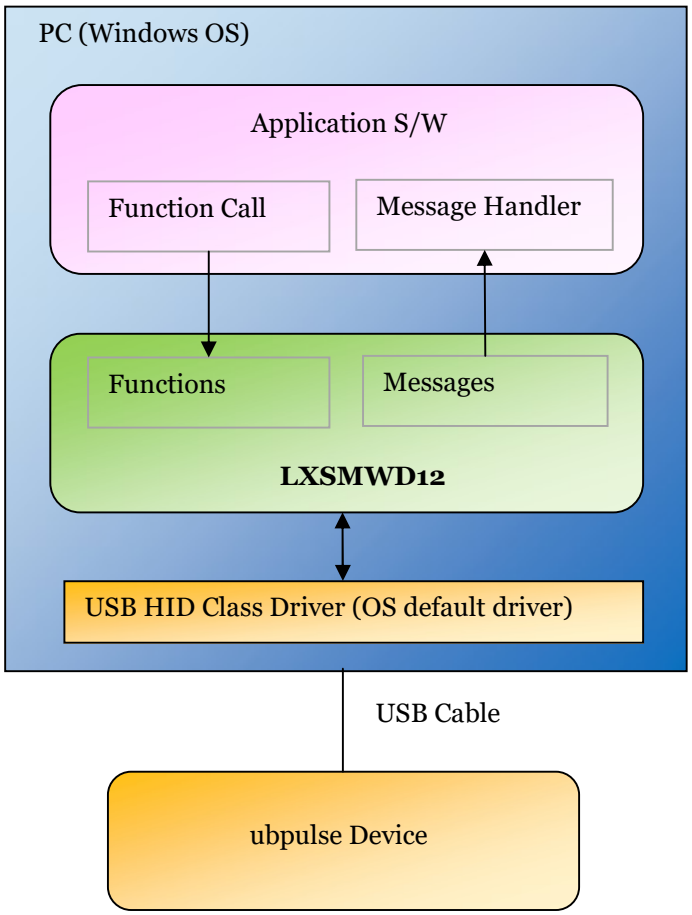
LXSMWD12 개발자 설명서

USB HID 기반 ubpulse HRV API

Doc. ID. LXD33 V4

Release Date. 2020-01-27 .

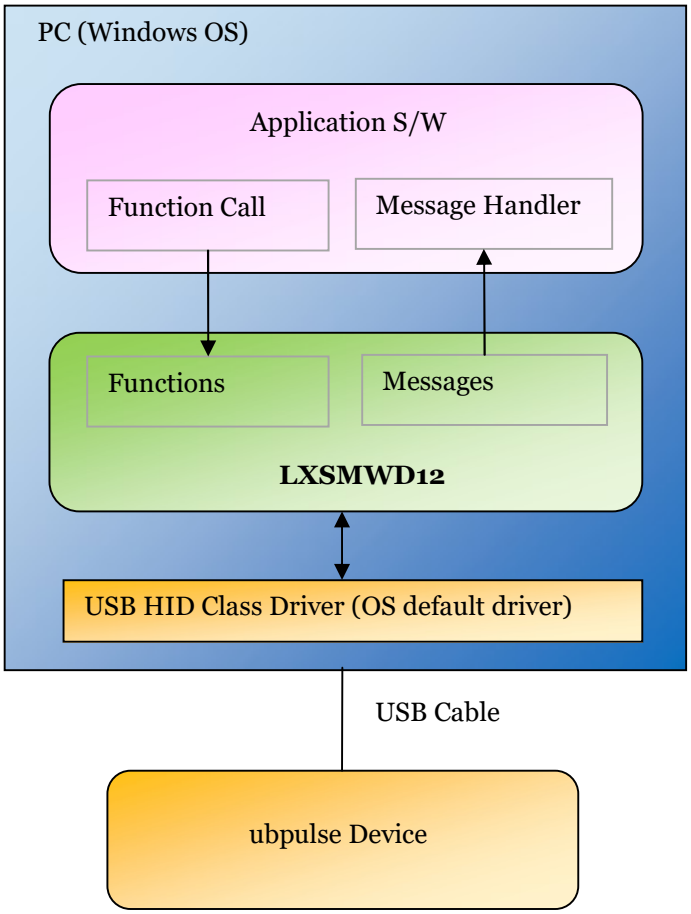
Abstract – LXSMWD12 는 ubpulse 장치와 응용프로그램 사이에서 통신매개하는 win32 API DLL(Dynamic Link Library). ubpulse 장치와의 실시간 통신하면서 PPG wave, 분당심박수, 혈류지수 등 확보 가능, 동시에 HRV(Heart Rate Variabilty)검사결과 활용가능.



목차

LXSMWD12 개요.

LXSMWD12 는 USB HID(Human Interface Device) 기반의 맥파 측정기기에서 송신되는 실시간 데이터수집 및 HRV 분석 데이터 확보하는 WIN32 API 형식 DLL.



LXSMWD12 특징.

- API type : DLL (Dynamic Link Libtary)
- DLL name : LXSMWD12
- API making tool : Visual C++ 2010. win32 API based dll project.
- Supporting platform : 64bit / 32bit application.
- Supporting OS : Windows 10, 8.1, 8, 7
- Recommended IDE for developing the application program : Visual C++ 2010, 2015, 2017. 2019

LXSMWD12 시작하기.

LXSMWD12 다운로드

LXSMWD12 64bit 용

다운로드 주소 : https://github.com/ubpulse/ubpulse-H3/raw/master/LXSMWD12_64bit.zip

LXSMWD12 32bit 용

다운로드 주소 : https://github.com/ubpulse/ubpulse-H3/raw/master/LXSMWD12_32bit.zip

주의 : 다운로드 받은 후 파일속성에서 차단해제 이후 압축풀기.

LXSMWD12 파일 복사.

files	Copy to
LXSMWD12.DLL	응응프로그램의 실행파일경로에 복사.
LXSMWD12.LIB	응용프로그램 소스 폴더에 복사. 응용프로그램에서 implicit linking 시켜서 사용한다.
LXSMWD12.h	응용프로그램 소스 폴더에 복사. 응용프로그램에서 인클루드 시켜서 사용한다.

LXSMWD12 DLL 임포팅.

Visual C++ 인 경우, 응용프로그램 프로젝트 소스에서 DLL implicit linking 하고, 헤더파일 인클루드.

```
#pragma comment(lib,"LXSMWD12.lib") // DLL implicit linking

#inlcude "LXSMWD12.h"
```

TEST_LXSMWD12 다운로드

TEST_LXSMWD12 download

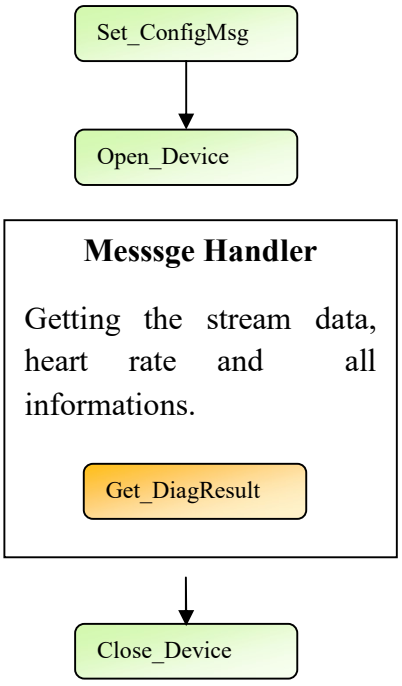
URL : https://github.com/ubpulse/ubpulse-H3/raw/master/TEST_LXSMWD12_VC2017_32_64.zip

- Visual C++ 2017 Project Source using LXSMWD12.DLL
- 주의 : 다운로드 받은 후 파일속성에서 차단해제 이후 압축풀기.

응용프로그램에서 LXSMWD12 활용 전체흐름.

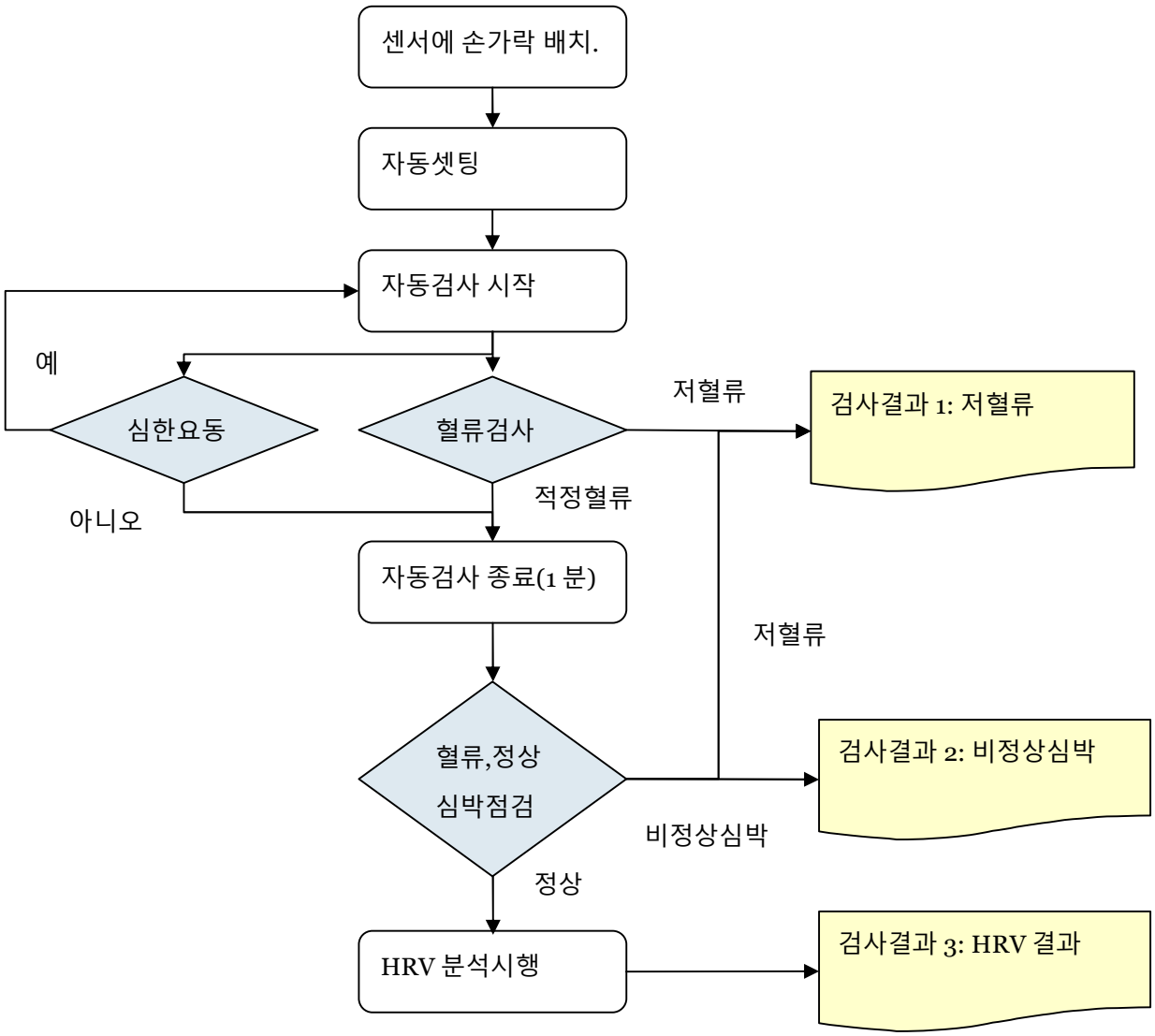
장치열기와 닫기.

응용프로그램에서는 LXSMWD12 에서 제공하는 함수 Set_ConfigMsg 를 호출하여 DLL 에서 전송되어 오는 메시지들을 수신할 윈도우 핸들을 넘기는 처리를 해둔다. 응용프로그램에서 DLL 함수 Open_Device 혹은 Open_Device_UsbSerial 호출하여 장치와의 연결 달성한다. 장치는 항상 PC 측으로 연속적으로 데이터를 전송하며, Open_Device 성공한 시점부터 DLL 은 장치와 통신시작 하면서 장치로부터의 데이터를 수신 및 처리하여 응용프로그램측으로 다양한 종류의 메시지 전송이 실시간으로 이뤄진다. 응용프로그램에서의 모든 처리는 메시지 기반으로 이뤄져야 한다. 한번 오픈된 장치를 더 이상 사용하지 않을 때는 Close_Device 함수 호출하여 닫기 처리해야하며 Close_Device 성공한 상태에서는 장치와 DLL 이 통신하지 않는 상태.



장치열기 성공이후의 작동흐름.

장치열기 성공한 이후 DLL 은 장치와의 통신을 수행하면서, 장치의 상태 및 측정된 데이터들을 실시간으로 수집하며 동시에 응용프로그램 측으로 알려야 할 정보들을 메시지로 전송한다. 이때 전송되는 메시지는 장치를 사용하는 사용자의 동작과 연동되어 작동된다.



먼저, 사용자가 손가락을 장치에 배치한 경우의 흐름은 그림과 같다. 손가락 배치되었음이 자동감지되면, 3 초정도 소요되는 자동셋팅과정이 진행되어 최적의 신호측정 상태를 위한 셋팅이 이뤄진다. 이후 자동검사진행되어 정상적인 혈류량인 경우에는 1 분의 측정이 완료된다. 만일 측정중에 비정상적으로 혈류량이 적은 사용자인 경우에는 1 분이 완료되기 전에 저혈류 판정결과를 제시한다. 응용 프로그램에서는 혈류량이 작음을 검사결과로 표현한다.

측정 중에 센서에 배치한 손가락을 심하게 움직인 경우에는 처음부터 다시 검사를 시작하게 된다.

손가락의 심한움직임이 없이 정상적으로 측정이 진행된 경우에는 1 분 동안 측정된 심박 시간격중에서 비정상심박여부를 판정하며, 이 단계에서 다시한번 비정상 혈류량 검사도 재차 진행한다. 저혈류인 경우 저혈류 판정결과를 제시하게된다. 저혈류가 아닌 경우 비정상심박여부를 검사하게되는데, 부정맥 같은 질환등이 이상심박을 보이는 가장 대표적인 원인이며 이와 같은 이상심박이 있는 경우에는 HRV 분석이 적용될 수 없다. 따라서, 이상심박인 경우에는 비정상심박 이라는 결과를 응용프로그래에서 제시한다.

저혈류도 아니고, 비정상 심박도 아닌 경우 HRV 분석이 시행되어 최종적인 결과들이 정리되어 제공되며 이 결과를 응용프로그램에서 사용자에게 표현한다.

DLL 에서 제공하는 함수.

필수 함수.

아래 3 개의 함수는 장치와의 통신을 위해 필수로 사용되어야 하는 함수들.

기능	함수	개요
장치열기	Open_Device(int pid)	인자 pid : 기기 고유아이디 전달. 기기별 고유아이디. ● ubpulse H3 : 33 ● ubpulse G3 : 58 ● ubpulse OEM1 : 56
	Open_Device_UsbSerial(int pid , char * serial)	pid: 기기 고유아이디 전달. serial : 기기 고유 시리얼 전달. 대소문자 구분.
장치닫기	Close_Device()	프로그램종료시 Close_Device() 필수호출한다.
검사결과확보	Get_DiagResult(st_DiagResult *diagresult)	검사결과를 구조체 diagresult 로 반환받는다. 구조체 st_DiagResult 타입정의는 LXSMWD12.H 에 있음.
메시지설정.	Set_ConfigMsg(int msgtype_idx, HWND hwnd_msgtarget,int msgid, unsigned char on_off)	DLL 에서 전송해올 메시지타입(msgtype_idx) quffh 수신받을 윈도우 핸들을 지정하고, 메시지 아이디 지정 및 해당 메시지를 수신받을지 여부(on_off)를 지정한다.

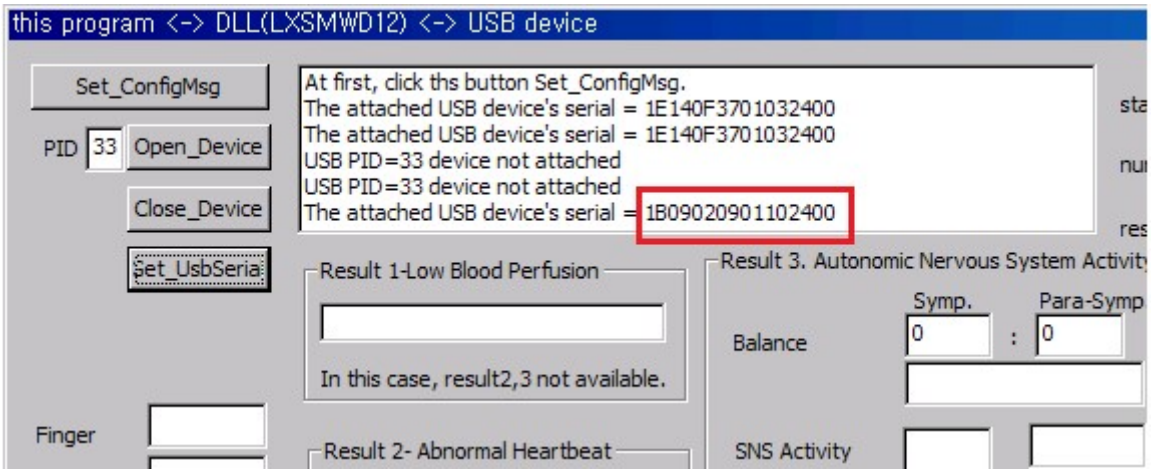
기타함수들.

아래 함수들은 응용프로그램에서 필요에 따라 사용혹은 사용하지 않아도 되는 함수들이다. 상황에 맞게 임의로 선택하여 활용한다.

기능	함수	개요
검사수동시작	Diag_ManualStart();	장치는 기본적으로 자동으로 검사진입기능이 있으나, 이 함수를 호출하면 응용앱에서 언제든지 검사진입 가능.
검사수동중지	Diag_ManualStop();	진행중이던 HRV 검사 중지.
DLL 내의 버퍼사이즈확보	Get_QBufferSize();	DLL 에 내장된 큐버퍼의 전체 용량크기 반환.
기기의 USB 시리얼 문자 확보	Get_UsbSerial(int pid, char * usbserial)	인자 pid 에 기기의 고유아이디 전달한다. ubpulse H3 : 33 시리얼문자사이즈 : 16 문자.

Get_UsbSerial 호출 코드 예.

```
void CDlgStatusView::OnBtnGetUsbSerial()
{
	short retv;
	CString str;

	char buf_serial[20]; // array size must be more than 17.

	UpdateData(TRUE);

	retv = Get_UsbSerial(33, buf_serial); // PID for ubpulse H3 :  33

	if(retv == 1) // success
	{
		str.Format(_T("The attached USB device's serial = %s"), buf_serial);
	}
	else if(retv == -1) // fail
	{
		str.Format(_T("USB PID=%d device not attached"),m_PID);
	}

	Display_Status1(str);

}
```



Open_Device_UsbSerial(int pid , char * serial) 코드예.

```
char buf_serial[20] = "1B09020901102400"; // 대소문자 구분.

Open_Device_UsbSerial (33, buf_serial);
```

DLL 이 발생하는 메시지.

응용프로그램이 DLL 측에 Open_Device 함수를 호출하여 성공적인 경우, DLL 은 모든 상황에 대하여 메시지를 응용프로그램측에 정보를 제공하게 된다. 응용프로그램은 DLL 에서 전송하는 메시지 처리 를 주로한 프로그램 동작을 구현해야한다.

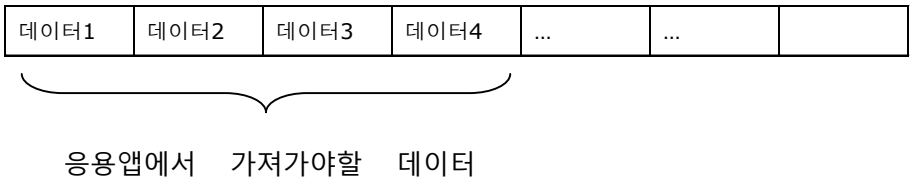
DLL 에서 발생하는 메시지의 종류는 아래표에 정리되어있다.

메시지의 타입인덱스	메시지 발생사유	WParam 값 (메시지전송시 첫번째 인자)	Lparam 값 (메시지전송시 2 번째 인자)
0	DLL 이 장치로부터 전송된 스트림(맥파파형)데이터를 4 샘플링 확보한 시점마다 발생. 이 데이터를 응용프로그램이 가져가야한다.	현재의 Q 버퍼에 남아있는 데이터수량. 자료형: unsigned int	스트림데이터가 저장된 float 형 배열의 시작주소. 자료형 : (float *)
1	심박발생시	0	분당심박수. 예: 78 자료형 : unsigned char
	혈류지수 변경시.	1	혈류지수: %단위의 값에 100 이 곱해진 값. 자료형 : unsigned int

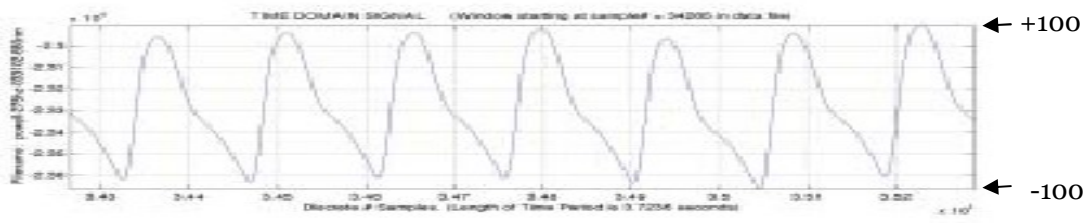
메시지의 타입인덱스	메시지 발생사유	WParam 값 (메시지전송시 첫번째 인자)	Lparam 값 (메시지전송시 2 번째 인자)
2	손가락 탈착.	0	0: 탈, 1: 착. 자료형 : unsigned char
	측정상태의 안정불안정.	1	0: 불안정,1:안정. 자료형 : unsigned char
	자동셋팅	2	0:셋팅중아님,1: 셋팅중. 자료형 : unsigned char
	검사상태	3	0:비진행, 1:검사진행중, 2: 검사중지.-손가락탈원인. 3:검사중지-저혈류원인 자료형 : unsigned char
	검사데이터보유여부	4	0: 검사데이터없음. 1:검사데이터 있음.
	검사잔여시간	5	초단위로 60~0 까지 다운카운팅되는 값이 전달됨. 자료형 : unsigned char
	검사중 심박수량.	6	검사중에 누적수집된 심박수량. 자료형 : unsigned char

스트림데이터의 구조

스트림(맥파파형) 전송 상태에서 메시지타입인덱스 0 은 데이터를 가져가라는 의미이며, 해당메시지의 2 번째 인수에 스트림데이터를 저장하고 있는 float 형 배열의 시작주소가 전달된다. 이때 배열에 맥파파형의 데이터 4 개가 저장되어 있는 상태이므로 이 파형정보를 활용하려면 4 개의 float 형 배열의 4 개의 데이터를 응용앱에서 읽어가야 한다.



맥파파형이란 아래 그림과 같은 파형 데이터를 의미한다. 맥파파형데이터의 최대값은 +100 최소값은 -100 이다.



메시지로 전달된 인자로부터 float 형 배열로 데이터를 받는 방법.

DLL 내에 float 형 배열이 있고, DLL 에서는 실시간으로 수집되는 데이터를 배열에 저장 한다. 실시간 데이터 수집과정에서 DLL 은 정해진 수량만큼 데이터가 확보된 시점에 메시지를 발생하며, 이를 수신한 프로그램측 메시지핸들러 함수 내에서는 데이터를 읽어와서 이용하는 처리가 이뤄져야 한다. 메시지로 전달되면서 데이터에 접근하기 위한 정보는 메시지 인자 중 lParam 에 그 정보가 있다. lParam 으로 DLL 의 배열의 포인터가 전달된다. 예를들어 array[10] 이라는 배열이 있다면, array 에 해당하는 것을 전송한다. 따라서, 우리는 lParam 을 이용하여 활용하기 위해서는 lParam 을 float 형 포인터로 형변환을 하고 (float *)lParam 이것이 마치 array 인것 처럼 활용하면된다. 구체적인 코드예를 이용하여 설명해보자.

우리의 응용프로그램에서 데이터를 받아둘 배열을 1 개 만들자. 배열의 크기를 4 로 한 이유는 한번에 받아오는 데이터 수량이 4 개이기 때문이다.

```
float save_data[4];
```

데이터 확보하는 전용함수를 선언하고, 함수인자로 float 형 포인터변수를 전달하자.

```
void Get_Data(float * my_data)
{
	for(int i = 0 ; i< 4 ;i++)
	{
		save_data[i] = *(my_data+i); // my_data 의 전체 데이터를 save_data 로 이전했다.
	}
}
```

위의 함수가 호출될 장소는 우리의 메시지핸들러인 OnStreamData 내에 두면된다.

즉,

```
void CChildView::OnStreamData(WPARAm wParam, LPARAM lParam )
{
Get_Data((float *)(lParam)); // DLL 속의 데이터를 우리가 사용가능한 배열로 이전하는 함수
}
```

검사 결과데이터 구조체 멤버

본 DLL 에서는 검사결과데이터를 구조체 st_DiagResult 에 기록되며, 각각의 멤버에 대한 설명이다.

구조체 멤버	설명
Type	검사결과타입. 3 인 경우가 정상적인 HRV 분석데이터가 있는 상태. 비정상심박인 경우 비정상심박수량은 NUM_FaultHBI 에 기록되어있다. 1: 저혈류, 2:비정상심박, 3 : HRV 검사결과
SNSBal	교감신경비율, 단위 : % 100*교감활성/(교감활성+부교감활성) 으로 계산된 값.
SNSBal_Level	교감신경비율 크기판정값.
PSNSBal	부교감신경비율, 단위 : % 100*부교감활성/(교감활성+부교감활성) 으로 계산된 값.
PSNSBal_Level	부교감신경비율 크기 판정값.
SNSAct	교감신경활성.
SNSAct_Level	교감신경활성 크기 판정값.
PSNSAct	부교감신경활성.
PSNSAct_Level	부교감신경활성 크기 판정값.
ANSAct	자율신경활성
ANSAct_Level	자율신경활성 크기 판정값.
HRVIndex	심박변이도.
HRVIndex_Level	심박변이도 크기 판정값.
HR	분당심박수 , 1 분측정 중의 평균 심박수.
HR_Level	분당심박수 크기 판정값.
PI	혈류지수 단위 %, 1 분측정 중의 평균 혈류지수.
PI_Level	혈류지수 크기 판정값.

위 표에서 “크기 판정값” 은 모두 5 단계로 0,1,2,3,4 의 값이 가능하며, 각 값에 따라 표준에서 벗어난 정도를 의미한다. 2 가 표준범위에 있음을 의미, 1 은 작음. 0 은 매우 작음. 3 은 큼, 4 는 매우큼을 의미한다.

구조체멤버	설명
HBI[255]	1 분동안 측정된 심박시간격 데이터 배열. 심박시간격 단위 밀리초(= 1/1000 초) 단위로 표현됨.
NUM_HBI	상기 HBI 배열에 저장된 심박수량. 1 분동안 심박수량은 측정할 때마다 변경되므로 이 값을 참조하여 위 HBI 배열에서 데이터를 확보한다.
NUM_FaultHBI	검사결과 타입이 비정상 심박인 경우 비정상심박수량.
Histogram[225]	심박시간격 데이터 배열로부터 구해진 히스토그램. 0.2 초부터 2 초구간을 0.008 초 단위로 세분하여 각 시구간에 해당하는 심박시간격을 보인 심박수량. 즉, Histogram[0] 은 심박시간격이 0.2 초~ 0.2008 초 사이에 있는 심박수량. Histogram[1] 은 심박시간격이 0.208 초~ 0.216 초 사이에 있는 심박수량. . . Histogram[224] 는 심박시간격이 1.992 초~ 2 초 사이에 있는 심박수량.

Revision History

Release Date	Doc. ID	Description of Change
2020-01-27	LXD33 v4	- 함수 Open_Device_UsbSerial 추가.
2018-09-12	LXD33 v3	- 함수 Get_UsbSerial 추가. ~~- MFC regular dll project~~ -> win32 api dll project
2018-04-26	LXD33 v2	오타수정.
2013-07-10	LXD33 V1	First release.