Periodic Advertising with Responses (PAwR) using AIROC™ CYW20829 Bluetooth® LE SoC

About this document

Scope and purpose

This application note introduces the Periodic Advertising with Response (PAwR) feature of Bluetooth® Core Specification 5.4. It explains the detailed test setup for integrating PAwR network using the AIROC™ CYW20829 Bluetooth® LE SoC. The firmware used in this application is provided in the blog Monitoring Temperature with Bluetooth® Using PAwR.

Intended audience

This application note is intended for those who wants to integrate the PAwR feature on CYW20829.

Table of contents

About this document **1**
Table of contents **2**
1 Introduction **3**
1.1 AIROC™ CYW20829 Bluetooth® LE SoC 3
1.2 Introduction to PAwR 3
2 Different types of advertising **5**
2.1 Legacy advertising 5
2.2 Advertising schemes from Bluetooth® 5.0 5
2.2.1 Extended Advertising 5
2.2.2 Periodic Advertising 6
2.3 Periodic Advertising with Response 6
2.4 PAwR timing diagram 7
2.5 Synchronization 8
2.5.1 Synchronization with ADV_EXT_IND PDUs 8
2.5.2 Periodic Advertising Sync Transfer (PAST) procedure 9
2.6 PAwR vs. Bluetooth® LE Mesh 9
3 PAwR implementation in CYW20829 ..... **11**
3.1 Advertiser 11
3.2 Scanner 14
4 Hardware setup **17**
4.1 CYW20829 evaluation board 17
4.2 PAwR test setup 17
4.3 Display board 18
5 Software and tools **19**
5.1 ModusToolbox™ software 19
5.2 Code examples 19
5.3 PAwR client application 19
5.4 PAwR server application 19
5.5 Build and flash the firmware 20
5.6 Download the project from the blog 20
6 Testing and results **21**
References **22**
Revision history **23**
Disclaimer **24**

1 Introduction

1.1 AIROC™ CYW20829 Bluetooth® LE SoC

AIROC™ CYW20829 is a dual-core Arm® Cortex® M33 device with one M33 core reserved as the Bluetooth® controller, and the other available for the user application. The CPU subsystem provides 256 KB RAM, XIP interface for external flash, and a rich set of peripherals to enable a diverse set of applications. CYW20829 is supported by the ModusToolbox™ development environment, which enables developers to accelerate time-to-market for Bluetooth®-enabled IoT solutions.

CYW20829 incorporates a Bluetooth® 5.4 LE subsystem (BLESS) that contains the physical layer (PHY) and Link Layer (LL) engines with an embedded security engine. The Bluetooth® LE SS supports all Bluetooth® LE 5.4 features including LE 2 Mbps, LE Long Range, LE Advertising Extensions, LE Isochronous Channels, Periodic Advertising with Responses (PAwR), Encrypted Advertising Data, LE GATT Security Levels Characteristic, and Advertising Coding Selection.

CYW20829 supports the new Periodic Advertising with Response (PAwR) feature of Bluetooth® Core Specification 5.4, an extension to the Periodic Advertising Broadcast feature available in Bluetooth® 5.0 specification. It allows synchronized observers (or scanners) to send responses back to a broadcaster (or advertiser), and to organize the former into groups using specific subevents in a PAwR interval. It is the only Bluetooth® LE protocol that allows bidirectional connectionless data communication in a large-scale one-to-many topology.

1.2 Introduction to PAwR

Periodic Advertising with Responses (PAwR) is an extension to the Periodic Advertising Broadcast feature available in Bluetooth® 5.0 specification. The following are some of the differences between the Periodic Advertising with Responses (PAwR) and Periodic Advertising Broadcaster (PADVB).

Table 1 Comparison of PADVB with PAwR

Feature	PADVB	PAwR
Data transmission	Only one-way transmission of data from a Broadcaster to Observers exclusively	PAwR Observers can send response packets back to the Broadcaster, enabling a two-way communication channel
Scheduling	Broadcaster schedules transmissions within advertising events	PAwR Broadcaster schedules transmissions in a series of events and subevents. Observers should synchronize in such a way as to listen during a specific subevent or subevents only.
Frequency of change of application data	Application data change only from time to time	PAwR is designed with the expectation that application data changes frequently.
Unique application data for each observer or sets	Identical application data is sent to all synchronized Observer devices within the same advertising set	Allows sending unique data to individual Observer devices or specific sets of Observer devices
SyncInfo field	SyncInfo field of AUX_ADV_IND PDUs contains synchronization information	Synchronization information is found within both the SyncInfo field and the ACAD field of AUX_ADV_IND PDUs

PAwR is ideally designed for systems that require exchange of messages between a main central hub and numerous other devices within a network. These messages may include instructions, readings from sensors, or

different types of data, as specified by the application layer. Some of the applications of the PAwR are as follows:

- **Electronic Shelf Label (ESL):** ESL is the main application of PAwR where the Electronic Shelf Label (ESL) profile uses PAwR to transport messages containing commands and responses.
- **Industrial applications:** PAwR can be used for temperature/condition monitoring of the devices in the industries. See the blog on Monitoring Temperature with Bluetooth® Using PAwR with the CYW20829 device.
- **Smart home:** PAwR can be used in smart home applications for operating various devices from a central hub. It can read the temperature, humidity from the different devices through PAwR response packets.
- **Health monitoring:** PAWR has the capability to provide more frequent updates and lower power consumption in remote patient monitoring, potentially surpassing traditional Bluetooth® LE and improving the monitoring of vital signs and medication adherence.

2 Different types of advertising

The first specification of Bluetooth® LE 4.0 introduced Legacy advertising, which uses the maximum broadcast data size of 27 bytes and uses only three channels for advertising. Extended Advertising introduced in Bluetooth® LE version 5.0 uses 37 general-purpose channels and the three primary advertising channels for advertising. In Periodic Advertising, the broadcasting device transmits packets at a fixed interval, deterministic schedule, and Observer devices can discover the Broadcaster's transmission schedule and synchronize their scanning to it. Bluetooth® LE 5.4 introduced PAwR, in which bidirectional, connectionless data communication is allowed between the advertiser and the scanner.

2.1 Legacy advertising

Legacy advertising involves transmitting identical copies of advertising packets on three primary advertising channels. These packets are transmitted one at a time in a pseudo-random sequence. The transmission of these advertising packets occurs during advertising events. The timing of these events is primarily controlled by a link layer timing parameter called *advInterval*.

However, to prevent persistent collisions with other advertising devices, the advertising events are intentionally made slightly irregular. This is accomplished by assigning a parameter called *advDelay*, which is a pseudo-random value ranging from 0 to 10 ms. The *advDelay* is then added to the fixed *advInterval*, resulting in perturbed timing for the advertising events.

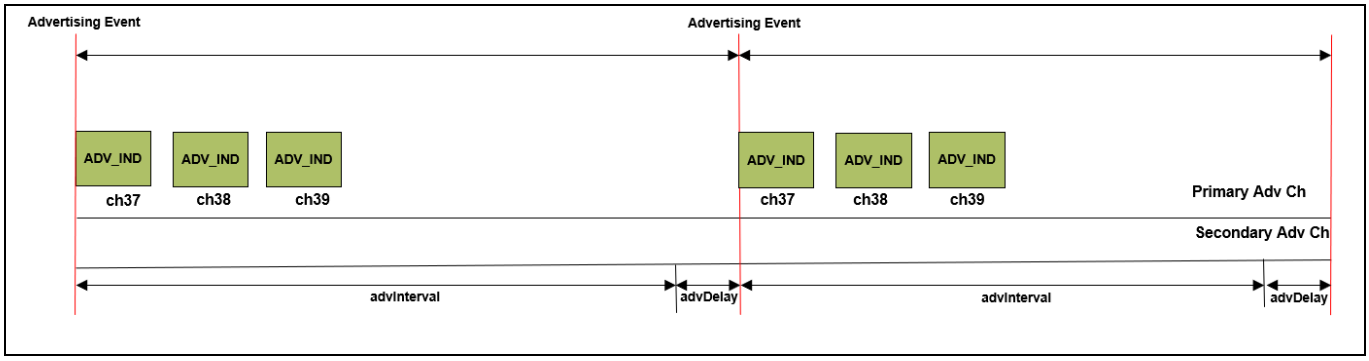

Figure 1 Legacy advertising

2.2 Advertising schemes from Bluetooth® 5.0

Bluetooth® version 5.0 introduced a new feature called "Extended Advertising", which uses both the primary and secondary advertising channels to send the data.

2.2.1 Extended Advertising

Extended Advertising allows to transmit packets of up to 255 bytes. This is achieved by transferring the application data on to secondary channels. Their transmission scheduling is irregular due to the addition of the random *advDelay* value in the range of 0 to 10 ms in calculating the advertising event times.

Extended Advertising differs from legacy advertising in a number of ways, including that distinct PDU types are used. Some of these PDU types are transmitted only on the primary advertising channels but may be linked by a pointer field called *AuxPtr* to others that are transmitted only on the secondary advertising channels. Larger payloads may be handled by fragmenting the data and transmitting it in multiple PDUs linked together or chained using *AuxPtr* in certain PDU types.

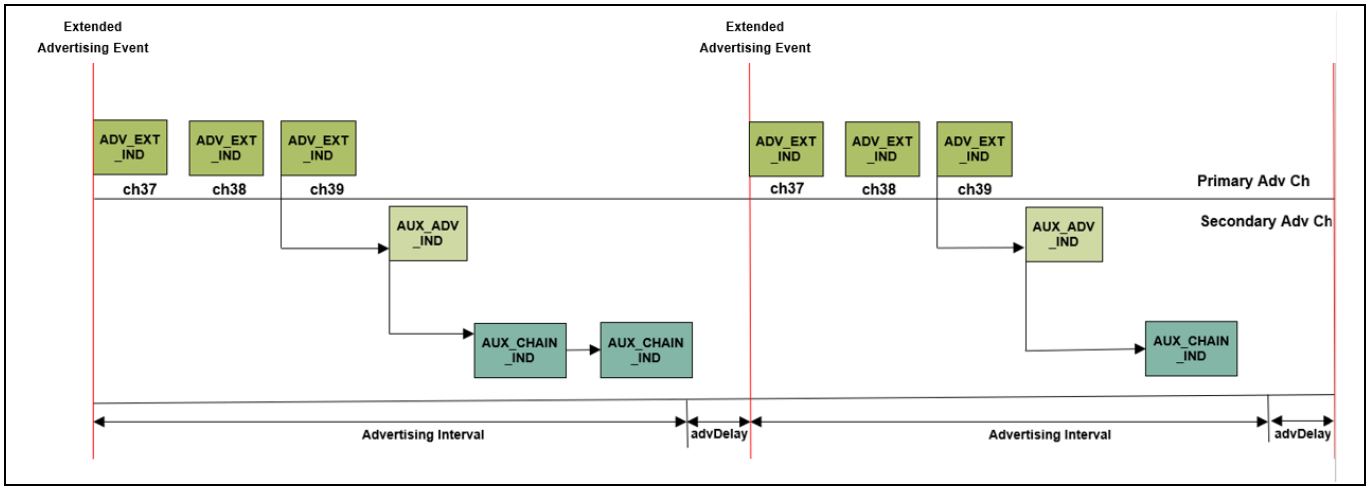

Figure 2 Extended Advertising

2.2.2 Periodic Advertising

Periodic Advertising enables the transmission of advertisements at regular intervals, allowing for changes in advertising data during these intervals. Multiple scanners can then listen for these advertisements, creating a multicast effect. The fixed interval provides a significant advantage over other advertising modes, as scanners can sleep between advertisement events instead of constantly being in receive mode.

The ADV_EXT_IND packets indicate the use of Periodic Advertising mode. These packets are sent on primary channels dedicated to advertising and point to AUX_ADV_IND packets, which contain detailed information about the periodic advertisement, including the interval, hopping sequence, advertiser address, and more. Additionally, the advertiser will send AUX_SYNC_IND packets at the specified interval, which contain the actual periodic advertisement data.

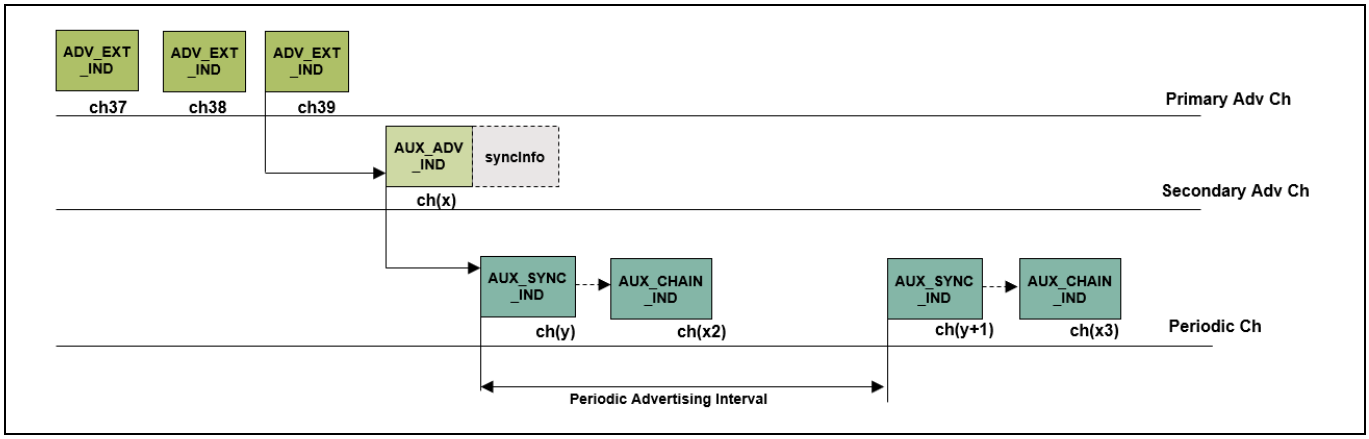

Figure 3 Periodic Advertising

2.3 Periodic Advertising with Response

Bluetooth® 5.4 introduced the Periodic Advertising with Responses (PAwR) feature, which is an extension of the Periodic Advertising Broadcast feature. As the name indicates, it allows synchronized devices or scanners to send back responses to the advertiser. PAwR supports bidirectional exchange of application data using connectionless communication, which was not previously possible with Bluetooth® LE. It can create a one-to-many topology with thousands of devices.

It operates by transmitting a sequence of packets at regular intervals within defined time slots called “subevents”. When a scanner is synced with the PAwR network, it will only monitor the select subevents that are synchronized with it. Observers can wake up on the specified subevents and listen for the subevent Indication and send the response if required and go back to sleep. This energy-efficient approach leads to the possibility of devices operating on a coin-cell battery for an extended period of time for several years.

PAwR is a suitable choice for applications that involve the transmission and reception of messages between a central hub device and a large number of other devices within a network. The Electronic Shelf Label (ESL) profile uses PAwR to transport messages containing commands and responses.

2.4 PAwR timing diagram

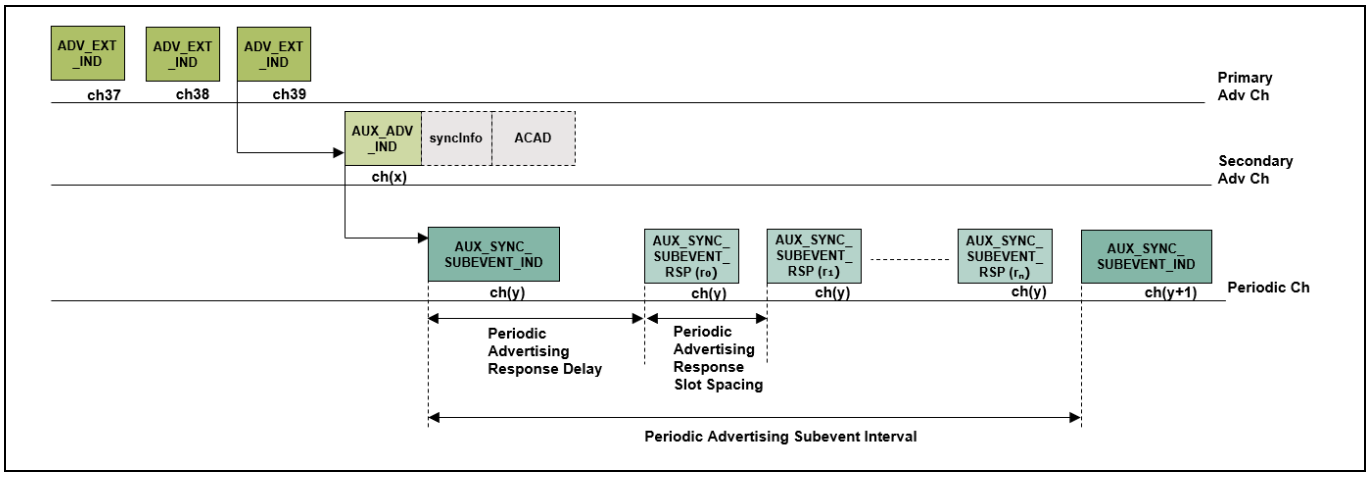

Figure 4 PAwR with subevents

PAwR operates on an interval called “periodic advertising interval”. As with other advertising modes, activity takes place in events, which are known as “periodic advertising with responses events” (PAwR events). These events occur at fixed intervals, with no random perturbation in scheduling. Each PAwR event consists of one or more subevents; during subevents, periodic advertising data and response data are transmitted.

As shown in Figure 1, advertisers will send ADV_EXT_IND packets on the primary advertising channel, followed by AUX_ADV_IND and AUX_SYNC_SUBEVENT_IND. Subevent indications (AUX_SYNC_SUBEVENT_IND) which carry data from the advertiser are sent periodically for every subevent interval. AUX_SYNC_SUBEVENT_RSP packets are the data packets, which are sent by the synchronized devices. The sequence of AUX_SYNC_SUBEVENT_IND and AUX_SYNC_SUBEVENT_RSP PDUs forms a periodic advertising with response train.

Scanners first listen to the ADV_EXT_IND packets, which give the information of the time offset, PHY, and the channel index of AUX_ADV_IND. The observer then scans on that channel at the given timing offset and receives an auxiliary advertising packet (AUX_ADV_IND). Synchronization information for PAwR is contained in the *SyncInfo* field and *ACAD field* of AUX_ADV_IND PDUs. The *SyncInfo* field contains the periodic advertising interval value and *syncPacketWindowOffset*. The observer then calculates how often PAwR events occur. The Additional Controller Advertising Information (ACAD) field of the AUX_ADV_IND PDU provides the extra information of PAwR timing. This information contains the Response Access Address, number of subevents, subevent Interval, response slot delay, and response slot spacing.

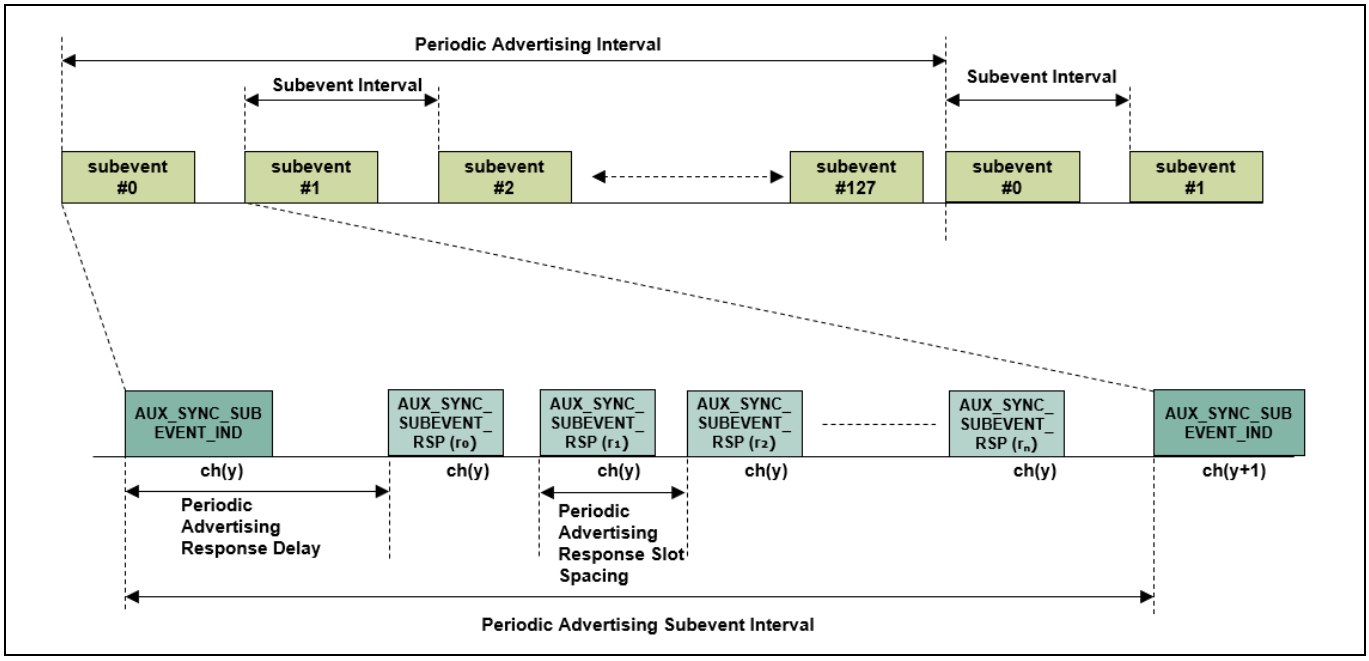

Figure 5 PAwR subevent with response slots

A PAwR event is divided into multiple subevents (up to 128) in each periodic advertising interval. In each subevent, the advertiser transmits one packet, which usually contains an AUX_SYNC_ SUBEVENT_IND PDU. After a delay, known as the “Periodic Advertising Response Slot Delay”, a series of time slots are reserved within the same subevent for receiving responses from synchronized devices. For AUX_SYNC_SUBEVENT_IND PDUs, responses will be sent in AUX_SYNC_SUBEVENT_RSP PDUs.

Scanners are synchronized to this subevent and scan this packet and process the payload. After the subevent packet transmission, there is a train of response slots (up to 255) for the scanners to send their responses back to the advertiser. The selection of scanner to send and response slot is up to the application to define. To avoid collision, only one scanner should respond in a response slot.

Table 2 Timing definition

Parameter	Definition
Advertising periodic interval	Integer multiple of 1.25 ms in the range 7.5 ms to 81.91875 s
Advertising subevent interval	Integer multiple of 1.25 ms in the range 7.5 ms to 318.75 ms
Periodic advertising response slot delay	Integer multiple of 625 µs in the range 1.25 ms to 10 ms
Periodic advertising response slot duration	Integer multiple of 625 µs in the range 625 µs to 158.75 ms

2.5 Synchronization

PAwR uses two procedures for observers to obtain the synchronization information:

- Establish synchronization with the advertiser from ADV_EXT_IND PDUs
- Using the periodic advertising sync transfer (PAST) procedure

2.5.1 Synchronization with ADV_EXT_IND PDUs

For synchronizing with the Advertiser using ADV_EXT_IND PDUs, the Observer device should have the essential details to efficiently detect and receive packets transmitted by the advertising device. This process doesn’t require connection but the Observer must know how often PAwR events will occur and when the next such

event will occur. This information is provided in the *SyncInfo* field of AUX_ADV_IND PDUs. It contains the periodic advertising interval and syncPacketWindowOffset.

The Observer also needs information about subevents such as their occurrence frequency and the number of subevents that can be accommodated by each periodic advertising with responses event. It requires specific details of the time slots allocated for transmitting responses within each subevent. This information is provided in the ACAD field of AUX_ADV_IND PDUs. This field contains the parameters known as *Subevent_Interval, Num_Subevents, Response_Slot_Delay, Response_Slot_Spacing*, and *Num_Response_Slots*.

The Observer needs to know which subevent number to synchronize to, and what response slot to use to respond to the Advertiser. One or more scanner devices may be synchronized to the same subevent. An individual scanner may be synchronized to receive during one or more subevents. The application layer of the scanner is responsible for these concerns.

For example, after synchronizing with the PAwR stream, the Observer will wake up on the specified subevent (#1, for example), and listen for the subevent indication. If the response is not required, the Observer may sleep for the periodic advertising interval until Subevent #1 is scheduled again. If a response is required, the Observer will wake up to send the response in the specified slot; the Observer can sleep until the next Subevent #1.

2.5.2 Periodic Advertising Sync Transfer (PAST) procedure

When using the PAST procedure, the PAwR Broadcaster can pass the required synchronization data over an LE ACL connection to the Observer. During this process, no scanning for AUX_ADV_IND PDUs is necessary by either device. Sync info, response access address, number of subevents, subevent Interval, response slot delay, and response slot spacing are passed over the LE ACL connection in a new PDU type called "LL_PERIODIC_SYNC_WR_IND".

2.6 PAwR vs. Bluetooth® LE Mesh

The following table provides a comparison between PAwR network and Bluetooth® LE Mesh.

Table 3 Difference between PAwR and Bluetooth® LE Mesh

Feature	PAwR	Bluetooth® LE Mesh	Notes
Range	Standard	Extended range with relays	Mesh natively supports extended range
Throughput/link Utilization	High	Low	Mesh is only using advertising channel
Synchronization	Synchronized; each node maybe assigned a time slot for transmission	Unsynchronized	PAwR efficiently uses bandwidth, no collisions on the network level
Channel use	Use 40 channels	Use three advertising channels	PAwR efficiently uses the 2 GHz band. It can use adaptive frequency hopping (AFH) to avoid channels under interference
Number of nodes	>10k nodes	1,000s of nodes	Requires application-dependent configuration
Coexistence	Good	Bad	Bluetooth® LE Mesh may cause higher interference on the advertising channel, causing problems with connection or discovery in a smart home or office

Different types of advertising

Feature	PAwR	Bluetooth® LE Mesh	Notes
Power consumption	Low	High	Bluetooth® LE Mesh will scan all the time for listening

3 PAwR implementation in CYW20829

3.1 Advertiser

Do the following to implement the advertiser role in CYW20829:

1. Initialize the `app_pawr_subevt_ind_cb` and `app_pawr_subevt_rsp_report_cb` callback functions in the code.
   - The `app_pawr_subevt_ind_cb` callback function is called when the advertiser has to update the subevent data packet. This will be called inside the `WICED_BT_BLE_PAWR_SUBEVENT_DATA_REQ_EVENT` event. The controller will send this event to the host; this event is generated when the controller is ready to transmit one or more subevents and requests the advertising data for these subevents. The controller should request data for as many subevents as it has memory to accept to minimize the number of events generated by the controller.
   - The `app_pawr_subevt_rsp_report_cb` callback function is called when there is a response from the scanners to subevents. This callback function will be called inside the `WICED_BT_BLE_PAWR_RSP_REPORT_EVENT EVENT`. This event indicates that one or more Bluetooth® devices have responded to a periodic advertising subevent during a PAwR train.
2. Initialize the `wiced_bt_ble_register_adv_ext_cback` function to register an application callback function to receive extended advertising events. Set the Extended Adv and PAwR parameters using `wiced_bt_ble_set_ext_adv_`parameters and `wiced_bt_ble_set_pawr_params` functions as follows:

Code listing 1

```
/* enum for Extended ADV parameters */
enum extend_adv_param_e
{
EXT_ADV_EVENT_PROP      = 0x00, /* extended adv event_properties */
EXT_ADV_CHANNEL_MAP     = 0x07, /* primary_adv_channel_map 37/38/39 */
EXT_ADV_TX_POWER        = 0x7F, /* extended adv tx power */
EXT_ADVSEC_ADV_MAX_SKIP = 0x00, /* extended adv secondary_adv_max_skip */
EXT_ADV_SET_ID          = 0x00, /* extended adv sid data in the subfield of the ADI field
*/
EXT_ADV_EVT_PERIOD      = 800   /* extended adv period *0.625 = 500ms */
};

/* enum for PAwR parameters */
enum pawr_param_e
{
PAWR_EXT_ADV_EVENT_PROP = 0x00, /* extended adv properties */
PAWR_SUB_EVT_NUM        = 16,   /* PAwR sub event number */
PAWR_EVT_PERIOD         = 4000, /* PAwR event period *1.25  = 5000ms */
PAWR_SUB_EVT_PERIOD     = 250,  /* PAWR_EVT_PERIOD/PAWR_SUB_EVT_NUM 312.5ms */
PAWR_RESPONSE_DELAY     = 20,   /* PAwR response delay *1.25  = 25ms */
PAWR_RESPONSE_SPACE     = 40,   /* PAwR response space  *0.125 = 5ms */
PAWR_RESPONSE_SLOTS     = 8,    /* PAwR response slot number */
PAWR_ADV_HANDLE         = 0x01  /* PAwR adv handle */
```

```
    };




    static void pawr_create_pawr_network(void)
    {
        wiced_bt_dev_status_t              status = WICED_BT_ERROR;
        wiced_bt_ble_periodic_adv_params_t periodic_adv_params;
        /* set the Extended Adv paramameters */
        status = wiced_bt_ble_set_ext_adv_parameters(PAWR_ADV_HANDLE,
                                                     EXT_ADV_EVENT_PROP,
                                                     EXT_ADV_EVT_PERIOD,
                                                     EXT_ADV_EVT_PERIOD,
                                                     EXT_ADV_CHANNEL_MAP,
                                                     BLE_ADDR_PUBLIC,
                                                     BLE_ADDR_PUBLIC,
                                                     NULL_BDA,
                                                     BTM_BLE_ADV_POLICY_ACCEPT_CONN_AND_SCAN,
                                                     EXT_ADV_TX_POWER,
                                                     WICED_BT_BLE_EXT_ADV_PHY_1M,
                                                     EXT_ADVSEC_ADV_MAX_SKIP,
                                                     WICED_BT_BLE_EXT_ADV_PHY_2M,
                                                     EXT_ADV_SET_ID,

    WICED_BT_BLE_EXT_ADV_SCAN_REQ_NOTIFY_DISABLE);
        printf("wiced_bt_ble_set_ext_adv_parameters returned:%d\n",status);
        periodic_adv_params.adv_int_min       = PAWR_EVT_PERIOD;
        periodic_adv_params.adv_int_max       = PAWR_EVT_PERIOD;
        periodic_adv_params.adv_properties    = PAWR_EXT_ADV_EVENT_PROP;
        periodic_adv_params.subevent_num      = PAWR_SUB_EVT_NUM;
        periodic_adv_params.subevent_interval = PAWR_SUB_EVT_PERIOD;
        periodic_adv_params.rsp_slot_delay    = PAWR_RESPONSE_DELAY;
        periodic_adv_params.rsp_slot_spacing  = PAWR_RESPONSE_SPACE;
        periodic_adv_params.rsp_slot_num      = PAWR_RESPONSE_SLOTS;

        /* set the Periodic ADV Parameters (v2) */
        status = wiced_bt_ble_set_pawr_params(PAWR_ADV_HANDLE, &periodic_adv_params);
        printf("set_pawr_params:status:%d, pawr_evt_period:%d, sub_evt_period:%d,
    sub_evt_num:%d\n",
                status,
                PAWR_EVT_PERIOD,
                PAWR_SUB_EVT_PERIOD,
                PAWR_SUB_EVT_NUM);
        pawr_finish_creating_pawr_network();
}
```

3. Complete the PAwR Network creation by starting periodic and extended advertisement using `wiced_bt_ble_start_periodic_adv` and `wiced_bt_ble_start_ext_adv` functions as follows:

Code listing 2

```
static void pawr_finish_creating_pawr_network(void)
{
    printf("pawr_finish_creating_pawr_network\n");
    wiced_bt_ble_ext_adv_duration_config_t ext_cfg = {PAWR_ADV_HANDLE, 0, 0};
    /* Set periodic ADV enable */
    printf("wiced_bt_ble_start_periodic_adv\n");
    wiced_bt_ble_start_periodic_adv(PAWR_ADV_HANDLE, WICED_TRUE);
    printf("wiced_bt_ble_start_ext_adv\n");
    /* Set extended ADV enable */
    wiced_bt_ble_start_ext_adv(WICED_TRUE, 1, &ext_cfg);
    printf("pawr start\n");
}
```

4. Within the `app_pawr_subevt_ind_cb` function, set the subevent indication data using the `wiced_bt_ble_set_pawr_subevent_ind_data` API as follows:

Code listing 3

```
void app_pawr_subevt_ind_cb(uint8_t adv_handle,uint8_t subevent_start,uint8_t
total_subevent)
{
    wiced_bt_dev_status_t                 status = WICED_BT_ERROR;
    wiced_bt_ble_pawr_subevent_ind_data_t ind;
    uint8_t                              *snd    = ind.ind_data;
    if (adv_handle != PAWR_ADV_HANDLE)
    {
        printf("rcv pawr subevent ind
error:handle:%d,sub:%d,total_sub:%d\n",adv_handle,subevent_start,total_subevent);
        return;
    }
    if (subevent_start == SUBEVT0)
    {
        ind.subevent_num    = SUBEVT0;
        ind.rsp_slot_start  = PAWR_SLOT_START;
        ind.rsp_slot_count  = PAWR_SLOT_COUNT;
        ind.ind_data_length = PAWR_BUF_SIZE;
        ARRAY_TO_STREAM(snd, pawr_subevent0_data,PAWR_BUF_SIZE);
    }
    else if (subevent_start == SUBEVT1)
    {
        ind.subevent_num    = SUBEVT1;
        ind.rsp_slot_start  = PAWR_SLOT_START;
        ind.rsp_slot_count  = PAWR_SLOT_COUNT;
        ind.ind_data_length = PAWR_BUF_SIZE;
```

```
            ARRAY_TO_STREAM(snd, pawr_subevent1_data,PAWR_BUF_SIZE);
        }
        status = wiced_bt_ble_set_pawr_subevent_ind_data(adv_handle, 1, &ind);
        if (status != WICED_SUCCESS)
        {
            printf("set_pawr_subevent_ind_data failed:0x%04x\n",status);
        }
}
```

3.2 Scanner

Do the following to implement the scanner role in CYW20829:

1. Initialize `app_pawr_se_rsp_cb`, `app_pawr_conn_up_cb`, and `app_pawr_conn_down_cb` callback functions in the code.
   - The `app_pawr_se_rsp_cb` callback function is called when the scanner wants to send the responses to subevents. This callback function will be called inside the `WICED_BT_BLE_PAWR_IND_REPORT_EVENT` event. The controller will send this event to the host; this event is generated when the controller has received a periodic advertisement data. Within the callback function, the scanner will send the subevent response data to the advertiser.
   - The `app_pawr_conn_up_cb` callback function is called when the `WICED_BT_BLE_PAWR_SYNC_ESTABLISHED_EVENT` event is received. This event will be generated when synchronization is established, and the scanner starts receiving periodic advertising packets.
   - The `app_pawr_conn_down_cb` function is called when the `WICED_BT_BLE_PERIODIC_ADV_SYNC_LOST_EVENT` event is received. This event will be generated when synchronization is lost and the scanner has not received a periodic advertising packet from the train within the timeout period. Within the callback function, the scanner will restart synchronization with periodic advertising from an advertiser.
2. Initialize the `wiced_bt_ble_register_adv_ext_cback` function to register an application callback function to receive extended advertising events. Start the scan and synchronize with the advertiser using `wiced_bt_ble_scan` and `wiced_bt_ble_create_sync_to_periodic_adv` functions.

Code listing 4

```
  void pawr_scan_for_pawr_network(void)
  {
     wiced_bt_dev_status_t           status = WICED_BT_ERROR;
      wiced_bt_ble_ext_scan_config_t  scan_cfg;
      scan_cfg.scanning_phys = WICED_BT_BLE_EXT_ADV_PHY_1M;
      scan_cfg.duration      = 0;
      scan_cfg.period        = 0;
      /* set extended scan parameters */
      status = wiced_bt_ble_cache_ext_scan_config(&scan_cfg);
      printf("wiced_bt_ble_cache_ext_scan_config:result:%d\n",status);
      /* set extended scan enable */
      status = wiced_bt_ble_scan(BTM_BLE_SCAN_TYPE_HIGH_DUTY, WICED_FALSE, NULL);
      printf("wiced_bt_ble_scan:result:%d\n",status);
      status =
  wiced_bt_ble_create_sync_to_periodic_adv(WICED_BT_BLE_IGNORE_SYNC_TO_PERIODIC_ADV_LIST,
```

```
                                                    EXT_ADV_SET_ID,
                                                    BLE_ADDR_PUBLIC,
                                                    pawr_central_address,
                                                    0,
                                                    PERIODIC_ADV_EXPIRD_TIME,
                                                    0);
    printf("pawr_scan_for_pawr_network:addr: ");
    app_bt_util_print_bd_address(pawr_central_address);
    printf("pawr_scan_for_pawr_network status:%d\n", status);
    printf("pawr start\n");
}
```

3. For every advertisement data, send back the response data to the advertiser using `wiced_bt_ble_set_pawr_subevent_rsp_data` as follows:

Code listing 5

```
void app_pawr_se_rsp_cb(uint16_t sync_handle, uint8_t *p_msg, uint16_t msg_len, uint8_t
subevent_num, uint16_t evt_counter)
{
    wiced_bt_dev_status_t  status      = WICED_BT_ERROR;
    static uint32_t        rcv_se0_cnt = 0;
    static uint32_t        rcv_se1_cnt = 0;
    uint8_t rcv_buf[PAWR_BUF_SIZE];
    uint8_t *rcv = rcv_buf;
    uint8_t snd_buf[PAWR_BUF_SIZE];
    uint8_t *snd = snd_buf;
    if (((subevent_num == SUBEVT0) || (subevent_num == SUBEVT1)) && (msg_len ==
PAWR_BUF_SIZE))
    {
        switch(subevent_num)
        {
            case SUBEVT0:
                printf("rcv:se:%d,cnt:%lu,\n",subevent_num,rcv_se0_cnt++);
                if (memcmp(p_msg,pawr_subevent0_data,PAWR_BUF_SIZE))
                {
                    printf("se0 error:\n");
                    app_bt_util_print_byte_array(p_msg,msg_len);
                    return;
                }
            break;
            case SUBEVT1:
                printf("rcv:se:%d,cnt:%lu\n",subevent_num,rcv_se1_cnt++);
                if (memcmp(p_msg,pawr_subevent1_data,PAWR_BUF_SIZE))
                {
                    printf("se1 error:\n");
                    app_bt_util_print_byte_array(p_msg,msg_len);
                    return;
```

```
                    }
                break;
            }
            STREAM_TO_ARRAY(rcv,p_msg,PAWR_BUF_SIZE);
            ARRAY_TO_STREAM(snd,rcv_buf,PAWR_BUF_SIZE);
            status = pawr_snd_se_rsp_central(sync_handle,
                                             evt_counter,
                                             subevent_num,
                                             subevent_num,
                                             PAWR_PERIPHERAL_RSP_SLOT,
                                             PAWR_BUF_SIZE,
                                             snd_buf);
            if (status != WICED_SUCCESS)
            {
                printf("pawr snd rsp error:%d\n",status);
                return;
            }
        }
    }
```

4 Hardware setup

4.1 CYW20829 evaluation board

The CYW20829 M.2 board (CYW920829M2IPA2) consists of the 56-pin silicon target device on a custom pinout M.2 card that is interfaced with the baseboard (CYW9BTM2BASE3-01). CYW20829 silicon pins are brought out to the baseboard through the M.2 connector. This board has peripherals like UART and I2C that can be used with sensors and displays. The board is powered using USB micro-B cable. See the CYW920829M2EVK-02 kit user guide for more information.

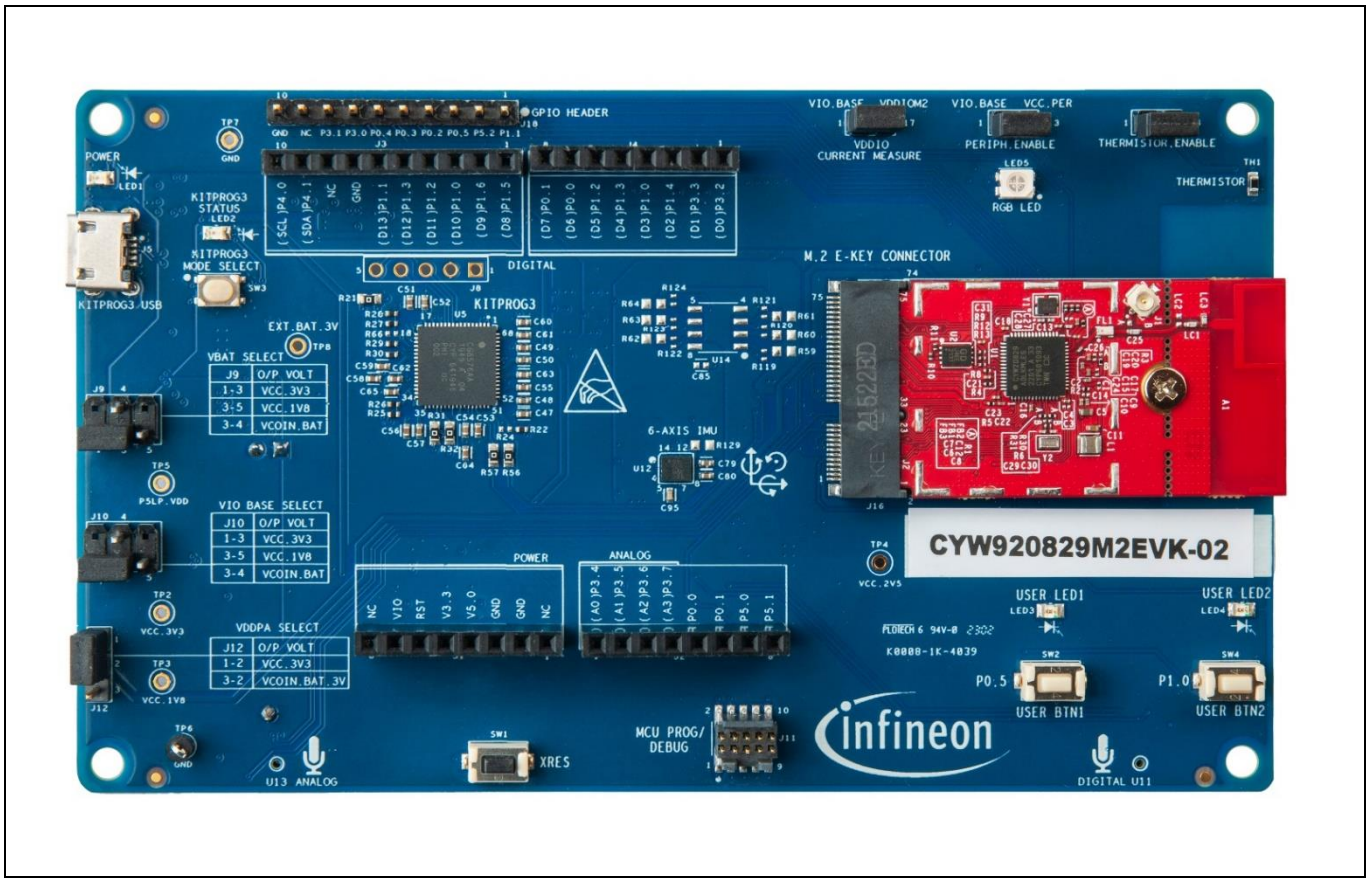

Figure 6 Evaluation board

4.2 PAwR test setup

Four evaluation boards and one CY8CKIT-028-SENSE display board are used to set up the basic PAwR demo. Three CYW920829M2EVK-02 boards are used in the scanner role and one board act as the advertiser. A CY8CKIT-028-SENSE display board is used to show the temperature data and response slots of the scanner devices. In this test, scanner devices will send the temperature data to an advertiser through responses.

The CYW920829M2EVK-02 board has a built-in thermistor sensor which is connected to the analog pin A0. Scanner devices measure the temperature using the thermistor sensor and send the data through responses.

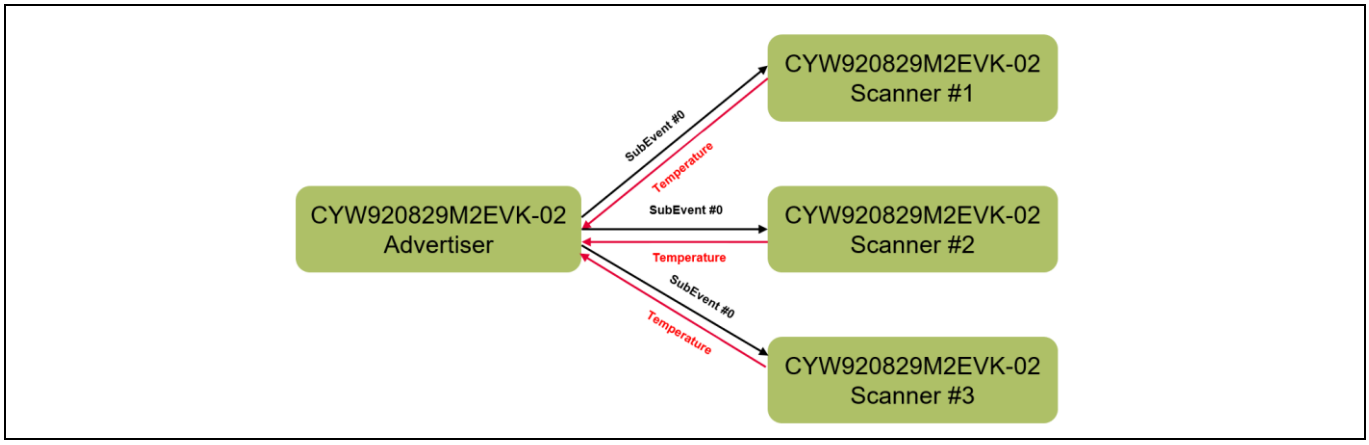

Figure 7 Test setup

The setup block is shown in Figure 4. All the evaluation kits are connected to a PC over a 2-meter-long USB cable. The boards are mounted on the 2-meter length of pipe. The boards are powered using the same USB connector and the serial logs are captured in the PC through the serial monitor. One board is configured as the server while the other acts as the client.

4.3 Display board

The display board is used to show the temperature data and response slots of all the scanner devices.

5 Software and tools

This section describes the required software tools to create and build the application for PAwR.

5.1 ModusToolbox™ software

ModusToolbox™ software is a modern, extensible development environment supporting a wide range of Infineon microcontroller devices. It provides a flexible set of tools and a diverse, high-quality collection of application-focused software. These include configuration tools, low-level drivers, libraries, and operating system support, most of which are compatible with Linux-, macOS-, and Windows-hosted environments.

Download ModusToolbox™ from here. See the Eclipse IDE for ModusToolbox™ user guide for more details.

5.2 Code examples

See the GitHub code example page for ModusToolbox™ examples. The following two code examples are available in the ModusToolbox™ 3.x for the PAwR application.

- mtb-example-btstack-freertos-le-pawr-client
- mtb-example-btstack-freertos-le-pawr-server

These examples can be imported using the Project Creator tool to create an application and use it directly with ModusToolbox™ tools.

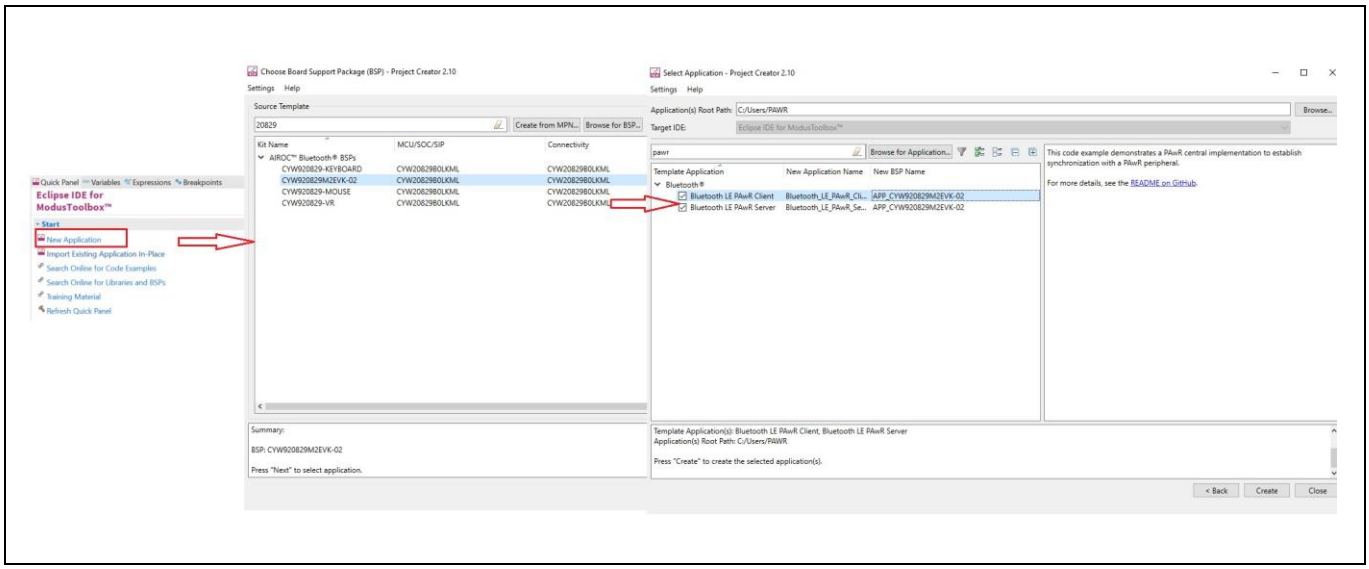

Figure 8 Importing code examples

5.3 PAwR client application

This code example configures the device as a Bluetooth® PAwR client, which works with the PAwR server. The PAwR client starts extended advertising with the PAwR parameters and PAwR advertising.

5.4 PAwR server application

The code example configures the device as a Bluetooth® PAwR server, which works with the PAwR client. The PAwR server starts scan when it finds the PAwR configuration in extended advertising, and then starts to sync with the PAwR client using PAwR parameters.

5.5 Build and flash the firmware

The PAwR application can be built with help of ModusToolbox™ or the modus-shell command line tool. The binary is generated in the *build* folder. The firmware can be loaded into the evaluation kits using KitProg3 or using external SWD with CYPRESS™ Programmer.

5.6 Download the project from the blog

Some changes are made to the Bluetooth® LE PAwR server and Bluetooth® LE PAwR client code example projects to showcase the temperature information of the scanner devices on the advertiser. The modified project is posted in the blog Monitoring Temperature with Bluetooth® Using PAwR with CYW20829. For the PAwR application demo, download the project present in the blog.

6 Testing and results

To test this project, four CYW920829M2EVK-02 boards and one CY8CKIT-028-SENSE display board are required. Three CYW920829M2EVK-02 boards are used in the scanner role and one board act as the advertiser. A CY8CKIT-028-SENSE display board is used to display the temperature data and response slots of the scanner devices.

1. Connect the CY8CKIT-028-SENSE kit to one of the CYW920829M2EVK-02 evaluation kits to set up an Advertiser device.
2. Connect all four boards to your PC using the provided USB cable through the USB connector.
3. Open four Tera Term (or your preferred serial terminal application tool) windows for all the boards. One for PAwR Advertiser board and the other three for PAwR Scanner boards. Configure the terminal application to access the serial port using the settings:
   - **Baud rate:** 115200 bps
   - **Data:** 8 bits
   - **Parity:** None
   - **Stop:** 1 bit
   - **Flow control:** None
   - **New line for receiving data:** Line Feed (LF) or auto setting.
4. Download the project zip file from the blog link and open the projects in ModusToolbox™ 3.x.
5. Build and Program the PAwR Advertiser from ModusToolbox™ 3.x and observe the OLED display with empty temperature data in °C as shown below. The device will start advertising immediately after programming the PAwR Advertiser project.

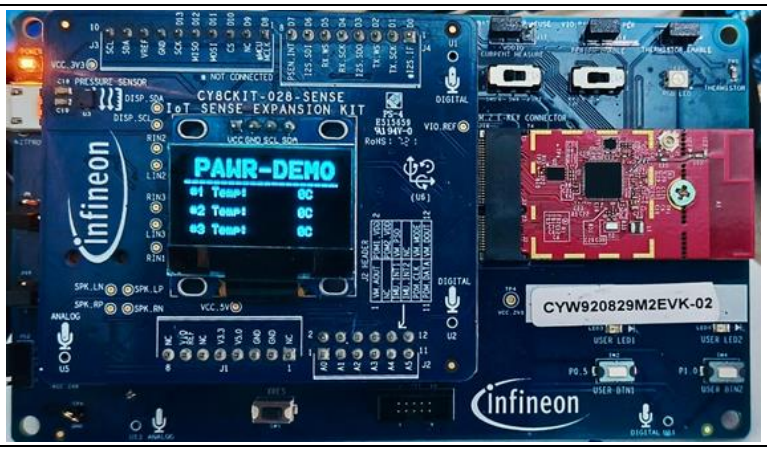

Figure 9 Advertiser displays empty temperature data in OLED display

6. In the PAwR Scanner project, change the Response slot number (`PAWR_PERIPHERAL_RSP_SLOT` macro in *pawr_app.h* file) and peripheral address (`app_peripheral_address` array in the *pawr_app.c* file) of different scanner devices and program the project. Devices will start scanning immediately after programming the PAwR Scanner project.
7. Reset all the boards once and wait for PAwR Advertiser and PAwR Scanners to establish synchronization. Devices will automatically sync after programming the kits and it will take less than 5 seconds for synchronization.

For detailed information on the demo, refer to the Monitoring Temperature with Bluetooth® Using PAwR with CYW20829 blog.

References

[1] Infineon Technologies AG: *GitHub Code Examples for ModusToolbox ™ software*; Available online

[2] Infineon Technologies AG: *AIROC™ CYW20829 Datasheet*; Available online

[3] Infineon Technologies AG: *ModusToolbox™ software*; Available online

[4] Infineon Technologies AG: *Bluetooth® Sig Spec*; Available online

[5] Infineon Technologies AG: *Bluetooth® Sig Technical overview of Bluetooth v5.4*; Available online

[6] Infineon Technologies AG: *A lecture on the topic of PAwR and ESL profile from Bluetooth® SIG*; Available online

[7] Infineon Technologies AG:*Infineon Periodic Advertising with Responses (PAwR) Deep Dive and Demo*; Available online

Revision history

Document revision	Date	Description of changes
**	2024-07-26	Initial release

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2024-07-26

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2024 Infineon Technologies AG. All Rights Reserved.

Do you have a question about this document?

Email: erratum@infineon.com

Document reference

002-40333 Rev. **

Important notice

The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

Warnings

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.