SystemProgramming Project

-2022113203 이현우-

1. 주제 소개

- 주제: 라즈베리파이를 이용한 날씨 및 미세먼지 관리 시스템

라즈베리 파이는 다양한 모듈들과 상호작용이 가능한 싱글보드 컴퓨터입니다. 그래서 리눅스 운영체제를 설치하여 여러가지 모듈들을 관리하고 또한 통신 처리 및 프로세스 관리가 가능합니다. 이러한 점을 이용해 라즈베리파이를 이용하여 날씨 및 미세먼지의 데이터들을 실시간으로 받아와 그에 따른 상황 대처를 확인해볼수 있는 프로그램을 제작해보았습니다.

2. 기능 및 설계

- 기능

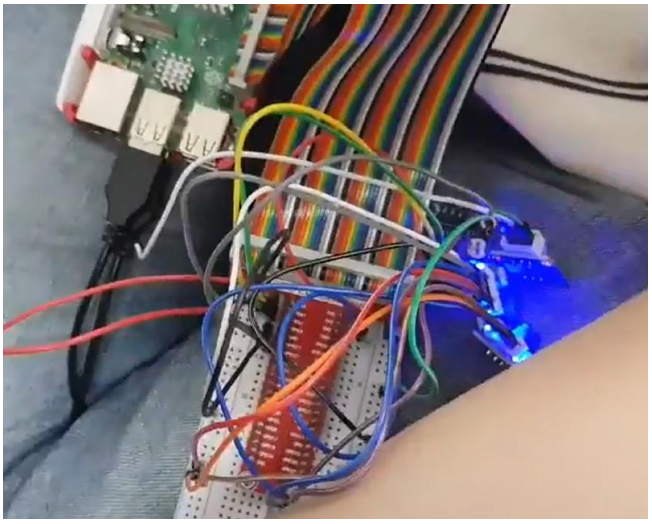
  가스 센서, 미세먼지 센서, 습도센서를 이용해 데이터들을 실시간으로 수집합니다. 가스 센서를 통해 실내 가스 누출을 감지하고 가스가 감지된다면 실내 공기 순환을 멈추고 실외공기 순환을 위해 팬이 돌아가게 됩니다. 미세먼지 센서는 실내외 미세먼지 농도를 측정합니다. 농도에 따라 다른색의 LED가 켜지며, 미세먼지 shdeeh가 높아지면 LED의 색이 청색에서 적색으로 바뀝니다. 또한 미세먼지 농도가 실내보다 실외에서 높을 경우 실외 공기 순환을, 그렇지 않을 경우 실내 공기 순환을 위한 팬이 돌아갑니다. 마지막으로 습도 센서는 실내 습도를 측정합니다. 습도가 60% 이상일 경우 실외공기 순환을 그렇지 않을 경우 실내 공기 순환을 진행합니다. 이처럼 센서들의 데이터들을 바탕으로 액추에이터(모터와 LED)를 제어 하여 실내외 환경을 적절하게 유지할수 있습니다.

- 설계

  - client.c
    - 사용 라이브러리: stdlib, string, unistd, arpa/inet, sys/types, sys/socket
    - 주요기능
      - Socket(): 서버에 연결하기 위한 소켓을 생성
      - Connect(): 주어진 IP주소와 포트 번호를 사용하여 서버에 연결
      - Send(): 서버에 “Hello, Server!”라는 메시지 전송
      - Recv(): 서버로부터 메시지를 수신하여 화면에 출력

- Inner_server.c
  - 사용라이브러리: stdlib, string, unistd, arpa/inet, sys/types, sys/socket
  - 주요기능:
    - Socket(): 클라이언트의 연결을 기다리기 위한 소켓 생성
    - Bind(), listen(): 주어진 포트 번호를 사용하여 서버를 설정하고, 클라이언트의 연결을 기다림
    - Accept(): 클라이언트의 연결을 수락
    - Send(): 클라이언트에 메제시를 전송
    - Recv(): 클라이언트로부터 메시지를 수신하여 화면에 출력
- Outer_server.c
  - 사용 라이브러리: string, stdlib, unistd, arpa/inet, sys/types, sys/socket, pthread, wiringPi, softPwm
  - 주요기능
    - Pthread_create(): 미세먼지센서와 가스센서, 온습도 센서를 각각 제어하는 3개의 스레드를 생성
    - Socket(), bind, listen(): 주어진 포트 번호를 사용하여 서버를 설정하고, 클라이언트의 연결을 기다림
    - Accept(): 클라이언트의 연결을 수락
    - Send(): 센서에서 읽은 데이터를 클라이언트에게 전송,
    - wiringPiSetup(), softPwmCreate(), digitalRead(), softPwmWrite(): GPIO 핀을 사용하여 센서를 제어, 센서의 데이터는 주기적으로 업데이트
- 사용한 모듈
  - 온습도 센서: DHT 11 (27번 핀)
  - 미세먼지 센서: GP2Y1010AU0F (1번 핀)
  - 가스 센서: MQ2 (26번 핀)

3. 수행결과



(Figure 1) hardware actuation

라즈베리파이와 연동된 3가지 모듈이 실시간으로 데이터들을 수집하여 outer_server로 데이터들을 전송합니다. 또한 데이터에 따른 여러가지 actuation(motor, led)들을 보여줍니다. 가스 센서는 실내 가스 누출을 감지하여 0 ~ 20 ppm 이면 파란불, 20 ~ 30 ppm 이면 주황불, 30 ppm 이상이면 빨간불을 나타냅니다 또한 20ppm이상 이면 팬이 작동합니다.

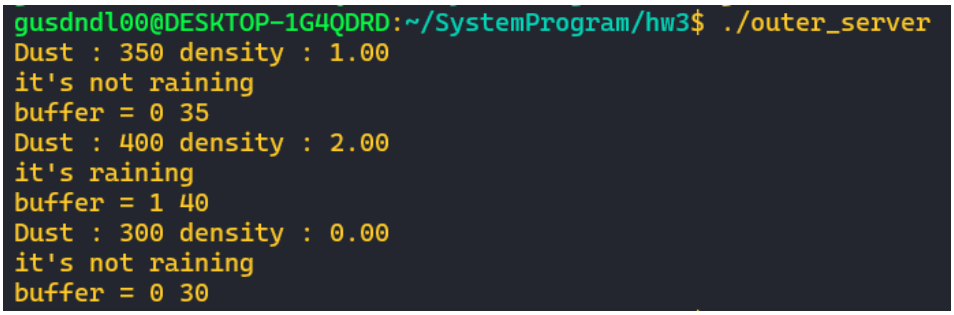
미세먼지 센서는 실내외 미세먼지 농도를 측정합니다. 0 ~ 100 CAI이면 파란불, 100 ~ 250 CAI 이면 주황불, 250 CAI 이상이면 빨간불이 켜집니다. 주황불 범위인 100 CAI 이상이면 팬이 동작합니다

습도 센서는 실내 습도를 측정합니다. 습도가 60% 이하인 경우만 팬이 동작합니다.

```
gusdndl00@DESKTOP-1G4QDRD:~/SystemProgram/hw3$ ./client

outside server msg: 1 200
out_rain: 1, out_dust : 200

gas detected -> outside circulation
outer_dust > inner_dust
humidity high -> outside circulation
==========================

outside server msg: 0 150
out_rain: 0, out_dust : 150

humidity low -> inside circulation
outer_dust < inner_dust
humidity high -> inside circulation
==========================
```

(Figure 2) client 실행 화면

(Figure 3) inner_server 실행 화면



(Figure 4) outer_server 실행 화면

위 각각의 사진은 client, inner_server, outer_server에 대한 실행 화면 입니다. 여러가지 경우의 수를 데이트하기 위해 몇몇 데이터는 임의로 조작된 데이터임을 참고해주시면 감사하겠습니다.

먼저 client에서는 outer_server, inner_server에서 전송해준 라즈베리파이 데이터들을 분석하는 역할을 해줍니다. 데이터들을 받아와 실외공기 순환을 시작할지 아니면 실내 공기 순환을 할지 판단합니다. 추가적으로 미세먼지는 외부 미세먼지와 내부 미세먼지를 둘다 확인하여 실내 공기순환 또는 실외 공기순환을 선택합니다.

Inner_server 에서는 라즈베리 파이와 외부 클라이언트 통신을 수행하여 라즈베리파이와 외부 클라이언트 사이에 데이터를 주고 받을수 있도록 합니다. 이로 인해 다른 디바이스와 상호 작용할수 있도록 도움을 줍니다. 여러 데이터중 실내에 해당하는 데이터들을 수집하는 역할을 해줍니다

Outer_server 에서는 inner_server와 마찬가지로 라이즈베리 파이에서 발생하는 여러 데이터들을 수집하는 역할을 해줍니다. 또한 클라이언트와 통신하여 라즈베리파이와 데이터를 주고 받을수 있도록 해줍니다. 여러 데이터중 실외에 해당하는 데이터들을 수집하는 역할을 해줍니다.

4. 이슈 사항 및 해결

프로젝트를 진행하다 몇가지 문제에 직면하게 되었는데 가장 큰 문제점은 하드웨어와의 통신과 멀트스레드 도중 스레드간의 동기화, 소켓 통신 이렇게 크게 3가지 부분에 대해 문제점이 발생하였습니다.

1. 하드웨어와의 통신

   GPIO핀을 제어하는 부분에서 오류가 발생하였습니다. 하드웨어와 같이 병행하는 프로젝트인 만큼 하드웨어에 관해 오류도 잦았는데 핀의 방향을 잘못 설정하거나 코드에서 핀번호를 잘못 설정하는 경우도 있었습니다. 추가적으로 센서와의 통신 도중 연결이 끊겼다 되었다 하는 현상이 발생하였는데, 오류 로깅을 해보니 센서에게 주어지는 전압의 크기가 일정하지 않아 접속이 불안정 한거였습니다. 그래서 전압을 계산하여 추가적으로 전압을 안정화 해주는 작업을 진행하였습니다.

2. 멀티 스레딩

   3개의 스레드를 동시에 실행하는 경우 스레드 간의 동기화에서 문제가 발생하였습니다. 여러 스레드가 동시에 같은 자원을 접근 하려고 해서 생기는 문제인데 힙 영역에서 모든 스레드 접근이 가능해 이 부분에서 동기화 문제가 발생하였습니다. Pthread_mutex_lock, pthread_mutex_unlock()과 같이 뮤텍스를 이용해 data race condition을 방지할수 있었습니다. 추가적으로 상호 배제의 원리를 따라 한번에 하나의 스레드만이 공유 데이터에 접근할수 있었습니다. 특히나 멀티스레딩을 이용해 프로세스 끼리 통신을 진행해야 한다는 점이 가장 까다로운 구현이였던 것 같습니다. 독립된 메모리 공간을 유지하는 도중에 프로세서 사이에 메시지를 주고 받아야하는 구현이 어려웠지만 pthread 공부를 통해 해결할수 있었습니다.

5. 매뉴얼

컴파일 방법

$ make install

$ make clean

$ make

실행방법

$ ./inside_server <port>
$ ./outside_server <port>
$ ./client <inside_server ip> <port> <outside_server ip> <port>
주의 사항 프로그램 실행전 raspberry pi가 연결 되어 있어야합니다