프로젝트 계획서

압축된 BMP이미지 파일의 변환

작성일	2022.10.12-10.20	제출일	2022.10.21
과 목	어드벤처디자인 3분반	전 공	컴퓨터공학
담당교수	김경수	조 명	1조
조 장	송제용	조 원	김현진
조 원	안현진	조 원	정지수

목 차

1. 문제 이해

본 보고서는 중간레벨 프로젝트 계획 보고서임을 명시한다.

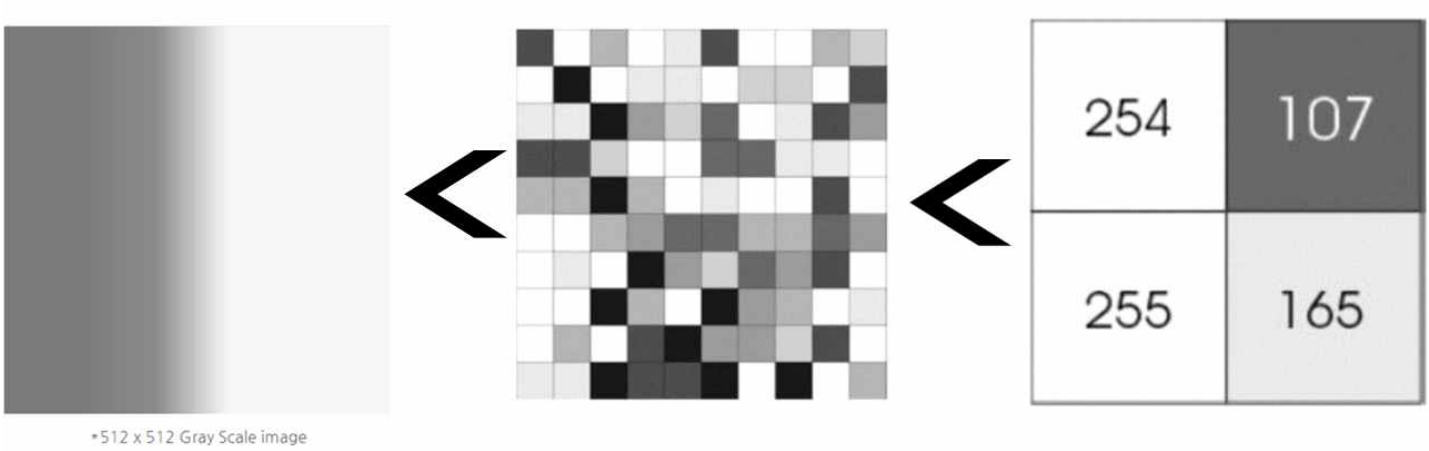
1-1. 프로젝트의 목표

이번 프로젝트의 목적으로써는 rle방식으로 압축된 8-bit gray-scale 영상에 대해서 압축 해제하고(1) 그 결과물을 이용하여 사각형 그리기, 색상 반전을 적용한 뒤(2)이 결과를 저장할 수 있는(3) 프로그램을 만드는 것이다. 이러한 목표를 위하여 rle 방식의 압축이 어떤 식으로 이루어지는지(1), 영상의 밝기 반전은 어떻게 실행할 수 있는지(2), 사용자가 지정한 크기만큼 사각형을 그리려면 어떻게 해야 하는지(3)에 대해서 구체적으로 조사해보고 생각할 필요가 있다.

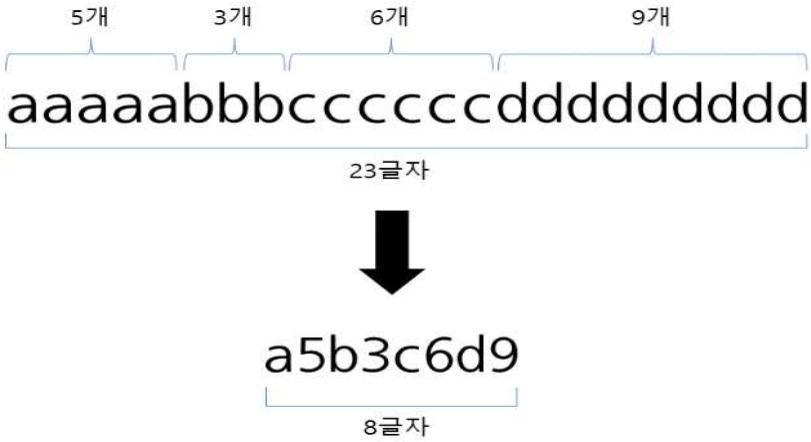
1-2. 문제 정의

이번 프로젝트의 문제는 "**RLE 방식으로 압축된 8bit gray-scale 영상을 저장하고 있는 BMP파일에 대해서 압축을 해제하고 그에 대한 색상 반전, 사각형 그리기를 수행한 후에 그 결과를 저장하라**"는 것이다. 우선 색상 반전, 사각형 그리기와 같은 세부적 기능은 문장 그대로 뜻을 파악할 수 있으므로 구현에 관한 세부적인 내용에 대해선 추후 다루도록 하겠다, 우선 우리가 알아야 할 키워드는 크게 **RLE 방식**, **8-bit gray-scale영상**, **BMP파일** 3가지이다.

먼저 **BMP 파일**이란 이미지 파일의 가장 기초적인 형식이라고 할 수 있으며 픽셀의 사각형 배열에서 각 픽셀의 색을 지정하는 비트 배열을 의미한다. 즉, 이미지를 수많은 점의 집합으로 보고 그 점에 해당하는 색상 값이 배열에 채워져 있는 형태라고 볼 수 있다. 이러한 방식으로 이미지를 표현할 시 컴퓨터에 부담을 덜 주기 때문에 주로 웹에서 이미지를 표시할 때 많이 사용하곤 하며 우리가 흔히 다루는 jpg, png도 비트맵 형식이라고 볼 수 있다.



그리고 **8bit gray-scale 영상** 역시 비트맵의 일종이라고 생각할 수 있다. 비트맵에 대해선 1bit부터 24bit의 형태로 나타낼 수 있는데 우리는 이번 프로젝트에서 8-bit의 비트맵을 사용한다. 여기서 말하는 bit수는 한 픽셀당 할당할 수 있는 색의 개수로 8bit gray-scale 영상의 경우 한 픽셀당 256가지의 색을 할당할 수 있다. gray-scale이란 한글로 직역하면 "회색조"라는 의미로 픽셀 당 사용되는 색상들이 모두 단일 색상으로 존재하며 픽셀 당 사용할 수 있는 256개의 값들이 밝기 레벨을 표현한다는 특징이 있다. 즉, 하나의 픽셀에 256개의 서로 다른 밝기 레벨을 가진 값들 중 하나로 채워 넣을 수 있다는 것을 의미한다. 정리하자면 8bit gray-scale은 각 픽셀당 256가지의 밝기 레벨을 담을 수 있는 비트맵 형식이다.



우리는 RLE방식으로 압축된 이러한 파일을 입력으로 받아 압축 해제, 사각형 그리기, 색상대비 등의 문제를 수행할 수 있어야 한다. **RLE방식**이란 Run-length-Encoding의 줄임말로 무손실 압축의 일종이며 간단한 압축기법 중 하나이다. 한 문자가 몇 번 반복되는지를 저장하는 방식이며 이러한 특성으로 인해 여러 번 반복되는 데이터에 대해 효율적인 방법이다. RLE방식은 encoded mode와 absolute mode 두 가지 방식 중 하나를 사용하여 각 byte들에 대한 압축 해제를 진행하게 된다. 이 부분에 대해서는 이 글의 다른 본문에서 자세히 다뤄 볼 예정이다.

2. 입력 및 출력과 실행흐름도

2-1. 입출력 및 내부 데이터 분석

입출력 및 내부데이터 분석은 후에 구현보고서에 들어가야 맞는 내용이지만 프로그램의 이해와 문제 이해 부분의 비트맵의 설명이 부족한 것 같아 비트맵의 이해를 돕기 위해 기술하였음.

입력 데이터
-RLE 방식으로 압축되거나 되지않은 8bit gray-scale BMP 파일(BMP 파일이 아닐 경우 에러메세지 출력)

내부 흐름 데이터
-만약 RLE방식으로 압축된 8bit gray-scale BMP 파일일 경우 압축풀기
-8bit gray-scale BMP 영상을 밝기반전시키기
-8bit gray-scale BMP 영상을 사용자가 지정한 크기의 사각형을 그리기
-8bit gray-scale BMP 영상의 이름을 지정

출력 데이터
-압축하지 않은 8bit gray-scale BMP 파일 (밝기 반전 및 사각형 추가)

내부 데이터 분석

데이터 블록 (.bmp => hex)

```
42 4D 36 04 04 00 00 00 00 00 36 04 00 00 28 00
00 00 00 02 00 00 00 02 00 00 01 00 08 00 00 00
00 00 00 00 04 00 00 00 00 00 00 00 00 00 00 01
00 00 00 00 00 00 00 00 00 00 01 01 01 00 02 02
02 00 03 03 03 00 04 04 04 00 05 05 05 00 06 06
06 00 07 07 07 00 08 08 08 00 09 09 09 00 0A 0A
0A 00 0B 0B 0B 00 0C 0C 0C 00 0D 0D 0D 00 0E 0E
0E 00 0F 0F 0F 00 10 10 10 00 11 11 11 00 12 12
12 00 13 13 13 00 14 14 14 00 15 15 15 00 16 16
16 00 17 17 17 00 18 18 18 00 19 19 19 00 1A 1A
1A 00 1B 1B 1B 00 1C 1C 1C 00 1D 1D 1D 00 1E 1E
1E 00 1F 1F 1F 00 20 20 20 00 21 21 21 00 22 22
22 00 23 23 23 00 24 24 24 00 25 25 25 00 26 26
26 00 27 27 27 00 28 28 28 00 29 29 29 00 2A 2A
2A 00 2B 2B 2B 00 2C 2C 2C 00 2D 2D 2D 00 2E 2E
2E 00 2F 2F 2F 00 30 30 30 00 31 31 31 00 32 32
32 00 33 33 33 00 34 34 34 00 35 35 35 00 36 36
36 00 37 37 37 00 38 38 38 00 39 39 39 00 3A 3A
3A 00 3B 3B 3B 00 3C 3C 3C 00 3D 3D 3D 00 3E 3E
3E 00 3F 3F 3F 00 40 40 40 00 41 41 41 00 42 42
42 00 43 43 43 00 44 44 44 00 45 45 45 00 46 46
46 00 47 47 47 00 48 48 48 00 49 49 49 00 4A 4A
4A 00 4B 4B 4B 00 4C 4C 4C 00 4D 4D 4D 00 4E 4E
4E 00 4F 4F 4F 00 50 50 50 00 51 51 51 00 52 52
52 00 53 53 53 00 54 54 54 00 55 55 55 00 56 56
56 00 57 57 57 00 58 58 58 00 59 59 59 00 5A 5A
```

위 사진은 girl.bmp 파일을 헥사값으로 변경한 이미지입니다. 파란색 배경이 헤더와 비트맵 정보를 나타냅니다.
(위 이미지에서는 일부만 나온것이며 실제로 263,222개의 HEX값이 있습니다.)

(헤더는 리틀 인디언 방식을 사용)

ex) 00 01 00 00 => 00 00 01 00
- BMP 헤더 : BMP 파일에 대한 일반 정보를 담고 있다. (빨간색 드래그 부분)
이 바이트 블록은 파일의 시작 부분에 있으며 파일을 식별하는 데 사용됩니다. 일반적인 응용 프로그램은 파일이 실제로 BMP 파일이고 손상되지 않았는지 확인하기 위해 먼저 이 블록을 읽습니다
0 ~ 1 : BMP 파일을 식별하는 데 쓰이는 매직 넘버: 0x42 0x4D (B와 M에 대한 ASCII 코드 포인트)
42 4D : BM 으로 비트맵을 나타냅니다.
2 ~ 5 : BMP 파일 크기 (바이트 단위)
36 04 04 00 : Little Endian방식으로 되어있으므로 "00 04 04 36" => 263,222bytes
6 ~ 7 : 예약필드. 실제값은 그림을 만든 데 쓰인 응용 프로그램에 따라 달라진다.
8 ~ 9 : 예약필드. 실제값은 그림을 만든 데 쓰인 응용 프로그램에 따라 달라진다.
10 ~ 13 : 비트맵 데이터를 찾을 수 있는 시작 오프셋 (바이트 단위)
36 04 00 00 : 1078bytes
- 비트맵 정보(DIB 헤더) : 비트맵 그림에 대한 자세한 정보를 담고 있다. (파란색 드래그 부분)
이 바이트 블록은 화면에 이미지를 표시하는 데 사용되는 이미지에 대한 자세한 정보를 애플리케이션에 알려줍니다.
14 ~ 17: 이 헤더의 크기 (40 바이트)
28 00 00 00 : 40bytes
18 ~ 21: 비트맵 가로 (단위는 화소, signed integer).
00 02 00 00 : 512
22 ~ 25 : 비트맵 세로 (단위는 화소, signed integer).
00 02 00 00 : 512
26 ~ 27 : 사용하는 색 판(color plane)의 수. 1로 설정해야 한다.
01 00 : 1
28 ~ 29 : 한 픽셀에 들어가는 비트 수이며 그림의 색 깊이를 뜻한다. 보통 값은 1, 4, 8, 16, 24, 32이다.
08 00 : 한픽셀당 비트수 : 8bit
30 ~ 33 : 압축 방식. 가능한 값에 대한 목록은 다음 표를 참조하라.

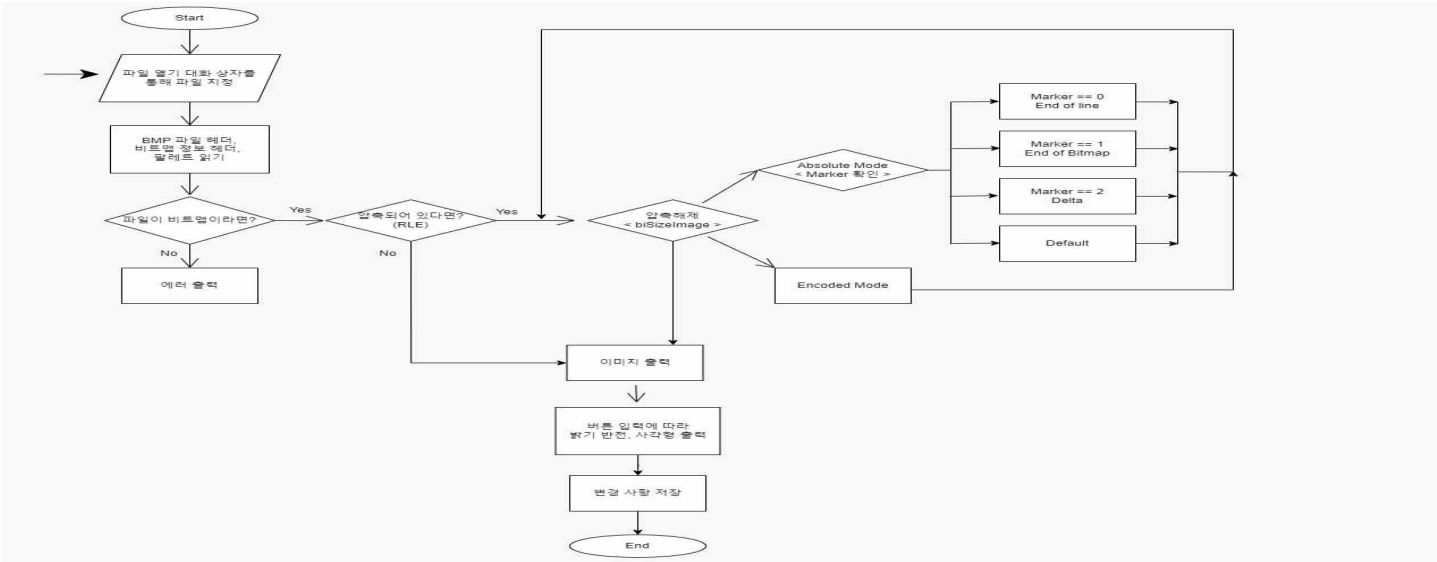
값	식별자	압축 방식	비고
0	BI_RGB	없음	가장 일반적이다
1	BI_RLE8	RLE 8비트/화소	8비트/화소 비트맵에만 사용할 수 있다.
2	BI_RLE4	RLE 4비트/화소	4비트/화소 비트맵에만 사용할 수 있다.
3	BI_BITFIELDS	비트 필드	16, 32비트/화소 비트맵에만 사용할 수 있다.
4	BI_JPEG	JPEG	비트맵은 JPEG 이미지를 포함한다.
5	BI_PNG	PNG	비트맵은 PNG 이미지를 포함한다.

34 ~ 37 그림 크기. 압축되지 않은 비트맵 데이터의 크기(아래 참조)이며, 파일 크기와 혼동하지 말 것.
38 ~ 41 그림의 가로 해상도. (미터 당 화소, signed integer)
42 ~ 45 그림의 세로 해상도. (미터 당 화소, signed integer)
46 ~ 49 색 팔레트의 색 수, 또는 0에서 기본값 2n.
00 01 00 00 : 256
50 ~ 53 중요한 색의 수. 모든 색이 중요할 경우 0. 일반적으로 무시.
00 00 00 00
- 색상 테이블
10 ~ 13 내용을 보시면 시작 오프셋이 1078bytes 인것을 확인 할 수 있습니다.
1078 - (14+40) 시작 오프셋 - (헤더 + 정보) : 1024bytes 입니다.
이 1024bytes는 색상 테이블 정보입니다.
위 사진은 4바이트 RGBA32 형식으로 총 256개의 색상이 있으며 각 색상은 4바이트를 차지하여 총 256 * 4 : 1024bytes입니다.
ex) rgba(44, 44, 44, 0)

추가로 압축된 비트맵의 내부데이터 분석은 후에 구현보고서에서 기술하여 구현보고서의 이해를 돕겠음.

2-2. 실행 흐름도

실행 흐름도는 후에 구현보고서에 들어가야 맞지만 앞으로 바뀔 수 있는 실행 흐름도의 기본구조이기도 하고 이해를 돕기위해 기술한다.



3. 실행 예시

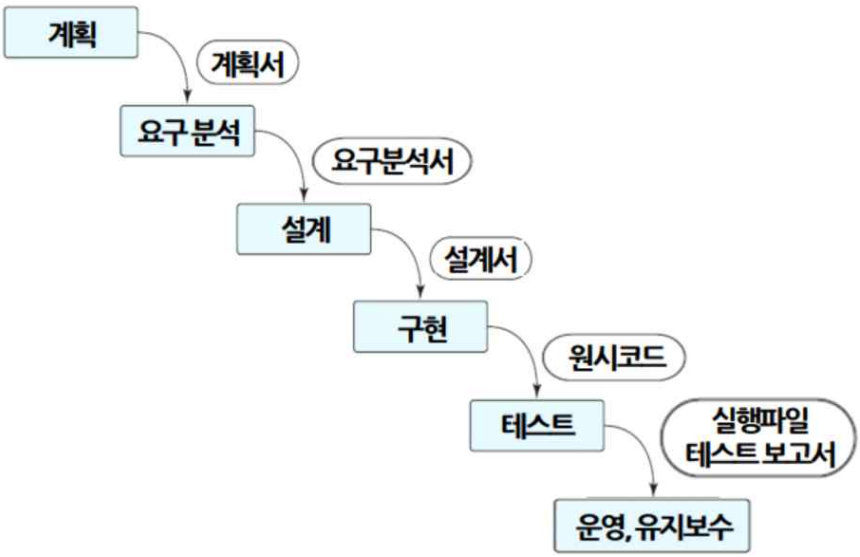
(실행예시를 그림으로 도식화함)

유저가 이용하기 용이하도록 GUI 형식으로 파일을 업로드, 그리고 버튼으로 영상의 색상을 반전하는 것과 사각형을 그리는 것을 구현하는 것이 목표이다. 하지만 중간 레벨에서의 주어진 시간은 그렇게 여유롭지 않은 관계로 현재 목표는 주어진 문제를 콘솔에서 파일 입출력을 통해 처리하는 것이고, 시간이 남는다면 추가적으로 MFC를 구성하는 것이다.

4. 개발과정 고려사항

4-1. 방법론

현재 중간 레벨 프로젝트의 진행 방법은 다음과 같다.
① 프로젝트 계획 수립 및 발표
② 프로젝트 설계 및 발표
③ 최종 보고서 작성 및 발표
따라서 본 프로젝트 내에서는 소프트웨어개발 프로세스 모델 중 하나인 폭포수 모델을 기반으로 중간레벨 문제의 설계와 구현을 심층적으로 구조화를 하기로 하였다.



4-2. 구현 언어 및 고려사항

-IDE :VisualStudio 2019
-구현 언어 :C++
-UI : MFC 혹은 콘솔
-RLE방식으로 압축한 BMP 파일은 포토샵에서 저장하여 사용

현재 생각하고 있는 구현의 방향은 Visual Studio를 사용하여 비트맵 헤더파일을 가지고 있는 C++, C언어 중 하나를 사용할 예정이다. 콘솔의 파일 입출력을 통해서 각 기능을 모두 구현이 완성된 후 추가적인 MFC GUI 설계와 구현을 통한 리팩토링을 고려하여 C++을 할 가능성이 높다. 또한 RLE방식으로 압축한 BMP 파일은 포토샵에서 저장하여 사용할 것이다.

5. 접근 방법 및 간단한 아이디어

5-1. 기술적 요구

문제를 분석하고 실제 프로그램으로 구현하는 과정에서 기술적으로 요구될 몇 가지 사항들은 아래와 같다.

1) 비트맵의 이미지는 상하가 뒤집힌 상태로 저장이 되므로 이를 해결할 방법이 필요하다.
2) 비트맵 파일의 가로길이는 항상 4바이트의 배수로 고정 시켜야 하므로 4바이트의 배수가 아닌 데이터 삽입 시 해결방안이 필요하다.
3) RLE 파일 압축 해제 시 각 모드에 따라 어떻게 읽어올 것인가에 대한 방법이 필요하다.
4) 밝기 반전 및 상하반전 등 영상처리 시에 어떻게 기술적으로 구현할 것인지에 대한 방법이 필요하다

5-2. 간단한 아이디어

비트맵 읽어오기의 경우 비트맵 파일은 상하 반전되어 저장되어 있으므로 반대로 밑에서부터 읽어오면 된다.

비트맵 파일의 가로길이는 항상 4바이트의 배수로 고정 시켜야 하므로 가로길이가 4의 배수가 아니라면 나머지 부분을 0으로 채워 4의 배수로 만들어 준다.

RLE 파일 해제의 경우 RLE 파일을 가져와서 앱솔루트 모드와 인코디드 모드를 나누어서 반복문 실행한다. 마커인 경우, 각 상황에 맞게 종료해주거나 위치를 옮겨준다. 불연속적인 픽셀들은 앱솔루트 모드이므로 홀수인 경우 00을 더해주고 인덱스값을 그만큼 높인다. 인코디드 모드는 연속된 값만큼 삽입한다.

상하 반전 구현 아이디어는 원래 bmp 파일은 파일의 특성 상 리틀 엔디안 방식을 고려하여 구현해야 하기 때문에 비트들을 반대로 출력해야 한다. 그러나 비트맵을 읽어온 그대로 프린트를 해주어 상하 반전 기능을 구현한다.

영상 밝기 반전은 픽셀 값을 반전하기 위해 0에 가까운 어두운 픽셀 값은 255에 가깝게 만들어주고, 255에 가까운 밝은 픽셀은 0에 가까운 픽셀 값으로 만들어주어 구현한다.

6. 추가 기능 & 기대효과

추가적인 기능은 다다익선 이므로 우리가 생각했던 것 들 중 현실성이 어느정도 내포되어있는 기능들을 아래 문단에 서술하였다.

현재 우리 조의 목표를 콘솔로 결과물을 출력하는 구현이나, 더 나아가 유저가 사용하기 좀 더 편리하게 GUI 기능으로 리팩토링하여 좀 더 결과물을 완성도 있게 제작할 수 있다. 이는 구현에 목적을 넘어 사용자가 편리하게 사용하는 것에 그 목적이 있으므로 기존 목표의 구현이 확실하게 이루어진 후 살펴봐야 할 점이기도 하다.

또 다른 추가 기능은 색상반전 외에도 영상 파일 좌우 및 상하로 반전 할 수도 있고 색상을 유저가 원하는 만큼 밝기를 조절할 수 있게 기능을 구현할 수도 있다. 따라서 우리가 기존에 접해온 영상의 편집이나 처리를 거의 동일하게 적용할 수 있고 이에 따라 반사이익도 얻을 수 있다.

기대효과로는 압축을 푸는 알고리즘을 설계하고 구현하였으면 역설계 역시 가능하므로 비트맵 파일 자체를 압축 할 수 있는 기능도 한 번 고려해 볼 수 있다. 이는 이미지 파일의 전송이 필요한 저성능 컴퓨터, 즉 IoT 장비나 이와 결을 같이 하는 임베디드 시스템에서 이미지를 사용하고 전송할 때 매우 효율적이라고 볼 수 있다.

추가적으로, 0에서 255(unsigned character)사이, 즉 8bit로 표현할 수 있는 데이터 형식에 대해 똑같이 RLE 방식 압축과 해제가 가능하므로 기존 코드의 재사용을 통해 형식에 구애받지 않고 유연하게 접근할 수 있다는 점이 존재하여 이 역시 긍정적으로 생각해 볼 수 있다.

7. 추진 일정 및 역할 분담

초기에는 위 사진과 같이 구성하였으나 계획의 변경과 역할 분담 기술을 위해 간트차트를 이용하여 기술하겠음.

(수행 기간 : 2022년 10월 12일 ~ 2022년 11월 16일)						
기간(주) / 개발내용	1	2	3	4	5	담당 팀원
문제분석 및 자료조사	O					전원
계획 보고서 작성	O					전원
설계 보고서 작성 및 발표자료 제작		O	O			김현진, 안현진
프로그램 구현			O	O	O	송제용, 정지수
결과 보고서 작성 및 발표자료 제작					O	김현진, 안현진

모든 개발내용들을 팀원 다같이 다룰 예정이긴하나 중점적으로 담당할 사람을 기술 하였음. 담당할 팀원은 프로젝트를 진행하며 유동적으로 조정가능. 보고서마다 각각 역할 분담 내용을 다시 기술하겠음

초반개발내용들은 효과적인 문제이해 및 개발계획을 위해 전원이 담당 하였음.