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.
