WIMOTO – BLE SMART DEVICE
HLD – VER 0.2.0

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, soil moisture) Sentry profile (light, temperature, humidity), Thermo profile (probe (NT) C temperature, thermopile temperature) and Water Profile (water presence, 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.

This release contains both the Climate profile and Grow profile and the driver codes for TMP102, TMP006, ISL29023, HTU21D, MMA7660FC and an Analogue Soil moisture sensor.

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 characteristics as specified by the profiles. During this time an iPhone/ Android device can connect to the device.

Grow profile

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

Climate profile

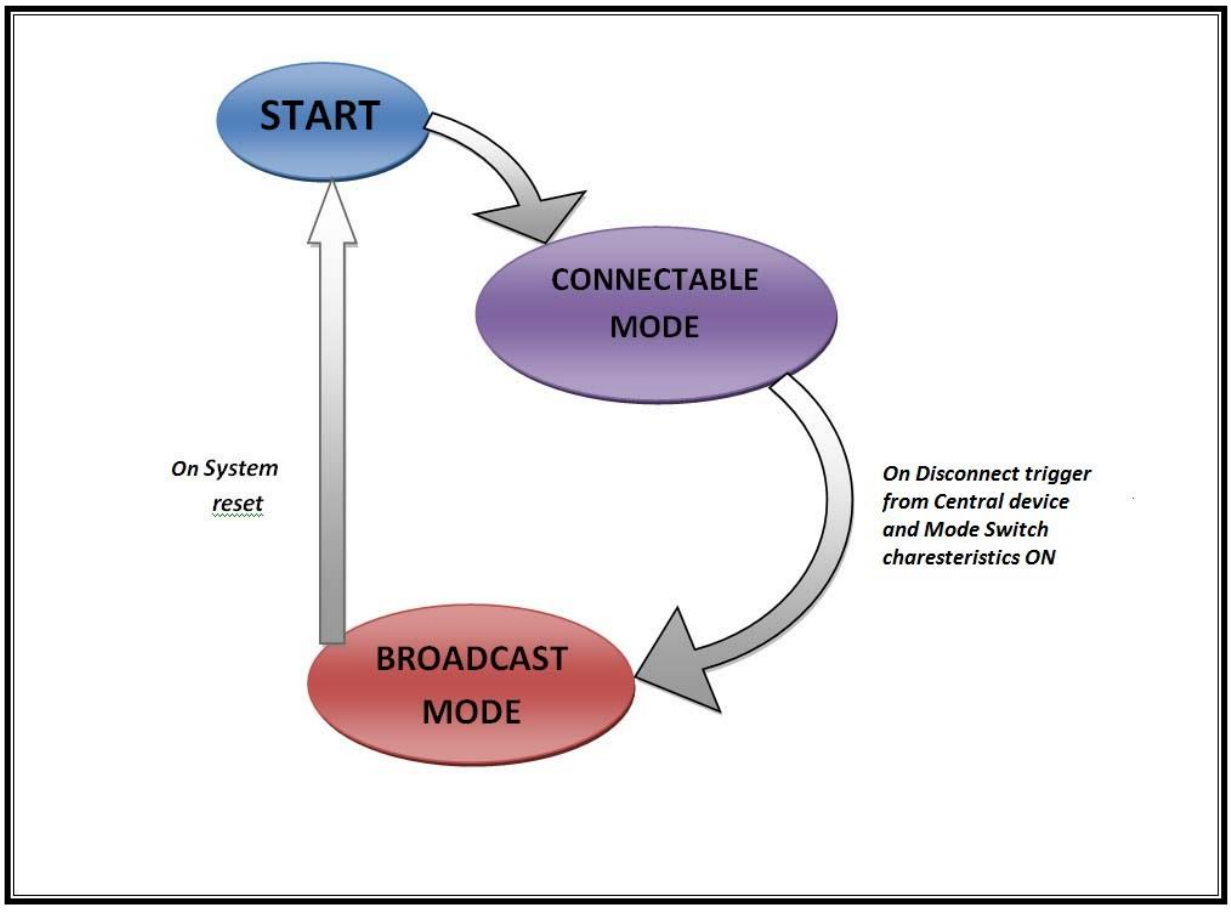
- Soil temperature alarm characteristics
  - Alarm low value
  - Alarm high value
  - Alarm set
- Light alarm characteristics
  - Alarm low value
  - Alarm high value
  - Alarm set
- Soil moisture characteristics
  - Alarm low value
  - Alarm high value
  - Alarm set

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

There is an additional characteristic provided in temperature alarm service, for mode switching, i.e. to switch from Connectable mode to Broadcast mode. Once this mode switch is set and the iPhone/ Android device is disconnected from the Wimoto device and the embedded application enters into Broadcast mode.

Then 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 or to the application flow.

The State Diagram of the control flow is shown below.



3 Generic Profile Architecture

In the profiles the service defines characteristics and functions for exposing the data read from a sensor device and for setting an alarm frame work for alarm enabled sensor output 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 execution start from the main() in main.c. The flag BROADCAST_MODE will be *false* at start up. Then the execution enters the *connectable_mode() in* connect.c. It will advertise the sensor data alarm service. If a central device connects, the characteristics 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 'Mode Switch' characteristics which is provided in the temperature alarm 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 code remains in the Broadcast mode until a system reset or power on reset is incurred.

4 Climate Profile – System Overview

In the climate profile, Light, Temperature and Humidity values that are read from sensor devices are broadcast across the BLE transport. There will be characteristics 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.

4.1 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

4.2 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.

4.2.1 Temperature Alarm Service

The temperature sensor module used for Climate profile is HTU21D. The temperature service and its characteristics 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 returned to the monitoring device like iPhone/Android devices.

While calculations for temperature this original register content is used without right intending the data. In order to convert this 14bit 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***	
connectable_mode	*Definition*	Implements the connectable mode service application.
	Parameter	void
	Return value	void
ble_temps_level_alarm_check	*Definition*	Reads the current temperature from HTU21D , updates the temperature characteristics and checks the alarm condition.
	Parameter1	Pointer to the temperature service structure.
	Return value	Error code
read_temperature	*Definition*	Read temperature by calling the HTU21D API.
	Return value	Current temperature
current_temperature_char_add	*Definition*	Adds the current temperature characteristics to the service.
	Parameter1	Pointer to Temperature Service structure.
temperature_low_level_char_add	*Definition*	Adds the temperature low value characteristics to the service.
	Parameter1	Pointer to Temperature Service structure.
	Parameter2	Pointer to initial value structure
temperature_high_level_char_add	*Definition*	Adds the temperature high value characteristics to the service.
	Parameter1	Pointer to the temperature service structure.
	Parameter2	Pointer to initial value structure
temperature_alarm_set_char_add	*Definition*	Adds the temperature alarm set characteristics to the service.
	Parameter1	Pointer to the temperature service structure.
	Parameter2	Pointer to initial value structure
temperature_alarm_char_add	*Definition*	Adds the temperature alarm set characteristics to the service.
	Parameter1	Pointer to the temperature service structure.
	Parameter2	Pointer to initial value structure
switch_mode_char_add	*Definition*	Adds switch mode characteristic
	Parameter1	Pointer to initial value structure
	Parameter2	Pointer to initial value structure

4.2.2 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 characteristics 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 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
	Parameter1	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 characteristics to the service.
	Parameter1	Pointer to Light Service structure.
light_low_value_char_add	*Definition*	Adds the light level low value characteristics to the service.
	Parameter1	Pointer to Light Service structure.
	Parameter2	Pointer to initial value structure
light_high_value_char_add	*Definition*	Adds the light level high value characteristics to the service.
	Parameter1	Pointer to the Light service structure.
	Parameter2	Pointer to initial value structure
light_alarm_set_char_add	*Definition*	Adds the light level alarm set characteristics to the service.
	Parameter1	Pointer to the Light service structure.
	Parameter2	Pointer to initial value structure
light_alarm_char_add	*Definition*	Adds the level alarm alarm set characteristics to the service.
	Parameter1	Pointer to the Light service structure.
	Parameter2	Pointer to initial value structure

4.2.3 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 characteristics and the functions for 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.

While 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 application and 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
	Parameter1	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 characteristics to the service.
	Parameter1	Pointer to humidity Service structure.
hum_low_value_char_add	*Definition*	Adds the humidity level low value characteristics to the service.
	Parameter1	Pointer to humidity Service structure.
	Parameter2	Pointer to initial value structure
hum_high_value_char_add	*Definition*	Adds the humidity level high value characteristics to the service.
	Parameter1	Pointer to the humidity service structure.
	Parameter2	Pointer to initial value structure
hum_alarm_set_char_add	*Definition*	Adds the humidity level alarm set characteristics to the service.
	Parameter1	Pointer to the humidity service structure.
	Parameter2	Pointer to initial value structure
hum_alarm_char_add	*Definition*	Adds the humidity value alarm alarm set characteristics to the service.
	Parameter1	Pointer to the humidity service structure.
	Parameter2	Pointer to initial value structure

5. 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 characteristics 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.

5.1. 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 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 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 climate profile application is implemented by the function *connectable_mode()* in the file connect.c.

5.2.1 Temperature Alarm Service

The temperature sensor module used for Grow profile is TMP102. The temperature service and its characteristics 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

While 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***	
connectable_mode	*Definition*	Implements the connectable mode service application.
	Parameter	void
	Return value	void
ble_temps_level_alarm_check	*Definition*	Reads the current temperature from TMP102, updates the temperature characteristics and checks the alarm condition.
	Parameter1	Pointer to the temperature service structure.
	Return value	Error code
read_temperature	*Definition*	Read temperature by calling the TMP102 API.
	Return value	Current temperature
current_temperature_char_add	*Definition*	Adds the current temperature characteristics to the service.
	Parameter1	Pointer to Temperature Service structure.
temperature_low_level_char_add	*Definition*	Adds the temperature low value characteristics to the service.
	Parameter1	Pointer to Temperature Service structure.
	Parameter2	Pointer to initial value structure
temperature_high_level_char_add	*Definition*	Adds the temperature high value characteristics to the service.
	Parameter1	Pointer to the temperature service structure.
	Parameter2	Pointer to initial value structure
temperature_alarm_set_char_add	*Definition*	Adds the temperature alarm set characteristics to the service.
	Parameter1	Pointer to the temperature service structure.
	Parameter2	Pointer to initial value structure
temperature_alarm_char_add	*Definition*	Adds the temperature alarm set characteristics to the service.
	Parameter1	Pointer to the temperature service structure.
	Parameter2	Pointer to initial value structure
switch_mode_char_add	*Definition*	Adds switch mode characteristic
	Parameter1	Pointer to initial value structure
	Parameter2	Pointer to initial value structure

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 alarm service application is implemented by the function *connectable_mode()* in the file connect.c. The light intensity service and its characteristics 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
	Parameter1	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 characteristics to the service.
	Parameter1	Pointer to Light Service structure.
light_low_value_char_add	*Definition*	Adds the light level low value characteristics to the service.
	Parameter1	Pointer to Light Service structure.
	Parameter2	Pointer to initial value structure
light_high_value_char_add	*Definition*	Adds the light level high value characteristics to the service.
	Parameter1	Pointer to the Light service structure.
	Parameter2	Pointer to initial value structure
light_alarm_set_char_add	*Definition*	Adds the light level alarm set characteristics to the service.
	Parameter1	Pointer to the Light service structure.
	Parameter2	Pointer to initial value structure
light_alarm_char_add	*Definition*	Adds the level alarm set characteristics to the service.
	Parameter1	Pointer to the Light service structure.
	Parameter2	Pointer to initial value structure

5.2.3 Soil Moisture Service

This application implements the functionality for reading soil moisture 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 soil moisture level alarm service application is implemented by the function *connectable_mode()* in the file connect.c. The soil moisture service and its characteristics 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 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 light level and checks for alarm condition
	Parameter1	Pointer to the Light 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 characteristics to the service.
	Parameter1	Pointer to Soil moisture Service structure.
soil_mois_low_value_char_add	*Definition*	Adds the soil moisture level low value characteristics to the service.
	Parameter1	Pointer to soil moisture Service structure.
	Parameter2	Pointer to initial value structure
soil_mois_high_value_char_add	*Definition*	Adds the soil moisture level high value characteristics to the service.
	Parameter1	Pointer to the soil moisture service structure.
	Parameter2	Pointer to initial value structure
soil_mois_alarm_set_char_add	*Definition*	Adds the soil moisture level alarm set characteristics to the service.
	Parameter1	Pointer to the soil moisture service structure.
	Parameter2	Pointer to initial value structure
soil_mois_alarm_char_add	*Definition*	Adds the soil moisture level alarm set characteristics to the service.
	Parameter1	Pointer to the soil moisture service structure.
	Parameter2	Pointer to initial value structure

6. 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
MMA7660FC	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.

6.1 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 four functions which act as the API in the driver section of the TMP102 file. They are given below

Function	***Details***	
read_register_content (uint8_t)	*Definition*	Read data register content (16 Bits)
	Parameter	Base address of the register to read
	Return value	Content of the register read
write_to_register (uint8_t,uint8_t,uint8_t)	*Definition*	Write data to register (16 Bits)
	Parameter1	Base address of the register to write to
	Parameter2	Most Significant Byte (8 Bits)
	Parameter3	Least Significant Byte (8 Bits)
	Return value	Boolean value(Success – true, Failure - false)
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)

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).

.

6.2 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. Mainly there are five functions for MMA7660FC among which two functions act as public functions and the remaining three as private functions.

Function	**Details**	
Public Functions		
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)
Private Functions		
TMP006_enable_continuous_conversion (void)	*Definition*	Enable continuous conversion mode
	Return value	Boolean value(Success –true, Failure-false)
TMP006_write_to_reg (uint8_t ,uint8_t,uint8_t)	*Definition*	Write 16 bit data to the registers of TMP006
	Parameter1	Base address of the register to write to
	Parameter2	Most Significant Byte (8 Bits)
	Parameter3	Least Significant Byte (8 Bits)
	Return value	Boolean value(Success –true, Failure-false)
TMP006_read_register(uint8_t)	*Definition*	Read the data from the registers of TMP006
	Return value	16 bit data read, if not read returns 0

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.

6.3 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.Mainly there are four functions for ISL29023 among which two function act as public functions and the other two as private functions.

Function	**Details**	
Public Functions		
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)
Private Functions		
ISL29023_read_register (uint8_t)	*Definition*	Read the contents of the registers of ISL29023
	Parameter	Read the contents of the registers of ISL29023
	Return value	8 bit data read , if not read returns 0
ISL29023_write_to_reg (uint8_t , uint8_t)	*Definition*	Write 8 bit data to the registers of ISL29023
	Parameter1	Base address of the register to write to
	Parameter2	8 bit data to write
	Return value	Boolean value (Success –true, Failure-false)

6.4 MMA7660FC Driver

MMA7660 is accelerometer module which outputs orientation of the module with reference to X, Y, Z axis. The orientation values are available in the X-out, Y-out and Z-out registers of MMA7660FC.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 1001100x, where x is the R/W (read/write) bit, so the 8bit data (slave address +R/W bit) for start condition for writing becomes 0x98 and for reading becomes 0x99. Mainly there are six functions for MMA7660FC among which two functions act as public functions and the remaining four as private functions.

Function	**Details**	
Public Functions		
MMA7660_config_standby_and_initialize (void)	*Definition*	Configure MMA7660FC in standby mode and initialize for 1 sample/second and disable tap detection
	Return value	Boolean value (Success –true, Failure-false)
MMA7660_read_xyz_reg_one_time (void)	*Definition*	Read the orientation data from the registers of MMA7660FC
	Return value	Contents of X-out, Y-out and Z-out registers as a 32 bit value)
Private Functions		
MMA7660_enable_active_mode (void)	*Definition*	Enable active mode for continuous conversion
	Return value	Boolean value (Success –true, Failure-false)
MMA7660_enable_standby_mode (void)	*Definition*	Enable standby mode for no conversion
	Return value	Boolean value (Success –true, Failure-false)
MMA7660_read_register (uint8_t)	*Definition*	Read the contents of registers of MMA7660FC
	Parameter	Base address of the register from which to read
	Return value	8 bit data read , if not read returns 0
MMA7660_write_to_reg (uint8_t b,uint8_t)	*Definition*	Write 8 bit data to the registers of ISL29023
	Parameter1	Base address of the register to write to
	Parameter2	8 bit data to write
	Return value	Boolean value(Success –true, Failure-false)

6.5 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. Mainly there are twelve functions for HTU21D among which three functions act as public functions and the remaining nine as private functions.

Function	**Details**	
Public Functions		
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
Private Functions		
HTU21D_CheckCrc(uint8_t, uint8_t, uint8_t)	*Definition*	Checks for CRC error
	Parameter1	16 bit data as an array of 8 bits
	Parameter2	Number of bytes
	Parameter3	Checksum value got along with data
	Return value	Boolean value (Success –true, Failure-false)
HTU21D_MeasureHM (etHTU21MeasureType)	*Definition*	Measure Humidity using Hold Master Mode
	Parameter1	Type of the data to read ,Humidity/Temperature
	Return value	16 bit data
HTU21D_MeasurePOLL (etHTU21MeasureType)	*Definition*	Measure Humidity using No Hold Master Mode
	Parameter1	Type of the data to read ,Humidity/Temperature
	Return value	16 bit data

Private Functions continued….		
HTU21D_PollMasterTransfer (uint8_t)	*Definition*	Function to assist 'HTU21D_MeasurePOLL' function
	Parameter	Measurement Command
	Return value	16 bit Humidity/Temperature data
HTU21D_ReadMeasurementValue(uint8_t))	*Definition*	Function to assist 'HTU21D_MeasureHM' function
	Parameter	Measurement Command
	Return value	16 bit Humidity/Temperature data
f32CalcTemperatureC(uint16_t)	*Definition*	Calculates Temperature in Degree Celsius
	Parameter	16 bit temperature value
	Return value	Temperature in Degree Celsius
f32CalcRH(uint16_t)	*Definition*	Calculates relative humidity value.
	Parameter	16 bit humidity value
	Return value	Relative humidity value

6.6 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 off nRF51822 evaluation board kit.
The functions for Soil moisture sensor are follows

Function	***Details***	
adc_init	*Definition*	Initializes ADC for Soil moisture measurement
	Return value	
do_soil_moisture_measurement	*Definition*	Function to read soil moisture value from the ADC after conversion
	Return value	8 bit data

7.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 characteristics.
- ble_light_alarm_service.c – implements the light level alarm service and creates the characteristics.
- ble_humidity_alarm_service.c – implements the humidity alarm service and creates the characteristics.
- ble_soil_alarm_service.c – implements the soil moisture alarm service and creates the characteristics.
- 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