差

6.3 数字化PWM生成

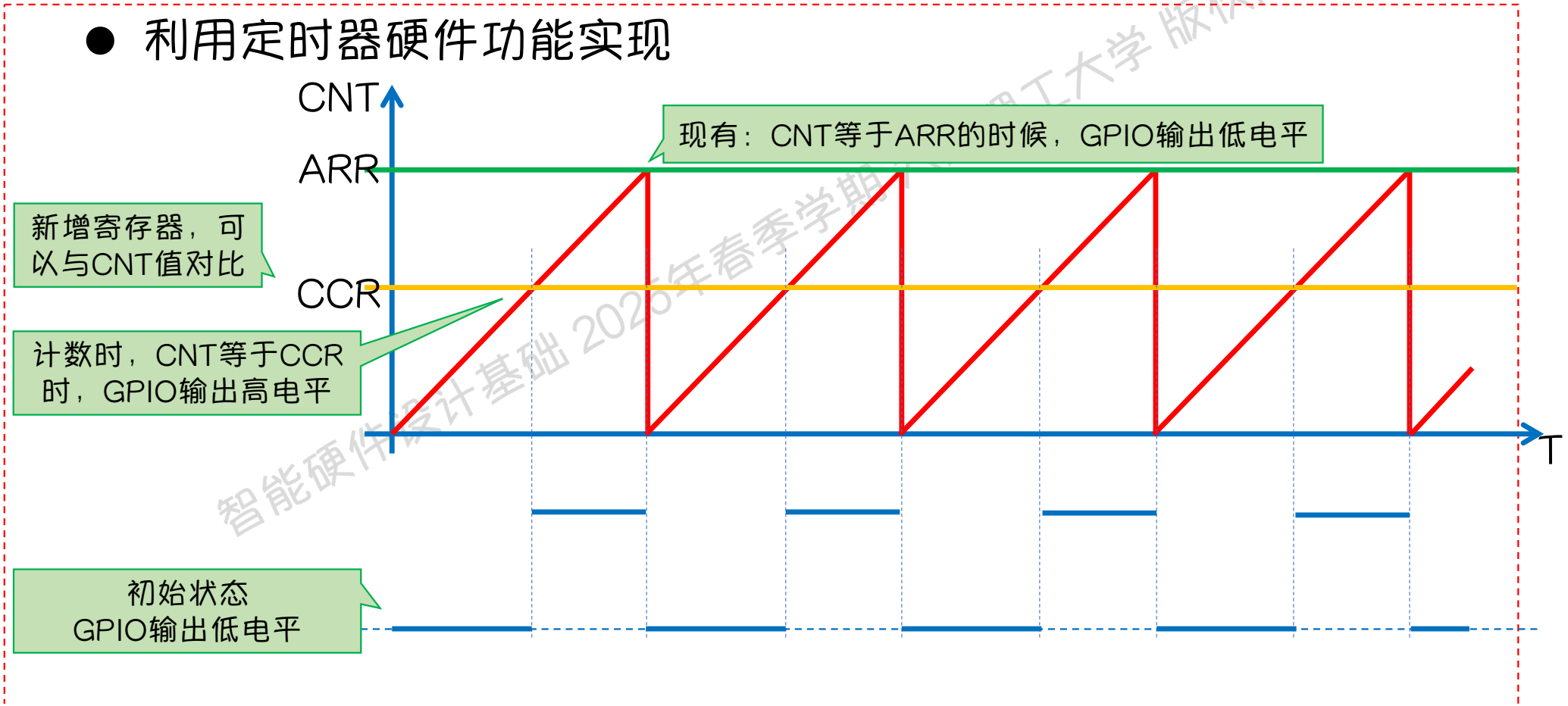
- [6.3.3] 基于定时器硬件的生成方法
  - 利用定时器硬件功能实现：回忆一下下图



  - 提示：上面的是定时器的原理示意图(时间外设章节内容)
  - 依据上述定时器的原理，设计一种基于定时器的PWM生成方式
    - 可以对定时器的硬件功能和软件结构增加一些设计

6.3 数字化PWM生成

- [6.3.3] 基于定时器硬件的生成方法
  - 利用定时器硬件功能实现



6.3 数字化PWM生成

- [6.3.3] 基于定时器硬件的生成方法
  - MCU/SoC硬件PWM的优点
    - 硬件生成：PWM在持续生成的过程中，无需系统和用户端软件干预
      - 生成PWM的定时器及内部比较功能由硬件控制，工作稳定
    - 软件配置：PWM生成前的参数配置，完全由用户设定
      - PWM信号的主要参数(频率和占空比)可以实现高度的可定制化
      - 思考1：哪一个寄存器的值决定PWM信号的频率(周期)
      - 思考2：哪一个寄存器的值决定PWM信号的占空比
  - MCU/SoC硬件PWM的缺点
    - 硬件生成，但是属于外设模块，受处理器整体稳定性(含软件)影响
    - 数字计数，PWM信号的占空比的精度有限

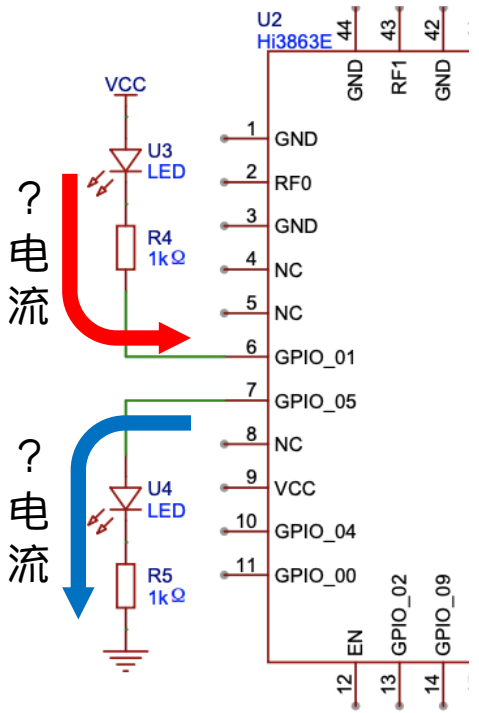
6.4 智能硬件系统中的PWM应用

[6.4.1] 智能硬件的PWM调控输出设备

- 发光类输出设备：LED为例
  - 智能硬件设备可以使用常规亮度或高亮度LED
    - GPIO控制LED亮灭：简单、高效、占用资源少
      - GPIO不发生变化时，不占用任何系统资源
      - 显示状态单一，亮度不能调节，不能实现复杂效果
    - 智能硬件设备的LED常用呼吸灯效果
      - 实现简单：PWM占空比逐渐先增加再减少的循环
      - 占用资源：不断调整PWM的占空比
  - LED亮度调整时的PWM频率问题
    - 频率过低：亮度较低时会产生闪烁
    - 频率过高：LED的结电容消耗能量

常规LED(指示灯类)
普通红/黄/绿：1.7V~，几mA
高亮红/黄/绿：3V~，几十mA

1K ~ 10KHz



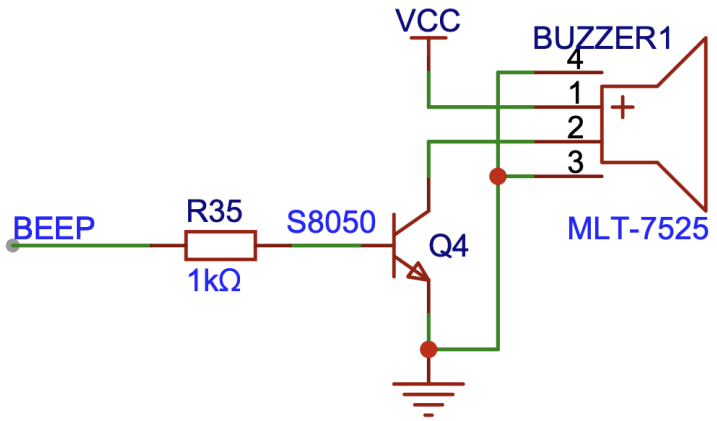
6.4 智能硬件系统中的PWM应用

[6.4.1] 智能硬件的PWM调控输出设备

- 发声类输出设备：蜂鸣器(Buzzer)为例
  - 蜂鸣器：一种常发出Bee~~或Wing~~等提示或警告等声音的电子元件
    - 工作原理：利用压电效应，电流通过压电陶瓷片时使其变形，产生声音
    - 声音类型：声音频率高，音调单一且连续(不同于喇叭)
    - 无源蜂鸣器(常用)：内部没有振荡电路，依靠外界信号控制声音属性
    - 工作电压：MCU/SoC板载常用3V或5V
    - 工作电流：几mA至几十mA

声音尖锐刺耳、音调单一
声音品质远不如普通喇叭

PWM能控制蜂鸣器的那些声音参数

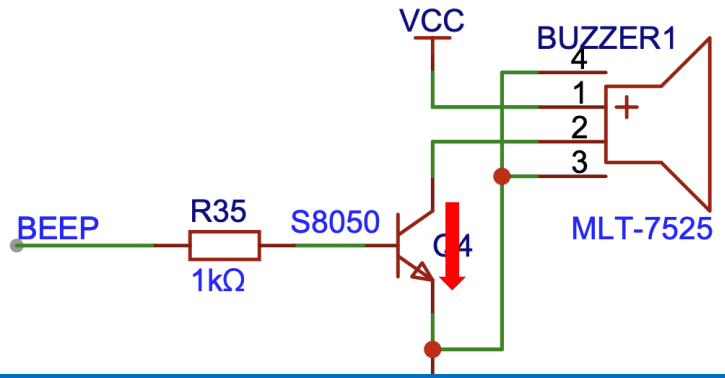


6.4 智能硬件系统中的PWM应用

[6.4.1] 智能硬件的PWM调控输出设备

- 发声类输出设备：蜂鸣器(Buzzer)为例
  - PWM控制的声音主要属性：
    - 音调(频率)：频率越高，音调越尖锐；频率越低，音调越低沉
    - 音量：占空比越高，蜂鸣器功率越大，声音越响；占空比越低，声音越弱
  - 蜂鸣器的驱动电路：
    - 难以直接使用GPIO驱动
    - 使用小功率三极管或MOSFET驱动

BEEP=1时
Q4导通，电流C->E，蜂鸣器工作
BEEP=0时，Q4截止，蜂鸣器不工作



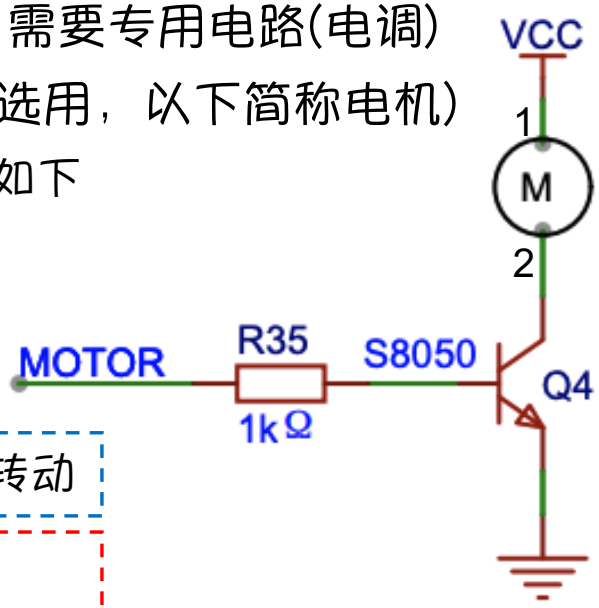
在数字系统中，三极管和MOSFET应被设计工作在导通/截止状态下，作为负载的开关使用

6.4 智能硬件系统中的PWM应用

●[6.4.1] 智能硬件的PWM调控输出设备

- 动作类输出设备：电机为例
  - 电机的作用：将电能转换为机械能，以旋转的形式输出能量、执行动作
  - 直流电机的种类：直流有刷电机、直流无刷电机
    - 直流无刷电机：结构简单、控制复杂、转速高、需要专用电路(电调)
    - 直流有刷电机：控制简单，PWM直接控制(课程选用，以下简称电机)
      - 直流电机的旋转方向受电流方向控制，假定关系如下

电流方向	电机旋转方向
1 -> 2	顺时针旋转
2 -> 1	逆时针旋转



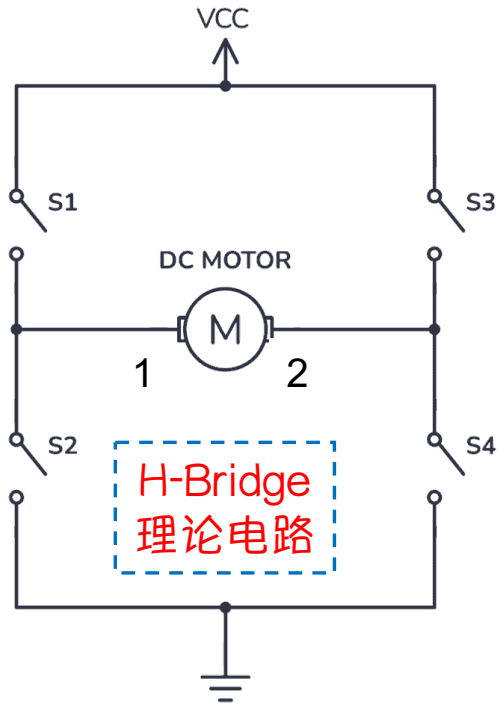
定性分析：MOTOR为高电平时，Q4导通，电机顺时针转动

思考：设计何种电路才能实现电机的方向和转速的控制

6.4 智能硬件系统中的PWM应用

●[6.4.1] 智能硬件的PWM调控输出设备

- 动作类输出设备：电机为例
  - 改变旋转方向的本质：改变流经电机的电流方向：H-Bridge



上臂S1	下臂S2	上臂S3	下臂S4	电机状态	电路状态
断	断	断	断	不工作	不工作
通	断	断	通	1->2，顺转	正常✅
断	通	通	断	2->1，逆转	正常✅
通	通	-	-	不工作	异常❌
-	-	通	通	不工作	异常❌
[S1通S2断S3通S4断]或[S1断S2通S3断S4通]				刹车	正常✅

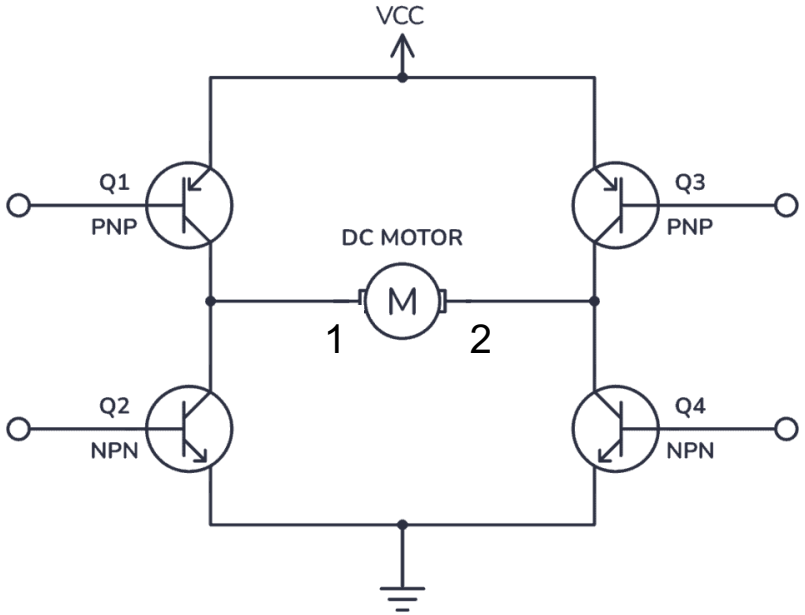
S1、S2、S3和S4：可以是三极管，也可以是MOSFET

MCU/SoC控制S1、S2、S3和S4状态切换时，注意开关顺序

6.4 智能硬件系统中的PWM应用

●[6.4.1] 智能硬件的PWM调控输出设备

- 动作类输出设备：电机为例
  - 改变旋转方向的本质：改变流经电机的电流方向：H-Bridge



基于三极管的简化的H-Bridge结构

假定Q1-Q4的开关状态与电平的关系如下

电平状态	Q1	Q2	Q3	Q4
低电平(0)	导通	截止	导通	截止
高电平(1)	截止	导通	截止	导通

Q1	Q2	Q3	Q4	电机状态	电路状态
1	0	1	0	不工作	不工作
0	0	1	PWM	1->2，顺转	正常✅
1	PWM	0	0	2->1，逆转	正常✅
其他状态(略)				不工作或刹车或异常	

思考：实现了方向控制后，如何实现转速控制

6.4 智能硬件系统中的PWM应用

●[6.4.1] 智能硬件的PWM调控输出设备

- 动作类输出设备：电机为例
  - 改变旋转方向的本质：改变流经电机的电流方向：H-Bridge
    - 结构简单、但控制逻辑和保护逻辑复杂、独立元件体积大

编程的重要原则：高内聚、低耦合

■高内聚指的是一个模块或类内的功能相互之间紧密相关，模块内部的各个部分协作完成单一的、明确的任务。高内聚意味着模块内部的职责清晰，相关性强，模块能够独立地完成某个特定功能

■低耦合意味着各个模块之间的依赖关系尽可能少。模块之间通过简洁、明确的接口进行交互，减少不必要的直接依赖，从而降低系统的复杂性和模块之间的相互影响

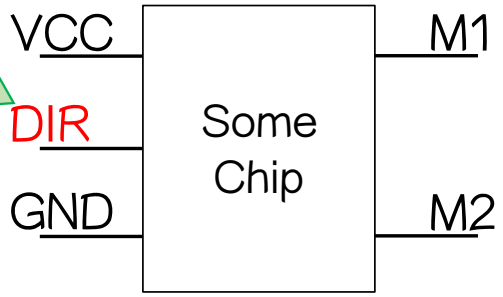
运用到智能硬件电路设计或芯片功能设计领域

电机驱动电路(芯片)的功能设计目标

1 建立内部的各项控制和保护功能

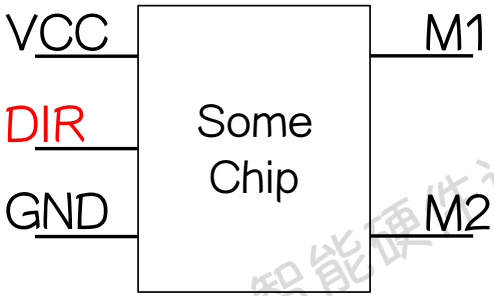
2 简化MCU/SoC信号的控制逻辑

方案不合理
无法调速
无法停止

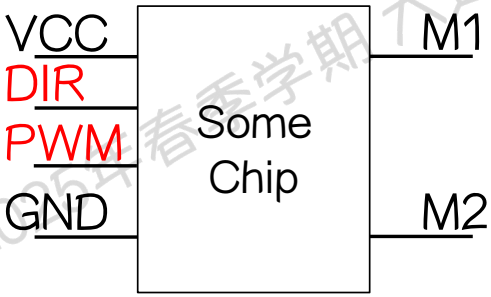


6.4 智能硬件系统中的PWM应用

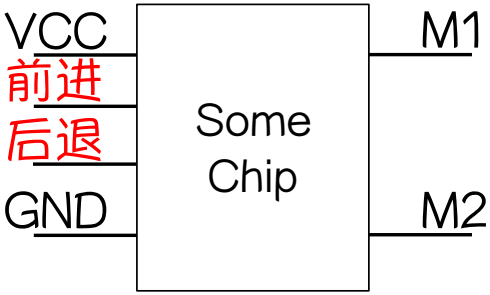
- [6.4.1] 智能硬件的PWM调控输出设备
  - 动作类输出设备：电机为例
    - 改变旋转方向的本质：改变流经电机的电流方向：H-Bridge



方案不合理
无法调速和停止



改进方案1：
增加1个引脚，输入PWM信号，
用于调整电机的转速
（具体内部实现机制略）



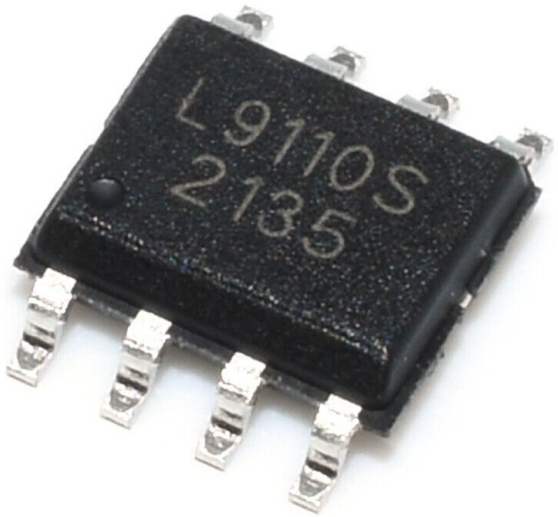
实践平台方案

改进方案2：
增加1个引脚，使两个引脚分别
用于控制电机的正转和反转
（具体内部实现机制略）

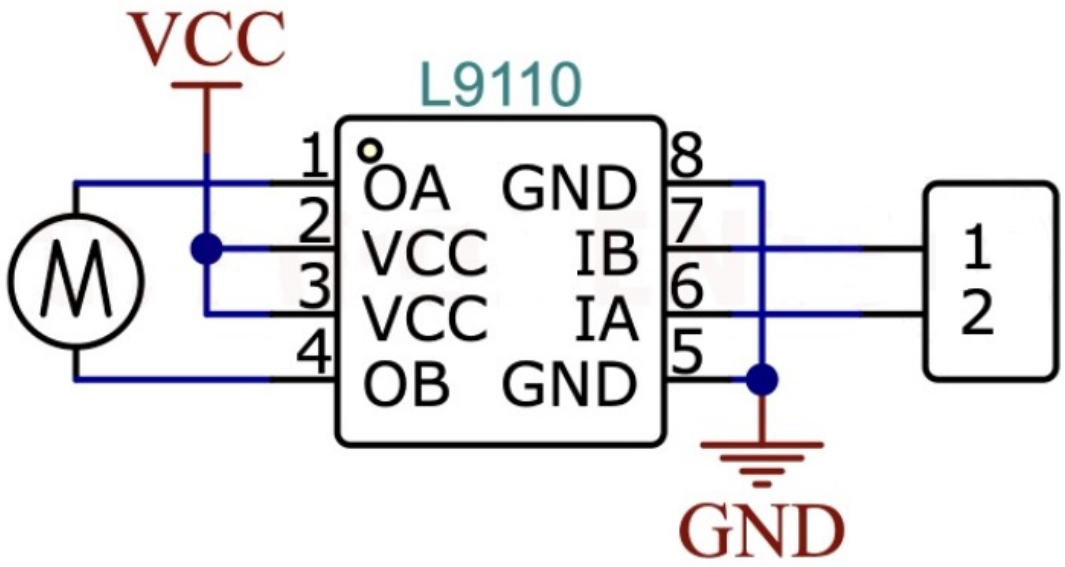
6.4 智能硬件系统中的PWM应用

●[6.4.1] 智能硬件的PWM调控输出设备

- 动作类输出设备：电机为例
  - 改变旋转方向的本质：改变流经电机的电流方向：H-Bridge芯片L9110S



H-Bridge芯片L9110S外观
6.3mm×5.0mm×1.8mm

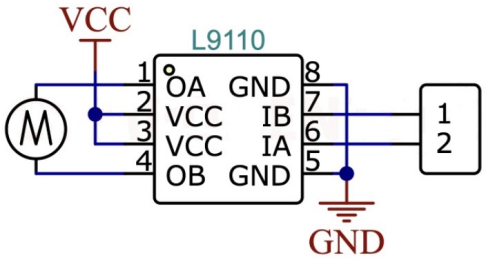


L9110S内部集成逻辑控制电路、具备高效、低压工作等特性

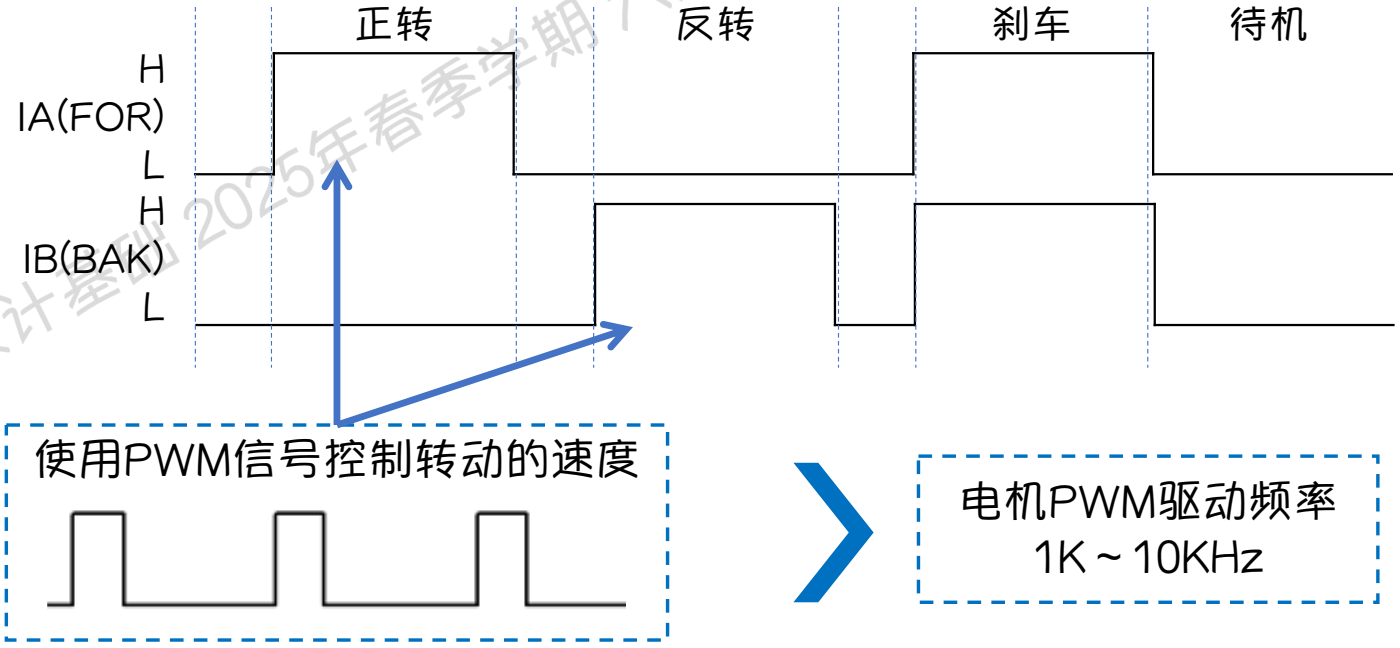
6.4 智能硬件系统中的PWM应用

●[6.4.1] 智能硬件的PWM调控输出设备

- 动作类输出设备：电机为例
  - 改变旋转方向的本质：改变流经电机的电流方向：H-Bridge芯片L9110S



1	OA	正转输出
2	VCC	电源 (2.0~5.5V)
3	VCC	
4	OB	反转输出
5	GND	GND
6	IA(FOR)	正转逻辑输入
7	IB(BAK)	反转逻辑输入
8	GND	GND



6.5 MCU/SoC的PWM系统

●[6.5.1] 与定时器共用的PWM系统

● 与定时器共用的PWM信号生成系统：STM32某型为例

选择具有PWM功能的定时器 → 计算PWM要求配置定时器参数 → 配置PWM输出复用引脚 → 启动定时器

STM32 MCU内部多组定时器，配置不同、功能不同，例如(其中TIM是官方文档对定时器的命名)

- TIM1和TIM8：支持自动重装载(ARR)，16位计数器(CNT)，16位预分频器(PSC)
- TIM3和TIM4：支持自动重装载(ARR)，16位计数器(CNT)，16位预分频器(PSC)
- TIM2和TIM5：支持自动重装载(ARR)，32位计数器(CNT)，16位预分频器(PSC)
- TIM9至TIM14：支持自动重装载(ARR)，16位计数器(CNT)，16位预分频器(PSC)

选择16位计数器或32位计数器会有何区别

```
TIM_HandleTypeDef htim2;                                    //定义用于设置的结构体
htim2.Instance = TIM2;
htim2.Init.Prescaler = 72-1;                                //预分频器PSC=72：72MHz/72=1MHz
htim2.Init.CounterMode = TIM_COUNTERMODE_UP;                //增计数模式
htim2.Init.Period = 1000-1;                                 //实际ARR=1000, 1kHz的PWM频率
htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;//不分频
HAL_TIM_PWM_Init(&htim2);
```

完成了吗?

6.5 MCU/SoC的PWM系统

●[6.5.1] 与定时器共用的PWM系统

● 与定时器共用的PWM信号生成系统：STM32某型为例

选择具有PWM功能的定时器 → 计算PWM要求配置定时器参数 → 配置PWM输出复用引脚 → 启动定时器

PWM除了频率之外，还必须要设置占空比

```
TIM_OC_InitTypeDef sConfigOC;
sConfigOC.OCMode = TIM_OCMODE_PWM1;
sConfigOC.Pulse = 500;                                  //CCR = 500, 50% 占空比
sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
HAL_TIM_PWM_ConfigChannel(&htim2, &sConfigOC, TIM_CHANNEL_1);
```

```
HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
```

```
GPIO_InitTypeDef GPIO_InitStruct;
GPIO_InitStruct.Pin = GPIO_PIN_0;
GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
```

STM32的代码简单了解即可，STM32的TIM和PWM都需要人工将时间换算为寄存器值

6.5 MCU/SoC的PWM系统

- [6.5.2] WS63的PWM系统
  - WS63为独立PWM信号生成系统，有如下特性
    - 8路独立的PWM输出通道，全部独立配置
    - API支持低电平时间和高电平时间独立设置
    - 支持PWM周期数设定，支持重复发送，支持发送完成中断等特性
  - WS63 PWM的使用流程(应先执行init)

API有独特特点
配置方式也不同

配置PWM属性的结构体 → 设置输出PWM的复用引脚 → 开启指定的PWM通道 → 设置PWM分组启动PWM分组

前面哪些外设配置需要使用结构体

WS63配置PWM自身属性的结构体应该包括哪些内容(STM32配置了什么)
①低电平时间：uint32_t low_time
②高电平时间：uint32_t high_time
③相位偏移：uint32_t offset_time
④输出周期数：uint16_t cycles
⑤重复发送：bool repeat

6.5 MCU/SoC的PWM系统

[6.5.2] WS63的PWM系统

WS63 PWM的使用流程(应先执行init)

配置PWM属性的结构体 → 设置输出PWM的复用引脚(略) → 开启指定的PWM通道 → 设置PWM分组启动PWM分组

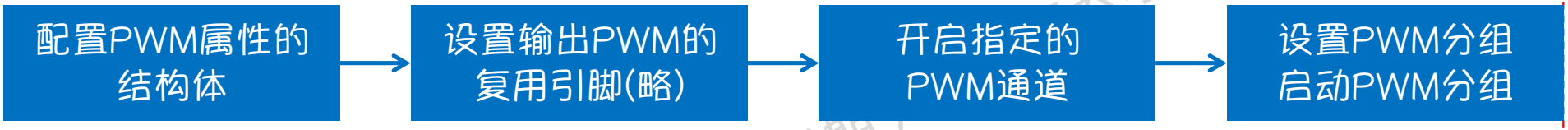
```
typedef struct pwm_config {
    uint32_t low_time;
    uint32_t high_time;
    uint32_t offset_time;
    uint16_t cycles;
    bool repeat;
} pwm_config_t;
```

例如：
pwm_config_t LEDConfig = {100, 100, 0, 0, true};

- errcode_t uapi_pwm_open(uint8_t channel,
  const pwm_config_t *cfg);
  - uint8_t channel：PWM通道，有效范围[0, 7]
  - const pwm_config_t *cfg：PWM配置结构体
    - 三个时间元素，如low_time，实际上的时间是 $low\_time \times T$
    - 课程使用的WS63版本的T=12.5ns
    - cycles为PWM信号周期数，有效范围[0, 32767]

6.5 MCU/SoC的PWM系统

- [6.5.2] WS63的PWM系统
  - WS63 PWM的使用流程(应先执行init)



    - 使用分组可以统一PWM输出，分组中最少要有一个通道
    - errcode_t uapi_pwm_set_group(uint8_t group,
      const uint8_t *channel_set, uint32_t channel_set_len);
      - uint8_t group：分组ID，有效范围[0, 7]
      - const uint8_t *channel_set：包含有该分组中成员通道的数组
      - uint32_t channel_set_len：该分组中的通道数量
    - errcode_t uapi_pwm_start_group(uint8_t group)：启动指定分组

6.5 MCU/SoC的PWM系统

[6.5.2] WS63的PWM系统

WS63 PWM的其他API

①： void uapi_pwm_deinit(void)	去初始化PWM
②： errcode_t uapi_pwm_stop(uint8_t channel)	停止指定通道的PWM信号输
③： uint32_t uapi_pwm_get_frequency(uint8_t channel)	获取指定通道的PWM信号工
④： errcode_t uapi_pwm_isr(uint8_t channel)	清除指定通道的PWM中断标
⑤： errcode_t uapi_pwm_register_interrupt(uint8_t channel, pwm_callback_t callback)	注册或去注册指定通道PWM的中断回调
⑥： errcode_t uapi_pwm_unregister_interrupt(uint8_t channel)	去注册指定通道PWM的中断回调
⑦： errcode_t uapi_pwm_clear_group(uint8_t group)	清除PWM通道的分组
⑧： errcode_t uapi_pwm_stop_group(uint8_t group)	停止指定分组的PWM

6.7 本章作业

- [6.7.0] 作业与思考

1. 基于线性电压调节器的功率控制有哪些明显的缺点
2. 脉冲宽度功率控制的核心思想是什么
3. 脉冲宽度调制PWM信号的主要参数有哪些
4. 简要说明基于三角波和电压比较器的PWM信号发生机制
5. 利用本章第15页波形生成PWM信号时，若定时器的时钟频率为1MHz，ARR=1000，CCR=500，问PWM信号频率和占空比
6. 简述设置WS63的GPIO_08所在引脚持续输出频率为10KHz，占空比25%的PWM信号的程序初始化过程，以及关键参数设置值
7. 思考：本章第22页电路中，上臂同时导通或者下臂同时导通时，理论上需要实现刹车功能，但能否实现这一功能，说明原因

6.6 闭环控制方法

●[6.6.1] 闭环控制系统

- 闭环控制(Closed-loop Control)，通过实时检测系统的输出，将其与目标值进行比较，调整输入，使系统保持在期望状态
  - 开环系统(Open-loop System)是一种无反馈的控制系统，即控制器根据预设的输入信号直接控制执行器，而不会监测或调整系统的输出
  - 开环系统的输入不受输出的影响，因此系统不会自动纠正误差

与闭环系统对应

输入量(需求) → 控制器(SoC) → 执行器(电机) → 系统输出(轮子)

结构简单：不需要反馈系统，容易实现
成本较低：硬件需求少，无需反馈设备
响应速度快：无需计算误差，快速执行
应用场景简单：定时路灯、定时开关等

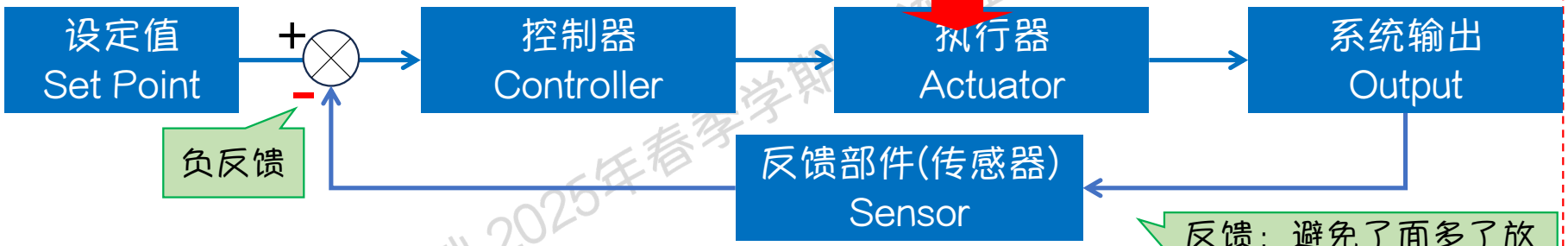
精度较低：误差不可修正，易受外部环境影响
适应性差：系统无法根据实际情况进行调整
易出现误差累积：偏离目标值后无法自动修正
无法应对复杂环境：不适用高精度控制的场景

开环系统是一种简单的控制方式，适用于低成本、误差要求不高的场景

6.6 闭环控制方法

●[6.6.1] 闭环控制系统

- 闭环控制(Closed-loop Control)，通过实时检测系统的输出，将其与目标值进行比较，调整输入，使系统保持在期望状态



反馈：避免了面多了放水，水多了放面的问题

自动调节：系统可以自动补偿误差，提高控制精度
抗干扰能力强：外界环境变化(如温度变化、负载波动)不会影响系统稳定性
稳定性好：可以长期维持目标值，减少漂移
适应性强：适用于复杂环境，例如工业控制、自动驾驶、智能机器人

实现复杂：需要传感器、控制器、反馈系统等额外组件
成本较高：相比开环控制，闭环控制系统硬件和计算成本较高
可能出现震荡：如果参数调整不当，可能会导致系统振荡、不稳定

6.6 闭环控制方法

●[6.6.1] 闭环控制系统

- 闭环控制(Closed-loop Control)，通过实时检测系统的输出，将其与目标值进行比较，调整输入，使系统保持在期望状态

对比项	闭环控制（Closed-loop）	开环控制（Open-loop）
反馈机制	依赖传感器反馈调整	无反馈，固定输出
精度	高，能自动修正误差	低，容易受外界干扰
抗干扰能力	强，能适应环境变化	弱，环境变化会影响控制结果
应速度	略慢（需要计算反馈）	快（无计算直接执行）
复杂度	高，需要控制算法和传感器	低，直接输出控制信号
应用场景	电机控制、无人机、自动驾驶	电子时钟、简单灯光控制

开环系统与闭环系统各有优缺点，应根据实际场景选择系统结构

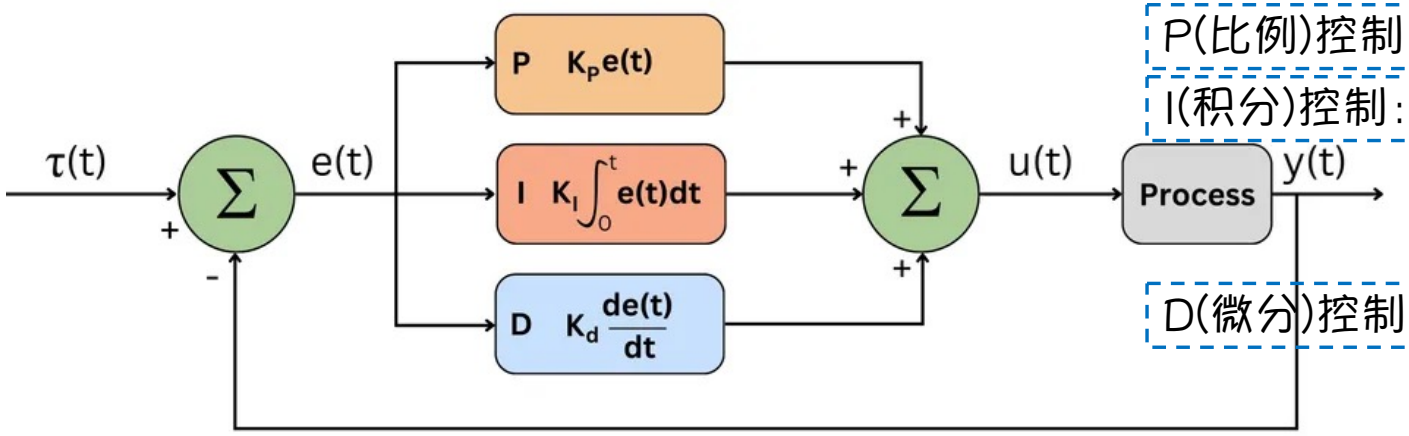
6.6 闭环控制方法

[6.6.2] 常见的闭环控制算法

- 常见的闭环控制算法
  - PID控制(Proportional-Integral-Derivative)：最常见算法，电机控制常用

$$\mu(t) = K_P e(t) + K_I \int_0^t e(t)dt + K_D \frac{de(t)}{dt}, \qquad e(t) = r(t) - y(t)$$

  - e(t)：误差；r(t)：设定值(目标值)；y(t)：实际系统输出(传感器测量值)



P(比例)控制：放大误差，提高响应速度

I(积分)控制：消除稳态误差，提高精度

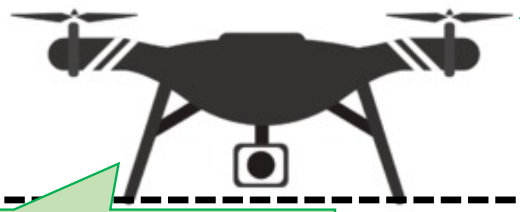
D(微分)控制：预测误差变化，防止振荡

6.6 闭环控制方法

●[6.6.2] 常见的闭环控制算法

- 常见的闭环控制算法
  - PID简单说明：无人机高度控制示例

目标高度
100m



螺旋桨提供升力，转速越快，升力越大，上升速度越快

地面

高度计，测量高度

仅有P项时： $K_P e(t)$
在地面上时，e(t)最大，螺旋桨转速大，升力大，上升速度快
接近100m过程中，e(t)逐渐变小，螺旋桨转速逐渐变小，升力逐渐减小，上升速度逐渐减小
最终，达到某个不到100米的高度，不再升高

e(t)不能为0

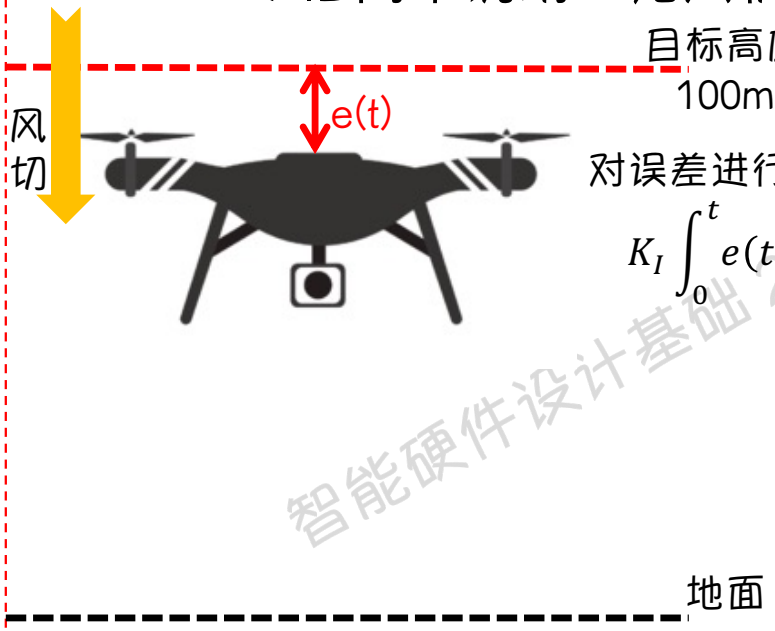
比例控制P：
优点：快速减小误差
缺点：存在稳态误差，无法被消除，一直存在

6.6 闭环控制方法

●[6.6.2] 常见的闭环控制算法

● 常见的闭环控制算法

● PID简单说明：无人机高度控制示例



新增加I项：$K_I \int_0^t e(t)dt$
e(t)增大 ->> 无人机继续上升，消除了稳态误差

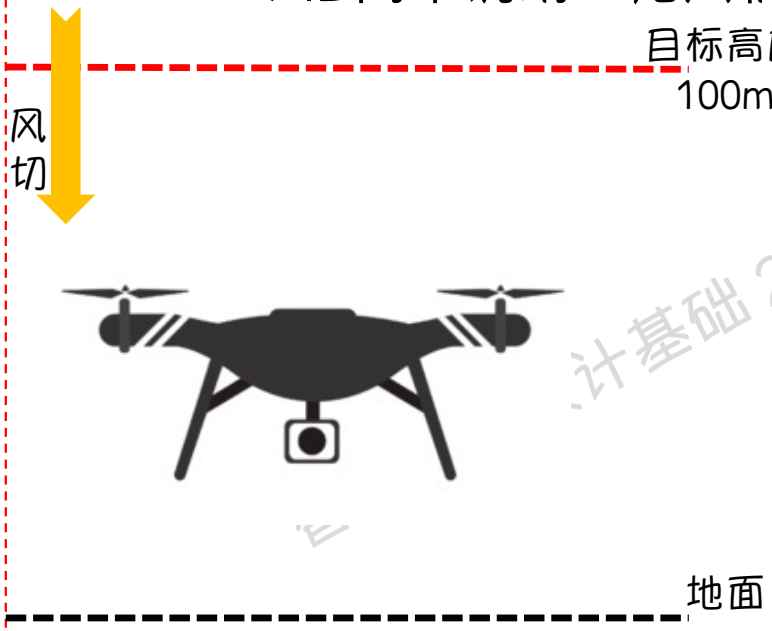
在没有外界扰动的情况下，到达目标高度，可能过冲

新的问题：
无人机工作在非理想环境中
外界突然来了一阵向下的垂直风切变
结果：
无人机在外界干扰的作用下迅速下降
需求：
无人机必须快速上升(比只有P更快)，以免坠毁

6.6 闭环控制方法

●[6.6.2] 常见的闭环控制算法

- 常见的闭环控制算法
  - PID简单说明：无人机高度控制示例



新增加D项：$K_D \frac{de(t)}{dt}$
e(t)趋势变大(导数变大)
螺旋桨转速快速增加，升力增大，阻止变化

D项的另一个作用，抑制PI的过冲问题
PI控制接近目标高度时，误差快速接近于0时
变化率为负值，控制量减小，无人机升力减小

积分控制提供惯性，微分控制提供阻尼

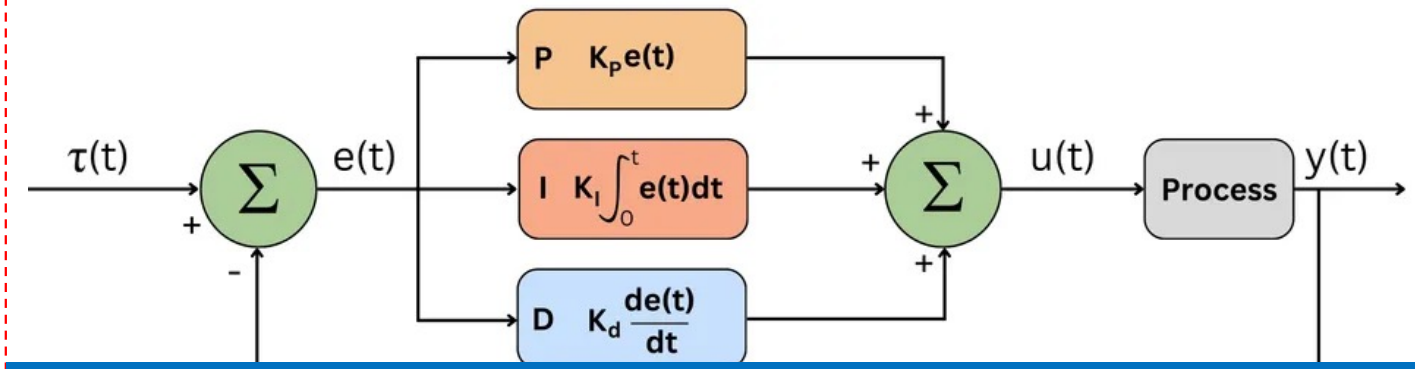
6.6 闭环控制方法

[6.6.2] 常见的闭环控制算法

- 常见的闭环控制算法
  - PID控制(Proportional-Integral-Derivative)：最常见算法，电机控制常用

$$\mu(t) = K_P e(t) + K_I \int_0^t e(t)dt + K_D \frac{de(t)}{dt}, \qquad e(t) = r(t) - y(t)$$

  - e(t)：误差；r(t)：设定值(目标值)；y(t)：实际系统输出(传感器测量值)



适用于大多数控制系统
结构简单，易于实现

需要调参，若K_P, K_I, K_d参数调整不当可能导致振荡
适用于线性系统，非线性系统可能效果较差

课程的实践环节，有能力的同学可以考虑在WS63上使用PID，更流畅的控制小车运动

6.6 闭环控制方法

[6.6.2] 常见的闭环控制算法

- 常见的闭环控制算法
  - 模糊控制(Fuzzy Control)
    - 核心思想：利用模糊逻辑推理来处理不确定性和非线性系统，不依赖精确数学模型，适用于复杂系统
    - 应用：家电(空调、洗衣机等)、智能控制系统、工业过程控制
  - 自适应控制（Adaptive Control）
    - 核心思想：实时调整控制参数，以适应环境或系统参数变化。
    - 常见类型：
      - 自适应 PID 控制：在线调整 PID 参数。
    - 应用：机器人控制、无人机、复杂工业过程。

其他还包括滑模控制、最优控制、模型预测控制、神经网络控制等，了解即可，不做要求