DIY 结题报告

12212215 姚奇霖 12211908 张承璧

2024 年 6 月 8 日

1 小车底盘

1.1 组成

机械结构：小车底盘由两个 JGB37-520 直流电机，和一个舵机 DS3225 组成。小车的后轮与直流电机连接。前轮通过连杆和舵机相连。舵机转动，前轮跟着转向。

电路组成：小车底盘的电路部分由 stm32f103c8t6 作为主控。它控制了一个 oled 屏幕。它输出一路 pwm 波给 tb6612 电机驱动板, 并读取电机编码器的信号，从而实现了电机速度闭环控制。12V 电源通过两个降压模块转为 3.3V 和 5V, 给单片机，屏幕，电机驱动板供电。

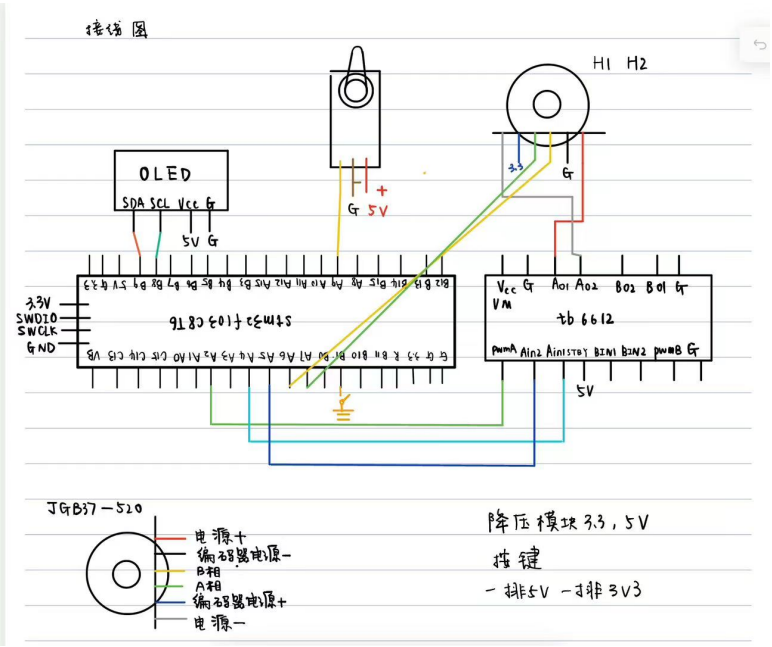
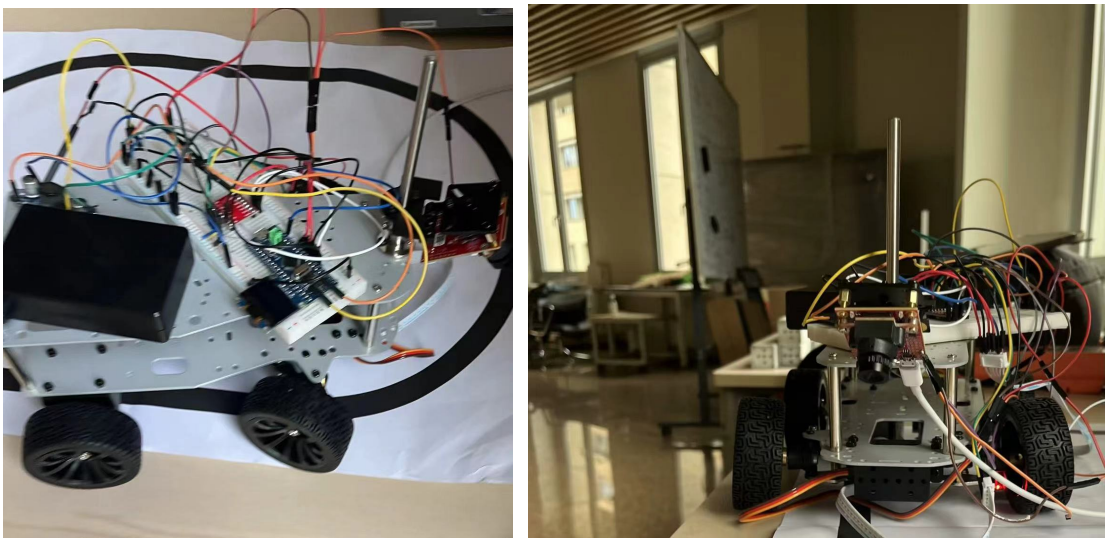


图 1: 接线图

1.2 原型验证

该小车实现了电机速度闭环控制，屏幕显示和舵机转向。我们先用面包板验证测试电路。在保证其功能可行的基础上，去绘制 PCB。



(a) (b)

图 2: 原型验证

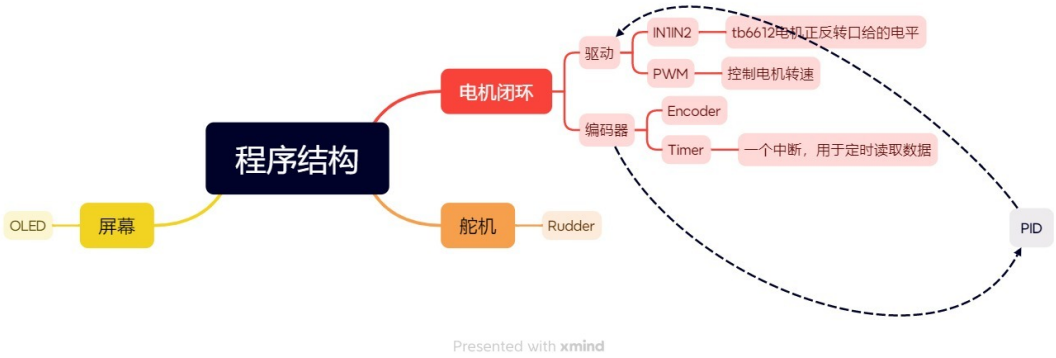


图 3: 程序结构

1.2.1 电机速度闭环控制

(a)stm32 输出 pwm 波控制转速

stm32 输出 pwm 波控制电机转速。然后电机编码器返回信号。

pwm 波决定了电机的转速。在高电平期间电机转动，在低电平期间，电机依靠惯性转动。

pwm 的产生：定时器接受由总线传来的时钟，经过预分配之后到达计数器。计数器向上计数。当计数器的值 (记为 cnt) 到达 arr 重装值。计数器归零。

在这个过程中，比较器或比较 ccr 和 cnt, 如果 cnt<ccr，通过 TIMx_CHx 通道对外输出高电平。如果 cnt>=ccr, 输出低电平。调整 ccr 和 arr 的比值就可以调节占空比。PSC（预分频数）和 aar 共同决定了 pwm 波的周期。

因此 pwm 波的各个参数可以得出：

PWM 频率：$Freq = CKPSC/(PSC+1)/(ARR+1)$

PWM 占空比：$Duty = CCR/(ARR+1)$

PWM 分辨率：$Reso = 1/(ARR+1)$

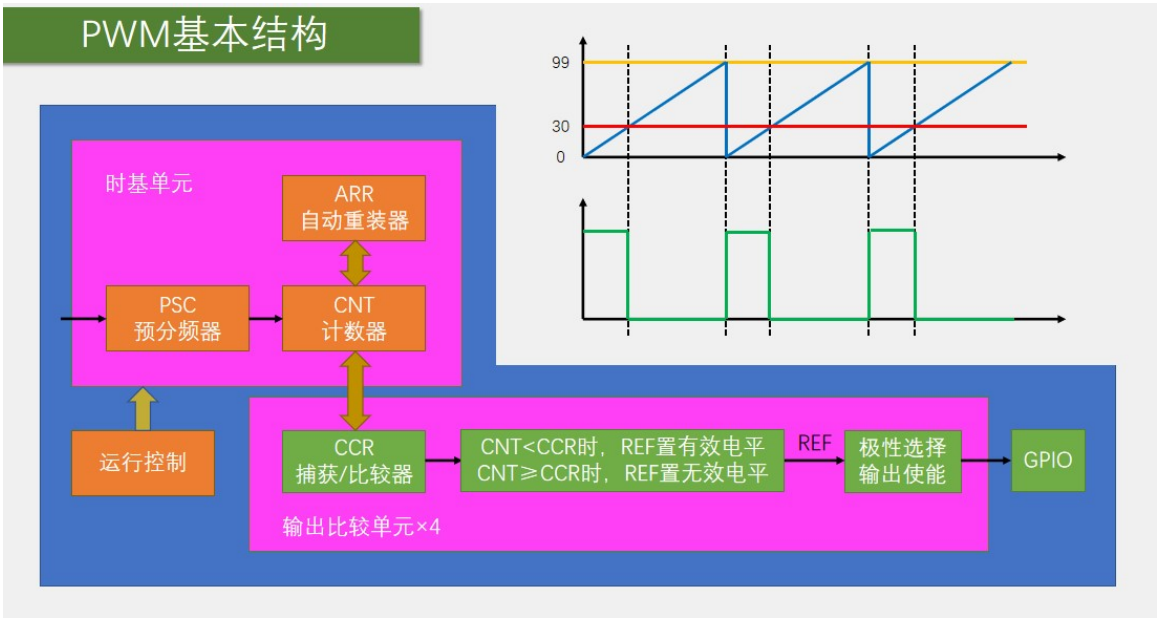


图 4: pwm 波

(b) 编码器测速

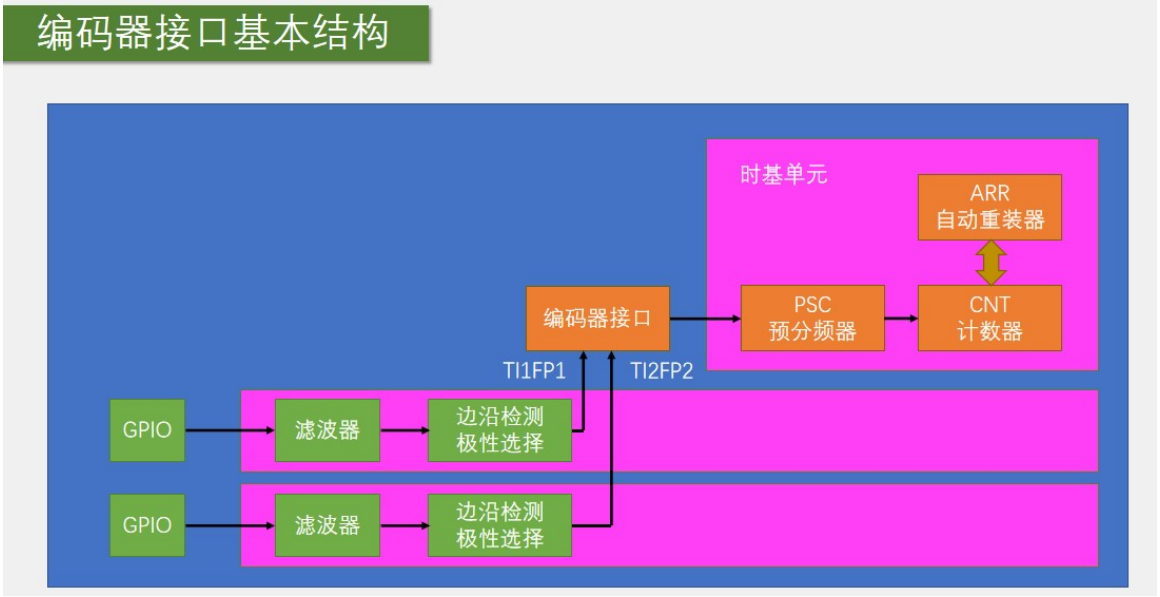


图 5: 编码器测速 1

该电机使用了霍尔编码器，它输出了两路有相位差方波。

编码器测速原理：霍尔编码器输出两路有一定相差的方波给 stm32。当 A 相（TI1）在上升沿，B 相（TI2）为低电平时，或者 B 相在上升沿，A 相为高电平，或者 A 相在下降沿，B 相为高电平, 或者 B 相在下降沿，A 相为低电平，计数器计次。

因此计数器寄存器存储的值是小车轮子位置信息我们使用了额外的一个定时器（TIM4）。使用中断定时地去读取中断编码器所对应的定时器（TIM3）里的值，并每一次读取都将 TIM3 中 cnt 的值清零。这样每次读取的数值就对应了编码器的速度信息。

有效边沿	相对信号的电平 (TI1FP1对应TI2, TI2FP2对应TI1)	TI1FP1信号		TI2FP2信号	
		上升	下降	上升	下降
在TI1和TI2上计数	高	向下计数	向上计数	向上计数	向下计数
	低	向上计数	向下计数	向下计数	向上计数

图132　编码器模式下的计数器操作实例

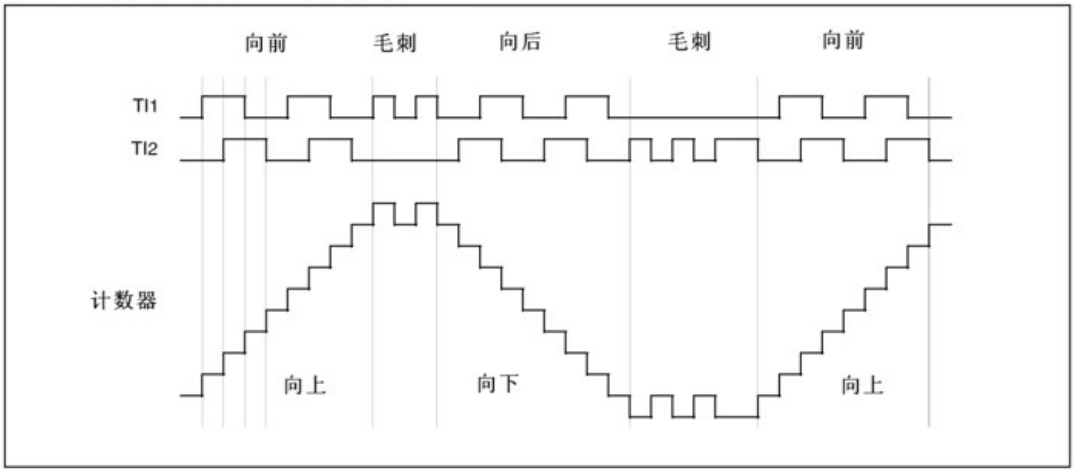


图 6: 编码器测速 2

(c)PID 调速凭借读取编码器的速度信息，我们可以有目的地去调整小车车速，使得它在遇到阻力时也能维持稳定的速度。

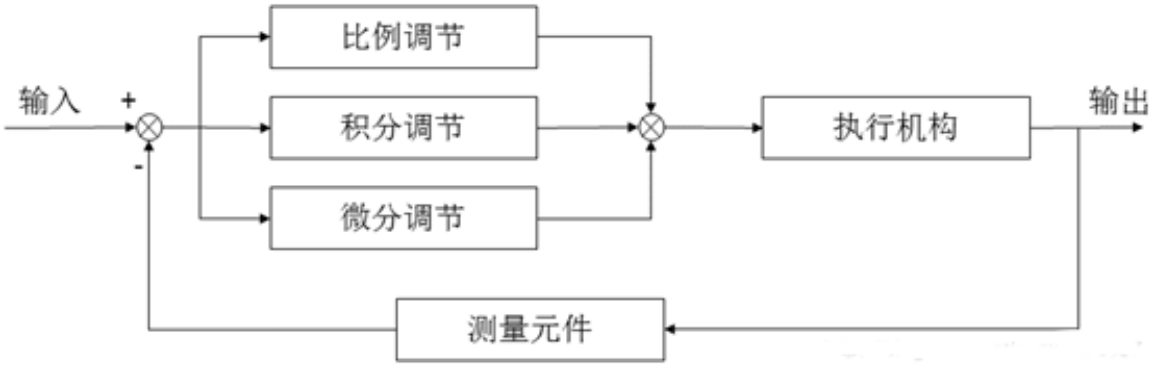


图 7: PID

1.3 PCB 设计

通过原型验证部分，我们得到了小车的接线图。按照接线图，我们绘制了 PCB。

我们的 PCB 用排针将电子元件相互连接，便于为其他模块供电。将 stm32 和 tb6612 多余的引脚用排母引出。我们专门为 GND,3V3,5V 留出了排针。在电压和 pwm 输出口增加了开关。电机的接线口位于板子的左下方。我们一共放置了两个开关，一个位于板子的左下角，用于打开或关闭电机。另外一个位于板子的右下角，用于打开或者关闭电源。

信号线宽度为 10mil, 电源线宽度为 48mil。

我们一共绘制了两个版本的 PCB。

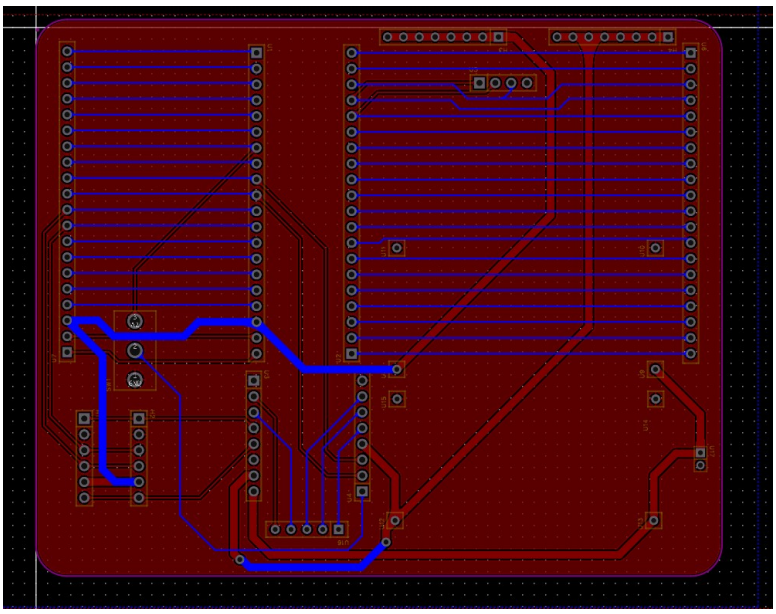


图 8: PCB 图 1

(1) 由于第一个版本的 tb6612 对应的排针物理尺寸画宽了，所以第二个版本进行了更改。

(2) 第二个版本还调整了屏幕方向，使得屏幕位于板子的边框内。

(3) 增加了一排 GND。

(4) 增加了一个开关用于控制电源通断

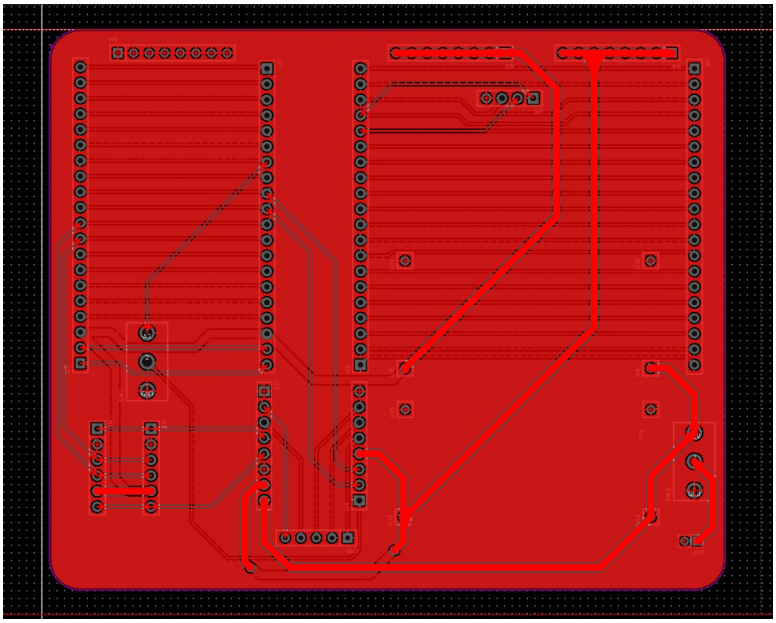


图 9: PCB 图 2

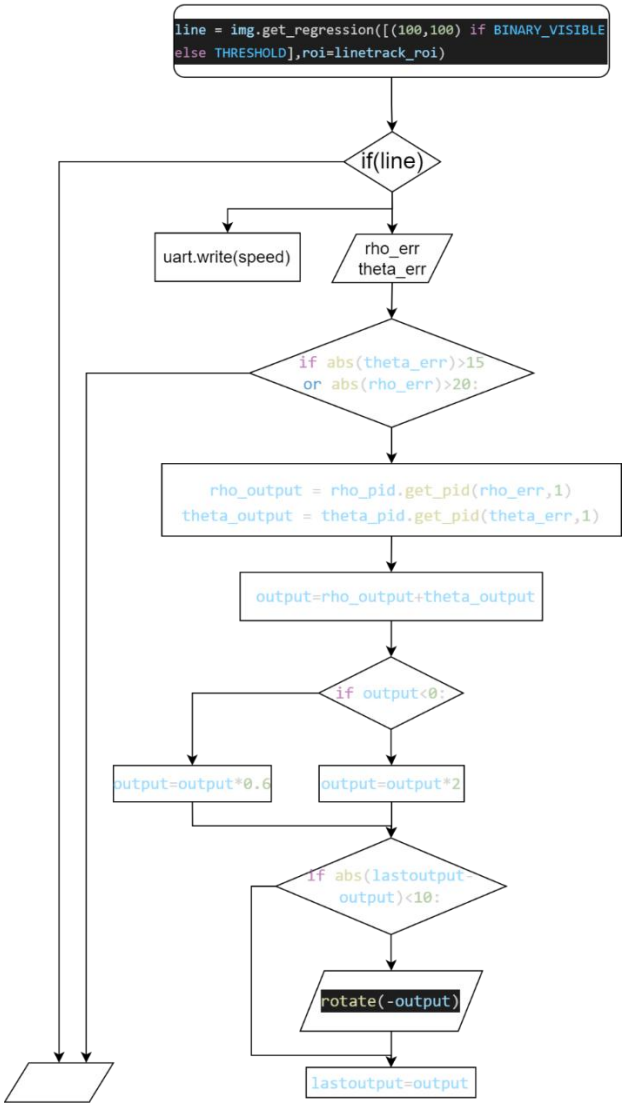
OpenMV 巡线

一、介绍：OpenMV 是一款可以编程的摄像头模组，可以实现简单的图像识别或更进阶的视觉处理功能。作为一个开源的机器视觉项目，其可以在 MCU 上运行 MicroPython 代码，并内置了一些图像处理算法，很容易使用。在本项目中，我们选择了星瞳代理发售的 OpenMV4H7plus 摄像头模块来完成基本的巡线，数字识别，黑白色块识别，十字路口检测以及小车转向舵机的控制，该摄像头模组与小车底盘主控 F103 核心板之间用蓝牙串口模块进行通信。

二、功能实现：

1. 基本巡线功能：

通过 OpenMV 内置的 **image.get_regression**(thresholds, invert=False, roi, x_stride=2, y_stride=1, area_threshold=10, pixels_threshold=10, robust=False)实现，该函数的基本原理是对图像所有阈值像素进行线性回归计算，这一计算通过最小二乘法进行，通常速度较快。若 robust 为 True，则使用 Theil-sen 线性回归算法，它计算图像中所有阈值像素的斜率中位数，鲁棒性更好，但是算法时间复杂度高所以不予采用。Thresholds 是元组列表，定义了想要进行计算的颜色阈值。Invert 是翻转阈值操作。Roi 是感兴趣区域的矩形元组（x, y, w, h），操作范围仅限于 roi 区域内的像素。x_stride 和 y_stride 是调用函数跳过的 x, y 像素数，area_threshold 边界框区域，pixels_threshold 像素数阈值。代码流程图：



Theta_err 和 rho_err 分别为拟合直线的角度偏差值和中点偏差值，0 值分别代表垂直于窗口以及横坐标均值位于窗口中点。首先对两个偏差值绝对值判断，若小于阈值则不进行计算减小巡线抖动，然后使用 pid 算法得到两个偏差值对舵机转向角度的控制输出并相加，注意在对舵机控制前将正负输出分别乘以系数抵消小车转向装置左右不对称问题，最后判断前后输出是否小于阈值进防止舵机大幅度摆动。

舵机控制介绍：

通过控制 pwm 波的占空比改变脉冲宽度来调整舵机角度，脉冲宽度范围：0.5ms-2.5ms，舵机转向范围：0-180°，实测出舵机在小车上的最大转向范围并加以控制，具体代码如下：

```
#舵机控制初始化
tim = Timer(4, freq=100) # Frequency in Hz
ch1 = tim.channel(1, Timer.PWM, pin=Pin("P7"), pulse_width_percent=17)
#12-25 右-左 17 中间
pwp = 17
def rotate(angle): #(-90~90)
    if angle>90:
        angle = 90
    elif angle<-27:
        angle = -27
    pwp = 15 + 20*(angle)/180
    ch1.pulse_width_percent(round(pwp))
def qtleft():
    rotate(90)
def qtright():
    rotate(-27)
def straight():
    rotate(18)
```

rotate 控制角度，输入为角度，正负值区分转向左右，0 表示中间角度。

功能展示：双击图片播放

2.数字识别

由于识别对象较少，识别场景单一，openmv 版上资源有限，故选择 openmv 内置的模板匹配算法识别数字。

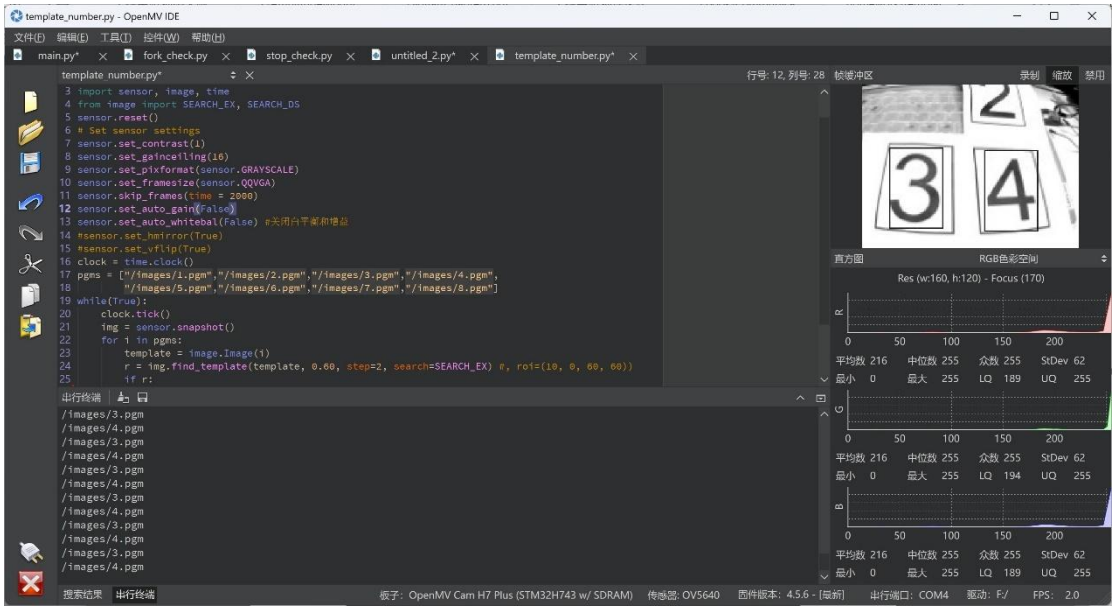
Image.find_template(template, threshold[,roi[,step=2[,search=image.SEARCH_EX]]])此函数尝试使用归一化互相关(NCC)算法在图像中找到第一个模板匹配的位置。返回匹配位置的边界框元组(x, y, w, h), 否则返回 None。Trmplate 是与对象匹配的小模板图像一般格式为

pgm，threshold 是浮点数较小值能提高检测速率但是误报率较高，roi 是感兴趣区域的矩形窗口，step 是查找模板时需要跳过的像素数量可以提高运算速度，search 选择模板所用算法 SEARCH_DS 较快，SEARCH_EX 更详尽。
首先需要用 openmv 拍摄合适的小图像模板，此步骤直接决定了识别的准确度。
将图像窗口调整至刚好能够包括数字，直接在 OpenMV 内图像窗口截取并另存为 bmp，在在线网站转为 pgm：

多模板匹配代码：

```
import sensor, image, time
from image import SEARCH_EX, SEARCH_DS
sensor.reset()
# Set sensor settings
sensor.set_contrast(1)
sensor.set_gainceiling(16)
sensor.set_pixformat(sensor.GRAYSCALE)
sensor.set_framesize(sensor.QQVGA)
sensor.skip_frames(time = 2000)
sensor.set_auto_gain(False)
sensor.set_auto_whitebal(False) #关闭白平衡和增益
#sensor.set_hmirror(True)
#sensor.set_vflip(True)
clock = time.clock()
pgms = ["/images/1.pgm","/images/2.pgm","/images/3.pgm","/images/4.pgm",
        "/images/5.pgm","/images/6.pgm","/images/7.pgm","/images/8.pgm"]
while(True):
    clock.tick()
    img = sensor.snapshot()
    for i in pgms:
        template = image.Image(i)
        r = img.find_template(template, 0.60, step=2, search=SEARCH_EX)
#, roi=(10, 0, 60, 60))
        if r:
            img.draw_rectangle(r)
            print(i)
```

识别效果如图：

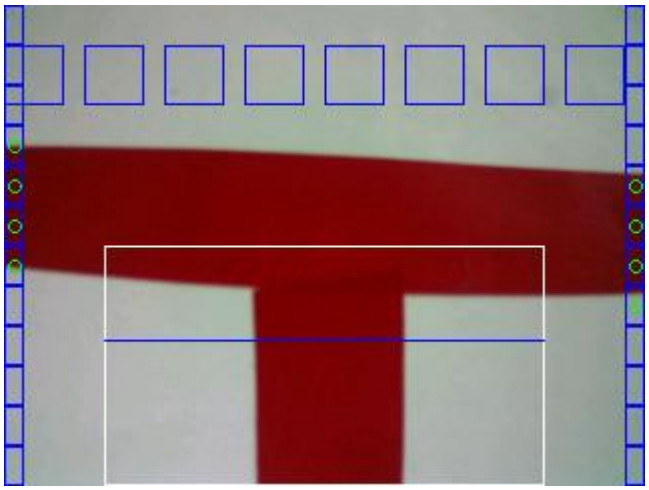


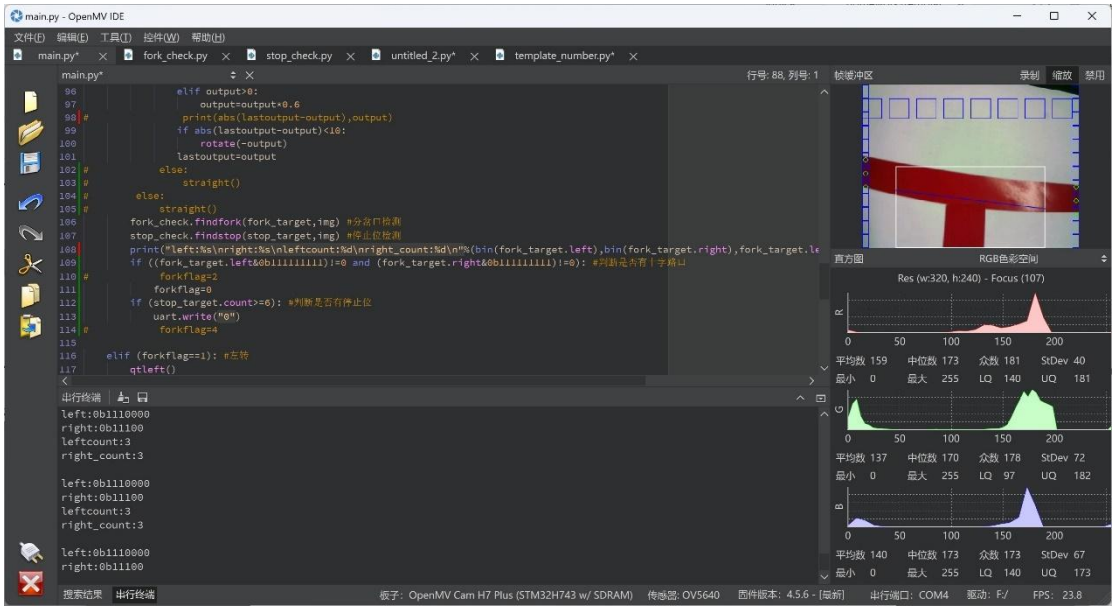
3.十字路口检测：

在这一任务中使用 OpenMV 内置的 **image.find_blobs**(thresholds, roi=Auto, x_stride=2, y_stride=1, invert=False, area_threshold=10, pixels_threshold=10, merge=False, margin=0, thresholds 是颜色的阈值，可以包含多个颜色。roi 是“感兴趣区”。在使用统计信息中已经介绍过了。x_stride 就是查找的色块的 x 方向上最小宽度的像素，默认为 2。y_stride 就是查找的色块的 y 方向上最小宽度的像素，默认为 1。invert 反转阈值，把阈值以外的颜色作为阈值进行查找 area_threshold 面积阈值，如果色块被框起来的面积小于这个值，会被过滤掉 pixels_threshold 像素个数阈值，如果色块像素数量小于这个值，会被过滤掉 merge 合并，如果设置为 True，那么合并所有重叠的 blob 为一个。margin 边界，如果设置为 1，那么两个 blobs 如果间距 1 一个像素点，也会被合并。

如上所示，在图像窗口两侧划分区域寻找色块，首先在 main.py 中调用 fork_check.py，初始化 fork_target=fork_check.check()，然后在主函数中调用 findfork()就可以得到检测结果，储存在 fork_target 中。

具体演示：





4.黑色色块识别：原理与十字路口检测原理相同，都使用了 openmv 内置的识别色块算法，在图像窗口上方规定一系列感兴趣区域并对识别到的色块计数，当计数大于某一阈值时小车停止。

5.板间通信:

两板间的通信思路：OpenMV 作为舵机驱动板和视觉模块起到了控制小车转向(pid)以及识别分析环境信息的作用，F103 作为双电机的主控板起到了速度 pid 控制以及屏幕按键等外设驱动的作用。所以 OpenMV 需要从 F103 得到任务信息即切换任务的指令，包括返回起始点，远近端控制，启停指令等，而 F103 需要从 OpenMV 得到电机速度的控制指令，包括速度大小，正反转等，以协同舵机完成任务。

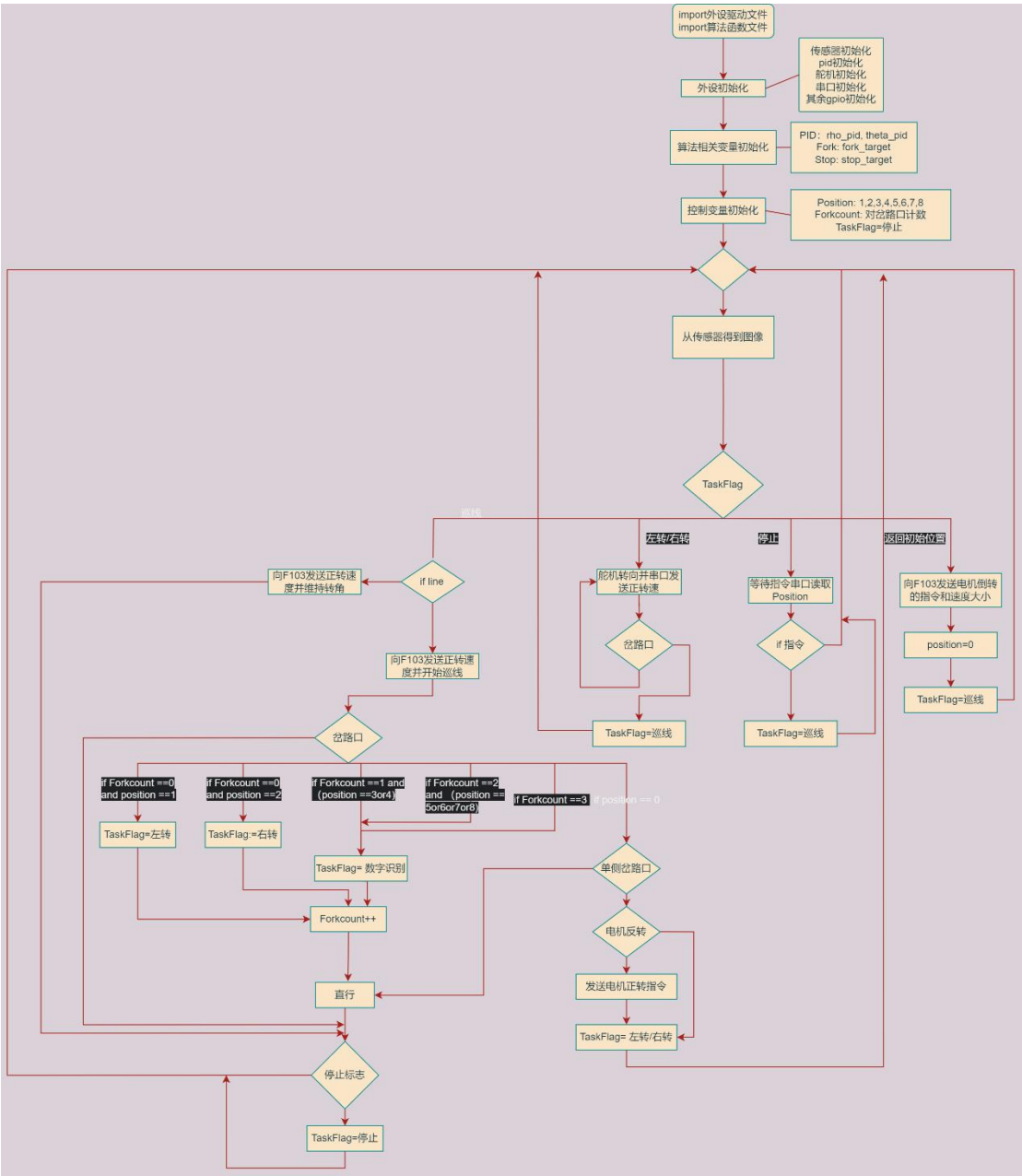
使用蓝牙串口模块连接 OpenMV 与 F103 进行通信，蓝牙模块在完成配对后使用方法与串口一致。

Openmv 串口初始化代码:

```
#串口初始化 p4-pb7 p5-pb6
uart = UART(3, 9600)  #初始化串口 3，波特率为 9600（注意：上位机记得也配置成 9600）
```

由于 F103usart1 的默认引脚被占用，故对其做 gpio 功能重映射

系统方案：



测试方案

1.基本巡线功能测试

（1）绕红色巡线圈固定速度巡线。

（2）不断增加速度调整 pid，使巡线速度尽可能快，巡线尽可能稳定。速度>1.5m/s

2.指定岔路口转弯与停止标志识别测试

（1）用红色电工胶带在喷绘布上制作三个连续的十字路口并在路口尽头制作好停止标志。

（2）串口发送 1-8 任务指令，小车接收并在指定路口左转/右转，在路口尽头停止。

3.数字识别测试

（1）在中远岔路口增加数字标识。

（2）在 2 的基础上串口发送 1-8 指令，小车数字识别自行判断转向。

4.返回初始位置测试
（1）在路口尽头给小车发送返程指令。
（2）小车正确返回初始位置并停止。
5.电机闭环控制：给电机施加阻力
电机能够在一定阻力下维持恒定速度。
当阻力施加时，电机速度能快速收敛到设定值。

测试结果及分析

测试项目：巡线
结果：pid 控制正常，舵机控制正常。
测试项目：数字识别
结果：能够在同一画面中正确识别多个数字。

附录代码：

巡线代码如下：

```
    if (forkflag==0): #巡线
        line = img.get_regression([(100,100) if BINARY_VISIBLE else
THRESHOLD],roi=linetrack_roi)
        if (line):
            speed="40"
            uart.write(speed) #后轮速度
            rho_err = (0.2*(line.x1()-img.width()/2)+0.8*(line.x2()-
img.width()/2))/2 #拟合直线中心横坐标点
            if line.theta()>90:
                theta_err = line.theta()-180
            else:
                theta_err = line.theta()
            img.draw_line(line.line(), color = 127)
            print("rho_err:%f theta_err:%f"%(rho_err,theta_err))
            print(rho_err)
            if abs(theta_err)>15 or abs(rho_err)>20: #设置 pid 阈值防止抖
动
                rho_output = rho_pid.get_pid(rho_err,1)
                theta_output = theta_pid.get_pid(theta_err,1)
                print("rho_output:%f
theta_output:%f"%(rho_output,theta_output))
                output = rho_output+theta_output
                if output<0:
                    output=output*2
                elif output>0:
                    output=output*0.6
                print(abs(lastoutput-output),output)
                if abs(lastoutput-output)<10:
                    rotate(-output)
                lastoutput=output
#            else:
#                straight()
#        else:
#            straight()
```

Pid 算法代码如下：

```
from pyb import millis
from math import pi, isnan

class PID:
```

```
    _kp = _ki = _kd = _integrator = _imax = 0
    _last_error = _last_derivative = _last_t = 0
    _RC = 1/(2 * pi * 20)
    def __init__(self, p=0, i=0, d=0, imax=0):
        self._kp = float(p)
        self._ki = float(i)
        self._kd = float(d)
        self._imax = abs(imax)
        self._last_derivative = float('nan')

    def get_pid(self, error, scaler):
        tnow = millis()
        dt = tnow - self._last_t
        output = 0
        if self._last_t == 0 or dt > 1000:
            dt = 0
            self.reset_I()
        self._last_t = tnow
        delta_time = float(dt) / float(1000)
        output += error * self._kp
        if abs(self._kd) > 0 and dt > 0:
            if isnan(self._last_derivative):
                derivative = 0
                self._last_derivative = 0
            else:
                derivative = (error - self._last_error) / delta_time
            derivative = self._last_derivative + \
                                      ((delta_time / (self._RC +
delta_time)) * \
                                         (derivative -
self._last_derivative))
            self._last_error = error
            self._last_derivative = derivative
            output += self._kd * derivative
        output *= scaler
        if abs(self._ki) > 0 and dt > 0:
            self._integrator += (error * self._ki) * scaler *
delta_time
            if self._integrator < -self._imax: self._integrator = -
self._imax
            elif self._integrator > self._imax: self._integrator =
self._imax
            output += self._integrator
        return output
```

```
    def reset_I(self):
        self._integrator = 0
        self._last_derivative = float('nan')
```

十字路口检测代码如下：

```
black = (0, 11, -21, 6, -23, 25) #颜色闸值
#red = (48, 76, 21, 66, 10, 49)
red = (17, 45, 20, 49, 3, 38)
#左侧岔路识别区域
left_roi=[(0,0,10,20),(0,20,10,20),(0,40,10,20),(0,60,10,20),(0,80,10,2
0),(0,100,10,20),
          (0,120,10,20),(0,140,10,20),(0,160,10,20),(0,180,10,20),(0,2
00,10,20),(0,220,10,20)]
#右侧岔路识别区域
right_roi=[(310,0,10,20),(310,20,10,20),(310,40,10,20),(310,60,10,20),(
310,80,10,20),(310,100,10,20),
          (310,120,10,20),(310,140,10,20),(310,160,10,20),(310,180,10,
20),(310,200,10,20),(310,220,10,20)]


class check(object):
    left = 0x00000000
    right = 0x00000000
    left_count = 0
    right_count = 0
target=check() # 储存检测结果

def draw_rois(img):
    #绘制中间左右 24 个检测区域
    for rec in left_roi:
        img.draw_rectangle(rec,color=(0,0,255))
    for rec in right_roi:
        img.draw_rectangle(rec,color=(0,0,255))

def findfork(target,img):
    target.left=0
    target.right=0
    target.left_count = 0
    target.right_count = 0
    #检测色块
    for i in range(0,12): #left
        left_blobs =
img.find_blobs([red],roi=left_roi[i],merge=True,mergin=10)
        #如果识别到了红色，hor_bits 对应位置 1
```

```
        for b in left_blobs:
            target.left=target.left|(0x01<<(11-i))
            target.left_count+=1
            img.draw_circle(int(left_roi[i][0]+left_roi[i][2]*0.5),int(
left_roi[i][1]+left_roi[i][3]*0.5),3,(0,255,0))
    for i in range(0,12):
        right_blobs =
img.find_blobs([red],roi=right_roi[i],merge=True,mergin=10)
        #如果识别到了红色，hor_bits 对应位置 1
        for b in right_blobs: #right
            target.right=target.right|(0x01<<(11-i))
            target.right_count+=1
            img.draw_circle(int(right_roi[i][0]+right_roi[i][2]*0.5),in
t(right_roi[i][1]+right_roi[i][3]*0.5),3,(0,255,0))

if __name__ == '__main__':
    while True:
        clock.tick()  # Update the FPS clock.
        img = sensor.snapshot()  # Take a picture and return the image.
        draw_rois(img)
        findfork(target,img)
        print("left:%s\nright:%s\nleftcount:%d\nright_count:%d\n"%(bin(
target.left),bin(target.right),target.left_count,target.right_count))
```