|  |
| --- |
| Wyres |
| **Functional specification BLE all-in-one V2** |
| Année 2020 |



|  |
| --- |
| **Brian Wyld**  23/04/2020 |
|  |

# System overview

App smartphone

Figure 1 Principe du système

BLE gatt ‘serial’ profile connection

Host (UART) (W\_BASE)

UART

**Module BLE NRF51822**

Remote BLE device (master)

Host (USB) (W\_BASE)

USB/UART

iBeacon frames

BLE device (scanner)

## Functionalities

* iBeacon type advertising, with 3 different ‘slots’ with independent config:
  + UUID
  + Major/minor
  + Tx power / RSSI@1m indication
  + Interval
* iBeacon specific info including battery level, gpio input (button or other digital value), A/D input
* Text based serial comm command interface available via physical UART, I2C UART emulation (nRF52 only) or BLE GATT ‘serial’ profile.
* BLE slave connect to access command interface
* BLE master connect to remote serial profile -> connect local UART transparently to remote BLE module to access its command interface.
* UART-UART transparent serial connection mode
* Per module secret key, configurable via physical UART interface only, used to authenticate (challenge / response) both master and/or BLE module as “valid” for config or auth session
* Scan on demand for iBeacons for specific period with or without UUID /major / minor filtering

# Connection

The module can have serial type connections from 3 sources:

1/ UART

This connection is always active (\*) for AT command processing except in the case of a remote master connection which has activated ‘UART connection mode’ ie the transparent passage of data to/from the UART. In this case data received from the UART is not interpreted by the module as commands but is sent to the remote master.

2/ remote master / local slave BLE connection

The BLE module will accept remoter master connection requests as a serial emulation. When this connection is active, data to/from the remote end is generated/interpreted by the module’s AT command line function (with the exception of the ‘UART connection mode’ where the data is sent to/from the local UART)

The connection can be broken by the remote end, by the local BLE stack, or via a AT+DISC command received from the UART.

Even in UART connection mode, the “AT+DISC\n\r” command is always checked for by the module (even though it is supposed to echo data transparently between the BLE connection and the UART), and will break the BLE connection.

3/ local master / remote slave BLE connection

From a AT+CONN command, the BLE module has initiated a BLE serial connection to a remote slave. The local module acts as a transparent pass-thru for data to/from the BLE connection to the UART.

As above, even though passing data transparently, the local module checks for the “AT+DISC\n\r” sequence and disconnects if seen.

The physical UART connection is active for 10s after bootup and then for 10s after the last character received. It’s state then depends on an IO pin (pulled low normally) – if the pin is HIGH the UART is enabled, if LOW then it will become disabled (until either reboot, or the pin transitions HIGH)

# Authentication

2 distinct authentication actions are possible:

### 1/ authentication to change config

Before any AT command that alters the configuration can be accepted, the remote system must authenticate its right to do so by proving that it knows the module specific secret key.

This is done by the command AT+AUTH. The response will be AT+CH=<X digit hex string>. The remote system is expected to encrypt this byte sequence using the secret key, and send it back using AT+AUTH=<hex string>. If the encrypted block matches the same encrypt as done by the module, subsequent AT commands are accepted. The authentication lapses if the BLE connection is broken, or after 30s since the last command received on the UART.

### 2/ authentication to validate remote operations

In this case, it is the remote system which detects the BLE modules iBeacon frames, and may wish to use its presence to activate a system (eg open a door). The remote system will connect to the serial gatt service of the module (perhaps only when a button or other detection input is active, or when the BLE module indicates a particular value in its beacon frame relating to its physical inputs).

It sends the AT+CH=<challenge hex string> command, and expected a response of AT+AUTH=<response hex string>. The response will be the challenge bytes encrypted using the key. If this matches the local encryption, the BLE module can be accepted as being valid.

# iBeaconning

The iBeacon functionality involves sending BLE advertisement frames in the Apple iBeacon format, every Xms on 3 separate channels.

3 ‘slots’ are defined, each of which can be enabled/disabled and have a specific inter-beacon interval, tx power and parameters for the beacon frame contents.

The contents of the frame are:

UUID : a 16 digit uuid value (configurable per slot)

Major : 2 digit id (configurable per slot)

Minor : 2 digit id (configurable per slot)

RSSI@1m : a 1 byte value indicating the estimated RSSI the receiving device might expect if they were 1m from the module. (configurable per slot)

Physical inputs : a 1 byte value that contains an indicator of physical inputs to the module : battery level (2 bits), GPIO input (1 bit) and an A/D input (5 bits).

The emission of the iBeacons can be globally stop/started via AT command.

## Note:

No iBeacons are emitted when the module is either doing a ‘scan’ (ie listening for other iBeacons!) or ‘connected’ in serial mode to a remote device, or in the initial 10s after boot.

# Scanning

Scanning involves the BLE in receive mode for advertisement packets. The scan is done for a certain period of time, and can then optionally repeat every X seconds subsequently.

At each scan, a list is built up of all ibeacons seen during this period, with duplicates removed. In ‘PUSH’ mode each ibeacon received that has not already been sent, is also immediately sent to the UART connection (which may therefore receive multiple times the same ibeacon within the period).

The list can be retrieved via AT+POLL : if a scan is active then the list from the period scan will be returned.

Each iBeacon has a CSV line of data :

**devAddr** : BLE 3 byte address

**UUID** : 16 byte advert UUID

Major : 2 byte major number

**Minor** : 2 byte minor number

**rssi@1m** : 1 byte from ibeacon signaling the rssi the remote end expected at a distance of 1m

**extrainfo** : battery, button, etc data

**RSSI rxd** : actual RSSI received.

During scanning, the emission of any iBeacons that are configured will be paused.

If a BLE connection is active and a scan should be performed, the scan will be cancelled this time.

### Note : scan time

The length of the ibeacon scan period should be greater than 3x the ibeacon emission interval of the ibeacons that you are trying to detect for reliable results.

# Physical configuration

## UART

Configurable via the UART interface (only)

liaison UART :

* BaudRate 115200
* Data 8 bytes
* Parity None
* No RTS/CTS
* End of command with CR+LF

## I2C

TBD i2c address, registers for UART like data transfer

### Inputs

GPIO pin U is configured to control the enable/disabling of the UART connection.

GPIO pin X is configured as a momentary button input. It is debounced in the firmware, and its status is indicated in the iBeacon frame or via an AT command.

GPIO pins Y-Z are configured as digital inputs and can be read via AT command

GPIO pin A is configured as a A/D measurement. Its 5 MSB are in the iBeacon frame, and its full value can be read via an AT command

Battery voltage is read on GPIO input X. Its value is expressed in the iBeacon frame as 11=100%, 10=80%, 01=20%, 00=critical. The actual voltage can be read via AT Command

### LEDs

The module assumes it has a user visible LED connected to its GPIO pin X. This LED is used as follows:

ON 5s : reset

Fast Flash (10 Hz, 10ms on): beaconing and button pressed

Slow Flash (2Hz, 50ms on): connected to remote BLE device

ON 1s : responded to a authentication challenge

### Outputs

GPIO pins B-C are configured as outputs and can be set via AT command to high or low.

GPIO pin D is configured as PWM output, an AT command allows it to generate a frequency for a given period (eg for a buzzer)

# AT Commands

## Commands

Each command should be terminated by CR/NL.

All commands generate a response, by default “OK” unless otherwise shown.

|  |  |  |
| --- | --- | --- |
| **Command** | **Parameters** | **Description** |
| AT+SET=<key>,<value> | key : which configuration value to change (see below for configurable keys)  value : the new value to set | Configure les paramètres du scanner. Note that the AT+AUTH sequence must have been validated before this command will be accepted. |
| AT+GET[=key,key..] | optional list of config keys to get values for | Get all configuration values, or just those in the list |
| AT+IB\_START[=UUID, Major, Minor, extra, interval, txpower] | UUID, major, minor, etc are the params for the specific ibeacon frame. | All parameters are in hex.  If params are present, then used for the ibeacon if present, ignoring the other configured slots. |
| AT+IB\_STOP |  | Halt all ibeaconning (whether due to IB\_START or just default action.) |
| AT+IBEACON=<bitmask> | 1 bit per slot, 1=enable, 0=disable | Enable or disable automatic ibeaconing (per slot) at bootup. |
| AT+START[=<UUID>[,<maj>[,min]]] | uuid : only return ibeacons with this UUID  maj : only return ibeacons with this major  min : only return ibeacons with this minor | Start ibeacon scanning (note : iBeaconning transmission will be paused during scanning)  Received ibeacon info will be returned each one in a formatted line per device:  <devAddr>, <uuid>,<major>,<minor>,<rssi1m>,<extrainfo>  Data is returned immediately on reception in PUSH mode, and in response to AT+POLL in POLL mode.  The scanning activity will respect the configured scan.time and scan.inter values. |
| AT+STOP | none | Stop scanning. |
| AT+POLL | none | Get currently visible iBeacon list (ie list from last complete scan period with duplicates removed) |
|  |  |  |
| AT+AUTH[=<hex>] | hex : N digit hex string | If no parameter is present, this is indicates a request to challenge authentication (via a AT+CH command).  If present, the hex string is the response to previous challenge, being the result of encrypting the nonce with the shared secret key. |
| AT+CH=<hex> | hex : N digit hex string being the challenge nonce to be encrypted. | Challenge the receiver for its ability to authenticate using the shared secret key. The nonce must be between 4 and 16 bytes long, and cannot be all 0 or all F. Next return line is expected to be AT+AUTH=<hex string of the response.< |
| AT+CONN=<remote> | remote : devAddr of remote BLE module | Connect to serial gatt profile of the given remote module, and once connected, transparently transfer data to/from local UART. Only possible on local UART connection.  Returns either OK once setup, or NOK and an error if not possible. |
| AT+DISC | none | Disconnect any BLE connection currently set up. This can be used from the BLE master side of the connection to disconnect, or from the local UART to break a BLE slave connection. It is always interpreted by the module even in “transparent” connection modes. |
| AT+CONNSTATUS | none | Get the current BLE connection status and role (eg “DISC” = disconnected, “CONNM” = connected as master, “CONNS” = connected as slave) |
| AT+TRUART | none | Enter transparent UART mode (normally on the remote slave end) to transfer data to/from the UART to/from the BLE connection. This creates a serial pipe between the local UART and the remote UART… |
|  |  |  |
| AT+BUTTON | none | get button state (1=pressed, 0=not pressed) |
| AT+BATT | none | get battery voltage as decimal value eg 3.2 |
| AT+RIO=<pin> | pin : gpio pin reference | read pin state (1=high, 0=low) |
| AT+WIO=<pin>,<value> | pin : gpio pin reference  value : 0=gnd, 1=Vcc | set pin state |
| AT+RAD |  | read A/D input value as 16 bit decimal |
| AT+PWM=<freq>,<time> | freq : frequency in Hz  time : time to generate for in ms | Use PWM to generate tone of given frequency for given time. Time=0 means generate until new command received. Freq=0 cancels current generation immediately. |
|  |  |  |
| AT+VERSION | none | Renvoie la version de firmware présente dans le scanner. |
| AT+WHOIS |  | Get type of daughter card |
| AT+RESET |  | Reboot the module |
|  |  |  |
| AT+SETKEY=<key> | key : shared secret key used for authentication for this module expressed as 16 hex digits. | Set the shared secret key. This should be a 8 byte randomly generated value, never all 00 or all FF.  This command is only accepted on the physical UART interface, not on the BLE serial profile.  NO AUTHENTICATION IS REQUIRED TO SET THIS KEY. |

## Configurable keys

|  |  |  |
| --- | --- | --- |
| **Key** | **Use** | **Values** |
| ib.<s>.uuid | iBeacon tx UUID for slot <s> | 16 bytes as 32 hex digits |
| ib.<s>.major | iBeacon tx major for slot <s> | 4 hex digits giving the major |
| ib.<s>.minor | iBeacon tx minor for slot <s> | 4 hex digits giving the minor |
| ib.<s>.txpower | iBeacon tx power level for slot <s> | -40, -20, -10, -5, 0, 4 |
| ib.<s>.rssi1m | iBeacon value to put in the rssi@1m field for slot <s> | -100 to 0. |
| scan.time | Time to scan (rx) for ibeacons each time in ms | 100-10000ms |
| scan.inter | Interval time between scanning when active, in ms | 0 – disable continuous scanning ie auto stop after each scan action  1-60000ms – time between scans |
| scan.uuid | Set UUID to filter for | empty string – no UUID filter  32 hex digits – UUID to look for |
| scan.major | set major number to filter for | empty string – no filter  4 hex digits –number to filter for |
| scan.minor | set minor number to filter for | empty string – no filter  4 hex digits –number to filter for |
| scan.mode | set the way in which scan results are returned. | ‘POLL’ - remote must request ‘seen’ list explicitly  ‘PUSH’ - each seen ibeacon’s data is transmitted immediately on reception |
| conn.esttimeout | timeout in ms for connection establishment in master mode | 100-1000ms |
| conn.idletimeout | timeout in ms to disconnect if no activity on a BLE connection | 500-60000ms |

Note that all configuration is stored in Flash for persistence over reboots.

Exemple d’utilisation des commandes :

AT+CONFIG,0200,0200,0001,0,0  
AT+START **ou** AT+START,E2C56DB5DFFB48D2B060D0F5A71096E0  
AT+PUSH / AT+POLL  
AT+STOP

Exemple d’utilisation des commandes :

AT+CONFIG,0200,0200,0001,0,0  
AT+START **ou** AT+START,E2C56DB5DFFB48D2B060D0F5A71096E0  
AT+PUSH / AT+POLL  
AT+STOP

Equally the current status for scanning and ibeaconning function is stored, and the device will continue after reboot in the same operation modes ie if iBeaconning was enabled the device will continue to do so automatically.