KST3320 Firmware Specification

Revision E - Last Updated December 19th, 2019

Table Of Contents

Introduction

The purpose of this document is to detail the firmware architecture and operation of the KST3320. This document will be divided into three major sections: Architecture, GATT Profile and GAP Profile.

Document Revision History

Revision	Date	Author	Changes
A	May 14, 2019	CW	Initial Document, copy of 3300
B	May 29, 2019	HS	Updated LoRaWAN Uplink Messages
C	July 5, 2019	HS	Added LPP Data Types to LoRa packet definitions
D	November 18th, 2019	CW	Added downlink definition to document.
E	December 19th, 2019	CW	Updated join process with backoff schedule and flowchart

Overview

The KST3320 is an LPWAN-enabled Distance sensor. The firmware is built using the UnitySDK library to maintain a consistent interface to embedded and mobile developers. The device uses LoRa as the LP-WAN mechanism and a distance sensor.

The firmware is built against the Nordic nRF5 SDK 14.1.0 and uses the S132 SoftDevice 5.0.0. This is determined by UnitySDK.

Architecture

Operational Description

At a high-level, the KST3320 is a BLE and LPWAN enabled distance sensor that takes measurements at a specified interval and communicates that information over LoRa. The device also supports downlink messages via LoRa so that configuration parameters can be sent to the device. The device advertises BLE packets to support configuration as well as provide an interface to the on-board sensors. The BLE packets only contain enough information to identify the device and does not contain any sensor data. This conserves power by only providing sensor data if a user is connected and streaming. When not connected, the sensor can be powered down and battery life extended.

State Machine

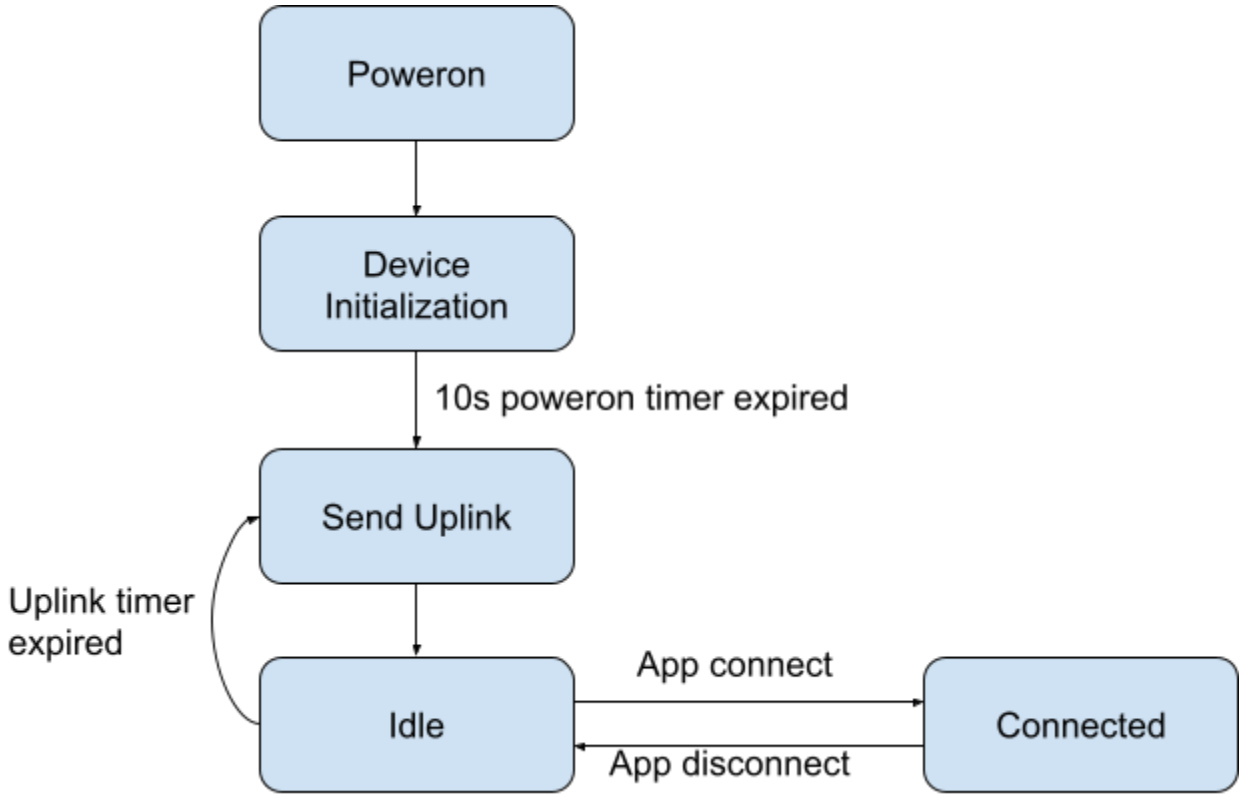


Figure 1. KST3320 Primary State Machine

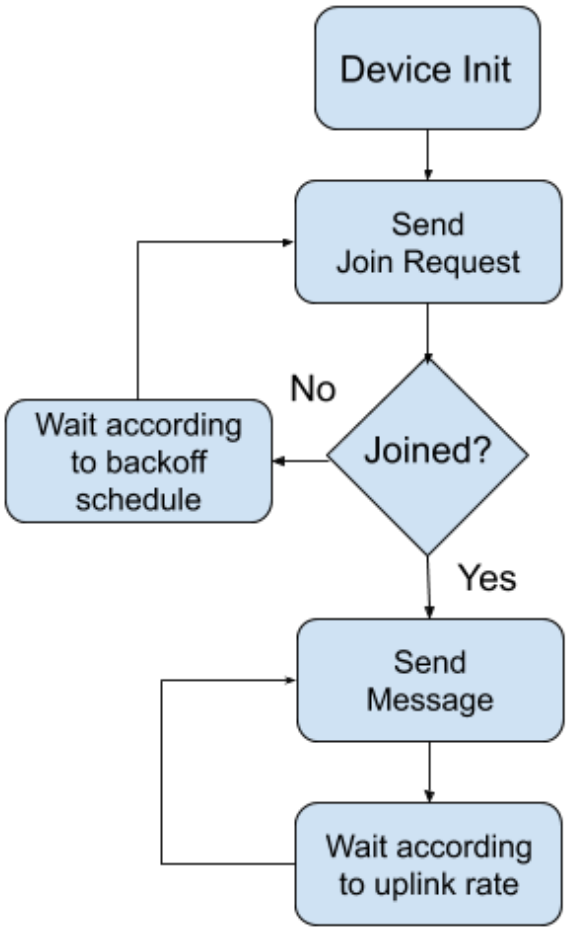
The primary state machine can be seen in Figure 1. When power is first applied, the device initializes the memory, sensors, and the BLE interface. From there, the device waits 10 seconds before sending an initial uplink message. Since the UART operations are asynchronous, the poweron delay ensures enough time has elapsed for any initial UART operations to complete. The initial push has downlinks enabled so that configuration information can be updated as

soon as the device is ready. Once the uplink push is complete, the device goes into the Idle state. The device transitions from the Idle state to the Send Uplink state when the uplink timer expires. The uplink interval is configured over BLE or Sigfox. There is a daily limit for uplink messages, so it is up to the end-user to pick values that best fit their application.

If a mobile device with an appropriate app connects to the device over BLE, the device transitions to the Connected state. In the connected state, the uplink timer does not run to prevent UART-BLE concurrency issues. Once the app disconnects, the device transitions back to the Idle state and repeats the process of waiting for the uplink interval to elapse.

Join Process

The LoRaWan standard has requirements for the maximum amount of time an end device can spend transmitting. Speed of join and compliance with the transmit time requirement are balanced using a backoff schedule. The first join requests happen quickly, and then slow down as the join requests go unanswered. The diagram below shows the join process and backoff schedule.



Backoff Schedule	
Join Attempt	Delay
1	15 seconds
2	30 seconds
3	1 minute
4	5 minutes
5	30 minutes
>5	60 minutes

Uplink Messages

Uplink messages are used to communicate device information to the cloud. The device uses LPP packet structure to communicate. The LPP packet structure requires a different packet for each different type of uplink. There are currently 4 different packet types supported by the KST3320. Each packet is for a different sensor on the device. The frequency of the uplinks is user configurable. An uplink can also be triggered by an event such as a sudden change in accelerometer orientation.

IPSO Packet ID	LPP[1] Type	Packet Type	Data
3330	0x82	Distance	Length: 4 Byte 1: LoRa Channel Byte 2: LPP Data Type Byte 3-4: Distance Data (mm) Example: 01820036 01: LoRa Channel 82: Distance Data Type 000036 -> 54mm Distance
3320	0x78	Battery	Length: 3 Byte 1: LoRa Channel Byte 2: LPP Data Type Byte 3: Battery Data Example: 01780063 01: LoRa Channel 78: Battery Data Type 0063 -> 99% Battery
3313	0x71	Accelerometer	Length: 14 Byte 1: LoRa Channel Byte 2: LPP Data Type Byte 3-6: X-Axis Data (Float) Byte 7-10: Y-Axis Data (Float) Byte 11-14: Z-Axis Data (Float)
3303	0x67	Temperature	Length: 4

[1] https://github.com/myDevicesIoT/cayenne-docs/blob/master/docs/LORA.md

			Byte 1: LoRa Channel Byte 2: LPP Data Type Byte 3: MSB Temperature (0.1C/bit) Byte 4: LSB Temperature (0.1C/bit)
3304	0x68	Humidity	Length 3: Byte 1: LoRa Channel Byte 2: LPP Data Type Byte 3: Percent Humidity (0.5%/bit)
3315	0x73	Barometric Pressure	Length 2: Byte 1: LoRa Channel Byte 2: LPP Data Type Byte 3: MSB Pressure (0.1 hPa/bit) Byte 4: LSB Pressure (0.1 hPa/bit)
3336	0x88	GPS	Length: 11 Byte 1: LoRa Channel Byte 2: LPP Data Type Byte 3-5: Latitude (Float) Byte 6-8: Longitude (Float) Byte 9-11: Altitude (Centimeters) The latitude and longitude are 0.0001 degrees. This results in a ~10m accuracy. Example: 018805F371F006170372EE 01: LoRa Channel 88: GPS Data Type 05F371 -> 39.0001 degrees Latitude F00617 -> -104.7017 degrees Longitude 0372EE -> 2260.30 meters
3336	0x88	GPS (extended)	Length: 20 Byte 1: LoRa Channel Byte 2: LPP Data Type Byte 3-5: Latitude (Float) Byte 6-8: Longitude (Float) Byte 9-11: Altitude (Meters) (Float) Byte 12-15: Horizontal Accuracy (mm) Byte 16-19: Vertical Accuracy (mm) Byte 20: Number of Satellites Example: 018805F371F006170372EE00018D800000FA3604

			01: LoRa Channel 88: GPS Data Type 05F371 -> 39.0001 degrees Latitude F00617 -> -104.7017 degrees Longitude 035f6e -> 2260.30 meters 00018D80 -> 101.760m Horizontal Accuracy 0000FA36 -> 64.054m Vertical Accuracy 04 -> 4 Satellites Note: If the GPS is not able to lock position in less than 90 seconds it will push all 0's

Downlink Messages

Downlink messages are sent from the cloud down to the device. The table below shows the currently defined packets for downlink messages. The device can receive a downlink packet after any uplink packet.

Packet ID	Packet Type	Data
0	Config Update	Length: 3 Byte 1: Packet ID (0x00) Byte 2: Uplink interval MSB Byte 3: Uplink interval LSB Example: To set the uplink interval to 10 minutes send the message 0x00, 0x00, 0x0A To set the uplink interval to 270 minutes send the message 0x00, 0x01, 0x0E To set the uplink interval to 60 minutes send the message 0x00, 0x00, 0x3C

GAP Profile

When the KST3320 is not under connection, it will advertise openly and allow connections from any Central. Advertisements will be sent out at a rate of 1285 ms, which is a tradeoff between discoverability and power. The advertisement will include a standard Unity format to allow for seamless integration using the Unity mobile SDK. The Unity packet includes an advertised service as well as other identification information. The raw advertisement data will have the format shown below:

Length	Type	Value
2	0x01 (Flags)	0x06 (LE General Disc | BREDR not supported)
3	0x03 (16-Bit service UUID list complete)	0xEF9A (Unity Service)
12	0x16 (Service data)	Byte 0: 0xEF (Unity Service LSB)) Byte 1: 0x9A (Unity Service MSB) Byte 2: mac[0] Byte 3: mac[1] Byte 4: mac[2] Byte 5: mac[3] Byte 6: mac[4] Byte 7: mac[5] Byte 8: 0x02 (Packet type) Byte 9: 0x01 (Device type) Byte 10: Firmware version

The device will advertise with a Tx Power of -8 dBm. The default Tx power chosen is a trade off between discoverability and power. The Tx power is configurable via the GATT profile.

When the device is not under connection, it will advertise openly and allow connections from any central. Advertisements can be sent out at a rate of 100ms - 1285ms, which is the maximum allowable to conform to Section 3.5 of Apple's "Bluetooth Accessory Design Guidelines for Apple Products" specification. This also conforms with the 1285ms timing requirement required by eddystone on Android.

The Tx power of the device is also adjustable, but keep in mind that high Tx powers will greatly affect battery life, and low Tx powers will make discoverability difficult. At the lowest Tx power of -40dBm, the central will need to be placed immediately next to the device for discoverability as the experience is almost NFC-like in terms of communication distance.

Security

If the device uses the Security service, the device will initially be locked with a public key and will not allow any transaction until the device is unlocked. The central has 60 seconds to unlock the device or else it will be forcibly disconnected. This prevents unauthorized users from holding on to the connection. The unlock process is shown in figure 1. If a central successfully unlocks the device, the device will be relocked when the central disconnects. **NOTE:** The device will stop advertising once a connection is established with a central, even if the central doesn't unlock the device. A new passcode can be sent to the device by writing the new passcode to the lock state characteristic (After it has been encrypted with the current passcode).

Once unlocked, the central can start interacting with the GATT profile. Any service that does not use the "secure" flag in its initialization structure can be modified while the device is locked. **NOTE**: Standard services cannot be locked.

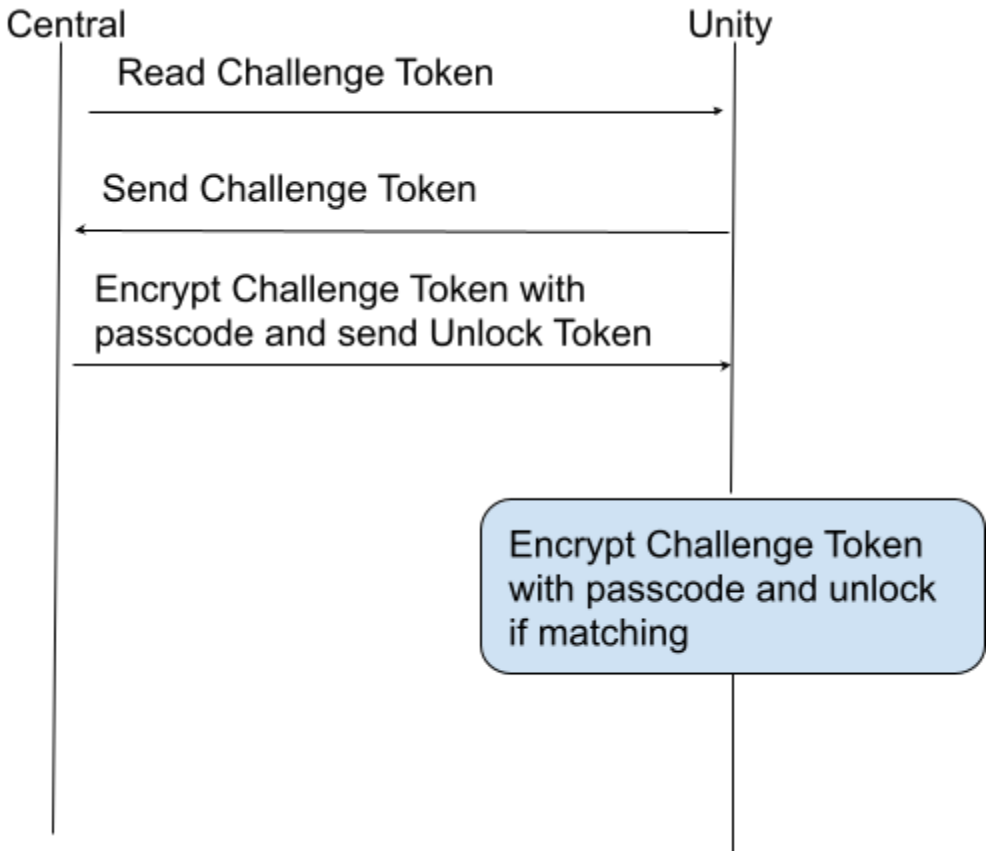


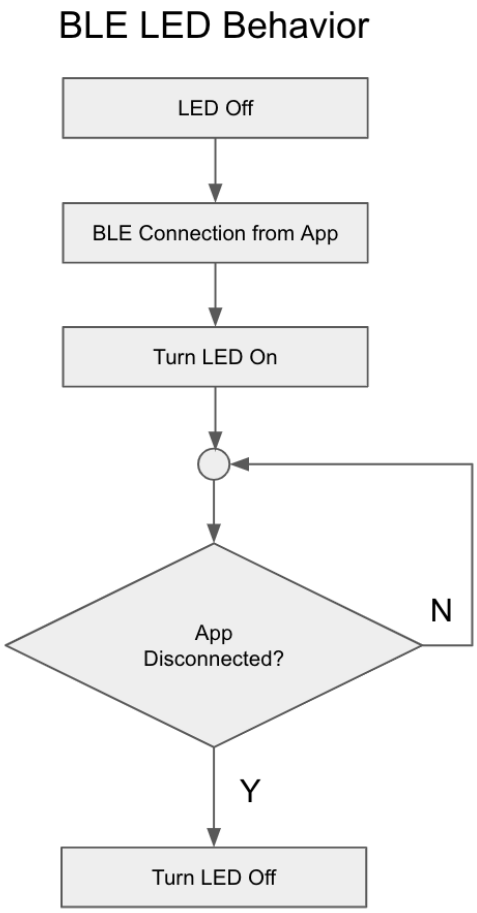
Figure 2. Transaction diagram when unlocking device

GATT Profile

Once connected, the Central Device can enumerate all of the BLE Services and Characteristics that are available for the device. These values, collectively known as the GATT Profile, provide a structured approach to sending and receiving data with the device.

The GATT profile is not fixed and depends on the end application. The best approach for determining if a particular service is available is check for its existence in the GATT table after enumeration.

Finally, the LED indicates connection state as shown below.



Services and Characteristics

When connected, Unity shows all the available services, as shown in the following table. Note that all private services have the UUID Format **`0x9AEFXXXX-12A6-11E8-B642-0ED5F89F718B`** where **`XXXX`** is the value shown in the table.

Table 3: Available Unity Services

UUID	Description	Details
0x1800	Generic Access	Standard BLE 4.0 Service
0x1801	Generic Attribute	Standard BLE 4.0 Service
0x180A	Device Information Service	Standard BLE 4.0 Service
0x180F	Battery Service	Standard BLE 4.0 Service
0x0100	Private Service	Unity Service
0x0400	Private Service	Distance Service
0x0E00	Private Service	LoRa Service
0x0A00	Private Service	Security Service
0x0F00	Private Service	Location/GPS Service

Each Service supports a set of Characteristics as enumerated in the following tables.

Table 4: Generic Access Service Characteristics

UUID	Description	Type	Data
0x2A00	Device Name	Read, Write	UTF8
0x2A01	Appearance	Read	uint8
0x2A04	Peripheral Preferred Connection Parameters	Read	uint8
0x2AA6	Central Address Resolution	Read	uint8

Table 5: Generic Attribute Service Characteristics

UUID	Description	Type	Data
0x2A00	Service Changed	Indicate	uint8

Table 6: Device Information Service Characteristics

UUID	Description	Type	Data
0x2A29	Manufacturer Name String	Read	UTF8
0x2A24	Model Number String	Read	UTF8
0x2A25	Serial Number String	Read	UTF8

0x2A26	Firmware Revision String	Read	UTF8
0x2A27	Software Revision String	Read	UTF8

Table 7: Unity Service Characteristics

UUID	Description	Type	Data
0x0101	Advertisement Interval	Read/Write	uint8
0x0102	Tx Power	Read/Write	uint8
0x0103	Packet Type	Read/Write	uint8
0x0104	Local Name	Read/Write	UTF8
0x010A	Reset	Read	uint8

Table 8: Distance Service Characteristics

UUID	Description	Type	Data
0x0401	Sample Rate	Read/Write	uint8
0x0402	Distance Data	Read/Notify	uint8
0x0403	Timing Budget	Read/Write	uint8
0x0404	Command	Write/Notify	uint8

Table 9: Lora Service Characteristics

UUID	Description	Type	Data
0x0E01	Uplink Interval	Read/Write	uint8
0x0E02	Downlink Rate	Read/Write	uint8
0x0E03	AppEUI	Read	uint8
0x0E04	DevEUI	Read	uint8
0x0E05	AppKey	Read	uint8
0x0E06	NwkSKey	Read	uint8
0x0E07	AppSKey	Read	uint8
0x0E08	DevAddr	Read	uint8
0x0E09	Network Service Type	Read	uint8
0x0E0A	Network State	Read	uint8
0x0E0B	Network Type	Read	uint8

Table 10: Security Service Characteristics

UUID	Description	Type	Data
0x0A01	Lock State	Read/Write	uint8
0x0A02	Unlock	Read/Write	uint8

Table 11: Accel Service Characteristics

UUID	Description	Type	Data
0x0601	Sample Rate	Read/Write	uint8
0x0602	Accelerometer Scale	Read/Write	uint8
0x0603	Accelerometer Data	Read/Notify	uint8

Table 12: GPS Service Characteristics

UUID	Description	Type	Data
0x0F01	Time To Wait For Fix (s)	Read/Write	uint8
0x0F02	Fix Quality	Read+Write	uint8
0x0F03	LocationData	Read	uint8
0x0F04	LocationSupportData	Read	uint8

BLE Data Transfer

The following tables show how the mobile device can interpret the data it receives either from reading GATT characteristics or receiving notifications from subscribed characteristics. It is important to note that only the Private Services are detailed below. For instructions on how to interpret data received from standard BLE 4.0 Services, please reference the *Services* section of the *GATT Specification* published by the Bluetooth SIG here.

Table 12: Unity Service Details

Characteristic	Property	Byte 0 -N
Advertisement Interval	Read/Write	Length: 0x02 Data: Byte 0: Adv Int MSB Byte 1: Adv Int LSB
Tx Power	Read/Write	Length: 0x01 Supported Values: 0xD8: -40 0xEC: -20 0xF0: -16 0xF4: -12 0xF8: -8 0xFC: -4 0x00: 0 0x03: 3 0x04: 4
Packet Type	Read/Write	Length: 0x01 Sets the packet type based on the ones available per the device type. See Appendix C for more information regarding packet types
Local Name	Read/Write	Sets/gets the local name that the device advertises. **NOTE**: Only applicable for packets that contain a local name
Adv Mode	Read/Write	Length: 0x01 0x00: Dual device Advertisement Mode 0x01: Eddystone Advertisement Mode 0x02: idevice Advertisement Mode
Conf Mode	Read/Write	Length: 0x01 0x00: Always Configurable 0x01: Configurable for 10 minutes after power-on 0x02: Configurable for 10 minutes after button press 0x03: Configurable for 10 minutes after power-on or button press 0x04: Non-configurable during business hours; Always configurable outside business hours
Reset	Write	Length: 0x01 Supported Values: 0x00: Soft Reset 0x01: Factory Reset

Table 13: Distance Service Details

Characteristic	Property	Byte 0 -N
Sample Rate	Read/Write	Length: 0x04 MSB first The sample rate of the distance sensor.
Distance Data	Read/Notify	Length: 0x02 MSB first Distance range is mm. Read for single value, notify for updates based on sample rate
Timing Budget	Read/Write	Length: 0x04 MSB first The time in microseconds that the sensor is allowed to range for. The default for the standard ranging profile is 200,000, High Speed is 30,000, Long Range is 33,000
Command	Write/Notify	Length: 0x14 Byte 0x00: Command Prefix (0x04) Byte 0x01 - 0x02: Command Byte 0x03: Command Length Byte 0x04: Error Source Byte 0x05+: Command Data Command List: PERFORM_OFFSET_CAL 0x0001 PERFORM_XTALK_CAL 0x0002 SET_OFFSET_CAL 0x0003 SET_XTALK_CAL 0x0004 GET_OFFSET_CAL 0x0005 GET_XTALK_CAL 0x0006 SET_TIMING_BUDGET 0x0007 GET_TIMING_BUDGET 0x0008 SET_OFFSET 0x0009 GET_OFFSET 0x000A GET_AMBIENT 0x000B

Table 14: LoRa Service Details

Characteristic	Property	Byte 0 -N
Uplink Interval	Read/Write	Length: 0x04 MSB first The interval, in minutes, that the LoRa modem should send an uplink message
Downlink Rate	Read/Write	Length: 0x04 How often, in number of packets, a downlink should be requested on uplink
AppEUI	Read	Length: 0x10
DevEUI	Read	Length: 0x10
AppKey	Read	Length: 0x20

NwkSKey	Read	Length: 0x20
AppSKey	Read	Length: 0x20
DevAddr	Read	Length: 0x04
Network Service Type	Read	Length: 0x01 0x00: Unlocked 0x01: Private 0x02: MachineQ 0x03: IoT America
Network State	Read	Length: 0x01 0x00: Unknown/Default 0x01: Initialization 0x02: Join Requested 0x03: Joined 0x04: Modem Failed
Auth Type	Read	Length 0x01 0x00: OTAA 0x01: ABP

Table 16: Security Service Details

Characteristic	Property	Byte 0 -N
Lock State	Read	Returns the lock state. State values are 0x00: Locked 0x01: Unlocked 0x02: Unlocked and Automatic Relock Disabled
Lock State	Write	0x00: A single byte of 0x00 written to this characteristic will transition the interface to the LOCKED state without changing the current security key value. 0x00 + key[16]: A single byte of 0x00 followed by a 16 byte encrypted key value written to this characteristic will transition the interface to the LOCKED state and update the security key to the unencrypted value of key[16]. 0x02: A single byte of 0x02 written to this characteristic will disable the automatic relocking capability of the interface To prevent the new lock code from being broadcast in the clear, the client shall AES-128-ECB encrypt the new code with the existing lock code. The device shall perform the decryption with its existing lock code and set that value as the new code.
Unlock	Read	Returns a 128-bit challenge token. This token is for one-time use and cannot be replayed.

Unlock	Write	accepts a 128-bit encrypted value that verifies the client knows the device's lock code. To securely unlock the device, the host must write a one-time use unlock_token into the characteristic. To create the unlock_token, it first reads the randomly generated 16-byte challenge and generates it using AES-128-ECB.encrypt (key=device_lock_code[16], text=challenge[16]). This unlock_token is then written to the device in this characteristic. (Note: as a result the secret is never sent in the clear). The device then repeats this process internally using the challenge and the device_lock_code, performing the AES-128 ECB encrypt function. If the result of this calculation matches the unlock_token written to the characteristic, the device is unlocked.

Table 17: Accel Service Details

Characteristic	Property	Byte 0 -N
Sample Rate	Read/Write	Length: 0x04 MSB first The interval, in milliseconds, that the accelerometer should be sampled at
Accel Scale	Read/Write	Length: 0x01 The full scale range of the sensor. Valid values are: 0x02: +/-2G 0x04: +/-4G 0x08: +/-8G 0x10: +/-16G
Accel Data	Read/Notify	Length: 0x06 Data is in milli-g's of the form <Accel X MSB><Accel X LSB><Accel Y MSB><Accel Y LSB><Accel Z MSB><Accel Z LSB>

Table 18: GPS Service Details

Characteristic	Property	Byte 0 -N
Time To Wait For Fix	Read/Write	Length: 0x04 MSB first The interval, in seconds that the GPS will wait for a fix.
Fix Quality	Read/Write	Length 0x01 0x00 No Fix 0x01 Dead Reckoning 0x02 2D Fix 0x03 3D Fix
Location Data	Read	Length: 0x09 Byte 0x08: Altitude Byte 0 (Decimeters) Byte 0x07: Altitude Byte 1

		Byte 0x06: Altitude Byte 2 Byte 0x05: Longitude Byte 0 (0.00001°) Byte 0x04: Longitude Byte 1 Byte 0x03: Longitude Byte 2 Byte 0x02: Latitude Byte 0 (0.00001°) Byte 0x01: Latitude Byte 1 Byte 0x00: Latitude Byte 2
Support Data	Read	Length: 0x09 Byte 0x08: Number of Satellites in View Byte 0x07: Vertical Accuracy Byte 0 (Millimeters) Byte 0x06: Vertical Accuracy Byte 1 Byte 0x05: Vertical Accuracy Byte 2 Byte 0x04: Vertical Accuracy Byte 3 Byte 0x03: Horizontal Accuracy Byte 0 (Millimeters) Byte 0x02: Horizontal Accuracy Byte 1 Byte 0x01: Horizontal Accuracy Byte 2 Byte 0x00: Horizontal Accuracy Byte 3

BLE transaction diagrams

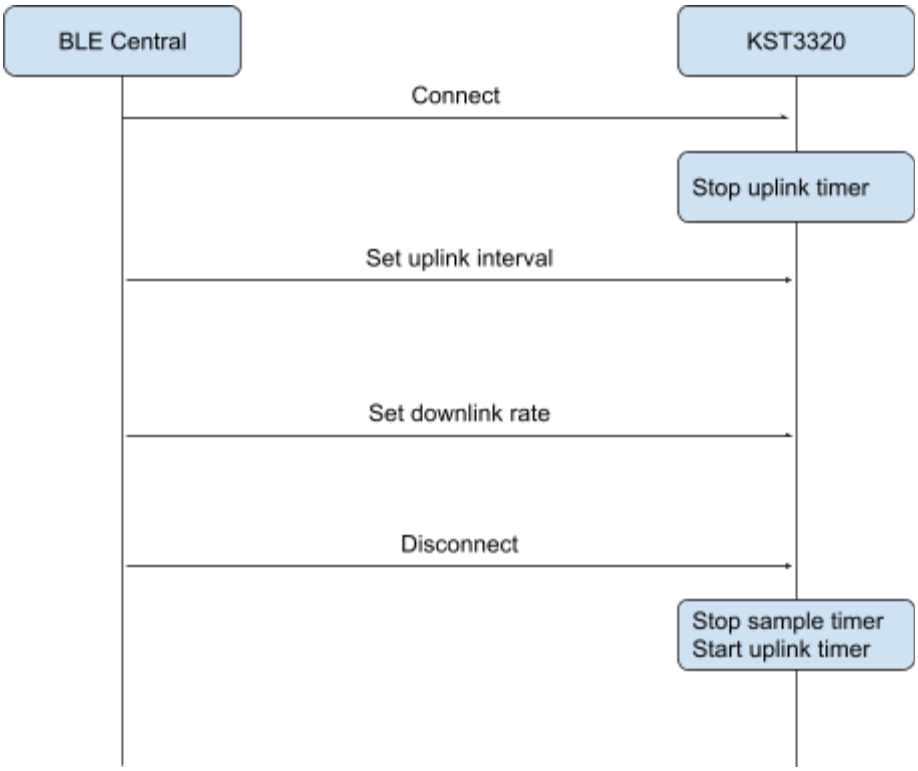


Figure 3: Modifying LoRa parameters

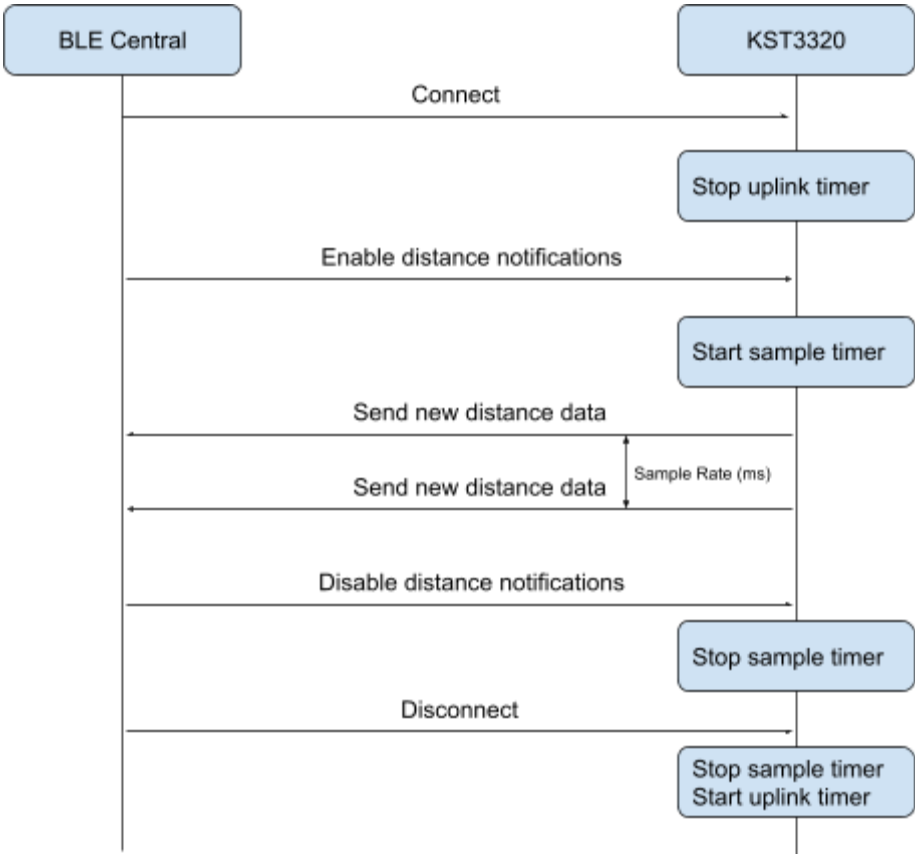


Figure 4: Streaming distance data

Other Considerations

GATT Profile Caching

It is important to note that devices running iOS will cache the GATT profile of the peripheral when it connects. This causes problems when the GATT profile has changed on the peripheral side but iOS is still using the outdated cached version; which can cause erratic behavior such as not being able to read, write, or subscribe to notifications on any new characteristics. The current fix for this is to cycle the BLE radio on the iOS device.

Appendix A: Memory Map

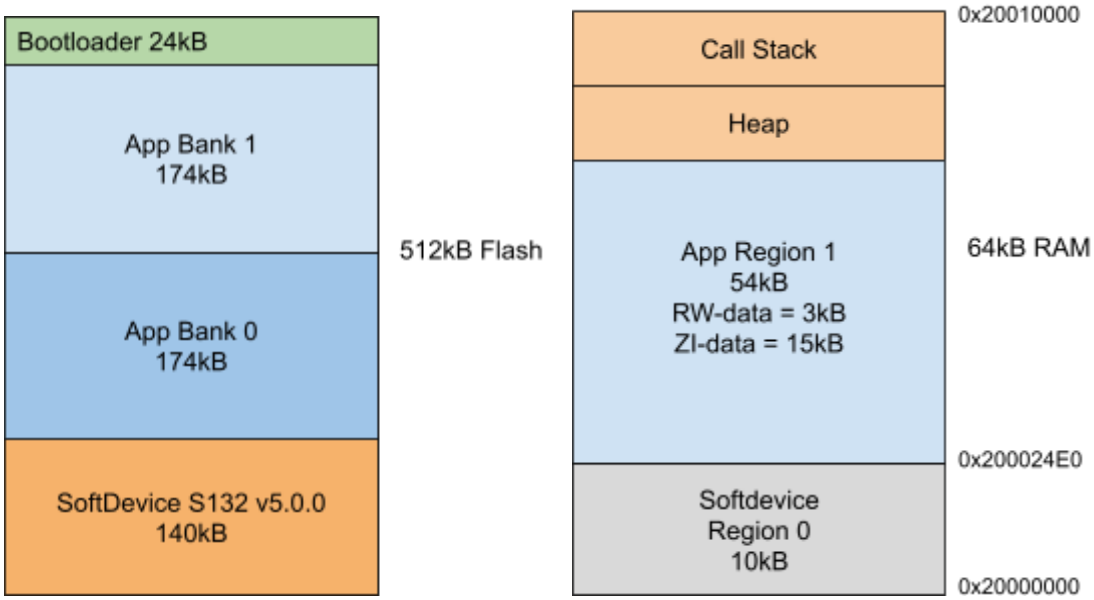


Figure A.1: KST3320 Memory Usage

Memory Usage:

Code	Actual code size	62832
RO-data	Constants placed in flash	2616
RW-data	Data variables that are different from 0	2952
ZI-data	Zero initialized data, data variables set to 0	14848

Flash Usage:
Code + RO-data + RW-data
Total used = 62832 + 2616 + 2952 = 68400

RAM Usage:
ZI-data + RW-data
Total used = 14848 + 2952 = 17800