# MCU 技术及课程设计

# 实　验　报　告

小组成员：陈鲲龙，支喆为，钱思畅，刘翔宇

日期：2024.9.14

实验名称　图片识别与打靶系统设计

实验类型　设计型

## 一、　实验背景

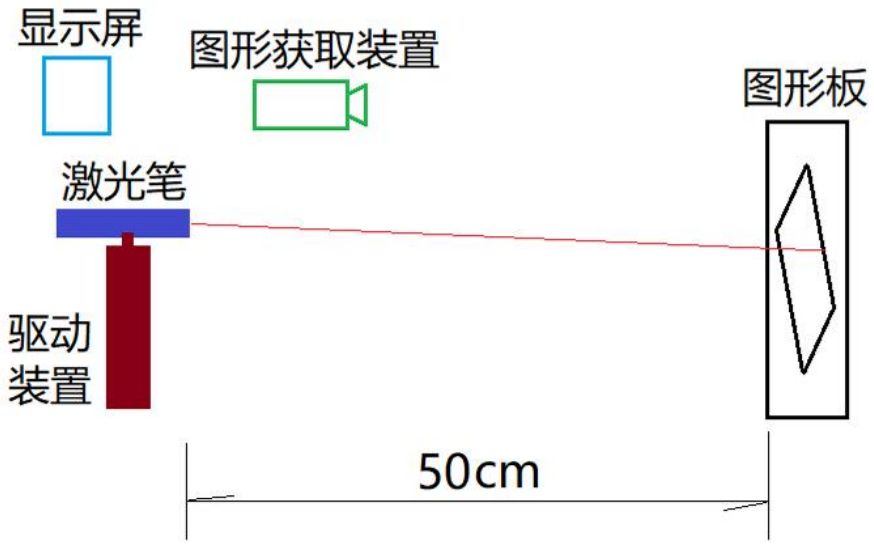
激光打点技术是一种广泛应用于工业、科研和医疗等领域的精密测量工具。它通过利用激光束来进行高精度的点位标定和测量。以下是激光打点技术设计背景的详细描述：1. 精密测量的需求在现代工业制造和科学研究中，精确的空间定位和尺寸测量至关重要。例如，在制造业中，产品的尺寸和位置精度直接影响到最终产品的质量和性能。在建筑领域，精确的点位标定对于施工和装配也极为重要。2. 激光打点的优势激光打点技术通过发射高精度的激光束来确定目标点的位置，其主要优势包括：高精度：激光点的位置可以达到亚毫米级的精度。长距离测量：激光束可以在较长距离上保持较高的测量精度。非接触测量：激光打点技术不需要接触被测物体，这对于测量软质或易损物体尤为重要。实时反馈：激光打点设备可以提供实时的测量数据，便于及时调整和校正。3. 应用领域的广泛性激光打点技术在多个领域都有重要应用：建筑和施工：用于测量和标定建筑物的结构、设备安装位置等。制造业：在生产线中用于检测和校正机械零部件的位置。科研：用于精密实验中的数据收集和空间定位。医疗：在一些医疗设备中，用于精确定位和导引。

## 二、实验仪器与元器件

（1）软件：STM32CubeIDE、Keil5
（2）舵机*2
（3）摄像头
（3）激光笔
（4）STM32F407ZGT6 开发板

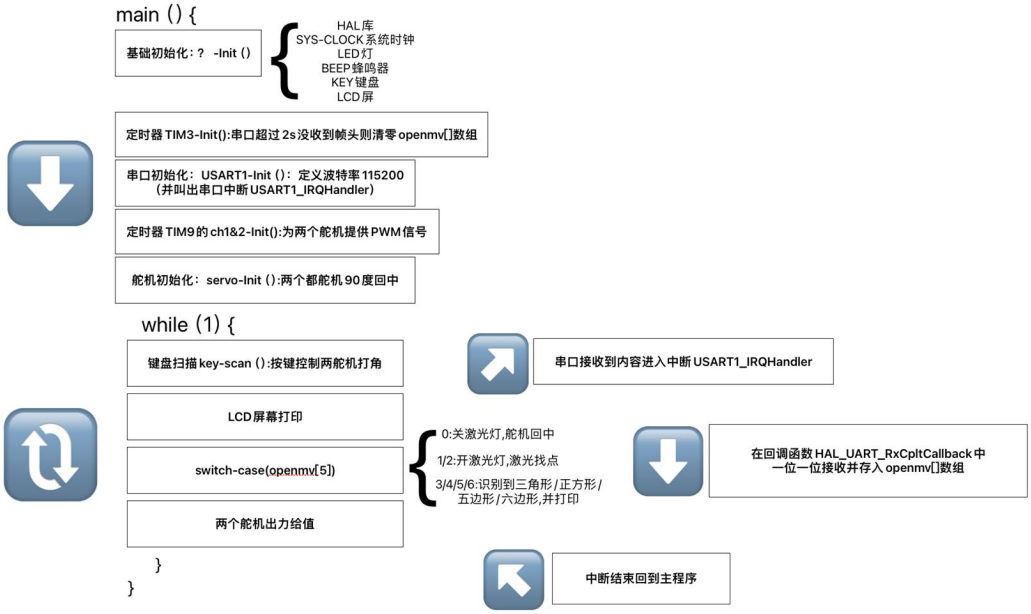
## 三、设计过程及步骤

### 1、绘制模拟图：



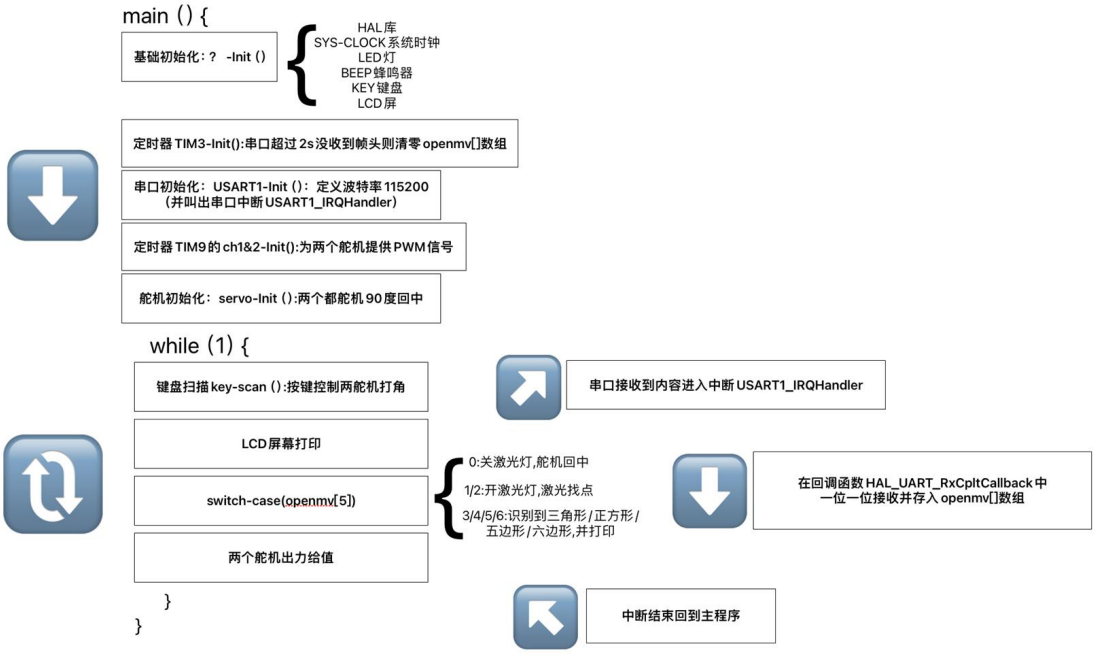

2、绘制软件设计流程：

单片机：

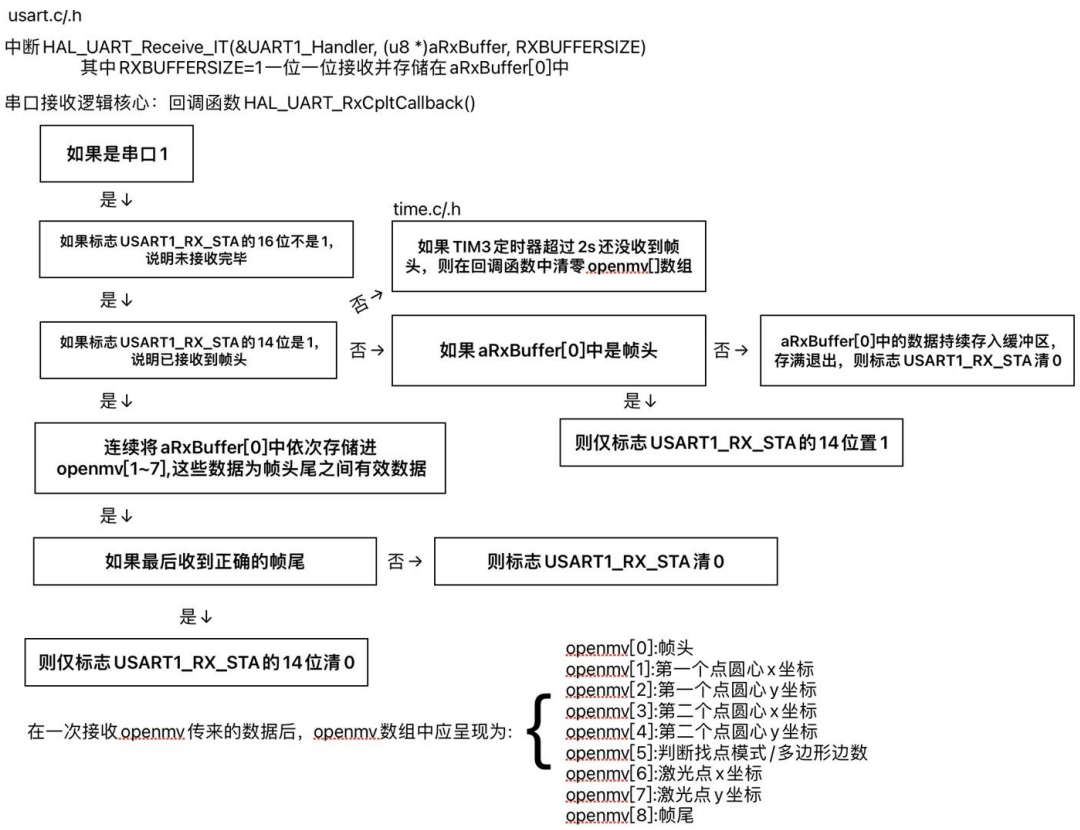



主函数上电或复位时先进行库函数、时钟和一些基础 IO 元器件的初始化；然后定时器 TIM3 初始化，用于检测串口是否停止接收数据，并清空数据存储区域；

摄像头：

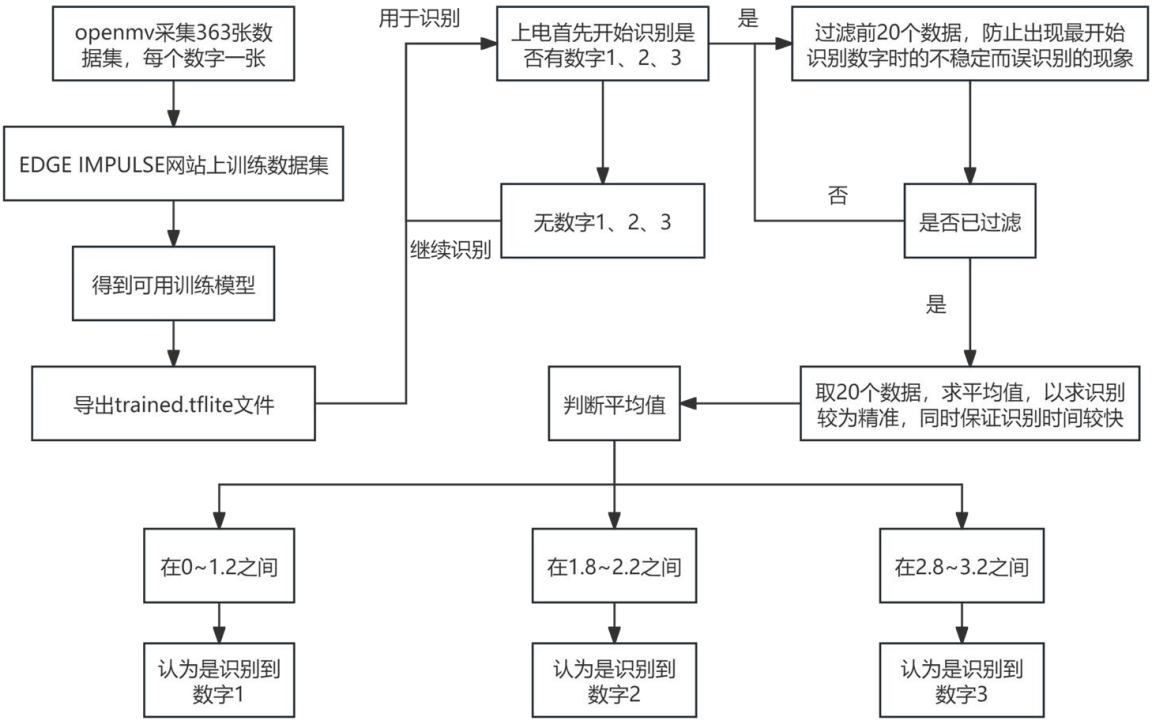



## 3、编写串口接收控制代码：

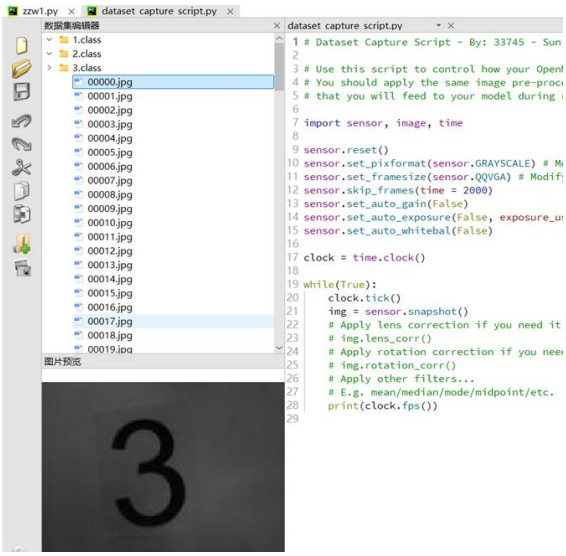


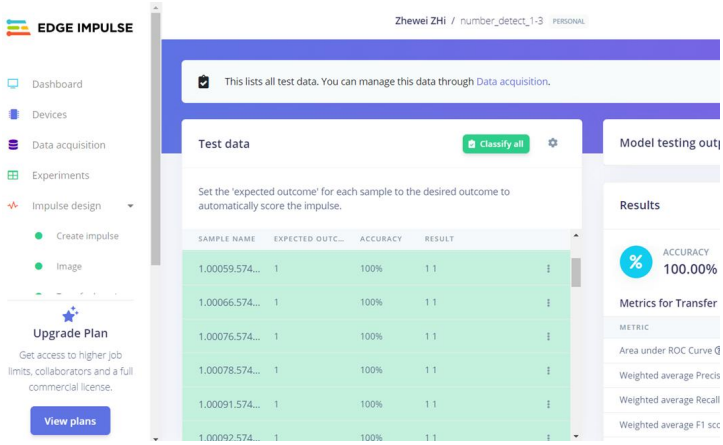

## 4、编写图像识别及激光打靶代码：

### （1）数字识别：




利用 OpenMV 采集数据集，然后在 EDGE IMPULSE 网站上训练数据集并得到用于识别的文件 trained.tflite。

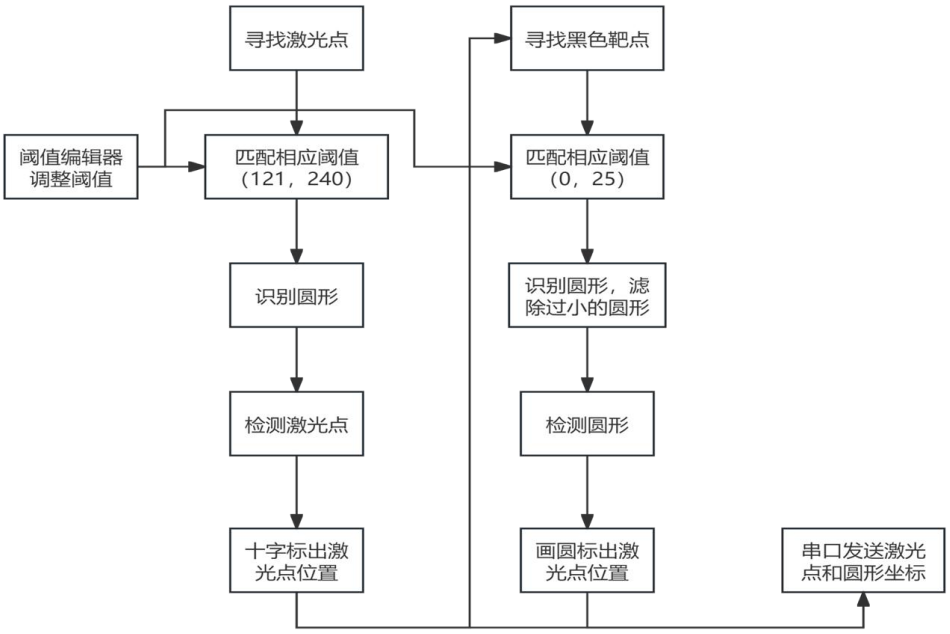




过滤初始图像不稳定时的图像，然后多次识别取平均值，作为判定进行找点功能还是识别功能的依据。

```
    for obj in tf.classify("trained.tflite", img_copy, min_scale=1.0, scale_mul=0.5, x_overlap=0.0, y_overlap=0.0):
        output = obj.output()
        number = output.index(max(output))
        # 在图像中框出识别区域
        img.draw_rectangle(obj.rect(), color = (255,255,255))
        # 在框的旁边显示识别到的数字
        x, y, w, h = obj.rect()
        img.draw_string(x, y - 10, str(number), color=(255, 255, 255))  # 在框的上方显示数字
        lcd.write(img)
        print(number)

    if count_number != 40:
        count_number += 1
    if count_number>=20 and count_number < 40:
        number_sum += number
    if count_number == 40 and function == 0:
        number_average = number_sum / 20
        print('识别停止', number_average)

        if number_average < 1.2 and number_average != 0:
            number_c = 1
        elif number_average < 2.2 and number_average > 1.8:
            number_c =  2
        elif number_average < 3.2 and number_average > 2.8:
            number_c =  3
        else:
            number_sum = 0
            number_average = 0
            count_number = 0
```

（2）找点打靶



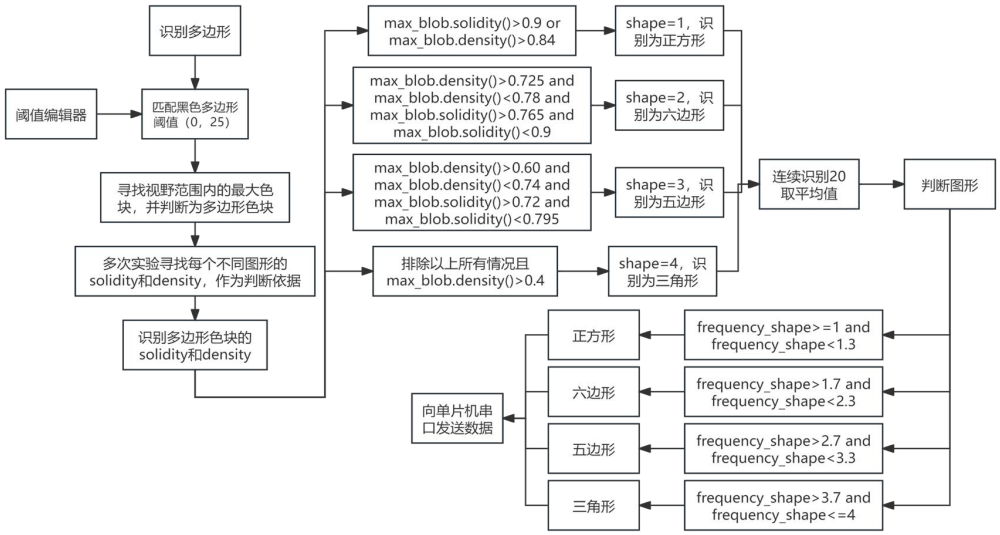

要寻找激光点，以及小圆靶点，并将识别到的两个图形的信息串口发送到单片机。

```
# 寻找激光点
def find_laser(threshold):
    blobs = img.find_blobs([threshold], x_stride=1, y_stride=1)
    if blobs:
        b = blobs[0]
        cx = b.cx()
        cy = b.cy()
        if cx is not None and cy is not None:
            # 绘制十字来表示激光笔的位置
            img.draw_cross(cx, cy)
            print("找到了激光点")
        return cx, cy
    print("未找到激光点")
    return None, None
```

```
# 识别圆形靶点方法2
def detect_circles(img):
    detected_circles=[]

    blobs=img.find_blobs([threshold_black], x_stride=2, y_stride=2,merge=False)
    if blobs:
        for blob in blobs:
            cr = int((blob.w()+blob.h())/4)
            img.draw_circle(blob.cx(),blob.cy(),cr)
            detected_circles.append([blob.cx(),blob.cy()])
            print('圆形坐标')
            print(blob.cx(),',',blob.cy())
            coordinate_str = f"坐标：{blob.cx()},{blob.cy()}"
            img.draw_string(blob.cx()-20, blob.cy()+5, coordinate_str)
```

```
# 串口发送识别出的圆形信息
def send_circle(circles,laser_x,laser_y):
    data = bytearray([0x0d])  # 起始标志

#    print(len(circles))

    if laser_x!=None and laser_y!=None:
        for c in circles:
            data.extend(struct.pack(">BB", c[0], c[1]))
#             data.extend(struct.pack("B",c[0]))
#             data.extend(struct.pack("B",c[1]))
#             data.append(c[0])
#             data.append(c[1])
    if len(circles)==1:# 如果只识别到一个圆形
        data.extend(struct.pack(">BB", 0xFF, 0xFF))
#         data.append(-1)
#         data.append(-1)
        data.append(1)
#         data.append(laser_x)
#         data.append(laser_y)
        data.extend(struct.pack(">BB", laser_x, laser_y))
    elif len(circles)==2:# 如果识别到两个圆形
        data.append(2)
#         data.append(laser_x)
#         data.append(laser_y)
        data.extend(struct.pack(">BB", laser_x, laser_y))
```

（3）识别多边形




识别白板上的图形，如果是根据边数来向单片机串口发送数字（对应着几边形）。多次识别取平均值，保证识别正确。

```
# 识别多边形
def detect_polygon(max_blob):#输入的是寻找到色块中的最大色块
    shape=-1 #保存形状
#    print(max_blob.solidity())

    if max_blob.solidity()>0.9 or max_blob.density()>0.84:
        img.draw_rectangle(max_blob.rect())
        shape=1#表示矩形

    elif max_blob.density()>0.725 and max_blob.density()<0.78 and max_blob.solidity()>0.765 a
        img.draw_rectangle(max_blob.rect())
        shape=2#表示六边形

    elif max_blob.density()>0.60 and max_blob.density()<0.74 and max_blob.solidity()>0.72 and
        img.draw_rectangle(max_blob.rect())
        shape=3#表示五边形

    elif max_blob.density()>0.4:
        img.draw_rectangle(max_blob.rect())
        shape=4#表示三角形

#    elif max_blob.density()>0.6:
#        img.draw_circle((max_blob.cx(), max_blob.cy(),int((max_blob.w()+max_blob.h())/4)))
#        shape=5#表示圆形
if count==20:
    n=0
    frequency_shape=frequency_shape/count
    print(frequency_shape)
    if frequency_shape>=1 and frequency_shape<1.3:
        print('是正四边形')
         shape=1
        n=4
    elif frequency_shape>1.7 and frequency_shape<2.3:
        print('是正六边形')
         shape=2
        n=6
    elif frequency_shape>2.7 and frequency_shape<3.3:
        print('是正五边形')
         shape=3
        n=5
    elif frequency_shape>3.7 and frequency_shape<=4:
        print('是正三边形')
         shape=4
        n=3
    if n!=0:
        send_polygon(n)
         break
    frequency_shape=0
    count=0
```

5、编写舵机工作 PWM 信号代码：

servopwm.c/.h

晶振 8M Hz 倍频→主频 168MHz

使用 APB1（被分频为 84MHz）下的 TIM9 的 ch1&2

选取 per=1000-1=999

选取 psc=1680-1=1679 } 在 main.c 中初始化时给定参数

使得 PWM 频率为 APB1/ (per+1)(psc+1)=50Hz
即舵机所需的工作在 20ms，此时 0.5ms~ 2.5ms 对应 0°～ 180°

TIM_SetCompare（）接口函数：compare 参数 50 ～ 250 对应 0.5ms~ 2.5ms 即对应 0°～ 180°，赋给 TIM9 的 CCR1/2 对应两个舵机，具体接线引脚定义为 PE5/6

6、编写舵机打角代码：

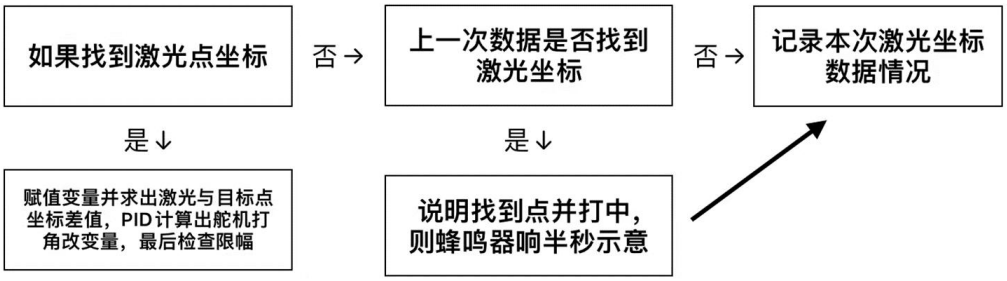
servo.c/.h

宏定义舵机的最大/最小/中值角度值

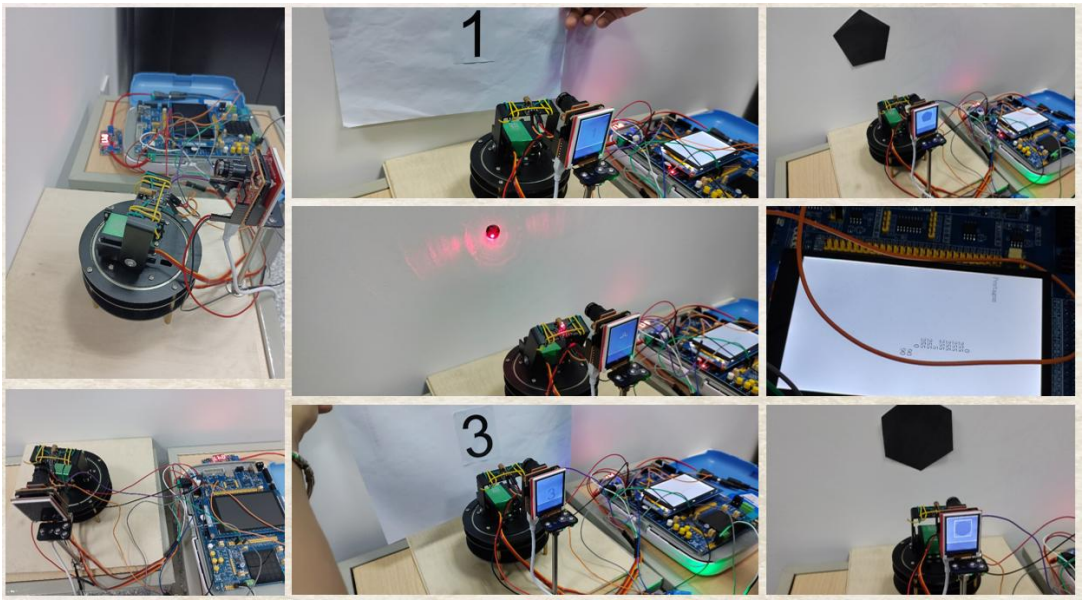
SetServoAngle 接口函数完成 0°～ 180°角度值与 50 ～ 250PWM compare 值之间的换算，并调用 TIM_SetCompare 接口函数

anglelimit 函数是舵机的角度限幅保护

激光找点核心函数：lasertrack（）




## 四、 实验结果展示



## 五、 创新点分析及实验小结

1、 创新点：

（1）将 OpenMV 与 STM32 结合使用的创新点在于构建一个智能视觉控制系统。OpenMV 负责实时图像处理，如目标检测和识别，而 STM32 则接收处理结果并执行相应的控制操作，如驱动电机或调整设备。这种结合能够将高效的图像分析与灵活的控制系统整合，提高自动化设备的智能化水平，实现快速响应和精确控制。

（2）OpenMV 识别多边形放弃了官网上传统的利用 canny 算子和霍夫变换进行直线识别。然后判断直线边数的方法。借鉴 CSDN 上“万无一失的 OpenMV 识别矩形、圆形、三角形方法”，选择根据 max_blob.solidity()和 max_blob.density()来判断形状，使得判断形状稳定性提高。

2、实验小结：

本次单片机实验通过搭建基础电路、编写控制程序、调试系统，成功实现了课设的功能。实验过程中，我加深了对单片机工作原理的理解，掌握了硬件连接与编程技巧，同时提升了问题解决能力。遇到的一些电路和编程问题，通过仔细调试和修正得以解决，进一步提高了实践操作的能力。此次实验不仅巩固了理论知识，还增强了动手实践的经验，为未来的学习打下了坚实的基础。