Seminar Ausgewählte Kapitel der Systemsoftware (AKSS)

Bluetooth Low Energy In Intermittently-Powered Wireless Communication Systems - An Introduction

January 24, 2024

Marvin März

Friedrich-Alexander-Universität Erlangen Nürnberg

Chair in Distributed Systems and Operating Systems

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

FACULTY OF ENGINEERING

Table of Contents

Introduction

Introduction

- Bluetooth Low Energy (BLE) used in many IoT sensors & devices
- BLE Real-time requirements
- Enable undisrupted bi-directional communication in intermittent systems
- Battery-free, using capacitors and energy harvesting

Bluetooth Low Energy (BLE)

Bluetooth Low Energy

- Introduced in Bluetooth version 4.0 (2010)
- Low power variant of Bluetooth Classic
- Different protocol stack and hardware
- Bursty and low bandwidth communication
- Able to work on coin-cell batteries

⇒ Incompatible because fundamentally different

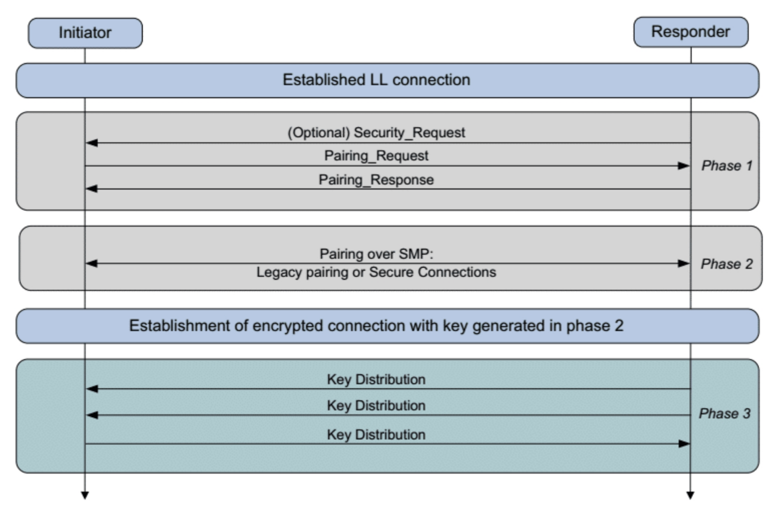
- **But most modern consumer devices support both**

BLE Operation

- BLE operates in the ISM Band (2.4 GHz electromagnetic frequency spectrum)
- Shared with WLAN, medical instruments, microwaves, ...
- Spectrum subdivided into 40 channels
  - 3 advertising channels for pairing
  - 37 data channels for transmissions
  - Adaptive frequency hopping algorithm in data channels to avoid noise & other signals
- Frequency ("*key*") for "0" and "1" in each channel

$\Rightarrow$ **Gaussian Frequency Shift *Keying*** for data transfer

BLE Connection Phases

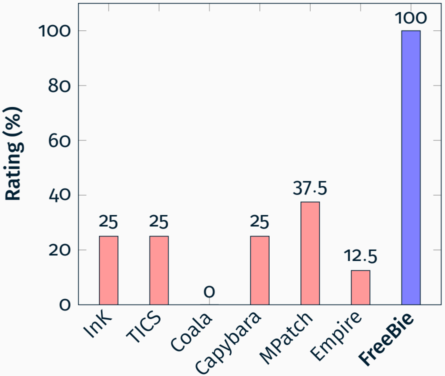


https://www.bluetooth.com/blog/bluetooth-pairing-part-1-pairing-feature-exchange/

Overview of Intermittent Approaches

previous attempts:

- too many undesirable traits
- poor performance, lengthy reconnection (around 40s)
- breaking protocol's timing and real-time requirements



Intermittent Implementation - FreeBie

FreeBie Architecture

A BLE ARM-based microcontroller unit (MCU)

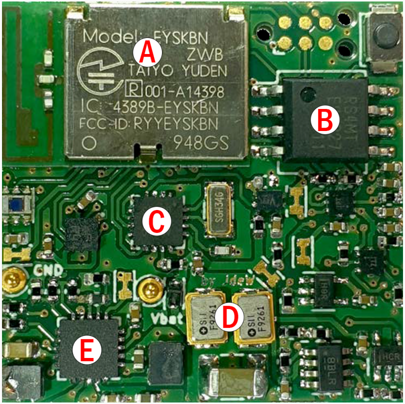
B Non-volatile FRAM

C external Real-Time Clock (RTC)

D Capacitors

E Energy harvester

- solar panels & display on the back



J. de Winkel, H. Tang, and P. Pawełczak. **Intermittently-Powered Bluetooth That Works.** In *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services. MobiSys '22*, pp. 287-301.

Main Changes

1. Time-Deterministic Checkpoint and Real-Time Restoration
2. Virtualisation Layer
3. Dynamic Network Handling

Time-Deterministic Checkpoint and Real-time Restoration

- Checkpoint is a function that is introduced to the original protocol
- Saves system state until the function call
- Triggered by end of wireless protocol process
- Stored in non-volatile FRAM
- Process & memory separation into network, application & OS
- Classify processes as real-time or non-real-time
- Scheduler decides whether to sleep or turn off power to MCU

Turn-off procedure

1. Determine next power-on time T_{wakeUp}
2. Determine real-time processes for restoration
3. Checkpoint processes
4. OS Checkpoint
5. Switch off power domain

Process Restoration

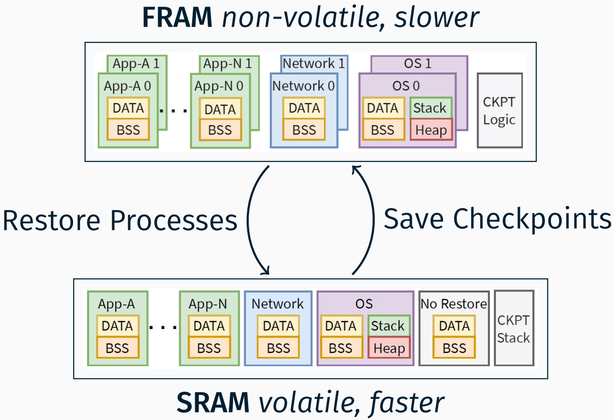
Real-time processes:

- Restored at boot, before scheduler starts
- If checkpoint present:
    - restore OS and network
    - restore real-time application processes
- Synchronise to external RTC
- Compensate for power-off time
- Resume normal operation

Non-real-time processes:

- Restored dynamically prior to execution

Checkpoint, Restoration, Memory Map Overview



Virtualisation of Time and Peripherals

Masking time and intermittent effects to host and applications to support time-deterministic restoration.

- System is divided into processor power domain & "always-on" ultra-low-power domain
- Energy consuming components like processor can be switched off, if next process is scheduled later than T_{minOff} = 20ms
- External RTC belongs to always-on domain

⇒ RTC can be set to switch processor on again at T_{wakeUp}

Calculating T_{wakeUp}

T_{wakeUp}

$$T_{wakeUp} = T_{sync} - T_{startUp} - T_{restore}$$

$T_{startUp}$ = 10 ms (start up duration of system)

$T_{restore}$ = 10 ms

T_{sync} := *known* time of next sync with external RTC needed to reset internal MCU time registers after power failure to correct time (from external RTC) $\Rightarrow$ masking intermittent time effects

Dynamic Network Handling (DNH)

Main areas:

1. Network recovery
2. Dynamic network adaptation

Improves:

- System connectivity
- Performance
- Energy efficiency

Important Connection Parameters

Parameters $\rightarrow$ can be adjusted

Supervision Timeout (ST) [s] $\in [0, 32]$

Time after which a connection is considered lost (i.e. Connection Timeout)

Connection Interval (CI) [s] $\in [1, 4]$

Time between two consecutive data transfers between devices

Slave Latency (SL) [int] $\in [0, 3]$

How many connection events can be skipped by end device

Network Recovery

Needed when system …

- Performs normal restoration (after power failure or power off)
- Was unable to turn back on at T_{wakeUp}

⇒ Possible when still within ST window

⇒ ST set to maximum of 32s

Steps:

1. Skip missed connection events
2. Schedule network process for next connection event
3. Retransmit lost packages (or attempt to)

Dynamic Network Adaptation

Adjusting CI and SL depending on energy levels:

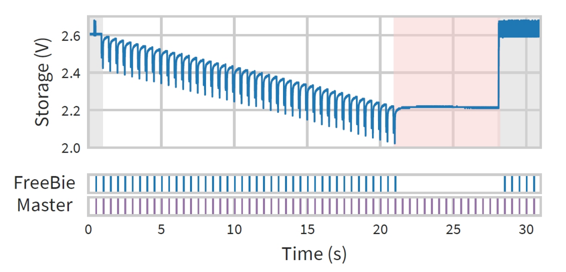
Energy Level (Luminosity)	CI	SL	ST
Low (200-300lx)	4s	3	32s
Medium (600lx)	2s	1	32s
High (10klx)	1s	0	32s

- Low energy: increase CI and SL
  - Lower throughput and bandwidth
  - ⇒ But energy available longer
- High energy: decrease CI and SL
  - ⇒ Increases responsiveness, throughput and overall performance

Findings & Conclusion

Findings & Conclusion (1-2 slides)

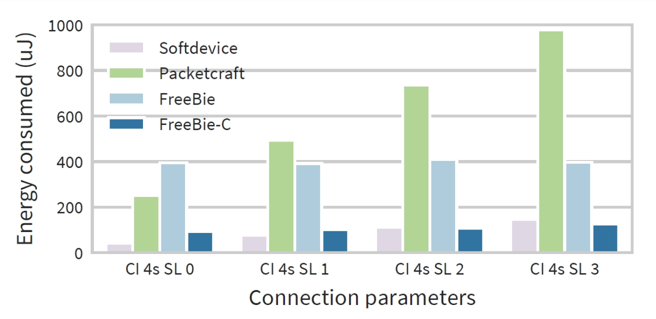
- FreeBie evaluation mostly successful
- Most notably 24h benchmark & connection retention
- In idle up to **9.5 times** less energy consumption than reference device



FreeBie network activity (blue) around power failure (pink) and connection retention

Findings & Conclusion

- **But** significant external memory access overhead



FreeBie-C: memory overhead data removed, showing significant impact

⇒ different MCU with internal non-volatile FRAM/MRAM could be used to eliminate external memory

Thanks for your attention!

Questions?