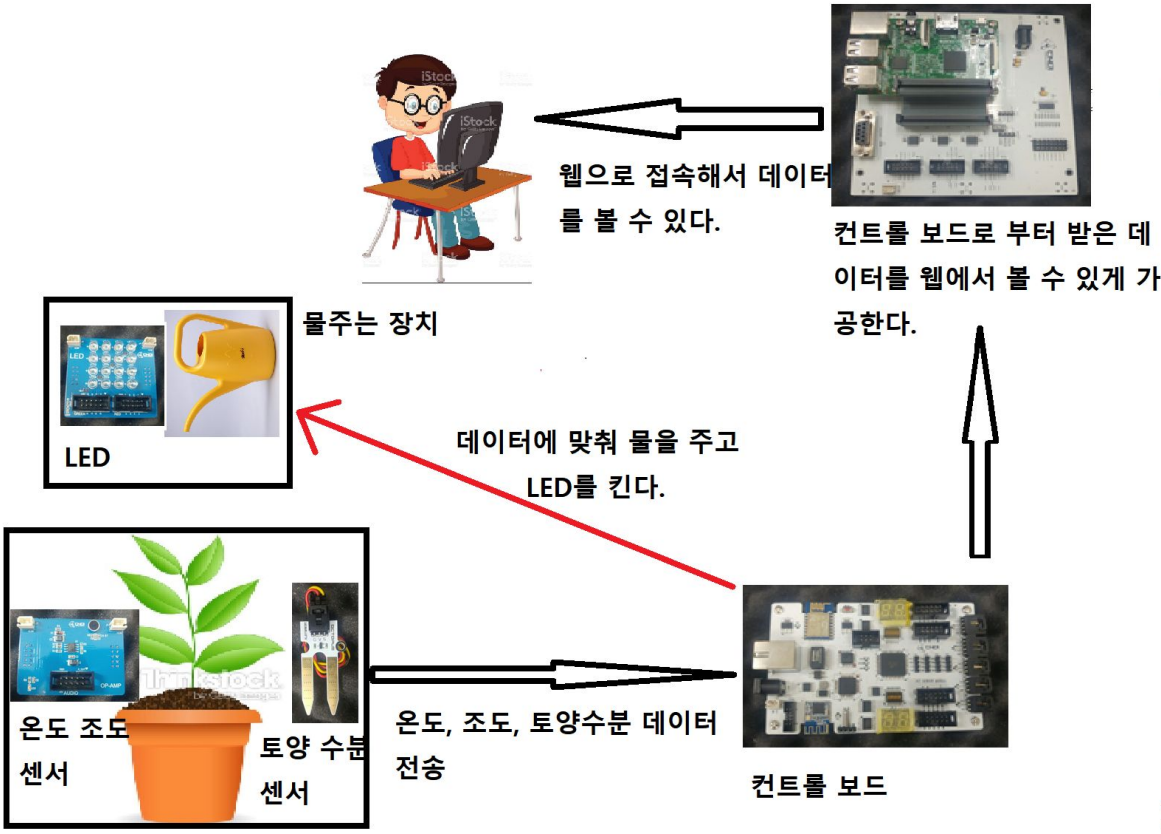


2조_입문설계_중간 보고서	
프로젝트명	식물 재배 환경 알림이
팀원	권민석, 정동호, 박상원, 이석원, 강병운
프로젝트 구상 초안	
목적	센서모듈을 통하여 획득한 재배환경 데이터를 wifi를 통해 라즈베리파이 서버에 저장하고 인터넷으로 라즈베리파이 서버에 접속하여 데이터를 확인할 수 있는 장치입니다.
감시할 데이터	온도, 토양 수분, 조도 센서를 통해 데이터를 취득합니다.
환경 조절	atmega 컨트롤 보드에서 조도 조절과 토양의 수분 조절을 해줍니다.
프로젝트 소스코드	<a href="https://github.com/gwnuysw/RasPiPod">https://github.com/gwnuysw/RasPiPod</a> 의 workingdir파일이 현재 실습중인 파일입니다.
구상도	
 <p>웹으로 접속해서 데이터를 볼 수 있다.</p> <p>컨트롤 보드로 부터 받은 데이터를 웹에서 볼 수 있게 제공한다.</p> <p>물주는 장치</p> <p>데이터에 맞춰 물을 주고 LED를 켜다.</p> <p>LED</p> <p>온도 조도 토양 수분 센서</p> <p>온도, 조도, 토양수분 데이터 전송</p> <p>컨트롤 보드</p>	
재배할 식물은 방울 토마토	

**1. 잘 자라는 환경조건**

- 싹트는 온도 : 28°C
- 잘 자라는 온도 : 25~27°C, 낮 25~30°C, 밤 18~20°C
- 지온(땅속 온도) 20± 2°C
- 낮 30°C, 밤 20°C 이상이나 13°C 이하에서는 낙과, 열과 및 기형과 발생
- 햇빛의 세기 : 강한 광선을 좋아하는 채소로 햇빛을 충분히 쬌어주는 것이 좋다.
- 토양조건 : 과습에 약함. 양토 또는 식양토가 좋다.

**역할 분담**

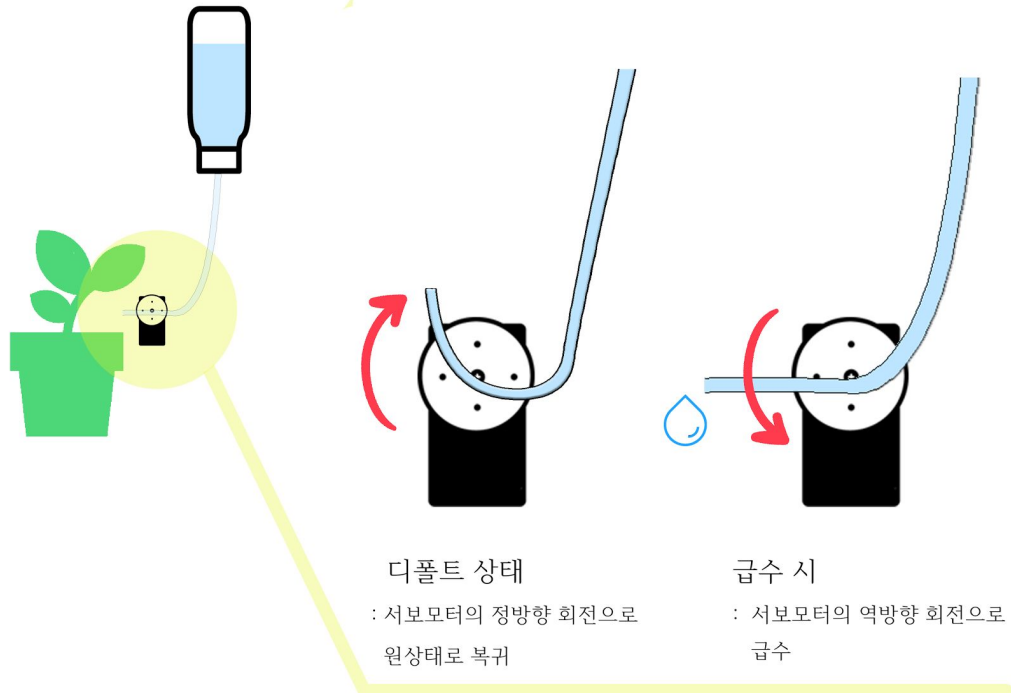
물주는 장치를 만들고 토양 수분 정도에 맞춰 작동 시키는 역할	컨트롤 보드와 라즈베리파이 서버의 통신, 라즈베리파이 서버와 외부 인터넷망 사이의 통신을 담당	문서 종합, 작성
1팀 : 권민석, 정동호	2팀 : 박상원, 이석원	3팀 : 강병운

**실습 일자**

일시	2017년10월27일 09:00~12:00	장소	W6 301호	인원	권민석, 박상원, 이석원, 정동호, 강병운
금일 목표	1팀 권민석, 정동호	급수 환경 구상			
	2팀 이석원, 박상원	교재 p328 무선랜을 통한 TCP 서버에 데이터 전송 실험			
	3팀 강병운	보고서 작성			

**1팀 실습 내용 : 물주는 장치 고안**

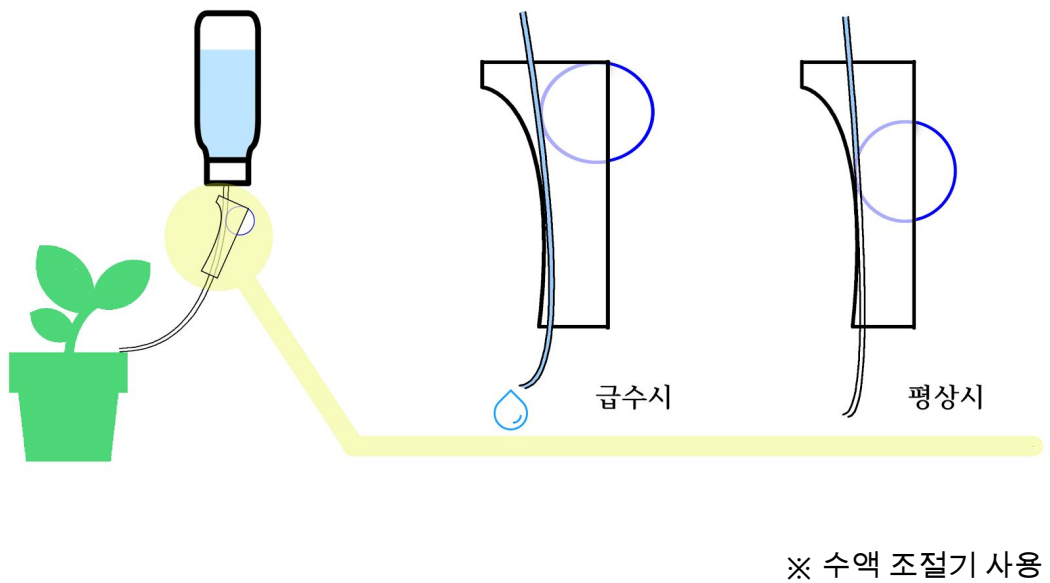
구상1



Q1 : 급수를 하지 않는 평상시에 호스가 안정적으로 고정되는가?

Q2 : 모터를 얼마만큼 회전시킬것인가?


구상 2



Q1 : 레버를 조절하기 위해 모터를 어떻게 활용할 것인가

Q2 : 모터가 레버를 돌릴만한 힘이 있는가

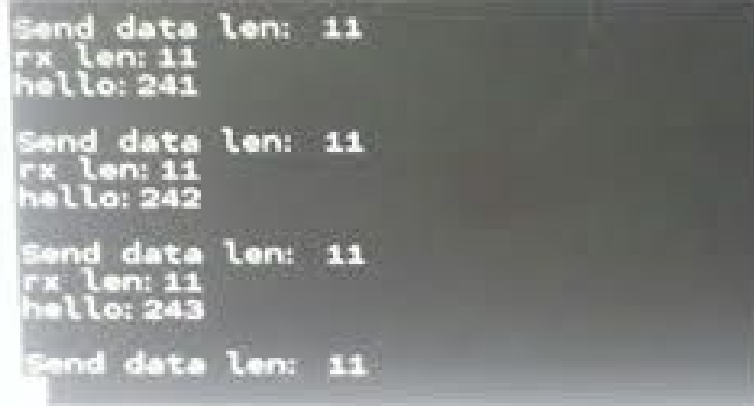
기타 고려 사항	수중 펌프로 쉽게 장치를 구성할 수 있지만 구비되어 있는 센서와 장치들만으로 구성하는 방향을 택함. // 추후 추가
<b>실습 도중 문제 해결 방법 또는 앞으로 개선점</b>	
<p>다음 주차 수업에서는 재료를 구비하여 모의실험 후 최선의 안을 선택한다.</p> <p>모터를 제어하는 소스를 수정하고 원하는 방향, 회전 정도를 조절한다.</p>	

<b>2팀 실습 내용 : 무선랜을 통한 TCP 서버에 데이터 전송 실험</b>	
사용한 장치	예제 프로그램, echo_server 응용 프로그램, AP(공유기), "USB to serial", USB 확장 케이블, 부품 및 데이터시트, DC 12V/1A adaptor, 라즈베리 파이
장치를 연결한 모습	
라즈베리 파이와 컨트롤 보드간 wifi 통신 확인	

컨트롤 보드에서  
보낸 데이터를 찍은  
사진입니다  
1초마다 데이터를  
전송하고 받습니다.  
TX가 보낸  
데이터이고 RX가  
받은 데이터인데,  
이상한 점이  
마우스로 드래그  
해서 블록을 채워야  
RX가 보입니다.



라즈베리 파이에서  
받은 데이터와 보낸  
데이터를 찍은 사진  
입니다.  
hello:%d 문자열이  
컨트롤 보드로부터  
받은 데이터입니다.



### 실습 도중 문제 해결 또는 앞으로 개선점

실습 장비 중에  
USB to serial  
케이블이란 것이  
있는데 serial쪽 선이  
어떻게 전압이고  
어떻게 gnd고  
어떻게 데이터 인지  
구분이 안되서  
중간에 헤맨 시간이  
많았지만 인터넷에  
자료를 찾아서 계속  
진행 할 수  
있었습니다.



디버깅 메시지가  
계속 AP connection  
fail이 떠서 시간을  
많이 허비 했습니다.  
차근차근 책과

```
17
18 #define SERVER_IP_STR "192.168.1.33" //라즈베리파이 커고 수정해야할 부분
19 #define SERVER_PORT 50001
20
21 static void eventCallback(int eventType, uint8_t* rxBuff, int rxSize)
22 {
```

소스를 읽어보다가  
18행과 63행이  
의심되서 바꿔보니  
접속성공 했습니다.  
18행은  
라즈베리파이  
서버의 ip주소를  
입력하고 63행은  
이용하는 wifi의 id와  
pw를 입력했습니다.

```

60
61 // connect AP
62 debugprint("\r\n");
63 if ( !wifiConnectAP("301", "gwnucomse"))
64 {
65     debugprint("AP connected.\r\n");
66 }
67 else
68 {
69     debugprint("AP connection fail.\r\n");
70 }

```

## 서버 프로그램 (echo\_server.c) 소스코드 분석

```

126 while(1)
127 {
128     readBufSize = read(clnt_sock, rcv_buf, RX_DATA_MAX); //소켓으로부터 문자를 읽어서 rcv_buf에 저장 하는데 그 크기는 최대 크기는 2048BYTE다
129
130     if(readBufSize > 0)
131     {
132         printf("rx len:%d\n", readBufSize); //read()함수는 정상적으로 읽었다면 읽은 BYTE수를 반환 한다.
133
134         rcv_buf[readBufSize] = 0; //맨 뒷자리를 0으로 셋팅
135         printf("%s\n", rcv_buf); //읽어온 대로 출력
136
137         // echo TX
138         writeBufSize = write(clnt_sock, rcv_buf, readBufSize); //받은 데이터를 clnt_sock으로 되돌려 줍니다.
139         printf("Send data len: %d\n", writeBufSize); //되돌려준 데이터의 길이를 알려줍니다.
140         if(writeBufSize < 0)
141         {
142             printf("echo write() error\n");
143             break;
144         }
145     }

```

코드가 길고 복잡하지만 통신관련 코드는 안봐도 됩니다. 캡처한 화면은 실제 데이터를 받아서 화면에 띄우고 다시 보내는 부분입니다. 실질적으로 저 부분만 수정 하면 되는데 데이터라고 해도 단순히 문자열이고 문자열을 주고 받는 것도 잘보면 파일입출력이랑 크게 다르지 않습니다. 어떻게 잘만 손보면 다양한 응용이 가능할 것 같습니다.

```

while(1)
{
    //TODO:: Please write your application code
    wifiMain();

    if ( isElapsed())
    {
        sprintf(strTemp, "hello:%d\r\n", counter++); //가장 핵심적인 부분
        wifiSendData(strTemp, strlen(strTemp)); //가장 핵심적인 부분
        debugprint("TX:%s\r\n", strTemp); //가장 핵심적인 부분
    }
}

```

클라이언트 wifitest.c 코드에서 실질적으로 데이터를 보내는 부분의 코드는 상당히 적습니다. strTemp도 단순히 문자열입니다. 마찬가지로 잘만 손보면 다양한 응용이 가능합니다.

```

30 int main(void)
31 {
32     uint8_t strTemp[256];
33     debugInit();
34     wifiInit();
35
36     sei();
37
38     debugprint("wifi test start\r\n");
39     _delay_ms(1000);

```

Specifier	Common Equivalent	Signing	Bits	Bytes	Minimum Value	Maximum Value
int8_t	signed char	signed	8	1	-128	127
uint8_t	unsigned char	unsigned	8	1	0	255
int16_t	short	signed	16	2	-32,768	32,767
uint16_t	unsigned short	unsigned	16	2	0	65,535
int32_t	long	signed	32	4	-2,147,483,648	2,147,483,647
uint32_t	unsigned long	unsigned	32	4	0	4,294,967,295
int64_t	long long	signed	64	8	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
uint64_t	unsigned long long	unsigned	64	8	0	18,446,744,073,709,554,615

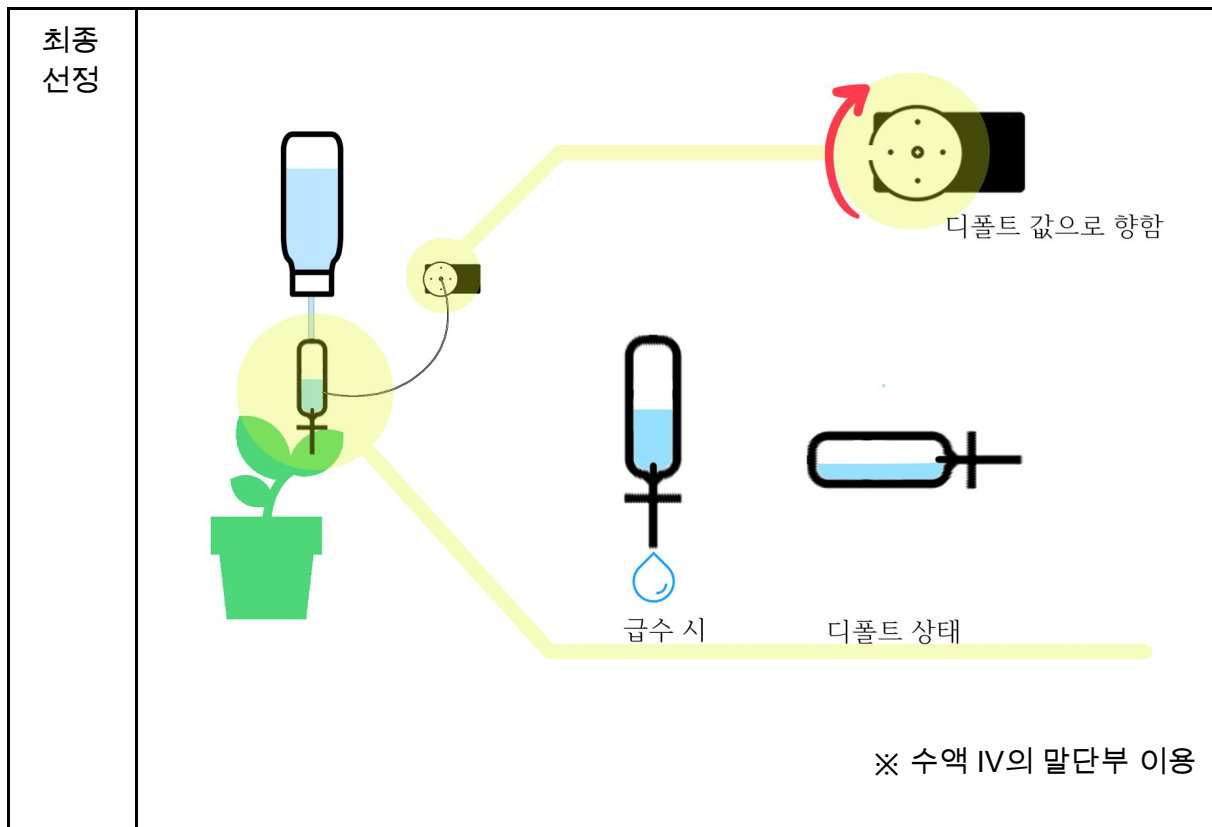
### 실습 일지

일시	2017년11월03일 09:00~12:20	장소	W6 301호	인원	권민석, 박상원, 이석원, 정동호
금일 목표	1팀 권민석, 정동호	지난 주차 구상 후보를 테스트 하여 최종 선정 한다. 선정된 후보의 모터 작동을 제어한다.			
	2팀 이석원	컨트롤 보드로부터 받은 데이터를 라즈베리파이에서 파일로 저장하기.			
	2팀 박상원	외부 인터넷 망에서 라즈베리파이 node.js 의 자바 스크립트 파일 접근하기			

### 1팀 실습 내용

선정 과정	<p>1 안은 모의 실험 결과 물 높이 차이로 생기는 위치에너지로 인해 물이 항상 배출되는 문제가 있었다. 또한 모터가 호스를 직접 방향을 바꾸는 데에는 모터의 힘이 부족하였다.</p> <p>2 안의 수액 조절기를 이용할 때 역시 모터의 힘이 부족한 데에서 문제가 발생했다. 조절 레버를 약 두바퀴 정도 돌려 주어야 하는데 레버를 돌리지 못하고 모터가 멈춰 버렸다.</p> <p>모터의 동력이 생각보다 적은 것을 파악 후 최소한의 동력을 활용하는 방안을 구상하였다.</p>
----------	--





1 안에서는 호스가 하늘을 향하고 있어도 물이 계속 뿜어져 나오는 문제점을 갖고 있었다.  
 2 안에서는 모터가 수액 조절기의 레버를 돌릴만한 힘이 없다는 것을 확인하였다  
 // 나중에 추가

#### 구상 및 작동



물이 공급되지 않을 때: //추후 설명 추가





물을 공급 할 때 : //추후 설명 추가

// 수정된 모터 소스 추가

작동 영상 : <https://youtu.be/Zi8lasbFekQ>

### 실습 도중 문제 해결 방법 또는 앞으로 개선점

모터와 연결되는 실감기 장치 고안

1. 실이 엉키지 않아야 함
2. 실보다 두꺼운 것을 사용해야 함

다음주차 수업에서는 토양 습도 센서로부터 값을 전달 받아 모터를 제어한다.

### 2팀 실습 내용

컨트롤 보드로부터 받은 데이터를 라즈베리파이에서 파일로 저장하기.

먼저 파일  
입력도중에  
일정 크기를  
넘어가면  
자동으로  
저장하고  
새파일을  
여는  
프로그램을  
작성해봤습  
니다.  
filetest.c

```

1  # include <stdio.h>
2  # include <stdlib.h>
3  int main()
4  {
5      int i = 0;
6      int j ;
7      int cur, start, end;
8      char stringtest[20];
9      char filename[20] = "filetest0.txt";          //파일 이름
10
11      FILE* stream = fopen(filename, "w");          //파일 오픈
12      if(stream == NULL)                            //오픈 확인
13      {
14          printf("file open error \n");
15          exit(1);
16      }
17
18      while(i < 10)                                  //반복
19      {
20          if(ftell(stream) > 100)                    //각 파일의 크기 검사
21          {
22              printf("new file opened \n");
23              if(fclose(stream) == EOF)                //크기를 벗어나면 파일 닫음
24              {
25                  printf("file close error \n");
26                  exit(1);
27              }
28
29              filename[8]++;                            //새로운 파일을 열기 위한 단계
30              stream = fopen(filename, "w");            //새 파일 오픈
31
32              if(stream == NULL)                        //오픈 확인
33              {
34                  printf("file open error \n");
35                  exit(1);
36              }
37
38              {
39                  printf("file open error \n");
40                  exit(1);
41              }
42              i++;                                     //프로그램 종료 조건 ----빠도 됨
43          }
44
45          scanf("%d", &j);                             //데이터 입력 -----
46
47          fprintf(stream, "test : %d ", j);             //파일 입력
48
49          end = ftell(stream); //-----
50
51          printf("file end pointer loc : %d \n", end); //-----
52
53          if(j == -1)                                    //프로그램 종료조건 -----
54          {
55              if(fclose(stream) == EOF)
56              {
57                  printf("file close error \n");
58                  exit(1);
59              }
60              exit(0);
61          }
62      }
63
64      exit(0);
65  }

```

실행화면  
파일크기  
100을  
넘어서  
자동으로  
폐쇄된  
filetest0.txt  
와 자동  
으로 생성된  
filetest1.txt

C:\Users\PC\Desktop\newfolder\1

```
1
file end pointer loc : 9
2
file end pointer loc : 18
3
file end pointer loc : 27
4
file end pointer loc : 36
5
file end pointer loc : 45
6
file end pointer loc : 54
7
file end pointer loc : 63
8
file end pointer loc : 72
9
file end pointer loc : 81
10
file end pointer loc : 91
11
file end pointer loc : 101
new file opened
12
file end pointer loc : 10
123
file end pointer loc : 21
4123
```

filetest0.txt - 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

test : 1 test : 2 test : 3 test : 4 test : 5 test : 6 test : 7 test : 8 test : 9 test : 10 test : 11

filetest1.txt - 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

test : 12 test : 123 test : 4123 test : -1

테스트를  
토대로  
echo\_ser  
v.c를  
수정했습니  
다.

```

122 char filename[20] = "filetest0.txt"; //파일 이름 -----수정 v
123
124 FILE* stream = fopen(filename, "w"); //파일 오픈
125 if(stream == NULL) //오픈 확인
126 {
127     printf("file open error \n");
128     exit(1);
129 } //-----수정 ^
130
131 while(1)
132 {
133     if(ftell(stream) > 1000) //각 파일의 크기 검사 -----수정 v
134     {
135         printf("new file opened \n");
136         if(fclose(stream) == EOF) //크기를 벗어나면 파일닫음
137         {
138             printf("file close error \n");
139             exit(1);
140         }
141         filename[8]++; //새로운 파일을 열기 위한 단계
142         stream = fopen(filename, "w"); //새 파일 오픈
143         if(stream == NULL) //오픈 확인
144         {
145             printf("file open error \n");
146             exit(1);
147         }
148     } //-----수정 ^
149
150     readBufSize = read(clnt_sock, rcv_buf, RX_DATA_MAX); //소켓으로부터 문자를 읽어서 rcv_buf에 저장 하
151     if(readBufSize > 0)
152     {
153         printf("rx len:%d\n", readBufSize); //read()할 수는 정상적으로 읽었다면 읽은 BYTE수를 반환 한다.
154         rcv_buf[readBufSize] = 0; //맨 뒷자리를 0으로 셋팅
155         printf("%s\n", rcv_buf); //읽어온 대로 출력
156         fprintf(stream, "%s", rcv_buf); //파일 입력 -----수정 =====
157     }
158 }

```

그러나  
컴파일 하는  
도중에  
문제가  
생겼습니다.

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <unistd.h>
5 #include <arpa/inet.h>
6 #include <sys/socket.h>
7 #include <pthread.h>

```

일부 헤더파일은 유닉스 운영체제의 표준 헤더파일이기 때문에 devc++도 아니고  
atmelstudio도 아닌 라즈베리파이 debian 운영체제에서 gcc로 컴파일 해야 합니다. 그러기  
위해서는 코드를 라즈베리파이로 옮겨야 하는데 usb가 없어서 gmail로 전송 했지만  
라즈베리파이의 safari버전은 더이상 gmail이 지원하지 않는다고 하여 크롬 브라우저를  
설치하기로 했습니다.

## 외부 인터넷 망에서 라즈베리파이 node.js 의 자바 스크립트 파일 접근하기

라즈베리파  
이에 설치  
할 수 있는  
nodejs 설치  
파일을  
찾습니다.

Linux Binaries (x86/x64)

[Linux Binaries \(ARM\)](#)

Source Code

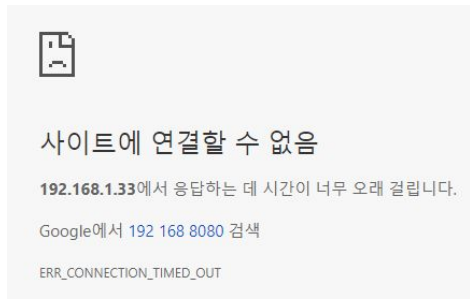
32-bit		64-bit	
ARMv6	ARMv7	ARMv8	
node-v8.9.0.tar.gz			

nodejs.org/en/download 화면을 캡처했습니다.

위키백과에서 arm 아키텍처에 관해 검색한 화면입니다.	Cortex	ARMv7-R	Cortex-R4(F)	임베디드 영상, (FPU)
		ARMv6-M	Cortex-M1	FPGA와 연동, 마이크로컨트롤
		ARMv7-M	Cortex-M3	마이크로컨트롤러 형상, Thumb
		ARMv7E-M	Cortex-M4	마이크로컨트롤러 형상
		ARMv8-A	Cortex-A53	64비트 명령어 지원
			Cortex-A57	64비트 명령어 지원
			Cortex-A72	64비트 명령어 지원

실습 자료 중에 라즈베리 파이3 데이터시트를 캡처한 화면입니다.	<b>Processor</b> Broadcom BCM2387 chipset. 1.2GHz Quad-Core ARM Cortex-A53 802.11 b/g/n Wireless LAN and Bluetooth 4.1 (Bluetooth Classic and LE)
-------------------------------------	--

nodejs를 설치후 예제 프로그램을 실행해봤습니다.	
-------------------------------	--

그런데 다른 컴퓨터에서 접속되지 않습니다.	
-------------------------	--

공개sw 강의에서 nodejs는 크롬기반으로 만들어져서 크롬이 깔려있어야 한다고 강사가 말했던 기억이 나서 크롬을 설치하기로 했습니다.

실습 도중 문제 해결 방법 또는 앞으로 개선점	
크롬 설치를 시도했습니다.	<h2>Linux용 Chrome 다운로드</h2> <p>Debian/Ubuntu/Fedora/openSUSE</p> <p>다운로드 패키지를 선택하세요.</p> <ul style="list-style-type: none"> <li><input checked="" type="radio"/> 64 bit .deb (Debian/Ubuntu용)</li> <li><input type="radio"/> 64 bit .rpm (Fedora/openSUSE용)</li> </ul> <p>Debian/Ubuntu나 Fedora/openSUSE 사용자가 아닌가요? 커뮤니티요.</p>

정식으로 배포하는 크롬은 amd 아키텍처 기반으로 만들어졌는데 우리가 쓰는 장비는 arm을 쓰고 있어서 오류가 나고, 설치가 안됐습니다. 그래서 인터넷에서 찾아보다가 크롬의 유사품인 크로미움 브라우저를 설치하기로 했습니다.

그러나  
크로미움도  
설치가  
안됐습니다.

이번 시간은 금일목표에 한참 모자라는 결과를 얻었습니다. 그리고 애초에 nodejs 예제프로그램은 오로지 로컬망에서 연결 가능 하다는 것도 알았고, 크롬을 설치해도 문제가 해결 될지도 모르겠습니다. 서버프로그램부분은 팀원들이 모르는 부분이 너무 많아서 지금까지 추측과 인터넷정보에 의지해왔는데, 앞으로 어떻게 해야 할지 고민입니다.