# Lab Assignment 3 CS301 2024FALL

## 作业内容

基于STM32的视频播放器设计与实现

## 任务概述

使用STM32微控制器的串口通信、LCD显示、按键输入和定时器功能，设计与实现一个视频播放器。

播放器可以通过串口接收上位机（PC）发送的视频帧数据，并在LCD屏幕上播放。播放器需要支持视频的播放、停止、画面大小调整、帧存储等基本功能，并选择性支持暂停、继续、快进、慢进等扩展功能。

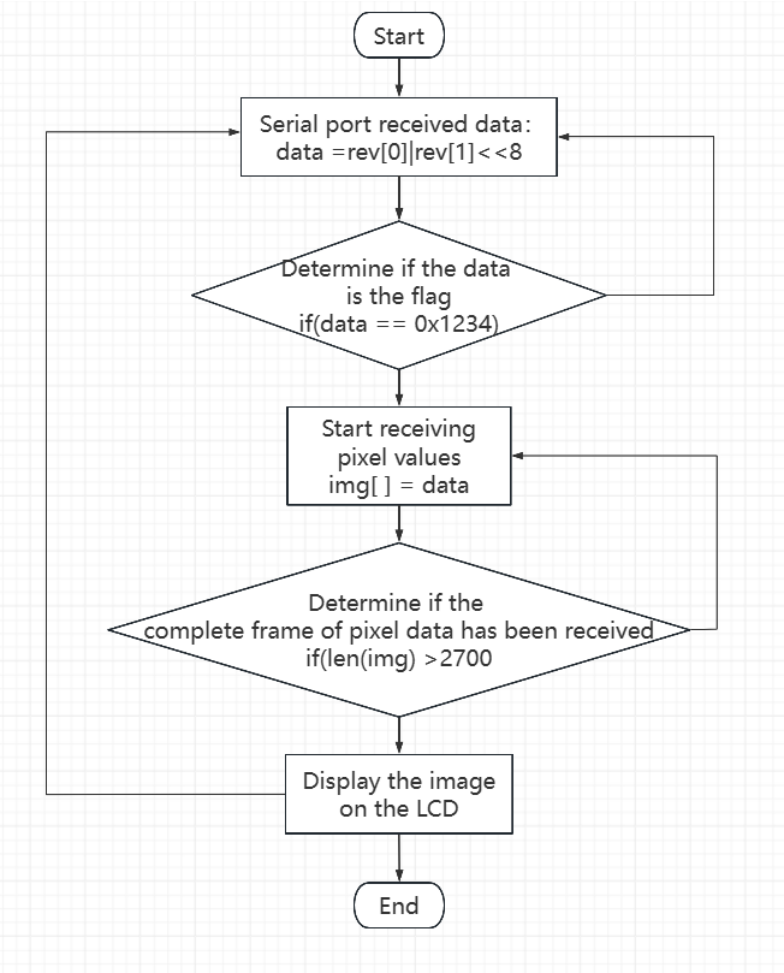
## 任务要求 (no more than 100%)

### 第1部分：基本功能（70%）

#### 任务 1：串口通信与数据接收（20%）

1. 通过串口通信协议接收上位机传来的每一帧画面的像素点信息。
2. 串口通信协议参数为：
    - 波特率为1MHz；
    - 每次只发送一个像素点的信息；
    - 图片格式为16位真色彩格式，分辨率为45*60；
    - 发送每一帧图片的第一个像素点前，首先发送一个标志位0x1234。

      （eg. 传输的数据流为0x1234 0x（一帧的像素数据）0x1234（另一帧的像素数据）...）
3. 这里提供一个数据接收后将接收到的数据处理为一帧完整视频图像的简易流程。




4. 此部分功能，能够显示一帧图像即可得分。

   这里给出LCD显示16位真彩图片的示例代码：

```
void lcd_showpic(uint16_t x, uint16_t y, uint16_t column,uint16_t
row,unsigned short *pic)
{
    uint16_t m, n;
    uint16_t *data = (uint16_t *)pic;

    for (j = 0 + y; j < row + y; j++) {
        for (i = 0 + x; i < column + x; i++) {
                    lcd_draw_point(m, n, *data++);
        }
    }
```

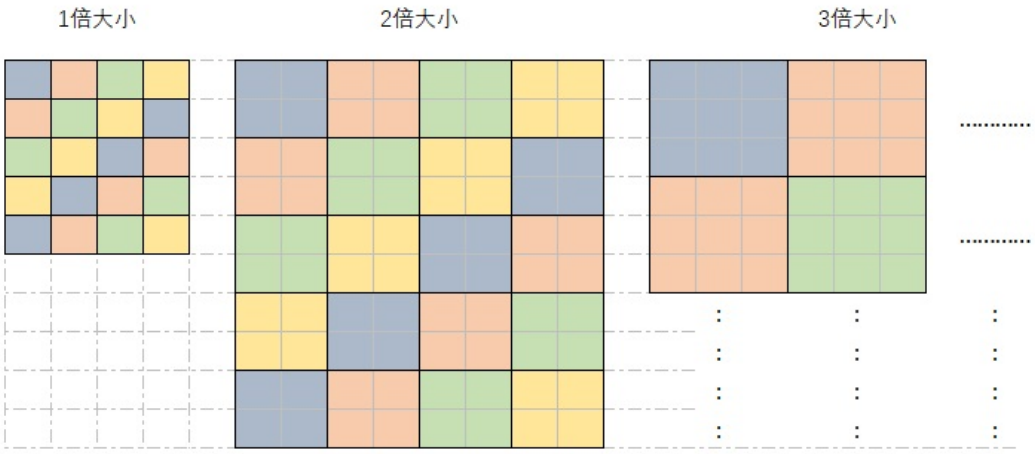
## 任务 2：视频播放（20%）

1. 通过接收到的数据信息，将图片逐帧显示在LCD显示屏上，实现视频播放的功能。
2. 视频播放参考流程。

   接收数据 ----> 判断帧起始数据0x1234 ----> 显示第一帧图像 ----> 接收数据 ----> .....

### 任务 3：视频播放控制（20%）

1. 按下KEY0键，可以对视频画面进行大小调整（15%）
    - 画面大小按照1倍大小、2倍大小、3倍大小的顺序循环调整；
    - 视频正常播放时，按下KEY0键，画面切换至2倍大小；
    - 视频2倍大小播放时，按下KEY0键，画面切换至3倍大小；
    - 视频3倍大小播放时，按下KEY0键，画面切换至1倍大小；
    - 画面调整方式：2倍大小播放时，可将原帧的一个像素点信息，扩大为4个相邻的像素点信息；3倍大小播放时，可将原帧的一个像素点信息，扩大为9个相邻的像素点信息，如下图所示：




2. 按下KEY1键，可以停止播放视频，退出视频播放功能（5%）

### 任务 4：视频播放时间显示（10%）

1. 在LCD屏上显示视频播放的时间
    - 要求使用定时器的计数功能；
    - 不可以使用HAL_Delay( )方法，若使用HAL_Delay( )方法实现时间显示功能，则本部分不得分。

## 第2部分：扩展功能（no more than 40%）

### 任务 5：通信协议修改（20%）

1. 通过修改上位机和STM32开发板的通信协议，实现视频的暂停与继续。（10%）
2. 通过修改上位机和STM32开发板的通信协议，实现视频的快进、慢进功能。（10%）
3. 提示：本部分功能的实现，需要修改通信协议，此处通信协议的修改，涉及PC端及STM32开发板端，其中
    - 上位机端（PC端）需要修改video_uart.py文件；
    - STM32开发板端需要修改工程代码；
    - 双方修改后的协议的具体实现细节，需要根据自己设计的系统，自行定义并实现。

### 任务 6：帧存储与显示（10%）

1. 按下KEY_WKUP可以存储当前播放的一帧画面，并显示在视频框的下面。

### 任务 7：播放器UI设计（10%）

1. 可以自行设计播放器的UI，将对应信息以合理的布局显示出来。

### 任务 8：其他功能（no more than 40%）

1. 能够适配本系统的其他合理的功能模块。
2. 每一个功能点的得分最多不超过15%，具体得分需要按照实现难度的不同，参照前面任务的难度，给出适量的分数。
3. 功能代码量过于简单的，可能不会获得分数。

## 提交要求

- 在截止日期前完成实验。
- 将整个项目打包成压缩包，并提交至Blackboard平台。
- 作业要求个人单独完成，不按照project分组完成。

## 实验环境

- STM32微控制器开发板
- STM32CubeIDE软件
- 串口调试助手或上位机软件
- LCD显示屏
- 按键

## 实验步骤

1. **系统基本配置**：设置按键KEY_WAKEUP、KEY0和KEY1为输入设备，LCD屏幕为显示设备。
2. **视频显示**：接收上位机发送的像素点信息，并在LCD屏幕上显示。
3. **视频播放控制**：通过按键调整视频画面大小和控制视频播放状态。
4. **视频播放时间显示**：使用定时器计数，并在LCD屏幕上显示视频播放时间。
5. **通信协议修改**：修改上位机和STM32开发板的通信协议，实现视频的暂停、继续、快进和慢进功能。
6. **帧存储与显示**：通过按键存储当前播放的帧，并在视频框下方显示。
7. **播放器UI设计**：设计播放器的用户界面。

## 注意事项

- 确保代码编写规范，变量、宏定义和函数声明应位于相应位置。
- 确保代码注释清晰，便于理解和维护。
- 确保所有功能模块化，便于测试和验证。、

# 补充材料

## 系统运行流程

1. 上位机的串口通讯采用video_uart.py文件进行，不使用串口通信助手。
2. 修改上位机程序video_uart.py，适配于自己的环境。具体修改包括2处位置：
    - video_path变量，指向视频文件在上位机的存储路径
    - ser变量，将serial.Serial( )方法中的第一个参数，指定为自己PC机的串口号。
3. 使用串口线连接PC机及STM32开发板。
4. 在STM32开发板上运行视频播放程序。
5. 运行上位机video_uart.py程序，开始数据传送。
6. 在STM32开发板上观察视频播放效果及各种功能的视线。

## 上位机程序的使用方法

### 视频处理+串口助手程序video_uart.py

vedio_uart.py > ...

```python
import cv2
import numpy as np
import serial
import time  # 导入time模块用于等待
```

导入opencv库，如果没有需要自行安装

```python
# 视频文件的路径
video_path = r"C:\SUSTECH\LESSON\TA\TASK3\apt.mp4"
```

设置视频素材的文件路径

```python
# 创建串口对象
ser = serial.Serial("COM13", 1000000, 8, "N", 1, timeout=None)
```

创建串口，串口号即串口下载时使用的串口号，每台PC机有所不同，可以通过其他串口助手程序来查看

```python
# 计算每秒需要处理的帧数
frames_per_second = 8
```

这里可设置程序每秒处理的视频帧数，以此控制视频播放的速度

```
# 将RGB图像转换为16位真色彩（RGB565格式）
for y in range(height):
    for x in range(width):
        r = (resized_frame[y, x, 0] >> 3) << 11  # 红色5位
        g = ((resized_frame[y, x, 1] >> 2) & 0x3F) << 5  # 绿色6位
        b = (resized_frame[y, x, 2] >> 3) & 0x1F  # 蓝色5位
        frame_16bit[y, x] = r | g | b
```

串口发送函数，用于通过串口向单片机发送处理好的视频像素点信息