개발완료보고서 [자유공모]

1. 팀 정보

팀명	G.I.JOE		
팀장	조윤호	팀원	이동규
팀원	오현정	팀원	전정호

2. 개발완료보고서

0. 근전도 신호를 이용한 재활용 운동(rehabilitation exercise)

1. 개요

1.1. 작품 개요

근전도 신호를 통하여 재활용 운동에 초점을 맞춰 개발한다.

활동성과 프로세스의 진행성을 높이기 위해 **1차측회로**(신체에 가까운 부분)와 **2차측회로**(컴퓨터 처리 부분)으로 나누어 설계하였다.

1차측 회로에서 근전도(ElectroMyoGram)센서를 이용하여 신체로부터 근전도 신호를 입력받는다. 때, High Pass Filter와 Low Pass Filter를 사용하여 noise가 매우 적고 증폭이 잘된 근전도를 1차측 회로에서 추출한다. 또한, 보행기의 손잡이에 자이로센서를 부착하여 재활용 운동 게임의 방향 이동을 제어한다. (재활용 운동이므로 격하게 좌우로 움직이는 것을 보행기 손잡이 키로 대체하였다) 이렇게 필터링된 근전도 신호와 자이로센서의 신호를 ATMega128 master 보드에서 입력받는다. 입력받은 신호를 변환하여 정제된 새로운 신호를 Bluetooth 모듈을 사용하여 2차측 회로에 전송한다.

2차측 회로에서는 Bluetooth로 수신한 각 키보드 키값의 신호들을 Arduino Leonardo 보드에 입력한다. 입력받은 신호는 보드를 통하여 키보드의 출력을 낸다. 이때 출력을 컴퓨터로 받으며 이 신호들이 키보드를 대신한다. 즉 Arduino Leonardo가 일종의 키보드를 대신하는 것이며 이때 컴퓨터에서 게임을 실행시키면 아두이노 키보드의 키대로 게임이 진행되는 것이다. 결과적으로 1차측 회로에서 정제한 근전도 신호 값에 따른 키보드 키가 게임에 쓰이게 된다.

1.2. 필요성

힘든 재활 치료에 활력과 재미를 줄 수 있으며 VR을 통해 치료의 폭을 넓힐 수 있다. 현재 프로젝트에 사용된 축구게임뿐만 아니라 다른 종목의 스포츠 게임 등 여러 게임이 폭넓게 사용될 수 있다. 게임 캐릭터의 한 움직임에 필요한 키와 사용자의 실제 움직임의 키를 동일하게 하여 근전도 입력을 받으므로 더 즐겁고 생생한 재활을 할 수 있다.

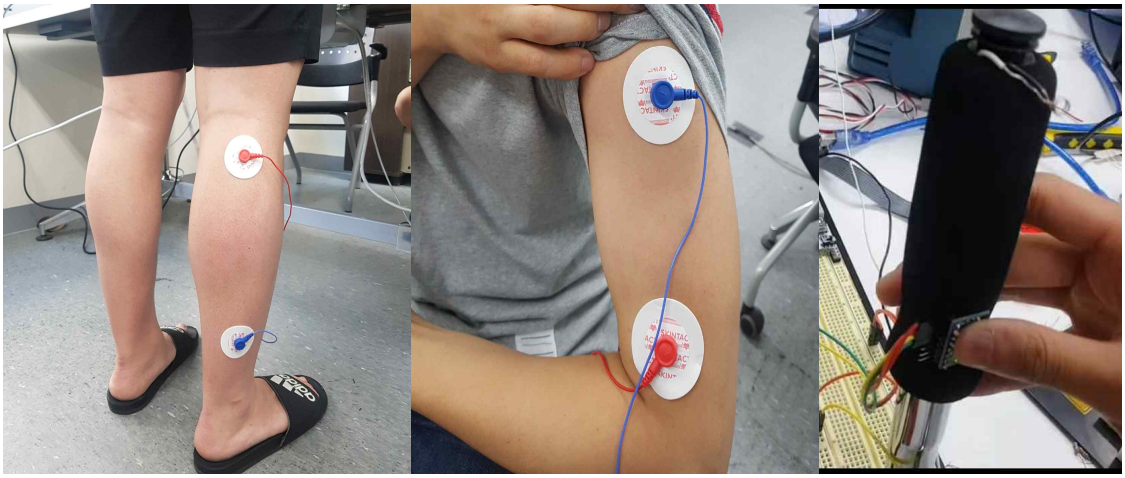
1.3. 개발 목표

근전도 신호를 이용하여 재활치료와 게임을 접목하여 작품을 이끌어낸다.

물리적 재활이 필요한 환자나, 가벼운 운동이 필요한 노인들에게 적합한 신체 운동게임을 설계한다. 필요한 부위의 근전도를 이용하여 게임을 진행하고, 그 최소한의 움직임을 게임 안에서 극대화시켜 재미 요소를 더하여 즐거운 재활 치료를 할 수 있도록 돕는 게임을 설계한다. 근전도 신호와 키보드를 연동하여 재활 환자들이 여러 온라인 게임을 즐김과 동시에 재활 효과까지 볼 수 있도록 한다.

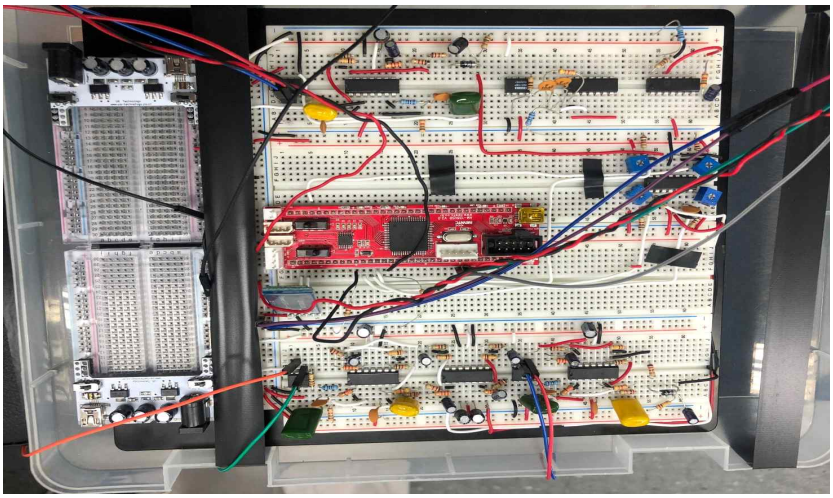
2. 작품 설명

2.1. HW 구성

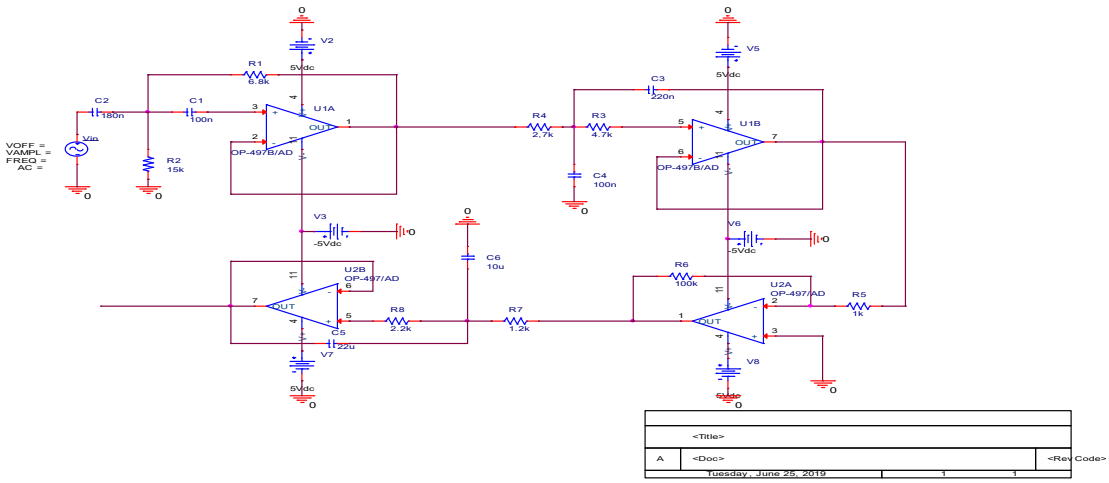


1단계 : 입력부 → 근전도 센서, 스냅전극, 스위치, 자이로센서

2단계 : MCU(Micro Controller Unit) → 회로도(INA118{Non inverting amplifier} → OP-AMP{high pass filter, low pass filter} → 아날로그 비교기 → ATmega128 → HC06(bluetooth-master)

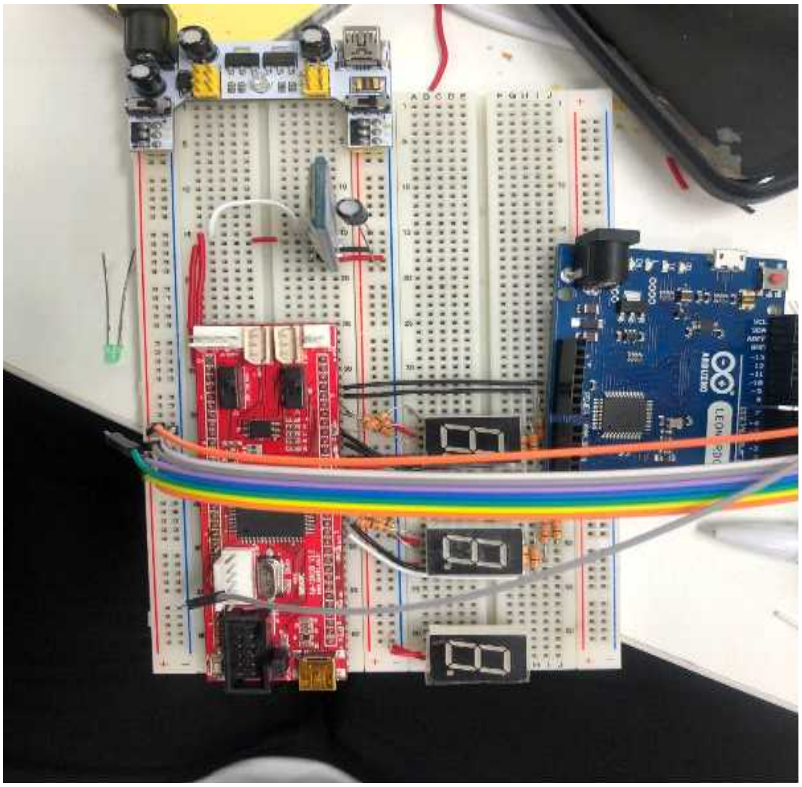


[1차측 회로 : 근전도 인식 및 Pass Filter]



[회로1의 OP-Amp 회로도]

3단계 : (출력부)→회로도 HC06(bluetooth-slave)→ATmega128→Arduino→Computer



[회로2]

2.2. HW 기능(제어 방법 등 서술)

1.근전도를 측정할 부위(팔)에 근전도 패치를 부착 → 스냅 전극을 통하여 INA118에 근전도 신호전달 → INA118에서 (Non Inverting Amplifier) 50배 신호 증폭 → High Pass Filter(100Hz) 적용 → Low Pass Filter(300Hz) 적용 → OP-AMP를 이용한 Non Inverting Amplifier 100배 증폭 → 아날로그 비교기를 통하여 사용자에 맞게 임의로 설정한 threshold의 전압과의 비교기 결과 출력 → ATMEGA128 master → ATMEGA128 slave (무선통신) → Arduino Leonardo Keyboard Interrupt → Computer → VR

2.근전도를 측정할 부위(다리)에 근전도 패치를 부착 → 스냅 전극을 통하여 INA118에 근전도 신호전달 → INA118에서 (Non Inverting Amplifier) 50배 신호 증폭 → High Pass Filter(100Hz) 적용 → Low Pass Filter(300Hz) 적용 → OP-AMP를 이용한 Non Inverting Amplifier 100배 증폭 →

{ 1. 걷기 키 입력 : 다이오드-캐패시터 충방전 회로 }

{ 2. 슛 /패스 키 입력 : Low Pass Filter (7Hz) 적용→ OP-AMP를 이용한 Non Inverting Amplifier 100배 증폭 → Low Pass Filter(7Hz) 적용 }

→ OP-AMP를 이용한 아날로그 비교기 → ATMEGA128 master → ATMEGA128 slave(무선통신) → Arduino Leonardo Keyboard Interrupt → Computer → VR

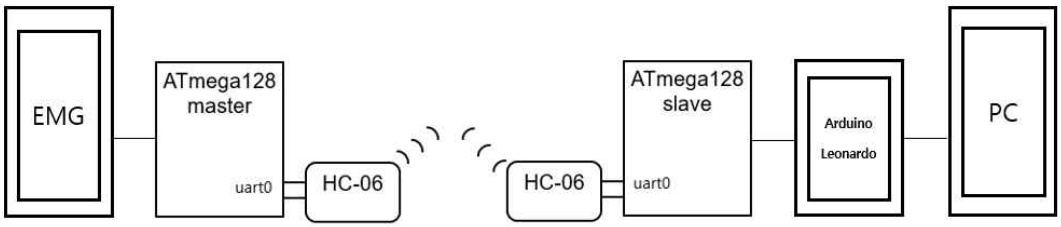
3. Switch, Gyro Sensor → ATMEGA128 master →ATMEGA128 slave (무선통신) → Arduino Leonardo Keyboard interrupt → Computer →VR

근전도(EMG), 스위치, 자이로센서를 통하여 각종 신호를 실시간으로 받아 분석하여 제어 후 컴퓨터에 입력하여 연동한다.

2.3. Software 구성

ATmega128(AVR studio 사용), Arduino Leonardo(sketch 사용)를 통하여 설계하였다.

2.4. Software 설계도



2.5. Software 기능

AVR을 통하여 ATmega에서 근전도 센서, 스위치, 자이로센서의 신호를 받아 분석한 후 블루투스로 다른 ATmega에 전송하여 연결된 Arduino를 통하여 PC와 통신한다.

2.6. 프로그램 사용법 (Interface)

근전도 패치를 각 부위에 부착 → 스냅 전극 연결 → 각 전원 연결 → 컴퓨터와 연동 후 축구게임{각종 프로그램(온라인게임, CD게임, VR, 치료프로그램)} 실행

각 행동에 맞게 작동 → 걷기-캐릭터 앞으로 걷기, 오른발로 슛- 캐릭터 슛, 왼손 들기- 캐릭터 패스 call, 왼발 패스- 캐릭터 패스, 자이로 센서로 R, L 방향-캐릭터 R, L 이동

2.7. 개발환경 (언어, Tool, 사용시스템 등)

개발 Tool은 ATMega 코딩에 사용된 AVR Studio와 Arduino 코딩에 사용된 Sketch이고 사용된 프로그래밍 언어의 기반은 C언어 기반이다.

3. 프로그램 설명

3.1. 파일 구성

3.2. 함수별 기능 (주석과 코드 전문 첨부)

ATmega128 master :

```
#include <stdlib.h>
#include <util/delay.h>
#include <compat/ina90.h>
#include <math.h>
#include <avr/io.h>
#include <string.h>
#include <avr/interrupt.h>
volatile typedef unsigned char byte;
volatile unsigned int ax=0, k=0,b;
volatile unsigned char buffer;
volatile byte MPU_6050_read(byte addr);
void MPU_6050_write(byte addr,unsigned char data);
void main_key();
void run();
```

```
void shoot();
void pass();
void passcall();
void only_direction();
int main (void) {

    DDRD = 0x00;
    DDRE = 0x02;
    DDRB = 0xF0;
    DDRF = 0xFE;
    DDRA = 0xFF;        //GPIO 세팅

    cli();

    TWCR = 0x04;
    TWSR = 0x00;
    TWBR = 0x12;        //TWI 세팅

    TIMSK = 0x04;
    TCCR1A =0x00;
    TCCR1B = 0x05;

    TCNT1H = 0xff;
    TCNT1L = 0xC0;        //Timer 세팅

    UCSR0B |= 0x08;
    UCSR0C |= 0x06;
    UBRR0H = 0x00;
    UBRR0L = 0x67;        //USART 세팅
    EICRA = 0x30;
    EIMSK = 0x04;        //Interrupt2 세팅
    MPU_6050_write(0x6B,0x00);    //sensor on
    MPU_6050_write(0x6C,0x00);
    sei();
    while(1){        //USART 전송 준비가 되면 반복문 시작
    if((PINF & 0x01) == 0x01){        //앞키 on

        if((ax>192)&(ax<224)) {        //왼키 on
        k = 1;    //direction flag
        main_key();
        }
        if((ax>32)&(ax<128)) {        //오른키 on
        k = 2;
        main_key();
        }
        else {
        k = 5;
        main_key();
        }
    }
    else{                        //앞키off
        if((ax>192)&(ax<224)) {        //왼키 on
```

```
        k = 3;
        main_key();
        }
        if((ax>32)&(ax<128)) {        //오른키 on
        k = 4;
        main_key();
        }
        else {
        k = 6;
        main_key();
        }
    }
    k=0;    //direction flag reset
    }
}

void MPU_6050_write(byte addr, unsigned char data){     //센서를 설정하는 함수
    _delay_us(10);
    TWCR = 0xA4;
    while(((TWCR & 0x80) == 0x00 || ((TWSR & 0xF8) != 0x08)));     //전송 & 신호대기
        TWDR = 0xD0;
        TWCR = 0x84;
    while(((TWCR & 0x80) == 0x00 || ((TWSR & 0xF8) != 0x18)));    //전송 & ACK 대기
        TWDR = addr;
        TWCR = 0x84;
    while(((TWCR & 0x80) == 0x00 || ((TWSR & 0xF8) != 0x28)));    //ACK
        TWDR = data;
        TWCR = 0x84;
    while(((TWCR & 0x80) == 0x00 || ((TWSR & 0xF8) != 0x28)));    //ACK
        TWCR = 0x94;
    _delay_us(10);
}
volatile byte MPU_6050_read(byte addr){        //센서의 값을 읽어오는 함수
    unsigned char data;
    _delay_us(100);
    TWCR = 0xA4;
    while(((TWCR & 0x80) == 0x00 || ((TWSR & 0xF8) != 0x08)));    //통신 & 신호대기
        TWDR = 0xD0;
        TWCR = 0x84;
    while(((TWCR & 0x80) == 0x00 || ((TWSR & 0xF8) != 0x18)));    //통신대기 & ACK
        TWDR = addr;
        TWCR = 0x84;
    while(((TWCR & 0x80) == 0x00 || ((TWSR & 0xF8) != 0x28)));
        TWCR = 0xA4;
    while(((TWCR & 0x80) == 0x00 || ((TWSR & 0xF8) != 0x10)));
        TWDR = 0xD1;
        TWCR = 0x84;
    while(((TWCR & 0x80) == 0x00 || ((TWSR & 0xF8) != 0x40)));
        TWCR = 0x84;
    while(((TWCR & 0x80) == 0x00 || ((TWSR & 0xF8) != 0x58)));
        data = TWDR;
        TWCR = 0x94;
```

```
    _delay_us(100);
    return data;    //data 값 return
}
ISR(SIG_OVERFLOW1){
    cli();
    TCNT1H = 0xFE;
    TCNT1L = 0x0B;                    //타이머 주기 설정
    buffer = MPU_6050_read(0x3B);    //가속도의 x축 high bit 값
    ax = (int)buffer;                //int형으로 변환 후 ax 변수 저장
    sei();
}
ISR(SIG_INTERRUPT2)
{
    cli();
    UDR0 = 0x80;
    sei();
}
void main_key(){
    if ((PINB & 0x01) == 0x01) run();            //PIN B의 0번 핀이 high이면 run 함수 실행
    if ((PINB & 0x02) == 0x02) shoot();           //PIN B의 1번 핀이 high이면 shoot 함수 실행
    if ((PINB & 0x04) == 0x04) pass();            //PIN B의 2번 핀이 high이면 pass 함수 실행
    if ((PINB & 0x08) == 0x08) passcall();        //PIN B의 3번 핀이 high이면 passcall 함수 실행
    else only_direction();                       //PINB의 입력이 없으면 only_direction 함수 실행
}
void run() {    //direction flag에 따른 값 전송 함수
    switch(k){
    case 1 :    UDR0 = 0x0B;    break;    //앞키 + 왼키 + 달리기
    case 2 :    UDR0 = 0x0D;    break;    //앞키 + 오른키 + 달리기
    case 3 :    UDR0 = 0x0A;    break;    //왼키 + 달리기
    case 4 :    UDR0 = 0x0C;    break;    //오른키 + 달리기
```

ATmega128 slave :

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

volatile unsigned char key;
void out(unsigned char key);
int main(){
    DDRE = 0x00;
    DDRB = 0xFF;        //GPIO 세팅
    cli();
    UCSR0A |= 0x00;
    UCSR0B |= 0x90;
    UCSR0C |= 0x06;
    UBRR0H = 0x00;
    UBRR0L = 0x08;        //USART 세팅
    sei();
    while(1){
    }
}
ISR(USART0_RX_vect){
    cli();
```

```
    key = UDR0;
    _delay_us(1);
    PORTB = key;
    sei();
}
```

Arduino:

```
#include <Keyboard.h>
void setup() {
Keyboard.begin();
}
void loop() { //코드를 무한반복합니다.
  int sensorValue_9U = digitalRead(9);
  int sensorValue_8L = digitalRead(8);
  int sensorValue_7R = digitalRead(7);
  int sensorValue_6RUN = digitalRead(6);
  int sensorValue_5SHOOT = digitalRead(5);
  int sensorValue_4PASS = digitalRead(4);
  int sensorValue_3CALL = digitalRead(3);
  int sensorValue_2SKIP = digitalRead(2);
  if(sensorValue_9U == 1){
   Keyboard.press(KEY_UP_ARROW);
   delay(10);
  }
  else if(sensorValue_9U != 1){
   Keyboard.release(KEY_UP_ARROW);
   delay(10);}
   if(sensorValue_8L == 1) {
   Keyboard.press(KEY_LEFT_ARROW);
   delay(10);

  }
  else if(sensorValue_8L != 1){
   Keyboard.release(KEY_LEFT_ARROW);
   delay(10);}
   if(sensorValue_7R == 1) {
   Keyboard.press(KEY_RIGHT_ARROW);
   delay(10);
  }
  else if(sensorValue_7R != 1){
   Keyboard.release(KEY_RIGHT_ARROW);
   delay(10);}
  if(sensorValue_6RUN == 0) {
   Keyboard.press('e');
  }
  else if(sensorValue_6RUN != 0){
   Keyboard.release('e');
   delay(10);}
  if(sensorValue_5SHOOT == 1) {
   Keyboard.write('d');
   delay(50);
   Keyboard.release('d');
```

```
  }
  else if(sensorValue_5SHOOT != 1){
   Keyboard.release('d');
   delay(10);}
  if(sensorValue_4PASS == 1) {
   Keyboard.press('s');
   delay(50);
   Keyboard.release('s');
  }
  else if(sensorValue_4PASS != 1){
   Keyboard.release('s');
   delay(10);}
  if(sensorValue_3CALL == 1) {
   Keyboard.write('c');
   delay(10);
   Keyboard.release('c');
   Keyboard.write('c');
   delay(10);
   Keyboard.release('c');
   Keyboard.write('c');
   delay(10);
   Keyboard.release('c');
  }
  else if(sensorValue_3CALL != 1){
   Keyboard.release('c');
   delay(10);}
  if(sensorValue_2SKIP == 1) {
   Keyboard.write('v');
   delay(50);
  }
  else if(sensorValue_2SKIP != 1){
   Keyboard.release('v');
   delay(10);}
}
```

3.3. 주요 함수의 흐름도

ATmega 1: 자이로센서 읽는 함수, 스위치, 아날로그 비교기에서 들어온 각종 신호 →캐릭터 키 반응 함수→블루투스 각종 신호 송신 함수

↓ ↓ ↓

Atmega 2: 각종 신호 수신 함수→ 각 PIN에 0,1 송출

↓ ↓ ↓

Arduino: 각 PIN에 맞게 키보드 KEY 입력 함수

3.4. 기술적 차별성

최대한의 간단한 소프트웨어 설계와 아날로그 비교기를 이용하여 설계하였다. 또한, 같은 부위의 근전도를 두 가지의 키로 구분하여 사용하여 기술적 차별성을 두었다.

키보드 입력을 제어하는 시스템과 운동 게임에 접목시켜 실제 모션이 필요한 근육과 입력 키가 매칭되어 게임을 통해 재활운동을 할 수 있다.

4. 개발 중 장애요인과 해결방안

1. 슛과 걷는 행동의 근전도를 모두 다리에서 측정하여야 하였으므로 다리를 움직이면 둘 다 측정이 되며 근전도 차이가 잘 드러나지 않아 어려움을 겪었다. 이에 따라 슛, 즉 다리에 순간적인 급격한 힘 변화를 더 상세히 측정하기위해 High Pass Filter, Low Pass Filter, Non Inverting Amplifier를 한 번 더 적용하여 다리에서 슛과 걷는 근전도의 차이를 얻어 낼 수 있었다.

2. 날씨, 사람의 상태 등등 많은 요소 때문에 개개인의 Threshold Voltage가 미세하게 달라져 결과값이 매일매일 달랐었다. 오차를 줄이기 위해 바로 ATmega128의 ADC로 가지 않고 아날로그 비교기에 가변저항을 사용하여 매일 변화하는 값에 현장 조치가 가능하게 되었으며 오차도 크지 않게 되었다.

3. 제품의 편의성을 위하여 각 Atmega128끼리 무선으로 연결시켰으며 그 과정 중 많은 어려움이 있었다. 그 중 통신 속도가 9600으로 맞지 않아 근전도 반응에 느린 반응을 보였으나 많은 실험을 통하여 통신 속도를 115200으로 변화한 후 빠른 반응속도를 얻을 수 있었다.

4. 근전도 측정 중 제일 큰 문제는 Noise로 인한 오차였다. 그 오차를 줄이기 위하여 일반 전선이 아닌 장착부분이 흔들리지 않고 장착할 수 있는 스냅전극을 사용하였으며 근전도 패치의 특정 젤을 발라 Noise를 줄이는데 성공하였다.

5. 전원을 가해주기 위하여 DC 5V Converter를 사용하였는데 –5V를 사용하는 데에 있어 문제점이 생겼지만 두 개의 Converter를 통하여 이를 해결하였다.

5. 개발결과물의 차별성

완성된 모듈을 사용하지 않고 개발에 임하였으며 하드웨어와 소프트웨어 간의 최대한 간략하게 이용할 수 있도록 조화시켰다. 또한 현재 의료쪽의 VR을 통한 치료가 많아짐에 따라 환자들에게 괴롭고 힘든 치료가 아닌 재미난 재활치료를 제공하는 것에 초점을 맞춰 다양한 완성돼있는 프로그램(온라인, 각종 게임,VR 프로그램 등)과 연동할 수 있도록 준비하였다.

6. 단계별 개발계획 및 실제 참여인원 및 업무 분장

근전도에 대한 각종 실험 및 연구 → 주마다 아이디어 인당 2개 제출 → 4주간 모인 아이디어 회의(실현 및 발전 가능성, 흥미, 목적) → 하드웨어(MCU1, MCU2) 설계 → 무선연결 및 Software 설계 → Arduino and Computer 연결 → VR과 연결

모든 실험, 개발 등 4인이 그 날의 주제 하나를 갖고 모두 서로 도와가며 참여하였으며 나눠야할 부분이 있으면 2인 1조를 통하여 실험하였다. 그중 맡은 부분의 주요한 역할은 다음과 같다.

전정호	조윤호	이동규	오현정
- MAIN : Analog Circuits - SUB : Subject	- MAIN : Analog Circuits - SUB : Arduino Code	- MAIN : Arduino Code - SUB : Atmega Code	- MAIN : Atmega Code - SUB : Gyro sensor