综合设计与实践 A 综合实验（四）设计报告

摘要：

本文介绍了“综合设计与实践 A 综合实验（四）”的基于 STM32G0 及红外反射传感器的手势识别器的设计。 从手势识别的设计要求入手，主要从硬件电路和软件编程两个方面阐述和设计实现的过程，最终呈现实现的效果。

Abstract:

This report introduces "Comprehensive Design and Practice A Comprehensive Experiment (IV)" focusing on the design of a gesture recognizer based on STM32G0 and infrared reflection sensors. Starting with the design requirements for gesture recognition, it elaborates on the implementation process from two main aspects: hardware circuitry and software programming, ultimately presenting the achieved results.

一. 题意分析

本实验题目是用 STM32G0 单片机及 16 个 ITR20001/TR24 红外对射传感器，制作可以识别“石头”、“剪刀”、“布”手势的手势识别器。

红外对射传感器的反射强度通过 ADC 采集接收端光电流获取，当红外对射传感器被遮挡时，获取的光电流会发生变化，依次来判断手遮挡的位置。

已知类别的大量手势识别的数据在 PC 上进行训练，之后再单片机上实现推理，判断结果后通过 LED 的点亮来展示。

二. 原理分析与方案选择

原理：

1.ITR20001/TR24 红外对射传感器：包括一个红外发射管（红外发光二极管）和一个红外接收管（NPN 硅光电晶体管），发射管发出红外光，接收管接收反射回来的光。如果有物体遮挡时（如手），发射管发出的红外光被接收管所接收，否则将接收不到。

接收端的光电二极管将接收到的红外光转换为电流（光电流）。光电流的大小

与接收到的红外光强度成正比。模拟电压信号输入芯片的 ADC 引脚，转换为数字信号（即数字值）。

2.K-Means:K-Means 是一种无监督学习算法,用于将数据分成 K 个不同的簇，通过迭代优化簇中心的位置，使每个数据点都归属于最近的簇中心，从而最小化簇内数据点到中心的距离平方和。

3.总体：每一个手势通过 ADC 采集接收端光电流数据，一共 16 个数据，相当于这个数据的 16 个维度，接着使用 K-Means 算法进行聚类。求出每种手势数据的聚类中心，之后输入新的数据，可以通过求与聚类中心的距离来判断属于哪一个聚类。

方案选择：

这里有两种方案，第一种是训练、推理流程都要在 PC 上实现，将结果推送至单片机并显示。第二种是训练过程在 PC 上实现，推理过程在单片机上实现。

第二种方法显然更好，因为可以不用依赖 PC，只要用移动电源供电就可以使用 PCB。并且在原理上实现也很简单，训练好的结果即三种中心的聚类中心可以直接赋定值给单片机代码的变量中，推理过程就是求输入数据与聚类中心的更新，找到最近的距离，即可实现分类。

值得注意的是，通过前期与老师的交流确认，本实验实现的效果，仅仅是识别“石头”、“剪刀”、“布”手势与“什么都没有”的无遮挡状态，因此，不需要辨认出不属于这三种手势中的任何一种状态的手势，只需要排除空状态后，进行分类操作即可。

三． 电路实现

1.16 个红外对射管

由于 STM32G0 芯片中 ADC 引脚不满 16 个，所以我们将两个红外对射管连在同一个 ADC 引脚上，将 16 个红外对射管分为两组，使用 8 个 ADC 引脚，如图 1 所示。

每组的 8 个红外对射管连在一起，接一个三极管，通过 GPIO 口输出的高低电平信号 BANK0 与 BANK1，控制一组红外对射管的使能。采集数据时分时切换，轮流进行两组的扫描，获取 16 个完整的数据。

考虑到手的大小，我在 15cm*15cm 的 PCB 上 4*4 阵列，尽可能分散地排布对射红外管。接着，因为是两组轮流采集，所以我将两组在行上间隔排列，尽量减少轮流采集对晃动的手势采集造成的问题。具体来说，编号 0~15 的传感器排列阵列为{[0,1,2,3][8,9,10,11][4,5,6,7][12,13,14,15]}

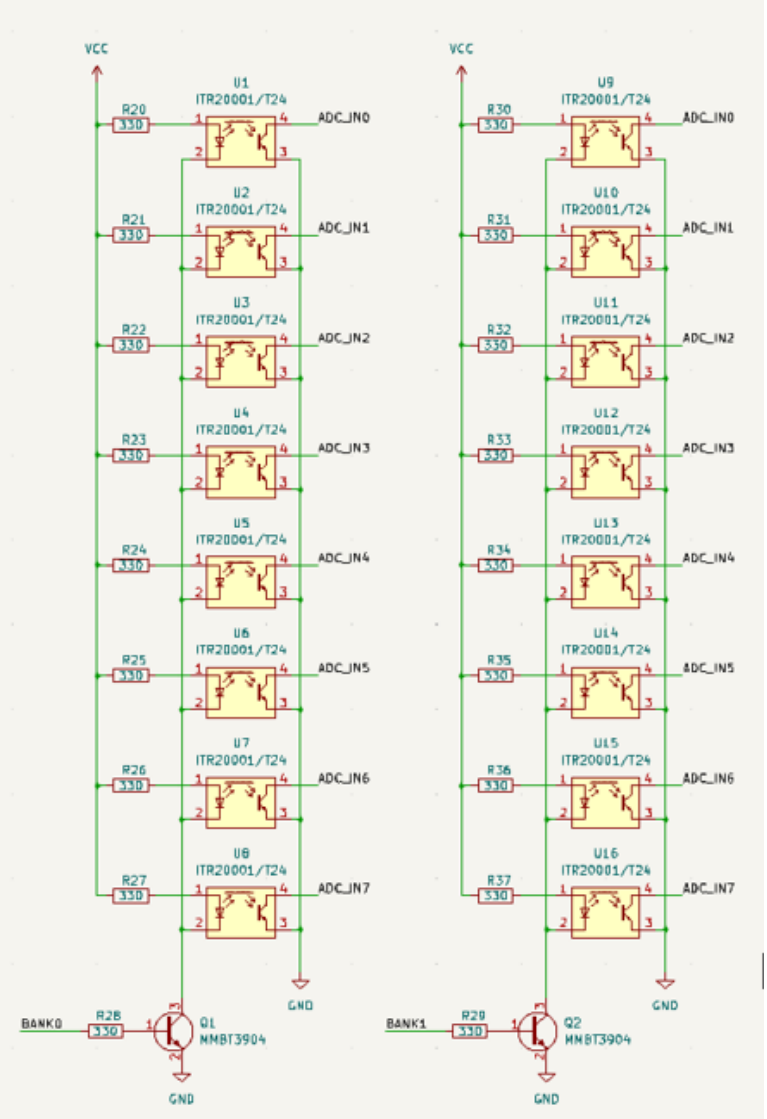


图 1 16 个红外对射传感器

2. 调试接口

SWD 接口：通过在芯片上接出 SWDIO，SWCLK，以及 VCC、GND 给到四个排针，构成 SWD（Serial Wire Debug）调试接口，如图 2 所示，主要用于程序的烧录。本实验中，在最小系统板中焊接对应的排母，排母与排针对应相接，将最小系统板当作下载器，向本作品的 STM32G0 中烧录程序。

UART 接口：在芯片上引出 UART_TX 与 UART_RX,分别接在 CH340N 的 RXD 与 TXD 上，如图 3，因为写要和读接在一起，才能实现数据的传递，这样做就可以通过插 Type-C 接口转 USB 读取串口数据。

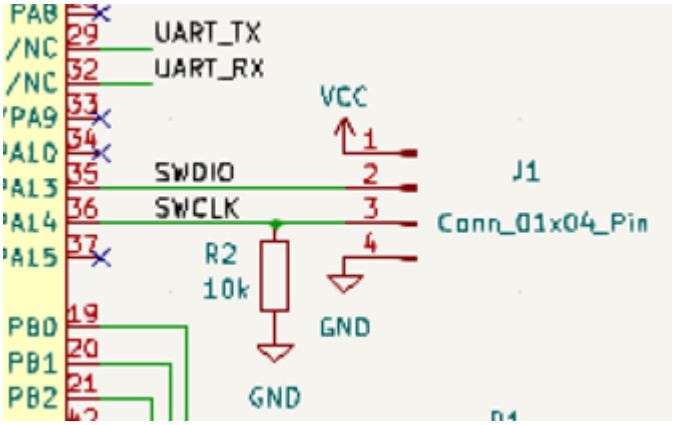


图 2 调试接口

3.USB 通信与取电

本作品使用 USB 口与计算机通信与取电，使用了 USB Type-C 插件，计算机供电时 5V，图 3 中的 VBUS 为 5V 输出。将此插件与 CH340N 相连，实现 USB 转串口的协议转换。UART 串口，在调试接口部分已经介绍。

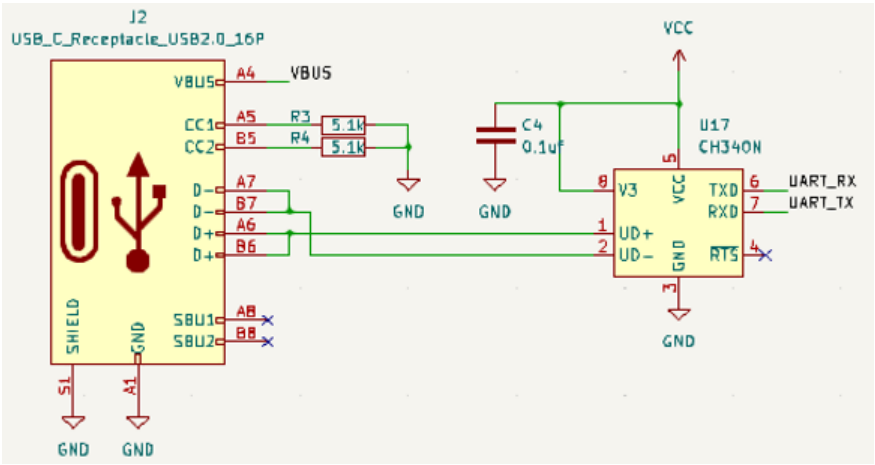


图 3 USB 通信与取电

4.降压获取 3.3V 电源

因为 ADC 连接红外对射管工作电压是 3.3V，因此如图 4 所示设计降压与稳压电路，将直接从计算机取电得到的 5V 的 VBUS 降压并稳定成 3.3V 的 VDD 供给工作模块。

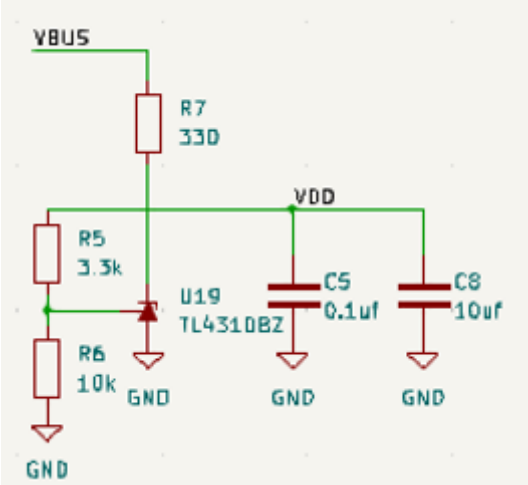


图 4 降压与稳压模块

5.LED 指示灯

将三个 LED 灯接在三个 GPIO 口，当判断出手势的类型之后，通过改变 GPIO 口的高低电平，实现 LED 灯的点亮与关闭。三个 LED 灯分别对应石头、剪刀、布，在 PCB 的丝印层画了对应图像。当无遮挡，即“空”状态，用 3 个 LED 全亮表示。

四. 软件算法

1.CubeMX 引脚配置

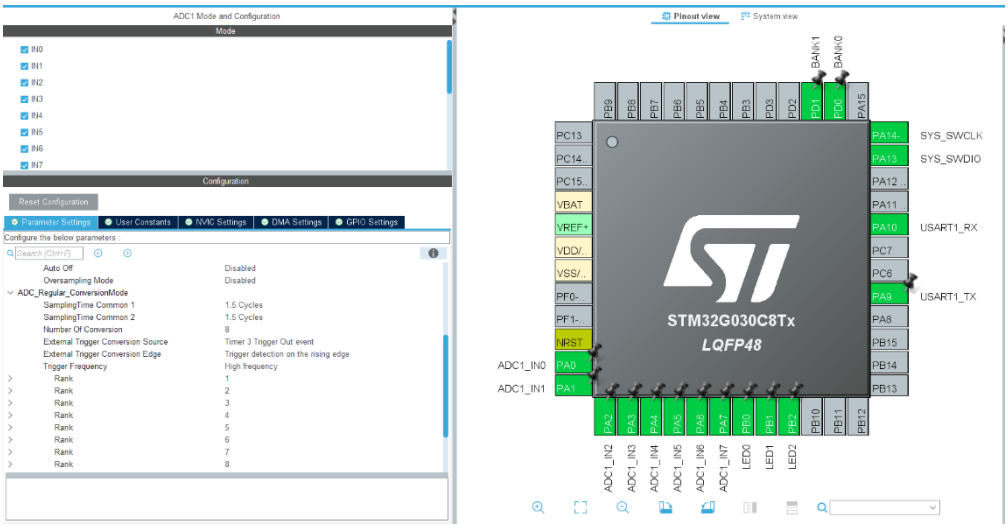


图 5 CubeMX 引脚配置

本设计在嵌入式开发的过程中对于单片机引脚、时钟、中断等的配置和初始化使用 STM32CubeMX 软件实现（如图 5）

其中关键是 8 个 ADC 引脚设置，在 ADC1 的 Parameter Settings 中，设置

Number of Conversion 为 8，因为共有 8 个引脚，同时有 8 个 ADC 信号被采集。还需要设置每个引脚的 Rank 以及对应 Channel。

2. 总流程

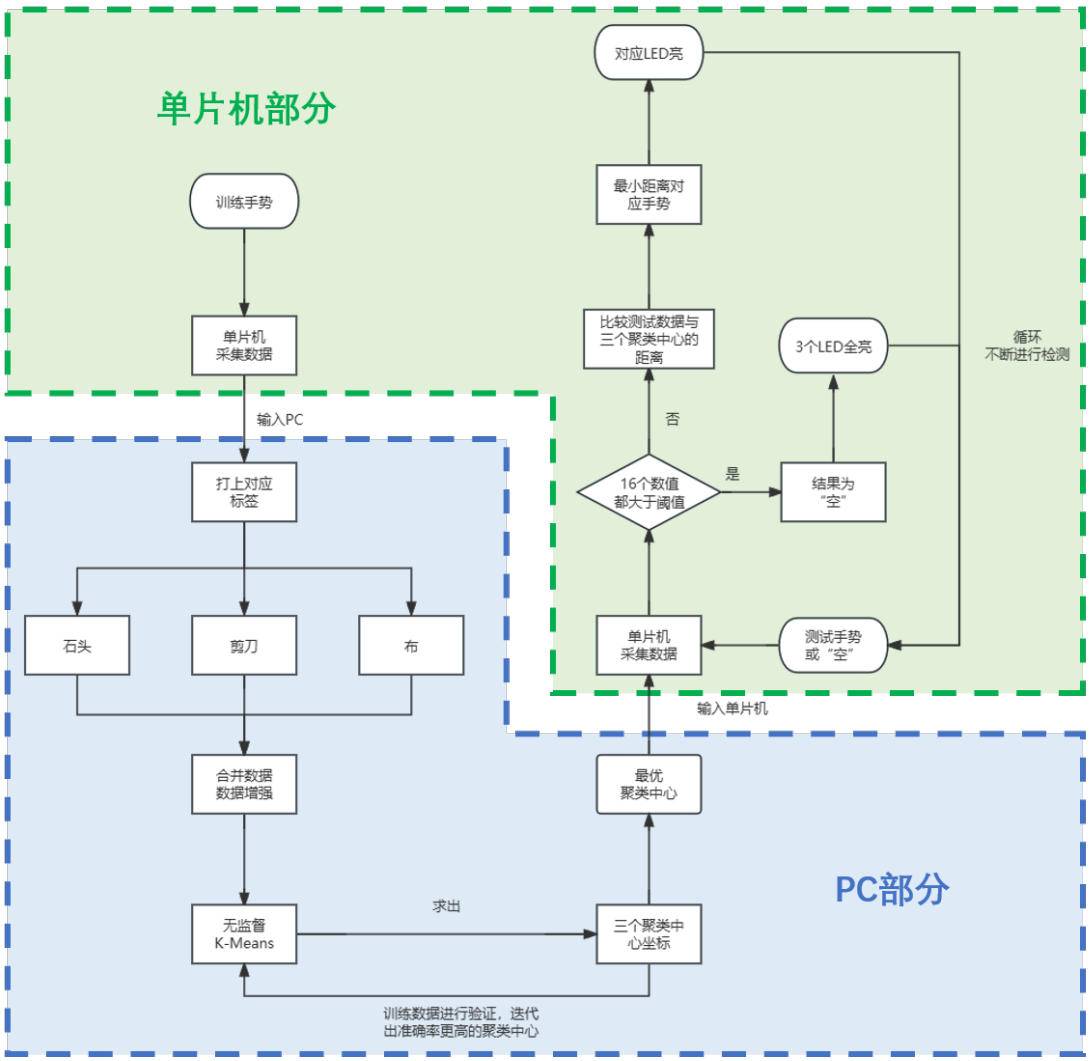


图 6 总流程图

整个算法的总流程如图所示，流程在 PC 部分进行训练，求出聚类中心的结果，在单片机上进行推理，识别出手势。特别强调需要的是，单片机部分和 PC 部分是离线操作的，就是在单片机采集完数据通过串口发送给 PC 后，PC 进行训练，求出聚类中心。聚类中心在单片机代码中直接手动赋值给对应变量，之后计算并比较距离，因此训练与推理过程分离，单片机只需要供电就可以完成推理，不需要连接 PC。

下面是流程描述：

首先采集大量“石头”、“剪刀”、“布”手势的数据，通过串口发送给 PC，分别存储。将数据打上标签后，进行合并。接着进行数据增强，具体来说就是将一个方向的数据拓展到四个方向上。然后用得到的数据进行 K-Means 聚类，聚类采用的是 Sklearn 库函数 K-Means，求得三个聚类中心。之后用所有训练数据进行验证，看距离哪个聚类中心的距离最近，就是对应的手势。因为每次 K-Means 的聚类结果不定，不断循环这个过程，迭代出验证准确度最高的最优聚类中心。

```
for iteration in range(max_iterations):
    # K-means 聚类
    n_clusters = 3
    kmeans = KMeans(n_clusters=n_clusters, random_state=iteration)
    kmeans.fit(data)

    # 获取聚类中心
    cluster_centers = kmeans.cluster_centers_

    # 预测训练数据的标签
    predicted_labels = np.array([classify(d, cluster_centers)[0] for d in data])

    # 计算准确率
    accuracy = compute_accuracy(predicted_labels, labels)

    if accuracy >= accuracy_threshold:
        break
```

图 7 最优聚类中心求解

在单片机代码中复制 PC 的聚类结果，手动赋值给三个聚类中心，之后每次输入的测试数据，首先判断是否每个都大于阈值（即没有遮挡），如果是，那么就是“空”状态，3 个 LED 全亮；否则，求出测试数据与聚类数据距离的平方，比较出最小的那个，就是对应手势。

$$dis = \sum_{i=0}^{16}(x_i - Centre_i)^2 \tag{1}$$

求出对应的手势之后，只亮对应的 LED 表示结果。

```
int pd(float adc[SAMP*2])
{
  int total = 0;
  float rock_dis = 0, scissor_dis = 0, paper_dis = 0;
  for(int i=0;i<16;i++)
  {
     if(adc[i]<2300)
       total++;
     rock_dis += (adc[i]-Rock_Centre[i])*(adc[i]-Rock_Centre[i]);
     scissor_dis += (adc[i]-Scissor_Centre[i])*(adc[i]-Scissor_Centre[i]);
     paper_dis += (adc[i]-Paper_Centre[i])*(adc[i]-Paper_Centre[i]);
  }
  if(total==0)
    return 3;
  int minn, flag=0;
  minn = rock_dis;
  if(scissor_dis < minn)
  {
    minn = scissor_dis;
    flag = 1;
  }
  if(paper_dis < minn)
  {
    minn = paper_dis;
    flag = 2;
  }
  return flag;
}
```

图 8 推理手势

3. ADC 数据采集

在不断循环中，通过切换两个 GPIO 引脚（BANK0 和 BANK1）的状态来控制两组红外对射管的开启和关闭，并使用 DMA 启动 ADC 采样，将采样结果存储到 `AD_DMA` 数组中。每次完成采样后，程序将采样值转换为浮点数存储在 `adc1` 数组中，并通过 `printf` 输出每个通道的 ADC 值到 UART 串口中。

printf 进行了重映射，实现通过串口调试对传感器数值进行检测和后续使用。

值得注意的是，不同于作业三，温度和电压值呈特定的关系，所以 ADC 得到的值需要*3.3/4096 换算到电压值，但是本次的数值作为手势数据的 16 维特征仅用于 K-Means 聚类确定聚类中心，因此不需要做什么处理，K-Means 前也可以在 PC 中归一化，所以直接读取 ADC 值即可。

经过实验，ADC 无遮挡时，数值为 3000+，而有遮挡时，会降低至<2300。

```
while (1)
{
  //set 1 = open; set 0 = close
  HAL_GPIO_WritePin(BANK0_GPIO_Port,BANK0_Pin,1);
  HAL_GPIO_WritePin(BANK1_GPIO_Port,BANK1_Pin,0);
  HAL_Delay(10);
  HAL_ADC_Start_DMA(&hadc1, AD_DMA, SAMP);
  HAL_Delay(100);
  if(dma_end_flag == 1)
  {
    dma_end_flag = 0;
    for(uint8_t i=0;i<8;i++)
    {
      adc1[i] = (float)AD_DMA[i];
      printf("adc%d = %f\r\n",i,adc1[i]);
    }
  }

  HAL_GPIO_WritePin(BANK0_GPIO_Port,BANK0_Pin,0);
  HAL_GPIO_WritePin(BANK1_GPIO_Port,BANK1_Pin,1);
  HAL_Delay(10);
  HAL_ADC_Start_DMA(&hadc1, AD_DMA, SAMP);
  HAL_Delay(100);
  if(dma_end_flag == 1)
  {
    dma_end_flag = 0;
    for(uint8_t i=0;i<8;i++)
    {
      adc1[i+8] = (float)AD_DMA[i];
      printf("adc%d = %f\r\n",i+8,adc1[i+8]);
    }
  }
```

图 9 ADC 数据采集

4.串口通信：手势数据传输

在硬件电路部分，已经介绍了串口输出的硬件电路，因此，插入 Type-C 转 USB 后，PC 上就可以接收串口数据。

串口通信是通过串行接口进行数据传输的一种方式，它按照一定的顺序将数据逐位传输，通常使用 RS-232 标准进行电气信号传输。单片机和 PC 都配有 UART（Universal Asynchronous Receiver/Transmitter）模块，用于实现串口通信。UART 模块将并行数据转换为串行数据并传输，同时将接收到的串行数据转换为并行数据。

如果想要实现实时获取，可以使用 Python 的 pySerial 库等实现，然而由于本作品的流程设计，没有必要实时传输串口数据，因此我选用了更加简单直观的串口调试工具，采集一定量的数据口，复制保存进 txt 文件，python 程序读取 txt 文件得到训练数据，更加方便。

五． 测试方法、结果与分析

1.接线

接线分为烧录和测试两种状态。

烧录时，需要将最小系统板与 PCB 分别通过 Type-C 转 USB 与电脑进行相连，之后最小系统板与 PCB 通过排针排母进行相连，相连时 VCC、SWDIO、SWCLK、GND 对应相连，具体接线如图 10。烧录完按按钮进行 reset。

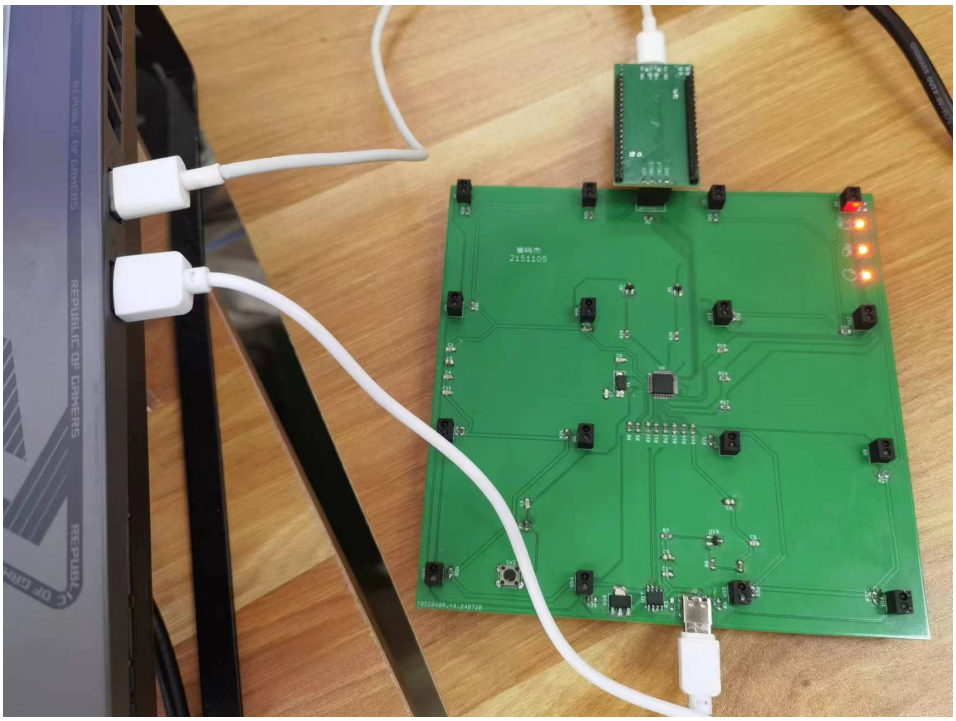


图 10 烧录接线图

测试：

测试时，只需要接 PCB 电源就行，图 11 为测试时接线图，也是空状态效果图。图 12、13、14 分别是“石头”、“剪刀”、“布”的效果图。

测试时需要注意的是，由于红外对射管的局限性，它只能检测极小范围内的遮挡情况，以及红外对射管数量少，分布稀疏，因此做出剪刀手势时，食指和中指需要将最上排红外对射管遮挡，否则对于 ADC 采集的数据信号来说，剪刀和石头无法区分。

测试中，在没有手势时，所有的 LED 全部亮起，而在做出特定手势，并进行合适的遮挡之后，只剩对应的 LED 会亮起，虽然期间会有一定的跳动，那是因为红外对射管的局限性和数量不够多，但是稳定一段时间，合适遮挡后，会正确亮灯，基本符合预期。

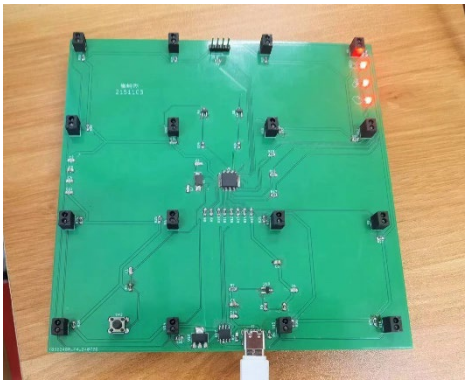


图 11 测试接线及空状态

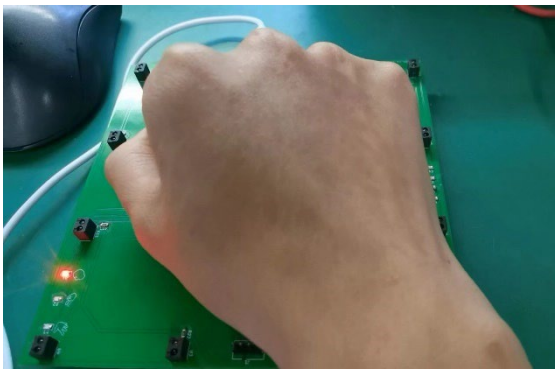


图 12 “石头”效果

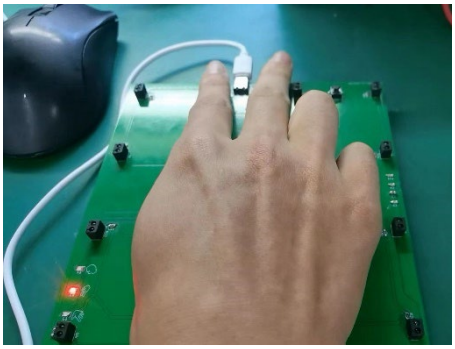


图 13 “剪刀”效果

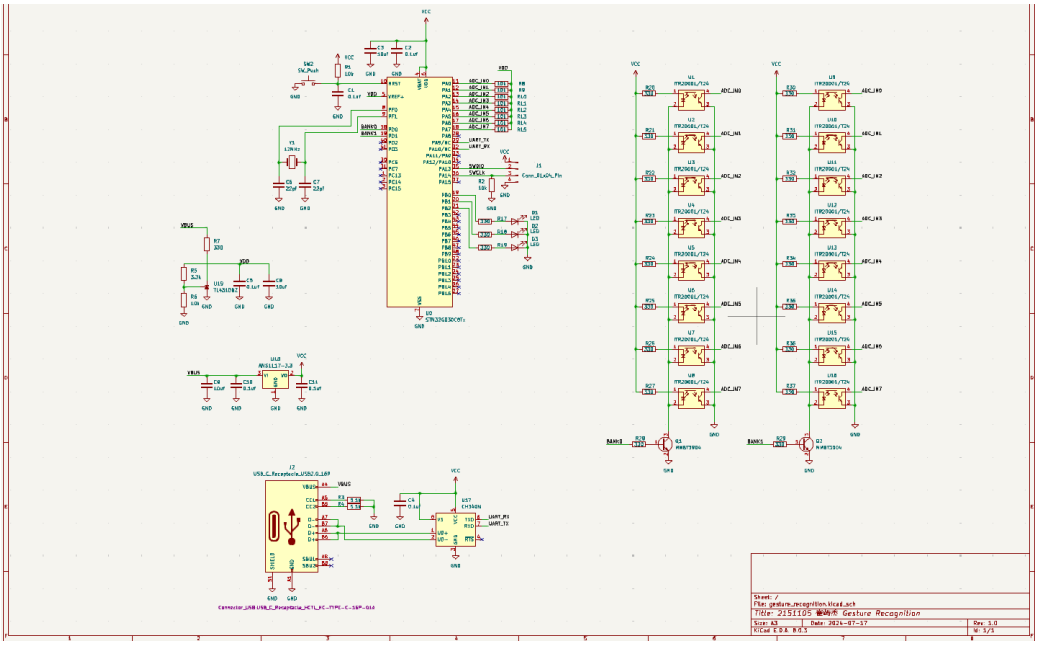
图 14 “布”效果

六. 总结

本次的综合实验全面包含了嵌入式系统开发的软硬件各方面的流程，从 STM32 单片机到 ADC 采集电路入手，完成了从元件到系统的全栈开发。硬件方面，在电路设计、PCB 绘制和焊接上都进行了实践，学习使用了 Kicad 软件开发，以及焊接 STM32G0 芯片，贴片电阻等较高难度元件；软件方面，我学会了在 Linux 系统中使用 VScode 进行开发，虽然在本次综合实验中，由于环境配置等问题还是选用了 MDK-ARM 开发，但是之前的最小系统板开发已经能较好适应，同时我也温习巩固了 CubeMX 的使用配置引脚。

这次综合实验中，虽然在过程中遇到了各种各样的困难，但是得益于蒋磊老师与同伴的帮助，最终顺利完成了实验，能力得到了很大的提升。

附录 A：电路原理图



附录 B：完整电路 PCB