# **ThaneHunt BLE HID Keyboard — Full Project Documentation**

**Project name:** ThaneHunt\_Project  
 **Target:** Zephyr RTOS on Nordic nRF54L15 (tested with multiple boards)  
 **Role:** BLE HID **keyboard** peripheral with optional IMU (LSM6DSO), power-management, LEDs, GPIO button, Battery Service (BAS), and optional passkey authentication.  
 **Author (from headers):** *Engineer Akbar Shah* **Last edited (from source headers):** 2025-09-14

## **1) High-level overview**

This application is a **Bluetooth Low Energy (BLE) HID keyboard** built on **Zephyr RTOS**. It exposes the standard HID over GATT profile so a phone/PC can pair with it and receive keyboard key-presses. The project is modularized into **components** for BLE, HID, GPIO Button/LED, IMU (LSM6DSO), keycodes, and sleep/idle management. A simple **main.c** wires those modules together.

The build is Zephyr **sysbuild**-enabled for nRF54 dual-core radio IPC setups. Multiple **board overlays** are provided to adapt GPIOs/I²C and peripherals for three boards:

* **xiao/nrf54l15/nrf54l15/cpuapp**
* **nrf54l15dk/nrf54l15/cpuapp**
* **panb511evb/nrf54l15/cpuapp**

Optional features controlled by Kconfig/prj.conf:

* **Passkey authentication** (CONFIG\_ENABLE\_PASS\_KEY\_AUTH, CONFIG\_BT\_FIXED\_PASSKEY).
* **IMU LSM6DSO** driver/logic (CONFIG\_IMU\_LSM6DSO).
* **Idle auto powerdown** (CONFIG\_DEVICE\_IDLE\_TIMEOUT\_SECONDS), BLE disconnect, and deep sleep.

## **2) Repository & folder layout**

ThaneHunt\_Project/

├─ .gitignore

├─ CMakeLists.txt

├─ Kconfig

├─ Kconfig.sysbuild

├─ boards/

│ ├─ nrf54l15dk\_nrf54l15\_cpuapp.overlay

│ ├─ panb511evb\_nrf54l15\_cpuapp.overlay

│ └─ xiao\_nrf54l15\_nrf54l15\_cpuapp.overlay

├─ components/

│ ├─ app\_ble/

│ │ ├─ app\_ble.c

│ │ └─ app\_ble.h

│ ├─ app\_button/

│ │ ├─ app\_button.c

│ │ └─ app\_button.h

│ ├─ app\_hid/

│ │ ├─ app\_hid.c

│ │ └─ app\_hid.h

│ ├─ app\_imu/

│ │ ├─ app\_imu.c

│ │ └─ app\_imu.h

│ ├─ app\_keycodes/

│ │ ├─ app\_keycodes.c

│ │ └─ app\_keycodes.h

│ └─ app\_sleep/

│ ├─ app\_sleep.c

│ └─ app\_sleep.h

├─ create\_component.py

├─ prj.conf

├─ sample.yaml

├─ src/

│ └─ main.c

└─ sysbuild/

└─ ipc\_radio/

└─ prj.conf

**Note:** Many source files include concise doc-comments and some elisions (...) in the uploaded snapshot. Function names, declarations, and behavior are still evident and are documented here.

## **3) Build, flash, and run**

### **3.1 Prerequisites**

* **Zephyr RTOS** environment with west and toolchains installed (matching a revision that supports **nRF54L15**).
* Nordic board support and the appropriate **board definitions** present (the overlays here assume nRF54L15 family).
* Python 3 (for west and helper scripts).

### **3.2 Recommended build commands (sysbuild)**

The project uses **sysbuild** to split the application core and radio core configuration (see sysbuild/ipc\_radio/prj.conf). Typical build:

# From the project root (folder that has CMakeLists.txt)

west build -p always -b xiao/nrf54l15/nrf54l15/cpuapp

# or

west build -p always -b nrf54l15dk/nrf54l15/cpuapp

# or

west build -p always -b panb511evb/nrf54l15/cpuapp

To flash (board-specific):

west flash

### **3.3 Runtime behavior (summary)**

* On boot, BLE is initialized and HID is prepared.
* **Advertising** begins; a **user LED** provides activity feedback.
* A **GPIO button** triggers HID key events (e.g., example sends Space key on a reset action).
* **Battery Service (BAS)** notifications are periodically sent.
* If enabled, **LSM6DSO** IMU is initialized and raw samples can be read/logged; the IMU can be powered down at idle.
* After CONFIG\_DEVICE\_IDLE\_TIMEOUT\_SECONDS of inactivity, the device disconnects and can enter deep sleep.

## **4) Kconfig & configuration (Kconfig, prj.conf, Kconfig.sysbuild)**

### **4.1 Kconfig**

Defines a menu **“ThaneHunt BLE HID KEYBOARD”** with helpful custom options:

* PROJECT\_VERSION (string): Defaults to "0.0.0", can be surfaced in logs/advertising/DIS.
* ENABLE\_PASS\_KEY\_AUTH (bool): Toggles passkey pairing callbacks.
* NFC OOB and Settings defaults (selected for convenience in some stacks).
* Storage backends gated by SoC flash type: ZMS/NVS toggles.

### **4.2 prj.conf (key options)**

# BLE core

CONFIG\_BT=y

CONFIG\_BT\_PERIPHERAL=y

CONFIG\_BT\_DEVICE\_NAME="ThaneHunt\_BLE\_HID\_KEYBOARD"

CONFIG\_BT\_DEVICE\_APPEARANCE=961 # HID Keyboard appearance

CONFIG\_BT\_MAX\_CONN=2

CONFIG\_BT\_MAX\_PAIRED=2

CONFIG\_BT\_SMP=y # security/pairing

CONFIG\_BT\_BAS=y # Battery Service

CONFIG\_BT\_HIDS=y # HID Service

CONFIG\_BT\_HIDS\_MAX\_CLIENT\_COUNT=1

CONFIG\_BT\_HIDS\_DEFAULT\_PERM\_RW\_ENCRYPT=y

CONFIG\_BT\_SMP\_ALLOW\_UNAUTH\_OVERWRITE=y

CONFIG\_BT\_ID\_UNPAIR\_MATCHING\_BONDS=y

CONFIG\_BT\_GATT\_UUID16\_POOL\_SIZE=40

# C library and system

CONFIG\_NEWLIB\_LIBC=y

CONFIG\_NEWLIB\_LIBC\_FLOAT\_PRINTF=y

CONFIG\_HWINFO=y

CONFIG\_POWEROFF=y

# Project feature toggles

CONFIG\_IMU\_LSM6DSO=y

CONFIG\_ENABLE\_PASS\_KEY\_AUTH=y

CONFIG\_PROJECT\_VERSION="1.0.0"

CONFIG\_DEVICE\_IDLE\_TIMEOUT\_SECONDS=30

# Pairing method (fixed passkey for demo)

CONFIG\_BT\_FIXED\_PASSKEY=y

# Sensor ID typo guard (explicitly disabled different part)

CONFIG\_LSM6DS0=n

**Tip:** Consider replacing CONFIG\_BT\_FIXED\_PASSKEY=y with numeric CONFIG\_BT\_PASSKEY=<value> if supported by your Zephyr version, or remove fixed passkey for production.

### **4.3 Kconfig.sysbuild**

source "share/sysbuild/Kconfig"

config NRF\_DEFAULT\_IPC\_RADIO

default y

Enables default **IPC radio** setup for nRF54 splits.

### **4.4 sysbuild/ipc\_radio/prj.conf**

Configures the **radio core** side when using HCI over IPC:

CONFIG\_IPC\_RADIO\_BT=y

CONFIG\_IPC\_RADIO\_BT\_HCI\_IPC=y

CONFIG\_BT\_HCI\_RAW=y

CONFIG\_BT\_MAX\_CONN=2

CONFIG\_MBOX=y

CONFIG\_IPC\_SERVICE=y

# Debug

CONFIG\_ASSERT=y

CONFIG\_DEBUG\_INFO=y

CONFIG\_EXCEPTION\_STACK\_TRACE=y

This enables the **HCI raw** controller with IPC transport and basic debugging.

## **5) Board overlays (boards/\*.overlay)**

### **5.1 Common themes**

* Define an alias lsm6ds0i2c = &i2cXX so the IMU can locate its I²C bus via DEVICE\_DT\_GET(DT\_ALIAS(lsm6ds0i2c))-style bindings.
* Provide a **gpio-keys** button node (button\_0) with a pull-up, active-low configuration.
* Provide an LED node (led\_0) mapped to a GPIO.

**Wake-on-button (sense) config**: Overlays set sense-edge-mask on GPIO ports to enable wake from deep sleep on specific pins.

### **5.2 nrf54l15dk\_nrf54l15\_cpuapp.overlay**

* aliases.lsm6ds0i2c = &i2c20
* buttons/button\_0 on &gpio1 11 (GPIO\_PULL\_UP | GPIO\_ACTIVE\_LOW)
* i2c20\_default pinctrl: SCL = P1.08, SDA = P1.09

### **5.3 xiao\_nrf54l15\_nrf54l15\_cpuapp.overlay**

* aliases.lsm6ds0i2c = &i2c30
* A template LED (led\_0) on &gpio2 0 (active-low).
* i2c30 explicitly clock-frequency = <I2C\_BITRATE\_FAST>
* Disables any auto-probed &lsm6dso node to avoid conflicts (status = "disabled").

### **5.4 panb511evb\_nrf54l15\_cpuapp.overlay**

* aliases.lsm6ds0i2c = &i2c20
* Enables &uart30 and sets chosen nordic,nus-uart = &uart30 (if using NUS/UART in other samples).

**Adjust pins** as needed for your board revision; these overlays are good starting points.

## **6) Build system (CMakeLists.txt, sample.yaml, create\_component.py)**

### **6.1 CMakeLists.txt**

* Finds Zephyr: find\_package(Zephyr REQUIRED HINTS $ENV{ZEPHYR\_BASE})
* Declares project: project(ThaneHunt\_Project)
* Adds application sources:  
  + Core: src/main.c
  + Each component under components/<name>/<name>.c
* Adds include paths for each component:  
  + target\_include\_directories(app PRIVATE ${CMAKE\_CURRENT\_SOURCE\_DIR}/components/<name>)

**Implication:** Components are cleanly isolated; headers live alongside implementation under components/<name>.

### **6.2 sample.yaml**

Defines a Zephyr **sample & CI test**:

sample:

description: ThaneHunt project for BLE HID Keybpoard # (typo: "Keybpoard")

name: ThaneHunt\_Project

tests:

sample.bluetooth.peripheral\_hids\_keyboard.build:

sysbuild: true

build\_only: true

integration\_platforms:

- xiao/nrf54l15/nrf54l15/cpuapp

- nrf54l15dk/nrf54l15/cpuapp

- panb511evb/nrf54l15/cpuapps # (typo: "cpuapps")

platform\_allow:

- xiao/nrf54l15/nrf54l15/cpuapp

- nrf54l15dk/nrf54l15/cpuapp

- panb511evb/nrf54l15/cpuapp

tags: [bluetooth, ci\_build, sysbuild]

**Fixes suggested:** correct “Keybpoard → Keyboard” and “cpuapps → cpuapp”.

### **6.3 create\_component.py**

Helper to scaffold a new component:

* Creates components/<newname>/
* Generates <newname>.c/.h stubs
* **Appends** to CMakeLists.txt to include the new component and its include path

Usage:

python create\_component.py my\_feature

After running, implement your logic in the generated files.

## **7) Source modules — detailed notes**

### **7.1 src/main.c**

**Purpose:** overall startup/orchestration (BLE, HID, GPIO, LED, button thread, IMU, BAS notifications, indicators).  
 **Key patterns and behavior (from comments and references):**

* Initializes logging/subsystems and calls **BLE enable** (enable\_bt() in app\_ble).
* Registers **passkey auth callbacks** when CONFIG\_ENABLE\_PASS\_KEY\_AUTH=y.
* Initializes **HID** (hid\_init()), **GPIO/LED** (init\_user\_led(), init\_user\_buttons()), and **button thread** (button\_thread\_start()).
* If CONFIG\_IMU\_LSM6DSO=y: initialize imu\_lsm6dso\_init(), periodically read raw data via imu\_readDisplay\_raw\_data() and allow **power-down** path (imu\_power\_down flag and lsm6dso\_accel\_gyro\_power\_down() from app\_imu).
* Maintains **advertising LED feedback** while is\_adv (global flag in app\_ble) is true.
* Periodically calls **bas\_notify()** to simulate/notify battery level.

The main loop sleeps ~1000 ms between iterations, keeping the system lightweight when idle.

### **7.2 components/app\_ble — BLE stack, advertising, pairing, BAS**

**Public API (from app\_ble.h):**

extern volatile bool is\_adv;

extern volatile bool isBle\_connected;

void connected(struct bt\_conn \*conn, uint8\_t err);

void disconnected(struct bt\_conn \*conn, uint8\_t reason);

int enable\_bt(void);

void bas\_notify(void);

int ble\_disconnect\_safe(void);

#if (CONFIG\_ENABLE\_PASS\_KEY\_AUTH)

int bt\_register\_auth\_callbacks(void);

#endif

**Responsibilities:**

* Build **advertising data** (flags, UUIDs, complete name) and start/stop advertising.
* Track **connection state** via connected()/disconnected() callbacks, update isBle\_connected, is\_adv.
* **Security** (when enabled): register pairing/auth callbacks, handle passkey display/entry, and pairing results.
* **BAS notifications**: periodically notify battery level over GATT.
* Provide a **safe disconnect** helper used by the idle/sleep module.

**Notes:**

* Device name from prj.conf: "ThaneHunt\_BLE\_HID\_KEYBOARD".
* Appearance code set to 961 (HID keyboard) so hosts show a keyboard icon.
* For production, consider privacy settings, resolvable addresses, and removing fixed passkey.

### **7.3 components/app\_hid — HID over GATT (keyboard)**

**Public API (from app\_hid.h):**

struct keyboard\_state; // internal state struct (opaque to callers)

void hid\_init(void);

int connect\_bt\_hid(struct bt\_conn \*conn);

int disconnect\_bt\_hid(struct bt\_conn \*conn);

int key\_report\_con\_send(const struct keyboard\_state \*state, bool boot\_mode, struct bt\_conn \*conn);

int hid\_buttons\_release(const uint8\_t \*keys, size\_t cnt);

int hid\_buttons\_press(const uint8\_t \*keys, size\_t cnt);

**Responsibilities:**

* Initialize the **HID Service** and **report map** (keyboard usage page 0x07).
* Manage **boot/report protocol mode**, **output report** (e.g., **Caps Lock** LED state) — see caps\_lock\_handler() in source.
* Provide helpers to **press/release** keycodes (arrays of HID\_KEY\_\*).
* Marshal keyboard reports from an internal keyboard\_state to the GATT characteristic via **key\_report\_con\_send()**.

**Integration:**

* Called from **BLE callbacks** on connect/disconnect.
* Consumed by **button** logic to send press/release events.

### **7.4 components/app\_button — Button, LED, and idle activity**

**Public API (from app\_button.h):**

int init\_user\_led(void);

void user\_led\_turn\_on(void);

void user\_led\_turn\_off(void);

void user\_led\_toggle(void);

void button\_thread\_start(void);

void init\_user\_buttons(void);

**Responsibilities:**

* Configure a **user LED** (GPIO) and convenience control.
* Configure a **user button** with **interrupt**, debounce (via **work queue**), and translate events to **HID key reports**:  
  + Example shown: onButton\_reset\_send\_spaceBar() sends HID\_KEY\_SPACE.
* Spawn a **button thread** for processing events outside ISR context.
* Interact with **idle timer** (via app\_sleep) to reset/start timers on activity.
* Optionally toggle an LED to indicate **advertising**/activity (used by main.c).

**Board dependence:** uses pins defined in the overlays (button\_0, led\_0).

### **7.5 components/app\_imu — LSM6DSO IMU (I²C)**

**Public API (from app\_imu.h):**

extern bool imu\_power\_down;

int imu\_lsm6dso\_init(void);

void imu\_readDisplay\_raw\_data(void);

int lsm6dso\_accel\_gyro\_power\_down(void);

**Responsibilities:**

* Acquire I²C device from alias **lsm6ds0i2c** (set by board overlays).
* Low-level **register read/write** helpers to the IMU (private static functions).
* Initialize **accelerometer/gyroscope** (ODR, full scale, filtering) and verify **WHO\_AM\_I**.
* Periodically read **raw LSB** samples and **log** them (intended for bring-up/debug).
* Provide a **power-down** helper to reduce consumption when the system idles.

**Integration:**

* Guarded by CONFIG\_IMU\_LSM6DSO in main.c and app\_sleep.c.
* The **sleep module** calls lsm6dso\_accel\_gyro\_power\_down() before deep sleep.

### **7.6 components/app\_keycodes — HID Usage codes**

**Content:**

* app\_keycodes.h defines **USB HID keyboard** usage codes (Usage Page 0x07), mapping human-readable names to numeric codes — e.g. HID\_KEY\_A = 0x04, HID\_KEY\_SPACE = 0x2C, modifiers, numbers, symbols, etc.
* app\_keycodes.c includes the header and exists as a compilation unit.

**Usage:**

* Include this header in any module that needs to send keyboard keys (button logic, test tasks, etc.).

### **7.7 components/app\_sleep — Idle timer & deep sleep**

**Public API (from app\_sleep.h):**

void start\_idle\_timer(void);

void reset\_idle\_timer(void);

**Responsibilities:**

* Maintain a **k\_work** or **timer** that considers the project’s inactivity period from CONFIG\_DEVICE\_IDLE\_TIMEOUT\_SECONDS.
* When the timer fires:  
  1. Optionally **power down the IMU** (if CONFIG\_IMU\_LSM6DSO).
  2. Perform **safe BLE disconnect** (ble\_disconnect\_safe() from app\_ble).
  3. Transition to **deep sleep/system-off**.
* Provides **APIs** for other modules to start/reset the timer on user activity.

**Logging:** emits informative logs (LOG\_DBG, LOG\_INF, LOG\_WRN) about transitions.

## **8) Data flow & runtime sequence**

1. **Boot** → main.c initializes logging and subsystems.
2. **BLE init** → enable\_bt() sets up advertising data & callbacks; **advertising starts** and is\_adv=true.
3. **HID init** → hid\_init() prepares report map.
4. **GPIO/LED** → LED gets configured (init\_user\_led()), button ISR/work set up (init\_user\_buttons()), and button\_thread\_start() launched.
5. **IMU (optional)** → imu\_lsm6dso\_init() configures LSM6DSO via I²C alias from overlay.
6. **Main loop** → toggles LED/status during advertising, periodically calls bas\_notify(); reads IMU raw data if enabled.
7. **User Input** → button press triggers hid\_buttons\_press()/hid\_buttons\_release() with HID\_KEY\_\* codes.
8. **Idle** → inactivity triggers app\_sleep to power down IMU, disconnect BLE, and enter deep sleep; wake is via GPIO sense/edge on the button.

## **9) Security & pairing**

* **SMP enabled** (CONFIG\_BT\_SMP=y), **HIDS** requires **encrypted** read/write (CONFIG\_BT\_HIDS\_DEFAULT\_PERM\_RW\_ENCRYPT=y).
* **Fixed passkey** is enabled for demo (CONFIG\_BT\_FIXED\_PASSKEY=y) and **CONFIG\_ENABLE\_PASS\_KEY\_AUTH** wires UI callbacks for passkey entry/confirm and pairing result logs.
* **Recommendations for production:**
  + Avoid fixed passkeys; prefer **LE Secure Connections** and numeric comparison or Just Works based on UX.
  + Consider **privacy** (RPA), reduce advertised data, and enable **bonding** with proper erase flow.

## **10) Power considerations**

* **Idle disconnect & deep sleep** after CONFIG\_DEVICE\_IDLE\_TIMEOUT\_SECONDS (default: **30s**).
* **IMU power down** path is included to reduce draw during idle.
* **GPIO sense** configured in overlays to wake on button edge.
* **LED** is active-low on some boards; ensure correct polarity to avoid constant current draw.

## **11) Extending the project**

* Use create\_component.py to scaffold new features (e.g., media keys module, battery gauge, debounced rotary encoder, etc.).
* Add new files under components/<name>, then add usage in src/main.c.
* Update board overlays for additional sensors (provide aliases and pinctrl).
* For **mouse** support, extend HID report map and add motion → report translation.
* For **Device Information Service (DIS)**, add service init and characteristics (model, serial, fw version).

## **12) Troubleshooting & tips**

* **Build fails about board**: ensure your Zephyr install has the **nRF54L15** boards and that you selected one of the three supported targets in this repo.
* **Cannot see advertising**: check that enable\_bt() succeeds; verify BT\_DEVICE\_NAME and **appearance** are set; ensure radio core sysbuild is included.
* **Pairing issues**: with **fixed passkey**, hosts may cache; try removing bonds (CONFIG\_BT\_ID\_UNPAIR\_MATCHING\_BONDS=y allows programmatic unpair).
* **IMU not found**: verify lsm6ds0i2c alias maps to a valid I²C node; check pins and pull-ups on SDA/SCL; ensure &lsm6dso DT node isn’t creating conflicts.
* **Deep sleep never triggers**: confirm CONFIG\_DEVICE\_IDLE\_TIMEOUT\_SECONDS > 0; ensure modules call start\_idle\_timer()/reset\_idle\_timer() appropriately.
* **CI YAML typos**: fix “Keybpoard” and “cpuapps” in sample.yaml.

## **13) Public interfaces — quick reference**

### **13.1 BLE (app\_ble.h)**

extern volatile bool is\_adv;

extern volatile bool isBle\_connected;

int enable\_bt(void);

void bas\_notify(void);

int ble\_disconnect\_safe(void);

void connected(struct bt\_conn \*conn, uint8\_t err);

void disconnected(struct bt\_conn \*conn, uint8\_t reason);

#if CONFIG\_ENABLE\_PASS\_KEY\_AUTH

int bt\_register\_auth\_callbacks(void);

#endif

### **13.2 HID (app\_hid.h)**

struct keyboard\_state;

void hid\_init(void);

int connect\_bt\_hid(struct bt\_conn \*conn);

int disconnect\_bt\_hid(struct bt\_conn \*conn);

int key\_report\_con\_send(const struct keyboard\_state \*state, bool boot\_mode, struct bt\_conn \*conn);

int hid\_buttons\_release(const uint8\_t \*keys, size\_t cnt);

int hid\_buttons\_press(const uint8\_t \*keys, size\_t cnt);

### **13.3 Button/LED (app\_button.h)**

int init\_user\_led(void);

void user\_led\_turn\_on(void);

void user\_led\_turn\_off(void);

void user\_led\_toggle(void);

void button\_thread\_start(void);

void init\_user\_buttons(void);

### **13.4 IMU (app\_imu.h)**

extern bool imu\_power\_down;

int imu\_lsm6dso\_init(void);

void imu\_readDisplay\_raw\_data(void);

int lsm6dso\_accel\_gyro\_power\_down(void);

### **13.5 Sleep (app\_sleep.h)**

void start\_idle\_timer(void);

void reset\_idle\_timer(void);

### **13.6 Keycodes (app\_keycodes.h)**

* Dozens of #define HID\_KEY\_\* macros for letters, digits, symbols, and modifiers.
* Example: HID\_KEY\_A 0x04, HID\_KEY\_SPACE 0x2C, etc.

## **14) Known nits in the uploaded snapshot**

* Several source files include ... elisions where unrelated boilerplate was omitted. This documentation reflects the APIs and comments that are present and infers standard Zephyr/HID flows accordingly.
* sample.yaml contains two typos (spelling and board path). See section **6.2** for proposed fixes.

## **15)attribution**

**Author (from headers):** *Engineer Akbar Shah* **Project version:** "1.0.0" (from prj.conf)

## **16) Appendix — example workflow (pair & send a key)**

1. Power the device; it starts advertising as **ThaneHunt\_BLE\_HID\_KEYBOARD** with HID appearance **961**.
2. On a host (PC/phone), scan and pair. If **passkey** is enabled, complete the passkey step as prompted.
3. Press the **user button**: the app calls hid\_buttons\_press() with HID\_KEY\_SPACE then hid\_buttons\_release(), and the host receives a Space keystroke.
4. After **30 seconds** of inactivity (default), the device **disconnects** and may enter deep sleep; pressing the button wakes it.