시스템 프로그래밍 Assignment #4

라즈베리 파이 간 GPIO 통신 시스템 설계 및 구현

Note

GPIO(General Purpose Input/Output)는 동적으로 방향(in/out)과 값(High/Low)을 제어할 수 있는 단순한 디지털 I/O 인터페이스이다. 주로 센서(예: 버튼)의 상태를 읽거나 액추에이터(예: LED)를 제어하는 데 활용된다.

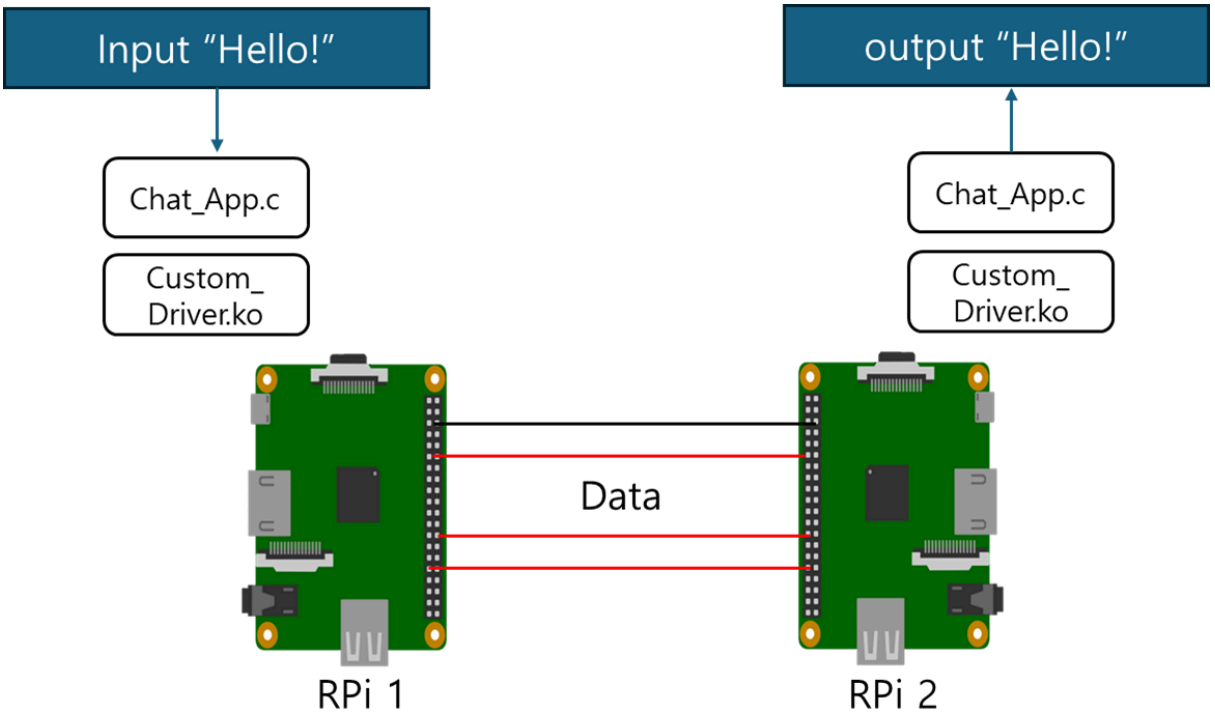
GPIO는 단순 제어 기능을 넘어서, 서로 다른 시스템 간 간단한 통신 수단으로도 활용될 수 있다. 이는 일종의 하드웨어 기반 IPC (Inter-Process Communication) 구조로 볼 수 있다. GPIO의 High/Low 신호를 특정 규칙에 따라 변조하고 인코딩하면, 디지털 데이터를 전송하는 통신 채널로 사용할 수 있기 때문이다.

이러한 방식은 I2C(Inter-Integrated Circuit), SPI(Serial Peripheral Interface), UART(와 같이 GPIO 핀을 기반으로 동작하는 정식 통신 프로토콜들의 저수준 동작 원리와 유사하다.

Goal

본 과제의 최종 목표는 Raspberry Pi 간 GPIO를 연결하고, 이 인터페이스만을 이용한 통신 시스템을 설계 및 구현하는 것이다. 특히 통신 프로토콜 및 SW를 설계할 때, 수업에서 학습한 내용을 최대한 적용해 보는 것이 주요 목표이다. 이를 위한 세부 목표는 다음과 같으며, 과제의 개요는 그림과 같다.

- 통신 기법 설계
- Device Driver, API 설계 및 구현
- 응용 프로그램 구현
- 동작 검증 및 성능 평가 수행



Requirements

1. 주제 및 타겟 응용 서비스 선정
    - 평소 관심사를 토대로 통신이 적용될 가상의 응용 서비스 선정 (GPIO와 관계없이 통신이 활용되는 모든 응용 선택 가능)
        - IoT 산불 감지 시스템, 에어드랍, 실시간 동영상 스트리밍, 차량 내 ECU 간 통신 등 자유롭게 선택
    - 선택한 응용의 요구사항 분석 수행
2. 통신 프로토콜 설계
    - 요구사항 분석을 토대로 통신 기법 설계 수행
    - 물리적 연결 구성 설계 (핀 연결), 동작 방식 설계 (동기/비동기, 데이터 표현, 버퍼링, ...), 신뢰성 제공 방법 설계 (오류 검증, 재전송, ...), ... 등 제한 없이 자유롭게 통신 설계
    - 수업에서 학습한 내용들 및 다양한 기존 지식들 활용 (TCP 등)
3. SW 설계 (디바이스 드라이버, API)
    - 디바이스 드라이버는 `file_operations` 구조체 기반으로 `open`, `close`, `read`, `write`, `ioctl` 등의 인터페이스를 제공하도록 구현하는 것을 권장

- API는 필수 설계 요소는 아니며, 필요할 경우 디바이스 드라이버의 system call을 조금 더 효율적으로 사용할 수 있도록 기능 제공 ex) glibc의 fopen, fread, fwrite, fclose 등
- GPIO들을 각각의 I/O 디바이스로 생각하여 역할에 따라서 적절한 OS의 자원 관리 및 최적화 기법들을 최대한 적용하여 설계

4. SW 구현
    - 디바이스 드라이버 구현 시 GPIO 제어 기능은 <linux/gpio/consumer.h>의 gpiod 함수들 사용
    - API는 응용 프로그램과 파일을 분리하여 따로 제작
    - 응용 프로그램은 동작 테스트 용으로 설계 평가 요소로는 반영하지 않음
        - 구현 시스템의 동작 및 성능을 잘 보여줄 수 있는 간단한 응용 프로그램을 작성
        - 선정이 어려운 경우 간단한 채팅 프로그램을 작성
        - 디바이스 드라이버가 다양한 모드를 제공할 경우 이를 모두 보여줄 수 있도록 응용 프로그램 작성
    - 응용 프로그램은 반드시 제작된 Device Driver를 통해서 GPIO를 제어해야 하며, 다른 GPIO 제어 라이브러리를 사용 시 평가 불이익 크게 발생
    - 모든 SW 구현은 AI를 활용 가능하지만, 설계와 다를 경우 큰 불이익 발생 (AI를 활용하여 생성한 코드의 경우, 해당 코드의 동작 원리와 구조에 대한 충분한 이해가 반드시 필요하며, 설계서와 구현 내용이 일치하지 않을 경우 대폭 감점)

Submission

제출 기한: 6/28 (토) 11:59 PM

제출 파일

- device_driver.zip
    - 디바이스 드라이버 소스코드
    - Makefile (반드시 포함)
- app.zip
    - 응용 프로그램 소스코드 (송/수신 코드가 다를 경우 모두 제출)
    - API 소스코드
- Report.pdf

    - 반드시 PDF 형식으로 파일을 변환하여 제출

Report

보고서는 다음 제시된 목차 및 내용이 반드시 포함되도록 하여 추가적으로 필요한 내용들을 자유롭게 포함하여 작성

- 주제 선정 및 요구사항 분석
- 통신 기법 설계
    - 물리 연결 설계 (pin 사용 및 역할), 통신 방식 설계
    - 각종 추가적 설계 내용
- SW 설계
    - SW 구조 및 동작 설계
    - 자원 관리 및 최적화 설계 방식 상세 설명
    - 데이터 흐름 분석 (송신 Pi에서 `Hello` 라는 문자열을 입력하면, 어떤 구조를 통해 신호가 출력되고, 수신 Pi에서 이를 어떻게 파악하고 문자열로 복원하는지를 단계별로 설명할 것)
- SW 구현
    - Device Driver, API, 응용 프로그램의 소스코드 분석
    - Device Driver, API 기능 및 사용 방법
- 동작 검증 및 성능 분석
    - 응용 프로그램 동작에 대해서 스크린샷을 이용하여 검증 제공 (문자열 전송)
    - 통신 속도, 신뢰성, 자원 효율 등 통신 성능에 대한 분석 (이론 및 실측)
    - 자신의 설계와 팀원의 설계 또는 기존 통신 프로토콜(I2C, SPI, UART) 중 하나를 비교하여, 설계 철학, 타이밍, 처리 구조, 신뢰성 측면에서 비교 분석할 것

참고사항

- 본 과제는 개인 과제의 형태로 모든 개인이 각자의 제출물을 작성하여 제출해야 함

- 팀원과 다른 주제를 선택해야 하며, 팀원 간 설계 및 구현 과정에서 토론 및 협업 하는 것을 권장
- 단일 라즈베리 파이로도 과제 충분히 가능하기 때문에 다양한 사정으로 팀원 간 소통이 어려운 경우 혼자 진행해도 무방
- 통신 기법은 GND 핀을 제외하여 최대 5개의 GPIO Pin 이용 가능하도록 설계
- 타이밍 제어는 밀리초 단위로 여유롭게 수행해도 모두 감안할 예정
    - 기존 기술들처럼 마이크로초 수준의 제어의 어려움 감안
- **GPIO out 핀 끼리 연결하여 통신하지 않도록 주의!!!**
    - ex) GPIO 17 pin (out) — GPIO 20 pin (out) 상태로 High 신호 발생 시 고장 발생
    - 반드시 수신 핀은 direction을 in 상태로 설정한 후 전선 연결 수행
- 라즈베리 파이 원결 연결 문제, 전선 문제, 파이 고장 등 도움이 필요한 경우 조교님께 연락하여 신속하게 조치 받으시길 바랍니다.