

STM32F103 기반 그래픽 LCD 임베디드 팩맨 게임 구현



김주희 신홍민 이윤철





CONTCT

STM32F103

01 프로젝트 수행 개요

04 구성 및 동작

02 개발 환경

05 최종

03 프로젝트 블록도

06 결과 및 고찰

01 프로젝트 수행 개요

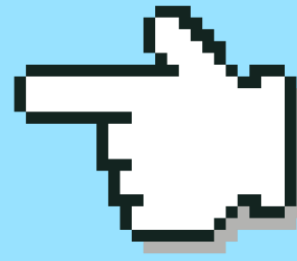


1) 프로젝트 목표

NUCLEO-F103RB 보드와 ILI9341 LCD를 이용하여 **팩맨 게임** 구현

- ILI9341 **LCD 병렬 통신**을 통해 실시간 **게임 그래픽 출력**
- **피에조 부저**(PWM 채널 활용)를 통해 게임 **배경을** 및 **효과음** 재생
- 4개의 **GPIO 스위치**로 조이스틱 구현
- 최고 점수 차순 **기록 저장 (I2C EEPROM)**
- 시스템 상태 리셋 및 완료 조건에 따른 상태 전환 제어 구현

2) 일정



DAY 1

프로젝트 주제 (**팩맨 게임**) 선정, 목표 설정

DAY 2

하드웨어 회로 구성 (GPIO 스위치·LCD), 게임 그래픽 **리소스 제작**(팩맨, 고스트, 도트 등)

DAY 3

입력 기반 캐릭터 동작 구현 (**스위치 GPIO Pin 활용**)

DAY 4

LCD와 MCU 인터페이스 구현 (**병렬 데이터**), 사운드 구현 (피에조 부저 **PWM 제어**)

DAY 5

점수 표시용 CLCD I2C 통신 구현, **시스템 통합** (부저, 점수 CLCD, 게임화면 LCD, 스위치)

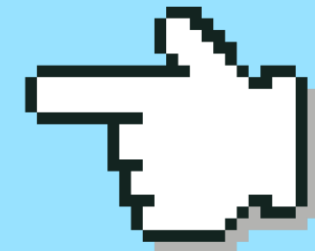
DAY 6



게임기 외형(틀) 제작, 개발 내용 정리 및 문서작업



3) 업무 분담



★ 하드웨어 회로 구성	신홍민
★ 팩맨, 고스트, 실시간 출력 핵심 코드 제작	김주희
★ 사운드 및 효과음 코드 제작	이윤설
★ 게임 화면 그래픽 코드 제작	신홍민
★ 코드 통합 핵심 구현	김주희
★ 블록 다이어그램	ALL
★ PPT 제작	이윤설
★ 외형 제작	이윤설
★ 발표	김주희

02 개발 환경

▶ 하드웨어 환경

하드웨어	종류 / 총 용량	사용량 (사용률)
CPU core	ARM Cortex-M3	
Flash Memory	128 KB	32.43KB (25.7%)
SRAM	20 KB	4.05 KB (20.7%)

▶ 소프트웨어 환경

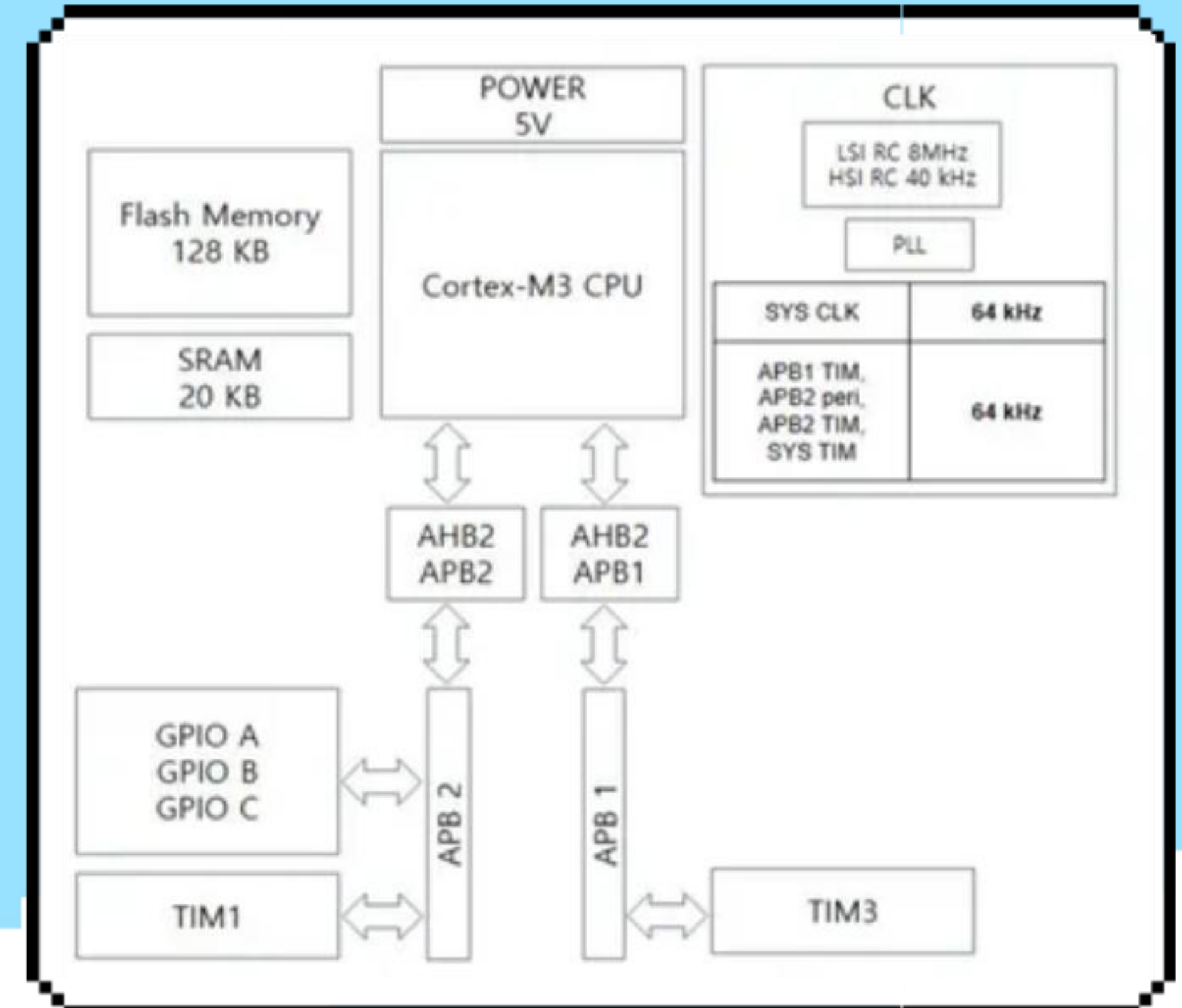
컴파일러: ARM GCC 내장, 최적화 빌드 지원

디버깅: ST-Link를 이용한 실시간 소스 레벨 디버깅 및 UART printf 로깅 활용

라이브러리: STM32 HAL 라이브러리 기반으로 주변장치 제어

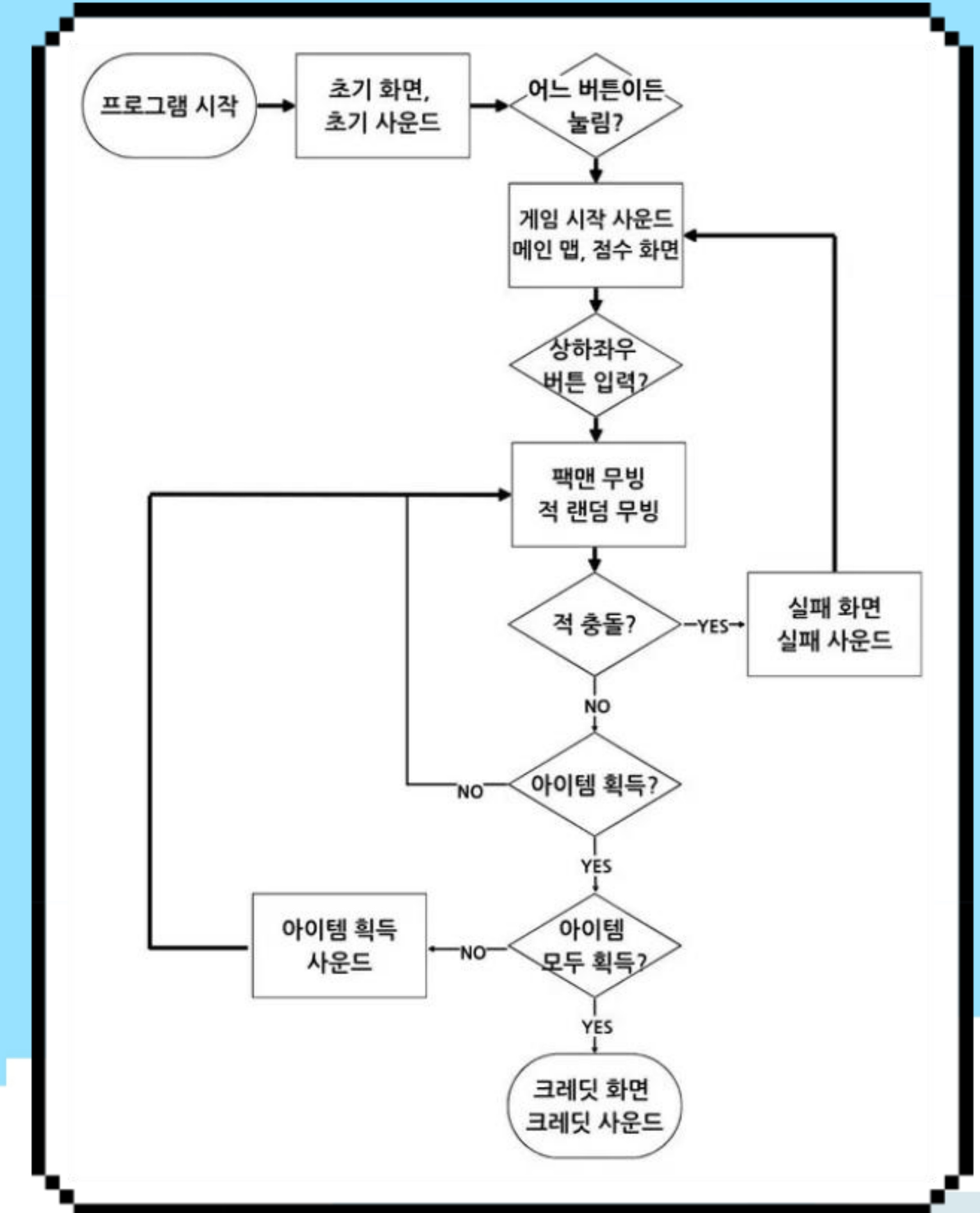
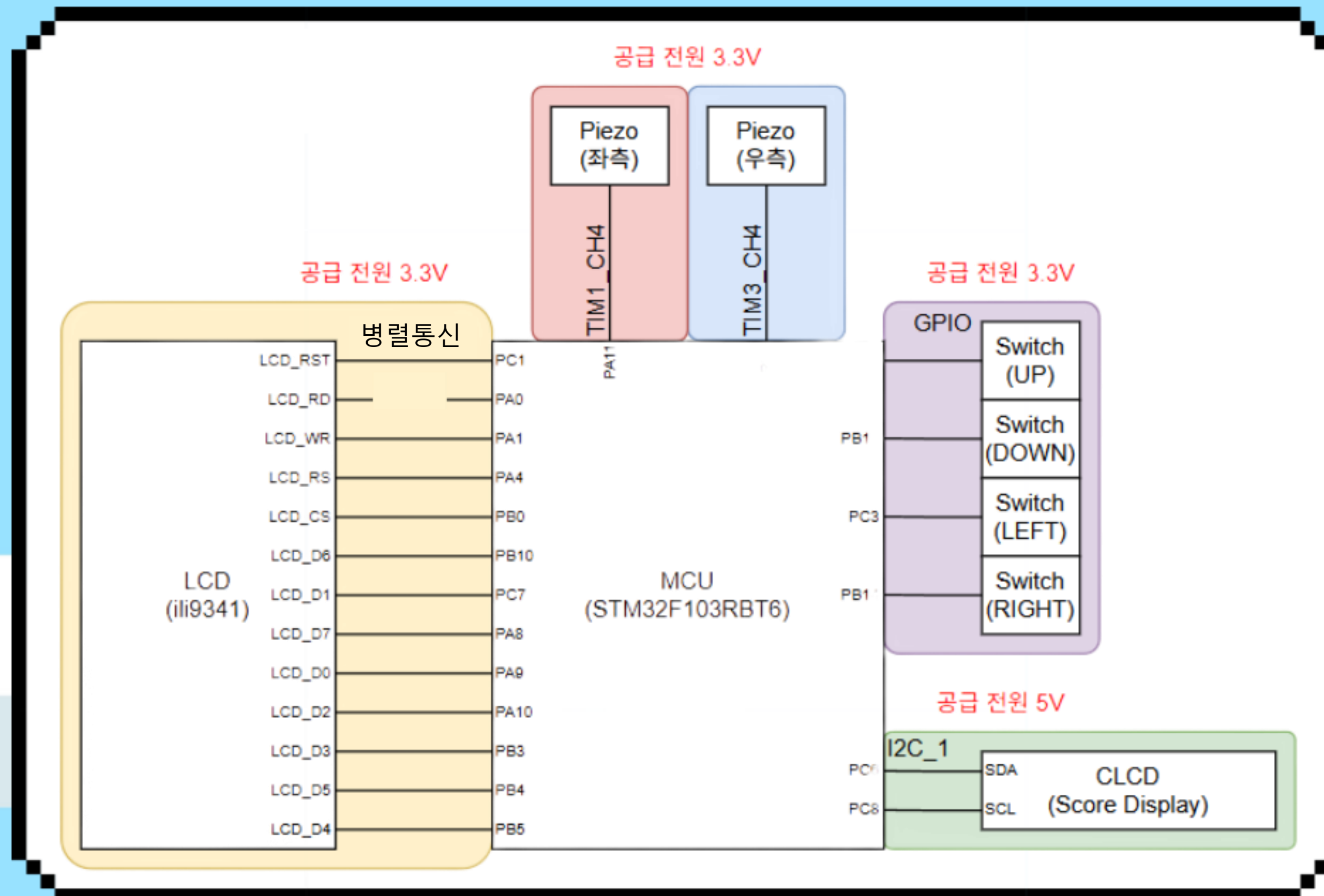
디스플레이 초기 구현 시 전체 화면 전송으로 인한 지연 문제를 부분 변경 영역만 갱신하는 방식과 전송 적용으로 대폭 속도 개선

▶ 시스템 블록 다이어그램

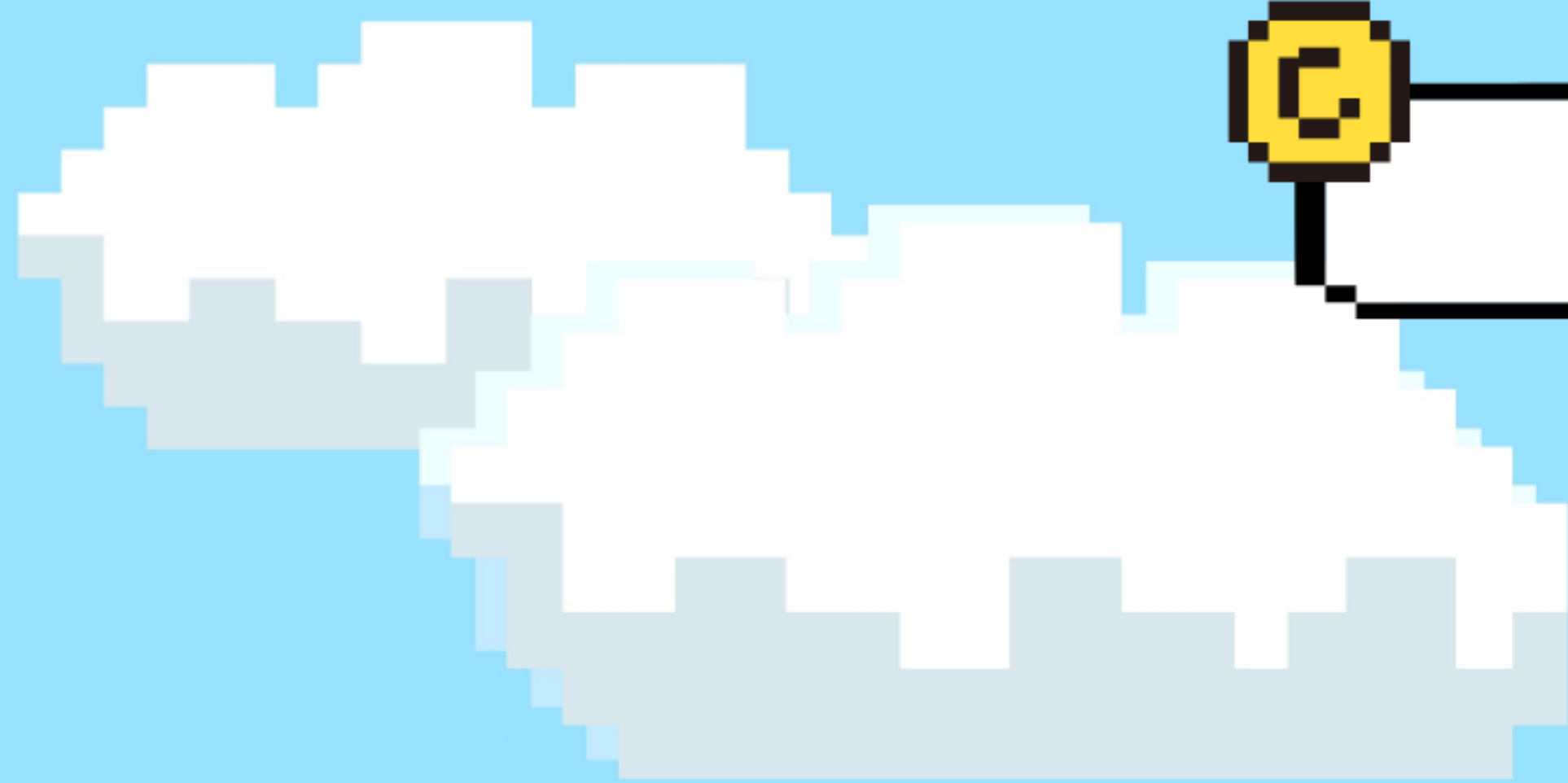
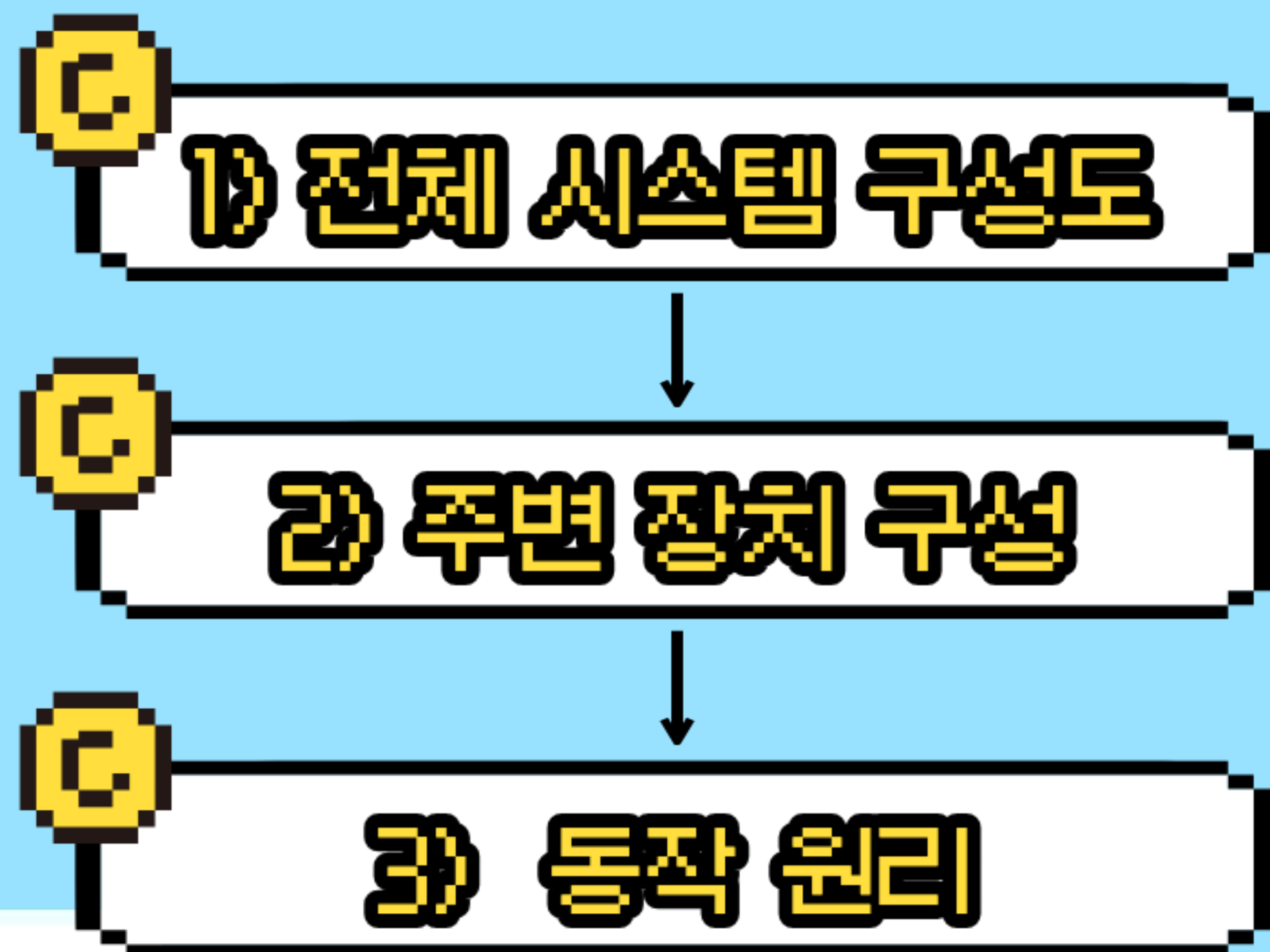


03 프로젝트 블록도

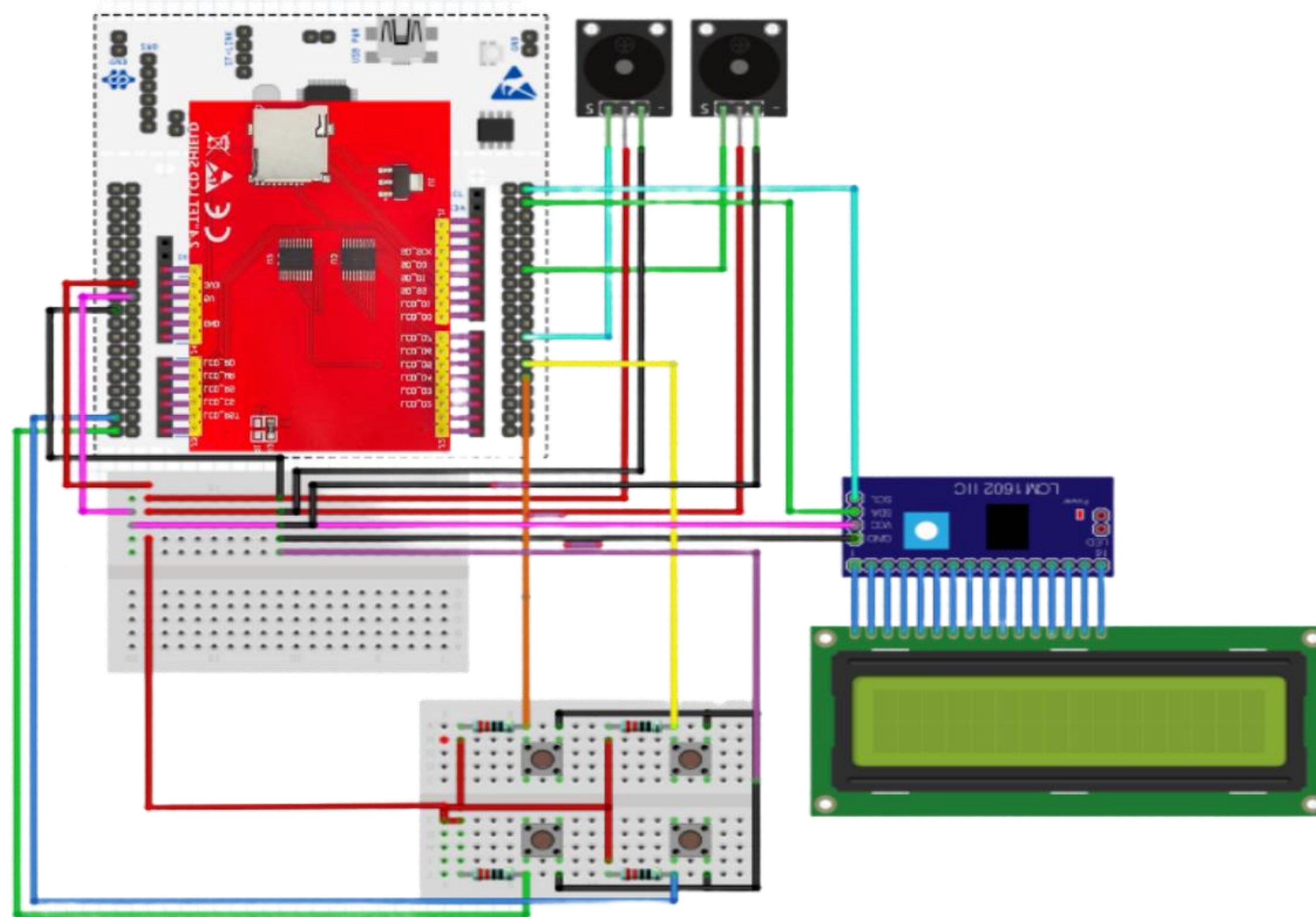
▶ 하드웨어



04 구성 및 동작



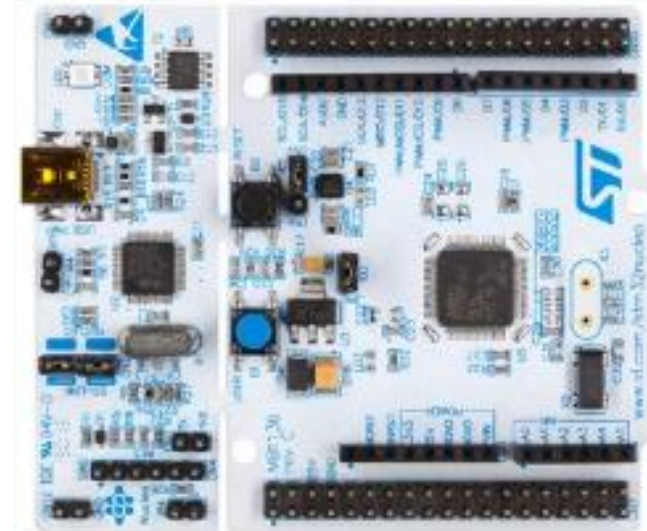
1) 전체 시스템 구성도



2) 주변장치 구성

하드웨어 ☒

▶ 동작 구현

이동: 버튼 (GPIO) ☒

10kΩ 저항 4EA

▶ 누름- 게임 시작 / 캐릭터 이동

▶ GPIO 사용 (풀업 적용)

#define BIN_PRESSED:HAL_GPIO_ReadPin 사용 버튼이
눌렸을 때, RESET이 되도록 **풀업 적용**

사운드: Piezo 부저 (타이머) ☒

▶ 사운드-게임 시작/아이템 섭취/게임 오버/클리어

▶ PWM 출력을 이용한 2성부 사운드 구현

▶ BUZZ_SetFreq_TIMx: 두 부저의 주파수를
독립적으로 설정 가능: **TIM1, TIM3** 타이머 사용 CH4

디스플레이: ILI9341 LCD ☒▶ 화면- 초기화면 출력 / 맵화면 출력 / 게임 오버 /
게임 클리어▶ 통신 방식: **8비트 병렬 인터페이스** (소프트웨어 처리)

▶ LCD_D0~LCD_D7까지 데이터 핀과 **제어핀**
(LCD_CS, LCD_RS 등) **매크로 정의**

▶ 화면 해상도: **240 x 320**로 정의

Score: LCD (I2C) ☒

▶ 점수 갱신

▶ 통신 방식: **I2C** (소프트웨어 처리)▶ **SDA/SCL** 제어 핀 매크로 정의

3) 동작 원리

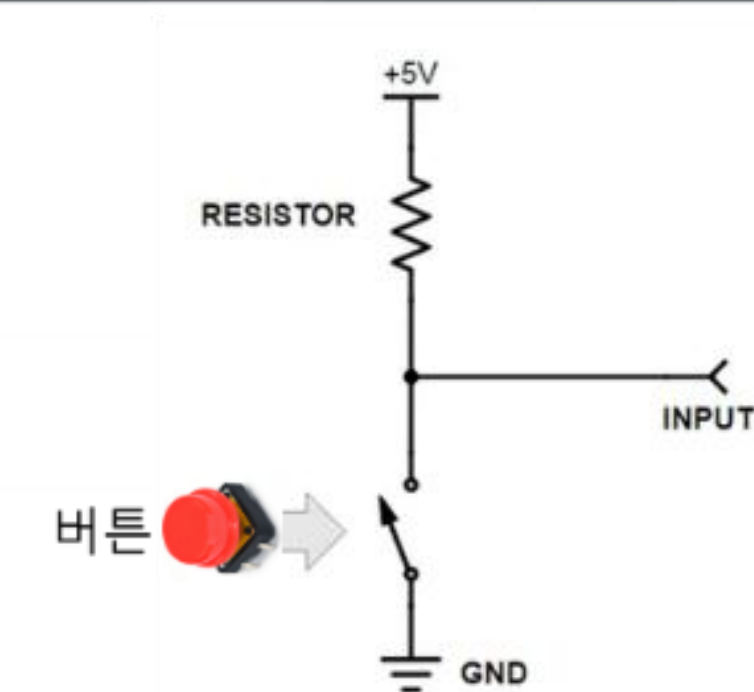
버튼 동작 (GPIO pin)

버튼 동작 (GPIO pin) 버튼 한쪽은 GND, 다른 한쪽은 STM32의 GPIO 입력 핀에 연결

- GPIO 핀 "버튼이 눌렸는지" 계속 감지 (센서 역할)
- Pull-Up 저항을 이용하여 플로팅 방지

버튼을 누르지 않으면 핀이 HIGH(1), 누르면 LOW(0)으로 신호가 안정적으로 구분

- 소프트웨어에서 HAL_GPIO_ReadPin() 함수로 상태(HIGH/LOW) 읽어 사용

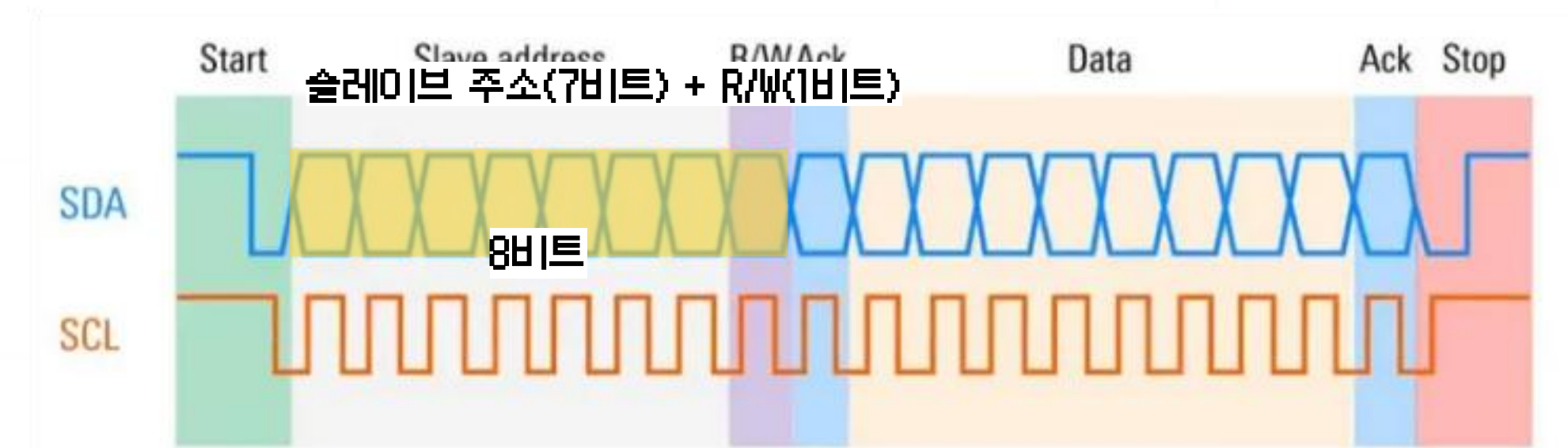


I2C (I2C)

두 개의 선 (SDA: 데이터, SCL: 클록)

- 모든 데이터 전송은 SCL(클록) 신호 기준으로 SDA에 올바른 비트가 전달될 때 이루어짐
- 슬레이브 주소(7비트) + R/W(1비트)로 시작 신호(8비트) 전달
- ACK(0): 신호 수신 성공/다음 단계 준비 신호
- 0: Write, 1: Read

이 방식으로 마스터가 슬레이브와 정확하게 통신을 주고받음



➡
NEXT

3) 동작 원리

LCD (병렬 통신)

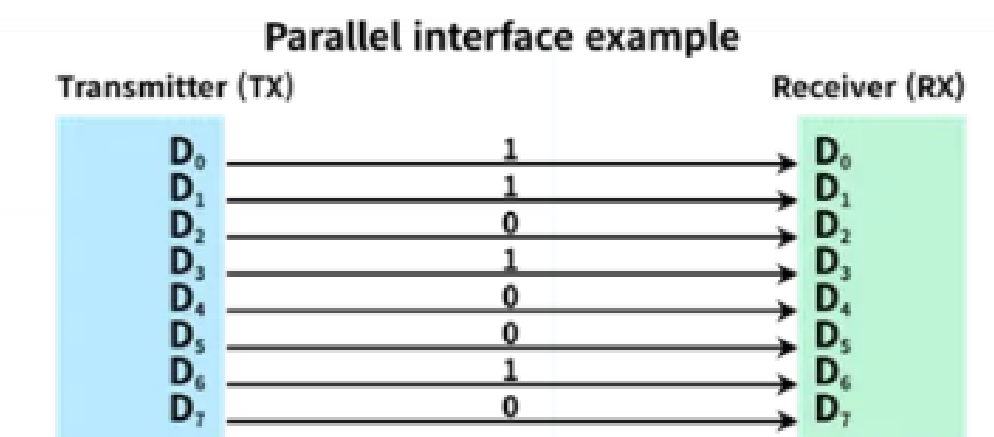
8080 병렬통신 인터페이스

⇒ 데이터를 동시에 전송하는 방식으로, 빠른 화면 갱신에 사용

- CS로 대상 LCD를 선택, RS로 데이터/명령을 지정, D0~D7에 전송할 값 지정
- WR 신호가 Low에서 High로 전환될 때 데이터가 LCD에 기록(데이터시트 참고 필요)

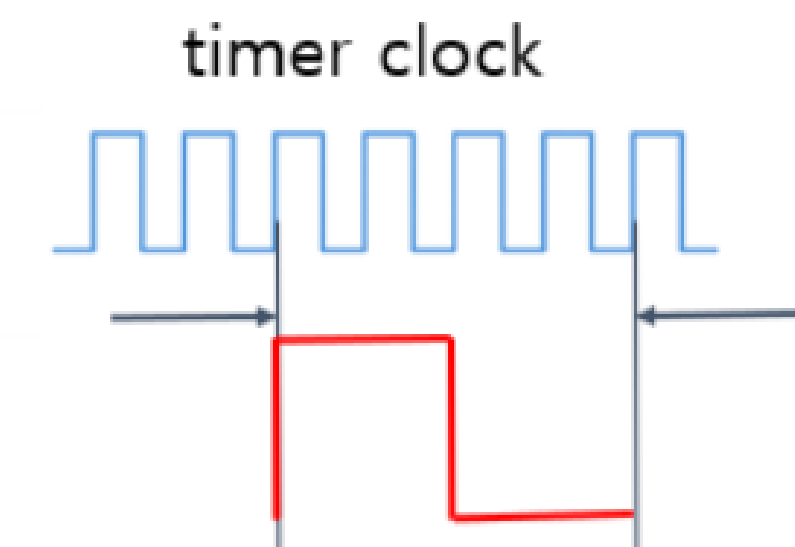
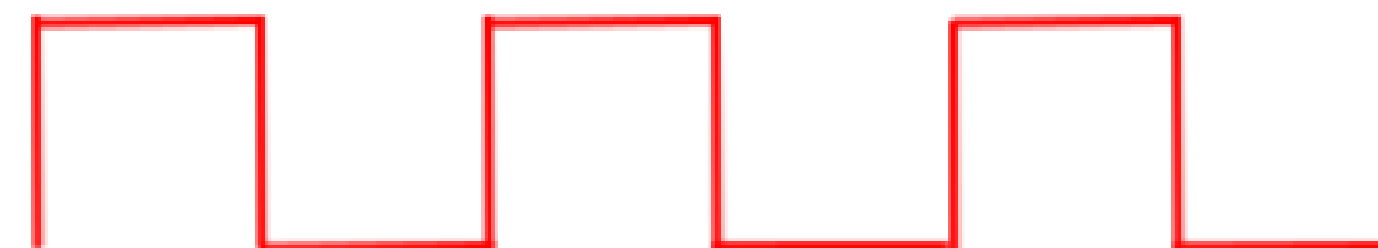
이 과정을 반복해 한 번에 많은 데이터를 빠르게 전달

- 신호제어 핀 : CS, RS, WR, RD, RST
- Write Cycle (MCU → LCD)
- 데이터 핀: D0~D7



Piezo 부저 (타이머 PWM)

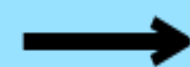
- 타이머를 이용해 원하는 음계의 주파수 펄스 PWM 신호 생성 PWM 신호는 출력 핀을 통해 피에조 부저에 전달
- 2성부 사운드 구현: 각각 두 개의 타이머 채널에 PWM 신호가 동시에 부저에 출력
 - 타이머 PWM 출력 핀 → PA11: TIM1_CH4, PB1: TIM3_CH4



05 최종



1) 최종 사진



2) 최종 동영상

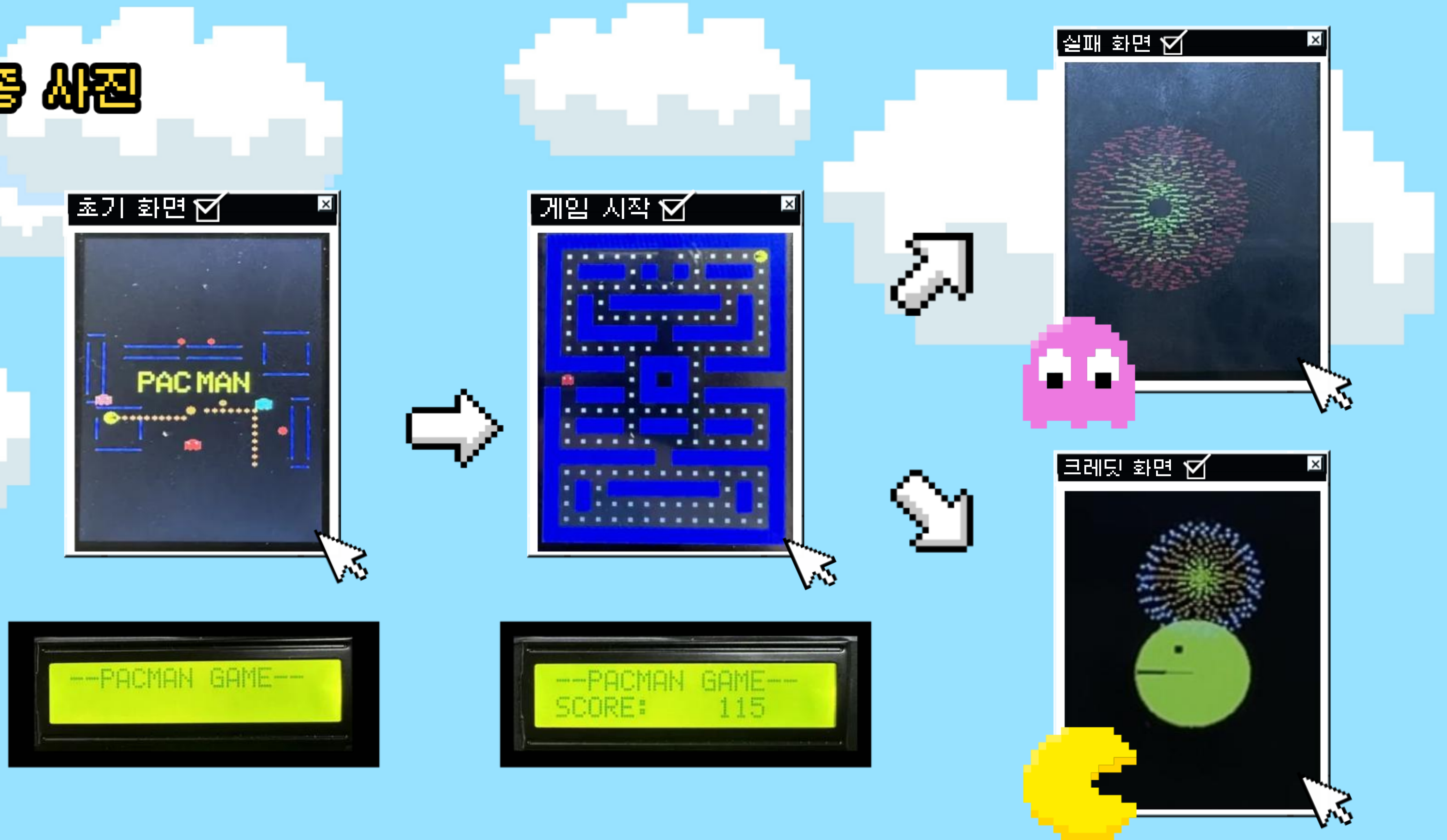
1) 최종 사진



▶
NEXT



1) 최종 사진



2) 최종 동영상

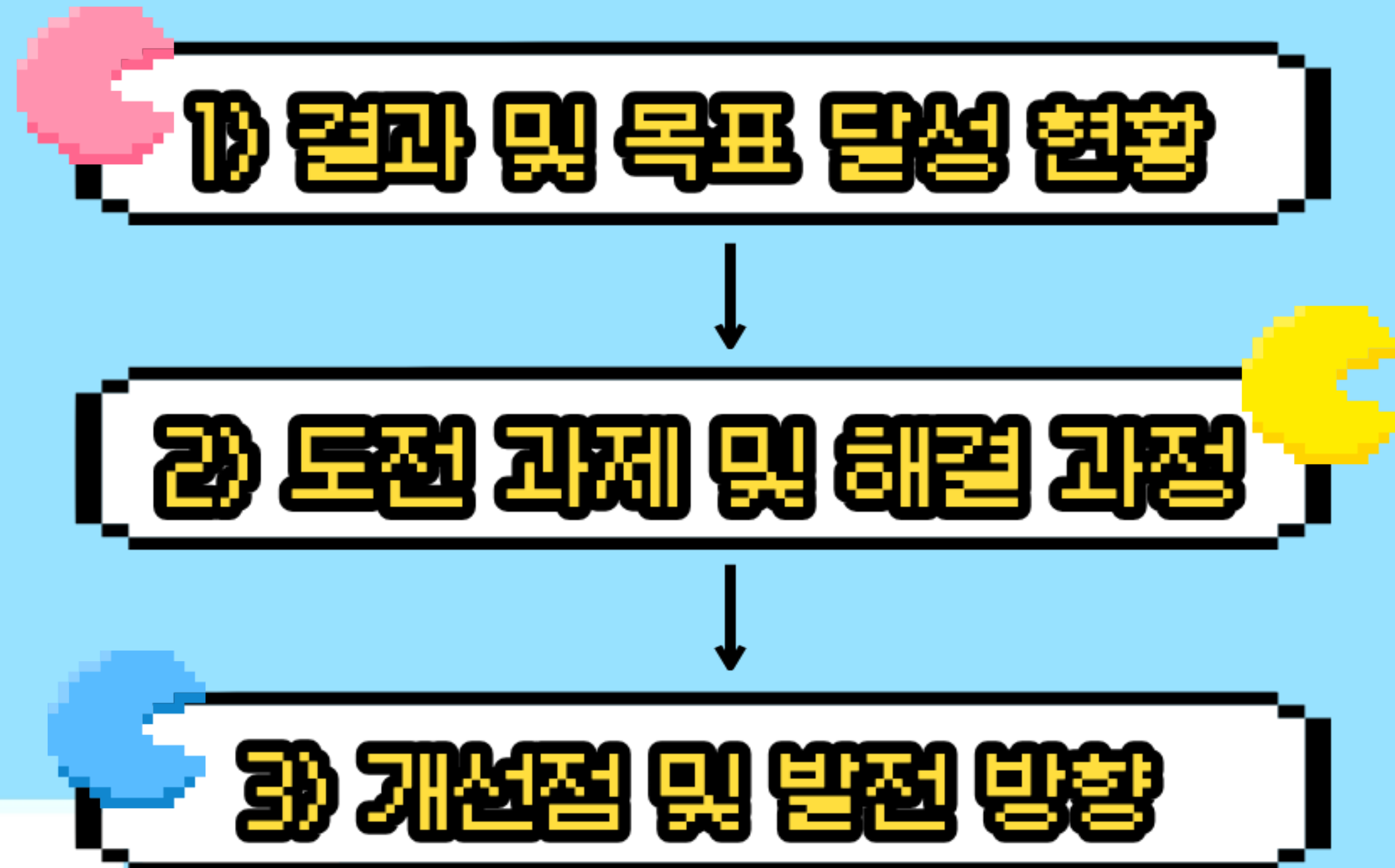
▶ 성공



▶ 고스트 접촉



06 결론 분석



1) 결과 및 목표 달성 현황

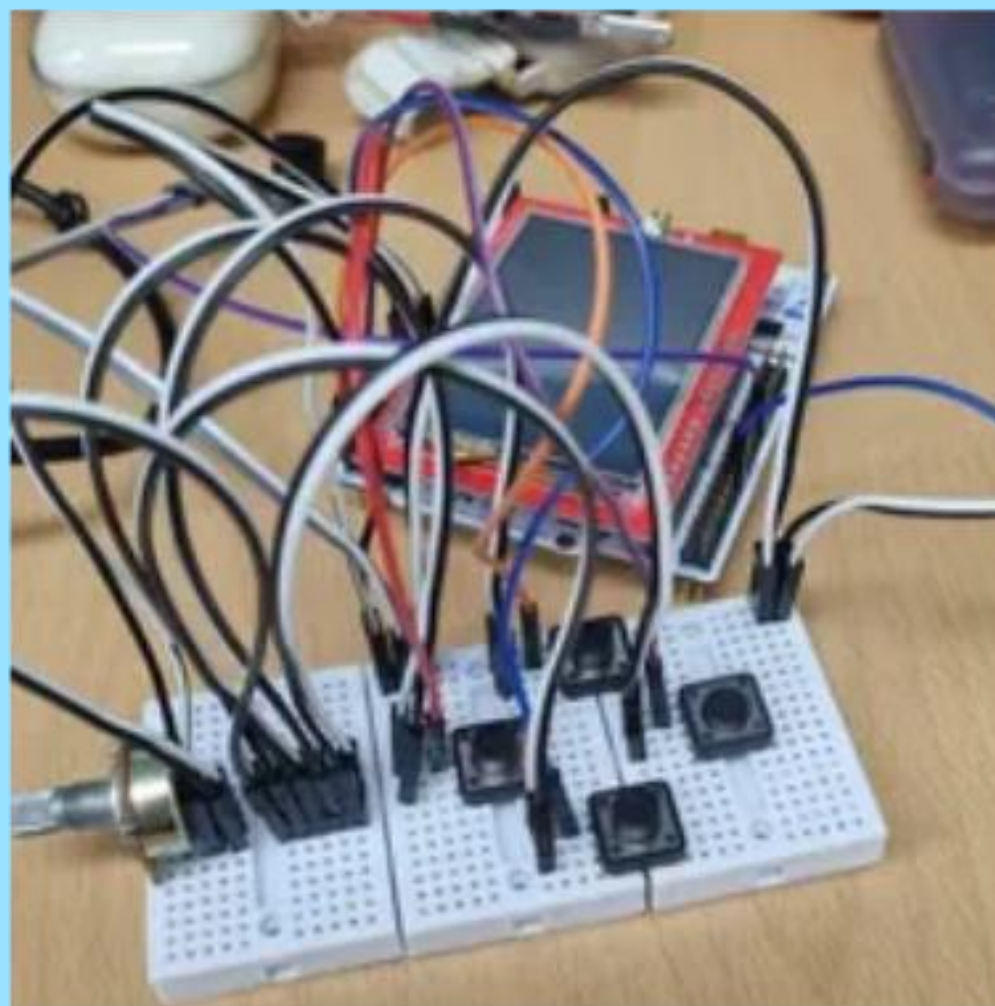
NUCLEO-F103RB 보드와 IL19341 LCD를 기반으로 팩맨 게임 구현

- ☒ 4개의 GPIO 스위치를 통해 팩맨 캐릭터 **이동 제어**
- ☒ 피에조 부저를 PWM 방식으로 제어하여 게임 배경음과 효과음 **사운드 출력**
- ☒ L19341 LCD와 병렬 통신으로 **실시간 게임 화면 출력**
- ☒ 시스템 상태 리셋 및 완료 조건에 따른 상태 전환 제어 구현
- ☐ **최고 점수 차순 기록 저장 기능 미달성**

=> 단일 맵 구성에 최고 점수 기능 의미가 없었고, 개발 기간 부족으로 구현이 어려웠음

2) 도전 과제 및 해결 과정

- 선 정리를 위해 납땜 진행 과정에서 **버튼의 하드웨어 문제** 발생
 - 원인 추적 결과, **스위치 소자 자체 불량**으로 최종 판단
 - 불량 스위치 제거 후 **새 부품으로 교체**, 동일 방식으로 재납땜
 - **정상 동작 확인**
- ⇒ 하드웨어 디버깅 경험 축적, 최종적으로 입력 시스템 안정적으로 구현
- 풀업저항 가변저항 ⇒ 리드 저항 변경
 - 입력 신호 안정성 강화.



3) 개선점 및 발전 방향

- SD카드: 저장/호출 구조 적용해 데이터 저장 효율 및 시스템 확장성 개선.
 - 다음 레벨에 고스트 추가해 난이도와 게임성 증가
 - 새로운 맵 구조 지원
 - 레벨 도입 후 점수 저장 기능 추가 계획.
- 코드상에서 스위치 채터링(Chattering) 일부 소프트웨어 처리하고 있으나, 단일 입력에도 다중 신호가 발생함. 디바운스 알고리즘 보완 개선 필요

PAC



PAC

THANK YOU!

