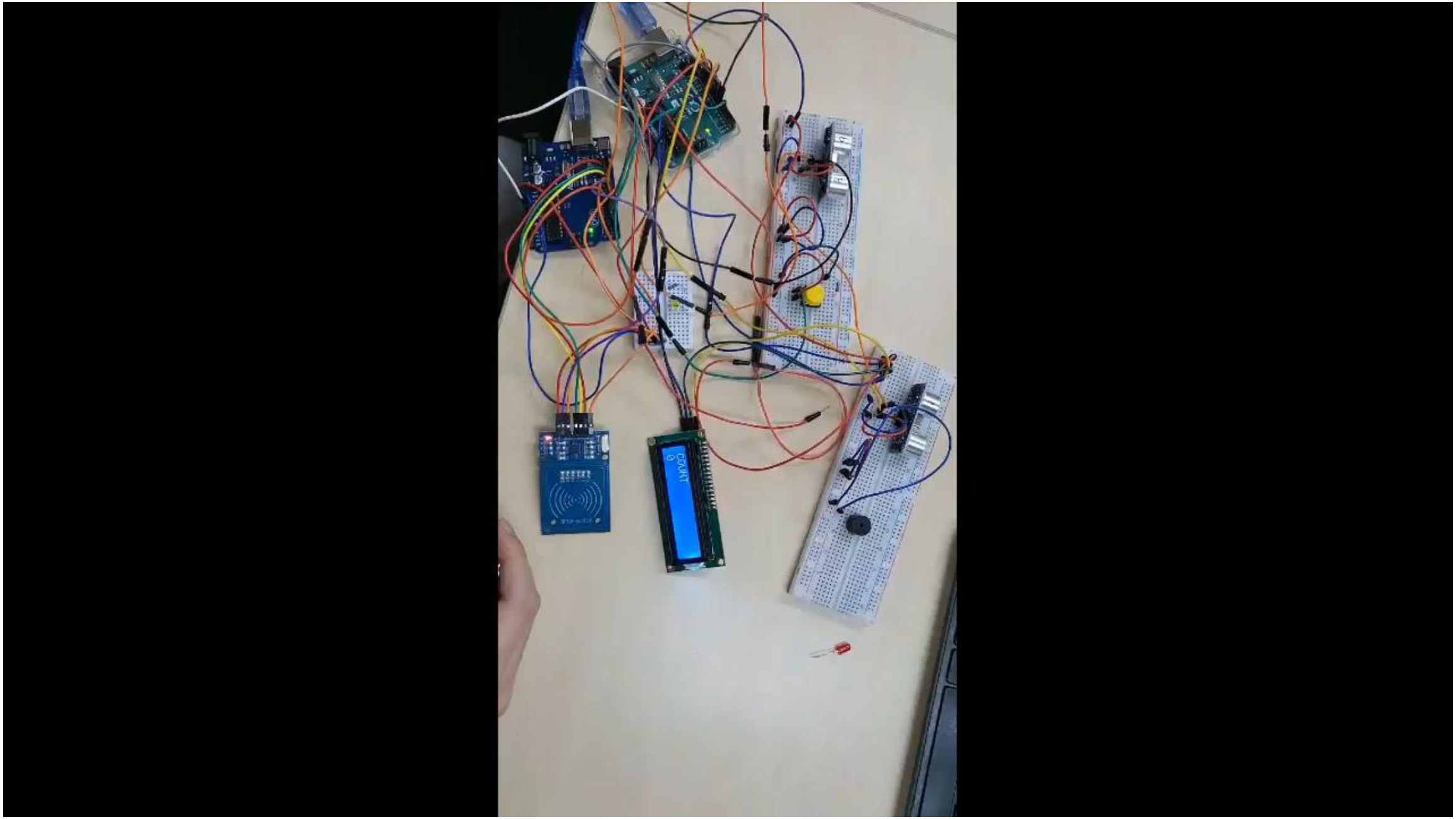
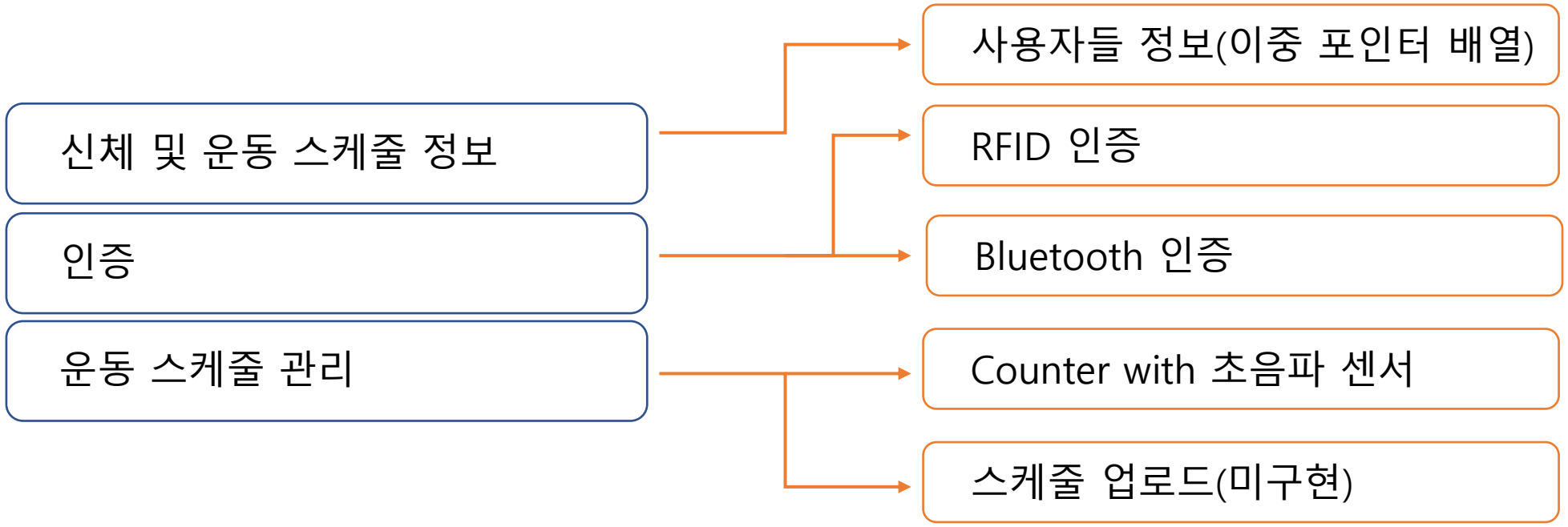
Fitness Management

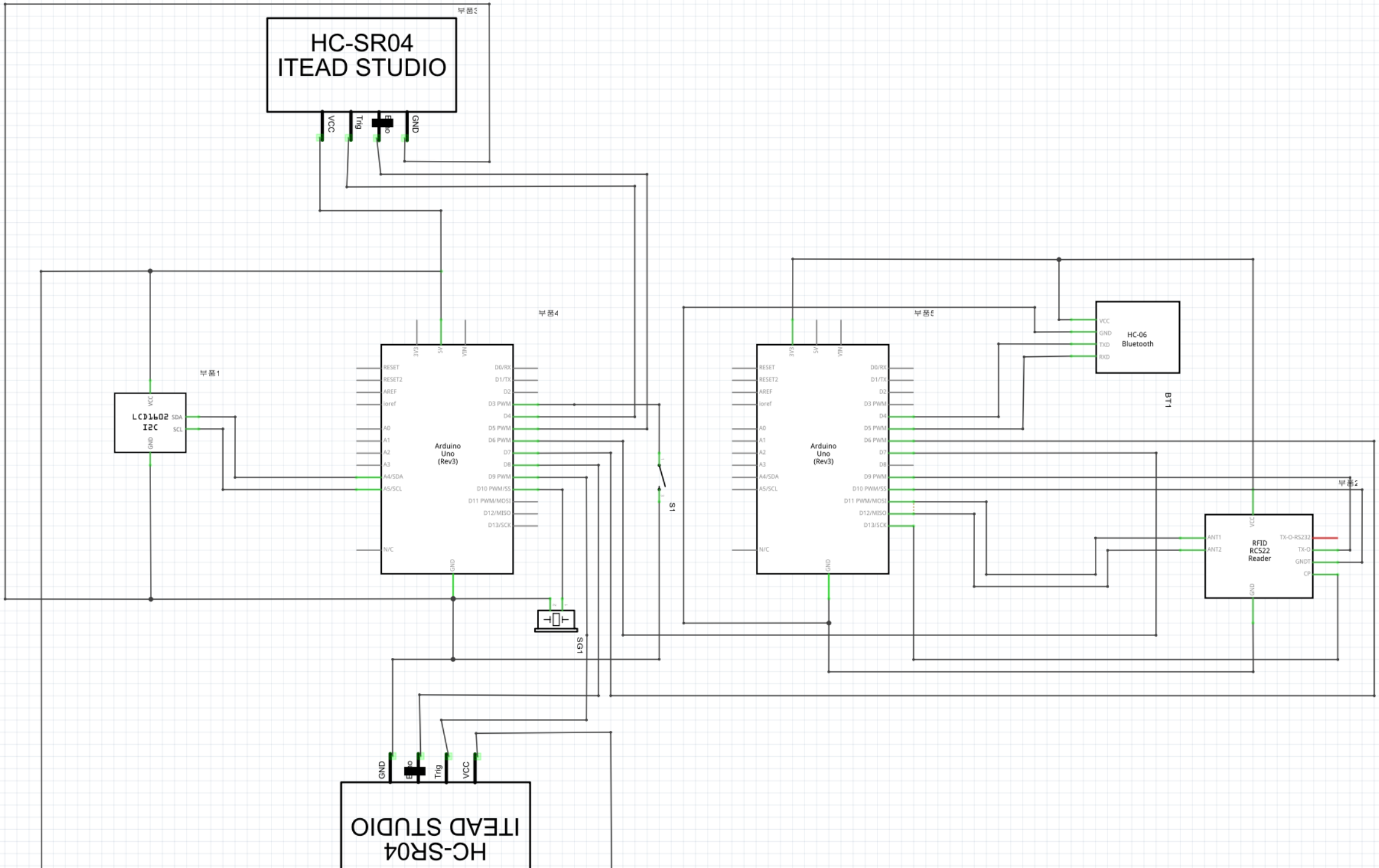
조용현 / 오석준

목차

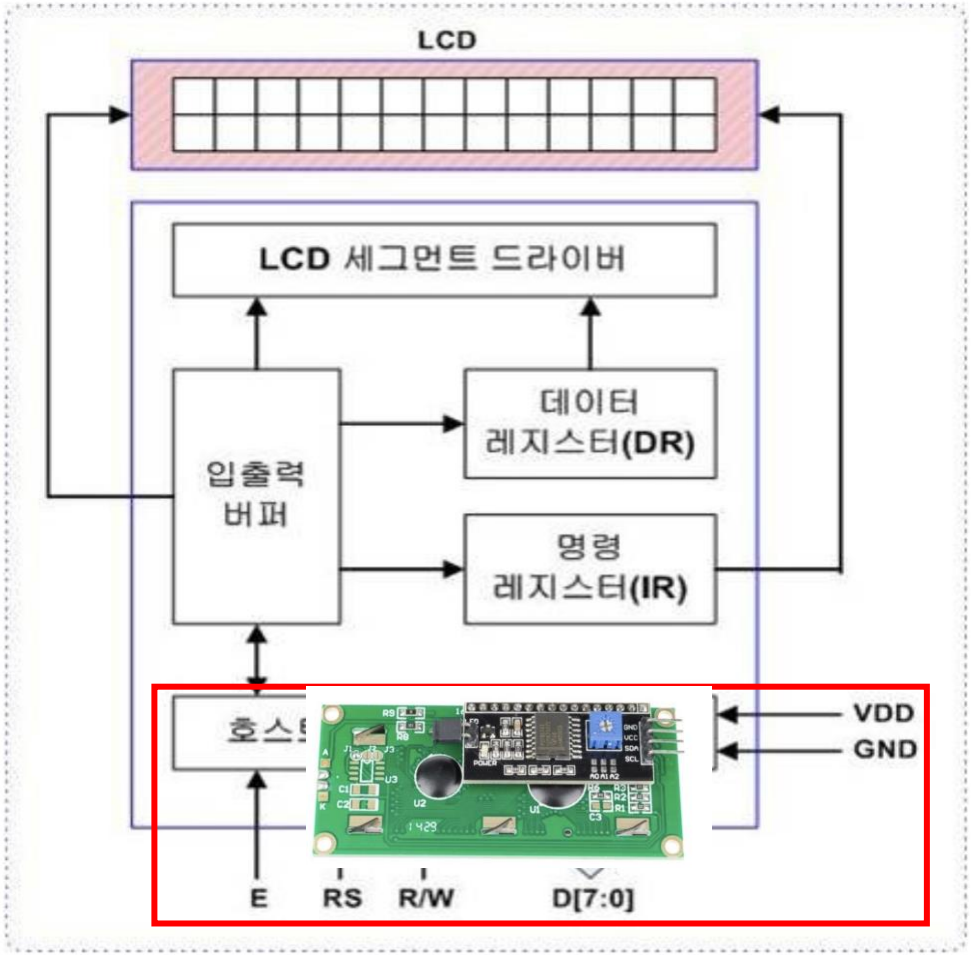


아이템 목적 / 방향


신체 및 운동 스케줄 정보
인증
운동 스케줄 관리
사용자들 정보(이중 포인터 배열)
RFID 인증
Bluetooth 인증
Counter with 초음파 센서
스케줄 업로드(미구현)


HC-SR04
ITEAD STUDIO
VCC
Trig
Echo
GND
부품1
LCD1602
I2C
SDA
SCL
부품4
Arduino
Uno
(Rev3)
RESET
RESET2
AREF
ioref
A0
A1
A2
A3
A4/SDA
A5/SCL
N/C
D0/RX
D1/TX
D2
D3 PWM
D4
D5 PWM
D6 PWM
D7
D8
D9 PWM
D10 PWM/SS
D11 PWM/MOSI
D12/MISO
D13/SCK
S1
SG1
부품5
HC-06
Bluetooth
BT1
RFID
RC522
Reader
ANT1
ANT2
TX-O-RS232
TX-O
GNDT
CP
부품2
HC-SR04
ITEAD STUDIO

Transmission



Fitness Counter

<Logic>

```
void tb_count(long distance1, long distance2){
  if (bottom > 0 && distance2 < 20) // top increase
  {
    top++;
    tone(10, 262, 50); // buzzer
    delay(50);
    if (top > 0 && bottom > 0) {  //count
      count++;
      top = 0;
      bottom = 0;
      if (count == max_count) { //personal target value
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Break Time");
        delay(3000);
        count = 0;
      }
    }
  }
  if (distance1 < 20) { // bottom count increase
    bottom++;
    tone(10, 294, 50);
    delay(50);
  }
}
```

Fitness Counter

<Interrupt>

```c
attachInterrupt(digitalPinToInterrupt(Button), COUNT_IN, FALLING);

void COUNT_IN() {
  interrupts(); // To enable all disabled interrupts
  cur_time = millis();
  if(cur_time-pre_time>=400){
    count = 0;
    bottom = 0;
    top = 0;
    lcd.clear();
    lcd.setCursor(0,0); // LCD position
    lcd.print("RESET");
    delay(2000); //2 seconds print
    lcd.clear();//important
    pre_time=cur_time;//with compare
  }
}
```

CLCD

<Library & address>

```
#include <LiquidCrystal_I2C.h> // lcd
LiquidCrystal_I2C lcd(0x3F, 16, 2);  // LCD address

void setup() {
lcd.init();           // lcd reset
lcd.backlight();      // lcd backlight on
```

CLCD

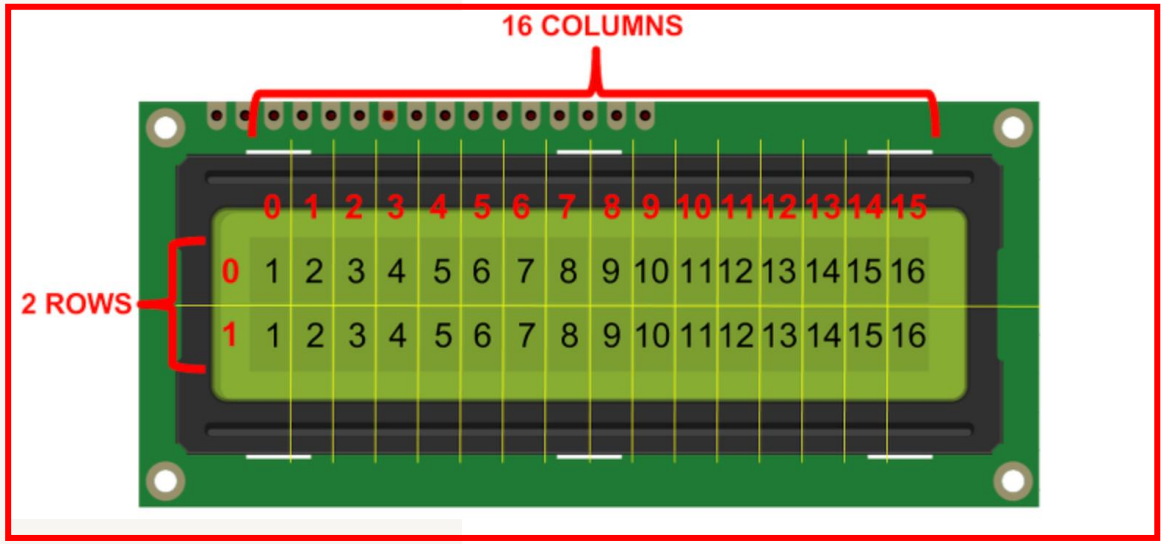
```
void Split_data(){
  char str[100];
  char *sArr[6];
  int index=0 , i = 0;
  char * ptr = NULL;
  if(P_Serial.available()){
    while(P_Serial.available()){
      str[index++] = P_Serial.read();
    }
    str[index]='\0';

    ptr = strtok(str, "#");

  while(ptr != NULL)
  {
    sArr[i++] = ptr;
    ptr = strtok(NULL, "#");

  }
```

```
    String a = sArr[0];
    String b = sArr[1];
    String c = sArr[2];
    String d = sArr[3];
    String e = sArr[4];
    max_counter = atoi(sArr[5]);
    profile_clcd(a,b,c,d,e);
    }
  }
```

CLCD

<Print>

```
void profile_clcd(String a, String b, String c, String d, String e){
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("loading");
  delay(1000);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(a);  // name
  lcd.setCursor(0, 1);
  lcd.print(b); // age
  lcd.setCursor(3, 1);
  lcd.print(c); // height
  lcd.setCursor(7, 1);
  lcd.print(d); // weight
  lcd.setCursor(10, 1);
  lcd.print(e); // bmi
  delay(3000);
}
```



ID_Check with RFID&Bluetooth

<Information>

C++

```
byte key[4]={0, 0, 0, 0}; //input_RFID_KEY_VALUE
char *ptr0[6] = {"None", "None", "None", "None", "None","None"}; //Dismatch_ID_INFORMATION
char *ptr1[6]= {"JoYhongHyeon","30","177","73","23","10"}; //Frist_ID_INFORMATION (Name/Age/Height/Weight/Bmi/Max_Count)
char *ptr2[6] , *ptr3[6],*ptr4[6],*ptr5[6],*ptr6[6],*ptr7[6],*ptr8[6],*ptr9[6],*ptr10[6],
*ptr11[6],*ptr12[6],*ptr13[6],*ptr14[6]; //Blank_INFORMATION(MAX=15)
char **profile[15] = {ptr0,ptr1,ptr2,ptr3,ptr4,ptr5,ptr6,ptr7,ptr8,ptr9,ptr10,ptr11,ptr12,ptr13,ptr14}; //INFORMATION_LIST
byte *key0[4] = {0,0,0,0}; //Error_RFID_KEY_VALUE
byte *key1[4] = {141,171,7,50}; //First_RFID_KEY_VALUE
byte *key2[4],*key3[4],*key4[4],*key5[4],*key6[4],*key7[4],*key8[4],*key9[4],*key10[4],
*key11[4],*key12[4],*key13[4],*key14[4]; //Blank_RFID_KEY_VALUE
byte **key_index[15]= {key0,key1,key2,key3,key4,key5,key6,key7,key8,key9,key10,key11,key12,key13,key14}; //RFID_LIST
```

ID_Check with RFID&Bluetooth

<RFID>

```
C++

  void RFID_Check(){
    byte match1,match2,match3,match4 = 0; //Inforamtion에 저장되어 있는 RFID_KEY_VALUE
    byte matchkey = 0; //match ? 1:0
    byte length=sizeof(key_index)/sizeof(key_index[0]); //=15

    Serial.print("Card UID:");

    for(byte i=0; i<4; i++){ //store RFID_KEY_VALUE
      Serial.print(mfrc.uid.uidByte[i]);
      Serial.print(" ");
      key[i]=mfrc.uid.uidByte[i];
    }
```

```
Serial.println(); //몇 번째 사용자의 RFID_KEY_VALUE인지 판별
for (int i = 0; i < length; i++) {
  if (key[0] == key_index[i][0]) match1 = 1;
  else match1 = 0;
  if (key[1] == key_index[i][1]) match2 = 1;
  else match2 = 0;
  if (key[2] == key_index[i][2]) match3 = 1;
  else match3 = 0;
  if (key[3] == key_index[i][3]) match4 = 1;
  else match4 = 0;
  if (match1*match2*match3*match4==1) {
      matchkey = i;
      break;
  }
}
send_data(matchkey); //matchkey=0 이면 none
}
```

ID_Check with RFID&Bluetooth

<Bluetooth>

```
C++
  void bluetooth_ID(){
    String Name=""; //Read_Data
    byte match_bluetooth=0; //match? 1 : 0

    while(bluetooth.available())  //existing data in buffer
    {
      char myChar = (char)bluetooth.read();  //문자 하나씩 저장
      Name+=myChar;   //String Name에 문자 하나씩 증
      delay(5);    //수신되는 문자 끊김 방지
    }
    if(!Name.equals(""))//Name 문자열에 data가 존재하면
    {
      Serial.println("input value : "+Name);
    for(i; i<15; i++){  //15명의 이름 비교
      if(Name.equals(profile[i][0])){ //같은 이름이 있는 프로필 발견시
        send_data(i); //2nd Board에 데이터 전송
        match_bluetooth = 1; //일치하면 match_bluetooth = 1로 표현
        break; //일치하면 반복문 종료
      }
      else{
        match_bluetooth =0; //불일치하면 match_bluetooth = 0 유지
      }
    }
    if(match_bluetooth==0){ //불일치하면 불일치 문구 전송
        Serial.println("no match!");
        Serial.println("input your name again");
        bluetooth.println("no match!");
        bluetooth.println("input your name again");
    }
   Name=""; //이름 초기화
   }
}
```

ID_Check with RFID&Bluetooth

<Main>

```C++
void loop() {
  if(bluetooth.available()){ // bluetooth에서 값이 들어오면 만족(available() : input_data의 크기(byte) 반환)
    bluetooth_ID();
  }
  else{
    if(!mfrc.PICC_IsNewCardPresent() || !mfrc.PICC_ReadCardSerial()){ //새로운 카드가 인식되거나 읽힌 RFC_Data가 있
    delay(500);
    return;
    }
    RFID_Check();
    Serial.println();
  }
}
```

문제해결

인터럽트 서비스 루틴에 대해서

ISR은 다른 대부분의 기능에는 없는 몇 가지 고유한 제한이 있는 특수한 종
다.또한 어떤 파라미터도 할 수 없습니다.

일반적으로 ISR은 가능한 한 짧고 고속이어야 합니다.스케치가 여러 ISR을
다.현재 ISR이 종료된 후 우선순위에 따라 다른 인터럽트가 실행됩니다.mi
내부에서는 증가하지 않습니다.부터delay()는 인터럽트를 필요로 합니다.IS
음에는 동작하지만 1~2밀리초 후에 불규칙하게 동작하기 시작합니다. dela
적으로 동작합니다.

일반적으로 글로벌 변수는 ISR과 메인 프로그램 간에 데이터를 전달하기 위
니다.volatile.

인터럽트에 대한 자세한 내용은 Nick Gammon의 메모를 참조하십시오.

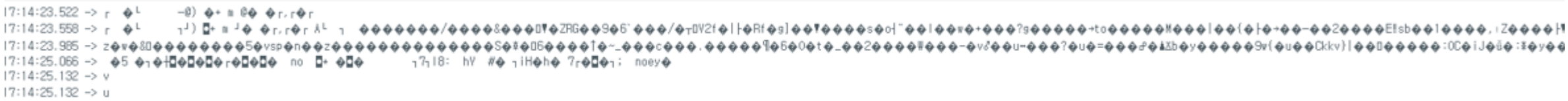
I2C 통신에서 Master는 transmit과 receive에서 interrupt 발생, Slave에서는 Master로부터 data가 수신되거나 송신 요청이 왔을 때 interrupt 발생

->CLCD에서 I2C adapter 사용

-> 모든 interrupt을 활성화 해주는 interrupts() 사용해 강제적으로 모든 interrupt 활성화

문제해결

- 동적할당 이용했을 때



- 이중 포인터 배열

```
char *ptr0[6] = {"None", "None", "None", "None", "None","None"}; //Dismatch_ID_INFORMATION
char *ptr1[6]= {"JoYhongHyeon","30","177","73","23","10"}; //Frist_ID_INFORMATION (Name/Age/Height/Weigh
char *ptr2[6] , *ptr3[6],*ptr4[6],*ptr5[6],*ptr6[6],*ptr7[6],*ptr8[6],*ptr9[6],*ptr10[6],
*ptr11[6],*ptr12[6],*ptr13[6],*ptr14[6]; //Blank_INFORMATION(MAX=15)
char **profile[15] = {ptr0,ptr1,ptr2,ptr3,ptr4,ptr5,ptr6,ptr7,ptr8,ptr9,ptr10,ptr11,ptr12,ptr13,ptr14};
byte *key0[4] = {0,0,0,0}; //Error_RFID_KEY_VALUE
byte *key1[4] = {141,171,7,50}; //First_RFID_KEY_VALUE
byte *key2[4],*key3[4],*key4[4],*key5[4],*key6[4],*key7[4],*key8[4],*key9[4],*key10[4],
*key11[4],*key12[4],*key13[4],*key14[4]; //Blank_RFID_KEY_VALUE
byte **key_index[15]= {key0,key1,key2,key3,key4,key5,key6,key7,key8,key9,key10,key11,key12,key13,key14};
```

보완점

1) Socket.io - Server(Java) / Client(Arduino

2) Input New Information

```
if(Name.equals("New")){
 char New[100];
 char *sArr[6];
 int index=0 , i = 0;
 char * ptr = NULL;
 bluetooth.println(input information (Name/Age/Height/Weigth/BMI/MAX_COUNT));
 while(bluetooth.available() < 1) {};
 while(bluetooth.available()){
       str[index++] = P_Serial.read();
     }
     str[index]='\0';

     ptr = strtok(str, "#");

   while(ptr != NULL)
   {
     sArr[i++] = ptr;
     ptr = strtok(NULL, "#");
   }
   for(i=0; i<15; i++){
     if(profile[i][0].equals==""){
       profile[i][0] = sArr[0];
       profile[i][1] = sArr[1];
       profile[i][2] = sArr[2];
       profile[i][3] = sArr[3];
       profile[i][4] = sArr[4];
       profile[i][5] = sArr[5];
       break;
     }
   }
   send_data(i);
}
```