

창의SW기초설계 결과보고서

아두이노를 활용한 심장재활 보조장치 제작보고서

스마트기기 1분반 5조

팀 명: 주도준준

21011913 김도연, 22011891 이예준,

22011894 박주현, 22011902 이준수



목차

I. 서론

1. 추진배경 및 제작동기	3
2. 초기 구상안	4
3. 간트차트	5

II. 본론

1. 설계 조건과 주요 부품	
1) 아두이노 우노보드	5
2) 심박수 센서: sen0203 heart rate sensor	6
3) LCD: 1602 LCD	6
4) 푸쉬 버튼 스위치	8
5) 서보모터: Micro servo SG-90	9
6) 소형 스피커: FIT0502 Stereo Enclosed Speaker	10
7) mp3모듈: DFR0299 DFplayer mini	10
2. 부품구매내역	13
3. 구현 상세 내용	14
4. 회로도 및 외관	
1) 회로도 설계	16
2) 외관 설계	17
3) 작품 제작	18
5. 구현 코드	
1) healthcare.ino	21
2) set_age_intensity.ino	25
3) interrupt.ino	29
4) servo.ino	31
6. 성능시험 및 결과	
1) 시험 환경 설정	32
2) 성능 평가 기준	32
3) 시험 결과	34
7. 기대효과	38

III. 결론	39
---------------	----

IV. 참고문헌	40
----------------	----

I. 서론

1. 추진배경 및 제작동기

작년 22년 9월 통계청에서 발표된 2021년 사망원인 통계에 의하면 심장질환으로 인한 사망이 2위에 위치하고 있다. 특히나 전체 사망자 중 60대 이상의 비율이 85.7%, 40대 이상의 비율이 97.2%에 육박하는 와중 연령대별 사망원인을 분석한 통계에서 심장 질환으로 인한 사망자는 고령층으로 갈수록 점차 높아지고 있다. 이때의 심장질환은 대표적으로 협심증, 심근경색증, 심부전 등이 해당된다. 이러한 질병들은 수술, 약물치료 등 의학적 치료에 성공해도 재발률이 높다. 협심증, 심근경색증과 같은 관성동맥질환의 경우 환자 10명 중 약 5명은 재발을 겪는다. 또한 재발했을 시의 사망률은 68~85%에 이른다.

(단위: 인구 10만 명당 명)			
순위	사망원인	사망률	'20년 순위 대비
1	악성신생물(암)	161.1	-
2	심장 질환	61.5	-
3	폐렴	44.4	-
4	뇌혈관 질환	44.0	-
5	고의적 자해(자살)	26.0	-
6	당뇨병	17.5	-
7	알츠하이머병	15.6	-
8	간질환	13.9	-
9	패혈증	12.5	↑(+1)
10	고혈압성 질환	12.1	↓(-1)

그림 1 사망원인 순위 추이

■ 남녀전체 (단위: 인구10만 명당 명, 명)											
순위	40대		50대		60대		70대		80세 이상		
1	악성신생물	38.1 3,107	악성신생물	105.2 8,985	악성신생물	267.3 18,390	악성신생물	649.7 23,865	악성신생물	1,342.4 27,155	
2	고의적 자해	28.2 2,298	고의적 자해	30.1 2,569	심장 질환	51 3,510	심장 질환	171.4 6,297	심장 질환	917.1 18,562	
3	간 질환	11.6 948	심장 질환	23.5 2,010	뇌혈관 질환	40.1 2,757	뇌혈관 질환	141.1 5,182	폐렴	791.7 16,015	
4	심장 질환	10.0 812	간 질환	22.9 1,959	고의적 자해	28.4 1,951	폐렴	123.2 4,525	뇌혈관 질환	607.7 12,292	

그림 2 연령별 주요 사망원인

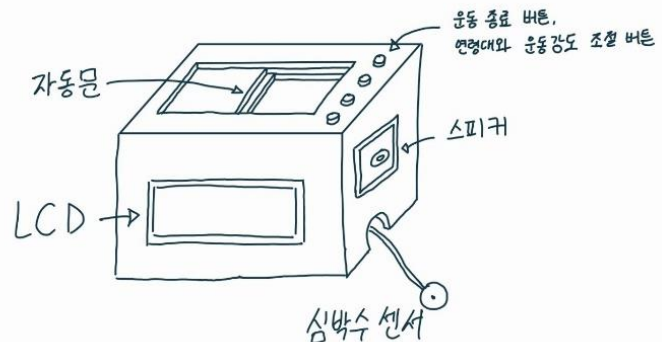
따라서 심장질환은 병원에서의 치료뿐만 아니라 사후 관리의 중요성이 두드러지는데 이때 사후 관리로써 실행되는 것이 심장재활 프로그램이다. 심장재활 프로그램은 심장질환으로 인해 약해진 심폐기능과 운동기능을 회복하고 위험인자를 제거하여 재발을 막기 위해 시행한다. 실제로 미국 메이요 클리닉에 따르면 심장재활에 참여한 환자는 그렇지 않은 환자보다 10년내의 사망률이 46% 감소했다고 밝혔다.

심장재활은 입원 치료 기간과 퇴원 후 통원 치료 기간동안 의료진의 감독 하에 이루어지는 프로그램을 병원 재활센터에서 진행하고 있다. 심장질환자들은 심장에 무리가 가지 않는 범위에서 적당한 강도의 운동을 진행하는 것이 중요하기 때문에 재활치료 전 운동부하검사를 진행하여 심장 운동기능을 평가받고 운동을 실시한다. 운동 중 심전도, 심박수, 혈압 등을 실시간으로 체크하며 적절한 강도의 운동을 진행한다. 재활센터에서의 심장재활 치료가 종료되면 가정에서의 운동을 진행한다. 이때는 심장의 기능이 상당히 정상화된 상태로 병원에서의 처방에 따라 운동하거나 연령에 따른 목표 심박수에 맞춰 강도를 조절하며 환자 스스로 운동한다. 이렇게 가정에서 재활을 진행할 때 의료진을 대신하여 환자의 상태를 실시간으로 점검하며 운동을 감독할 수 있는

시스템을 도입하여 환자가 더욱 안전한 환경에서 재활을 진행할 필요성을 느껴 시스템 제작을 계획했다.

2. 초기 구상안

재활센터에서 의료진의 감독 하에 진행되는 재활치료와 달리 가정에서는 환자 스스로 운동을 진행하여야 한다. 또한 심장 재활에서 진행되는 운동은 충분한 강도의 운동을 안전하게 시행하는 것이 중요하다. 따라서 재활 진행 중 심박수를 측정하여 목표 심박수에 맞춰 운동을 시행해야 한다. 부산대학



교병원 심뇌혈관센터의 심뇌재활센터에서는 주 3~5회 등에 땀이 날 정도로, 하지만 가슴통증이 없고 약간의 힘이 드는 정도의 운동을 진행해야 한다고 말했다. 운동은 런닝 머신, 자전거타기, 계단 오르기 등의 유산소운동을 시행하는 것이 좋다고 안내했다. 또한 삼성병원 예방재활센터에서는 본 운동 전후에 심혈관계와 근육계를 활성화시켜주는 준비운동과 말초로 많이 가 있는 혈액을 심장으로 원활히 돌아오게 하는 정리운동을 필수적으로 시행해야 하며 재활 환자의 심박수가 운동강도의 지표로서 가장 중요하다고 안내했다. 또한 심장질환자가 운동을 진행할 때 협심증상이 나타날 시 혈관확장제를 복용하라고 말했다.

스피커에서 연령대와 운동강도를 입력하라는 안내가 나오면 해당 값을 입력 받는다. 목표 심박수는 최대 심박수의 40%를 기본을 설정하여 재활센터의 처방이나 환자의 건강상태에 따라 운동강도를 조절할 수 있도록 설정한다. 스위치의 '증가', '감소' 버튼을 이용하여 원하는 값으로 설정하고 '저장' 버튼은 누르면 값이 확정된다. 값을 입력하는 과정은 LCD에 출력되어 화면을 확인하며 값을 조절한다. 최대 심박수는 카보넨 공식을 이용하여 207에서 연령에 0.7을 곱한 값을 빼서 계산한다. 계산이 완료되면 운동 시작을 스피커를 통해 알리고 운동 중에는 LCD에 심박수와 사전에 계산한 목표 심박수를 함께 보여준다. 운동을 종료하려고 할 때는 '종료' 버튼을 누르면 LCD에 운동 종료 문구를 출력하고 스피커로 운동종료를 알리며 시스템을 종료한다. 만약 운동 중 심박수가 목표 심박수를 초과하면 스피커를 통해 알리고 운동을 즉시 중지시킨다. 또한 LCD에도 운동을 중지하라는 문구를 출력하고 혈관확장제를 복용할 수 있도록 약통을 모터를 이용하여 연다.

3. 간트 차트

번호	작업 제목	1단계			2단계				3단계					4단계			
		1주	2주	3주	4주	5주	6주	7주	8주	9주	10주	11주	12주	13주	14주	15주	16주
1	프로젝트 구상 및 착수																
1.1	아이디어 수립 및 기획																
1.2	자료조사																
2	프로젝트 정의 및 계획																
2.1	센서 분석																
2.2	코드 분석																
2.3	순서도																
2.4	예산																
3	프로젝트 구상 및 착수																
3.1	심박수 센서 + LCD																
3.2	적외선 센서 + 리모컨																
3.3	mp3모듈 + 스피커																
3.4	모터																
3.5	모니터링																
4	프로젝트 성과																
4.1	검토																
4.2	프로젝트 성과																

II. 본론

1. 설계 조건과 주요 부품

가정에서의 심장재활을 지도해주기 위한 시스템을 제작하기 위하여 아두이노 우노 보드를 이용한다. 운동 진행 중의 알림은 MP3모듈을 이용하고 이와 연결된 스피커를 통한다. 초기 입력 값과 운동 종료는 버튼 스위치를 활용해 조작한다. 또한 심박수를 측정하고 목표 심박수에 도달하면 운동을 중지시키기 위해 심박수 센서를 연결한다.



측정한 심박수를 LCD 모니터에 표시하고 목표 심박수를 초과했을 시 혈관확장제가 담긴 약 상자 뚜껑 모터를 이용해 열리게 한다.

1) 아두이노 우노보드

6개의 아날로그 핀과 14개의 디지털 핀이 있으며 디지털 핀 중 PWM핀(3, 5, 6, 9, 10, 11핀)은 아날로그 신호처럼 제어할 수 있다. 외부 전원은 7~12V를 공급받을 수 있다. ATmega328P 마이크로 컨트롤러를 사용하고 32KB의 플래시메모리와 2KB의 SRAM을 제공한다. PC와의 시리얼 통신을 위해 디지털 0, 1, 2번 핀을 사용한다.

2) 심박수 센서: sen0203 heart rate sensor

Arduino용 LED 심박수 센서는 LED와 포토다이오드를 사용하여 사람의 손가락이나 귓볼의 혈액량 변화를 감지하는 특정 유형의 심박수 센서이다. 심장이 뛰면 혈액이 혈관을 통해 펌핑되어 피부가 흡수하는 빛의 양에 약간의 변화가 생긴다. 혈류량이 높을 때는 빛이 많이 반사되고 혈류량이 낮을 때는 빛이 많이 투과된다. Arduino용 LED 심박수 센서는 피부에 특정 파장의 빛을 방출하고 포토다이오드를 사용하여 반사되는 빛



의 양을 감지하여 작동한다. 포토다이오드는 감지된 빛의 양에 비례하는 전압 신호를 생성한 다음 Arduino 보드에서 읽을 수 있는 디지털 신호로 변환된다. Arduino 보드가 디지털 신호를 수신하면 데이터를 처리하여 심박수를 계산하고 연결된 디스플레이에 표시하거나 추가 분석을 위해 다른 장치로 전송할 수 있다.

Sen0203 심박수 센서는 전원부 VCC, GND 2핀과 센서를 제어할 제어용 1핀으로 구성되어있다. 세부 사양은 오른쪽 표과 같다.

입력전압(Vin)	3.3 - 6V(5V 권장)
출력전압	0 - Vin(아날로그), 0/Vin(디지털)
작동 전류	10mA
규격	28*24(mm)
인터페이스	PH2.0~3P

본 프로젝트에서는 LED 심박수 센서를 사용하여 환자가 재활 치료를 위해 운동하는 동안 환자의 심박수를 실시간으로 측정한다. 사용자의 심박수가 목표 심박수에 도달하면 운동을 중단하라는 경고를 사용자에게 보낸다.

3) LCD: 1602 LCD

Arduino용 LCD 모니터는 Arduino 마이크로 컨트롤러 보드에 연결하여 개발 중인 프로젝트에 대한 시각적 피드백과 정보를 제공할 수 있는 디스플레이 장치이다. LCD(Liquid Crystal Display) 모니터는 두 개의 투명 전극 사이에 끼워진 액정층으로 구성되어 있으며 영숫자 문자, 기호 및 그래픽을 표시하는 데 사용할 수 있다.



LCD 모니터에 정보를 표시하기 위해 Arduino 보드는 선택한 인터페이스를 통해 디스플레이에 명령과 데이터를 보낸다. 디스플레이는 이러한 명령과 데이터를 해석하여 화면 내용을 업데이트한다. 예를 들어 LCD 모니터에 텍스트 메시지를 표시하려면 라이브러리에서 적절한 기능을 사용하여 화면의 특정 위치에 텍스트를 쓴다.

Arduino용 LCD는 센서 판독값, 시스템 상태 및 사용자 입력과 같은 실시간 데이터를 표시하

는 데 사용할 수 있다. LCD 모니터를 사용하여 메뉴 옵션, 오류 메시지 및 프롬프트와 같은 지침과 피드백을 사용자에게 제공할 수도 있다.

1602 LCD는 RGB chracter LCD로 2줄 16문자를 출력하며 한 문자당 5*8 사이즈의 도트로 이루어져있다. 백라이트는 blue이고 영문자와 특수문자를 지원한다. 총 16개의 핀으로 이루어져 있으며 각 핀에 대한 설명은 아래의 핀맵을 참고한다.

No.	핀 이름	레벨	기능
1	Vss	-	GND
2	Vdd	-	+5V
3	V0	-	LCD 밝기 제어
4	RS	H/L	레지스터 선택 H: Data input L: Instruction input
5	RW	H/L	읽기/쓰기 모드 선택 H: 읽기 모드 L: 쓰기 모드
6	E	H	Enable, 쓰기모드 활성화
7	D0	H/L	데이터 입출력 핀 8비트 모드인 경우 사용
8	D1	H/L	
9	D2	H/L	
10	D3	H/L	
11	D4	H/L	데이터 입출력 핀 4비트, 8비트모드일 때 모두 사용
12	D5	H/L	
13	D6	H/L	
14	D7	H/L	
15	BLA	-	Backlight +5V
16	BLK	-	Backlight GND

spi방식의 LCD를 사용하기 위하여 관련 함수를 지원하는 LiquidCrystal.h 라이브러리를 추가하여 사용한다. 해당 라이브러리에는 LCD 설정 및 객체 생성, LCD 초기화 및 통신 시작, 화면 모두 지우기, 지정된 위치로 커서 이동 등의 함수를 제공한다.

LiquidCrystal(uint8_t rs, uint8_t enable, uint8_t d0, uint8_t d1, uint8_t d2, uint8_t d3)		
기능	LCD 설정 및 객체 생성	
매개변수	rs	LCD의 RS핀
	enable	LCD의 E핀
	d0	LCD의 D4핀

	d1	LCD의 D5핀
	d2	LCD의 D6핀
	d3	LCD의 D7핀
리턴 값	-	없음

void begin(uint8_t cols, uint8_t rows, uint8_t charsize = LCD_5x8DOTS)		
기능	LCD 초기화 및 통신 시작	
매개변수	cols	LCD의 열 개수
	Rows	LCD의 행 개수
	Charsize	점 사이즈(기본 5*8)
리턴 값	void	없음

void clear()		
기능	화면 모두 지우기	
매개변수	void	없음
리턴 값	void	없음

void setCursor(uint8_t col, uint8_t row)		
기능	지정된 위치로 커서 이동, 해당 위치부터 데이터 출력	
매개변수	col	열 지정
	row	행 지정
리턴 값	void	없음

4) 푸쉬 버튼스위치

푸쉬버튼은 디지털 입력을 감지하는 데 사용되는 일반적인 하드웨어 장치이다. 푸쉬버튼은 보통 "누름 버튼"이나 "전환 스위치"와 같은 형태를 가지며, 사용자의 입력에 따라 전기 신호를 변경한다.

푸쉬버튼은 4개의 다리가 2개씩 연결되어있고 내부에 기계적인 접점이 있어 사용자가 스위치를 누름으로써 접점 간에 연결 또는 끊김을 발생시킨다. 이러한 접점의 상태 변화는 전기 신호의 흐름을 제어하는 역할을 한다. 푸쉬버튼이 눌렸는지 또는 눌리지 않았는지를 나타내는 이진 상태로 변환된다.



푸쉬버튼을 연결하는 방식으로는 풀업(pull up)방식과 풀다운(pull down)방식으로 나뉘는데 풀업방식은 10k옴의 저항에 5V 전원을 연결하는 것이고 풀다운 방식은 10k옴의 저항에 접지 GND를 연결하는 것이다. 풀업 방식일 때 푸쉬버튼이 눌리면 LOW, 눌리지 않으면 HIGH상태가 된다. 풀다운 방식으로 푸쉬버튼이 연결되었을 때 버튼이 눌리면 전류가 흐르게 되고, 전기 신호는 HIGH 상태로 감지된다. 반대로 푸쉬버튼이 눌리지 않았을 때, 접점이 열리면 전기 신호는 LOW 상태로 감지된다.

풀업 방식		
연결	저항	전원 5V, 입력 핀
	푸쉬버튼	접지 GND
동작	눌림	LOW
	안 눌림	HIGH

풀다운 방식		
연결	저항	접지 GND, 입력 핀
	푸쉬버튼	전원 5V
동작	눌림	HIGH
	안 눌림	LOW

아두이노와 같은 마이크로컨트롤러는 이러한 디지털 입력 신호를 읽어와서 처리할 수 있다. 푸쉬버튼을 디지털 핀에 연결하고, digitalRead() 함수를 사용하여 해당 디지털 핀의 상태를 확인한다. 이를 통해 푸쉬버튼의 입력 상태에 따라 원하는 작업을 수행할 수 있다.

스위치 센서의 원리는 간단하지만 매우 유용하며, 다양한 전자 장치와 시스템에서 입력 장치로 널리 사용된다. 본 프로젝트에서는 '증가', '감소', '저장', '종료'에 대한 4개의 버튼스위치 센서를 이용해 사용자의 입력을 감지하여 연령대와 운동 강도를 입력하고 운동종료를 설정하는 데에 쓰인다.

5) 서보모터: Micro servo SG-90

서보모터는 모터 내부의 컨트롤러로 얼마나 회전했는지를 센싱하여 입력된 각도만큼 회전하는 모터이다. 180도 회전을 기본으로 PWM을 이용해 회전각도를 조절한다. PWM신호는 펄스 폭 변조 신호로 주기적으로 생성되는 펄스의 폭을 조절하여 각도를 설정한다. 20ms주기로 약 1ms에서 2ms의 펄스 폭을 입력받아 동작한다. 1.5ms의 펄스 폭을 중립 위치로 인식한다.



서보모터는 전원부 Vcc, GND의 2핀과 서보모터를 제어할 제어용 1핀으로 구성된다. 서보모터를 활용하기 위해서는 Servo.h 라이브러리를 이용하여 조작한다. 해당 라이브러리에서 제공하는 함수 중 대표적으로는 모터 연결설정, 모터 각도 변경, 모터 신호 제공 멈춤 함수가 있다.

void attach(int pin)	
기능	지정된 핀으로 모터 연결

매개변수	int pin	서보 모터 제어용 핀
리턴 값	void	없음

void write(int value)		
기능	모터 각도 변경	
매개변수	int value	변경할 각도
리턴 값	void	없음

void detach(void)		
기능	PWM신호 제공을 하지 않게 함	
매개변수	void	없음
리턴 값	void	없음

본 프로젝트에서는 서보모터를 약상자를 구성할 때 사용한다. 심전도 센서에서 정한 기준치 이상으로 심박수가 올라가면 약상자가 열리게 되고 이때 모터가 뚜껑을 열 때 사용한다.

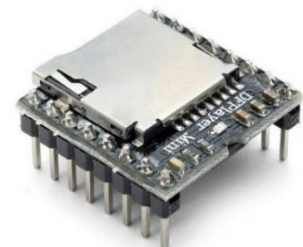
6) 소형 스피커: FIT0502 Stereo Enclosed Speaker

스피커는 전기 신호를 소리로 출력하는 장치로 진동판이 움직이며 소리가 발생한다. Stereo Enclosed Speaker는 아두이노 보드나 모듈에 연결하여 사용하며 +단자와 -단자에 해당하는 2개의 핀으로 구성되어 있다. JST PH2.0 인터페이스를 갖추고 있으며 8옴 3W의 전력을 필요로 하는 곳에 사용 가능하다. 무게는 24g, 규격은 70*30*16mm이다. 아래의 DFplayer mini에 연결하여 사용할 예정이다.



7) mp3모듈: DFR0299 DFplayer mini

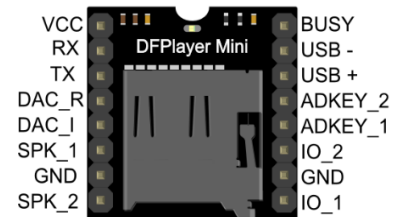
DFplayer mini는 MP3 모듈 중 하나로, SD카드를 삽입하여 사용할 수 있다. 모듈의 SD 카드에 저장된 음원파일을 모듈에 연결된 스피커를 통해 재생한다. 먼저, SD 카드에 저장된 MP3 파일을 읽어와서 내부 버퍼에 저장하고 이를 오디오 신호로 변환한다. 이 신호는 모듈에 내장된 오디오 출력 잭을 통해 외부 스피커를 통해 출력된다.



MP3, WAV, WMA 파일을 재생할 수 있으며, 이전/다음 노래 재생, 볼륨 조절 등의 기능을 지원한다. 모듈의 크기는 약 24mm x 18mm x 3mm 정도이며 5V 전원으로 구동되고 스위치와 스피커를 연결하면 모듈 단독으로도 사용 가능하다. I/O 제어모드, 시리얼모드, AD버튼 제어모드 등 다

양한 방법으로 제어할 수 있으며 샘플링속도는 8/11.025/12/16/22.05/24/32/44.1/48 khz이고 최대 100개 폴더와 각 폴더별 최대 255개의 음원파일을 저장하여 재생할 수 있다. 또한 FAT32, FAT64 파일 시스템과 최대 32G의 TF카드와 U disk까지 지원한다.

DFplayer mini는 중앙의 SD카드 삽입부를 기준으로 좌측과 우측에 각각 8개의 핀이 존재한다. 전원부인 VCC 입력 전압, GND 핀과 DFplayer mini를 제어하기 위한 제어용 핀인 시리얼 통신, 이어폰 출력, 스피커 연결, AD키와 USB용 포트 등과 관련된 핀으로 구성되어 있다. RX, TX는 각각 serial input, output으로 우노 보드의 디지털 핀과 1kΩ 저항을 연결하여 사용한다. VCC는 5V, GND는 우노 보드의 GND와 연결하며 +단자에 해당하는 SPK1과 -단자에 해당하는 SPK2에 3W이하의 스피커와 연결해야 한다. 핀 별 상세한 기능은 아래의 핀맵을 참고한다.



No.	핀 이름	기능
1	VCC	입력 전압 3.2~5V
2	RX	UART 시리얼 통신 입력핀
3	TX	UART 시리얼 통신 출력핀
4	DAC_R	이어폰이나 앰프 오른쪽 오디오 출력
5	DAC_L	이어폰이나 앰프 왼쪽 오디오 출력
6	SPK_1	스피커-, 2개의 선 중 하나와 연결
7	GND	전원 GND 연결
8	SPK_2	스피커+, 2개의 선 중 하나와 연결
9	IO_1	버튼 연결, 이전 곡 재생, 길게 누르면 볼륨 감소
10	GND	전원 GND 연결
11	IO_2	버튼 연결, 다음 곡 재생, 길게 누르면 볼륨 증가
12	ADKEY_1	AD키로 제어 시 사용
13	ADKEY_2	AD키로 제어 시 사용
14	USB+	USB용 포트
15	USB-	USB용 포트
16	BUSY	재생 중일 시 LOW, 정지 시 HIGH

모듈이 아두이노를 이용하여 제어할 때는 시리얼 통신으로 명령을 보내고, 모듈은 명령을 해석하여 MP3 파일을 재생하거나 제어한다. 이때 SoftwareSerial.h 라이브러리를 이용하여 아두이노 보드의 디지털 핀을 시리얼 포트 RX핀과 TX핀으로 작동하게 만들어 준다. 그리고 DFRobotDFPlayerMini.h 라이브러리를 이용하여 재생, 볼륨설정을 시행한다. 예를 들어, play 명령을 보내면 모듈은 현재 저장된 MP3 파일 중 첫 번째 파일을 재생한다. next 명령을 보내면 다음

MP3 파일을 재생한다. 또한, volume_up 또는 volume_down 명령을 보내면 볼륨을 조절할 수 있다. 해당 라이브러리의 대표적인 함수는 다음과 같다.

SoftwareSerial mySoftwareSerial(int pin1, int pin2)		
기능	소프트웨어 시리얼 핀 지정	
매개변수	int pin1	핀 번호
	int pin2	핀 번호
리턴 값	-	없음

myDFPlayer.volume(int volume)		
기능	볼륨 설정, 0부터 30사이 값 선택	
매개변수	int volume	볼륨 값 입력
리턴 값	-	없음

myDFPlayer.volumeUp()		
기능	볼륨 1단계 상승	
매개변수	-	없음
리턴 값	-	없음

myDFPlayer.volumeDown()		
기능	볼륨 1단계 감소	
매개변수	-	없음
리턴 값	-	없음

myDFPlayer.next()		
기능	다음 음원 파일 재생	
매개변수	-	없음
리턴 값	-	없음

myDFPlayer.play(int track)		
기능	지정한 트랙의 음원파일 재생	
매개변수	int track	재생을 원하는 음원파일의 번호
리턴 값	-	없음

myDFPlayer.playFolder(int folder, int track)		
기능	지정된 폴더의 지정된 음원파일 재생	
매개변수	int folder	재생을 원하는 폴더의 번호
	int track	재생을 원하는 트랙의 음원파일 번호
리턴 값	-	없음

본 프로젝트에서는 사용자가 운동을 진행할 시 나이와 운동강도 입력, 운동시작, 운동 마침, 목표 심박수 초과로 인한 운동 중단 알림을 MP3 모듈을 스피커에 연결하여 해당 음성파일이 재생되는 방식으로 활용한다.

2. 부품구매내역

No.	제품명	수량	단가	금액
1차/2차 공동구매				
1	[DFROBOT] 소형 스테레오 스피커 Stereo Enclosed Speaker - 3W 8Ω [FIT0502]	1	4600	4600
	https://www.devicemart.co.kr/goods/view?no=1322046			
2	[DFROBOT] 아두이노용 심박 센서 Heart Rate Monitor Sensor For Arduino [SEN0203]	1	20500	20500
	https://www.devicemart.co.kr/goods/view?no=1327536			
3	[SanDisk] SanDisk microSDHC, SDQM 16GB, C4, [SDSDQM-016G-B35]	1	4080	4080
	https://www.devicemart.co.kr/goods/view?no=14825248			
4	[DFROBOT] DFPlayer - A Mini MP3 Player [DFR0299]	1	8900	8900
	https://www.devicemart.co.kr/goods/view?no=1278727			
5	[SMG] 플라스틱 기어 34개입 세트 [SZH-GNP230]	1	5500	5500
	https://www.devicemart.co.kr/goods/view?no=1341605			
6	[OEM] 아두이노 아크릴 고정판 [SZH-EK016]	1	1300	1300
	https://www.devicemart.co.kr/goods/view?no=1264626			
7	[동우전선] 3색 점퍼용 단선- 0.6 파이 [1미터단위]	3	400	1200
	https://www.devicemart.co.kr/goods/view?no=1153807			
8	[Coms] 아크릴 전용접착제, 투명, 30g [LV0231]	1	5900	5900
	https://www.devicemart.co.kr/goods/view?no=12519728			
9	[Coms] [BF164] 커터(RG-334) Wide Blade / 플라스틱 절단용 커터칼 / 아크릴 / 디자인 / 색상 랜덤발송	1	2100	2100
	https://www.devicemart.co.kr/goods/view?no=10892537			
10	[일렉트로쿠기] PCB기판 브레드보드형 만능기판 Half Size(아두이	1	2190	2190

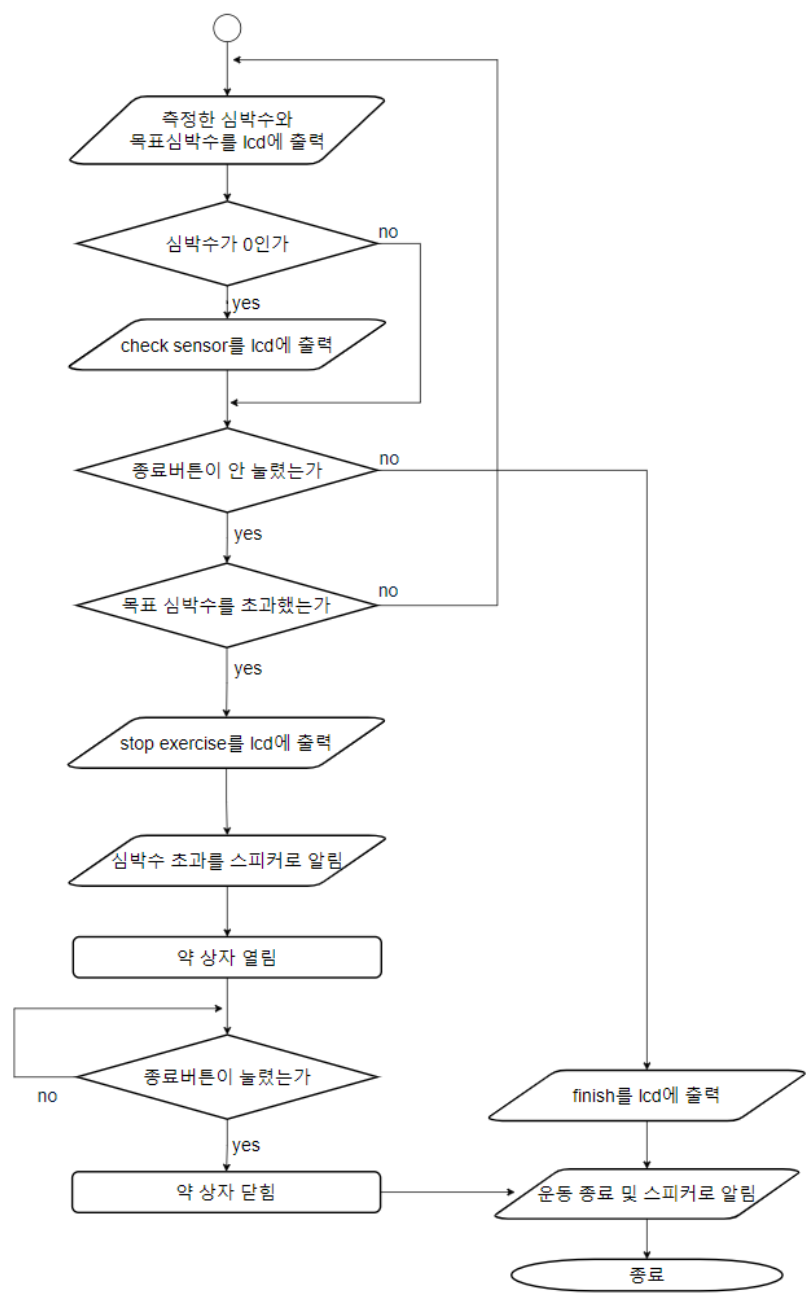
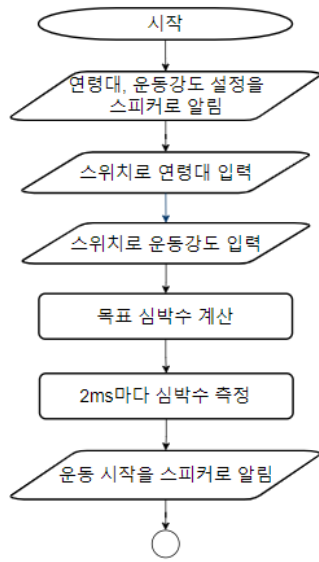
	노 및 개발용)- 89x52mm 블랙			
	https://www.devicemart.co.kr/goods/view?no=12506669			
11	[하나아크릴] 2T 스모그(투명) 아크릴 옵션: 285*425(mm)	8	4100	32800
	https://www.devicemart.co.kr/goods/view?no=7848			
			합계	89070
개별 구매				
12	우드락 5mm 검정	1	3100	3100
	군자관 서점 구매			
13	아두이노 12X12X7.3mm 택트 스위치+버튼 캡 25개 세트	1	4400	4400
	https://link.coupang.com/a/0xAtG			
			합계	7500
			총합	96570

장치의 외관 제작을 위해 아크릴과 아크릴 커터칼, 아크릴 접착제를 구매하였으나 2차 부품 공동구매 시 물품 배송이 지연되어 사용하지 못하였다. 대신 우드락 보드를 추가 구매하여 제작하였다.

3. 구현 상세 내용

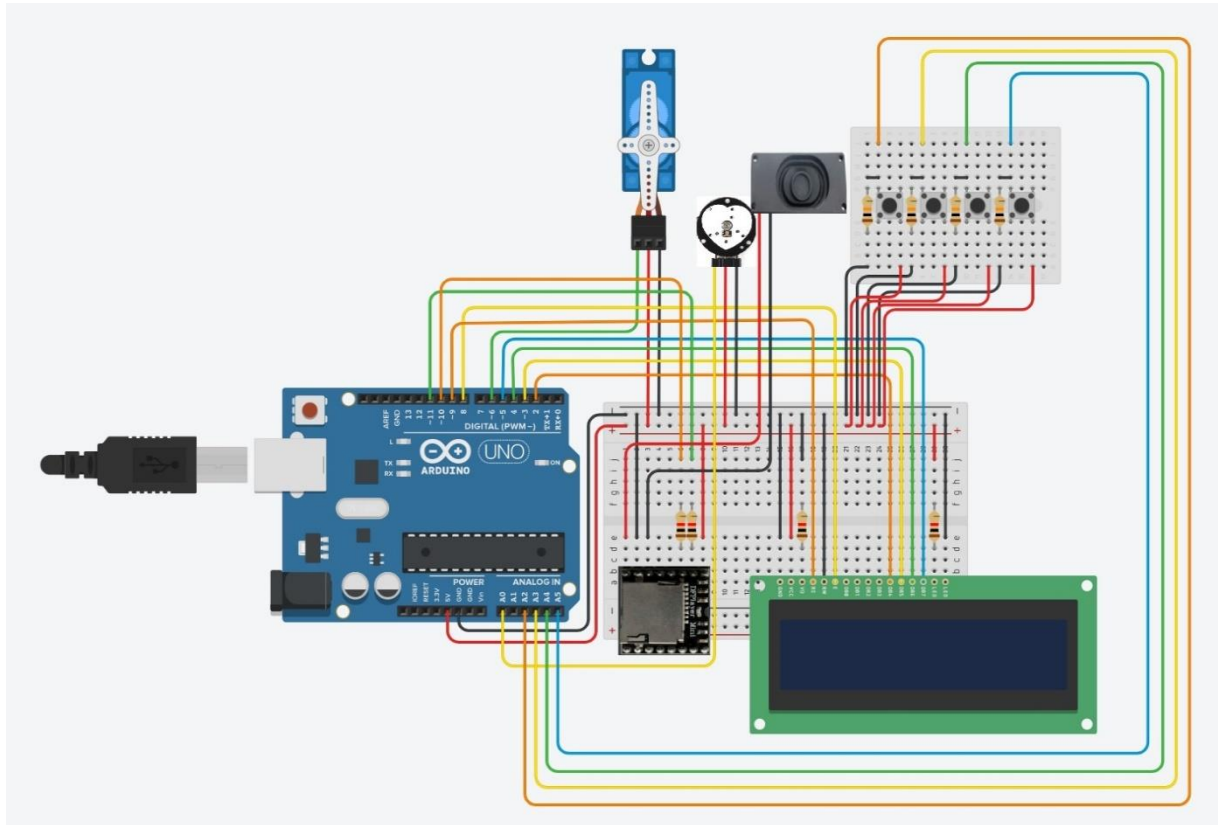
스피커로 연령대와 운동강도를 입력하라는 스피커의 안내 후 4개의 버튼 중 '증가', '감소', '저장' 3가지의 버튼을 이용하여 LCD에서 출력되는 연령대와 운동강도를 확인하며 버튼으로 조절하여 값을 입력한다. 카보넨 공식을 이용하여 $207에서\ 연령대 \times 0.7$ 을 빼서 최대 심박수를 구하고 최대 심박수에 $운동강도 \times 0.01$ 을 곱하여 나온 값을 목표 심박수로 저장한다. 운동을 시작한다는 것을 스피커를 통해 알리고 심박수는 2ms마다 측정한다.

운동 시작과 동시에 측정된 심박수를 LCD의 첫 줄에, 목표 심박수를 두번째 줄에 출력한다. 심박수가 0일 때는 두번째 줄에 센서의 오작동이 의심되니 확인하라는 안내 문구를 출력한다. 종료버튼이 눌릴 때까지 해당 동작을 반복한다. 종료버튼이 눌리면 운동종료 안내문구를 LCD 두번째 줄에 출력하고 스피커로도 운동종료를 알린다. 다만 운동 중 측정한 심박수가 목표 심박수를 초과했을 때는 LCD에 운동을 멈추라고 안내 문구를 출력하고 심박수가 높다는 것과 운동종료를 스피커로 알린다. 또한 약 복용을 위해 서보모터를 사용하여 약 상자 뚜껑을 연다. 종료버튼이 눌렀다면 뚜껑을 닫는다.



4. 회로도 및 외관

1) 회로도 설계



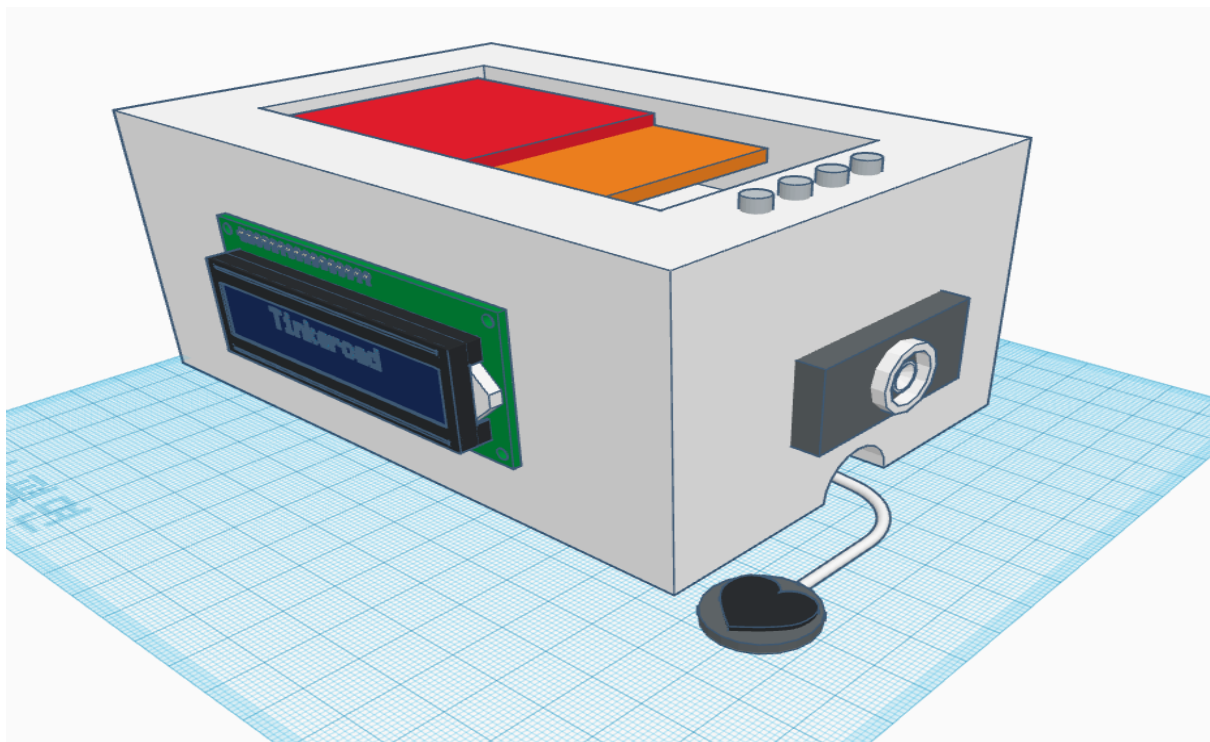
본 심장재활 보조장치의 회로도는 위와 같다. 먼저 심박수 센서는 전원부는 5V와 GND에 연결하고 제어용 핀은 아날로그 0번에 연결한다. LCD는 VSS, BLK, RW핀은 GND에 VDD는 5V에 연결하며 V0와 BLA는 각각 1K옴의 저항을 연결한 상태로 GND와 5V에 연결한다. 그리고 RS와 E핀은 디지털 9번과 디지털 8번에 연결한다. 4비트 모드일때 데이터 입출력을 담당하는 D4, D5, D6, D7핀은 디지털 2번, 디지털 3번, 디지털 4번, 디지털 5번과 연결한다. 다음으로 서보 모터의 전원부는 5V와 GND에 연결하고 제어 핀은 디지털 6번에 연결한다.

DFplayer mini는 VCC는 5V에 DFplayer mini의 GND는 우노 보드의 GND에 연결한다. SPK_1과 SPK_2는 각각 Stereo Enclosed Speaker의 -단자와 +단자에 하나씩 연결한다. 그리고 DFplayer mini의 RX, TX핀은 각각 serial input, output으로 우노 보드의 소프트웨어 시리얼 핀인 디지털 11번과 디지털 10번에 1k Ω 저항을 연결한 상태로 연결한다.

연령대와 운동강도를 입력하고 운동 종료를 설정하는 데에 사용하는 4개의 '증가', '감소', '저장', '종료'를 뜻하는 버튼스위치는 풀다운 방식으로 회로도를 구성하였다. '증가', '감소', '저장', '종료'는 각각 아날로그 2핀, 아날로그 3핀, 아날로그 4핀, 아날로그 5핀에 연결하였고 본래 버튼 스위

치는 디지털 핀에 주로 연결하지만 본 프로젝트에서는 디지털 핀의 개수가 부족하여 우노 보드가 아날로그 핀에 연결된 값을 digitalRead()을 이용하여 디지털 핀으로 사용할 수 있다는 것을 활용하여 회로도를 설계하였다. 10K옴의 저항에 접지 GND와 입력 핀, 버튼의 한쪽 다리에 연결하고 반대편에는 5V를 연결하였다.

2) 외관 설계

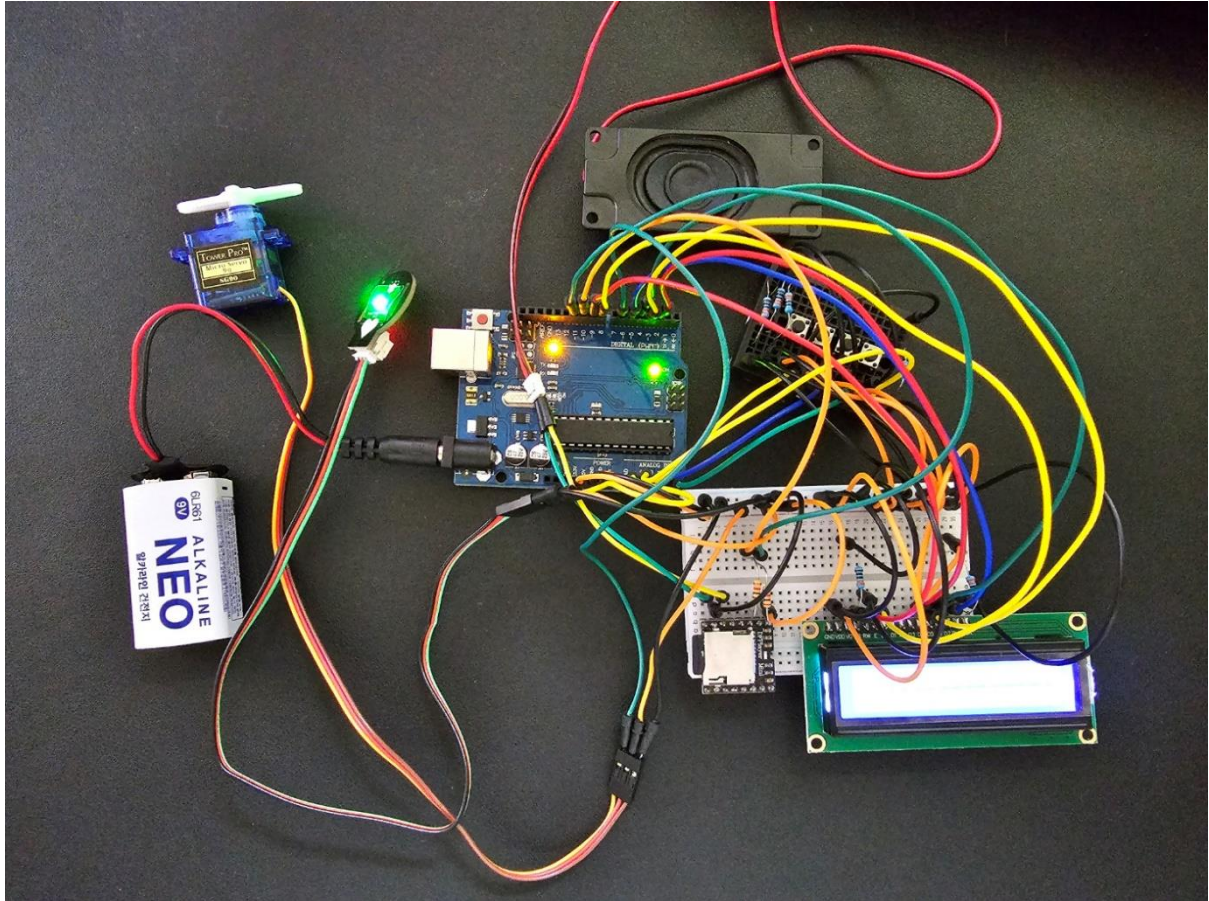


Tinkercad의 3D 디자인을 이용해서 심장재활 보조장치의 외관을 나타내면 위와 같다. 윗면의 커버는 모터를 이용하여 열고 닫히는 약통 케이스의 역할을 한다. 커버의 우측에는 스위치 4개가 달려있다. 스위치들은 각각 '증가', '감소', '저장', 그리고 '종료'의 신호를 보낸다. '증가'와 '감소'는 사용자가 본인의 연령대와 운동 강도의 값을 지정할 때 사용된다. '저장'은 지정된 값을 입력할 때, '종료'는 운동을 마무리할 때 사용하는 스위치이다.

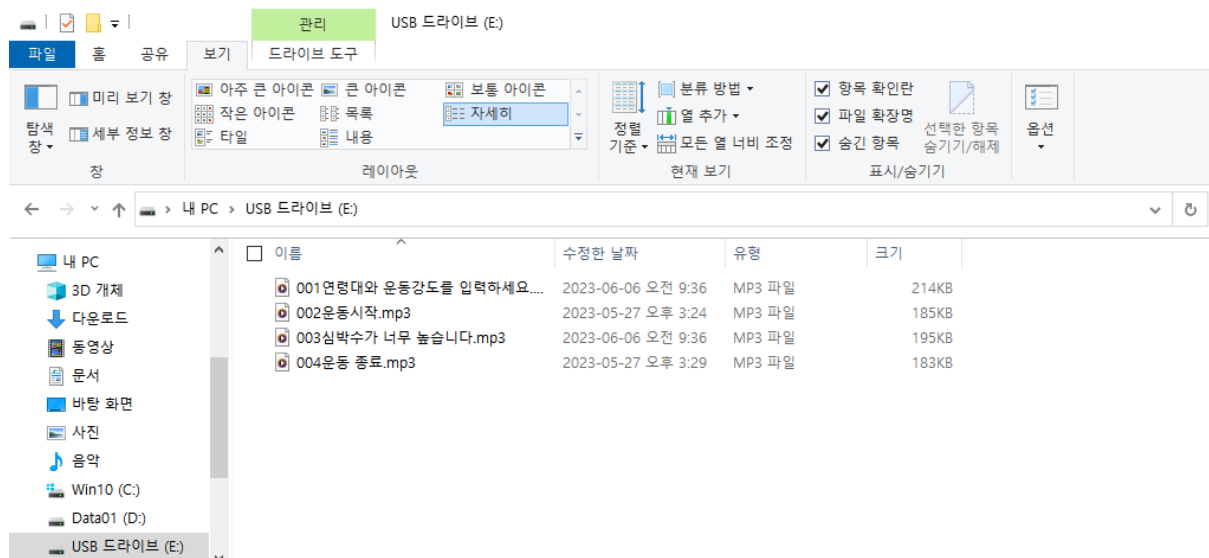
시스템의 앞면에는 LCD 모니터가 있다. LCD 모니터는 사용자에게 시작할 때의 입력하는 값과 운동하는 도중의 심박수를 보여준다. 옆면에는 스피커가 있고, 이 스피커는 본인의 연령대와 운동 강도의 값을 입력하라는 것과 같은 음성 신호를 사용자에게 보낸다. 시스템의 아래쪽에는 심박수 센서가 있어 LED와 포토다이오드를 사용하여 사람의 손가락이나 귓볼의 혈액량 변화를 감지한다.

3) 작품 제작

위에서 제시한 회로도와 외관을 기준으로 장치를 제작한다. 먼저 우노보드와 브레드보드, 전선을 이용하여 각 센서들을 연결한다. 아래 사진에서는 9V건전지까지 연결한 모습이다.

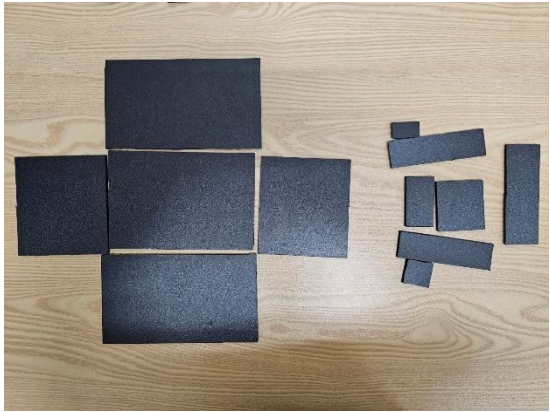
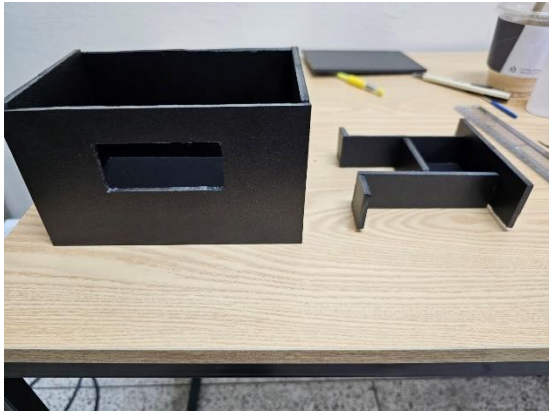


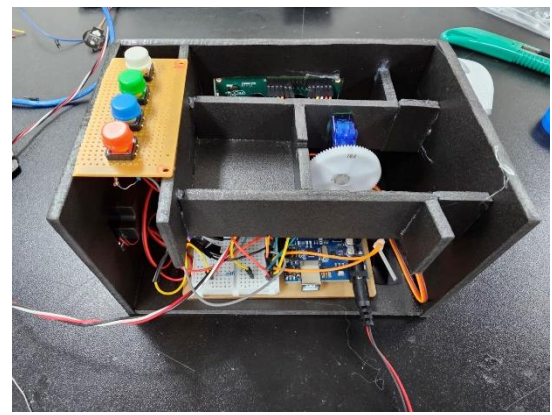
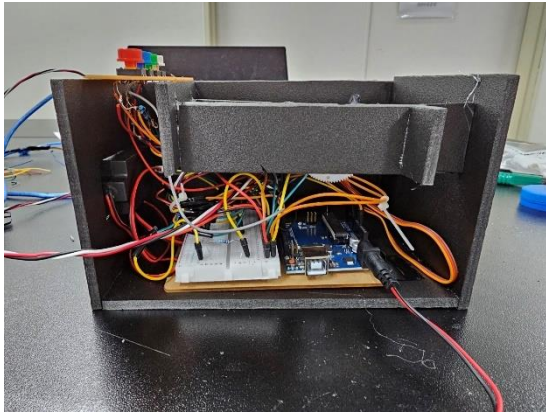
DFplayer에 삽입하는 sd카드에는 연령대와 운동강도 입력, 운동시작, 심박수 초과, 운동종료



에 대한 4개의 음원파일이 저장되어 있다.

이제 우노보드와 여러 센서들을 넣을 케이스를 제작한다. 우드락을 커터칼과 열선을 이용하여 자르고 우드락 전용 본드와 글루건을 이용하여 부착한다. 그 과정은 아래와 같다.

<1>	<2>
<p>장치의 겉면과 내부에 부착할 약통을 제작할 면을 재단한다. 밑면은 18*12, 옆면은 19*11 2장, 12*11 2장으로 하고 약 상자 바닥은 5.5*6, 옆면은 12*4, 6*3.3, 11.5*3.3 2장으로 자른다. 약상자를 단단하게 부착하기 위해 지지대를 세우는데 이는 3.5*3.3, 2.5*3.3이다.</p>	<p>장치 겉면 중 앞면에는 LCD를 부착할 구멍을, 왼쪽 면에는 스피커를 부착할 구멍을 뚫고 <1>에서 자른 조각들을 우드락 전용 본드를 이용하여 붙인다.</p>
	
<3>	<4>
<p>전선을 케이블 타이를 이용해 정리하고 장치 앞면에 LCD를 부착하고 옆면에는 스피커를 부착한다. 버튼 스위치는 납땜하여 장치 뒷면에서 보이도록 상단에 고정한다. 그 후 장치에 아두이노 보드, 브레드보드와 전선, 센서와 같은 부품들을 전부 넣고 고정한다.</p>	<p><4> 약 상자 개폐를 위해 서보 모터를 약 상자 옆에 고정한다.</p>

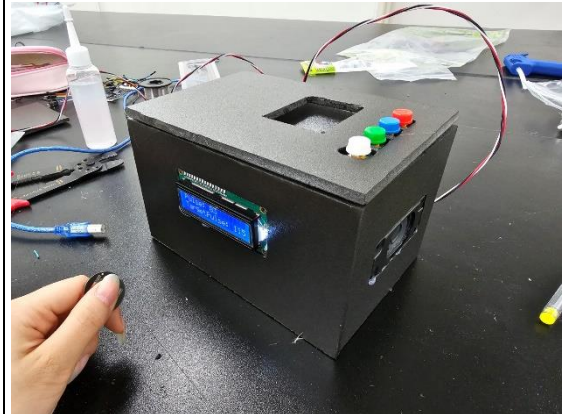
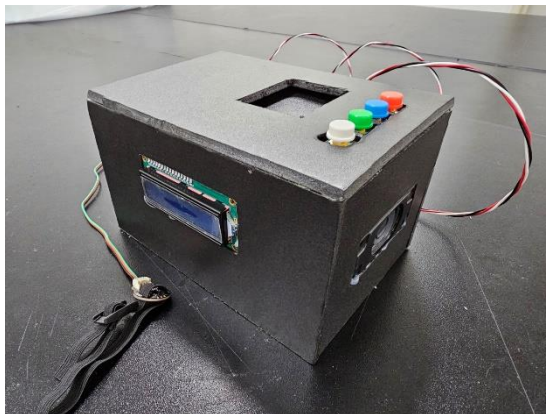


<5>

<6>

윗면을 19*13으로 자르고 앞서 내부에 부착한 약 상자 위치에 맞게 윗면에 구멍을 뚫는다. 약 상자 우측에 위치한 '증가', '감소', '저장', '종료' 버튼을 위치시킬 구멍을 뚫는다. 윗면을 붙인다.

9v 건전지를 연결하여 동작시킨 모습이다.



5. 구현 코드

코드는 총 4개의 소스파일로 나뉘어져 있다.

- healthcare.ino: 헤더파일과 여러 변수 선언, setup함수와 loop함수
- set_age_intensity.ino: 연령대와 운동강도를 입력받는 setage함수와 setintensity 함수
- interrupt.ino: 2ms마다 심박수를 측정하게 해주는 interruptSetup함수와 ISR함수
- servo.ino: 서보모터 움직여 약 상자 뚜껑 개폐시켜주는 openservo함수

1) healthcare.ino

```
1  #include <Servo.h>
2  #include <LiquidCrystal.h>
3  #include "Arduino.h"
4  #include "SoftwareSerial.h"
5  #include "DFRobotDFPlayerMini.h"
6
7  DFRobotDFPlayerMini myDFPlayer;
8  Servo servo;
9
10 SoftwareSerial mySoftwareSerial(10, 11);
11 LiquidCrystal lcd(9,8,2,3,4,5);
12 int pulsePin = 0;          // 심박수센서 아날로그 0핀
13
14 const int SERVO = 6;      // 서보모터 디지털 6핀
15
16 int endbutton=A2;         // 종료버튼 아날로그 2핀
17 int end_prestate=0;
18 int endbuttonstate;
19
20 int upbutton=A3;          // 증가버튼 아날로그 3핀
21 int up_prestate=0;
22 int upbuttonstate;
23
24 int downbutton=A4;        // 감소버튼 아날로그 4핀
25 int down_prestate=0;
26 int downbuttonstate;
27
28 int savebutton=A5;        // 저장버튼 아날로그 5핀
29 int save_prestate=0;
30 int savebuttonstate;
31
32 volatile int BPM;         // 심박수
33 volatile int Signal;      // 심박수 센서의 신호값
34 volatile int IBI = 600;   // 심장 박동 간격
35 volatile boolean Pulse = false; // 실시간 심장 박동 여부
36 volatile boolean QS = false; // 심장 박동 감지 여부
37
38 int age=0;                // 연령대
39 int intensity=40;         // 운동 강도
40 int max_hearttrate = 0;   // 최대심박수
41 int target_hearttrate = 0; // 목표심박수
42
```

먼저 코드에 필요한 헤더파일을 선언하였다. DFPlayer와 서보모터를 선언한다. 10, 11번 핀을 소프트웨어시리얼 핀으로 지정하고 9,8,2,3,4,5번 핀을 LCD핀으로 연결하고 심박수센서는 아날로그 0핀으로 지정한다. 서보모터를 아두이노의 디지털 6핀과 연결한다.

그리고 시스템에 필요한 4가지 스위치를 선언한다. endbutton은 아날로그 2핀에 연결하며 운동을 마무리할 때 사용하는 종료버튼의 역할을 한다. upbutton은 아날로그 3핀, downbutton은 아날로그 4핀에 연결하며 운동을 하기전 나이와 운동강도를 지정할 때 값을 증가하거나 감소시킬 때 사용한다. 마지막으로 savebutton은 5핀에 연결하고 앞에서 지정한 값을 저장할 때 쓴다.

그 다음은 심박수 센서를 위한 변수들을 선언하고 있다. BPM은 심박수, Signal은 심박수 센서의 신호값, IBI는 심장 박동 사이의 간격, boolean pulse는 실시간 심작 박동의 여부와 QS는 심작 박동의 감지 여부를 확인하는 변수를 선언한다. 마지막은 사용자와 관련된 변수들을 선언한다. 연령대와 운동강도는 upbutton과 downbutton으로 값을 지정하고, max_heartrate과 target_heartrate는 사용자의 연령대와 희망하는 운동강도로 정해진다.

```

43 void setup(){
44     Serial.begin(115200);
45
46     mySoftwareSerial.begin(9600);
47     Serial.println();
48     Serial.println(F("DFRobot DFPlayer Mini Demo"));
49     Serial.println(F("Initializing DFPlayer ... (May take 3~5 seconds)"));
50     if (!myDFPlayer.begin(mySoftwareSerial)) {                // dfplayer 연결 여부 확인
51         Serial.println(F("Unable to begin:"));
52         Serial.println(F("1.Please recheck the connection!"));
53         Serial.println(F("2.Please insert the SD card!"));
54         while(true){
55             delay(0);
56         }
57     }
58     Serial.println(F("DFPlayer Mini online.));
59
60     myDFPlayer.volume(20);                // dfplayer 음량 설정 0~30
61
62     pinMode(upbutton, INPUT);              // 증가버튼 입력모드 설정
63     pinMode(downbutton, INPUT);           // 감소버튼 입력모드 설정
64     pinMode(savebutton, INPUT);           // 저장버튼 입력모드 설정
65     pinMode(endbutton, INPUT);            // 종료버튼 입력모드 설정
66
67     lcd.begin(16,2);                      // lcd 초기화
68
69     myDFPlayer.play(1);                   // 연령대와 운동강도 설정 알림 재생
70
71     setage();                             // 연령대 설정 함수 호출
72     setintensity();                       // 운동 강도 설정 함수 호출
73
74     interruptSetup();                    // 2ms마다 심박 센서 신호를 읽음
75
76     myDFPlayer.play(2);                   // 운동 시작 알림 재생
77     delay(3000);
78 }

```

위의 코드는 setup함수에 대한 코드다. LCD모니터를 이용하여 사용자에게 DFplayer을 초기화 중임을 알린다. DFplayer의 연결 여부를 확인하고, 연결을 할 수 없으면 SD카드가 올바르게 연결되었는지 확인하라는 메시지가 뜬다. 제대로 연결이 된다면 DFplayer이 작동 중이라는 메시지를 띄우고 DFplayer의 음량을 설정한다.

그 다음 증가, 감소, 저장, 종료 버튼 4가지 모두 입력모드로 설정한다. 그리고 LCD를 초기화한 후 연령대와 운동강도를 설정하라는 알림을 재생과 연령대를 설정하는 함수와 운동강도를 설정하는 함수를 호출한다. 마지막으로 운동을 시작한다는 알림재생과 함께 심박수 센서 신호를 2ms마다 읽기 시작한다.

```

82 void loop(){
83     lcd.setCursor(0,0);          // lcd 첫번째줄에 심박수 출력
84     lcd.print("Pulse: ");
85     lcd.print(BPM);
86
87     endbuttonstate = digitalRead(endbutton);    // 종료버튼 상태
88
89     if(endbuttonstate==HIGH) {            // 종료버튼을 누르기 시작한 시점
90         if(end_prestate==0) {
91             delay(10);
92             end_prestate =1;              // 종료버튼 이전 상태 '눌림'으로 설정
93         }
94     }
95     if(endbuttonstate==LOW) {              // 종료버튼에서 손을 뗀 시점
96         if(end_prestate==1) {
97             delay(10);
98             end_prestate =0;              // 종료버튼 이전 상태 '안 눌림'으로 설정
99             lcd.setCursor(0,1);          // lcd 두번째줄에 finish 출력
100            lcd.print("Finish!!!!!!!!!!");
101
102            myDFPlayer.play(4);            // 운동 종료 알림 재생
103            delay(10000);
104
105            exit(0);                      // 종료
106        }
107    }
108 }
109
110 if(BPM == 0) {
111     lcd.setCursor(0,1);                // lcd 두번째줄에 check sensor출력
112     lcd.print("check sensor!!!!");
113 }
114 else {
115     lcd.setCursor(0,1);                //lcd 두번째줄에 목표심박수 출력
116     lcd.print("TargetPulse: ");
117     lcd.print(target_heartrate);
118 }
119
120 Serial.print("1: ");
121 Serial.println(BPM);

```

void loop가 시작하면서 LCD의 첫번째 줄에 심박수를 출력한다. ("Pulse: BPM") 아날로그 2핀에서 디지털로 읽어온 종료버튼의 현재 상태를 변수 endbuttonstate에 저장한다. 종료버튼의 현재 상태를 나타내는 변수 endbuttonstate이 HIGH(눌림)이고 종료버튼의 이전 상태를 나타내는 변수 end_prestate가 0(안 눌림)이면 버튼이 눌리기 시작한 시점으로 판단하여 end_prestate를 1(눌

림)으로 설정한다. 반대로 종료버튼의 현재 상태를 나타내는 변수 `endbuttonstate`이 LOW(안 눌림)이고 종료버튼의 이전 상태를 나타내는 변수 `end_prestate`가 1(눌림)이면 버튼에서 손을 떼기 시작한 시점으로 판단하여 `end_prestate`를 0(안 눌림)으로 설정한다. LCD에 운동이 종료 문구를 출력하고("Finish!!!!!!!!!!") DFplayer로 운동을 종료한다는 알림을 재생하고 시스템을 종료한다.

운동을 종료하는 것이 아니면 BPM이 0일때는 센서 오작동 알림 문구를 LCD의 두번째 줄에 출력하고("check sensor!!!!"), 0이 아니면 목표 심박수를 출력한다. ("Targetpulse: *target_hearttrate*")

```

123   if (QS == true){           // 심작박동을 감지했을 때
124       lcd.setCursor(0,0);      // lcd 첫번째줄에 심박수 출력
125       lcd.print("Pulse: ");
126       lcd.print(BPM);
127
128       if (BPM > target_hearttrate) {           // 심박수가 목표심박수를 초과했을 때
129           Serial.println("overoverover");
130
131           lcd.setCursor(0,1);                  // lcd 두번째줄에 운동중지 멘트 출력
132           lcd.print("Stop exercise!!!");
133
134           myDFPlayer.play(3);                  // 심박수가 높다는 알림 재생
135           delay(3000);
136           myDFPlayer.play(4);                  // 운동 종료 알림 재생
137           delay(3000);
138
139           openservo();
140           delay(10000);
141           Serial.println("servo end");
142
143           exit(0);
144       }
145
146       if(BPM == 0) {
147           lcd.setCursor(0,1);                  // lcd 두번째줄에 finish 출력
148           lcd.print("check sensor");
149       }
150       else {
151           lcd.setCursor(0,1);                  //lcd 두번째줄에 목표심박수 출력
152           lcd.print("TargetPulse: ");
153           lcd.print(target_hearttrate);
154       }
155
156       Serial.println("2: ");
157       Serial.println(BPM);
158
159       QS = false;                             // 심작박동 감지 여부 false로 설정
160   }
161
162   delay(20);
163   lcd.clear();
164 }

```

심박수를 감지했을 때는 감지된 심박수를 LCD 첫번째 줄에 출력한다. ("Pulse: *BPM*") 심박수가 목표 심박수를 초과했을 때 lcd 두번째 줄에 운동을 중지하라는 메시지를 출력한다. ("Stop exercise")심박수가 높다는 알림과 운동을 종료한다는 알림을 재생하면서 `openservo` 함수 호출하고 시스템을 종료한다.

심박수가 목표심박수를 초과하지 않았다면 BPM이 0일때는 센서 오작동 알림 문구를 LCD의 두번째 줄에 출력하고("check sensor!!!!"), 0이 아니면 목표 심박수를 출력한다. ("Targetpulse: *target_hearttrate*") 심박수 감지여부를 false로 설정하고 LCD 화면을 모두 지운다.

2) set_age_intensity.ino

```

1  void setage() {           // 연령대 설정과 최대 심박수 계산
2      while(1) {           // 연령대 설정
3          upbuttonstate = digitalRead(upbutton);    // 증가버튼 상태
4          downbuttonstate = digitalRead(downbutton); // 감소버튼 상태
5          savebuttonstate = digitalRead(savebutton); // 저장버튼 상태
6
7          if(savebuttonstate==HIGH) {               // 저장버튼을 누르기 시작한 시점
8              if(save_prestate==0) {
9                  delay(10);
10                 save_prestate =1;                 // 저장버튼 이전 상태 '눌림'으로 설정
11             }
12         }
13
14         if(savebuttonstate==LOW) {                 // 저장버튼에서 손을 뗀 시점
15             if(save_prestate==1) {
16                 delay(10);
17                 save_prestate =0;                 // 저장버튼 이전 상태 '안 눌림'으로 설정
18                 break;                           // 반복종료(나이 설정 완료)
19             }
20         }
21     }

```

먼저 setage함수에서 연령대를 입력 받고 최대 심박수를 계산할 것이다. age는 0으로 초기화 되어 있고 저장버튼이 눌릴 때까지 while문으로 증가, 감소버튼의 상태를 파악하여 연령대를 설정한다. 아날로그 3핀에서 디지털로 읽어온 증가버튼의 현재 상태를 변수 upbuttonstate에 저장한다. 아날로그 4핀에서 디지털로 읽어온 감소버튼의 현재 상태를 변수 downbuttonstate에 저장한다. 아날로그 5핀에서 디지털로 읽어온 저장버튼의 현재 상태를 변수 savebuttonstate에 저장한다. 저장버튼의 현재 상태를 나타내는 변수 savebuttonstate이 HIGH(눌림)이고 종료버튼의 이전 상태를 나타내는 변수 save_prestate가 0(안 눌림)이면 버튼이 눌리기 시작한 시점으로 판단하여 save_prestate를 1(눌림)으로 설정한다. 반대로 종료버튼의 현재 상태를 나타내는 변수 savebuttonstate이 LOW(안 눌림)이고 종료버튼의 이전 상태를 나타내는 변수 save_prestate가 1(눌림)이면 버튼에서 손을 떼기 시작한 시점으로 판단하여 save_prestate를 0(안 눌림)으로 설정하고 while문을 탈출한다.

```

22     lcd.setCursor(0,0);          // lcd에 연령대 출력
23     lcd.print("age: ");
24     lcd.print(age);
25
26     if(upbuttonstate==HIGH) {      // 증가버튼을 누르기 시작한 시점
27         if(up_prestate==0) {
28             delay(10);
29             up_prestate = 1;        // 증가버튼 이전 상태 '눌림'으로 설정
30         }
31     }
32
33     if(upbuttonstate==LOW) {        // 증가버튼에서 손을 뗀 시점
34         if(up_prestate==1) {
35             age+=10;                // 나이 10 증가
36             delay(10);
37             up_prestate = 0;        // 증가버튼 이전 상태 '안 눌림'으로 설정
38         }
39     }
40
41     if(downbuttonstate==HIGH) {     // 감소버튼을 누르기 시작한 시점
42         if(down_prestate==0) {
43             delay(10);
44             down_prestate = 1;      // 감소버튼 이전 상태 '눌림'으로 설정
45         }
46     }
47
48     if(downbuttonstate==LOW) {      // 감소버튼에서 손을 뗀 시점
49         if(down_prestate==1) {
50             age-=10;                // 나이 10 감소
51             delay(10);
52             down_prestate = 0;      // 감소버튼 이전 상태 '안 눌림'으로 설정
53         }
54     }
55
56     delay(20);
57     lcd.clear();
58 }
59
60 max_hearttrate = 207-age*0.7;       // 설정된 연령대의 최대심박수 계산
61 Serial.println(max_hearttrate);
62 return;
63 }

```

LCD 첫번째 줄에 나이를 출력한다. ("age: age") 증가버튼의 현재 상태를 나타내는 변수 upbuttonstate이 HIGH(눌림)이고 증가버튼의 이전 상태를 나타내는 변수 up_prestate가 0(안 눌림)이면 버튼이 눌리기 시작한 시점으로 판단하여 up_prestate를 1(눌림)으로 설정한다. 반대로 증가버튼의 현재 상태를 나타내는 변수 upbuttonstate이 LOW(안 눌림)이고 증가버튼의 이전 상태를 나타내는 변수 up_prestate가 1(눌림)이면 버튼에서 손을 떼기 시작한 시점으로 판단하여 age를 10만큼 더하고 up_prestate를 0(안 눌림)으로 설정한다. 감소버튼의 현재 상태를 나타내는 변수 downbuttonstate이 HIGH(눌림)이고 감소버튼의 이전 상태를 나타내는 변수 down_prestate가 0(안 눌림)이면 버튼이 눌리기 시작한 시점으로 판단하여 down_prestate를 1(눌림)으로 설정한다.

반대로 감소버튼의 현재 상태를 나타내는 변수 downbuttonstate이 LOW(안 눌림)이고 감소버튼의 이전 상태를 나타내는 변수 down_prestate가 1(눌림)이면 버튼에서 손을 떼기 시작한 시점으로 판단하여 age를 10만큼 빼고 down_prestate를 0(안 눌림)으로 설정한다.

설정된 연령대와 카보넨 공식을 이용해 최대심박수를 계산한다.

```
70 void setintensity(){ // 운동 강도 설정(퍼센트)과 목표심박수 계산
71     while(1) { // 운동 강도 설정(퍼센트)
72         upbuttonstate = digitalRead(upbutton); // 증가버튼 상태
73         downbuttonstate = digitalRead(downbutton); // 감소버튼 상태
74         savebuttonstate = digitalRead(savebutton); // 저장버튼 상태
75
76         if(savebuttonstate==HIGH) { // 저장버튼을 누르기 시작한 시점
77             if(save_prestate==0) {
78                 delay(10);
79                 save_prestate =1; // 저장버튼 이전 상태 '눌림'으로 설정
80             }
81         }
82
83         if(savebuttonstate==LOW) { // 저장버튼에서 손을 뗀 시점
84             if(save_prestate==1) {
85                 delay(10);
86                 save_prestate =0; // 저장버튼 이전 상태 '안 눌림'으로 설정
87                 break; // 반복종료(운동강도 설정 완료)
88             }
89         }
90     }
```

그 다음 setintensity함수에서 운동강도를 입력 받고 목표 심박수를 계산할 것이다. intensity 변수는 40으로 초기화되어 있고 저장버튼이 눌릴 때까지 while문으로 증가, 감소버튼의 상태를 파악하여 운동강도를 설정한다. 아날로그 3핀에서 디지털로 읽어온 증가버튼의 현재 상태를 변수 upbuttonstate에 저장한다. 아날로그 4핀에서 디지털로 읽어온 감소버튼의 현재 상태를 변수 downbuttonstate에 저장한다. 아날로그 5핀에서 디지털로 읽어온 저장버튼의 현재 상태를 변수 savebuttonstate에 저장한다. 저장버튼의 현재 상태를 나타내는 변수 savebuttonstate이 HIGH(눌림)이고 종료버튼의 이전 상태를 나타내는 변수 save_prestate가 0(안 눌림)이면 버튼이 눌리기 시작한 시점으로 판단하여 save_prestate를 1(눌림)으로 설정한다. 반대로 종료버튼의 현재 상태를 나타내는 변수 savebuttonstate이 LOW(안 눌림)이고 종료버튼의 이전 상태를 나타내는 변수 save_prestate가 1(눌림)이면 버튼에서 손을 떼기 시작한 시점으로 판단하여 save_prestate를 0(안 눌림)으로 설정하고 while문을 탈출한다.

```

91     lcd.setCursor(0,0);          // lcd에 운동강도 출력
92     lcd.print("intensity: ");
93     lcd.print(intensity);
94
95     if(upbuttonstate==HIGH) {      // 증가버튼을 누르기 시작한 시점
96         if(up_prestate==0) {
97             delay(10);
98             up_prestate =1;        // 증가버튼 이전 상태 '눌림'으로 설정
99         }
100    }
101
102    if(upbuttonstate==LOW) {         // 증가버튼에서 손을 뗀 시점
103        if(up_prestate==1) {
104            intensity+=10;          // 운동 강도 10 증가
105            delay(10);
106            up_prestate =0;        // 증가버튼 이전 상태 '안 눌림'으로 설정
107        }
108    }
109
110    if(downbuttonstate==HIGH) {      // 감소버튼을 누르기 시작한 시점
111        if(down_prestate==0) {
112            delay(10);
113            down_prestate =1;       // 감소버튼 이전 상태 '눌림'으로 설정
114        }
115    }
116
117    if(downbuttonstate==LOW) {       // 감소버튼에서 손을 뗀 시점
118        if(down_prestate==1) {
119            intensity-=10;          // 운동 강도 10 감소
120            delay(10);
121            down_prestate =0;       // 감소버튼 이전 상태 '안 눌림'으로 설정
122        }
123    }
124
125    delay(20);
126    lcd.clear();
127 }
128
129 target_hearttrate = max_hearttrate * intensity * 0.01;    // 설정된 운동강도에 따른 목표심박수 계산
130 Serial.println(target_hearttrate);
131 return;
132 }
133 }

```

LCD 첫번째 줄에 운동강도를 출력한다. ("intensity: *intensity*") 증가버튼의 현재 상태를 나타내는 변수 upbuttonstate이 HIGH(눌림)이고 증가버튼의 이전 상태를 나타내는 변수 up_prestate가 0(안 눌림)이면 버튼이 눌리기 시작한 시점으로 판단하여 up_prestate를 1(눌림)으로 설정한다. 반대로 증가버튼의 현재 상태를 나타내는 변수 upbuttonstate이 LOW(안 눌림)이고 증가버튼의 이전 상태를 나타내는 변수 up_prestate가 1(눌림)이면 버튼에서 손을 떼기 시작한 시점으로 판단하여 intensity를 10만큼 더하고 up_prestate를 0(안 눌림)으로 설정한다. 감소버튼의 현재 상태를 나타내는 변수 downbuttonstate이 HIGH(눌림)이고 감소버튼의 이전 상태를 나타내는 변수 down_prestate가 0(안 눌림)이면 버튼이 눌리기 시작한 시점으로 판단하여 down_prestate를 1(눌림)으로 설정한다. 반대로 감소버튼의 현재 상태를 나타내는 변수 downbuttonstate이 LOW(안 눌림)이고 감소버튼의 이전 상태를 나타내는 변수 down_prestate가 1(눌림)이면 버튼에서 손을 떼기 시작한 시점으로 판단하여 intensity를 10만큼 빼고 down_prestate를 0(안 눌림)으로 설정한다.

최대 심박수에 운동강도*0.01을 곱해서 목표 심박수를 계산한다.

3) interrupt.ino

```

1  volatile int rate[10];           // 가장 마지막 10개의 ibi를 저장
2  volatile unsigned long sampleCounter = 0;    // 심장박동 시간을 결정할 때 사용
3  volatile unsigned long lastBeatTime = 0;     // 가장 마지막 박동시간, ibi계산 시 사용
4  volatile int P = 512;           // 심장박동 펄스 파형의 마루
5  volatile int T = 512;           // 심장박동 펄스 파형의 골
6  volatile int thresh = 530;      // 심장박동의 순간을 찾는 데 사용
7  volatile int amp = 0;           // 펄스 파형의 진폭을 유지하는데 사용
8  volatile boolean firstBeat = true; // 합리적인 BPM으로 시작할 수 있도록 속도 배열을 시드하는 데 사용됨
9  volatile boolean secondBeat = false; // 합리적인 BPM으로 시작할 수 있도록 속도 배열을 시드하는 데 사용됨
10
11 void interruptSetup(){ // 2ms마다 인터럽트 발생
12     TCCR2A = 0x02;      // 디지털 핀 3 및 11에서 PWM을 비활성화하고 CTC 모드로 전환
13     TCCR2B = 0x06;      // 강제 비교 금지, 256 프리스케일러
14     OCR2A = 0x7C;       // 500Hz 샘플 속도의 경우 카운트 상단을 124로 설정
15     TIMSK2 = 0x02;      // TIMER2와 OCR2A 간의 일치 시 인터럽트 활성화
16     sei();              // 인터럽트 활성화
17 }
18
19
20 ISR(TIMER2_COMPA_vect){ // Timer2가 124로 카운트될 때 트리거됨
21     cli();              // 인터럽트 비활성화
22
23     Signal = analogRead(pulsePin); // 심박수센서에서 신호를 읽음
24     sampleCounter += 2; // samplecounter 변수에 2ms 더함
25     int N = sampleCounter - lastBeatTime; // 마지막 박동 후 시간 확인
26
27     if(Signal < thresh && N > (IBI/5)*3){ // 마지막 IBI의 3/5를 대기하여 이색성 노이즈를 방지
28         if (Signal < T){ // 가장 낮은 값 trough에 저장
29             T = Signal;
30         }
31     }
32
33     if(Signal > thresh && Signal > P){ // thresh는 노이즈를 줄여줌
34         P = Signal; // 가장 높은 값 peak에 저장
35     }
36 }

```

일정시간마다 심박수를 측정하기 위해 인터럽트를 이용한다. 먼저 심박수를 측정하기 위한 여러 변수들을 선언 및 초기화한다. interruptSetup 함수를 이용하여 2ms마다 ISR을 실행해 심박수를 측정한다. 아날로그 0핀에서 신호를 받아 Signal에 저장하고 samplecounter에 시간을 저장하기 위해 2ms를 더한다. 그리고 N에 마지막 박동 후로부터의 시간을 저장한다. Signal이 갖는 가장 낮은 값을 T에 저장한다. 이것이 펄스 파형의 골이다. 반대로 가장 큰 값을 P에 저장하고 이 값이 마루에 해당하는 값이다.


```

37 if (N > 250) {
38     if ( (Signal > thresh) && (Pulse == false) && (N > (IBI/5)*3) ){ // signal이 thresh보다 크고 pulse가 false이고 N>ibi의 3/5라면
39         Pulse = true; // pulse에 true 저장
40
41         IBI = sampleCounter - lastBeatTime; // ibi에 마지막 박동 후 시간 저장
42         lastBeatTime = sampleCounter; // lastbeattime에 시간 저장, 다음 펄스에서 이용
43
44         if(secondBeat) { // 두번째 박동을 감지한 경우
45             secondBeat = false;
46             for(int i=0; i<=9; i++){ // rate 배열에 ibi 저장
47                 rate[i] = IBI;
48             }
49         }
50
51         if(firstBeat){ // 첫 박동 감지한 경우
52             firstBeat = false;
53             secondBeat = true;
54             sei(); // 인터럽트 활성화
55             return; // ibi 값 버림
56         }
57
58         word runningTotal = 0; // ibi 평균 저장할 변수
59         for(int i=0; i<=8; i++){ // rate에 저장된 ibi 한칸씩 미루고 runningtotal에 더함
60             rate[i] = rate[i+1];
61             runningTotal += rate[i];
62         }
63         rate[9] = IBI;
64         runningTotal += rate[9];
65         runningTotal /= 10; // 10개의 ibi 평균 구함
66
67         int d = 60000/runningTotal - BPM; // 새로운 bpm과 가장 최근 bpm의 차 구함
68         if (d < 0) {
69             d *= -1;
70         }
71
72         if (d < 100) { // bpm이 100이상 변할 때는 잘못된 심박수가 측정된 것으로 판단하고 제외시킴
73             BPM = 60000/runningTotal; // runningtotal에 저장된 평균 ibi를 이용해 bpm을 계산
74             QS = true; // qs에 true저장
75         }
76     }
77 }
78 }
79

```

마지막 박동으로부터 250ms가 지나고 펄스가 감지되었을 때 Pulse를 true로 바꾸고 IBI에 심장박동 시간 간격을 저장한다. lastBeatTime을 현재 시간으로 저장하고 secondBeat인 경우 rate 배열에 IBI를 저장하고 firstBeat인 경우에는 IBI값을 버린다. rate배열에는 가장 마지막에 측정된 10개의 IBI 값을 저장하고 이 값들의 평균을 구해서 BPM을 계산하는데 사용한다. runningTotal에 평균값을 저장하고 BPM은 60000ms에서 runningTotal을 나누면 된다. 하지만 이렇게 구한 BPM이 이전의 BPM과 100이상 차이 날 경우 오차가 큰 것으로 판별하고 그 값을 버린다.

```

80 if (Signal < thresh && Pulse == true){ // 심박수센서의 signal이 thresh보다 낮고 pulse가 true 이면
81     Pulse = false; // pulse에 false저장
82     amp = P - T; // amp에 진폭 저장
83     thresh = amp/2 + T; // thresh에 peek와 trough의 중간지점 저장
84     P = thresh;
85     T = thresh;
86 }
87
88 if (N > 2500){ // 2.5초가 지나도 박동이 감지 되지 않은 경우
89     thresh = 530; // 각 변수 기본값을 다시 설정
90     P = 512;
91     T = 512;
92     lastBeatTime = sampleCounter;
93     firstBeat = true;
94     secondBeat = false;
95 }
96
97 sei(); // 인터럽트 활성화
98 } // end isr

```

Signal이 thresh보다 작는데 Pulse가 true이면 Pulse를 false로 바꾸고 thresh, P, T를 P와 T의 중간값으로 저장한다. 만약 2.5초가 지나도 박동이 감지되지 않으면 변수를 초기화값으로 리셋시킨다.

4) servo.ino

```
1 void openservo()           // 약상자 뚜껑 열기
2 {
3     Serial.println("servo start");
4     servo.attach(SERVO);
5
6     for (int i=180;i>=30;i-=2){           // 뚜껑 천천히 열기
7         servo.write(i);
8         delay(30);
9     }
10    delay(2000);
11
12    while(1) {
13        endbuttonstate = digitalRead(endbutton);    // 종료 버튼 상태
14
15        if(endbuttonstate==HIGH) {    // 버튼을 누르기 시작한 시점
16            if(end_prestate==0) {
17                delay(10);
18                end_prestate =1;    // 버튼 이전 상태 '눌림'으로 설정
19            }
20        }
21
22        if(endbuttonstate==LOW) {    // 버튼에서 손을 뗀 시점
23            if(end_prestate==1) {
24                delay(10);
25                end_prestate =0;    // 버튼 이전 상태 '안 눌림'으로 설정
26                break;    // 반복 종료
27            }
28        }
29    }
30 }
31
32 for (int i=30;i<=180;i+=2){           // 뚜껑 천천히 닫기
33     servo.write(i);
34     delay(30);
35 }
36 delay(2000);
37
38 servo.detach();
39 return;
40
41 }
```

loop함수로부터 openservo함수가 호출되면 서보모터가 움직인다. 먼저 약 상자 뚜껑을 천천히 열기 위해 서보모터가 180도에서 30도가 될 때까지 2도 회전하고 30ms 대기하고를 반복한다. 그 후 종료버튼을 누르면 약 상자 뚜껑을 닫게 하기 위해 아날로그 2핀에서 디지털로 읽어온 종료버튼의 현재 상태를 변수 endbuttonstate에 저장한다. 종료버튼의 현재 상태를 나타내는 변수 endbuttonstate이 HIGH(눌림)이고 종료버튼의 이전 상태를 나타내는 변수 end_prestate가 0(안 눌림)이면 버튼이 눌리기 시작한 시점으로 판단하여 end_prestate를 1(눌림)으로 설정한다. 반대로 종료버튼의 현재 상태를 나타내는 변수 endbuttonstate이 LOW(안 눌림)이고 종료버튼의 이전 상태를 나타내는 변수 end_prestate가 1(눌림)이면 버튼에서 손을 떼기 시작한 시점으로 판단하여 end_prestate를 0(안 눌림)으로 설정하고 while문을 탈출한다. 그리고 뚜껑을 천천히 닫기 위해서 서보모터가 180도가 될 때까지 2도 회전하고 30ms 대기하고를 반복한다.

6. 성능시험 및 결과

1) 시험 환경 설정

장치의 성능을 테스트하기 위한 시험을 구상을 할 때 이 장치는 심장 질환을 앓고 있는 환자들을 대상으로 재활 운동을 보조하며 심박수를 측정하는 등의 기능을 하기위해 제작하였지만 기기의 안정성, 정확도가 입증되지 않은 상황에서 환자를 대상으로 테스트 하는 것은 위험성이 우려되어 실험대상자로 20대 남성을 선정하였다. 운동강도는 60%, 80%로 설정하고 두 번의 테스트를 진행한다. 60%로 설정한 경우는 운동 중 심박수가 목표 심박수 115를 초과했을 때를 가정하여 약 상자가 동작하는 것까지 테스트할 것이고 80%로 설정한 경우는 목표 심박수 154를 임의로 큰 값으로 설정하고 10분동안 운동하며 심박수가 제대로 측정되는지를 확인할 것이다. 운동시간은 10분이며 심박수 센서를 착용한 채로 사이클을 활용한 유산소운동을 진행한다. 또한 심박수가 얼마나 정확한 값으로 측정되는지 테스트하기위해 스마트 워치를 함께 착용하여 값을 비교한다.

2) 성능 평가 기준

테스트는 운동강도에 따라 60%와 80%로 두 번 진행한다. 60%일 때는 장치를 사용하기 시작하면서부터 목표 심박수를 초과하여 운동을 중지하는 경우이며 80%는 목표 심박수를 초과하지 않고 버튼을 눌러 운동을 종료하는 경우이다. 각 실험에서의 테스트할 성능은 다음과 같다.

(1) 운동강도가 60%인 경우

No.	진행 상태	기준	결과
1	운동 전	연령대, 운동강도설정 알림이 울리는가	-
2		LCD에 연령대, 운동강도 설정을 위한 화면이 잘 동작하는가	-
3		연령대, 운동강도 설정용 버튼스위치가 잘 인식되는가	-
4		운동시작 알림이 울리는가	-
5	운동중	측정된 심박수가 LCD에 출력되는가	-
6		측정된 심박수가 스마트 워치와 비교해서 얼마나 정확한가	-
7		심박수가 0일 때 check sensor가 출력되는가	-
8		심박수가 0이 아닐 때 목표 심박수 출력되는가	-
9		운동 중에 심박수가 목표 심박수를 초과하면 stop exercise가 출력되는가	-
10		운동 중에 심박수가 목표 심박수를 초과하면 심박수가 높다는 것과 운동을 종료한다는 알림이 울리는가	-
11		운동 중에 심박수가 목표 심박수를 초과하면 약 상자가 열리는가	-
12		운동 중에 심박수가 목표 심박수를 초과하여 약 상자가 열리고 종료 버튼을 눌렀을 때 약 상자가 닫히는가	-

(2) 운동강도가 80%인 경우

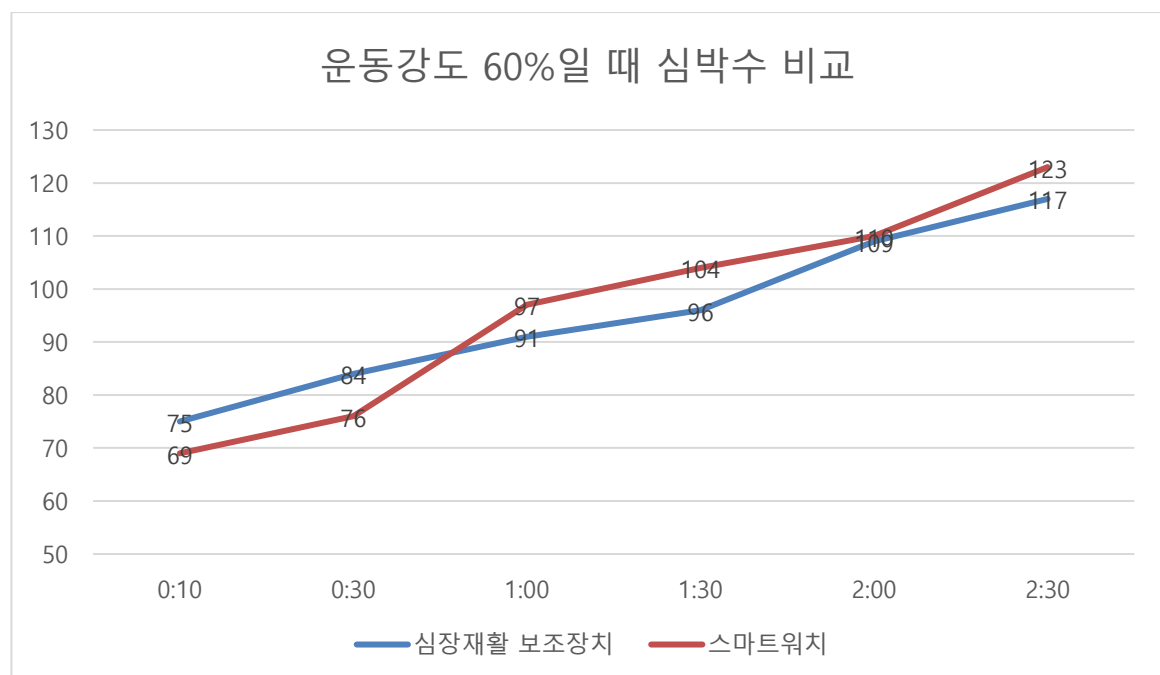
No.	진행 상태	기준	결과
1	운동 전	연령대, 운동강도설정 알림이 울리는가	-
2		LCD에 연령대, 운동강도 설정을 위한 화면이 잘 동작하는가	-
3		연령대, 운동강도 설정용 버튼스위치가 잘 인식되는가	-
4		운동시작 알림이 울리는가	-
5	운동중	측정된 심박수가 LCD에 출력되는가	-
6		측정된 심박수가 스마트 워치와 비교해서 얼마나 정확한가	-
7		심박수가 0일 때 check sensor가 출력되는가	-
8		심박수가 0이 아닐 때 목표 심박수 출력되는가	-
9		종료버튼이 눌리면 finish가 출력되는가	-
10		종료버튼이 눌리면 운동종료 알림이 울리는가	-

3) 시험 결과


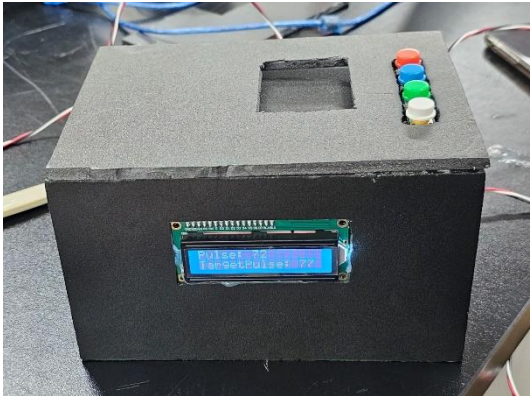
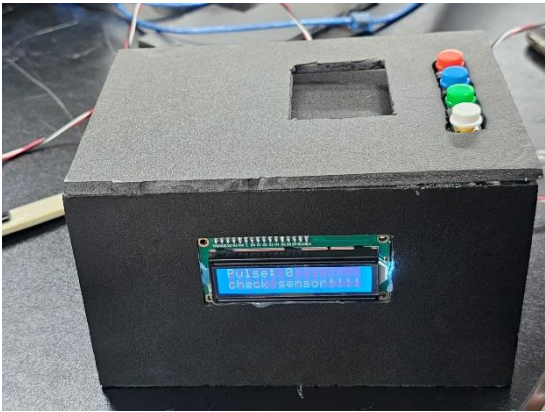
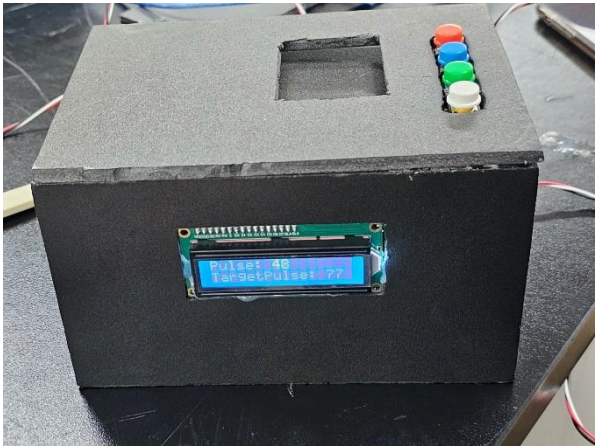
(1) 운동강도가 60%인 경우

No.	진행 상태	기준	결과
1	운동 전	연령대, 운동강도설정 알림이 울리는가	Y
2		LCD에 연령대, 운동강도 설정을 위한 화면이 잘 동작하는가	Y
3		연령대, 운동강도 설정용 버튼스위치가 잘 인식되는가	Y
4		운동시작 알림이 울리는가	Y
5	운동중	측정된 심박수가 LCD에 출력되는가	Y
6		측정된 심박수가 스마트 워치와 비교해서 얼마나 정확한가	도표
7		심박수가 0일 때 check sensor가 출력되는가	Y
8		심박수가 0이 아닐 때 목표 심박수 출력되는가	Y
9		운동 중에 심박수가 목표 심박수를 초과하면 stop exercise가 출력되는가	Y
10		운동 중에 심박수가 목표 심박수를 초과하면 심박수가 높다는 것과 운동을 종료한다는 알림이 울리는가	Y
11		운동 중에 심박수가 목표 심박수를 초과하면 약 상자가 열리는가	Y
12		운동 중에 심박수가 목표 심박수를 초과하여 약 상자가 열리고 종료 버튼을 눌렀을 때 약 상자가 닫히는가	Y

스마트 워치에서 측정된 값과 심박수를 비교하였다. 일정시간마다 변화하는 심박수를 그래프로 나타내었다.

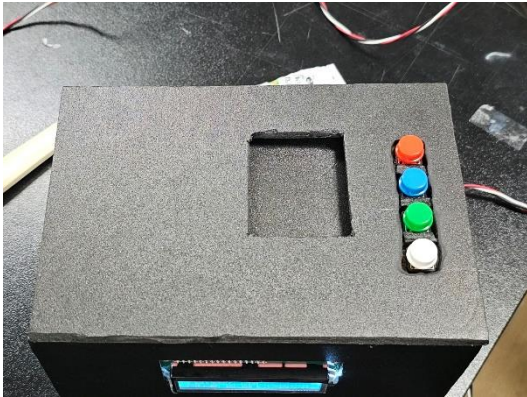


오차율은 6.482%로 나타났다. 스마트 워치는 측정되는 심박수가 바로 출력되지 않고 심박수 값이 도출되는 데까지 10초정도의 시간이 소요된다. 스마트 워치에서 심박수가 출력된 시점의 LCD의 심박수를 읽었지만 측정값 도출에서의 차이가 나기 때문에 도출된 오차율은 이로 인한 것이라고 예상된다. 아래는 테스트 항목별 동작 사진이다. LCD와 약 상자 개폐가 제대로 진행되었다.

항목 2	항목 5
	
항목 7	항목 8
	
항목 9	항목 11



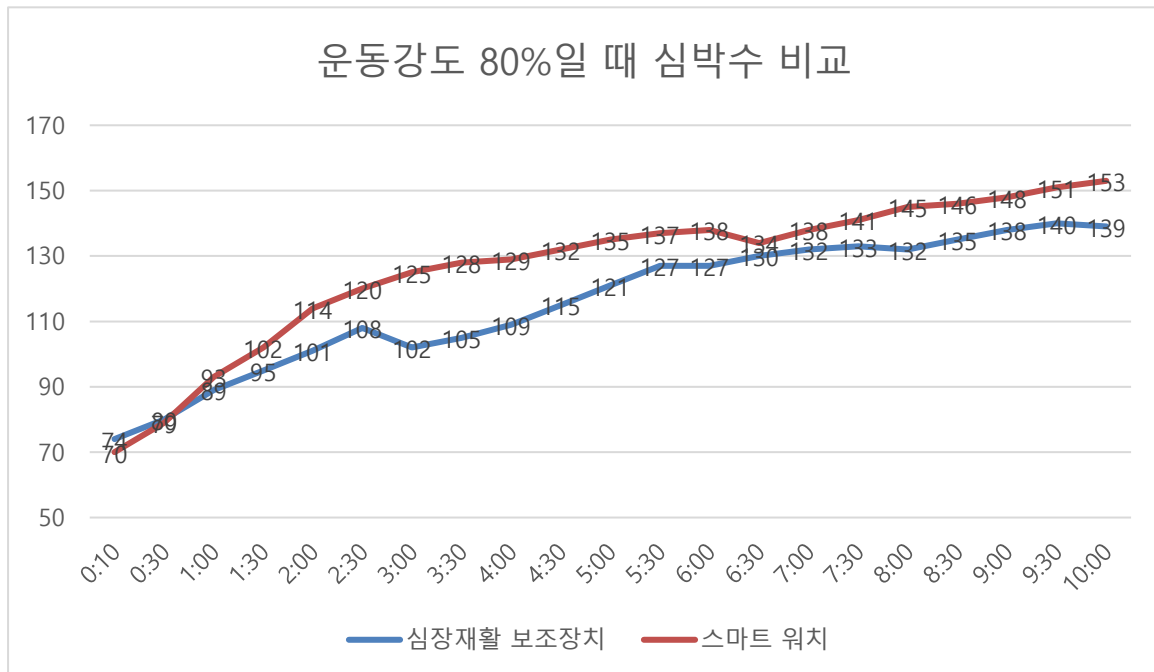
항목 12



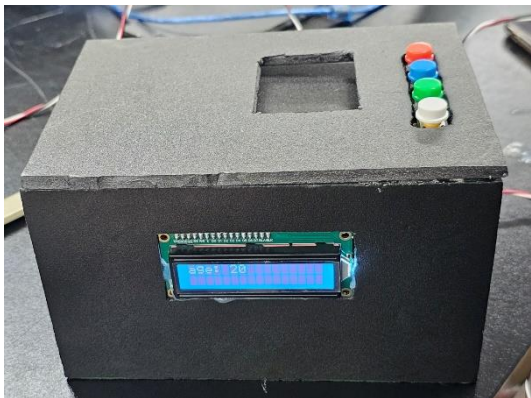
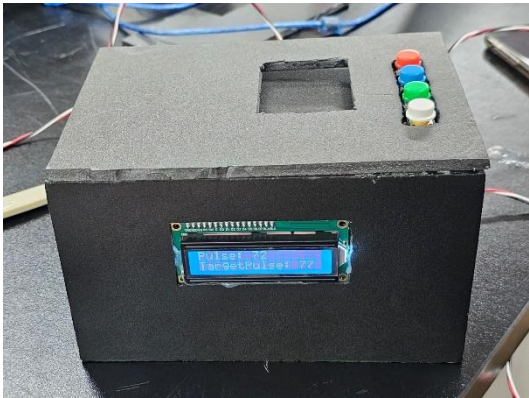
(2) 운동강도가 80%인 경우

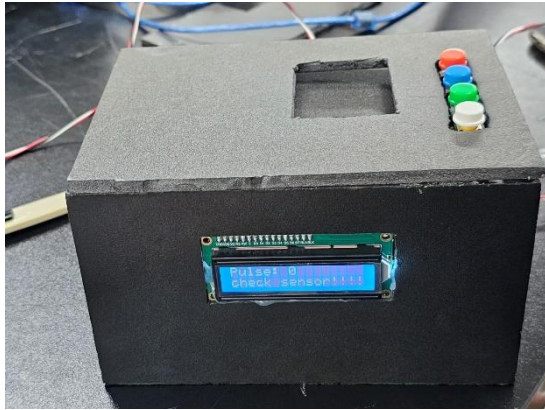
No.	진행 상태	기준	결과
1	운동 전	연령대, 운동강도설정 알림이 울리는가	Y
2		LCD에 연령대, 운동강도 설정을 위한 화면이 잘 동작하는가	Y
3		연령대, 운동강도 설정용 버튼스위치가 잘 인식되는가	Y
4		운동시작 알림이 울리는가	Y
5	운동중	측정된 심박수가 LCD에 출력되는가	Y
6		측정된 심박수가 스마트 워치와 비교해서 얼마나 정확한가	도표
7		심박수가 0일 때 check sensor가 출력되는가	Y
8		심박수가 0이 아닐 때 목표 심박수 출력되는가	Y
9		종료버튼이 눌리면 finish가 출력되는가	Y
10		종료버튼이 눌리면 운동종료 알림이 울리는가	Y

스마트 워치에서 측정된 심박수 값과 비교하여 일정시간마다 변화하는 심박수를 그래프로 나타내었다.

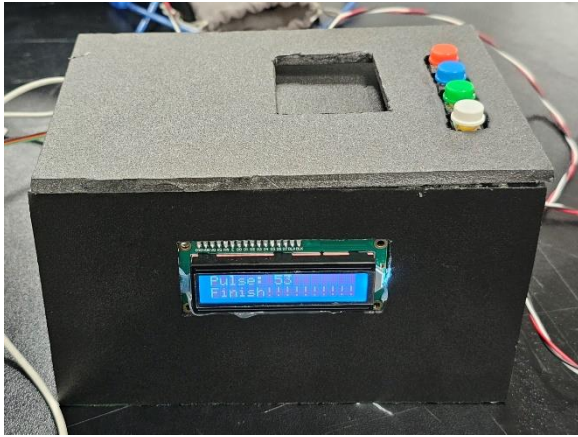


오차율은 8.316%로 나타났다. 앞선 운동강도 60% 실험과 마찬가지로 스마트 워치는 측정되는 심박수가 바로 출력되지 않고 심박수 값이 도출되는 데까지 10초정도의 시간이 소요되기 때문에 스마트 워치에서 심박수가 출력된 시점의 LCD의 심박수를 읽었지만 측정값 도출에서의 차이가 날 수 있다. 또한 운동 진행 중에 심박수 센서가 흔들리면서 오차가 발생한 것으로 예상된다. 아래는 테스트 항목별 사진이다.

항목 2	항목 5
	
항목 7	항목 8



항목 9



7. 기대효과

본 심장재활 보조장치는 세가지의 효과를 지닌다. 첫번째는 본인의 신체상태를 체크하며 운동을 조절하며 할 수 있다는 것이고 두번째는 응급상황에 약을 제때 복용할 수 있다는 점이 있다.

심장질환자들은 과도한 운동을 지양해야 한다. 질병으로 인해 약해진 심장이 운동을 하며 건강이 더 악화될 수 있기 때문이다. 재활센터에서는 의료장비를 착용하고 의료진이 실시간으로 심전도, 혈압 등을 확인하며 운동을 지도하지만 가정에서의 재활은 그렇지 못하다. 더불어 환자들은 스스로 운동강도를 조절하는 것에 불안감을 느낀다고 한다. 이 심장재활 보조장치는 이런 점을 해결해줄 수 있다. 운동강도가 약해 운동 효과를 제대로 내지 못하지도 운동강도가 지나쳐 건강에 해를 끼치지도 않게 도와준다. 운동을 진행하는 동안의 심박수를 실시간으로 체크하며 운동강

도를 조절한다. 단순히 사람이 느끼는 운동의 강도로 판단하는 것이 아니라 기기로 측정한 심박수로 판단하기 때문에 신뢰성이 있으며 안심하고 운동을 할 수 있다.

마지막으로 심장질환자들은 응급상황이 발생했을 때 빠르게 약을 복용하고 병원에서 치료를 받는 것이 중요하다. 제때에 응급처치가 이루어지지 않으면 환자가 사망에 이를 수 있다. 그렇기에 운동 중 측정되는 심박수로 목표 심박수를 초과하며 과도한 운동을 시행했을 때 약상자가 열려 혈관확장제를 복용하게 유도한다. 환자가 쓰러지기 전 운동을 멈추고 약을 복용하여 응급상황에 대처할 수 있다.

III. 결론

본 프로젝트에서는 심장재활의 중요성을 인지하여 가정에서 환자가 스스로 재활을 할 수 있는 장치를 제작하였다. 버튼 스위치로 연령대와 운동강도를 설정하고 운동을 하면서 센서로 심박수를 측정할 수 있도록 하는 시스템이다. 만약 심박수에 이상 수치가 뜬다면 운동 종료를 알려주면서 운동을 종료하고 혈관확장제를 즉시 복용할 수 있도록 설계되었다. 또한 재활 과정을 진행할 때 스피커가 기본 설정 알림, 운동의 시작과 종료를 알려준다.

장치 제작 후 성능테스트 결과 모든 음원이 정상적으로 재생되었다. LCD 출력도 양호했으며 푸쉬 버튼, 서보 모터도 정상 작동하였으며 심박수도 운동 강도 60%일 때, 오차율 6.482%와 운동강도 80%일 때, 오차율 8.316%로 높은 정확도를 보였다. 하지만 심박수의 경우 심박수가 측정되었을 때의 정확도는 높지만 센서를 착용하는 위치에 따라 심박수를 아예 인식하지 못하는 경우가 발견되었다. 따라서 심박수 센서를 여러 개 사용하여 맥이 감지되는 여러 신체부위에 부착하고 맥박 감지의 정확도를 높이면 더 정확한 결과가 나올 것으로 예상된다. 또한 현재는 밴드에 심박수 센서를 연결하여 손목에 고정하여 측정했지만 집게를 이용하여 손가락 끝에 고정하거나 의료용 테이프를 이용하여 귓볼에 부착하는 등의 방법으로 센서를 확실하게 고정하면 더 정확한 값을 얻을 수 있을 것이다.

또한 본 심장재활 보조장치에 측정한 심박수 데이터를 기록하고 축적하여 의료진에게 전송하는 기능을 추가하면 환자의 건강 회복에 더 탁월할 것을 예상된다. 데이터 분석 및 통신 기능은 재활 프로그램의 효과를 향상시킬 수 있을 것이다.

지속적인 모니터링 및 데이터 수집을 통해 비정상적인 심박수 패턴 또는 예상 회복 궤적의 편차를 감지할 수 있다. 이러한 잠재적인 문제의 조기 발견은 의료진의 신속한 개입 및 조정을 가능하게 하고 환자에게 맞춤형 치료를 제공할 수 있을 것이다. 의료진은 특정 개입이나 운동이

개별 환자의 심박수에 어떤 영향을 미치는지 더 깊이 이해할 수 있고, 이 데이터는 치료, 운동 강도, 기간 또는 빈도의 수정을 안내하여 보다 효과적이고 맞춤형 재활 프로그램으로 이어질 수 있다.

이 장치를 활용한다면 사용자들이 재활센터가 아닌 평시에도 심혈관에 큰 무리가 가지 않으면서 재활을 할 수 있을 것으로 예측된다. 광진구의 노약자 분들이 이 프로그램을 사용한다면 조금 더 건강 증진이 될 수 있을 것이다.

IV. 참고문헌

통계청(2022), 2021년 사망원인통계 결과

양영구, "심근경색 재발 예방, 레파타 적극 고려해야", 메디컬업저버, 2022.03.17., <http://www.monews.co.kr/news/articleView.html?idxno=311089>

안경진, "[건강 팁] 사망 위험 높은 '심혈관질환' 치료 후 심장재활은 선택 아닌 필수", 서울경제, 2023.02.02., <https://www.sedaily.com/NewsView/29LLFNV26R>

정혜선(2002), "심장재활 교육프로그램이 심근 경색증 환자의 질병관련 지식과 건강행위 이행에 미치는 효과", 대한간호학회지, 제32권제1호, 보건연구정보센터, 50-61.

정민영, 최혜란, 안정민(2020), 심장 재활 프로그램 참여 횟수와 재활 효과의 상관 관계 분석, Korean Journal of Family Practice, 10(6), 448-455.

부산대학교병원 심뇌혈관센터, <https://vas.pnuh.or.kr/TCF/cb/contents/view.do?menuIdx=45&id=4>, 심장재활, 2023.03.23.

삼성서울병원 예방재활센터, 심장질환 환자에서 운동 시 주의할 사항, http://www.samsunghospital.com/dept/main/index.do?DP_CODE=XB301&MENU_ID=002010, 2023.03.23