 辽宁大学

本科课程 课程设计

（2023-2024学年第二学期）

课程名称：控制技术课程设计

学 院：轻型产业学院

专业年级：电气工程及其自动化2021级

学生姓名：姓名（学号）

姓名（学号）

姓名（学号）

提交日期：2024年XX月XX日

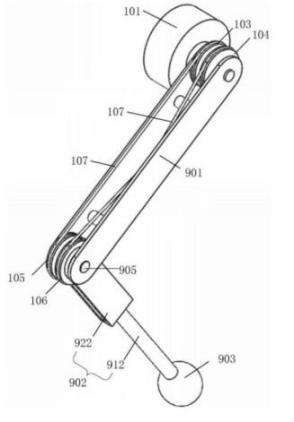
题目二：机器狗的运动控制

一、设计要求

通过对开发板编程实现机器狗的运动控制。

1. 理论分析
   1. 足部结构

我们选择平行布置形式结构，机体运动方向与腿的主运动平面一致，平行布置形式容易实现机器人的灵活运动和多运动姿态，没有偏转自由度时主要靠纵向行走。（如下左图）

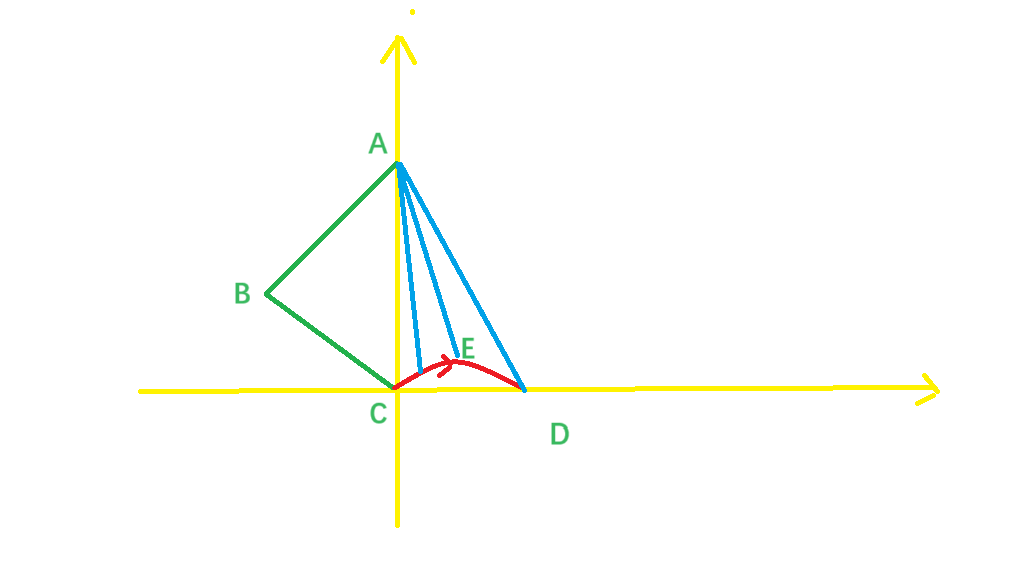
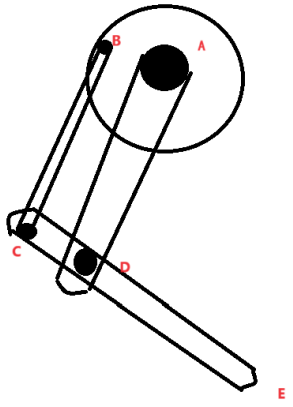


* 1. 腿型选择

我们选择腿足式腿部的串联腿结构，其属于开链式腿结构，结构简单，具有较强的姿态修复能力。运动范围更大、更清晰，运动状态较为丰富。（如上右图）

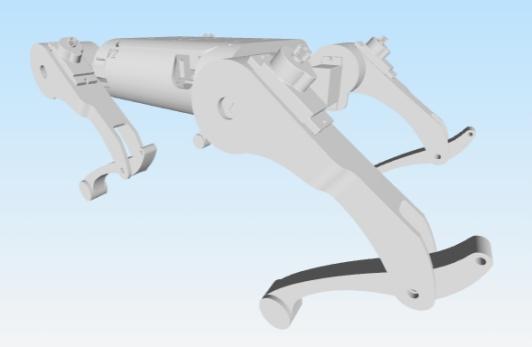
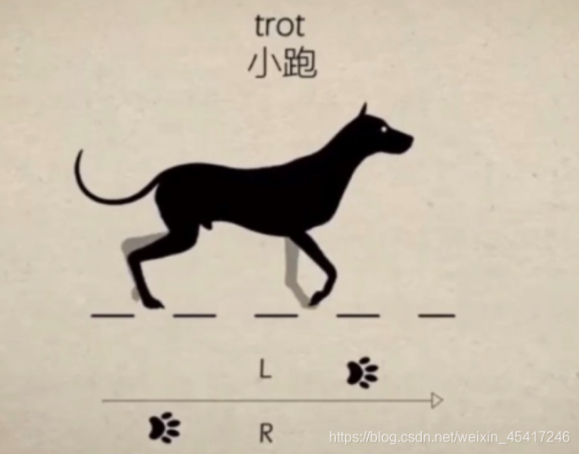
* 1. 单足运动状态

在运动中，在抬脚前半环节大臂摆动需小于小臂的角度，后半环节由于脚部要下落，小臂倒转，大臂继续向前旋转。抬腿环节结束后，为腿撑地向后摆的环节，再次环节中腿部一直着地，大腿关节与小腿关节恢复初始状态，最终的触地点运动轨迹为下图D→C轨迹，最终实现一条腿的行走全过程。（如下图）



* 1. 步态配合

按照平衡方式来分，四足机器人的步态可以分为静态步态，动态步态和准静态步态三种。我们所使用的Trot步态是一种动态步态，适用于中低速跑动，并且具有比较大的运动速度范围，另一个重要特征是在中等速度下的Trot步态具有最高的能量效率。这些优点使得Trot步态成为最常用的四足步态。Trot步态的特征是以对角的两条腿成对运动，即腿1和腿3运动一致，腿2和腿4运动一致，理想情况下对角腿同时抬起并同时着地。（如下左图）



1. 设计方案
   1. 前期准备
2. 物料准备
   1. 机器狗外壳（如上右图）
   2. 电源

14.8V-4S航模锂电池。（如下图）



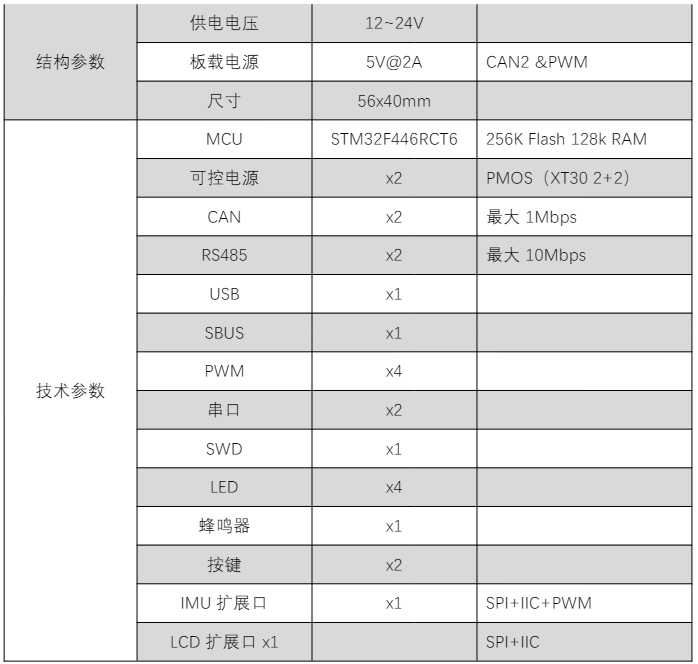
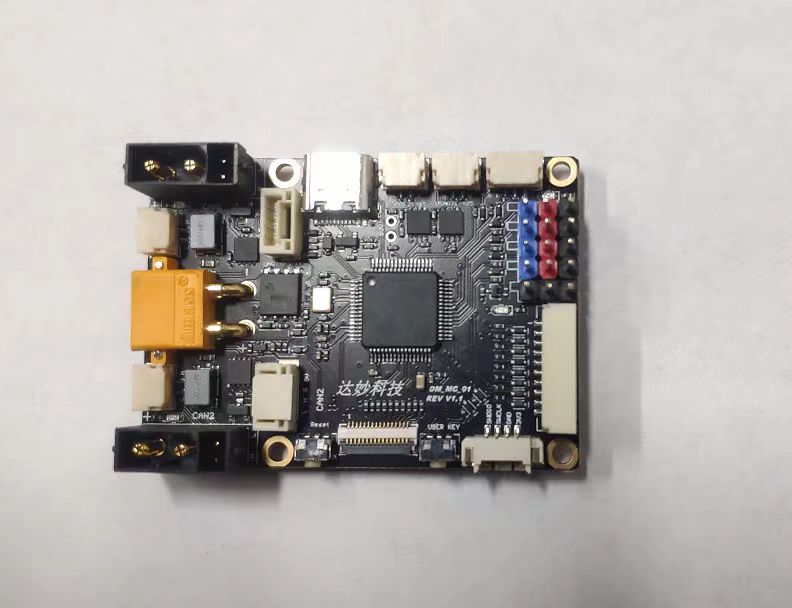
* 1. 树莓派4B及SD卡

用于进行图片拍摄、获取障碍物距离并发送到开发板以达到控制机器狗启停的效果。（如下图）



* 1. 嵌入式开发板

STM32F446开发板：用于输出pwm信号控制机器狗足部运动状态以及获取来自上位机（树莓派）的信号控制机器狗启停。（如下图）



* 1. 舵机

MG90S-180度-14克舵机。（如下图）



* 1. 含HDMI接口的屏幕（用其他接口也可以，只要有对应转接头即可）
  2. 其他杂件：包括面包板、万用表、杜邦线及各类导线、螺丝螺母螺丝刀等工具

1. 开发环境准备
   1. 开发嵌入式开发板的环境
      1. CLion版（我使用的）：参考文档[配置CLion用于STM32开发](https://zhuanlan.zhihu.com/p/145801160)
      2. Keil版（比较通用）：<Keil创建STM32项目并烧录使用.docx>
   2. 开发树莓派的环境
      1. SD卡格式化软件：SD Card Formatter
      2. 官方系统烧录软件：Raspberry Pi Imager
      3. SSH登录软件：PuTTY（会使用SSH者直接使用电脑自带终端即可），用于常规登录使用树莓派
      4. VNC登录软件：VNC Viewer，用于在自己的电脑上不使用额外屏幕的情况下，可视化地使用树莓派
      5. 初始环境配置文档：详见[树莓派初始化配置文档](https://github.com/xingwenzan/STProjectFiles/blob/master/RelevantInformation/AboutRaspberryPi/RaspberryPiInit.md)或[树莓派初始化配置（本地版）](RaspberryPiInit.pdf)
      6. Python虚拟环境：参考文档[树莓派python虚拟环境的设置与使用](https://blog.csdn.net/qq_59449692/article/details/136397131)（网络）和<python虚拟环境配置文档.docx>（本地）
   3. 电控的实现与调试
2. 上位机（树莓派）的启停控制
   1. 简述：

通过处理树莓派定时拍摄获取的图片信息，获取到机器狗到障碍物的距离，并根据该距离向下位机（开发板）发送启、停信号。

其中该部分功能分为：定时任务、图片截取、图片处理、数据发送4块内容，这4块内容前两块使用的是Linux命令完成，后两块使用python完成，且后三块的内容完成后将汇总形成一个shell脚本并赋给第一块内容完成功能的总体实现。

下面的所有代码都是在树莓派的终端或电脑ssh登录后的界面输入并运行（按回车），其中前者是图形化界面，即VNC登录或者树莓派有单独的屏幕时可以在树莓派桌面上找到的软件，但二者无本质不同，都是在“小黑窗”中使用Linux命令操作。

* 1. 定时任务（Cron）

Cron简介：

Cron是一个基于时间的作业调度器，它是类似Unix的操作系统（如Linux及其许多衍生产品）的一个组成部分。Cron可以定时、定期以及特定时刻地调用shell脚本或运行命令，即完成定时任务。

Cron的启用：

运行以下代码：

sudo crontab –e

注意：如果你从来没有使用过cron，它会提示你选择一个文本编辑器。建议选择nano，因为这个最容易快速学会。

Cron的使用：

在输入启用命令后会进入到cron的配置文件，在文件最下面根据下面的格式书写想要定时运行的命令：

m h dom mon dow command

以上6个变量中：前5个代表任务启动的时刻，最后一个代表要运行的命令。前5个变量从左到右分别代表该任务启动的分钟、小时、日期（Day Of the Month）、月份、周几（Day Of the Week）。

当我们为前五个变量赋值时，它将像闹钟一样在我们定好的时间运行对应命令。但我们想要每隔一段时间进行一次操作，那么我们可以把对应的变量赋值为“\*”或“\*/x”(x是一个数字，代表步长)，以达到定时器的效果。例：

每个月的3号2时1分执行command命令。

1 2 3 \* \* command

每5分钟执行一次command命令。

\*/5 \* \* \* \* command

参考文档：[什么是linux Cron作业，如何设置Ubuntu Crontab定时任务](https://www.labno3.com/2021/01/12/beginners-guide-to-cron-jobs-and-crontab/)

Cron的启停：

由于cron在树莓派启动后会自动运行，在调整的时候我们有时会需要暂停他的任务，这就需要这部分命令了。

检测你的service路径（这个路径是你下面几个命令中“路径”处所填写的内容）：

which service

查看cron服务状态：

sudo 路径 cron status

重新 cron 服务配置：

sudo 路径 cron reload

重新启动 cron 服务：

sudo 路径 cron restart

停止 cron 服务：

sudo 路径 cron stop

* 1. 图片截取

由于我们使用的是树莓派4B，该版本默认开启摄像头功能，将摄像头插入正确的接口再调用正确的命令即可使用，我们使用的摄像头为树莓派自带的摄像头。

常用命令如下：

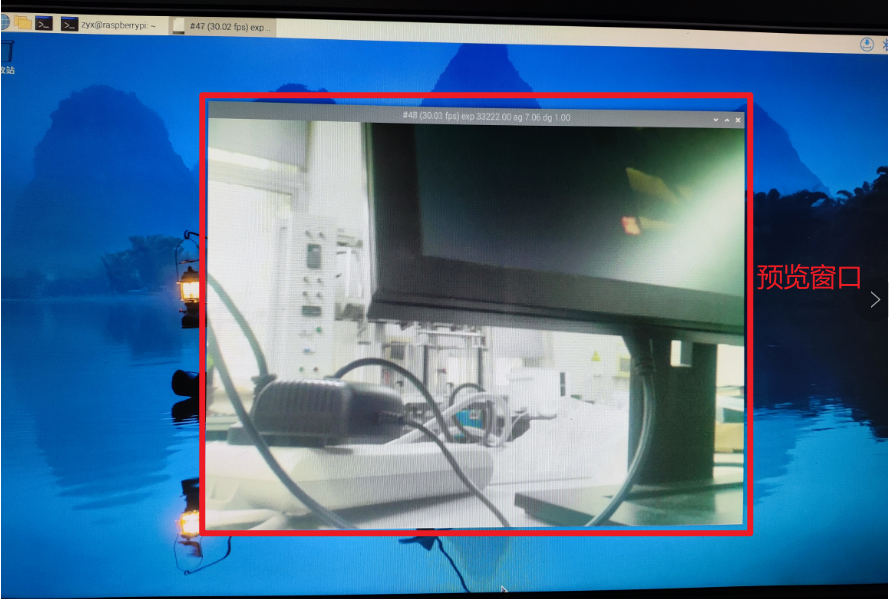
测试命令：

用于测试摄像头是否可用它会显示预览窗口约 5 秒钟，如果不好使再另找解决办法。

注意：该命令仅在图形化界面有效，ssh登录使用该命令不会有效果，因为ssh其实是调用树莓派的终端，因此仅能传输终端内容，该命令会弹出另外的窗口，故不会在自己的电脑上显示。

libcamera-hello [-t x]

-t 选项允许用户选择窗口的显示时间长度，其中x是要持续的时间，以毫秒为单位。若要无限期运行预览，x设置为0即可。（效果如下图）



拍摄命令：

用于拍摄图片并保存到指定路径下。

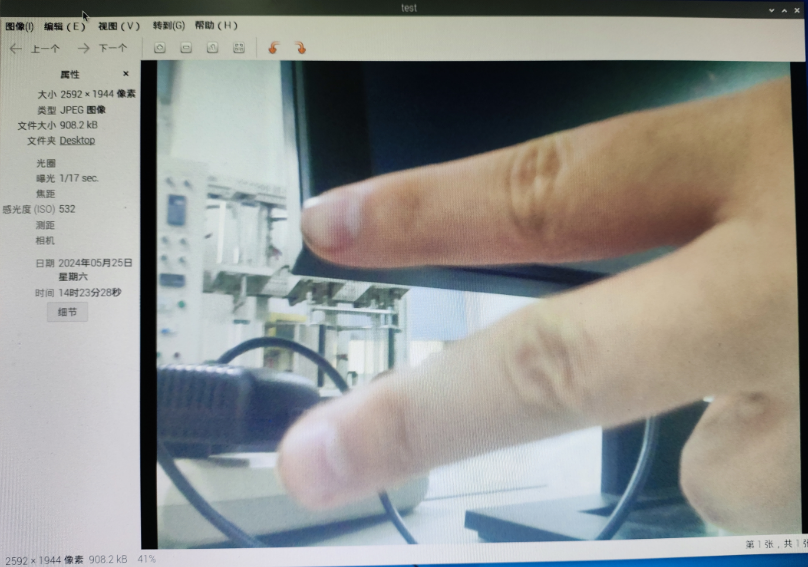
libcamera-jpeg -o test.jpg [-t x] [--width y] [--height z]

-o 选项后填写拍摄照片保存的路径及文件名，文件名前不写路径代表直接存放在本用户对应的默认文件夹中。

-t 选项 可用于更改预览显示的时间长度，默认预览5秒（-t 5000）后拍摄。

--width和 --height选项将更改捕获的静止图像的分辨率。

参考文档：[树莓派官方摄像头模块 V3 的配置及 libcamera 的使用教程](https://shumeipai.nxez.com/2023/05/14/camera-module-v3-configuration-and-libcamera-usage.html)



* 1. 图片处理

根据图片的基本属性，可得公式：，进而可知。

我们使用OpenCV的函数可以实现对特定的色块/图形实现测距。

OpenCV的下载安装：

打开python虚拟环境：

source /你的路径/venv/bin/activate

为python下载串口包：

pip install opencv-python

注意：OpenCV在python环境下安装的名字叫opencv-python，在python文件调用的名字叫cv2。

OpenCV用于读取本地图片的python代码：

cv2.imread(filepath,flags)

filepath:要读入图片的完整路径。

flags：读入图片的标志，可选标志如下：

cv2.IMREAD\_COLOR：默认参数，读入一副彩色图片，忽略alpha通道。

cv2.IMREAD\_GRAYSCALE：读入灰度图片。

cv2.IMREAD\_UNCHANGED：顾名思义，读入完整图片，包括alpha通道。

测距完整程序：

import cv2

import time

from collections import deque

import numpy as np

#设定蓝色阈值，HSV空间

blueLower = np.array([100, 100, 100])

blueUpper = np.array([120, 255, 255])

#初始化追踪点的列表

mybuffer = 64

pts = deque(maxlen=mybuffer)

#等待两秒

#time.sleep(2)

#创建一个线程类

def ceju\_xianshi():

# 从本地获取图片

frame = cv2.imread("/home/warm/timelapse/tmp.jpg",cv2.IMREAD\_COLOR)

# 转到HSV空间

hsv = cv2.cvtColor(frame, cv2.COLOR\_BGR2HSV)

# 根据阈值构建掩膜

mask = cv2.inRange(hsv, blueLower, blueUpper)

# 腐蚀操作

mask = cv2.erode(mask, None, iterations=2)

# 膨胀操作，其实先腐蚀再膨胀的效果是开运算，去除噪点

mask = cv2.dilate(mask, None, iterations=2)

# 轮廓检测

cnts = cv2.findContours(mask.copy(), cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)[-2]

# 初始化瓶盖圆形轮廓质心

center = None

# 如果存在轮廓

if len(cnts) > 0:

# 找到面积最大的轮廓

c = max(cnts, key=cv2.contourArea)

# 确定面积最大的轮廓的矩形

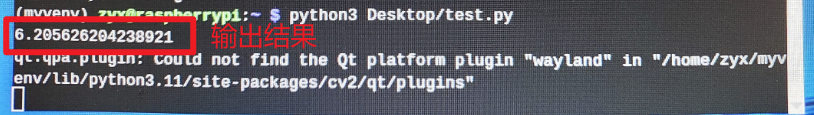
x, y, w, h = cv2.boundingRect(c)

juli = 1268/w\*2.54

print(juli) #输出距离

if \_\_name\_\_ == '\_\_main\_\_':

ceju\_xianshi()





* 1. 数据发送

为了使用的便捷与复现的容易，我们选择使用串口作为开发板与树莓派的通信工具。步骤如下：

设置并开启串口：详见<树莓派串口与外部设备通信.docx>

打开python虚拟环境：

source /你的路径/venv/bin/activate

为python下载串口包：

pip install serial

创建并打开python文件（路径、名字随意，能找到就行，名字后缀必须是“.py”）：

[sudo] nano XXX.py

注意：sudo代表该命令使用root权限，请谨慎使用

添加代码：

import serial

ser = serial.Serial("/dev/ttyAMA0", 115200)# 设置串口和波特率

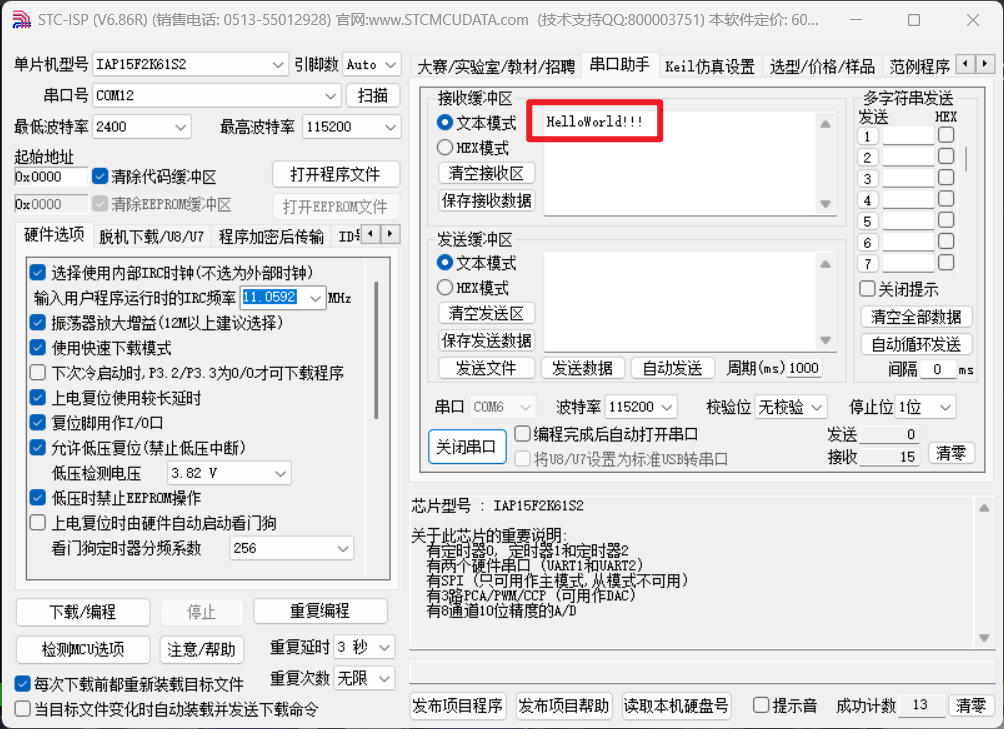
ser.flushInput() # 清除缓存

ser.write("要发送的内容\r\n".encode()) # 发送数据 \r\n可以实现换行 encode()默认是'utf-8'

保存代码后运行：

python3 XXX.py

XXX.py 填写的是刚才的python文件的路径



1. 下位机（主控，即开发板）的运动控制
   1. 简述：

开发板通过发送并改变PWM信号控制4组共8个舵机的角度实现机器狗的运动，同时通过串口接收由树莓派发送的信息来控制机器人的启停。

这部分的内容主要分为：PWM信号输出、定时器控制（用于在正确的时间做出动作）、串口接收。所有内容使用C语言编写，通过调用STM32的HAL库实现，代码简单易懂。

* 1. 定时器控制

简述：

该控制分为两种不同的方法，一种方法是通过系统定时器（SysTick）控制，另一种是通过基本定时器（Timer）控制。

二者的差别在于，前者镶嵌在内核上，且功能简单、只能递减、配置简单、优先级高，是24位的，后者是外设，功能多、配置相对复杂、优先级低、是16位的。前者中断优先级处于内核中断的末尾，但是却比所有外设的中断优先级要高，即除非系统异常，比如复位，否则前者将稳定运行。因此表现为前者稳定，不受外界干扰，后者容易受外界干扰。

因此前者一般用于操作系统时间，进程切换等，后者用于用户需求。但由于后者功能较多，能够用来控制PWM信号，且二者基本属于同一套控制流程，故我们选择后者进行使用。

基本定时器控制：

在软件层面，基本定时器控制包括初始化、中断服务配置两部分，前者需要在main文件的main函数中调用，后者需要在stm32f4xx\_it.c文件中声明发生该中断时应该做什么。

定时器的初始化包括NCIV配置和模式配置两部分。

在NCIV配置中，仅需要两行代码，前者用于设置抢占优先级，子优先级，后者用于设置中断来源：

HAL\_NVIC\_SetPriority(中断来源, 中断优先级, 中断子优先级);

HAL\_NVIC\_EnableIRQ(中断来源);

模式配置部分共分为3步，分别对应下面3块代码，先开启对应的定时器使能，然后通过使用定时器初始化结构体对定时器初始化，最后开启定时器更新中断：

\_\_TIMx\_CLK\_ENABLE();

htim\_base.Instance = TIMx;

/\* 累计 Period 个脉冲后产生一个更新或者中断\*/

htim\_base.Init.Period = 10000 - 1;

// 设定定时器频率

htim\_base.Init.Prescaler = 1600 - 1;

// 初始化定时器TIMx, x[6,7]

HAL\_TIM\_Base\_Init(&htim\_base);

HAL\_TIM\_Base\_Start\_IT(&htim\_base);

模式配置部分启动了定时器更新中断，在发生中断时，中断服务函数就得到运行。在服务函数内直接调用库函数HAL\_TIM\_IRQHandler函数，它会产生一个中断回调函数HAL\_TIM\_PeriodElapsedCallback，用来添加用户代码，控制在产生定时器中断时进行的操作，这就是中断服务配置，其代码如下：

void BASIC\_TIM\_IRQHandler(void) {

HAL\_TIM\_IRQHandler(&htim\_base);

}

// 定时器中断响应 - 代码重写

void HAL\_TIM\_PeriodElapsedCallback(TIM\_HandleTypeDef \*htim) {

if (htim == (&htim\_base)) {

Robot\_Control();//发生该中断时调用名为Robot\_Control的代码

}

}

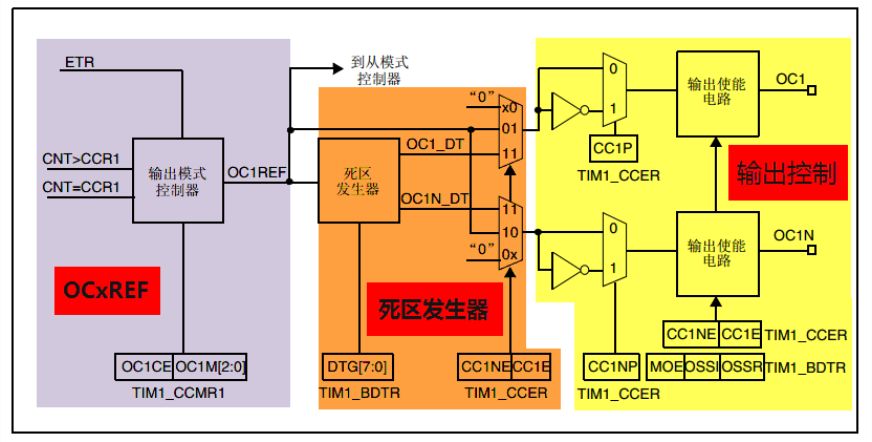
参考文档：[野火STM32F4xx系列基本定时器使用](https://doc.embedfire.com/mcu/stm32/f429tiaozhanzhe/hal/zh/latest/doc/chapter30/chapter30.html)

* 1. PWM信号输出

PWM信号输出功能通过高级控制定时器实现，其和通用定时器在基本定时器的基础上引入了外部引脚，可以输入捕获和输出比较功能。高级控制定时器比通用定时器增加了可编程死区互补输出、重复计数器、带刹车(断路)功能，这些功能都是针对工业电机控制方面。

高级控制定时器时基单元包含一个16位自动重载计数器ARR，一个16位的计数器CNT，可向上/下计数，一个16位可编程预分频器PSC，预分频器时钟源有多种可选，有内部的时钟、外部时钟。还有一个8位的重复计数器RCR，这样最高可实现40位的可编程定时。

输出比较就是通过定时器的外部引脚对外输出控制信号，有冻结、将通道X（x=1,2,3,4）设置为匹配时输出有效电平、将通道X设置为匹配时输出无效电平、翻转、强制变为无效电平、强制变为有效电平、PWM1和PWM2这八种模式，具体使用哪种模式由寄存器CCMRx的位OCxM[2:0]配置。其中PWM模式是输出比较中的特例，使用的也最多。



在这部分我们通过初始化3个结构体实现PWM输出初始化，并通过调用一个HAL库函数改变输出PWM频率，以实现舵机的角度控制。

时基结构体TIM\_Base\_InitTypeDef：

用于定时器基础参数设置，与TIM\_TimeBaseInit函数配合使用完成配置：

typedef struct {

uint16\_t Prescaler; // 预分频器

uint16\_t CounterMode; // 计数模式

uint32\_t Period; // 定时器周期

uint16\_t ClockDivision; // 时钟分频

uint8\_t RepetitionCounter; // 重复计算器

} Time\_Base\_InitTypeDef;

Prescaler：定时器预分频器设置，时钟源经该预分频器才是定时器计数时钟CK\_CNT，它设定PSC寄存器的值。计算公式为：计数器时钟频率 (f:sub:CK\_CNT) 等于 fCK\_PSC / (PSC[15:0] + 1)，可实现1至65536分频。

CounterMode：定时器计数方式，可设置为向上计数、向下计数以及中心对齐。高级控制定时器允许选择任意一种。

Period：定时器周期，实际就是设定自动重载寄存器ARR的值，ARR 为要装载到实际自动重载寄存器（即影子寄存器）的值，可设置范围为0至65535。

ClockDivision：时钟分频，设置定时器时钟CK\_INT频率与死区发生器以及数字滤波器采样时钟频率分频比。可以选择1、2、4分频。

RepetitionCounter：重复计数器，只有8位，只存在于高级定时器。

输出比较结构体TIM\_OCInitTypeDef：

用于输出比较模式，与TIM\_OCx\_SetConfig函数配合使用完成指定定时器输出通道初始化配置。高级控制定时器有四个定时器通道，使用时都必须单独设置。

typedef struct {

uint32\_t OCMode; // 比较输出模式

uint32\_t Pulse; // 脉冲宽度

uint32\_t OCPolarity; // 输出极性

uint32\_t OCNPolarity; // 互补输出极性

uint32\_t OCFastMode; // 比较输出模式快速使能

uint32\_t OCIdleState; // 空闲状态下比较输出状态

uint32\_t OCNIdleState; // 空闲状态下比较互补输出状态

} TIM\_OCInitTypeDef;

OCMode：比较输出模式选择，总共有八种，常用的为PWM1/PWM2。它设定CCMRx寄存器OCxM[2:0]位的值。

Pulse：比较输出脉冲宽度，实际设定比较寄存器CCR的值，决定脉冲宽度。可设置范围为0至65535。

OCPolarity：比较输出极性，可选OCx为高电平有效或低电平有效。它决定着定时器通道有效电平。它设定CCER寄存器的CCxP位的值。

OCNPolarity：比较互补输出极性，可选OCxN为高电平有效或低电平有效。它设定TIMx\_CCER寄存器的CCxNP位的值。

OCFastMode：比较输出模式快速使能。它设定TIMx\_CCMR寄存器的，OCxFE位的值可以快速使能或者禁能输出。

OCIdleState：空闲状态时通道输出电平设置，可选输出1或输出0，即在空闲状态(BDTR\_MOE位为0)时，经过死区时间 后定时器通道输出高电平或低电平。它设定CR2寄存器的OISx位的值。

OCNIdleState：空闲状态时互补通道输出电平设置，可选输出1或输出0，即在空闲状态(BDTR\_MOE位为0)时，经过死区时间后 定时器互补通道输出高电平或低电平，设定值必须与OCIdleState相反。它设定是CR2寄存器的OISxN位的值。

断路和死区结构体TIM\_BreakDeadTimeConfigTypeDef：

用于断路和死区参数的设置，属于高级定时器专用，用于配置断路时通道输出状态，以及死区时间。它与HAL\_TIMEx\_ConfigBreakDeadTime函数配置使用完成参数配置。

typedef struct {

uint32\_t OffStateRunMode; // 运行模式下的关闭状态选择

uint32\_t OffStateIDLEMode; // 空闲模式下的关闭状态选择

uint32\_t LockLevel; // 锁定配置

uint32\_t DeadTime; // 死区时间

uint32\_t BreakState; // 断路输入使能控制

uint32\_t BreakPolarity; // 断路输入极性

uint32\_t BreakFilter; // 断路输入滤波器

uint32\_t Break2State; // 断路2输入使能控制

uint32\_t Break2Polarity; // 断路2输入极性

uint32\_t Break2Filter; // 断路2输入滤波器

uint32\_t AutomaticOutput; // 自动输出使能

} TIM\_BreakDeadTimeConfigTypeDef;

OffStateRunMode：运行模式下的关闭状态选择，它设定BDTR寄存器OSSR位的值。

OffStateIDLEMode：空闲模式下的关闭状态选择，它设定BDTR寄存器OSSI位的值。

LockLevel：锁定级别配置， BDTR寄存器LOCK[1:0]位的值。

DeadTime：配置死区发生器，定义死区持续时间，可选设置范围为0x0至0xFF。它设定BDTR寄存器DTG[7:0]位的值。

BreakState：断路输入功能选择，可选使能或禁止。它设定BDTR寄存器BKE位的值。

BreakPolarity：断路输入通道BRK极性选择，可选高电平有效或低电平有效。它设定BDTR寄存器BKP位的值。

BreakFilter：断路输入滤波器，定义BRK 输入的采样频率和适用于 BRK的数字滤波器带宽。它设定BDTR寄存器BKF[3:0]位的值。

Break2State：断路2输入功能选择，可选使能或禁止。它设定BDTR寄存器BK2E位的值。

Break2Polarity：断路2输入通道BRK2极性选择，可选高电平有效或低电平有效。它设定BDTR寄存器BK2P位的值。

Break2Filter：断路2输入滤波器，定义BRK2 输入的采样频率和适用于 BRK2的数字滤波器带宽。它设定BDTR寄存器BK2F[3:0]位的值。

AutomaticOutput：自动输出使能，可选使能或禁止，它设定BDTR寄存器AOE位的值。

调用函数：

\_\_HAL\_TIM\_SetCompare(&对应定时器结构体,要调整的通道,调整后的占空比);

* 1. 串口接收

串口通讯(Serial Communication)是一种设备间非常常用的串行通讯方式，因为它简单便捷，因此大部分电子设备都支持该通讯方式，电子工程师在调试设备时也经常使用该通讯方式输出调试信息。

对于本次任务仅需要串口接收到正确的内容即可实现控制，对于串口的实现，我们仅需要两步，第一步是进行初始化配置，第二步则是应用。前者需要重写HAL\_UART\_MspInit和HAL\_UART\_MspDeInit两个函数完成对TX引脚和RX引脚的复用，并使用USART初始化结构体，以实现串口初始化的目的。后者由于STM32的HAL库里提前封装了关于串口的发送、接收代码，我们仅需要在main函数里面调用它即可。

关于USART初始化结构体：

typedef struct {

uint32\_t BaudRate; // 波特率

uint32\_t WordLength; // 字长

uint32\_t StopBits; // 停止位

uint32\_t Parity; // 校验位

uint32\_t Mode; // USART模式

uint32\_t HwFlowCtl; // 硬件流设置

uint32\_t OverSampling; // 过采样设置，8倍或者16倍

} USART\_InitTypeDef;

BaudRate：波特率设置。一般设置为2400、9600、19200、115200。HAL库函数会根据设定值计算得到UARTDIV值，并设置UART\_BRR寄存器值。

WordLength：数据帧字长，可选8位或9位。它设定UART\_CR1寄存器的M位的值。如果没有使能奇偶校验控制， 一般使用8数据位；如果使能了奇偶校验则一般设置为9数据位。

StopBits：停止位设置，可选0.5个、1个、1.5个和2个停止位， 它设定USART\_CR2寄存器的STOP[1:0]位的值，一般我们选择1个停止位。

Parity：奇偶校验控制选择，可选UART\_Parity\_No(无校验)、 UART\_Parity\_Even(偶校验)以及UART\_Parity\_Odd(奇校验)， 它设定UART\_CR1寄存器的PCE位和PS位的值。

Mode：UART模式选择，有UART\_Mode\_Rx和UART\_Mode\_Tx，允许使用逻辑或运算选择两个， 它设定USART\_CR1寄存器的RE位和TE位。

HwFlowCtl ：为是否支持硬件流控制，我们设置为无硬件流控制。

OverSampling：过采样为 16 倍还是 8 倍。

关于串口接收调用函数：

int ch;

HAL\_UART\_Receive(&huart, (uint8\_t \*) &ch, 1, 0xFFFF);

参考文档：[野火STM32F4xx系列串口通信](https://doc.embedfire.com/mcu/stm32/f429tiaozhanzhe/hal/zh/latest/doc/chapter20/chapter20.html)

1. 设计结果及分析

成功实现机器狗运动控制，让机器狗真正的走了起来，并且通过对树莓派的使用，实现了机器狗的启停控制，达到并扩展了最初的目标。





1. 设计的亮点

将树莓派与开发板结合，实现通信控制，完成功能扩展，实现机器狗的启停。

1. 经验体会

张馨文：

1. 具体贡献：负责对开发板和树莓派的环境二次配置和控制的实现工作。
2. 经验体会：完整的进行了从环境配置到运动与启停控制的流程，对嵌入式硬件开发有了很深刻的体会。

李星宇：

1. 具体贡献：负责实验基本的设备应用与环境配置调试工作。
2. 经验体会：了解并熟悉了各种设备的使用，对机器人控制所需环境有了深刻了解。

于海洋：

1. 具体贡献：负责实验过程的调试、分析与记录工作。
2. 经验体会：完整了解了实验的流程，并深刻理解了对各个模块的问题分析

刘航宇：

1. 具体贡献：负责实验的理论分析、基本物料准备、设备装配以及报告的撰写、排版与审阅工作。
2. 经验体会：了解了对机器狗的运动方式的原理，对此产生深刻理解。