dDSAU Software Overview

Caroline Ekchian & Andy Mo

1 Overview of Software Modules

Client Modules:

1. Client.h
2. Client.cpp
3. Form1.h
4. ProfileNameWindow.h
5. SimultaneousPattern.h
6. Warning_DateandPatientName.h
7. tinyxml2.h
8. tinyxml2.cpp

1. Client.h:
Contains the DsauClient class function prototypes. Currently there are only five functions in the DsauClient class: mkConnection(), sendThis(), recvThis(), saveThis(), and rmConnection(). The mkConnection and rmConnection functions sets up and shuts down a TCP connection to the server respectively. sendThis() is used to send commands to the server and recvThis is used to collect data from the server once a scan has started.

2. Client.cpp:
Contains the function implementations of the DsauClient class, which utilizes the WinSock library supplied by windows.

3. Form1.h:
Form1.h contains the majority of the functionality for the GUI and outlines the primary form (*Figure1*) for the project. There are 10 functions in this header file. The majority of these functions include the event handlers for the the main user interface window (e.g. button click events, responding to changes in the values of drop down menus and text in the text boxes). The names of these functions are: button4_Click(), button1_Click(), button6_Click_1(), SaveButton_Click(), profileNameCB_SelectedIndexChanged(), comboBox2_SelectedIndexChanged(), profileManagerButton_Click(), textBox2_TextChanged(), and textBox4_TextChanged(). Additionally, the last function, CreateGraph(), includes the code that creates the graph using the ZedGraph library that was downloaded from Sourcefourge.net and creates the .csv file that contains that data that is collected from the server. A metadata file is also created in this function in order to let the user know which parameters were used for each data file that was collected. This header file interacts with many of the modules, including

Client.h, ProfileNameWindow.h, SimultaneousPattern.h, Warning_DateandPatientName.h, Client.cpp, and tinyxml2.cpp.

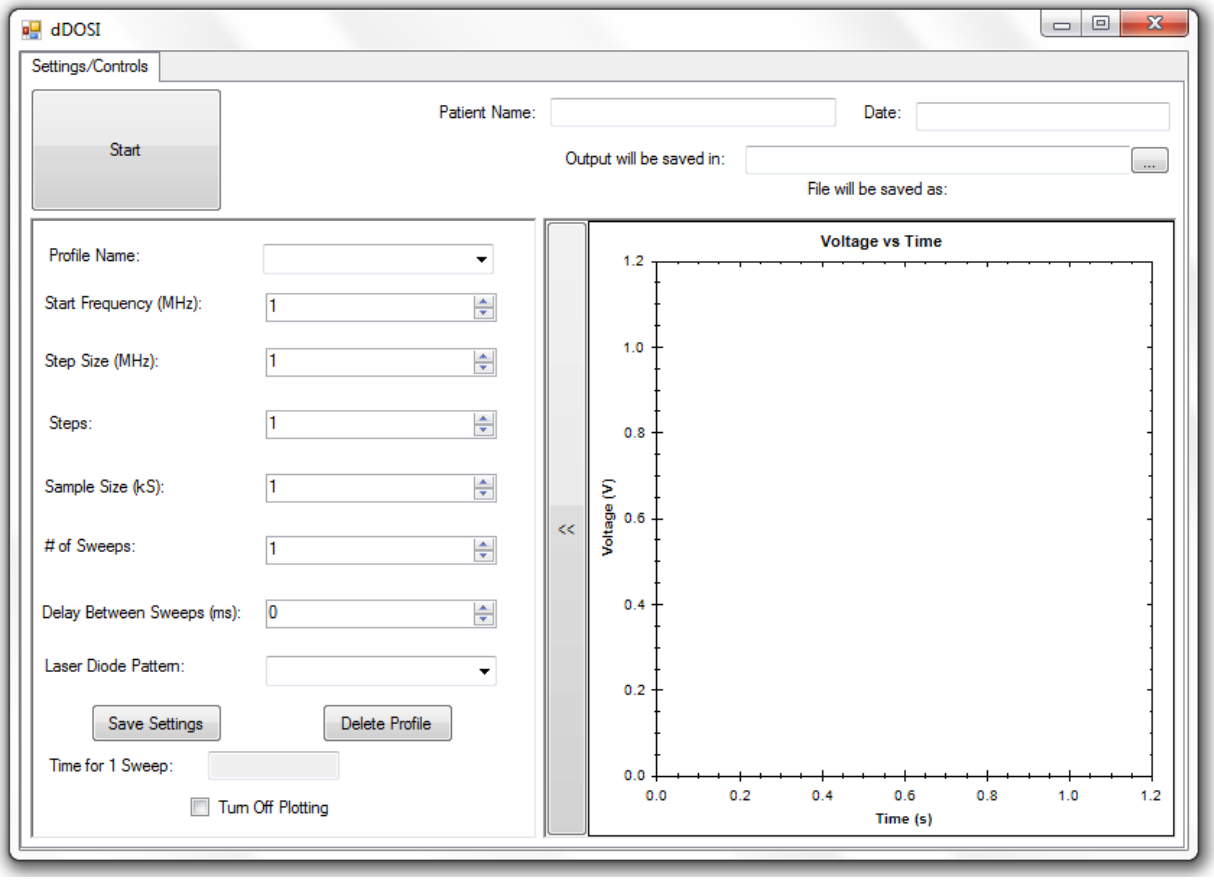

Figure 1: Form1.h Primary Form

4. ProfileNameWindow.h:

ProfileNameWindow.h outlines one of the secondary forms in the project (*Figure2*). This form allows the user to enter the profile name that they would like to save the parameter settings under. ProfileNameWindow.h contains two functions: button1_Click() and Save_Profile(). This header is used when the button1_Click() function in Form1.h is called. When the button1_Click() function is called in ProfileNameWindow.h file, there is error checking functionality to ensure that there profile name does not already exist. If it does not exist then the SaveProfile() function is called. The SaveProfile() function saves the profile name and the values of the parameters that the user entered into an XML file called ProfileName.xml. ProfileName.xml is saved to a location specified in the code. This header file interacts with Form1.h and Warning_DateandPatientName.h.

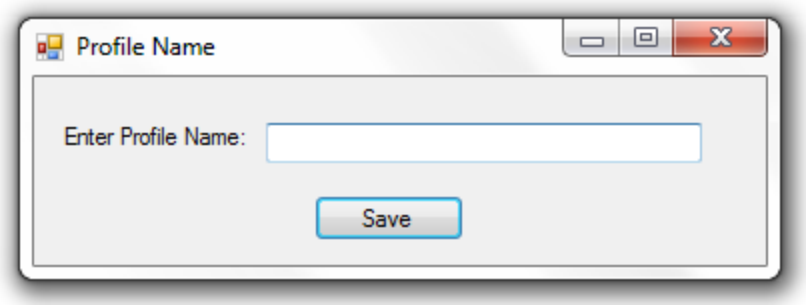

Figure 2: ProfileNameWindow.h Secondary Form

5. SimultaneousPattern.h:

SimultaneousPattern.h also outlines one of the secondary forms in the project. If the Simultaneous Laser Diode Pattern is selected in the main form (*Figure1*), the SimultaneousPattern window shows up. This allows the user to enter the offset for each of the laser diodes. There are 7 functions in this header file. One function, button1_Click(), is an event handler. This is called when the OK button is pressed. The other functions are: getDiode1Val(), getDiode2Val(), getDiode3Val(), getDiode4Val(), getDiode5Val(), and getDiode6Val(). These are public functions that allow the program to access the diode values that the user entered in the SimultaneousPattern window in Form1.h.

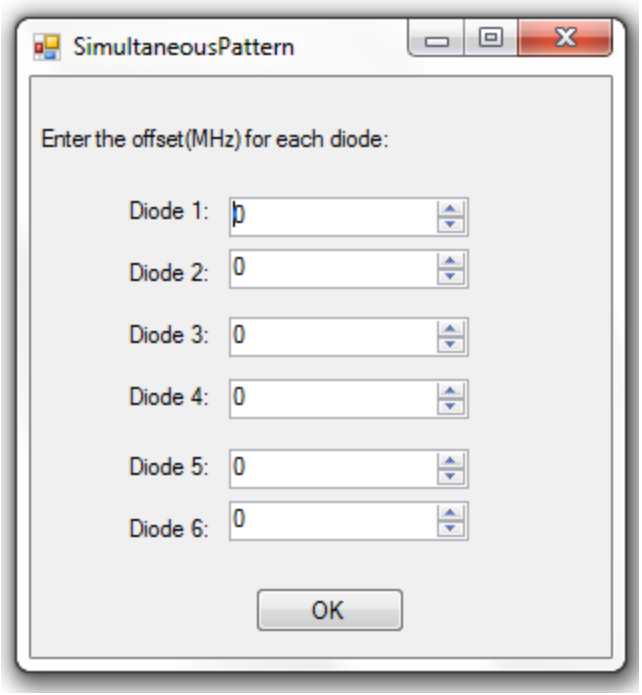

Figure 3: SimultaneousPattern.h Secondary Form

6. Warning_DateandPatientName.h:

Warning_DateandPatientName.h also outlines a secondary form for the project (*Figure4*). It serves as an error message Pop Up window. The error messages that it displays include the following: “Please enter a location for the file to be saved.”, “"Please Enter the Patient Name and Date.", "Unable to Connect to Server", "The Date and Patient Name fields cannot contain forward slashes or back slashes", and "A profile with this name already exists". There are two functions in this header file: button1_Click() and setErrorMessage(). The button1_Click() function hides the error window once the user presses the “OK” button. The setErrorMessage() is a public function that allows the program to set the error message in other header files where an error may occur (Form1.h and ProfileNameWindow.h).

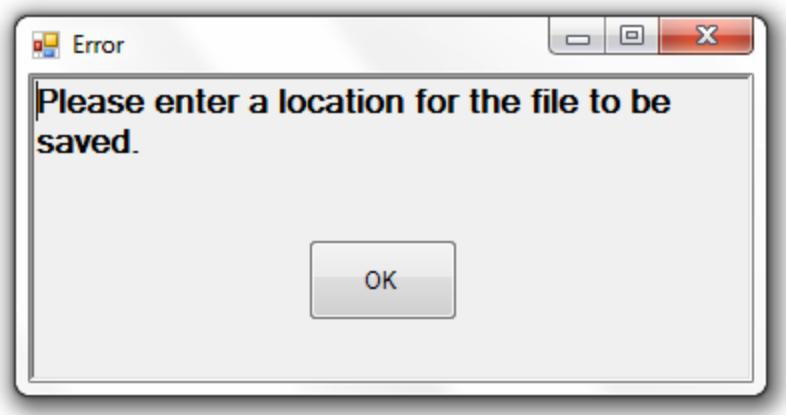

Figure 4: Warning_DateandPatientName.h Secondary Form Error Example

7. tinyxml2.h:
This header file was downloaded from GitHub.com. It contains the declarations for the functions that were used to write to and read from the XML file that contain the profile names. The functions that were used from this header file in order to write to the XML file are: LoadFile(), OpenElement(), CloseElement(), and PushAttribute(). The functions that were used to read from the XML file were: LoadFile(), CurrentNode(), FirstChild(), NextSibiling() and SaveFile(). This header file interacted with Form1.h and ProfileNameWindow.h.

8. tinyxml2.cpp:
This .cpp file was also downloaded from GitHub.com and it contains the code for the functions that were contained in the tinyxml2.h file. The only functions that were used from this .cpp file were: LoadFile(), OpenElement(), CloseElement(), PushAttribute(), CurrentNode(), FirstChild(), NextSibiling() and SaveFile(). This .cpp file interacts with the tinyxml2.h.

9. ZedGraph.dll:
This is the .dll that was downloaded from Sourceforge.net to graph the data on the GUI. The functions that were used from this .dll are PointPairList(), AddCurve(), Add(), AxisChange(), Refresh(), and Clear(). PointPairList() and AddCurve() were used to create the two seperate curves for ChannelA and ChannelB. Add() is used to add a point to a specific curve (either ChannelA or ChannelB). AxisChange() and Referesh() are used to refresh the graph and display the updated data. Clear() was used to clear the curves from the graph everytime the user presses the start button. This .dll file interacts directly with Form1.h.

Server Modules:

1. ddosi-contants.h
2. bitbang-spi.cpp
3. top.h
4. odscpp.cpp
5. collect.cpp
6. getSettings.cpp
7. SNC.cpp

1.ddosi-constants.h:
This file contains the constant definitions used for the ddosi project such as the LEDs and DDS Controls.

2.bitbang-spi.cpp
This module contains the SPI interface for the AD9910 DDS chip. It contains functions to set the dds configurations and write to dev channels. A bulk of the code is dedicated to setting up the appropriate connection to the GPIO device and “bitbanging”, or manually presenting a SPI communication packet, to the DDS. It can write up to 6 SPI devices in parallel. It also includes code to setup a standard DDS configuration. This configuration mostly sets the PLL parameters and changes the SDIO pin to SDI and activates the SDO pin for reads. Another function is provided for writing to a device, which streamlines the writing process.

3.top.h:
Header file for the DsauServer control structure. The structure stores holds variables that store the number of sweeps, number of steps, sample size, minimum frequency (start frequency), gain, and sweep delay.

4.odscpp.cpp:
This module contains the implementation for the DsauServer functions below.

loadSavedSettings()	Reads in a file containing information on the scan parameters
writeToAddr()	Sets the value of a parameter on the board
readFromAddress()	Reads in the value of a parameter on the board
isValid()	Checks if the value obtained from a_getPCVal is valid for writing to that dev on the board
a_getPCVal()	Extracts the value from the message buffer
saveToFile()	Saves the current DsauServer object data into a file
startCollection	Starts the collection process for the sweep and sends data to the client

5. SNC.cpp
This module runs the server and calls the DsauServer functions implemented in odscpp.cpp. The server reads in the first character of every client message to determine what to do with the rest of the message. If the first character is a ‘w’ or ‘r’ then it passes the rest of the message to the writeToAddr() and readFromAddress() functions. If it recieves an ‘s’, then it runs the startCollection function in collect.cpp to send data to the Client. Messages that begin with other characters are ignored.

6. collect.cpp
This module is the startCollecting function from odscpp.cpp that is integrated with the MicroZed. It collects data from the ADC and sends it to the client.

7. getSettings.cpp
This module contains the loadSaveSettigns and readFromAddress functions. It gets the settings for the DsauServer parameter values.

2 Installation and Compiling Information

In order to compile this Client and GUI portions of the software Visual Studio 2010 Service Pack 1 must be installed. This program can be downloaded through the msdn website. If Service Pack 1 is not installed there will be issues when trying to build the project. *Figure 5* shows the specific version of Visual Studio that was installed to develop the software. There are two external libraries that were used in the GUI: tinyxml2 and ZedGraph. Tinyxml2 was downloaded from GitHub.com and ZedGraph was downloaded from SourceFourge.net.

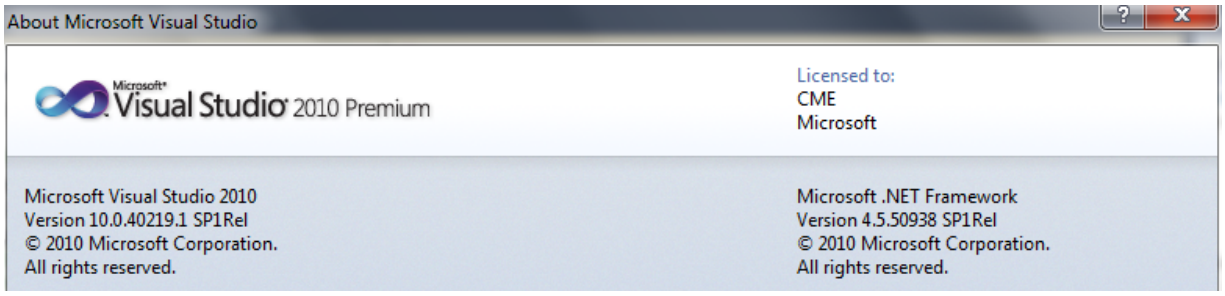

Figure 5: Information for Visual Studio Installation

To compile the GUI and Client code, the user must download the code from GitHub.com (https://github.com/bu-ddosi-team/GUI/tree/Final_GUI) or the Software/SourceCode/SD GUI folder of the Resource CD. Once the code is downloaded the user should open Visual Studio and Click on “Open Project”. The user should then find the location where they saved the project

and click on the SeniorDesign_1.sln file.

In order to have everything compile correctly there are a few lines of code that must be changed. For the profile name functionality of the GUI to be correct the user needs to change the path to the file that the profile name .xml file is being saved. There are two different locations where this file path needs to be changed. The file path is defined as XMLPATH at the beginning of the header files in Form1.h and ProfileNameWindow.h. These file paths will only need to be changed if the cmputer that the GUI is being used on changes. Additionally, the code assumes that the server has an IP address of “192.168.1.10” however if this is not the case it must be changed. This information can be changed in the Form1.h header file. The IP address is defined as IPADDRESS at the top of the header file.

After these file paths have to been changed to appropriate locations and the correct IP address is entered, the code can then be built, compiled and run using Visual Studio 2010 Service Pack 1. Note that after the project is compiled with the appropriate paths and IP address, the .exe file that is located in the Debug folder of the project can be run on its own from a different location without having to open visual studio. However, if the user would like to run it from the Desktop, for example, then the user must also copy and paste the appropriate .dll files to the desktop as well (ClientDLL.dll and ZedGraph.dll). The ClientDLL.dll is created in the debug folder of the SeniorDesign1 project directory.

For information on compiling and deploying the code on the development board please refer to the firmware report.

3 Flow Charts

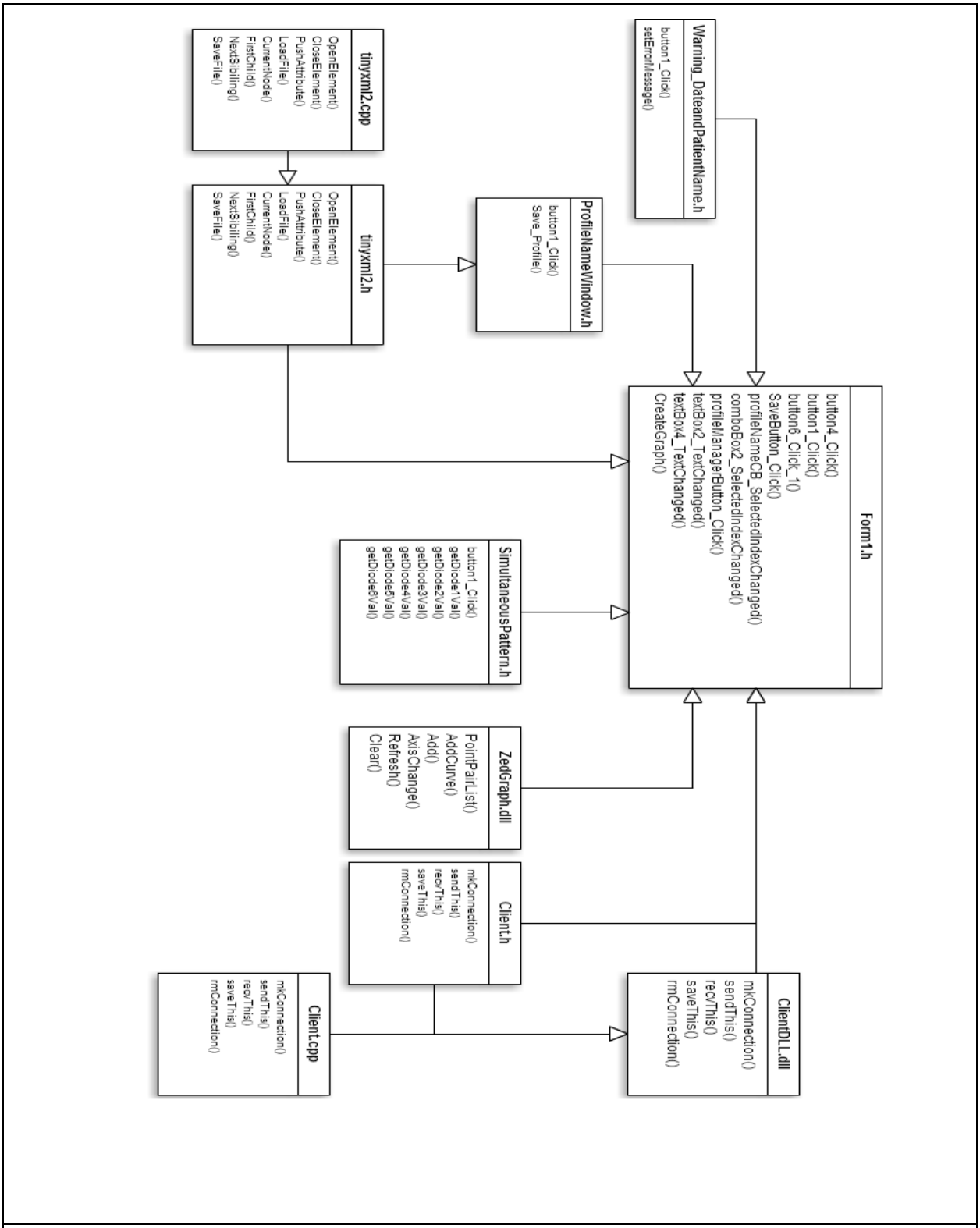

Figure 6: GUI and Client Flow Chart

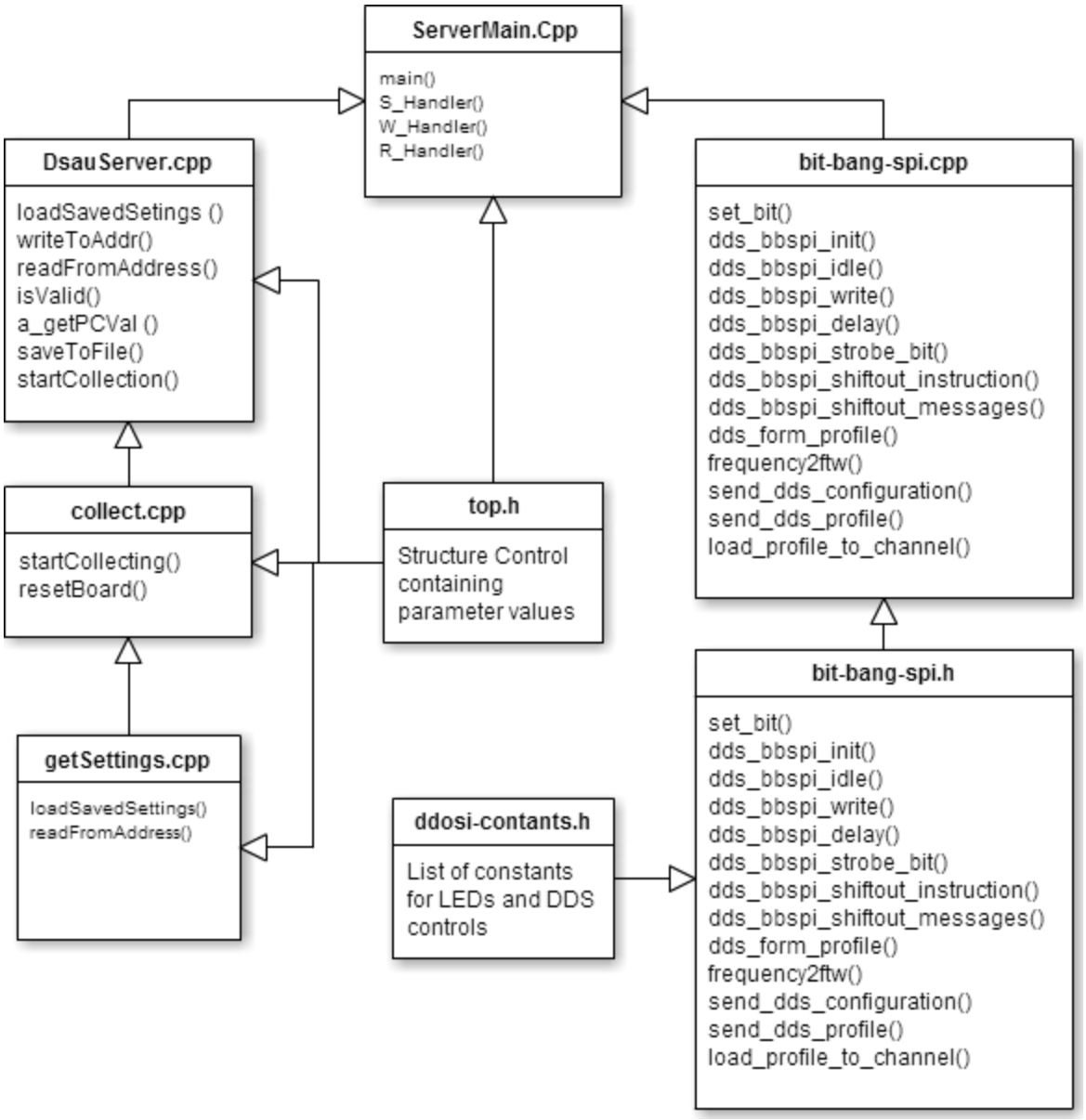

Figure 7: Server Code Flow Chart