컴퓨터 응용 설계 및 실험

텀 프로젝트

4차 보고서

2020.07.02

인체감지 센서(적외선 센서)와 초음파 센서를 이용한 침입자 확인 시스템

1조

201524582 정희석

201424458 박인철

목차

1. 프로젝트 주제

- 초음파 센서와 적외선 인체감지센서를 이용한 방문자 확인 및 위치 추정 시스템

주제선정 이유: 현재 배운 것으로 최대한 가능한 것은 초음파 센서를 이용하고 드라이버를 만드는 것을 배웠으므로 인체감지센서 드라이버를 만들고 WiFi 소켓통신을 이용하여 서버-클라이언트 통신을 이용하여 Host PC에 결과를 출력하도록 할 수 있다. 현재 가지고 있는 Display 출력 문제를 해결하면 카메라 모듈로부터 LCD에 영상을 출력할 수 있을 것 같아서 이 주제를 선정하게 되었다.

2. 프로젝트 목표

실험 시간에 배운 ACHRO보드 상에서 GPIO를 사용하여 센서를 부착하고 이를 Linux 운영체제에서 조작하며 WiFi 모듈을 통한 소켓 통신을 사용하여 센서의 결과를 보드의 Linux에서 처리하여 소켓 통신을 통해 Host PC에 출력하는 것이 1차 목표이고 최종 목표는 센서 값이 들어온다면(침입이 확인된다면) 보드의 카메라 모듈에 의해 촬영된 것을 디스플레이에 실시간으로 출력하여 확인이 가능하도록 하는 것이다.

3. 사용 보드, 센서 및 모듈

보드: ACHRO-I MX6Q Cortex-A9 CPU 2GB DDR3 RAM 32-bit

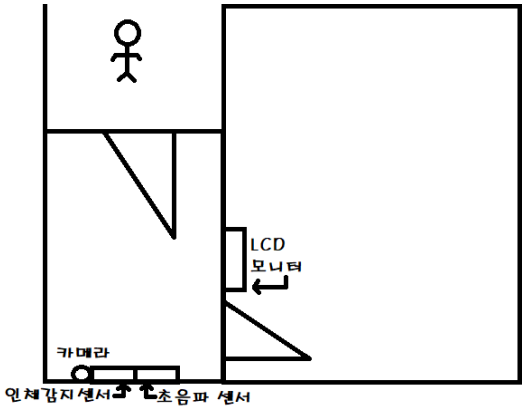
센서: 초음파 센서, 인체 감지 센서

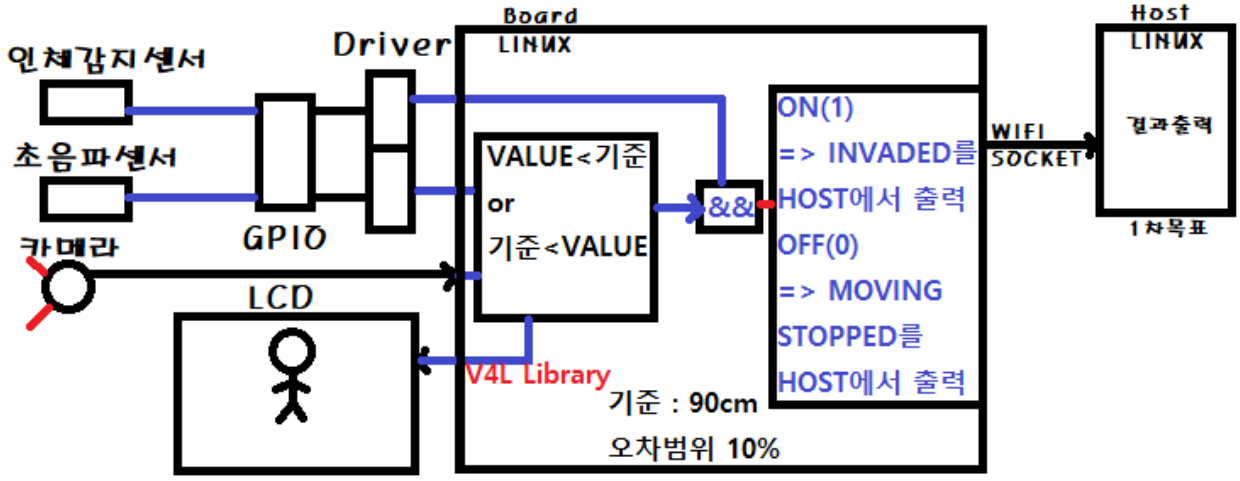
모듈: 카메라 모듈, (ON BOARD)LCD 모듈

ACHRO-I MX6Q 보드에 Huins사에서 제공한 Linux와 arm-2014.05 Toolchain을 설치

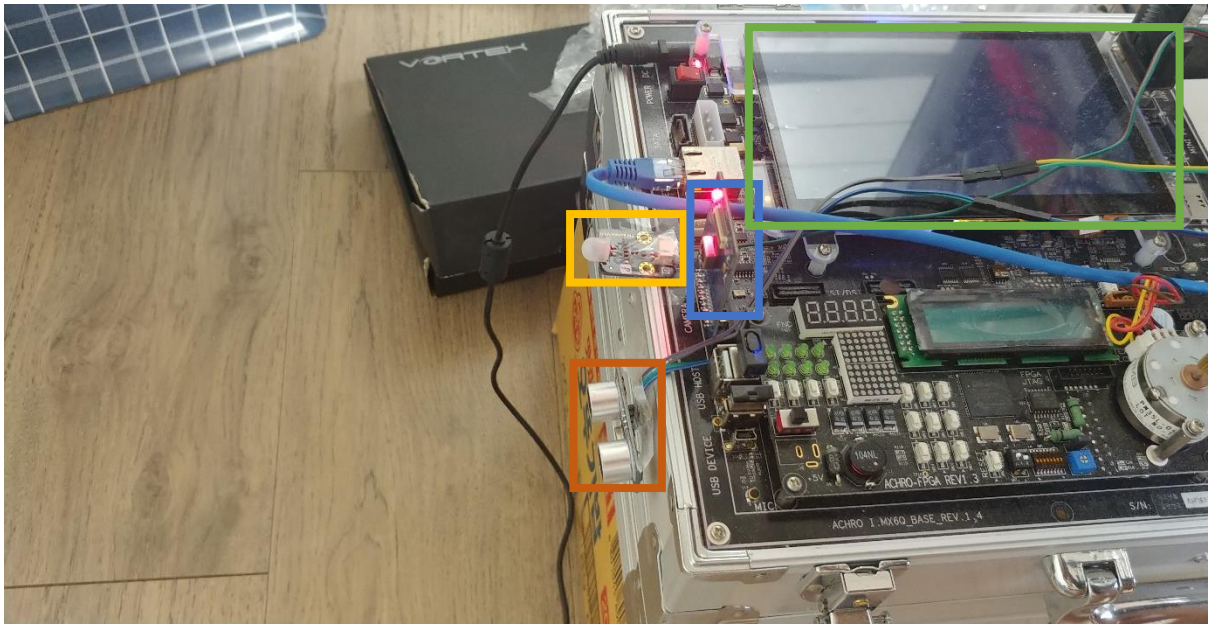
초음파 센서와 인체 감지 센서를 GPIO를 통해서 연결, 각 센서의 드라이버를 구현하고, 카메라 모듈을 위해 OpenCV를 설치, V4L Library를 사용, LCD를 위해 QT Library를 설치하여 사용하였다.

4. 구현 예상도





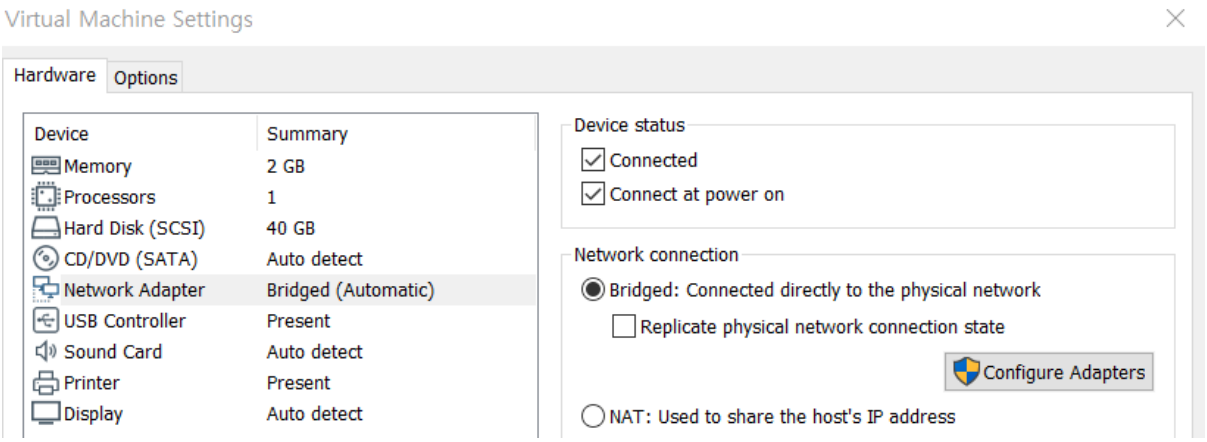
5. 구현 사진



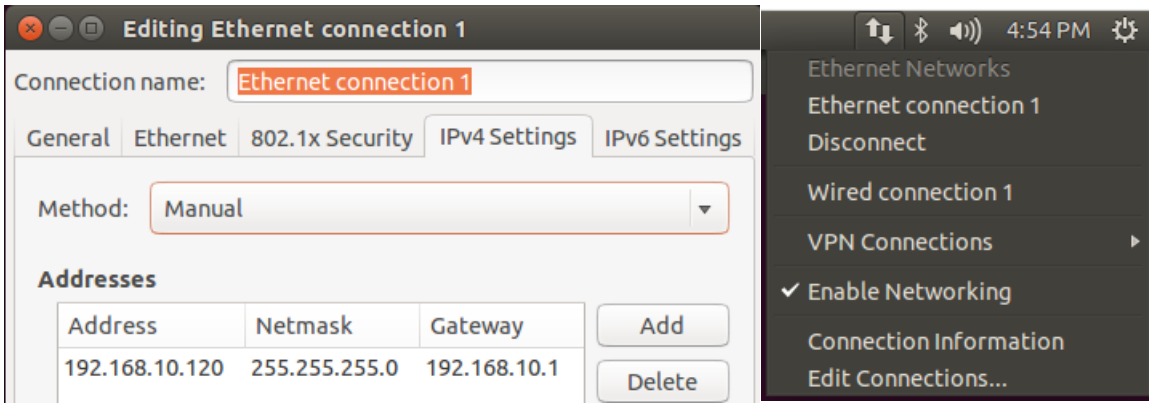
— 카메라 모듈 — 초음파 센서 모듈 — 인체 감지 센서 모듈 — LCD

6. 진행 환경

1) Direct Cable을 통한 NFS



<가상 머신 설정 -> 네트워크 Bridged 설정으로 Windows의 LAN포트에 직접 Mapping>

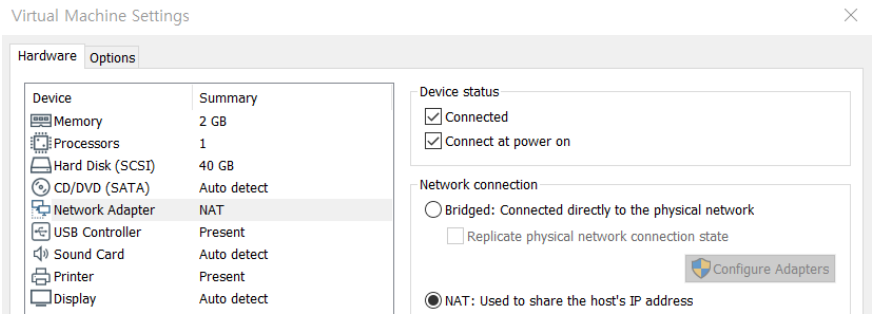


<보드와의 연결에 사용될 주소 설정>

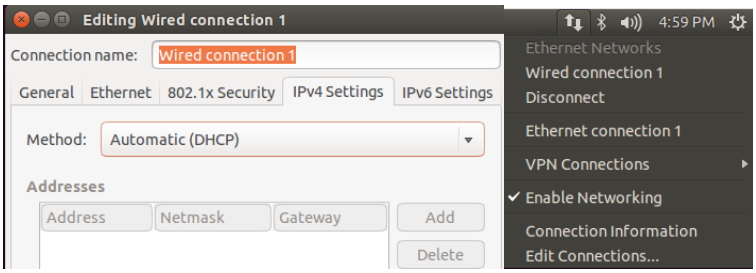
```
[root@ACHRO SERVER]# ifconfig eth0 192.168.10.121
[root@ACHRO SERVER]# /mnt/nfs/TermProject/SERVER
-sh: /mnt/nfs/TermProject/SERVER: Is a directory
[root@ACHRO SERVER]# cd /mnt/nfs/TermProject/SERVER
[root@ACHRO SERVER]# cp SVR SVR_4_SHOW ~/SERVER
[root@ACHRO SERVER]# cd ~/SERVER
[root@ACHRO SERVER]#
```

<보드에서는 121, 가상머신은 120을 사용하여 NFS를 통해 파일을 복사>

2) WiFi 연결



<가상 머신 설정 -> 네트워크 NAT 설정으로 Windows의 WiFi를 사용하도록 함>



<무선 어댑터로부터 자동으로 IP를 할당 받아 사용(Windows의 IP에 연결)>

```
[root@ACHRO SERVER]# ifconfig wlan0 up
[root@ACHRO SERVER]# wlan0: Trying to associate with 00:e0:4d:08:08:b4 (SSID='Nu-Network' freq=2452 M)
ioctl[SIOCSIWENCODEEXT]: Invalid argument
ioctl[SIOCSIWENCODEEXT]: Invalid argument
wlan0: Association request to the driver failed
wlan0: CTRL-EVENT-DISCONNECTED bssid=00:e0:4d:08:08:b4 reason=0
ioctl[SIOCSIWENCODEEXT]: Invalid argument
ioctl[SIOCSIWENCODEEXT]: Invalid argument
wlan0: Trying to associate with 00:e0:4d:08:08:b4 (SSID='Nu-Network' freq=2452 MHz)
wlan0: Association request to the driver failed
wlan0: Associated with 00:e0:4d:08:08:b4
wlan0: Authentication with 00:e0:4d:08:08:b4 timed out.
ioctl[SIOCSIWAP]: Operation not permitted
wlan0: CTRL-EVENT-DISCONNECTED bssid=00:e0:4d:08:08:b4 reason=3 locally_generated=1
ioctl[SIOCSIWENCODEEXT]: Invalid argument
ioctl[SIOCSIWENCODEEXT]: Invalid argument
ioctl[SIOCSIWENCODEEXT]: Invalid argument
ioctl[SIOCSIWENCODEEXT]: Invalid argument
wlan0: Trying to associate with 00:e0:4d:08:08:b4 (SSID='Nu-Network' freq=2452 MHz)
wlan0: Association request to the driver failed
wlan0: Associated with 00:e0:4d:08:08:b4
wlan0: WPA: Key negotiation completed with 00:e0:4d:08:08:b4 [PTK=CCMP GTK=CCMP]
wlan0: CTRL-EVENT-CONNECTED - Connection to 00:e0:4d:08:08:b4 completed [id=0 id_str=]

[root@ACHRO SERVER]# route add default gw 192.168.10.1 dev wlan0
[root@ACHRO SERVER]# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8): 56 data bytes
64 bytes from 8.8.8.8: seq=0 ttl=110 time=75.134 ms
^C
--- 8.8.8.8 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 75.134/75.134/75.134 ms
[root@ACHRO SERVER]#
```

<보드 WiFi 연결>

-> wlan0에 IP를 할당하고 공유기에 접속, 기본 gateway를 설정하여 인터넷 접근>

7. 진행 과정

목표 날짜	구현 내용
~6/2	초음파 센서 처리 => 완료
~6/9	인체 감지 모듈 드라이버 및 처리 => 진행중 Wifi 소켓 통신 구현 => 완료 (소켓 통신 + 초음파센서 진행중) 1차 목표 테스트 => 인체 감지 모듈 미구현으로 미시행
~6/16	Display에 카메라 화면 출력(Display 문제 해결) -> Opencv 설치 과정에서 오류 발생 확인 (소켓 통신 + 초음파센서 동작 확인 완료) 인체 감지 모듈 구현 중
~6/23	Display에 카메라 화면 출력(Opencv 오류 해결) ->카메라 동작 확인 완료 인체 감지 모듈 구현 중(조원 연락 두절)
~6/30	카메라 코드 분석 및 프로젝트에 이식 완료 WIFI 접속 외부망 -> 내부망 작업 완료 초음파 센서 값을 통한 카메라 동작 구현 완료
~7/1	인체 감지 센서 드라이버 구현 완료 및 프로젝트에 이식 완료 최종 목표 테스트
~7/2	최종 보고서 작성
7/3	최종 발표 및 시연, 프로젝트 제출

(1차 보고서 때 상황)

6/1 월요일 연구실 방문을 통해 인체 감지 모듈을 수령하였으며 현재 4주차의 GPIO 드라이버를 통한 센서 조작을 해보고 있습니다. 이를 소켓통신을 통해서 host PC에 값을 출력하는 것이 6/5 일까지의 목표로써 진행하고 있습니다. 초음파 센서의 드라이버와 동작 구현이 완료되면 인체 감지 모듈의 드라이버를 만들고 동작을 확인하는 것이 다음 목표이며 현재 QT Creator를 설치 및 사용 테스트 중입니다.

(2차 보고서 때 상황)

6/3 QT Porting Library, QT Creator 설치 완료 및 디스플레이에 시계 예제 출력 완료

6/5 초음파 센서 드라이버 및 동작 구현 중

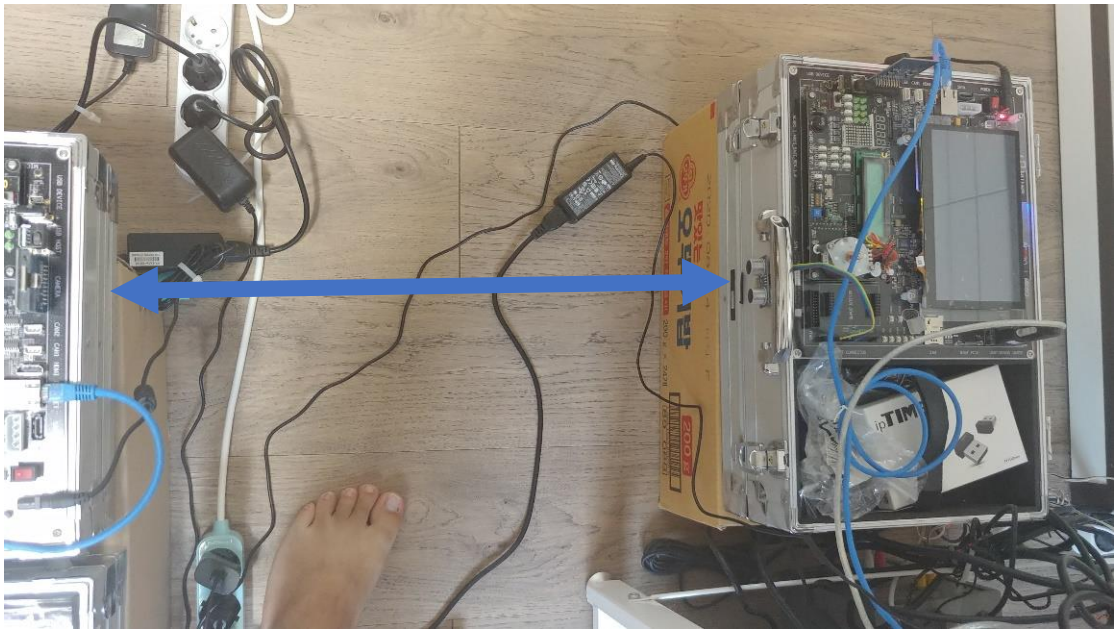
6/6 WiFi 소켓 통신 구현 -> ACHRO보드: 서버, HOST 컴퓨터: 클라이언트

=> 이유: HOST 컴퓨터는 VM 가상머신 상에서 동작 하므로 WiFi 소켓 서버를 올려서 동작하도록 하려면 각종 설정이 필요. VMware Workstation 15에서 진행하기 어렵다 판단했기 때문.

6/7 초음파 센서 드라이버 완료 및 WiFi 소켓 통신과 센서 동작을 하나의 프로그램으로 작성 중

(3 차 보고서 상황)

<정상 상태>



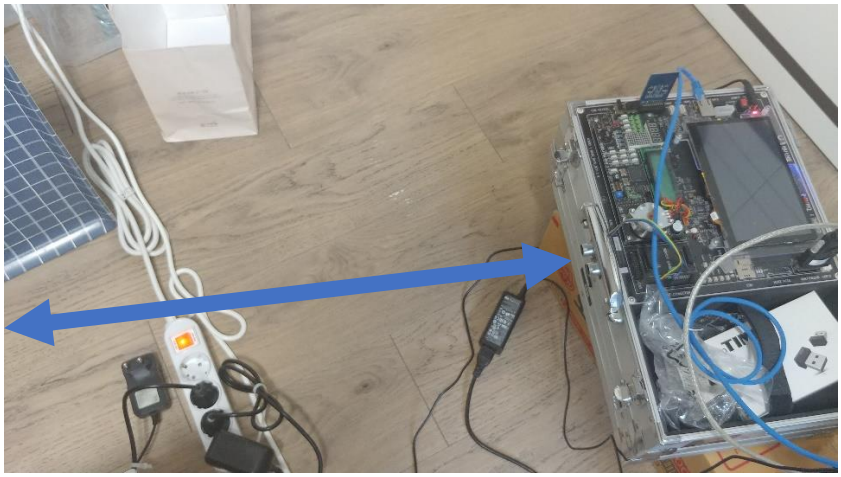
>클라이언트에 아무것도 안 뜸 -> 설정 50cm

```
root@ubuntu:/nfsroot/TermProject/CLIENT# ./CLI
complete!
```

```
root@ubuntu: ~
[  849.447862] <enter sonicburst>
[  849.451014] <exit sonicburst>
[  849.451497] <ECHO = 1>
[  849.454501] <ECHO = 0>
ultra sensor: 51
[  849.959990] <enter READ>
[  849.962636] <enter sonicburst>
[  849.965790] <exit sonicburst>
[  849.966271] <ECHO = 1>
[  849.969278] <ECHO = 0>
ultra sensor: 51
[  850.474751] <enter READ>
[  850.477373] <enter sonicburst>
[  850.480525] <exit sonicburst>
[  850.481004] <ECHO = 1>
[  850.483988] <ECHO = 0>
ultra sensor: 51
[  850.989510] <enter READ>
[  850.992132] <enter sonicburst>
[  850.995307] <exit sonicburst>
[  850.995789] <ECHO = 1>
[  850.998771] <ECHO = 0>
```

> 보드에서는 확인용으로 거리를 출력하게 함

<도난 상태>



>클라이언트에 거리가 뜨면서 알림을 출력

```
root@ubuntu:/nfsroot/TermProject/CLIENT# ./CLI
complete!

51SOMEONE INVADED
130
SOMEONE INVADED
130
SOMEONE INVADED
129
SOMEONE INVADED
129
SOMEONE INVADED
129
SOMEONE INVADED
129
SOMEONE INVADED
130
SOMEONE INVADED
```

```
root@ubuntu: ~
[  951.420646] <enter READ>
[  951.423288] <enter sonicburst>
[  951.426442] <exit sonicburst>
[  951.426922] <ECHO = 1>
[  951.432110] <ECHO = 0>
ultra sensor: 130
INVADED
[  951.934680] <enter READ>
[  951.937303] <enter sonicburst>
[  951.940455] <exit sonicburst>
[  951.940936] <ECHO = 1>
[  951.946105] <ECHO = 0>
ultra sensor: 130
INVADED
[  952.448657] <enter READ>
[  952.451279] <enter sonicburst>
[  952.454469] <exit sonicburst>
[  952.454951] <ECHO = 1>
[  952.460116] <ECHO = 0>
ultra sensor: 129
INVADED
[  952.962678] <enter READ>
```

>보드에는 센서 값과 알림을 확인을 위해 출력하게 하였다.

아직 개발 중으로 우선 ultrasonic의 값을 wifi를 통해서 클라이언트에서 출력이 가능함을 확인하는 과정을 진행하였다. 아직 인체감지센서 담당한 팀원에게서 진행상황을 묻고자 연락을 했으나 아무 소식이 없어서 여기까지 테스트를 진행하였다. 그리고 카메라 모듈을 위한 Opencv의 설치에서 cmake시 발생하는 unrecognized option "–fix-cortex-a8"오류가 있어서 이를 확인하고 있다.

(4 차 보고서 상황)

```
root@ubuntu:/work/achroimx6q/achroimx6q_opencv/build# cmake -DCMAKE_INSTALL_
PREFIX=/opt/toolchains/arm-2014.05/opencv -DSOFTFT=ON -DCMAKE_TOOLCHAIN_FILE
=../opencv-3.2.0/platforms/linux/arm-gnueabi.toolchain.cmake ../opencv-3.2.0
-- The CXX compiler identification is GNU 4.8.4
-- The C compiler identification is GNU 4.8.4
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- broken
CMake Error at /usr/share/cmake-2.8/Modules/CMakeTestCXXCompiler.cmake:54 (m
essage):
  The C++ compiler "/usr/bin/c++" is not able to compile a simple test
  program.

  It fails with the following output:

   Change Dir: /work/achroimx6q/achroimx6q_opencv/build/CMakeFiles/CMakeTmp
```

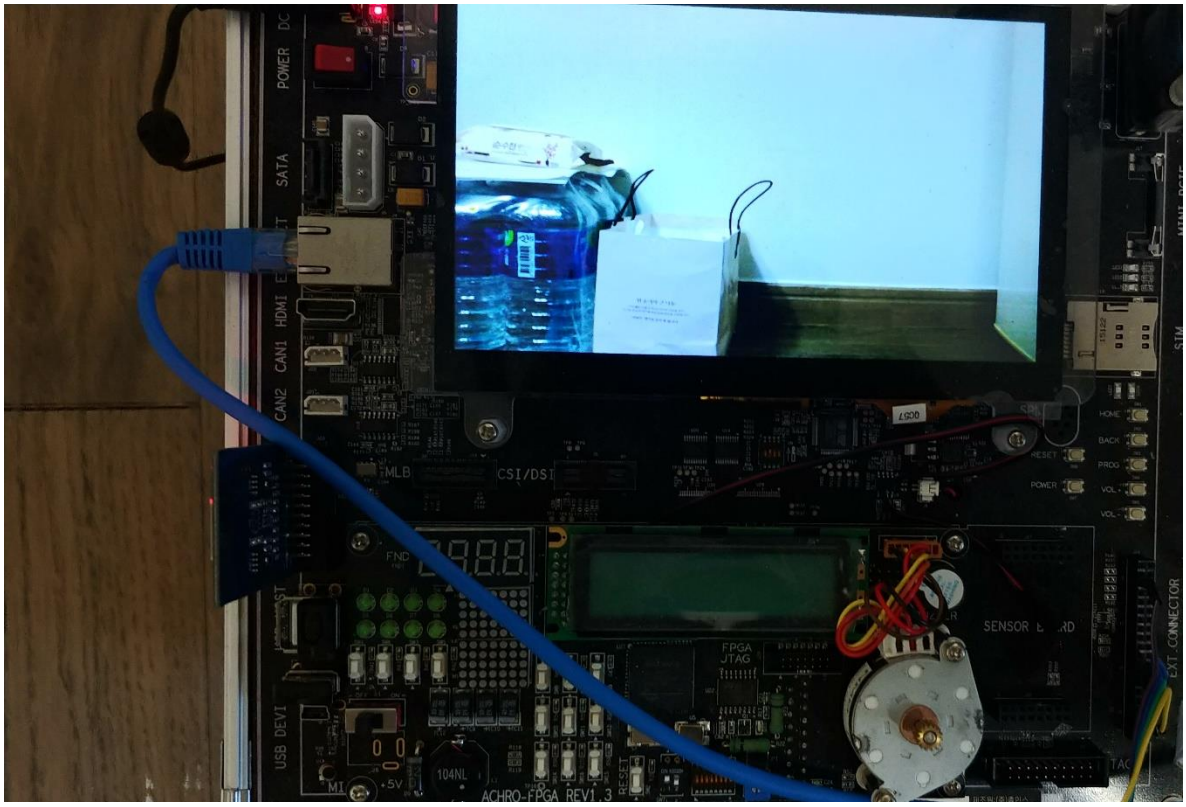
<오류 내역>

cmake시 발생하는 unrecognized option "–fix-cortex-a8"오류가 발생한 원인이 compiler의 위치를 제대로 잡지 못해서 발생하였다. 이를 해결하기 위해 파일을 찾아보다가 build내에 생성된 CMakeCache.txt파일에서의 CXX Compiler, C Compiler, AR의 위치를 재설정해주었다.

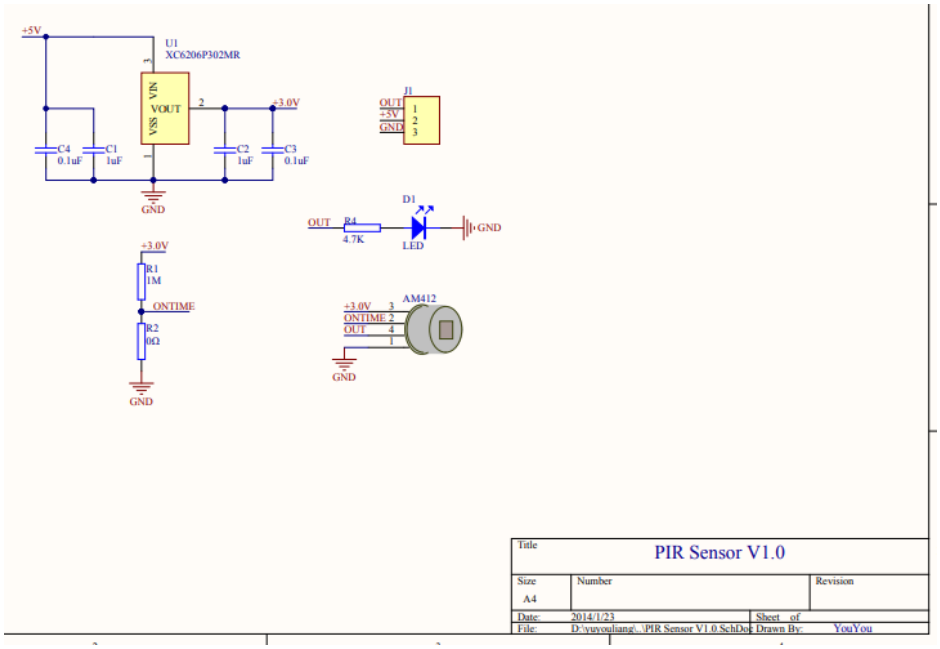
```
//Path to a program.
CMAKE_AR:FILEPATH=/opt/toolchains/arm-2014.05/bin/arm-none-linux-gnueabi-ar

//Choose the type of build, options are: None(CMAKE_CXX_FLAGS or
// CMAKE_C_FLAGS used) Debug Release RelWithDebInfo MinSizeRel.
CMAKE_BUILD_TYPE:STRING=

//Enable/Disable color output during build.
CMAKE_COLOR_MAKEFILE:BOOL=ON

//Configs
CMAKE_CONFIGURATION_TYPES:STRING=Debug;Release

//CXX compiler.
CMAKE_CXX_COMPILER:FILEPATH=/opt/toolchains/arm-2014.05/bin/arm-none-linux-gnueabi-g++

//Flags used by the compiler during all build types.
CMAKE_CXX_FLAGS:STRING=

//Flags used by the compiler during debug builds.
CMAKE_CXX_FLAGS_DEBUG:STRING=-g

//Flags used by the compiler during release minsize builds.
CMAKE_CXX_FLAGS_MINSIZEREL:STRING=-Os -DNDEBUG

//Flags used by the compiler during release builds (/MD /Ob1 /Oi
// /Ot /Oy /Gs will produce slightly less optimized but smaller
// files).
CMAKE_CXX_FLAGS_RELEASE:STRING=-O3 -DNDEBUG

//Flags used by the compiler during Release with Debug Info builds.
CMAKE_CXX_FLAGS_RELWITHDEBINFO:STRING=-O2 -g -DNDEBUG

//C compiler.
CMAKE_C_COMPILER:FILEPATH=/opt/toolchains/arm-2014.05/bin/arm-none-linux-gnueabi-gcc
```

<CMakeCache.txt에서 수정한 부분>

그리고 ppt의 20페이지에 cp -a /usr/local/lib/* . 에서 opencv의 내부 파일이 없어서 찾아보니 /work/achroimx6q/achroimx6q_opencv/build/lib에 있어서 이 파일들을 nfs를 이용하여 보드로 옮겼다.

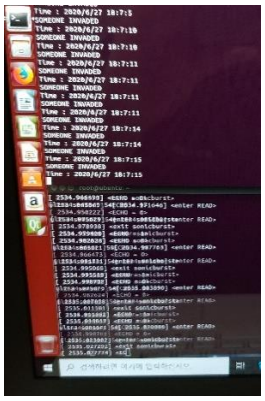


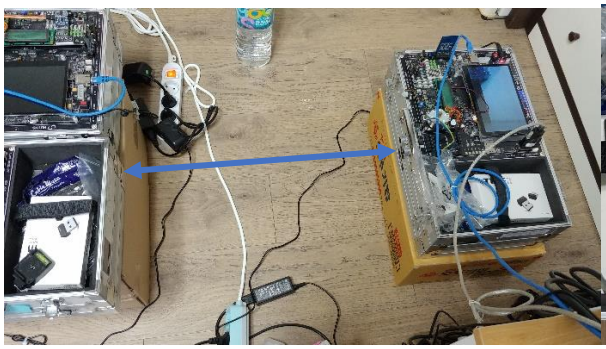
> 8주차 실험자료인 OpenCV를 이용하여 디스플레이에 카메라화면을 띄우는 camera_test의 실행에 성공하였다. 그리고 새로 받은 센서 SEN0171의 회로도를 구했다. 이 센서는 GPIO2_4(30)에 OUT, 5V(8)에 +, GND(38)에 -를 연결할 계획이다.



이와 DVD1-SRC의 gpio_driver의 예시를 분석하여 이 인체감지 센서의 드라이버를 만들 계획이다. 문제는 이 센서를 연결한 점프 케이블이 없어서 주중에 연구실을 방문하여 bread-board와 케이블을 대여할 계획이다.

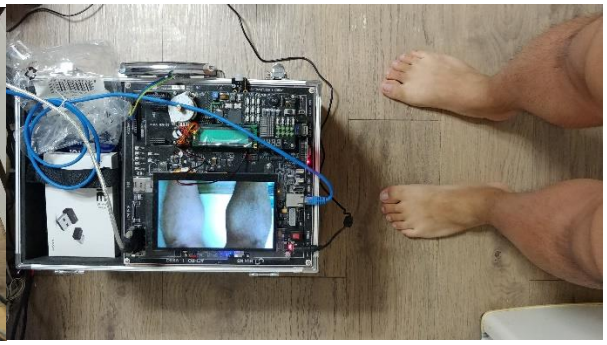
(5차 보고서 상황)


<-센서 값이 정상범위(40~60) 값은 54,55가 나온다




-> 카메라가 동작하지 않는다.

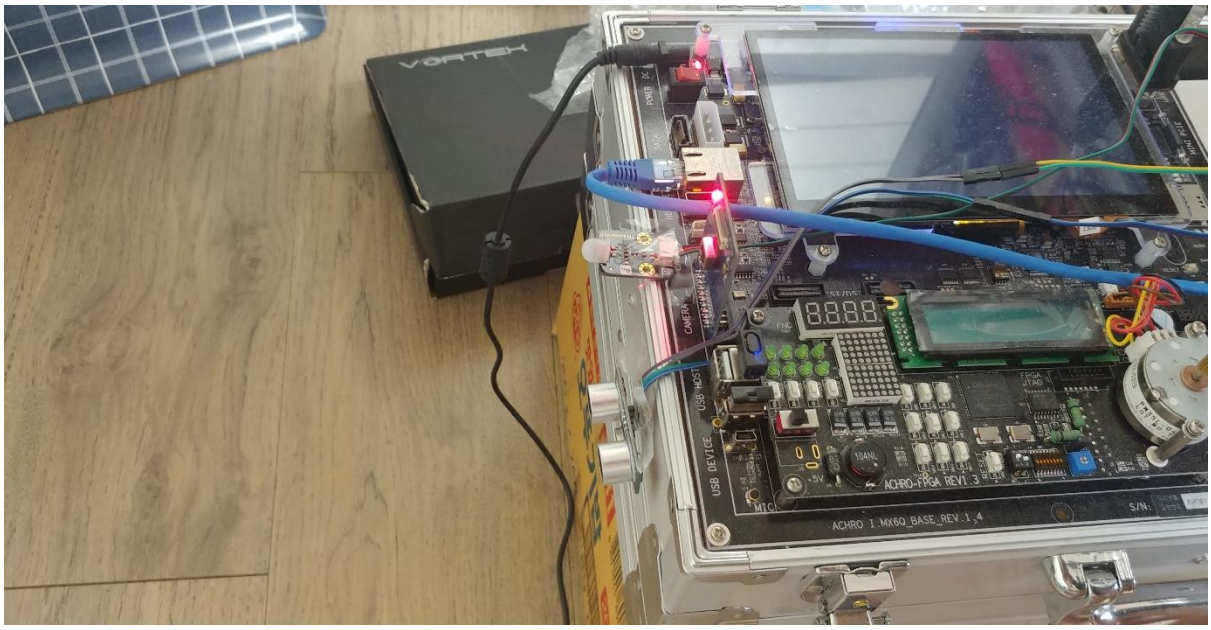

-> 센서 값이 낮음 -> 중간에 무언가 있는 경우

-> 중간에 물통이 있어서 카메라가 동작

이제 인체감지센서의 드라이버를 만들어서 초음파 센서의 값이 바뀐 후 인체감지센서의 값을 읽어 들여서 카메라를 출력하게 하여도 되고 지금처럼 초음파 센서의 값이 바뀌면 카메라를 동작시키고 인체감지 센서가 인식하면 host에 wifi로 침입여부를 알려줄 수 있을 것 같다.

(최종 보고서 상황)



5차 보고서 이후 연구실을 방문하여 점퍼 케이블과 Bread-Board를 대여하여 초음파 센서의 위치를 카메라의 위치로 옮기고 인체 감지 센서와 함께 테이프로 고정, 모든 동작이 구현이 완료되었다.

8. 역할 분담

정희석: WiFi 소켓 통신, Display, Camera 모듈 제어, GPIO 드라이버 제작 및 제어

~~박인철: 초음파 센서, 인체 감지 센서 GPIO 드라이버 제작 및 제어(연락 두절)~~

9. 코드 분석

1) Driver

-1) PIR(인체 감지 센서): pir_driver.c => 기본 제공 예시 ext_sensor_driver.c 를 변형

```
#include <linux/gpio.h>
#include <linux/uaccess.h>
#include <linux/input.h>
#include <linux/kdev_t.h>
#include <linux/cdev.h>

#define GPIO_DATA       IMX_GPIO_NR(2,4)

static int pir_open(struct inode *inode, struct file *file);
static int pir_release(struct inode *inode, struct file *file);
static int pir_read(struct file *filp, char *buf, size_t count, loff_t *f_pos);
static int pir_init(void);
static void pir_exit(void);
static int pir_register_cdev(void);

static int ext_major = 0, ext_minor = 0;
static dev_t pir_dev;
static struct cdev pir_cdev;


static int pir_open(struct inode *inode, struct file *file)
{
        printk(KERN_ALERT "< Device has been opened > \n");
        return 0;
}

static int pir_release(struct inode *inode, struct file *file)
{
        printk(KERN_ALERT "< Device has been closed > \n");
        return 0;
}

static int pir_read(struct file *filp, char *buf, size_t count, loff_t *f_pos)
{
        int res, ret;
        char data[10] = {0};
        res = gpio_get_value(GPIO_DATA);
        sprintf(data,"%d",res);
        ret = copy_to_user(buf,&data,count);
        return 0;
}

struct file_operations pir_fops = {
        .open = pir_open,
        .release = pir_release,
        .read = pir_read,
};
```

-> PIR 센서는 GPIO2 의 4 번핀(30)을 사용하며 pir_read 함수에서 값을 받아서 sprint, copy_to_user 를 통해 값을 전달하고 pir_fops 함수를 통해 함수의 실행을 mapping 한다.

Ex) server.c 에서 read 함수를 불렀을 때 file pointer 에 따라 pir_read 함수를 실행하게 한다.

```
static int pir_init(void)
{
        int result = 0;

        printk(KERN_ALERT "< EXT Sensor Module is up > \n");
        if((result = pir_register_cdev())<0){
                printk(KERN_ALERT "< EXT Sensor Register Fail > \n");
                return result;
        }

        result = gpio_request(GPIO_DATA,"DATA_PIN");
        if(result != 0 ){
                printk(KERN_ALERT "< DATA_PIN Request Fail > \n");
                return -1;
        }

        result = gpio_direction_input(GPIO_DATA);
        if (result != 0) {
                printk(KERN_ALERT "< DATA_PIN Configure set Fail > \n");
                return -1;
        }


        return 0;
}

static void pir_exit(void)
{
        printk(KERN_ALERT "< EXT Sensor Module is down > \n");
        gpio_free(GPIO_DATA);
        cdev_del(&pir_cdev);
        unregister_chrdev_region(pir_dev,1);
}
```

-> init 함수는 pir 을 insmod 로 설치했을 때 gpio 와 register 를 초기화하여 설정하는 함수이고 exit 함수는 pir 를 rmmod 로 삭제했을 때 호출되며 사용된 설정을 해제하는 함수이다.

```
static int pir_register_cdev(void)
{
        int error;

        if(ext_major){
                pir_dev = MKDEV(ext_major, ext_minor);
                error = register_chrdev_region(pir_dev,1,"pir");
        }else{
                error = alloc_chrdev_region(&pir_dev,ext_minor,1,"pir");
                ext_major = MAJOR(pir_dev);
        }

        if(error<0){
                printk(KERN_WARNING "pir : can't get major %d\n", ext_major);
                return -1;
        }

        printk(KERN_ALERT "major number = %d\n", ext_major);

        cdev_init(&pir_cdev, &pir_fops);
        pir_cdev.owner = THIS_MODULE;
        pir_cdev.ops = &pir_fops;
        error = cdev_add(&pir_cdev, pir_dev,1);

        if(error){
                printk(KERN_NOTICE "pir sensor Register Error %d\n", error);
        }

        return 0;
}
```

-> register_cdev 는 character device driver 를 초기화 및 설정하는 함수이며 major,minor number 가 있으면 해당 major number 로 device 를 초기화하고 등록, 없으면 새로 할당해서 등록한다. 그리고 fops 정보를 설정하여 연결시킨다.

-2) US(초음파 센서) us_driver.c -> 4 주차 실험 코드 응용

```
#include <linux/module.h>
#include <linux/cdev.h>
#include <linux/fs.h>
#include <linux/gpio.h>
#include <linux/delay.h>
#include <linux/kdev_t.h>
#include <linux/interrupt.h>
#include <asm/uaccess.h>

MODULE_LICENSE("GPL");

/* Funrction : IXM_GPIO_NR(x,y) */
/* -- x : Port # -- */
/* -- y : Pin # -- */
#define ULTRASONIC_ECHO IMX_GPIO_NR(2, 3)
#define ULTRASONIC_TRIGGER IMX_GPIO_NR(2, 2)
// US's ECHO -> GPIO2_PIN3
// US's TRIGGER -> GPIO2_PIN2
static int us_major=0, us_minor=0;
static int result, res;
static dev_t us_dev;
static struct cdev us_cdev;
int send_value=0;
char temp[100] = {0,};


/* Timeval 구조체 */    //in time.h
struct timeval after, before;
u32 irq = -1;

static int us_register_cdev(void);
static int us_open(struct inode *inode, struct file *filp);
static int us_release(struct inode *inode, struct file *filp);
static int us_read(struct file *filp, char *buf, size_t count, loff_t *f_pos);

void output_sonicburst(void);
int gpio_init(void);
static irqreturn_t ultrasonics_echo_interrupt(int irq, void *dev_id, struct pt_regs *regs);
```

-> ultrasonic 은 GPIO_2 의 2 번핀(26,TRIGGER)과 3 번핀(28,ECHO)을 사용

그리고 거리계산을 위한 시간측정용 timeval 구조체를 사용

```
/* Type : irqreturn_t */
/* Interrupt Callback function */
static irqreturn_t ultrasonics_echo_interrupt(int irq, void *dev_id, struct pt_regs *regs)
{
        // Ultrasonic의 ECHO pin의 GPIO 값에 대해서
        if(gpio_get_value(ULTRASONIC_ECHO))
        {
                // GPIO 값이 1일 때, driver module에서 시간을 측정한다.
                // driver module에서 시간을 측정하는 함수는 User 영역의 time 함수가 아닌 다른 함수 사
용
                //printk(KERN_ALERT "<ECHO = 1>\n");
                // 아래의 코드를 작성하시오
                do_gettimeofday( &before);
        }
        else
        {
                // GPIO 값이 0일 때, driver module에서 시간을 측정한다.
                //printk(KERN_ALERT "<ECHO = 0>\n");
                // 아래의 코드를 작성하시오
                do_gettimeofday( &after);
                // ECHO 값이 0일 때, driver Module에서 after 시간을 측정 후 거리를 계산한다.
                send_value = (after.tv_usec - before.tv_usec) / 58;
                // temp buf에 넣어 값을 메모리 상에 저장한다.
                sprintf(temp,"%d",send_value);
                memset(&before, 0 , sizeof(struct timeval ) );
                memset(&after , 0 , sizeof(struct timeval ) );
        }

        return IRQ_HANDLED;
}
```

-> GPIO 의 interrupt 동작을 정의한 함수로 us_read->output_sonicburst 의 trigger 의 1->0 이후 1ms 의 delay 동안 동작하는 함수이다. ECHO 값이 0 이라는 것은 아직 TRIGGER 값이 도착하지 않았다는 것이고 이때의 시간을 before 에 저장, TRIGGER 값이 도착한 ECHO 값이 1 인 시점의 시간을 after 에 저장하여 이 us 단위를 빼서 TRIGGER 가 물체에 부딪쳐서 ECHO 로 들어온 시간을 구하고 이를 58 로 나누어 cm 거리로 계산, 이를 결과 값으로 전달한다.

```
/* Function : us_open */
/* Ultrasonic Device Driver가 User Application에서 open 명령을 수행 했을 시 실행되는 함수 */
static int us_open(struct inode *inode, struct file *filp)
{
        printk(KERN_ALERT "< Device has been opened >\n");

        return 0;
}

/* Function : us_release */
/* Ultrasonic Device Driver가 User Application에서 release 명령을 수행 했을 시 실행되는 함수 */
static int us_release(struct inode *inode, struct file *filp)
{
        printk(KERN_ALERT "< Device has been closed > \n");

        return 0;
}

/* Function : us_read */
/* Ultrasonic Device Driver가 User Application에서 read 명령을 수행 했을 시 실행되는 함수 */
static int us_read(struct file *filp, char *buf, size_t count, loff_t *f_pos)
{

        //printk(KERN_ALERT "<enter READ> \n");
        output_sonicburst();
        mdelay(1);

        // Kernel 영역에서 User에게 값을 전달함
        copy_to_user(buf,temp,strlen(temp));
        return 0;
}

/* Type file_operations */
/* User Application에서 명령에 따른 동작을 Binding 하기 위한 구조체 */
struct file_operations us_fops = {//COMPLETE => 3rd week's fpga_xxx_driver.c check
        // open 동작 binding
        .open = us_open,
        // release 동작 binding
        // 아래의 코드를 작성하시오      //COMP
        .release = us_release,
        // read 동작 binding
        // 아래의 코드를 작성하시오      //COMP
        .read = us_read,
};
```

-> open, release, read 동작이며 read 에서 output_sonicburst 를 호출하여 TRIGGER 과 ECHO 에 관한 동작이 이루어지며 그 결과가 전역 변수로 선언된 char 형 배열 temp 에 저장, 이를 copy_to_user 를 통해 전달하게 된다. 그리고 us_fops 에 open, release, read 동작을 binding 함으로써 호출하여 사용할 수 있도록 한다.

```
/* Function : output_sonicburst */
/* Ultrasonic sensor의 수신부에서 거리 측정을 위해 Ping을 보내는 함수 */
void output_sonicburst(void)
{
        //printk(KERN_ALERT "<enter sonicburst> \n");
        // 아래의 코드를 작성하시오      //COMP
        // set high for 10us trigger pin, set low for wait
        gpio_set_value(ULTRASONIC_TRIGGER,1);
        udelay(10);
        gpio_set_value(ULTRASONIC_TRIGGER,0);
        //printk(KERN_ALERT "<exit sonicburst> \n");
}
```

-> Sonicburst 에서는 Trigger 핀을 통해 물체에 반사되어 돌아오는 신호를 만들어서 보낸다. 10us 의 길이를 가지는 1 의 신호가 TRIGGER 핀을 통해 나가서 ECHO 를 통해 들어온다.

```
/* Function : us_init */
/* module_init 함수의 파라미터로 사용되는, insmod 시 실행되는 함수 */
static int us_init(void)
{
        printk(KERN_ALERT "< Ultrasonic Module is up > \n");
        // timeval initialization
        // Ultrasonic의 chracter Device Driver init
        // 아래의 코드를 작성하시오      //COMP
        // device number registration
        // *us_cdev = cdev_alloc(); //substituted by cdev_init();
        // if us_cdev declared to pointer, have to use cdev_alloc();
        result = us_register_cdev(); //complete
        if(result <0)
        {
                printk(KERN_ALERT "< Ultrasonic Register Fail > \n");
                return result;
        }

        // Ultrasonic 제어에 사용할 GPIO init
        // 아래의 코드를 작성하시오      //
        res = gpio_init(); //complete but have to check
        if(res < 0 )
                return -1;

        return 0;
}

/* Function : us_exit */
/* module_exit 함수의 파라미터로 사용되는, rmmod 시 실행되는 함수 */
static void us_exit(void)
{
        printk(KERN_ALERT "< Ultrasonic Module is down > \n");
        // GPIO interrupt release
        free_irq(irq, NULL);
        // GPIO release
        gpio_free(ULTRASONIC_TRIGGER);
        gpio_free(ULTRASONIC_ECHO);
        // Chracter device driver configuration 제거
        cdev_del(&us_cdev);
        // Character device driver release
        unregister_chrdev_region(us_dev,1);
}
```

-> insmod 명령어를 통해 실행되는 init 함수에서는 character device driver 와 gpio 를 초기화 및 설정을 실행하게 된다

-> rmmod 명령어를 통해 실행되는 exit 함수에서는 interrupt, character device driver, gpio 를 해제하여 사용을 중단하게 한다.

```
/* Function : gpio_init */
/* IMX6Q chipset의 GPIO Configuration을 진행하는 함수 */
int gpio_init(void)
{
                int rtc;

        // TRIGGER GPIO pin init
        // 아래의 코드를 작성하시오      //COMP
        rtc = gpio_request(ULTRASONIC_TRIGGER,"us_tr");
    if (rtc!=0) {
                                printk(KERN_ALERT "<Trigger Pin Request Fail>\n");
                goto fail;
    }

        // ECHO GPIO pin init
        // 아래의 코드를 작성하시오      //COMP
        rtc = gpio_request(ULTRASONIC_ECHO,"us_ec");
    if (rtc!=0) {
                                printk(KERN_ALERT "<Echo Pin Request Fail>\n");
                goto fail;
    }

        // TRIGGER GPIO pin direction init //trigger -> out
        // 아래의 코드를 작성하시오      //COMP
        rtc = gpio_direction_output(ULTRASONIC_TRIGGER,0);
    if (rtc!=0) {
                                printk(KERN_ALERT "<Trigger pin Setting Fail>\n");
                goto fail;
    }

        // ECHO GPIO pin direction init //echo -> in
        // 아래의 코드를 작성하시오      //COMP
        rtc = gpio_direction_input(ULTRASONIC_ECHO);
    if (rtc!=0) {
                                printk(KERN_ALERT "<Echo Pin Setting Fail>\n");
                goto fail;
    }

  // ECHO GPIO pin Interrupt init       //int gpio_to_irq(unsigned int gpio)
        // 아래의 코드를 작성하시오      //COMP
        rtc = gpio_to_irq(ULTRASONIC_ECHO);
                if (rtc<0) {
                                printk(KERN_ALERT "<irq Pin GPIO Request Fail>\n");
                        goto fail;
                } else {
                        irq=rtc;
                }

        // ECHO GPIO pin Interrupt 시 작동하는 함수 및 옵션 설정
        // Interrupt flag : IRQF_TRIGGER_RISING , IRQF_TRIGGER_FALLING , IRQF_DISABLED 사용
        // -- IRQ_DISABLED : 인터럽트 핸들러를 실행하는 동안 모든 인터럽트 비활성화
        // -- IRQF_TRIGGER_RISING : 신호가 상승할 때 인터럽트 인지.
        // -- IRQF_TRIGGER_FALLING : 신호가 하강할 때 인터럽트 인지.
        // 아래의 코드를 작성하시오      //COMP but have to check
        rtc = request_irq(irq, (void*) ultrasonics_echo_interrupt,IRQF_TRIGGER_RISING|IRQF_TRIGGER_FAL
LING|IRQF_DISABLED,"us",NULL);
          if(rtc) {
                printk(KERN_ALERT "<irq Register Fail>\n");
                goto fail;
          }

    printk(KERN_INFO "Ultrasonic device Enable\n");
                gpio_set_value(ULTRASONIC_TRIGGER,0);
                return 0;

                fail:
                        return -1;
}
```

-> GPIO Pin 에 대한 초기화를 진행, ECHO PIN(GPIO2_3),TRIGGER PIN(GPIO2_2)의 사용을 요청, 신호의 방향 output, input 을 설정, 그리고 echo 값을 위한 interrupt 사용 설정을 수행한다.

```c
/* Function : us_register_cdev */
/* Ultrasonic에 사용할 chracter device driver 초기화 및 설정하는 init 함수 */
static int us_register_cdev(void)       //=>COMPLETE
{
        int error;

        // Chracter device driver 초기화
        if(us_major) {
                // major #이 존재할 때, device driver 초기화
                // 아래의 코드를 작성하시오      //COMP
                us_dev = MKDEV(us_major,us_minor);
                error = register_chrdev_region(us_dev,1,"us");
        }else{
                // major #이 존재하지 않을 때, device driver 초기화
                // 아래의 코드를 작성하시오      //COMP
                error = alloc_chrdev_region(&us_dev,us_minor,1,"us");
                us_major = MAJOR(us_dev);//get allocated major#
                //us_major = MAJOR(&us_dev); //=>have to check!
        }

        if(error<0) {
                printk(KERN_WARNING "us: can't get major %d\n", us_major);
                return result;
        }

        printk(KERN_ALERT "major number = %d\n", us_major);

        // Chracter deivce driver configuration의 File file_operations 설정
        // 아래의 코드를 작성하시오              //COMP

        cdev_init(&us_cdev,&us_fops);

        // Character device driver owner 및 file_operations 등록
        us_cdev.owner = THIS_MODULE;
        us_cdev.ops = &us_fops;
        // Character device driver에 chracter device driver configuration 등록
        // 아래의 코드를 작성하시오              //COMP

        error = cdev_add(&us_cdev,us_dev,1);

        if(error)
        {
                printk(KERN_NOTICE "us Register Error %d\n", error);
        }
        return 0;
}
// Module을 insmod 명령을 통해 Kernel 상에 등록 시켰을 경우 실행되는 함수
module_init(us_init);
// Module을 rmmod 명령을 통해 Kernel 상에서 삭제 시켰을 경우 실행되는 함수
module_exit(us_exit);
```

-> register_cdev 는 character device driver 를 초기화 및 설정하는 함수이며 major,minor number 가 있으면 해당 major number 로 device 를 초기화하고 등록, 없으면 새로 할당해서 등록한다. 그리고 fops 정보를 설정하여 연결시킨다.

3) 각각의 Makefile

```
obj-m += pir_driver.o

KDIR := /work/achroimx6q/achroimx_kernel

all:
        make -C $(KDIR) SUBDIRS=$(PWD) modules

app:
        arm-none-linux-gnueabi-gcc pir_app.c -o pir_test -static
install:
        cp -a *.ko /nfsroot/
        cp -a *test /nfsroot/
clean:
        rm -rf *.ko
        rm -rf *.mod*
        rm -rf .*.cmd
        rm -rf *.o
        rm -rf Module.symvers
        rm -rf modules.order
        rm -rf pir_test
```

```
obj-m += us_driver.o

KDIR := /work/achroimx6q/achroimx_kernel

all:
        make -C $(KDIR) SUBDIRS=$(PWD) modules

app:
        arm-none-linux-gnueabi-gcc us_app.c -o us_test -static
install:
        cp -a *.ko /nfsroot/
        cp -a *test /nfsroot/
clean:
        rm -rf *.ko
        rm -rf *.mod*
        rm -rf .*.cmd
        rm -rf *.o
        rm -rf Module.symvers
        rm -rf modules.order
        rm -rf us_test
```

<좌: PIR 센서 Makefile, 우: US 센서 Makefile>

-> 위의 Makefile 은 기본 제공 예시인 DVD-1_SRC/examples/linux/gpio_driver 의 예시의 Makefile 을 필요에 맞게 약간 바꾼 것이다.

2) Client (HOST PC)

-1) client.c

```
#include "inet.h"
#define MAXLINE 512

int writen(register int fd, register char *ptr, register int nbytes);
int readline(register int fd, register char *ptr, register int maxlen);
void str_cli(register FILE *fp, register int sockfd);

int main(int argc,char *argv[]){
        int sockfd;
        struct sockaddr_in serv_addr;
        pname = argv[0];

        bzero((char *) &serv_addr, sizeof(serv_addr));
        serv_addr.sin_family = AF_INET;
        serv_addr.sin_addr.s_addr = inet_addr(SERV_HOST_ADDR);
        serv_addr.sin_port = htons(SERV_TCP_PORT);
        //printf("open socket...\n");
        if((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
            printf("client : can't open stream socket\n");
        //printf("complete! connect...\n");
        connect(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr));
//          printf("client : can't connect to server\n");
//          exit(0);

        printf("complete!\n");
        str_cli(stdin,sockfd);

        close(sockfd);
        exit(0);
}
```

->main 에서는 socket 생성 및 연결을 진행하고 데이터를 받아서 출력하는 것은 str_cli 함수에 구현하였다.

```
//////////////////////////////////
void str_cli(register FILE *fp,register int sockfd){
        int n;
        time_t current_time;
        struct tm tm;
        char sendline[MAXLINE], recvline[MAXLINE + 1];
        while(1){
            current_time = time(NULL);
            tm = *localtime(&current_time);
            printf("Time : %d/%d/%d %d:%d:%d\n", tm.tm_year +1900, tm.tm_mon + 1, tm.tm_mday,tm.tm_hou
r,tm.tm_min, tm.tm_sec);
            n = readline(sockfd, recvline, MAXLINE);
            if(n < 0)
                printf("str_cli : reading error\n");
            recvline[n] = 0;
            fputs(recvline, stdout);
            if(!strcmp(recvline,"QUIT\n"))
                break;
        }
        if(ferror(fp))
            printf("str_cli : error reading file\n");
}
//////////////////////////////////////////////
int writen(register int fd, register char *ptr, register int nbytes){
        int nleft, nwritten;
        nleft = nbytes;
        while(nleft > 0) {
            nwritten = write(fd, ptr, nleft);
            if(nwritten <- 0)
                return nwritten;
            nleft -= nwritten;
            ptr += nwritten;
        }
        return nbytes-nleft;
}
//////////////////////////////////////////////
int readline(register int fd, register char *ptr, register int maxlen){
        int n, rc;
        char c;
        for(n=1; n < maxlen; n++){
            if((rc = read(fd, &c, 1)) == 1){
                *ptr++ = c;
                if(c== '\n') break;
            }
            else if(rc == 0){
                if(n == 1) return 0;
                else break;
            }
            else
                return -1;
        }
        *ptr = 0;
        return n;
}
```

-> str_cli 함수에서 연결된 socket 을 받아서 해당 socket 으로부터 계속 readline 을 통해 서버로부터 값을 읽어오고 localtime 을 통해 현재 시간 TimeStamp 를 출력하게 한다. 이 코드는 echo 통신의 client 함수를 참고해서 작성되었다.

=> 문제점은 코드를 종료하고 싶으면 Ctrl+C 를 통해서 강제로 탈출해야 한다는 점과 socket 이 연결되지 않은(즉 서버가 닫혀있는 경우에는 오동작이 발생한다는 문제가 있다. 이는 개선해야 할 부분이다.

-> 그리고 기존의 read 는 1 개 character 밖에 안 읽어와서 필요한 문장단위로 읽기 위해 readline 함수를 정의하였다.

-2) inet.h

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <time.h>

#define SERV_TCP_PORT 25555
#define SERV_HOST_ADDR "192.168.10.125"
//#define SERV_HOST_ADDR "127.0.0.1"
char *pname;
```

<client 의 inet.h>

-> client.c 에서 필요한 모든 헤더파일을 include 하고 접근할 서버의 Port 번호, 주소(192.168.10.125->보드의 주소)을 정의하였다.

-3) Makefile

```
CC = gcc
LIB = -lsocket -lnsl

all : client

*.o : *.c
        $(CC) -I. -g -c $<

client : client.o
        $(CC) -g -o CLI client.o
clean :
        rm -f *.o CLI
```

<Client 의 Makefile>

-> 간단하게 gcc 동작을 make 명령어로 수행할 수 있도록 하는 Makefile 이다.

3) Server (ACHRO-I MX.6Q)

-1) server.c

=> 8 주차의 OpenCV V4L 라이브러리를 이용한 카메라 기동 프로그램 코드 사용

```
#include "inet.h"

#include "opencv2/video/tracking.hpp"
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc_c.h"

#include "v4l_wrapper.h"
#define MAXLINE 512

//prototype declaration//
int readline(register int fd, register char *ptr, register int maxlen);
void str_echo(int sockfd);

//main function//
int main(int argc,char *argv[]){
        int sockfd, newsockfd, clilen, childpid;
        struct sockaddr_in cli_addr, serv_addr;
        pname = argv[0];
        if((sockfd = socket(AF_INET, SOCK_STREAM, 0))<0)
            printf("server : can't open stream socket\n");
        bzero((char *) &serv_addr, sizeof(serv_addr));
        serv_addr.sin_family = AF_INET;
        serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
        serv_addr.sin_port = htons(SERV_TCP_PORT);

        if(bind(sockfd,(struct sockaddr *) &serv_addr, sizeof(serv_addr))<0){
            printf("server : can't bind local address\n");
            exit(-1);
        }
        printf("waiting...\n");
        listen(sockfd, 5);
        for(;;){
            clilen = sizeof(cli_addr);
            printf("open client socket\n");
            newsockfd = accept(sockfd,(struct sockaddr *)&cli_addr, &clilen);
            if(newsockfd<0){
                printf("server : accept error\n");
                exit(-1);
            }
            if((childpid = fork()) <0){
                printf("server : fork error\n");
                exit(-1);
            }
            else if(childpid == 0){
                close(sockfd);
                str_echo(newsockfd);
                close(newsockfd);
                exit(0);
            }
```

->main 에서는 socket 생성 및 클라이언트 연결 대기 및 연결을 진행하고 센서 값을 읽고 판단하여 LCD 와 카메라를 사용, 클라이언트에 값을 전달하는 것은 str_echo 함수에 구현하였다.

```
//string echo function//
void str_echo(int sockfd){
        int n;
        char line[MAXLINE];
////////////////////////////////////////////////////////
        //for camera//
        fsl_v4l_cap mycamera;
        fsl_v4l_out mydisplay;
        int ret;
        char *buffer;
        int width = 640;
        int height = 480;
        IplImage *image;
////////////////////////////////////////////////////////
        int i;
        int dev_1;
        int dev_2;
        char buf[10] = {0};
        char buf_pir[10] = {0};
        int retn;
        //for current time//
        const time_t tim = time(NULL);
        struct tm *cur_time;
        cur_time = localtime(&tim);
        //board time check
        printf("current time %d:%d:%d\n",cur_time->tm_hour,cur_time->tm_min,cur_time->tm_sec);
        //device open
        dev_1 = open("/dev/us", O_RDWR);        //ULTRASOONIC DEVICE
        dev_2 = open("/dev/pir", O_RDWR);       //PIR SENSOR
        if (dev_1<0||dev_2<0){
                printf("Device Open Error\n");
                close(dev_1);
                close(dev_2);
                return ;
        }

        printf("GET COMMAND QUIT will off. \n");
```

-> 맨 위의 block 은 기본 통신용으로 사용할 변수, 중간 블록은 카메라와 LCD 를 위한 변수, 마지막 블록은 센서 값을 위한 변수이며 보드에 설정된 시간을 확인하고 dev_1 에 ultrasonic, dev_2 에 pir 을 open 한다.

```
////////////////////////////////////////////////////////
       //camera setting//
       printf ("\n\nInitialzing Camera Device: Video0 (640x480)...");

       ret = V4LWrapper_CreateCameraCapture (&mycamera, "/dev/video0", width, height);

       if (ret == V4LWrapper_SUCCESS)
       {
               printf ("OK\n");
       }

       else
       {
               printf ("\nDevice not found, make sure the driver are properly installed:");
               printf ("\nov5642_camera.ko, ov5640_camera_mipi.ko and mxc_v4l2_capture.ko\n");
               exit (0);
       }

       printf ("\nInitialzing Display Device: video17, fb0 (640x480)...");
       ret = V4LWrapper_CreateOutputDisplay (&mydisplay, "/dev/fb0", NULL, width, height);
       if (ret == V4LWrapper_SUCCESS)
               printf ("OK\n");
       else
       {
               V4LWrapper_CloseCameraCapture (&mycamera);
               exit (0);
       }

       printf ("\nAllocating data buffer...");
       buffer = (char *) malloc (mycamera.g_frame_size);

       if (buffer)
               printf ("OK\n");
       else
       {
               V4LWrapper_CloseCameraCapture (&mycamera);
               V4LWrapper_CloseOutputDisplay (&mydisplay);
               exit (0);
       }


       image = cvCreateImage (cvSize (width, height), IPL_DEPTH_8U, 3);
       IplImage *gray = cvCreateImage (cvSize (width, height), IPL_DEPTH_8U, 1);
```

-> V4L 라이브러리를 이용해서 CameraCapture 와 OutputDisplay 를 설정 및 buffer 공간을 할당.

```
////////////////////////////////////////////////////
        for(i=0;i<1000000;i++){
            memset(buf,0,sizeof(buf));
            read(dev_1,buf,3);
            //TEST
            retn = atoi(buf);
            if(retn <0)// -value is error bit -> skip
                retn = VALUE_OF_DISTANCE;
            if((retn<(VALUE_OF_DISTANCE-VALUE_MARGIN))||
                ((VALUE_OF_DISTANCE+VALUE_MARGIN)<retn)){
                V4LWrapper_QueryFrame (&mycamera,buffer);
                V4LWrapper_CvtColor (buffer, image->imageData, width, height, YUV422toRGB888);
                V4LWrapper_OutputDisplay (&mydisplay, buffer);
                read(dev_2,buf_pir,10);
                if(buf_pir[0] == '1'){//=> send_alart
                    printf("INVADED\n");
                    if(write(sockfd,"SOMEONE INVADED\n",16)!= 16)
                        printf("str_echo: writen error\n");
                }
                else{ //-> ultrasensor value back to normal -> escape
                    printf("NO_MOVING,BUT SOMETHING WEIRD\n");
                    if(write(sockfd,"MOVING STOPPED\n",15)!= 15)
                        printf("str_echo: writen error\n");
                }
            }
            usleep(20000);
        }
        close(dev_1);
        close(dev_2);
        V4LWrapper_CloseCameraCapture (&mycamera);
        V4LWrapper_CloseOutputDisplay (&mydisplay);
        free(buffer);
        cvReleaseImage(&image);
}
```

-> 본격적인 동작이 시작되는 부분이며 맨 처음 ultrasonic 으로부터 받을 buffer 를 비우고 buf 에 값을 읽어 들이는데 3 자리까지 읽어 들이도록 하였다. 이유는 3 자리부터 (100 이상)부터 정확도가 급격하게 낮아지기 때문이다. (판단 근거는 4 주차와 6 주차때 실험을 통해 ultrasonic 을 사용해봤을 때 주로 100cm(1m)가 넘어가면 값에 오차가 커졌기 때문이다.)

-> 그리고 ultrasensor 로부터 들어오는 값은 char 형의 배열로 들어오므로 이를 비교하기 위해서는 integer 값으로 변환해야 한다. 따라서 atoi 함수를 사용하여 int 형 변수 retn 으로 바꿨다. 그리고 이 값이 가끔 -값으로 나오는 경우가 있었는데 이는 주로 오류 비트였기 때문에 이를 걸러내도록 하였다. 그리고 센서 값이 거리 기준값(VALUE_OF_DISTANCE,90 으로 설정) +- 거리 오차 범위 (VALUE_MARGIN, 10%인 9 로 설정)를 벗어나게 되면 침입자가 발생했거나 이상현상이 발생했음으로 판단, V4LWrapper 를 통해 카메라를 실행하고 이를 LCD 에 출력하게 하였다. 그리고 인체감지센서의 값을 읽어서 움직임이 있다면 침입했다는 표시를, 값이 0 으로 멈추거나 다른 물체인 경우에는 움직임이 없지만 이상하다는 표시를 CLIENT(HOST)에게 보내게 된다. 그리고 ultrasonic 의 동작속도가 매우 빨라서 값의 오차가 많이 났었는데 이를 해결하기위해 20ms 동안 sleep 상태로 두어서 안정화하였다.

-2) inet.h

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <signal.h>
#include <time.h>

#include <assert.h>
#include <sys/time.h>
#include <math.h>
#include <malloc.h>

#define SERV_TCP_PORT 25555
#define SERV_HOST_ADDR "127.0.0.1"
//configuration
#define VALUE_OF_DISTANCE 90
#define VALUE_MARGIN 9

char *pname;
```

<server 의 inet.h 파일>

Server.c 에서 필요한 모든 헤더파일을 include 하고 서버를 열 Port 번호, 주소(LOCALHOST), 그리고 기본 설정 거리 값과 오차범위를 나타낼 MARGIN 을 정의하였다.

-3) Makefile

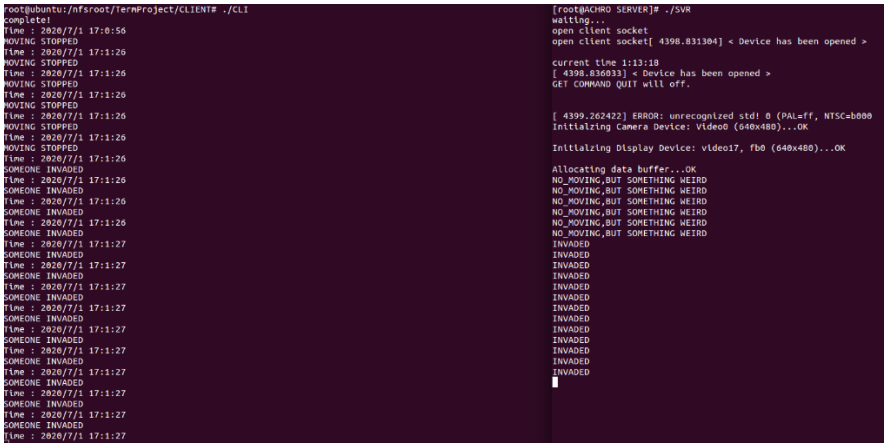
```
#before buildint it don't forget to export ROOTF_DIR and CROSS_COMPILER

APPNAME                 = SVR
TOOLCHAIN               = /opt/toolchains/arm-2014.05
CROSS_COMPILER          = $(TOOLCHAIN)/bin/arm-none-linux-gnueabi-
EXTERNAL_INCLUDE        = /work/achroimx6q/achroimx_kernel/include
CC                      = $(CROSS_COMPILE)g++
DEL_FILE                = rm -rf
CP_FILE                 = cp -rf

TARGET_PATH_LIB         = $(ROOTFS_DIR)/work/achroimx6q/achroimx6q_opencv/build/lib
TARGET_PATH_INCLUDE     = $(ROOTFS_DIR)/usr/arm-linux-gnueabihf/include

CFLAGS                  = -DLINUX -DUSE_SOC_MX6 -Wall -O2 -fpermissive -mfloat-abi=softfp\
                          -DEGL_API_FB -DGPU_TYPE_VIV -DGL_GLEXT_PROTOTYPES -DENABLE_GPU_RENDER_20 \
                          -I../include -I$(ROOTFS_DIR)/usr/src/linux/include -I$(TARGET_PATH_INCLUDE)

LFLAGS                  = -Wl,--library-path=$(TARGET_PATH_LIB),-rpath-link=$(TARGET_PATH_LIB) -lm  -l
pthread \
                           -lopencv_core -lopencv_imgproc -lopencv_highgui -lopencv_ml -lopencv_imgcod
ecs

OBJECTS                 = server.o v4l_wrapper.o

first: all

all: $(APPNAME)

$(APPNAME): $(OBJECTS)
        $(CC) -I $(EXTERNAL_INCLUDE) $(LFLAGS) -o $(APPNAME) $(OBJECTS)

server.o: server.c
        $(CC) -I $(EXTERNAL_INCLUDE) $(CFLAGS) -c -o server.o server.c `pkg-config opencv --cflags --l
ibs`

v4l_wrapper.o: v4l_wrapper.c
        $(CC) -I $(EXTERNAL_INCLUDE) $(CFLAGS) -c -o v4l_wrapper.o v4l_wrapper.c

clean:
        $(DEL_FILE) $(OBJECTS)
        $(DEL_FILE) *~ *.core
        $(DEL_FILE) $(APPNAME)

distclean: clean
        $(DEL_FILE) $(APPNAME)

install: all
        $(DEL_FILE) $(APPNAME)
```

<server 의 Makefile>

-> 위의 Makefile 은 8 주차 카메라 동작 확인 때의 V4L Library 관련 예제(week10)의 camera_test 의 Makefile 에 APPNAME 을 SVR 로 OBJECT 에 server.o 로 바꾸고 추가된 부분을 약간 바꿔서 사용하였다.

10. 실행 결과

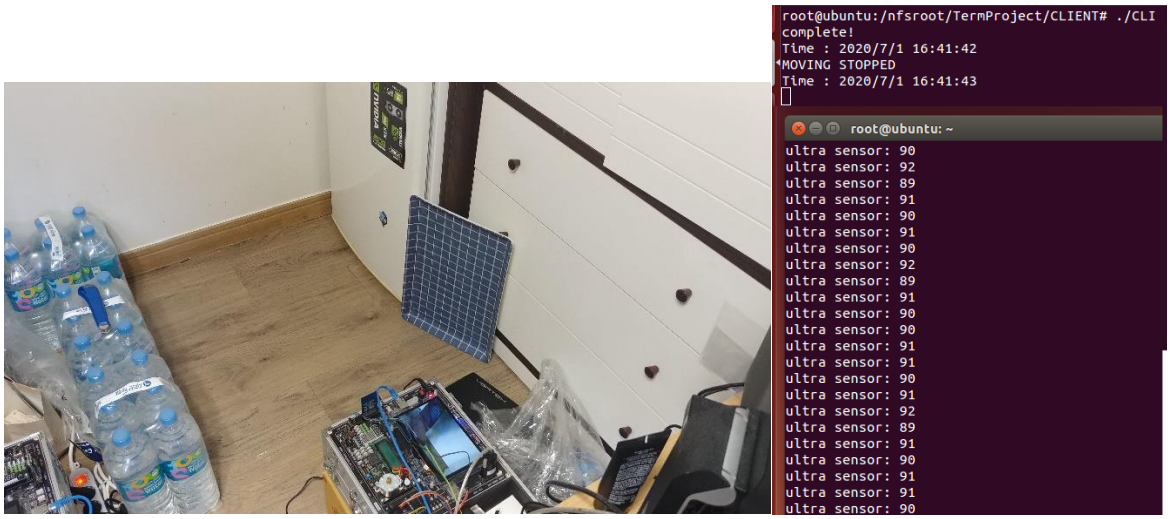
<최종 동작>



<좌: CLIENT(HOST) 우: SERVER(BOARD)>

-> 초음파 감지 센서의 값은 기준 값을 90으로 설정하고 센서의 품질을 고려하여 오차범위를 10%로 설정하였다. (이는 구현 과정에서 해당 위치에 설치하였을 때 나왔던 값이 96~ 84를 왔다 갔다 했기 때문) 그리고 중간에 센서의 속도와 print의 속도가 맞지 않아서 이상한 값(음수 값, 정상범위를 한참 넘는 값)이 나오는 것 역시 걸러내었다.

<1. 정상 상태의 동작>

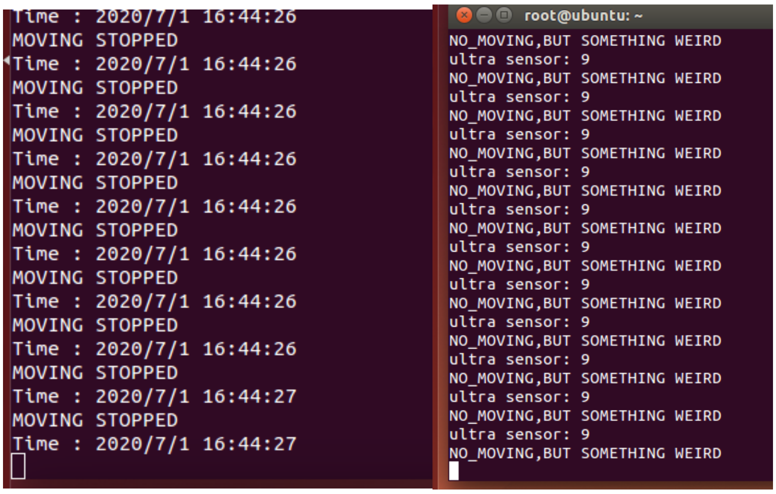


<좌: 보통 상태의 사진, 우 상단: CLIENT(HOST), 우 하단: SERVER(BOARD)

-> 아무것도 없어서 클라이언트에는 아무것도 보내지 않으며 캡처한 화면은 거리를 확인하기 위해 BOARD에 출력시켜서 거리 값을 확인하였다. 이는 이후 초음파 센서의 값을 출력하지 않았을 때 클라이언트와 서버 모두 아무것도 표시 안되는 것을 확인할 수 있었다.

<2. 물건이 놓여있는 경우 or 사람이 움직임이 없는 경우>

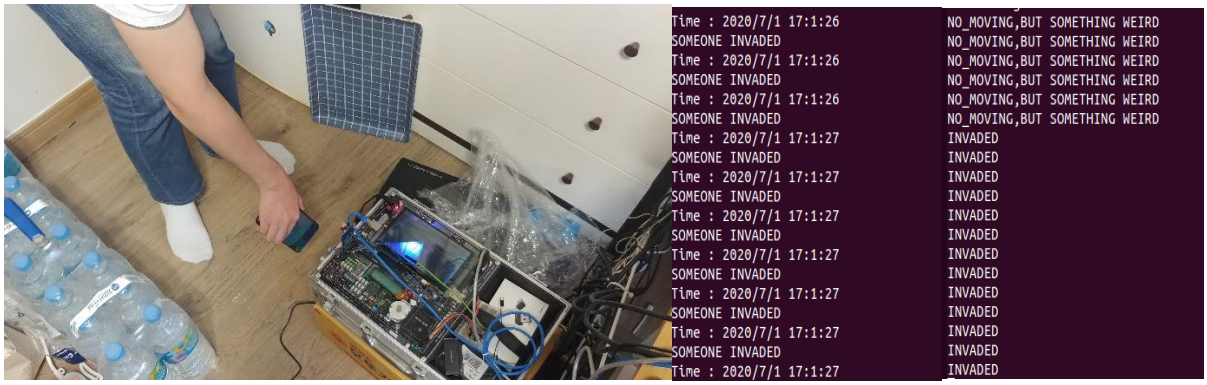
<물병이 센서 앞에 놓여있는 사진>



<좌: CLIENT(HOST), 우: SERVER(BOARD)>

-> 물병은 체온이 36.5도 이상이 아니어서 인체 감지 센서에는 인식이 안되지만 초음파 센서에는 인식이 되므로 카메라는 동작을 하되 클라이언트(HOST)에는 움직임이 멈춰있는 정보를 전달하고 Timestamp와 함께 출력한다.

<3. 사람이 침입한 경우>



<좌: 사람이 센서 앞에 서있는 사진, 중: CLIENT(HOST), 우: SERVER(BOARD)

-> 사람이 카메라 앞에 있으며 위의 사진은 휴대폰을 초음파 센서에 대고 있었으므로 거리가 가까워짐을 감지하여 카메라가 따라 동작하게 되었고 인체 감지 센서가 감지하여 서버(BOARD)와 클라이언트(HOST) 모두 침입여부를 출력하게 하였다. 특히 CLIENT에서는 해당 Timestamp와 함께 출력하였다.

위의 3가지 경우는 화면(Terminal)에 출력하게 하였지만 이는 linux상에서 실행할 때 (> 출력파일)로 나타내면 로그파일로 남길 수도 있다.

이렇게 원하는 모든 동작을 구현할 수 있었다.

11. 느낀 점 및 보완할 점

이번 프로젝트를 진행하면서 배울 수 있었던 것은 지난 학기(3학년 2학기)에 배웠던 임베디드 시스템 과목을 통해서 Cross Compile과 uC/OS 프로그래밍, 임베디드 시스템 설계 및 실험 과목을 통해서 Cross Compile 및 OS없는 임베디드 시스템 프로그래밍을 배운 것과 연결되어 이번 ACHRO-I .MX6Q 보드에 Linux Toolchain을 올려서 Serial통신과 NFS를 배우고 linux상에서 GPIO, FPGA, Bluetooth, WiFi, QT Library, LCD, OpenCV, CAMERA를 어떻게 다루는 지를 배울 수 있었다.

처음에 실험을 따라갈 때는 배웠던 것을 기반으로 실습을 하다가 개발 환경의 차이, 버전 차이 등등의 각종 어려움이 많았다. 이로 인해서 실습을 할 때 의욕도 많이 잃었었고 힘들었다.

특히 이번 학기는 COVID-19로 인한 비대면 실험 실습을 진행하여서 즉각적인 피드백이 어려워서 더더욱 진행이 안되었었다. 하지만 조교님들의 많은 도움을 통해 마지막 프로젝트까지 마무리할 수 있어서 다행으로 생각한다.

무엇보다 프로젝트를 진행하면서 점점 진행되어가는 과정을 확인할 때마다 소소한 성취감을 느낄 수 있어서 좋았다.

그리고 이번 프로젝트에서 아쉬웠던 점, 즉 보완할 점을 생각해 보면 먼저 센서 값이 너무 일정하지가 않고 인체 감지 센서의 동작 속도와 방식은 생각보다 느렸다. 이로 인해 더 좋은 성능은 낼 수 없었다.

그리고 인체 감지 센서의 구현이 가장 마지막으로 진행되어 더 완벽한 목표인 초음파 센서와 함께 인체 감지 센서의 값을 통해 카메라와 LCD, WiFi 통신의 동작이 진행될 수 있었지만 카메라 동작과 LCD는 초음파 센서 값을 기반으로, WiFi 통신은 초음파 센서 값 + 인체 감지 센서의 값을 기반으로 동작하게 되어서 너무 아쉬웠다.

이는 인체 감지 센서의 동작이 온도가 있는 물체가 움직임을 가질 때만 동작한다는 특성 때문에 어려웠던 것도 어느정도 있다.

그리고 센서의 값을 나타내는 속도(print)가 센서에서 값을 읽어오는 속도보다 한참 느려서 병목현상이 발생하여 임의로 동작에 sleep을 넣어 속도를 조절하여야 했던 것 역시 아쉬운 점이다.

그리고 무엇보다 같이 하던 조원이 어느 순간부터 연락이 두절되어서 혼자서 처음부터 모두 진행하게 되었던 것이 너무 아쉬웠다. (조원이 센서 드라이버를 만들기를 기다리다가 프로젝트 진행을 위해서 초음파 센서 역시 4주차에 만들었던 것을 사용하였다.)