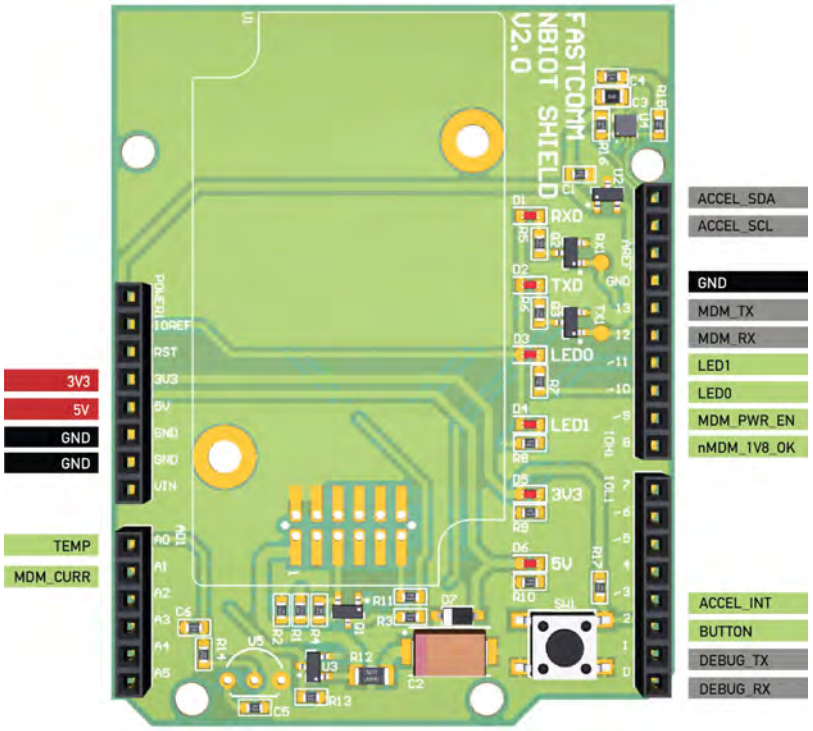
fastcomm
— Linking people places things —

NB-IoT Development kit User Guide

Board pinout diagram



Pin number	Pin name	Description
A0	TEMP	The analog output of the temperature sensor.
SDA	ACCEL_SDA	I2C Serial Data for the accelerometer.
SCL	ACCEL_SCL	
13	MDM_TX	These pins function as UART communication for the BG96 module through SoftwareSerial.
12	MDM_RX	
11	LED1	Active LOW user LEDs. To switch on the LEDs, the pin's state should be LOW.
10	LED0	
9	MDM_PWR_EN	This pin controls the modem power regulator. Set the pin HIGH to enable power to the modem.
3	ACCEL_INT	Interrupt 1 of the accelerometer.
2	BUTTON	This pin is pulled-up. When the button is pressed, the pin's state is LOW.
1	DEBUG_TX	Debug pins for the shield. Connected to the USB of the Arduino.
0	DEBUG_RX	

Shield components/features

The shield is made up from various components to provide the user with a variety of options.

NB-IoT Module

- The NB-IoT module is built with the Quectel BG96 module. It provides connectivity through NB-IoT with 2G fallback.
- The Arduino communicates with the module with a software serial on pins MDM_TX and MDM_RX.
- For more information on the BG96 refer to the AT command manuals (https://www.quectel.com/product/bg96.htm).

Temperature

- The LMT85LP is a temperature sensor with a linear analogue output voltage that is inversely proportional to the temperature.
- The sensor has a temperature range of -50 to 150.
- The temperature can be calculated with equation 2 on page 10 of the datasheet.
- The temperature sensor is connected to pin TEMP.
- For more information refer to the datasheet (http://www.ti.com/lit/ds/symlink/lmt85.pdf).

Button

- The button has an external pull-up resistor and is connected to pin BUTTON.
- For more information on the Arduino interrupt service refer to the Arduino website (https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/).

Accelerometer

- The LIS2HH12 is a 3-axis accelerometer that communicates on the Arduino's I2C/SPI interface and has one interrupt available.
- The I2C/SPI interface is connected to the ACCEL_SCL and ACCEL_SDA pins and the interrupt is connected to the ACCEL_INT pin.
- For more information on the accelerometer refer to the datasheet (https://www.st.com/content/ccc/resource/technical/document/datasheet/c9/16/f2/79/ca/31/47/89/DM00096789.pdf/files/DM00096789.pdf/jcr:content/translations/en.DM00096789.pdf).
- For more information on the interrupt service refer to the Arduino website

(https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/).
- Use the Adafruit LIS3DH library from the Arduino Library Manager with the accelerometer.

LED

- There are 6 LEDs on the shield.
- 2 red LEDs to indicate that there is 3.3V and 5V power.
- 2 red LEDs on the Rx and Tx lines between the shield and the modem to indicate communication
- 2 red active LOW test LEDs connected between IOREF and pins LED0 and LED1.

Analog reference

- The REF3120AIDBZT is a 2.048V voltage reference connected to the AREF pin of the Arduino to provide a custom analogue reference.
- The reference increases the resolution for the temperature measurements.
- For more information on the analog reference refer to the Arduino website (https://www.arduino.cc/reference/en/language/functions/analog-io/analogreference/).
- For more information about the voltage reference chip refer to the datasheet (http://www.ti.com/lit/ds/symlink/ref3112.pdf).

Library functions

Constructors

NBIoT
Constructor to create a modem instance with the required access technology.

Parameters: Access technology
Access technology options: EDGE; NB_IOT

NBIoT
Constructor to create a modem instance with the required access technology and controllable attributes.

Parameters: Access technology *(Access_technology_t)*
Array of controllable attributes *(attr array)*
Number of controllable attributes *(int)*
Access technology options: EDGE; NB_IOT

Power management

modemPowerUp
Power up the modem.

modemPowerDown
Power down the modem.

modemReset
Reset the modem.

SIM card

getCCID
Get the ICCID of the SIM card.

Return: ICCID *(char *)*

getIMSI
Get the IMSI of the SIM card.

Return: IMSI *(char *)*

Modem

modemInit

Initiate the modem.

Return: success status *(bool)*
This function should be run immediately after the modem is switched on.
The function sets up the communication between the Arduino and the modem by opening a software serial connection to the modem and changing the modem baud rate to 19200. This has to be done because the modem has a default baud rate of 115200 and the maximum reliable baud rate for a software serial connection is 57600.
The function tries the setup three times before returning the success as false.
It is recommended to restart the modem if this function fails.

getIMEI

Get the IMEI of the modem.

Return: IMEI *(char *)*
The IMEI is used as the device identifier and has to be saved in the imei variable.

sendDeviceID

Send the IMEI, IMSI and ICCID to the hellothing platform.

Return: success status *(bool)*
The IMEI, IMSI and ICCID has to be saved in the imei, imsi and iccid variables.

Network

setExtConfig

Set the Extended Configuration Settings of the modem..

Return: success status *(bool)*
Parameters: required frequency band to be scanned
GSM band options: GSM_NO_CHANGE; GSM_850; GSM_900; GSM_1800; GSM_1900; GSM_ANY
NB-IoT band options: LTE_B1; LTE_B2; LTE_B3; LTE_B4; LTE_B5; LTE_B8; LTE_B12; LTE_B13; LTE_B18; LTE_B19; LTE_B20; LTE_B26; LTE_B28; LTE_CATNB1_ANY; LTE_NO_CHANGE

getSignalQuality

Get the current signal quality.

Return: signal quality *(char *)*

getServiceMode

Get the selected access technology.

Return: access technology *(char *)*

sendSignalDetails
Send the current signal quality and selected access technology to the hellothing platform.

Return: success status *(bool)*

getNetworkReg
Get the network registration status.

Return: success status *(bool)*

setNetworkAttach
Set the network attach status.

Return: success status *(bool)*

getNetworkAttach
Get the network attach status.

Return: success status *(bool)*

getAvailableOperators
Get all the available network operators for the selected access technology.

Return: success status *(bool)*
The SoftwareSerial receive buffer (*_SS_MAX_RX_BUFF*) has to be increased to receive the complete message.

getCurrentOperator
Get the selected network operator.

Return: success status *(bool)*

setOperator
Set the required network operator.

Return: success status *(bool)*
Parameters: numeric operator code *(const char *)*

TCP/IP

setAPN
Set the APN for the TCP/IP context.

Return: success status *(bool)*
Parameters: APN string *(const char *)*

setDNS
Set the required DNS server.

Return: success status *(bool)*
Parameters: DNS server IP address *(const char *)*

deactContext
Deactivate the TCP/IP context.

Return: success status *(bool)*
Only required for EDGE connection

actContext
Activate the TCP/IP context.

Return: success status *(bool)*
Only required for EDGE connection

closeConnection
Close the TCP/IP connection.

Return: success status *(bool)*

openConnection
Open the TCP/IP connection.

Return: success status *(bool)*
Parameters: domain name *(const char *)*
port number *(const char *)*

sendData
Send a message via the TCP/IP connection.

Return: success status *(bool)*
Parameters: message string *(char *)*

Sensors/Attributes

getTemp
Get the temperature from the LMT85LP sensor.

Return: temperature reading *(float)*

registerOutputs
Register the controllable attributes on the platform.

Return: success status *(bool)*

Data types

Access_technology_t (typedef enum)
EDGE
NB_IOT

Attribute_type_t (typedef enum)
DIGITAL
DIGITAL_INVERT
ANALOG

attr (struct)
const char *key
int value
Attribute_type_t type
int8_t pin

Basic Program sequence

The diagram below describes a basic program sequence to be followed when working with the hellothing development kit.