WIMOTO – BLE SMART DEVICE
HLD – VER 1.0.0

Table of Contents

1 Overview

Wimoto is developing a smart device which integrates many sensors for BLE Climate profile (temperature, humidity and light level), Grow profile (light, soil temperature and soil moisture) Sentry profile (light, temperature, and humidity), Thermo profile (probe (NT) C temperature and thermopile temperature) and Water Profile (water presence and water level) on Nordic semiconductor nRF51822 based hardware platform. This device can be configured and managed by corresponding app on mobile devices like iPhone, Android and a proprietary gateway.

2 Application Control Flow

The application implements the functionality to advertise data in each profile as enhanced broadcast data and creates an alarm framework. The application flow first enters the alarm service (Connectable) mode and advertises the alarm service continuously; it advertises the alarm characteristic as specified by the profiles. During this time an iPhone/ Android device can connect to the device.

Climate profile

- Temperature alarm service
  - Alarm low value
  - Alarm high value
  - Alarm set
- Light alarm service
  - Alarm low value
  - Alarm high value
  - Alarm set
- Humidity alarm service
  - Alarm low value
  - Alarm high value
  - Alarm set

Grow profile

- Temperature alarm service
  - Alarm low value
  - Alarm high value
  - Alarm set
- Light alarm service
  - Alarm low value
  - Alarm high value
  - Alarm set
- Soil moisture service
  - Alarm low value
  - Alarm high value
  - Alarm set

Sentry profile

- PIR alarm service

Alarm set

- Accelerometer alarm service
  Alarm set

Thermo profile

- Probe temperature alarm service
  Alarm low value
  Alarm high value
  Alarm set
- Thermopile temperature alarm service
  Alarm low value
  Alarm high value
  Alarm set

Water profile

- Water level alarm service
  Alarm low value
  Alarm high value
  Alarm set
- Water presence service
  Alarm set

In the Connectable state the user can update the alarm low/ high level values and set alarm and receive notifications.

A service is provided in each profile for the device management. There are three characteristic provided in device management service,

1. Mode switching - to switch from Connectable mode to Broadcast mode,
2. Device firmware update - for updating the device's current firmware into a new one.
3. Time Stamp - Shows current time, date, month and year. This characteristics can also be updated with the user specified date, month, year and time value by writing a value to the characteristic

Whenever Device firmware update (DFU) characteristic is set and the iPhone/ Android device is disconnected from the Wimoto device, the embedded application enters into DFU mode. After a successful updating the firmware with a new valid image, the application control flow enters to the Connectable mode of the new firmware

If the Mode switch characteristic is set and the iPhone/ Android device is disconnected from the Wimoto device, the embedded application flow then enters Broadcast mode and broadcasts the services provided by the profile. The application remains in the Broadcast mode until a system reset/power on reset is incurred to the device.

The State Diagram of the control flow is shown below.

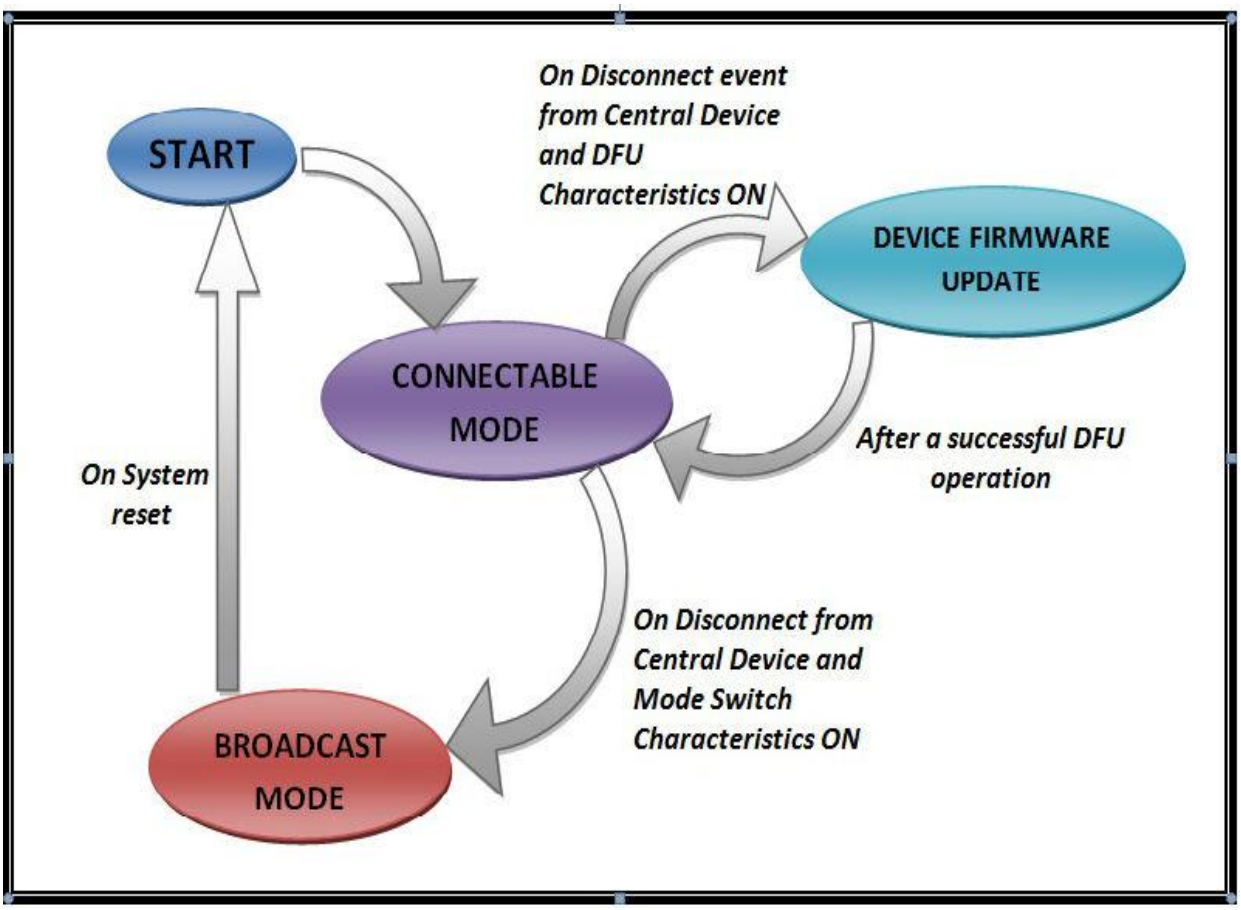


Figure 1

3 .Device Firmware Update

The DFU Bootloader is capable of receiving an application image on the BLE and updating the current running application on the nRF51822 chip. If there is no existing application in the nRF device, the device will be started in the bootloader mode

Device firmware update are envisaged to be performed on a device
a. With no existing application image
b. With an existing application image

A bootloader that is capable of preserving the existing application while receiving a new image is referred to as a 'dual bank' update and this method is used here. In the 256kB flash memory of nRF51822 evaluation kit the memory region from 0x3C800-0x40000 has been reserved for the DFU Bootloader. The memory layout is shown below

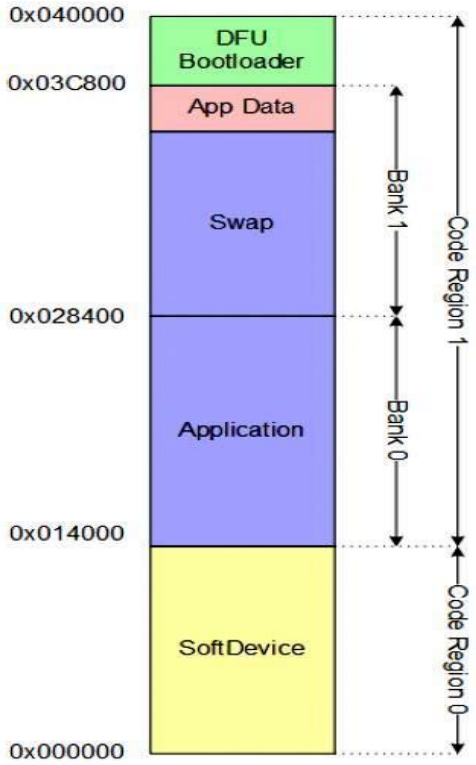


Figure 2

The SoftDevice will be using region 1 memory spanning from address 0x00000- 0x14000. Region 1 will contain the application and bootloader. The Application Data size is default set to 0x0000, meaning that all Application Data will be erased during a Device Firmware Update procedure.

The details about how Device Firmware Update works is summarized below

Device Firmware Update	
Step1	Before the Bootloader initializes, the active Application resides in Bank 0, while the content of Bank 1 is undefined.
Step2	As a first step the Bootloader erases Bank 1, where the received new Application will be copied. This ensures that if the update is interrupted the old Application is still available.
Step3	During transfer the received packets of the new Application are written into Bank 1. It is assumed that the packets are transmitted in the right order.
Step4	After all packets of the new Application are received, both the old and the new Application are present in the memory. This ensures that fallback to the old Application is possible if the new Application cannot be activated.
Step5	If the activation of the new Application is successful, Bank 0 is erased to accommodate the new Application. If DFU_APP_DATA_RESERVED is set, Application Data will be preserved.
Step6	As part of activating the new Application it is moved from Bank 1 to Bank 0.
Step7	After the copy procedure is complete, the system can be reset with running the new Application in Bank 0. Bank 1 won't be erased until the Bootloader initializes again.
Step8	If Application Data from the old Application was preserved, the new Application will append any new data written.

Table 1

In the embedded application whenever the DFU mode is set (E003561F - EC48 - 4ED0 - 9F3B - 5419C00A94FD- Climate Profile, DAF4471D – BFB0 – 4DD8 – 9293 – 62AF5F545E31- Grow Profile, 4209DC76 - E433 – 4420 - 83D8 - CDAACCD2E312- Sentry Profile, 497B8E5F - B61E - 4F82 - 8FE9 - B12CF2497338 – Thermo Profile, 35D8C7EA - 9D78 - 43C2 - AB2E - 0E48CAC2DBDA – Water Profile), a general purpose retention register value is set to '1' which will be initially '0'. So after disconnecting the system is incurred with a reset, during reset the code goes through the Bootloader located at 0x03C800 and goes to the Device Firmware Update mode only after checking whether the content of general purpose retention register is set to '1'. Now the device can be uploaded with a new Firmware which goes through all the steps mentioned in the above table.

4 Generic Profile Architecture

In the wimoto custom profiles, the service defines characteristic and functions for exposing the data read from a sensor device and for setting an alarm frame work for the service data. The main application integrates the broadcast application and the alarm service application.

An overview of the code flow is given below.

At start up the flag BROADCAST_MODE will be *false* and the execution enters the *connectable_mode() in* connect.c. It will advertise the sensor data and the alarm service. If a central device connects, the characteristic of the service are displayed to the user who can monitor and modify the values. It will remain in connected state till the user disconnects from the central device. Data are periodically read from the corresponding sensors and checked against the range set by the user. If it is out of range, alarm characteristic will be updated.

If the central device disconnects from the embedded application by setting the 'DFU' characteristic which is provided in the Device Management service, the application exits from the *connectable_mode() by* setting the DFU_MODE_ENABLE flag to true and goes to the boot loader mode where the user can update the device with a new firmware.

If the central device disconnects from the embedded application by setting the 'Mode Switch' characteristic which is provided in the Device Management service, the application exits from the *connectable_mode() by* setting the BROADCAST_MODE flag to true. In main.c, the execution will now enter the function *broadcast_mode()* in broadcast.c.. The *broadcast_mode()* function advertises the enhanced sensor data.

The embedded application remains in the Broadcast mode until a power on reset is done.

5 Climate Profile – System Overview

In the Climate profile light, temperature and humidity values that are read from sensors are broadcast across the BLE transport. There will be characteristic and functions defined for setting a low value and a high value for each of these variables and also to set alarm (on/ off) when any sensor value is out of the set range. If the sensor value is out of range and alarm set is on, a notification will be send to a central device.

5.1 Data Broadcast Application

This program broadcasts the temperature, light and humidity levels read from the sensor as enhanced BLE data. The sample code provided by Wimoto has been integrated with the HTU21D temperature and humidity sensor and ISL29023 light sensor driver programs for implementing the broadcast functionality. This application is implemented by the function *broadcast_mode()* in the file broadcast.c . This function is called from the *main()* in main.c

5.2 Alarm Services

This alarm service implements the functionality for setting a low value and high value for temperature, light and humidity range and the out of range alarm to be set/reset. This application is invoked from a Bluetooth stack event when the user sets the values from the central device.

All the alarm services for the climate profile application are implemented by the function *connectable_mode()* in the file connect.c.

5.2.1 Temperature Alarm Service

The temperature sensor module used for Climate profile is HTU21D. The temperature service and its characteristic and the functions for alarm frame work are implemented in the source file ble_temp_alarm_service.c

The HTU21D temperature and humidity sensor generates 14 bit temperature value which is left intended (i.e. 2 LSB bits will be zero) for temperature measurement. This value is send to the central device like iPhone/Android devices.

In calculations for temperature, this original register content is used without right intending the data. In order to convert this 14 bit data into a degree Celsius value, the user interface application in the iPhone/Android has to use the formula

Temperature °C = -46.85 + (175.72 * Value / 2^16)

The main functions used in the application and service are given below.

Function	***Details***	
ble_temps_level_alarm_check	*Definition*	Reads the current temperature from HTU21D and updates the temperature characteristic and checks for alarm condition.
	Parameter	Pointer to the temperature service structure.
	Return value	NRF_SUCCESS on success, otherwise an error
read_temperature	*Definition*	Read temperature by calling the HTU21D API.
	Return value	Current temperature
current_temperature_char_add	*Definition*	Adds the current temperature characteristic to the service.
	Parameter	Pointer to the temperature service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error
temperature_low_level_char_add	*Definition*	Adds the temperature low value characteristic to the service.
	Parameter	Pointer to the temperature service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error
temperature_high_level_char_add	*Definition*	Adds the temperature high value characteristic to the service.
	Parameter	Pointer to the temperature service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error
temperature_alarm_set_char_add	*Definition*	Adds the temperature alarm set characteristic to the service.
	Parameter	Pointer to the temperature service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error
temperature_alarm_char_add	*Definition*	Adds the temperature alarm characteristic to the service.
	Parameter	Pointer to the temperature service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error

Table 2

5.2.2 Light Alarm Service

This application implements the functionality for setting a low value and high value for light intensity and the out of range alarm to be set/reset. The light intensity service and its characteristic and the functions for the alarm frame work are implemented in the source file ble_light_alarm_service.c.

ISL29023 light sensor module returns a 16 bit data which is directly proportional to the ambient light intensity. In the embedded application code we use 64K as maximum LUX value considering the open environment conditions where the device would be used. The 16 bit register value is transferred to the monitoring device like iPhone/Android devices.

In order to convert this 16 bit data into a LUX value the user interface application in the iPhone/Android has to use the formula

LUX = Value * 0.96

The main functions used in the application and service are given below.

Function	***Details***	
ble_lights_level_alarm_check	*Definition*	Function reads and updates the current light level and checks for alarm condition
	Parameter	Pointer to the Light Service structure.
	Return value	NRF_SUCCESS on success, otherwise an error code.
read_light_level	*Definition*	Function to read light level from ISL29023 sensor.
	Return value	Current light level.
current_light_level_char_add	*Definition*	Adds the current light level characteristic to the service.
	Parameter	Pointer to Light Service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error code.
light_low_value_char_add	*Definition*	Adds the light level low value characteristic to the service.
	Parameter	Pointer to Light Service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error code.

light_high_value_char_add	*Definition*	Adds the light level high value characteristic to the service.
	Parameter	Pointer to Light Service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error code.
light_alarm_set_char_add	*Definition*	Adds the light level alarm set characteristic to the service.
	Parameter	Pointer to Light Service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error code.
light_alarm_char_add	*Definition*	Adds the light level alarm characteristic to the service.
	Parameter	Pointer to Light Service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error code.

Table 3

5.2.3 Humidity Alarm Service

This application implements the functionality for setting a low value and high value for humidity and the out of range alarm to be set/reset. The humidity service and its characteristic and the functions for the alarm framework are implemented in the source file ble_humidity_alarm_service.c.

The HTU21D temperature and humidity sensor generates 12 bit humidity value which is left intended i.e. 4 LSB bits will be zero for humidity measurement. This value is returned to the monitoring device like iPhone/Android devices.

In calculations for humidity, this original register content is used without right intending the data. In order to convert this 12 bit data into a relative humidity value, the user interface application in the iPhone/Android has to use the formula

Relative Humidity %RH = -6 + (125 * Value / 2^16)

The main functions used in the service are given below

Function	***Details***	
ble_hums_level_alarm_check	*Definition*	Function reads and updates the current humidity level and checks for alarm condition
	Parameter	Pointer to the Humidity Service structure.
	Return value	NRF_SUCCESS on success, otherwise an error code.
read_hum_level	*Definition*	Function to read humidity level from HTU21D
	Return value	Current humidity level.
current_hum_level_char_add	*Definition*	Adds the current humidity level characteristic to the service.
	Parameter	Pointer to the Humidity Service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error code.
hum_low_value_char_add	*Definition*	Adds the humidity level low value characteristic to the service.
	Parameter	Pointer to the Humidity Service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error code.
hum_high_value_char_add	*Definition*	Adds the humidity level high value characteristic to the service.
	Parameter	Pointer to the Humidity Service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error code.
hum_alarm_set_char_add	*Definition*	Adds the humidity level alarm set characteristic to the service.
	Parameter	Pointer to the Humidity Service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error code.
hum_alarm_char_add	*Definition*	Adds the humidity alarm characteristic to the service.
	Parameter	Pointer to the Humidity Service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error code.

Table 4

5.3 Device Management Service

Device Management service mainly implements three functionality

1) Device Firmware Updates
   Enables the user to update the device with a new Firmware which can be updated over the air. Once this characteristic is set and the device is disconnected , the code enters into boot loader mode where the device can be updated with new firmware
2) Mode Switching (switching from Connectable to Broadcast mode)
   Enables the user to switch from Connectable/peripheral mode to Broadcast mode. Once this characteristic is set and the device is disconnected, the code enters into Broadcast mode
3) Time stamp
   Displays current value of day, month, year and time, time gets updated every second. User can set the value of day, month, year and time so that after setting new value time gets updated from the value which was set.

The Device Management service and its characteristic and the functions are implemented in the source file ble_device_mgmt_service.c

The main functions used in the application and service are given below.

Function	**Details**	
ble_device_mgmt_check	*Definition*	Function that checks for device management condition
	Parameter	Device Management Service structure
	Return value	NRF_SUCCESS on success, otherwise an error code
dfu_mode_char_add	*Definition*	Function for adding the characteristic for Device firmware update the mode
	Parameter	Device Management Service structure
	Parameter	Information needed to initialize the service.
	Return value	NRF_SUCCESS on success, otherwise an error code
switch_mode_char_add	*Definition*	Function for adding the characteristic to switch the mode from Peripheral to Broadcast
	Parameter1	Device Management Service structure
	Parameter2	Information needed to initialize the service
	Return value	NRF_SUCCESS on success, otherwise an error code

time_stamp_char_add	*Definition*	Function for adding the characteristics for real time tracking
	Parameter1	Device Management Service structure
	Parameter2	Information needed to initialize the service
	Return value	NRF_SUCCESS on success, otherwise an error code

Table 5

5.4 Data Logger Service

Data logger service implements the data logging functionality. The data read from the sensors, along with time stamp are stored in the flash memory pages allocated for data logging. Each flash page is of 1024 bytes. A word in the flash is 32 bits. Three pages are allocated for data logging currently; page addresses 0xC0, 0xC1, and 0xC2. Start addresses of these pages are 0x30000, 0x30400, and 0x30800. Data logging starts if the Data log enable flag is set by the user.

A length of one record written to flash has four words.

- 2 words of time stamp
- 2 words of data from sensors

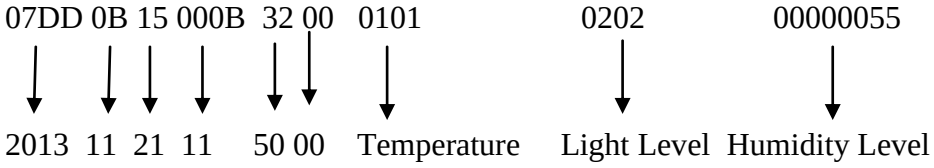
The data from sensors is packed in different formats in the two words in each profile. When a page is completely written, it advances to the next page, erases the page fully, and then writes each block sequentially.

The first two words of each block are used for time stamp. The data format for logging the sensor data in each profile is described below.

5.2.5.1 Climate Profile

Temperature and Light level are stored in the third word.
Humidity level is stored in fourth word.

An example of the data array received in master control panel is
07DD0B15000B32000101020200000055 which is arranged as shown below

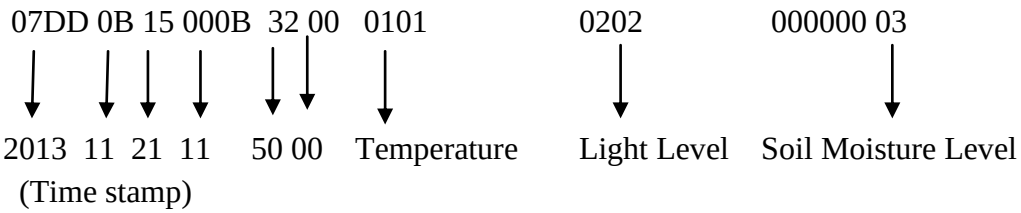


5.2.5.2 Grow Profile

Temperature and Light level are stored in the third word.
Soil moisture level is stored in fourth word.

An example of the data array received in master control panel is

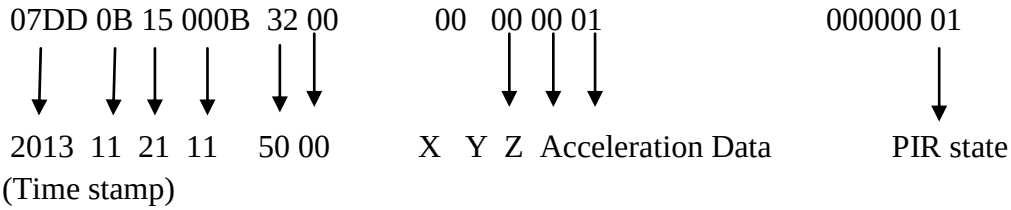
07DD0B15000B32000101020200000003 which is arranged as shown below



5.2.5.3 Sentry Profile

Accelerometer x, y, z acceleration data in the third word.
Passive infrared sensor data in the fourth word.

An example of the data array received in master control panel is 07DD0B15000B32000000000100000001 which is arranged as shown below

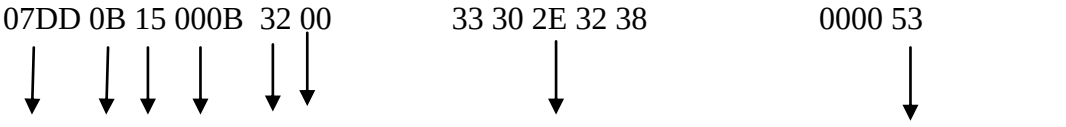


5.2.5.4 Thermo Profile

Thermopile temperature is stored in third word, partially.
Thermopile temperature, remaining and probe temperature are packed together into

fourth word

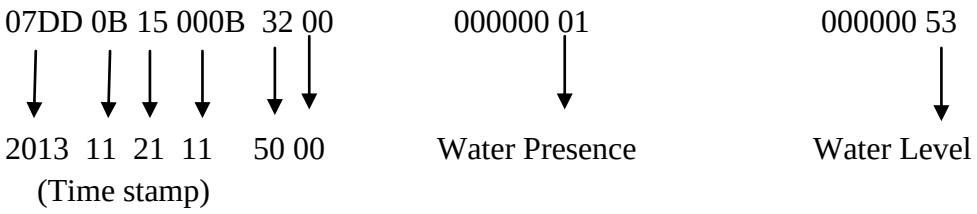
An example of the data array received in master control panel is
07DD0B15000B320033302E3238000053 which is arranged as shown below



5.2.5.5 Water Profile

Water presence data is stored in the third word.
Water level data is stored in fourth word.

An example of the data array received in master control panel is
07DD0B15000B32000000000100000053 which is arranged as shown below



The main functions used in the service are given below.

Function	Details	
data_logger_enable_char_add	*Definition*	Function for adding the data logger enable characteristics.
	Parameter	Data logger service structure.
	Parameter	Information needed to initialize the service.
	Return value	NRF_SUCCESS on success, otherwise an error code.
data_char_add	*Definition*	Function for adding the data characteristic.
	Parameter	Data logger service structure.
	Parameter	Information needed to initialize the service.
	Return value	NRF_SUCCESS on success, otherwise an error code.
read_data_switch_char_add	*Definition*	Function for adding the read data switch characteristic
	Parameter	Data logger service structure.
	Parameter	Information needed to initialize the service.
	Return value	NRF_SUCCESS on success, otherwise an error code.
write_data_flash	*Definition*	Function write sensor data to flash
	Parameter	Data logger service structure.
	Return value	NRF_SUCCESS on success, otherwise an error code.
send_data	*Definition*	Function to send data to the connected BLE central device.
	Parameter	Data logger service structure.
	Return value	NRF_SUCCESS on success, otherwise an error code.

read_data_flash	*Definition*	Function reading data to flash and sending to the connected BLE central device.
	Parameter	Data logger service structure.
	Return value	NRF_SUCCESS on success, otherwise an error code.
send_data_to_central	*Definition*	Function to send the data to the connected central device.
	Parameter	Data logger service structure
	Return value	NRF_SUCCESS on success, otherwise an error code.
get_dlog_en_val	*Definition*	Function to get the data logger enables switch characteristics.
	Parameter	Data logger service structure.
	Return value	NRF_SUCCESS on success, otherwise an error code.
get_read_data_switch	*Definition*	Function to get the data read data switch characteristics
	Parameter1	Data logger service structure.
	Return value	NRF_SUCCESS on success, otherwise an error code.
reset_data_log	*Definition*	Function to reset the data logger enable and read data switch characteristics
	Parameter1	Data logger service structure.
	Return value	NRF_SUCCESS on success, otherwise an error code.

Table 6

6 Grow Profile – System Overview

In the grow profile Light, Temperature and Soil Moisture values that are read from sensor devices are broadcast across the BLE transport. There will be characteristic and functions defined for setting a low value and a high value for temperature, light and soil moisture level and also to set alarm (on/ off) when any sensor value is out of the set range. If the sensor value is out of range and alarm set is on, a notification will be send to a central device.

6.1 Data broadcast application

This program broadcasts the data read from the sensor as enhanced BLE data. The sample code provided by Wimoto has been integrated with the TMP102 , ISL29023 and an analog proprietary soil moisture sensor driver programs for implementing the broadcast functionality. This application is implemented by the function *broadcast_mode()* in the file broadcast.c . This function is called from the *main()* in main.c.

6.2 Alarm Service

This alarm service implements the functionality for setting a low value and high value for temperature/ light/ soil moisture level and the out of range alarm to be set/reset. This application is invoked from a Bluetooth stack event when the user sets the values from the central role device.

All the alarm alarm services for the Grow profile application is implemented by the function *connectable_mode()* in the file connect.c.

6.2.1 Temperature Alarm Service

The temperature sensor module used for Grow profile is TMP102. The temperature service and its characteristic and the functions for alarm frame work are implemented in the source file ble_temp_alarm_service.c
The TMP102 temperature sensor generates 12 temperature value which is left intended (i.e. 4 LSB bits will be zero temperature measurement), this value is right intended by 4 bits and is returned to the monitoring device like iPhone/Android devices

In calculations for temperature, this data returned to the monitoring iPhone/Android devices, the user interface application has to perform the following calculation to convert temperature into degree Celsius

Temperature °C = Value * 0.0625

The main functions used in the application and service are given below.

Function	***Details***	
ble_temps_level_alarm_check	*Definition*	Reads the current temperature from TMP102 and updates the temperature characteristic and checks for alarm condition.
	Parameter	Pointer to the temperature service structure.
	Return value	NRF_SUCCESS on success, otherwise an error
read_temperature	*Definition*	Read temperature by calling the TMP102 API.
	Return value	Current temperature
current_temperature_char_add	*Definition*	Adds the current temperature characteristic to the service.
	Parameter	Pointer to the temperature service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error
temperature_low_level_char_add	*Definition*	Adds the temperature low value characteristic to the service.
	Parameter	Pointer to the temperature service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error
temperature_high_level_char_add	*Definition*	Adds the temperature high value characteristic to the service.
	Parameter	Pointer to the temperature service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error
temperature_alarm_set_char_add	*Definition*	Adds the temperature alarm set characteristic to the service.
	Parameter	Pointer to the temperature service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error
temperature_alarm_char_add	*Definition*	Adds the temperature alarm characteristic to the service.
	Parameter	Pointer to the temperature service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error

Table 7

6.2.2 Light Alarm Service

This application implements the functionality for setting a low value and high value for light intensity and the out of range alarm to be set/reset.

The light intensity alarm service application is implemented by the function *connectable_mode()* in the file connect.c. The light intensity service and its characteristic and the functions for alarm frame work are implemented in the source file ble_light_alarm_service.c

ISL29023 light sensor module returns a 16 bit data which is directly proportional to the ambient light intensity. In the embedded application code we use 64K as maximum LUX value considering the open environment conditions where it would be used. The 16 bit register value thus read in the monitoring device like iPhone/Android devices.

In order to convert this 16bit data into a LUX value the user interface application in the iPhone/Android has to use the formula

LUX = value * 0.96

The main functions used in the application and service are given below.

Function	***Details***	
ble_lights_level_alarm_check	*Definition*	Function reads and updates the current light level and checks for alarm condition
	Parameter	Pointer to the Light Service structure.
	Return value	NRF_SUCCESS on success, otherwise an error code.
read_light_level	*Definition*	Function to read light level from ISL29023 sensor.
	Return value	Current light level.
current_light_level_char_add	*Definition*	Adds the current light level characteristic to the service.
	Parameter	Pointer to Light Service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error code.

light_low_value_char_add	*Definition*	Adds the light level low value characteristic to the service.
	Parameter	Pointer to Light Service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error code.
light_high_value_char_add	*Definition*	Adds the light level high value characteristic to the service.
	Parameter	Pointer to Light Service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error code.
light_alarm_set_char_add	*Definition*	Adds the light level alarm set characteristic to the service.
	Parameter	Pointer to Light Service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error code.
light_alarm_char_add	*Definition*	Adds the level alarm characteristic to the service.
	Parameter	Pointer to Light Service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error code.

Table 8

6.2.3 Soil Moisture Service

This application implements the functionality for reading soil moisture level from an analog sensor using ADC to convert analog data into digital format. This application is invoked from a Bluetooth stack event when the user sets the values from the central role device.

The soil moisture level alarm service application is implemented by the function *connectable_mode()* in the file connect.c. The soil moisture service and its characteristic and the functions for alarm frame work are implemented in the source file ble_soil_alarm_service.c.

Since the Soil moisture sensor was not available while testing data reading function only was implemented. The input pin for the ADC conversion is **P0.05.** While ADC conversion 1MHz signal is generated at the pin **P0.08** for the soil moisture sensor.

The main functions used in the application and service are given below

Function	***Details***	
ble_soils_level_alarm_check	*Definition*	Function reads and updates the current Soil Moisture level and checks for alarm condition
	Parameter	Pointer to the Soil Moisture Service structure.
	Return value	NRF_SUCCESS on success, otherwise an error code.
read_soil_mois_level	*Definition*	Function to read soil moisture level
	Return value	Result of ADC after conversion
current_soil_mois_level_char_add	*Definition*	Adds the current soil moisture level characteristic to the service.
	Parameter	Pointer to Soil moisture Service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error code.
soil_mois_low_value_char_add	*Definition*	Adds the soil moisture low value characteristic to the service.
	Parameter	Pointer to Soil moisture Service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error code.
soil_mois_high_value_char_add	*Definition*	Adds the soil moisture high value characteristic to the service.
	Parameter	Pointer to Soil moisture Service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error code.
soil_mois_alarm_set_char_add	*Definition*	Adds the soil moisture alarm set characteristic to the service.
	Parameter	Pointer to Soil moisture Service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise error code

soil_mois_alarm_char_add	*Definition*	Adds the soil moisture alarm set characteristic to the service.
	Parameter	Pointer to Soil moisture Service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error code.

Table 9

6.3 Device Management Service

Device Management service mainly implements functionality for Device Firmware Udate over the air ,Mode switch characteristic (switching from Connectable/Peripheral to Broadcast mode) and Time Stamp (Real time tracking)

Refer to Page Number 15

6.4 Data Logger Service

Data logger service implements the data logging functionality. The data read from the sensors, along with time stamp are stored in the flash memory pages allocated for data logging

Refer to Page Number 16

7 Sentry Profile – System Overview

In the Sentry profile the data's read from Passive Infrared Sensor and Accelerometer are broadcast across the BLE transport. There will be characteristic and functions defined to set alarm (on/ off) if the PIR sensor triggers an alert and if the Accelerometer detects a movement. If the sensor triggers an alert based on the predefined conditions then alarm set is on, a notification will be send to a central device.

7.1 Data Broadcast Application

This program broadcasts the Passive infrared output (digital data) and Accelerometer X, Y, Z acceleration data read from the sensor as enhanced BLE data. The sample code provided by Wimoto has been integrated with the digital Passive infrared sensor and MMA8653 accelerometer driver programs for implementing the broadcast functionality. This application is implemented by the function *broadcast_mode()* in the file broadcast.c. This function is called from the *main()* in main.c

7.2 Alarm Services

This alarm service implements the functionality for setting alarm for Passive infrared and Accelerometer sensors. Alarm set/reset for PIR and Accelerometer is implemented., an extra characteristic called 'Alarm Clear' is added in the Accelerometer alarm service for clearing the alarm value even if the alarm set is enabled ,i.e. explicitly specifying that user understood the situation and problem is not critical.

This application is invoked from a Bluetooth stack event when the user sets the values from the central device. All the alarm services for the water profile application are implemented by the function *connectable_mode()* in the file connect.c.

7.2.1 Passive Infrared Alarm Service

The Passive infrared sensor module used for Sentry profile is digital sensor. The passive infrared service and its characteristic and the functions for alarm frame work are implemented in the source file ble_pir_alarm_service.c. This application is invoked from a Bluetooth stack event when the user sets the values from the central role device.

The AMS312 is a digital PIR sensor with active high output which is connected to the pin **P0.02** on nRF51822 Evaluation Kit**.** Whenever a change in logic level occurs at the desired pin **P0.02** the code enters a check condition which reads the current state of the pin and if it is active high it indicates a PIR sensor alarm.

The main functions used in the application and service are given below.

Function	Details	
ble_pir_alarm_check	*Definition*	Function reads and updates the current PIR data and checks for alarm condition
	Parameter	PIR Service structure.
	Return value	NRF_SUCCESS on success, otherwise an error code
pir_state_char_add	*Definition*	Function for adding the current PIR presence characteristics
	Parameter	PIR Service structure.
	Parameter	Information needed to initialize the service.
	Return value	NRF_SUCCESS on success, otherwise an error code
pir_alarm_set_char_add	*Definition*	Function for adding the PIR alarm set characteristics.
	Parameter	PIR Service structure.
	Parameter	Information needed to initialize the service.
	Return value	NRF_SUCCESS on success, otherwise an error code
pir_alarm_char_add	*Definition*	Function for adding the PIR alarm characteristics
	Parameter	PIR Service structure.
	Parameter	Information needed to initialize the service.
	Return value	NRF_SUCCESS on success, otherwise an error code

Table 10

7.2.2 Accelerometer Alarm Service

The Accelerometer alarm service application is implemented by the function *connectable_mode()* in the file connect.c. The Accelerometer service and its characteristic and the functions for alarm frame work are implemented in the source file ble_acclerometer_alarm_service.c

Accelerometer is configured such that when a movement occurs it generates an interrupt on the INT1 pin on MMA8653 accelerometer which is connected to the pin **P0.04** on nRF51822 Evaluation Kit. . The driver code is designed such that setting a threshold value for acceleration 0.504g, if the acceleration exceeds beyond that level an interrupt will be generated on INT1 pin of MMA8653. When a active low logic level occurs at the **P0.04** pin the X, Y, Z data's are read from the MMA8653 accelerometer sensor. Also a debounce value (5) for the alarm generation is also set in order to avoid the effects of undesired situvations.

The main functions used in the application and service are given below.

Function	Details	
ble_movement_alarm_check	*Definition*	Function reads and updates the current movement and checks for alarm condition
	Parameter	Movement Service structure.
	Return value	NRF_SUCCESS on success, otherwise an error code.
reset_alarm	*Definition*	Function to reset alarm if user specifies alarm can be cleared
	Parameter	Movement Service structure
	Return value	NRF_SUCCESS on success, otherwise an error code.
current_xyz_coordinates_char_add	*Definition*	Function for adding the current X Y Z acceleration data characteristics
	Parameter1	Movement Service structure
	Parameter2	Information needed to initialize the service.
	Return value	NRF_SUCCESS on success, otherwise an error code.
movement_alarm_set_char_add	*Definition*	Function for adding the movement alarm set characteristics.
	Parameter1	Movement Service structure
	Parameter2	Information needed to initialize the service.
	Return value	NRF_SUCCESS on success, otherwise an error code.
movement_alarm_clear_char_add	*Definition*	Function for adding the movement alarm clear characteristics.
	Parameter	Movement Service structure
	Parameter2	Information needed to initialize the service.
	Return value	NRF_SUCCESS on success, otherwise an error code.
movement_alarm_char_add	*Definition*	Function for adding the movement alarm characteristics.
	Parameter1	Movement Service structure
	Parameter2	Information needed to initialize the service.
	Return value	NRF_SUCCESS on success, otherwise an error code.

Table 11

7.3 Device Management Service

Device Management service mainly implements functionality for Device Firmware Udate over the air and Mode switch characteristic (switching from Connectable/Peripheral to Broadcast mode)

Refer to Page Number 15

7.4 Data Logger Service

Data logger service implements the data logging functionality. The data read from the sensors, along with time stamp are stored in the flash memory pages allocated for data logging

Refer to Page Number 16

8 Thermo Profile – System Overview

In the thermo profile, Thermopile Temperature and Probe Temperature values that are read from sensor devices are broadcast across the BLE transport. There will be characteristic and functions defined for setting a low value and a high value for each of these variables and also to set alarm (on/ off) when any sensor value is out of the set range. If the sensor value is out of range and alarm set is on, a notification will be send to a central device.

8.1 Data Broadcast Application

This program broadcasts the thermopile and probe temperatures, read from the sensor as enhanced BLE data. The sample code provided by Wimoto has been integrated with the TMP006 thermopile temperature sensor driver programs for implementing the broadcast functionality. This application is implemented by the function *broadcast_mode()* in the file broadcast.c . This function is called from the *main()* in main.c

8.2 Alarm Services

This alarm service implements the functionality for setting a low value and high value for temperature, light and humidity range and the out of range alarm to be set/reset. This application is invoked from a Bluetooth stack event when the user sets the values from the central device.

All the alarm services for the Thermo Profile application are implemented by the function *connectable_mode()* in the file connect.c.

8.2.1 Thermopile Temperature Alarm Service

The thermopile temperature sensor module used for Thermo profile is TMP006. The temperature service and its characteristic and the functions for alarm frame work are implemented in the source file ble_thermop_alarm_service.c

The TMP006 temperature sensor generates two 16 bit values in ambient and object registers. This value is used to calculate the target temperature and the floating point value is returned as a string to the monitoring device like iPhone/Android devices.

In calculations for target temperature the equation used is
Temperature °C = -46.85 + (175.72 * value / 2^16),

The user interface application in iPhone/Android doesn't have to bother about this conversion as it is taken care in the code.

The main functions used in the application and service are given below.

Function	Details	
ble_thermops_level_alarm_check	*Definition*	Reads the current Thermopile temperature from TMP006 updates the temperature characteristic and checks the alarm condition.
	Parameter	Pointer to the Thermopile Temperature service structure.
	Return value	NRF_SUCCESS on success, otherwise an error code.
read_thermopile	*Definition*	Read thermopile temperature by calling the TMP006 API.
	Return value	Current thermopile temperature
current_thermopile_char_add	*Definition*	Adds the current thermopile temperature characteristic to the service.
	Parameter	Pointer to the Thermopile Temperature service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error code.
thermopile_low_value_char_add	*Definition*	Adds the thermopile temperature low value characteristic to the service.
	Parameter	Pointer to the Thermopile Temperature service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error code.
thermopile_high_value_char_add	*Definition*	Adds the thermopile temperature high value characteristic to the service.
	Parameter	Pointer to the Thermopile Temperature service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error code.

thermopile_alarm_set_char_add	*Definition*	Adds the thermopile temperature alarm set characteristic to the service.
	Parameter	Pointer to the Thermopile Temperature service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error code.
thermopile_alarm_char_add	*Definition*	Adds the thermopile temperature alarm characteristic to the service.
	Parameter	Pointer to the Thermopile Temperature service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error code.

Table 12

8.2.2 Probe Temperature Service

This application implements the functionality for reading probe temperature level from an analog sensor using ADC conversion to convert analog data into digital format. This application is invoked from a Bluetooth stack event when the user sets the values from the central role device.

The probe temperature level alarm service application is implemented by the function *connectable_mode()* in the file connect.c. The probe temperature service and its characteristic and the functions for alarm frame work are implemented in the source file ble_probe_alarm_service.c.

Since the probe temperature sensor was not available while testing data reading function only was implemented.

The main functions used in the application and service are given below.

Function	**Details**	
ble_probes_level_alarm_check	*Definition*	Function reads and updates the current probe temperature level and checks for alarm condition
	Parameter	Function to read Probe Temperature level
	Return value	NRF_SUCCESS on success, otherwise an error code.

read_probe_temp_level	*Definition*	Function to read probe temperature level
	Return value	Result of ADC after conversion
current_probe_temp_level_char_add	*Definition*	Adds the current probe temperature level characteristic to the service
	Parameter	Pointer to the Probe Temperature Service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error code.
probe_temp_low_value_char_add	*Definition*	Adds the probe temperature level low value characteristic to the service.
	Parameter	Pointer to the Probe Temperature Service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error code.
probe_temp_high_value_char_add	*Definition*	Adds the probe temperature level high value characteristic to the service.
	Parameter	Pointer to the Probe Temperature Service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error code.
probe_temp_alarm_set_char_add	*Definition*	Adds the probe temperature level alarm set characteristic to the service.
	Parameter	Pointer to the Probe Temperature Service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error code.
probe_temp_alarm_char_add	*Definition*	Adds the probe temperature level alarm set characteristic to the service.
	Parameter	Pointer to the Probe Temperature Service structure.
	Parameter	Pointer to initial value structure
	Return value	NRF_SUCCESS on success, otherwise an error code.

Table 13

8.3 Device Management Service

Device Management service mainly implements functionality for Device Firmware Udate over the air and Mode switch characteristic (switching from Connectable/Peripheral to Broadcast mode)

Refer to Page Number 15

8.4 Data Logger Service

Data logger service implements the data logging functionality. The data read from the sensors, along with time stamp are stored in the flash memory pages allocated for data logging

Refer to Page Number 16

9 Water Profile – System Overview

In the water profile Water Presence and Water Level values that are read from sensor devices are broadcast across the BLE transport. There will be characteristic and functions defined for setting a low value and a high value for water level service and also to set alarm (on/ off) if the water level value is out of the set range. If the sensor value is out of range and alarm set is on, a notification will be send to a central device.
Water absence is also indicated by another alarm which can be set by a set alarm (on/off).

9.1 Data Broadcast Application

This program broadcasts the water level and water presence read from the sensor as enhanced BLE data. The sample code provided by Wimoto has been integrated with the analog water level sensor driver programs for implementing the broadcast functionality. This application is implemented by the function *broadcast_mode()* in the file broadcast.c. This function is called from the *main()* in main.c

9.2 Alarm Services

This alarm service implements the functionality for setting a low value and high value for water level and the out of range alarm to be set/reset. Alarm set/reset for water presence is implemented. This application is invoked from a Bluetooth stack event when the user sets the values from the central device.

All the alarm services for the water profile application are implemented by the function *connectable_mode()* in the file connect.c.

9.2.1 Water Level Alarm Service

The water level sensor module used for water profile is an analog sensor. The water level service and its characteristic and the functions for alarm frame work are implemented in the source file ble_waterl_alarm_service.c. This application is invoked from a Bluetooth stack event when the user sets the values from the central role device.

Since water level sensor is an analog sensor ADC conversion is required for converting the analog data into digital format. 8 bit digital data is obtained from the ADC after conversion. The input pin for the ADC conversion is **P0.01.**

The main functions used in the application and service are given below.

Function	Details	
ble_waterls_level_alarm_check	*Definition*	Function reads and updates the current water level and checks for alarm condition
	Parameter	Water level Service structure.
	Return value	NRF_SUCCESS on success, otherwise an error code.
read_waterl_level	*Definition*	Function to read water level from sensor interfaced to ADC
	Return value	Current water level
current_waterl_level_level_char_add	*Definition*	Adds the current water level characteristics.
	Parameter	Water level Service structure
	Parameter	Information needed to initialize the service.
	Return value	NRF_SUCCESS on success, otherwise an error code.
waterl_level_low_value_char_add	*Definition*	Adds the water level low value characteristics.
	Parameter	Water level Service structure.
	Parameter	Information needed to initialize the service.
	Return value	NRF_SUCCESS on success, otherwise an error code.
waterl_level_high_value_char_add	*Definition*	Adds the water level high value characteristics
	Parameter	Water level Service structure
	Parameter	Information needed to initialize the service.
	Return value	NRF_SUCCESS on success, otherwise an error code.
waterl_level_alarm_set_char_add	*Parameter*	Adds the water level alarm set characteristic.
	Parameter	Water level Service structure.
		Information needed to initialize the service.
	Return value	NRF_SUCCESS on success, otherwise an error code.
waterl_level_alarm_char_add	*Definition*	Adds the water level alarm characteristics.
	Parameter	Water level Service structure.
	Parameter	Information needed to initialize the service.
	Return value	NRF_SUCCESS on success, otherwise an error code.

Table 14

9.2.2 Water Presence Alarm Service

This application implements the functionality for detecting water presence by using GPIO pins on nRF51822 evaluation kit.

The Water Presence alarm service application is implemented by the function *connectable_mode()* in the file connect.c. The Water Presence service and its characteristic and the functions for alarm frame work are implemented in the source file ble_waterp_alarm_service.c

Initially the pin for detecting the water presence **P0.00** is pulled-up so that default value of the pin always will be active-high and whenever water is not present the pin will be pulled low. Whenever a change in logic level occurs at the desired pin **P0.00** for detecting the water presence an alert is produced and current water presence is checked. If water is not present alarm is set.

The water presence service and its characteristic and the functions for alarm frame work are implemented in the source file ble_waterp_alarm_service.c.

The main functions used in the application and service are given below.

Function	**Details**	
ble_waterps_alarm_check	*Definition*	Function reads and updates the current water presence and checks for alarm condition.
	Parameter	Water presence Service structure.
	Return value	NRF_SUCCESS on success, otherwise an error code.
current_waterpresence_char_add	*Definition*	Function for adding the current water presence characteristics.
	Parameter1	Water presence Service structure.
	Parameter2	Information needed to initialize the service.
	Return value	NRF_SUCCESS on success, otherwise an error code.
waterpresence_alarm_set_char_add	*Definition*	Function for adding the water presence alarm set characteristics.
	Parameter1	Water presence Service structure.
	Parameter2	Information needed to initialize the service.
	Return value	NRF_SUCCESS on success, otherwise an error code.

waterpresence_alarm_char_add	*Definition*	Function for adding the water presence alarm characteristics.
	Parameter1	Water presence Service structure.
	Parameter2	Information needed to initialize the service.
	Return value	NRF_SUCCESS on success, otherwise an error code.

Table 15

9.3 Device Management Service

Device Management service mainly implements functionality for Device Firmware Update over the air and Mode switch characteristic (switching from Connectable/Peripheral to Broadcast mode)

Refer to Page Number 15

9.4 Data Logger Service

Data logger service implements the data logging functionality. The data read from the sensors, along with time stamp are stored in the flash memory pages allocated for data logging

Refer to Page Number 16

10 Drivers for Sensor Modules

Wimoto project contains different sensors like TMP102, TMP006, HTU21D, etc., to measure various environmental parameters. Here, as the second phase of project drivers for

TMP102	Temperature sensor
TMP006	Temperature sensor
ISL29023	Light sensor
MMA8653FC	Accelerometer
HTU21D	Humidity and temperature sensor
Analog Soil Moisture Sensor	

are provided. Pin number 24 and 25 of nRF51822 Evaluation Kit board are configured for the I2C communication. Pin 24 is used for SCL (serial clock) and pin 25 is for SDA (serial data). The initializations for I2C communication are done prior to data transfer. The I2C bus is initialized using *twi_init()* function.

Note: The private functions for implementing the public functions (API's) are not mentioned here

10.1 TMP102 Driver

TMP102 is temperature sensor which outputs temperature in digital format can be accessed using I2C protocol or Two Wire Interface. The slave address of TMP102 is found out using the pin to which pin **ADD0** is connected. In the driver program it is assumed that the **ADD0** pin is connected to **ground**.

There are two functions which act as the API in the driver section of the TMP102 file as per the requirement for the specifications. They are given below

Function	***Details***	
config_tmp102_shutdown_mode (void)	*Definition*	Configure TMP102 in shut down mode
	Return value	Boolean value(Success – true, Failure - false)
get_tmp102_oneshot_temp (void)	*Definition*	Get temperature only when needed. Enables temperature conversion in TMP102 ,only when this function is called.
	Return value	Content of the temperature register in the correct format(after eliminating four '0' bits in the Least Significant Byte)

Table 16

The maximum temperature read by TMP102 is 0x7FF0 (128°C) and negative temperature is 0xC900 (-55°C). The lower nibble in the 16 bit data (0-3 bits) '0' is

neglected because it is by default set to '0' in the register, so while retrieving the data the temperature register is right shifted four bits (taken care in the driver code).

.

10.2 TMP006 Driver

TMP006 is a temperature sensor which is used to measure target temperature using IR light emitted from the target. Target temperature is calculated using the contents of T-ambient and V-object registers, these data's are in a digital format an can be read using I2C protocol or Two Wire Interface. While testing the breakout board used had pins **ADR1** and **ADR0** are connected to **ground,** so the slave address of TMP006 becomes 1000000x, where x is the R/W (read/write) bit, so the 8bit data (slave address +R/W bit) for start condition for writing becomes 0x80 and for reading becomes 0x81.

There are two functions which act as the API in the driver section of the TMP006 file as per the requirement for the specifications. They are given below

Function	**Details**	
TMP006_enable_powerdown_mode (void)	*Definition*	Enable power down mode of TMP006
	Return value	Boolean value (Success –true, Failure-false)
TMP006_get_onetime_data (void)	*Definition*	Function to read the value of V-object & T-ambient registers one time
	Return value	Returns the 32 bit value (two 16 bit V-object & T-ambient registers combined together)

Table 17

The data in the T-ambient register is 12 bit with 0th and 1st bit set to '0' as default so the correct data of T-ambient register is obtained by right shifting the data by two bits, this is taken care in the program.

10.3 ISL29023 Driver

ISL29023 is light sensor which outputs light in terms of LUX value by a digital format and can be accessed using I2C protocol or Two Wire Interface. The slave address of ISL29023 is 1000100x , where x is the R/W (read/write) bit, so the 8bit data(slave address +R/W bit) for start condition for writing becomes 0x88 and for reading becomes 0x89.

There are two functions which act as the API in the driver section of the ISL29023 file as per the requirement for the specifications. They are given below

Function	Details	
ISL29023_config_FSR_and_powerdown (void)	*Definition*	Change Full Scale Reading of LUX, after that enable Power down mode of ISL29023
	Return value	Boolean value (Success –true, Failure-false)
ISL29023_get_one_time_ALS (void)	*Definition*	Function to enable One time Light Sensing mode, after one conversion ISL29023 goes to power down mode automatically
	Return value	Contents of DATA -MSB & DATA-LSB registers (16 bit)

Table 18

10.4 MMA8653FC Driver

MMA8653 is accelerometer module which outputs acceleration data about the X, Y, Z directions. The data's are available in the X-out, Y-out and Z-out registers of MMA8653. These values are represented in a digital format and can be read using I2C protocol or Two Wire Interface. The slave address of MMA7660FC is 0011101x, where x is the R/W (read/write) bit, so the 8bit data (slave address +R/W bit) for start condition for writing becomes 0x3A and for reading becomes 0x3B.

There are two functions which act as the API in the driver section of the MMA8653 file as per the requirement for the specifications. They are given below

Function	Details	
MMA8653_ReadXYZdata	*Definition*	Function to read data from OUT_X_MSB, OUT_Y_MSB, OUT_Z_MSB registers
	Parameter	Pointer to the 32 bit value in the called function
	Return value	True- on success, False –on failure
MMMA8653_Init	*Definition*	Initialize all the associated registers for interrupt generation on motion detection in MMA8653
	Return value	True- on success, False –on failure

Table 19

10.5 HTU21D Driver

HTU21D is a humidity and temperature sensor which outputs temperature and humidity in digital format and can be accessed using I2C protocol or Two Wire Interface. The slave address of HTU21D is 1000000x, where x is the R/W (read/write) bit, so the 8bit

data (slave address +R/W bit) for start condition for writing becomes 0x80 and for reading becomes 0x81.

There are five functions which act as the API in the driver section of the HTU21D file as per the requirement for the specifications. They are given below

Function	**Details**	
eDRV_HTU21_Reset(void)	*Definition*	Enable Soft reset
	Return value	Boolean value(Success –true, Failure-false)
eDRV_HTU21_MeasureTemperature(void)	*Definition*	Read 16 bit Temperature value (status bits cleared)
	Return value	16 bit Temperature data with status bit cleared
HTU21D_WriteToUserRegister (uint8_t)	*Definition*	Write data to User register
	Parameter1	8 bit data to be written
	Return value	Boolean value(Success –true, Failure-false)
HTU21D_ReadUserRegister (void)	*Definition*	Read the contents of User register
	Return value	bit user register data
eDRV_HTU21_MeasureHumidity (void)	*Definition*	Read 16 bit Humidity value (status bits cleared)
	Return value	16 bit Humidity data with status bit cleared

Table 20

10.6 Analog Sensors

Analog sensors use the ADC for converting the analog value into digital value. So each analog sensor requires the ADC initialization before starting the conversion.

Function	**Details**	
adc_init	*Definition*	Initializes ADC for with the corresponding pin number as specified per PSEL register, and 8 bit resolution
	Return value	void

Table 21

10.6.1 Soil Moisture Sensor

An analogue proprietary soil moisture sensor is used. The analogue value is converted to a digital value using the internal ADC of nRF51822. The analogue soil moisture sensor should be connected to the P0.01 pin of nRF51822 evaluation board kit.

The functions for Soil moisture sensor are follows

Function	Details	
do_soil_moisture_measurement	*Definition*	Function to read soil moisture value from the ADC after conversion
	Return value	8 bit data soil moisture data after the ADC conversion

Table 22

10.6.2 Water Level Sensor

An analogue proprietary water level sensor is used. The analogue value is converted to a digital value using the internal ADC of nRF51822. The analogue water level sensor should be connected to the **P0.01** pin of nRF51822 evaluation board kit.

Function	Details	
do_waterl_adc_measurement	*Definition*	Function to read the sensor output value after using ADC (conversion of analog data into digital data using ADC)
	Return value	8 bit data water level data after the ADC conversion

Table 23

10.6.3 Probe Temperature Sensor

An analogue proprietary probe temperature sensor is used. The analogue value is converted to a digital value using the internal ADC of nRF51822. The analogue probe temperature sensor should be connected to the **P0.03** pin of nRF51822 evaluation board kit.

Function	Details	
do_probe_temperature_measurement	*Definition*	Function to read the sensor output value after using ADC (conversion of analog data into digital data using ADC)
	Return value	8 bit probe temperature data after the ADC conversion

Table 24

10.7 Digital Sensors

10.7 .1 AMS312 Passive Infrared Sensor

An digital proprietary Passive infrared sensor is used. The ams312 sensor generates an **active high** output voltage for an alert. The GPIO pins of nRF51822 evaluation board kit is used to detect the output from this sensor and depending upon the values of logic level at the monitoring pin necessary actions are taken accordingly in the alarm framework. The digital proprietary Passive infrared sensor should be connected to the **P0.02** pin of nRF51822 evaluation board kit.

10.7.2 Water Presence Sensor

An digital proprietary Water presence sensor is used. The Water presence generates an **active low** output voltage for an alert. The GPIO pins of nRF51822 evaluation board kit is used to detect the output from this sensor and depending upon the values of logic level at the monitoring pin necessary actions are taken accordingly in the alarm framework. The digital proprietary Passive infrared sensor should be connected to the **P0.00** pin of nRF51822 evaluation board kit.

11. Source Code Organization

The source code is organized in the following files.

- main.c - calls alarm service (connectable) function and broadcast function in the main loop.
- connect.c – application program for initializing and advertising the alarm services.
- broadcast.c - application for broadcasting the data from sensors in a connectionless mode.
- ble_temp_alarm_service.c – implements the temperature alarm service and creates the characteristic.
- ble_light_alarm_service.c – implements the light level alarm service and creates the characteristic.
- ble_humidity_alarm_service.c – implements the humidity alarm service and creates the characteristic.
- ble_soil_alarm_service.c – implements the soil moisture alarm service and creates the characteristic.
- ble_pir_alarm_service.c – implements the Passive infrared alarm service and creates the characteristic
- ble_accelerometer_alarm_service.c – implements the accelerometer/movement alarm service and creates the characteristic
- ble_probe_alarm_service.c – implements the probe temperature alarm service and creates the characteristic
- ble_thermop_alarm_service.c – implements the thermopile alarm service and creates the characteristic
- ble_waterl_alarm_service.c – implements the water level alarm service and creates the characteristic
- ble_waterp_alarm_service.c – implements the water presence alarm service and creates the characteristic
- ble_device_mgmt_service.c – implements the device management service and creates the characteristic

- ble_data_log_service.c – implements the data logger service and creates the characteristic
- ble_bas.c - Implements the battery service and crates the characteristic
- ble_dis.c - Implements the device information service and creats the characteristic
- wimoto_sensors.h – Header file consisting all the declarations and definitions for all the sensors used in the project
- wimoto.h - Contains all the defines used for the services for each profiles
- tmp102.c - driver code for the TMP102 temperature sensor
- tmp006.c - driver code for the TMP006 temperature sensor
- isl29023.c - driver code for the ISL29023 light sensor
- mma7660fc.c - driver code for the MMA7660FC accelerometer module
- htu21d.c - driver code for the HTU21D humidity and temperature sensor
- adc_soil_mois.c - driver code for the Soil moisture analog sensor module
- adc_water_level.c - driver code for the Water level analog sensor module
- adc_probe_temp.c - driver code for the Probe temperature analog sensor module
- square_wave.c – Code for generating 1MHz square waveform for soil moisture sensor