AN12872

使用GFSK反中继攻击

Rev. 1 — 09/2020 Application Note

URL：https://www.nxp.com.cn/docs/en/application-note/AN12872.pdf

目录

1 介绍

本文档介绍使用GFSK反中继攻击，重点介绍了多链路监控的概念和应用。
它提出了一个采用 KW36 或 KW38 无线 MCU 的系统实现。

Kinetis MKW36 或 KW38 是一种无线 MCU，支持蓝牙 LE v5.0 协议和通用 FSK (GFSK) 调制方式。

本文档的读者应具备Arm ®MCU架构和无线电通信的基础知识。

2 使用GFSK反中继攻击的用例

基于RSSI的定位是一种简单有效的定位解决方案。使用蓝牙 LE 进行基于 RSSI 的定位有两种主要方式，即无连接方式和有连接方式。关于基于RSSI的定位的详细知识和实现，请参见KW36 Localization Based on RSSI Ranging Application（文档AN12865）和KW38 Localization Based on RSSI Ranging Application（文档AN12977）。为了方便起见，我们以汽车进入系统为例，这是一个典型的需要距离估算的系统。

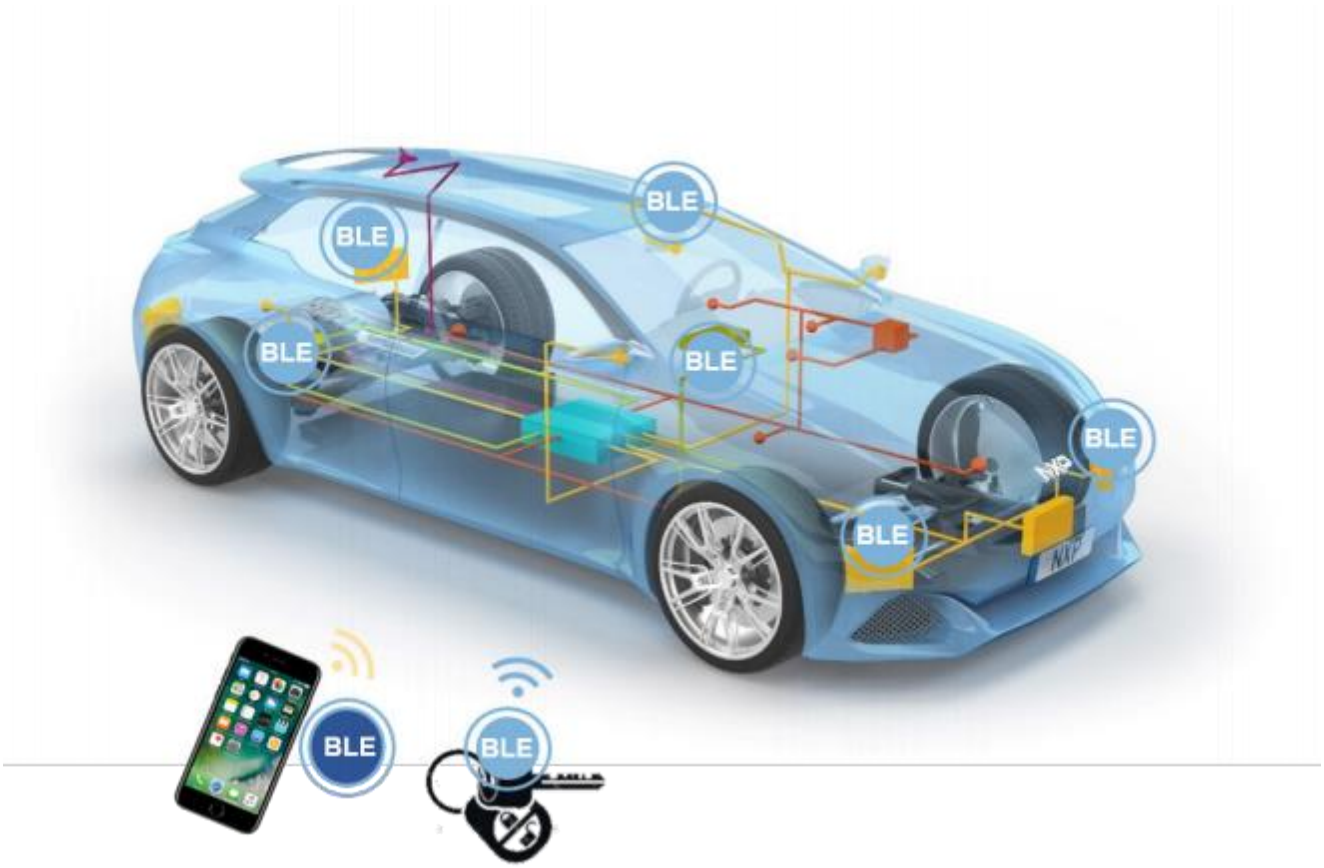


图 1. 无连接方式或有连接方式

无连接方式是通过三个广播通道实现的，设备之间没有连接。 这种方法实现简单，易于部署，但不能防止中继攻击。

有连接方式需要设备之间的蓝牙 LE 连接。 如果启用配对和绑定，则此方法可以实现反中继攻击。 但是，当有多个连接时，会出现功耗增加、响应速度降低、部署困难等问题。

为了避免以上两种方式的问题，我们建议使用GFSK来防止中继攻击。这种方法适用于有多个 BLE/GFSK 节点的汽车进入系统。这样，连接可以保证安全，监控可以避免多设备连接的存在，从而降低系统复杂度。

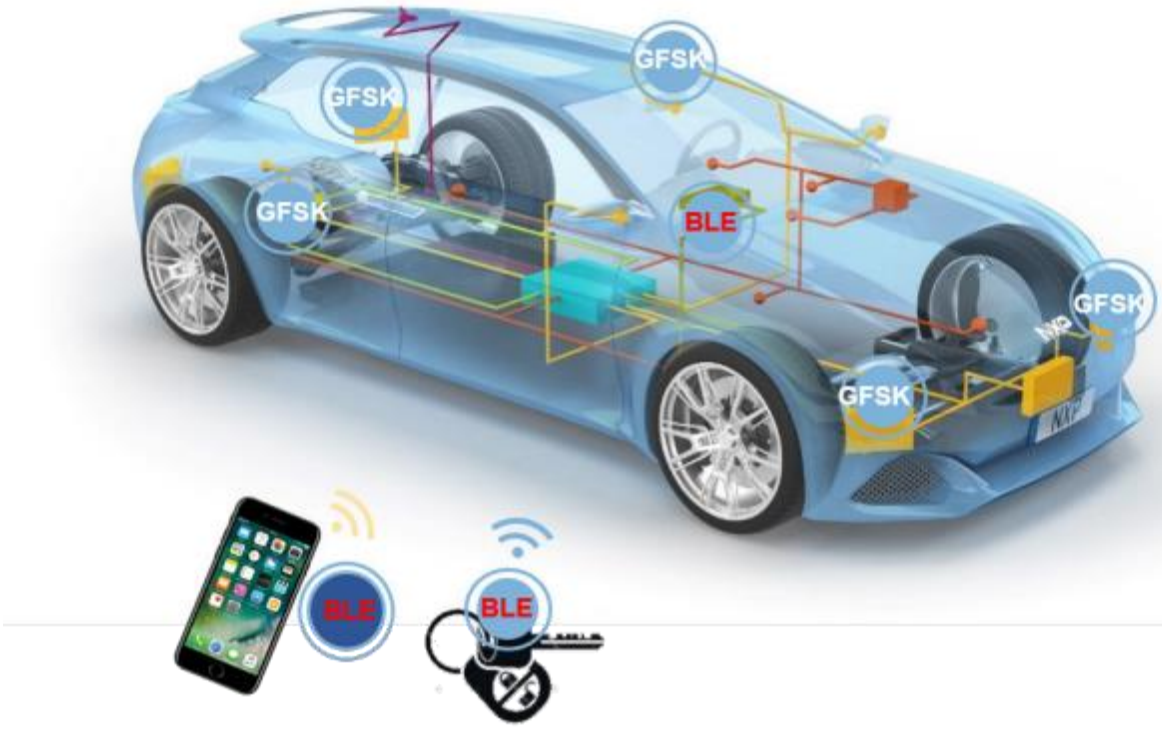


图 2. 使用GFSK反中继攻击

3 使用GFSK实现反中继攻击

GFSK 协议支持使用自定义 GFSK/GMSK 或 MSK 调制方式进行无线电操作，该方式是通过对一组 PHY 变量（例如 BT 产品类型、调制指数和调制滤波器系数等）进行编程来实现的。通过适当的参数配置，GFSK 可以接收蓝牙 LE 数据包。

通过对所有通道的使用并获得的以下连接信息后，我们可以根据通道选择算法追踪蓝牙 LE 的链接数据。

- 访问地址
- 连接间隔
- 信道跳跃增量
- CRC 种子

图 3 是单个蓝牙 LE 连接的框图。它包含三个角色，主机、从机和监控。主机和从机之间有蓝牙LE连接，主机和监控之间有CAN总线或LIN总线通信。蓝牙 LE 主机连接到从机，并通过 CAN 或 LIN 总线将连接信息传输到 GFSK 监控。然后，GFSK监控开始通过这些信息解析空中数据包。

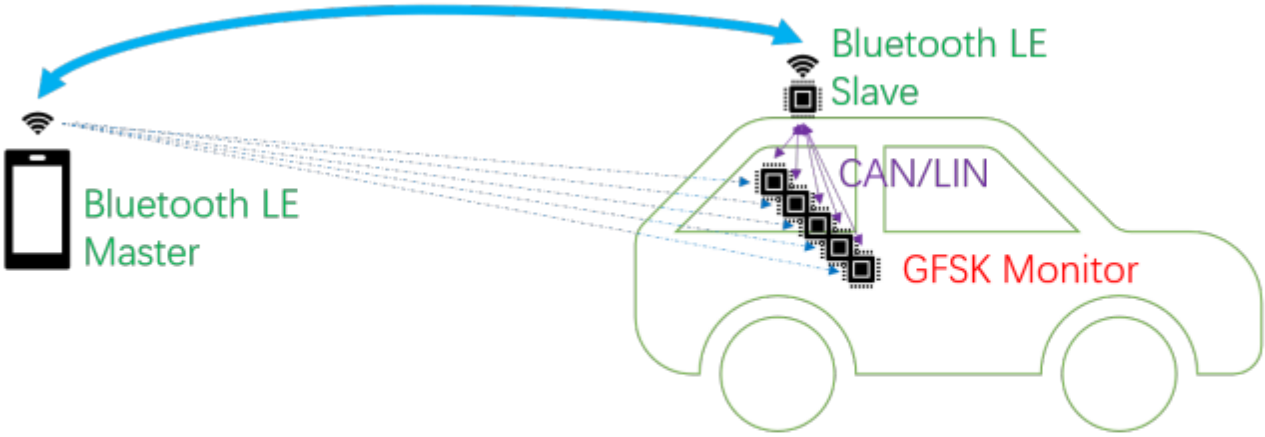


图 3. 监控一个连接

3.1 GFSK参数配置

修改以下宏配置以适应蓝牙 LE 的参数要求。

- #define gGenFskDefaultLengthFieldSize_c (8)/*!< Number of bits in the LENGTH field. */
- #define gGenFskDefaultH1FieldSize_c (0)/*!< Number of bits in the H1 field. */
- #define gGenFskDefaultH0Value_c (0x0000)/*!< H0 field value. */
- #define gGenFskDefaultH0Mask_c 0/*!< Mask to select which bits of H0 must match the h0_match field. */
- #define gGenFskDefaultH1Value_c (0x0000)/*!< H1 field value. */
- #define gGenFskDefaultH1Mask_c 0/*!< Mask to select which bits of H1 must match the h1_match field. */

3.2 连接信息

CONNECT_REQ PDU包含我们上面需要的连接信息。

表 1. CONNECT_REQ PDU的有效数据中的 LLData 字段结构

LLData									
AA (4 octets)	CRCInit (3 octets)	WinSize (1 octet)	WinOffset (2 octets)	Interval (2 octets)	Latency (2octets)	Timeout (2 octets)	ChM (5 octets)	Hop (5 bits)	SCA (3 bits)

表 1 是从空中接口数据包中获取的信息。该信息存在于 KW3x 的数据通道寄存器中。主机获取此信息，然后通过 CAN 或 LIN 传输到监控。

地址 16位	地址 32位	注册名称	描述
0x80	0x100	CONN_INTERVAL	—
0x88	0x110	PDU_ACCESS_ADDR_L_REGISTER	—
0x8A	0x114	PDU_ACCESS_ADDR_H_REGISTER	—
0xF4	0x1E8	CONN_PARAM1	连接参数，例如，在本次连接的connect_request中交换的sca、hop_increment及crc_init。
0xF6	0x1EC	CONN_PARAM2	连接参数，，在本次连接的 connect_request 中交换的 crc_init 位 24:7。

3.3 频道选择算法

蓝牙LE使用跳频收发器，因此要追踪蓝牙LE的空中接口数据包，监控设备也应实现相同的跳频算法。

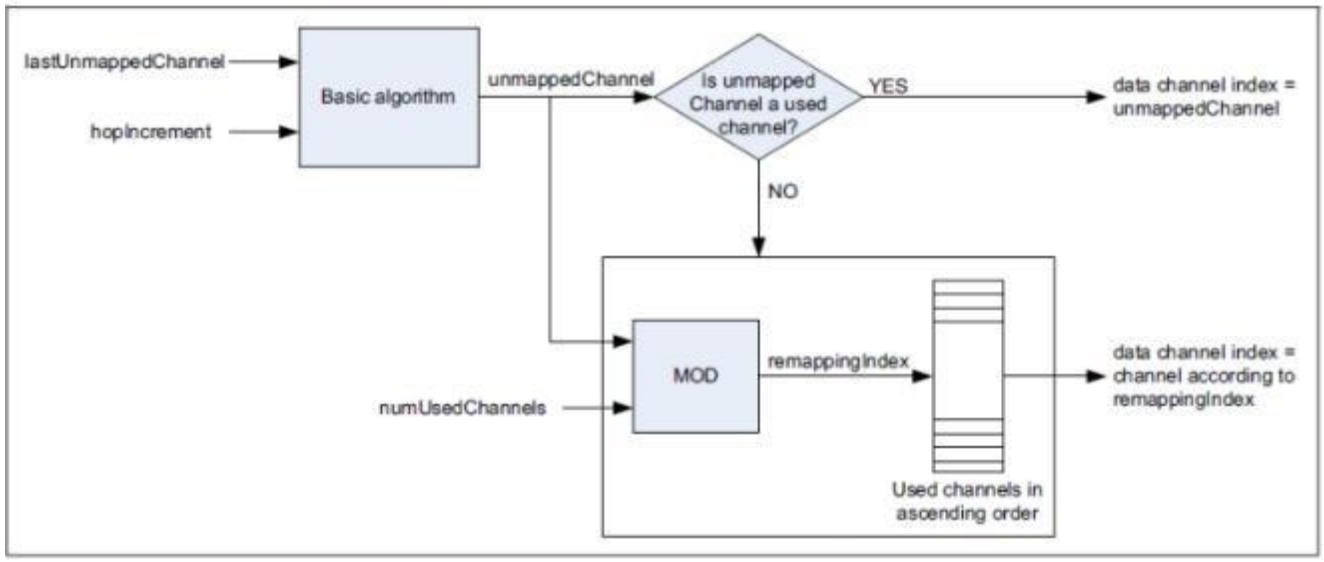


图 4. 数据通道选择算法框图

3.4 中断处理

由于每个连接事件都包含主机和从机数据包，因此需要修改 GFSK 中断处理程序以得到这两组数据。接收到主机数据包后，不清零超时定时器，并重新启动Rx。 接收到从机数据后，清零超时定时器并设置 Rx 完成事件。

```
#if LINK_MONITOR

#else
            GENFSK_TimeCancelEvent(&rxTimeoutTimer);
            GENFSK->T1_CMP &= ~GENFSK_T1_CMP_T1_CMP_EN_MASK;
```

```
#endif
            if  (genfskLocal[mGenfskActiveInstance].packetReceivedCallbackIsr  != NULL)
            {
                /* Examine RX buffer in ISR context if CallbackIsr is set */
                GENFSK_RxIsrContext();
            }

            if  (genfskLocal[mGenfskActiveInstance].enabledEvents  &  gGenfskRxEvent)
            {
#if LINK_MONITOR
                uint16_t  byteCount =  ((GENFSK->RX_WATERMARK  &  GENFSK_RX_WATERMARK_BYTE_COUNTER_MASK)
>>  GENFSK_RX_WATERMARK_BYTE_COUNTER_SHIFT);
                if(packetFromMaster == 1)          //master data
                {
                    GENFSK->XCVR_CTRL = 5;

                    packetFromMaster = 0;
                    masterPacketLen = byteCount;
                    FLib_MemCpy(masterPacket,  (uint8_t*)PACKET_BUFFER_BASE_ADDR, masterPacketLen);
                    masterRSSI  =  (int8_t)((GENFSK->XCVR_STS  &  GENFSK_XCVR_STS_RSSI_MASK)  >>
GENFSK_XCVR_STS_RSSI_SHIFT);
                }
                else                                //slave  data
                {
                    slaveRSSI  =  (int8_t)((GENFSK->XCVR_STS  &  GENFSK_XCVR_STS_RSSI_MASK)  >>
GENFSK_XCVR_STS_RSSI_SHIFT);
                    packetFromMaster = 1;
                    eventFlags  |= gGenfskRxEventFlag_c;

                    GENFSK_TimeCancelEvent(&rxTimeoutTimer);
                    GENFSK->T1_CMP  &= ~GENFSK_T1_CMP_T1_CMP_EN_MASK;
                }
#else
                eventFlags  |= gGenfskRxEventFlag_c;
#endif
            }
            else
            {
                /* No notification enabled */
                genfskLocal[mGenfskActiveInstance].genfskState  = gGENFSK_LL_Idle;
            }
```

3.5 禁用 DCOC 校准

为了处理从主机和从机接收到的数据包，由于DC偏移校准时间相对较长，所以DC偏移校准需要禁用。

```
static void BleGenfskDisableDcocCal(void)
{
    static uint8_t dcoc_cal_enabled = TRUE;
    if(TRUE  == dcoc_cal_enabled)
    {
        dcoc_cal_enabled = FALSE;

        XCVR_RX_DIG->RX_DIG_CTRL  &= ~XCVR_RX_DIG_RX_DIG_CTRL_RX_DCOC_CAL_EN_MASK;
        XCVR_TSM->TIMING36   = 0x3431FFFFU;
        XCVR_TSM->TIMING37   = 0x3231FFFFU;
        XCVR_TSM->TIMING39   = 0x3431FFFFU;
        XCVR_TSM->TIMING14   = 0x34316863U;
```

```
        XCVR_TSM->END_OF_SEQ = 0x34336E67U;
    }
}
```

3.6 多链路监控

图 5 显示了监控多个蓝牙 LE 连接的框图。它包含了三个角色，主机，从机和监控。主机和从机之间有四个蓝牙 LE 连接，主机和监视器之间有 CAN 总线或 LIN 总线通信以传输连接信息。

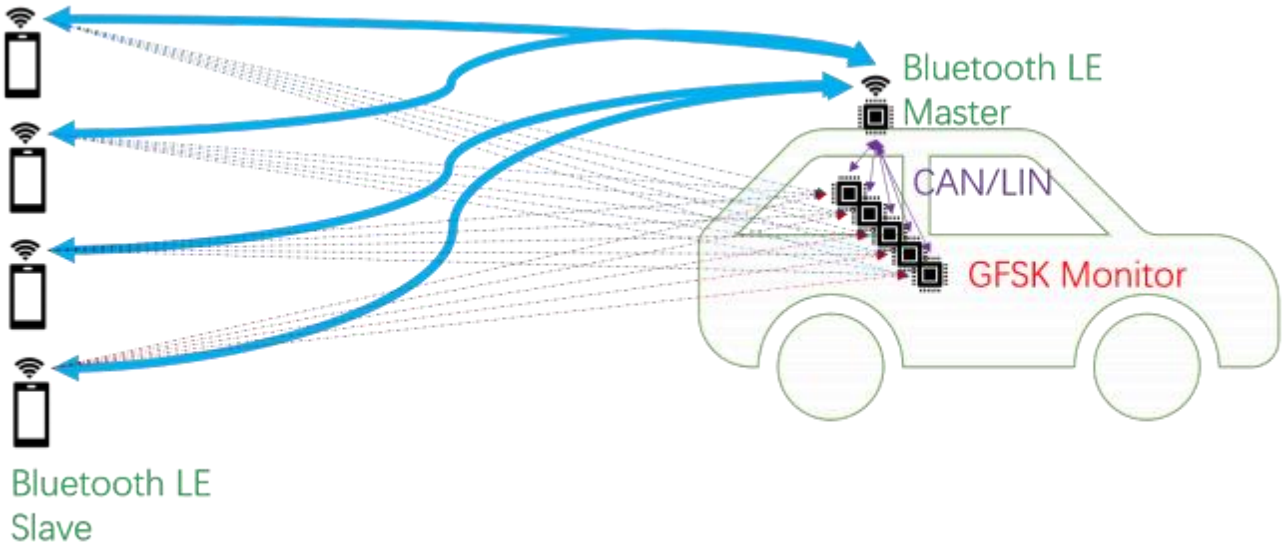


图 5. 监控多个连接

对于单链路的监控，我们可以在一个固定信道上等待有效的数据包。一旦接收到数据，就可以根据跳频算法追踪下一个数据。连接间隔简单，但多链路监控比较复杂，因为涉及到链路管理。

图 6 是一个主机连接到四个从机的即时时序图。以S1为锚点，该设备与下一个设备的间隔固定为33个时隙（约20毫秒）。

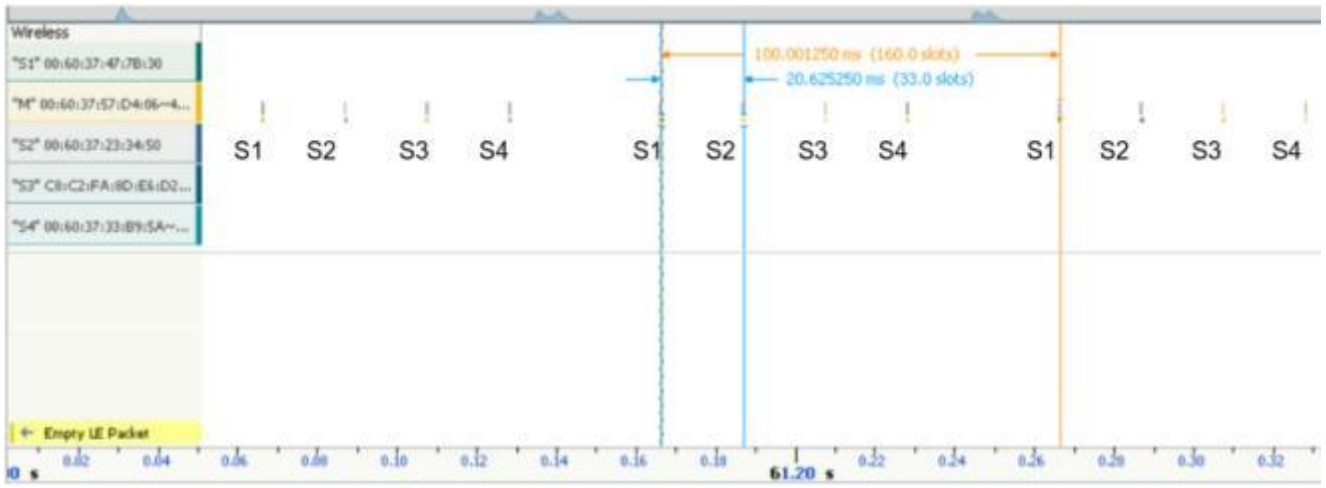
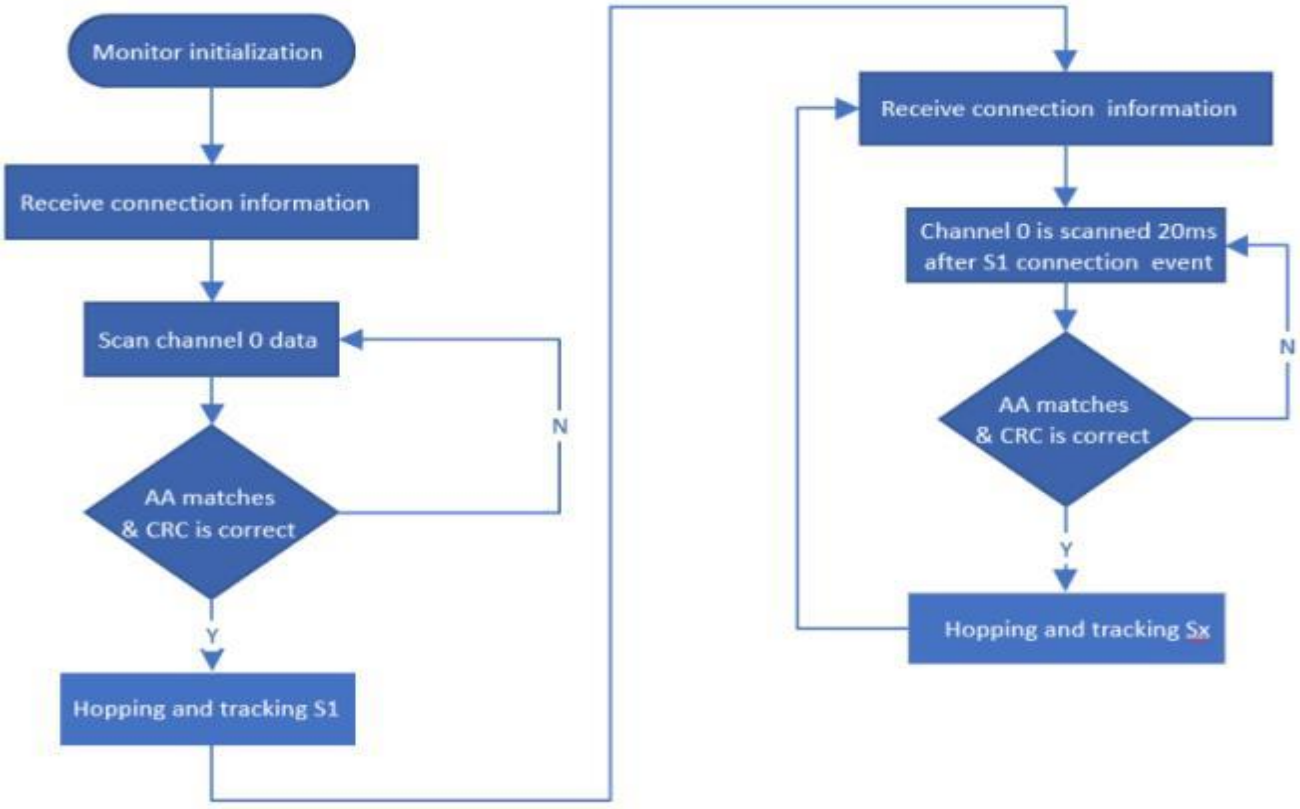


图 6. 4链路连接即时时序

我们可以通过监听固定信道来同步S1设备，然后以S1为锚点，延长20ms以同步其他设备。图 7 是操作流程图。



4 设置和结果

如图 8所示，为了简化设置，我们将监控的数量减少到一个。该演示包含一个主机、一个监控和四个从机。主机和从机通过蓝牙 LE 连接。 主机和监控通过 CAN 或 LIN 总线连接。

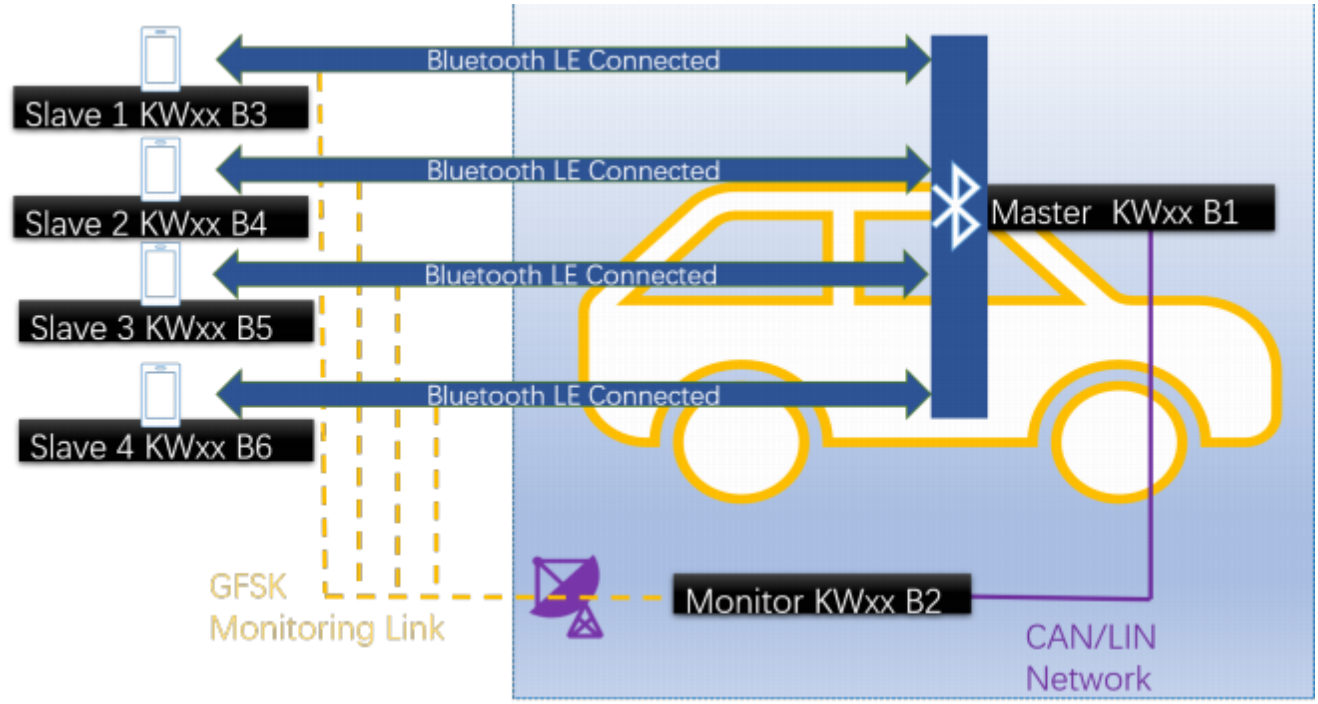


图 8. 设置

4.1 硬件需求

- 六根Mini/Micro USB 电缆。
- 六块 FRDM-KW36 或 FRDM-KW38 板。板子简称为 B。

— B1作为主机。

— B2作为监控。

— B3-B6作为从机。

- 个人电脑。
- 12 V电源适配器。
- 三根母对母杜邦线。

4.2 电路板设置

- 将 12 V 适配器连接到板 B1 或 B2 的 J32。
- 卸下 B2 板的 R34 和 R27 电阻。
- 连接板B1和B2的J13-1。
- 连接板B1和B2的J13-2。
- 连接板B1和B2的J13-4。

注意

使用自动波特率功能时，连接 J1-5 和 J2-9。

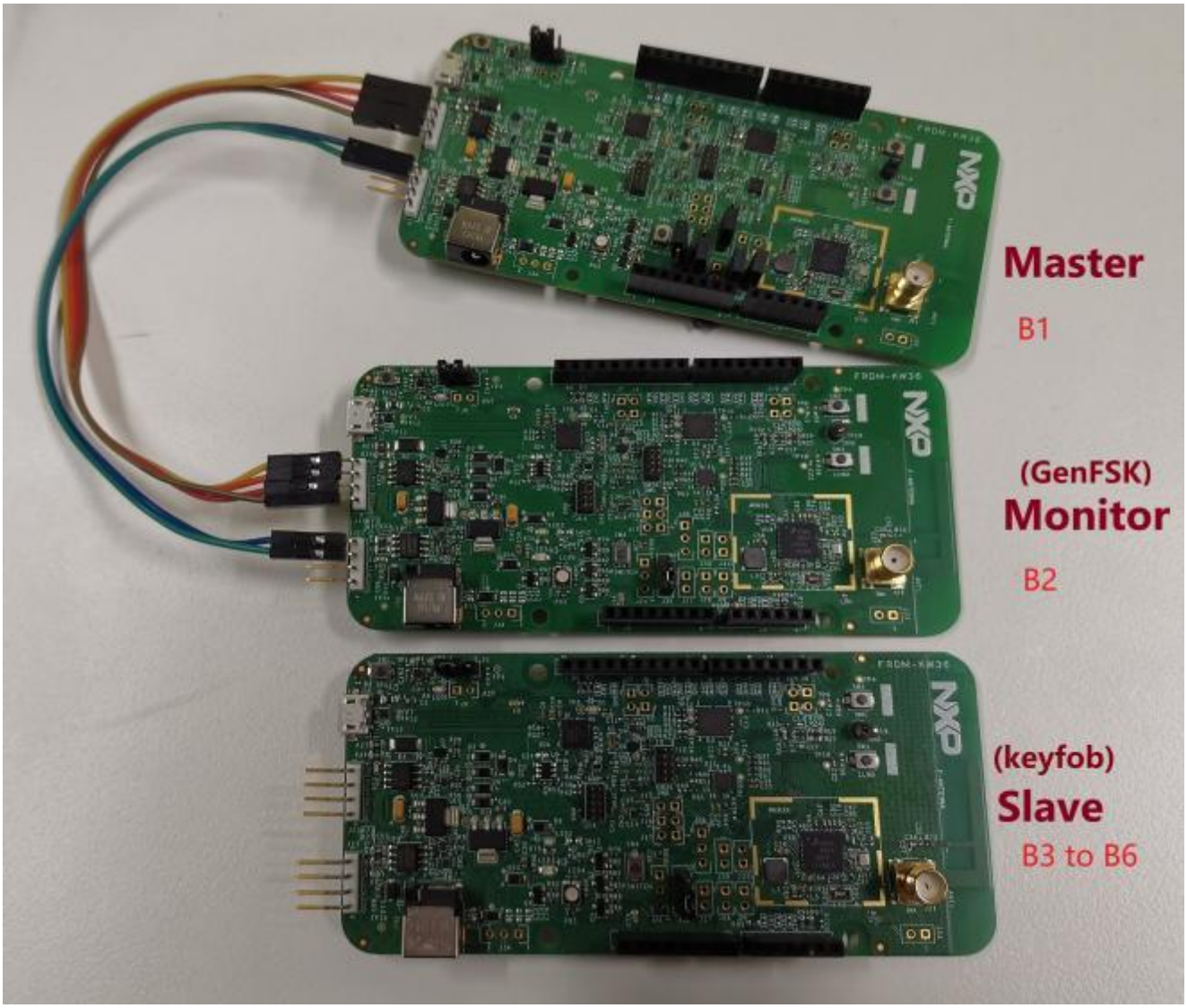


图 9. 具有四个从机的链路监控设置

4.3 支持工具链

- IAR Embedded Workbench 8.40.1

4.4 软件需求

此链路监控演示基于用FRDM-KW36 或 FRDM-KW38 实现的无线 UART 演示方案和 genfsk 演示方案，可在AN12872SW中找到。在此应用中，有三个角色，主机、从机和监控。

- 主机工程是从 wireless_uart 工程修改而来的。
- 监控工程是从 genfsk 工程修改而来的。
- 从机工程是 wireless_uart 工程。

在主机和监控工程中，我们增加了CAN和LIN的功能，实现了连接信息的传输。

有两种评估此应用的方法，如下所述。

- 直接使用移植好的SDK。

  为了快速评估，AN12872SW包含了移植好的 SDK，SDK_2.2.3_FRDM-KW36 和SDK_2.6.5_FRDM-KW38， 如图10所示。

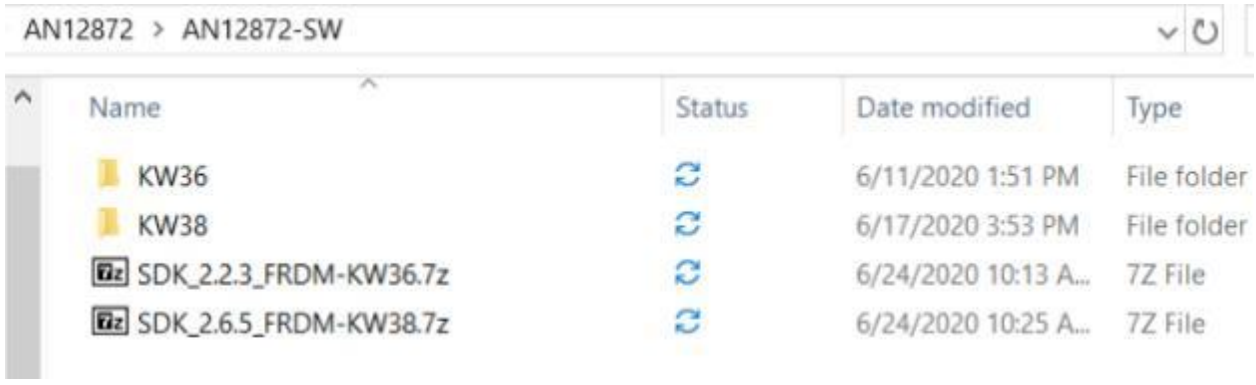


图 10. 使用受保护的 SDK 移植代码

—w_uart_linkMonitor 工程– 对应于B1作主机， 位于 SDK\boards\frdmkw36\wireless_examples\bluetooth\w_uart_linkMonitor\freertos。

—conn_test_linkMonitor 工程 – 对应于 B2作监控， 位于 SDK\boards\frdmkw36\wireless_examples\genfsk\conn_test_linkMonitor\freertos。

—w_uart 工程 – 对应于B3-B6 作从机，位于 SDK\boards\frdmkw36\wireless_examples\bluetooth\w_uart\freertos.

- 将代码移植到 SDK。

  AN12872 软件包还包含源文件和头文件，如图 11所示。

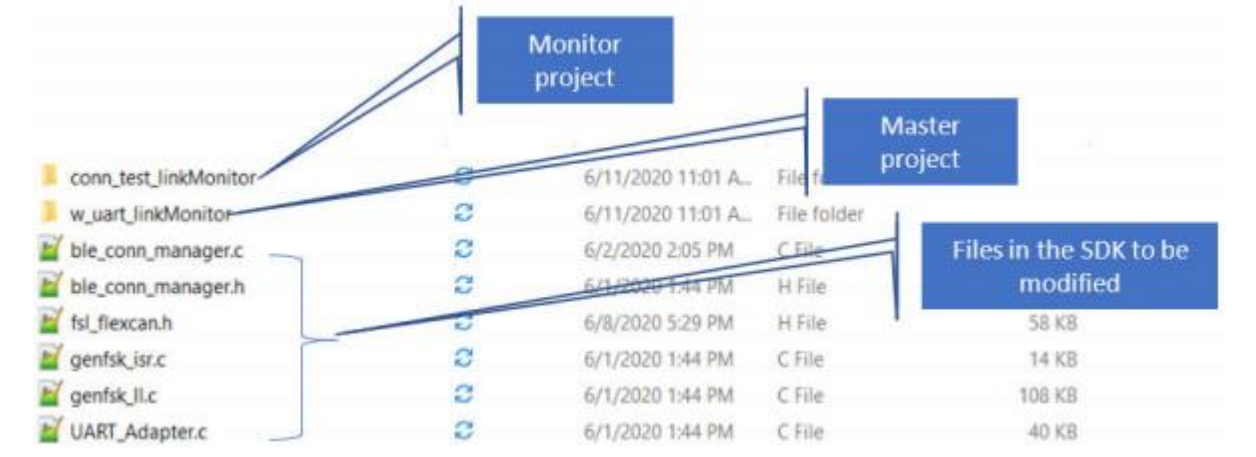


图 11. 自己移植代码

执行以下步骤，在 IAR Embedded Workbench 中创建新的演示项目，并替换为此演示方案而修改的源文件和头文件。

1. 将 conn_test_linkMonitor放到 SDK\boards\frdmkw3x\wireless_examples\genfsk目录下。
2. 将 w_uart_linkMonitor放到SDK\boards\frdmkw3x\wireless_examples\bluetooth目录下。
3. 替换以下目录中的五个文件：

— middleware/wireless/bluetooth_x.x.x/application/common/ble_conn_manager.c

— middleware/wireless/bluetooth_x.x.x/application/common/ble_conn_manager.h

— middleware/wireless/framework_x.x.x/SerialManager/Source/UART_Adapter.c

— middleware/wireless/genfsk_x.x.x/source/genfsk_isr.c

— middleware/wireless/genfsk_x.x.x/source/genfsk_ll.c

注意

上面创建的工程基于 KW36 的 SDK 2.2.3（2020-04-30 发布），KW38 的 SDK 2.6.5（2020-05-07 发布）。如果您使用的是其他版本的SDK，请相对应地修改工程文件。

4.5 测试方法

1. 将程序下载到目标板。
2. 在 PC 和板 B1 和 B2 上的 OpenSDA USB 端口之间连接micro USB 电缆。
3. 使用以下设置在 PC 上为 OpenSDA 串行设备打开串行终端：
   - 115200波特率
   - 8个数据位
   - 无奇偶校验
   - 1个停止位
   - 无流量控制
4. B3-B6上电，切换角色，开始广播。
5. 按 B1 板上的 **SW2** 开始扫描、连接和监控。

4.6 测试结果

图 12 显示测试结果，包括主机和从机的链路数据和RSSI值。 正如预期的那样，通过 GFSK 监控能看到所有的信息。

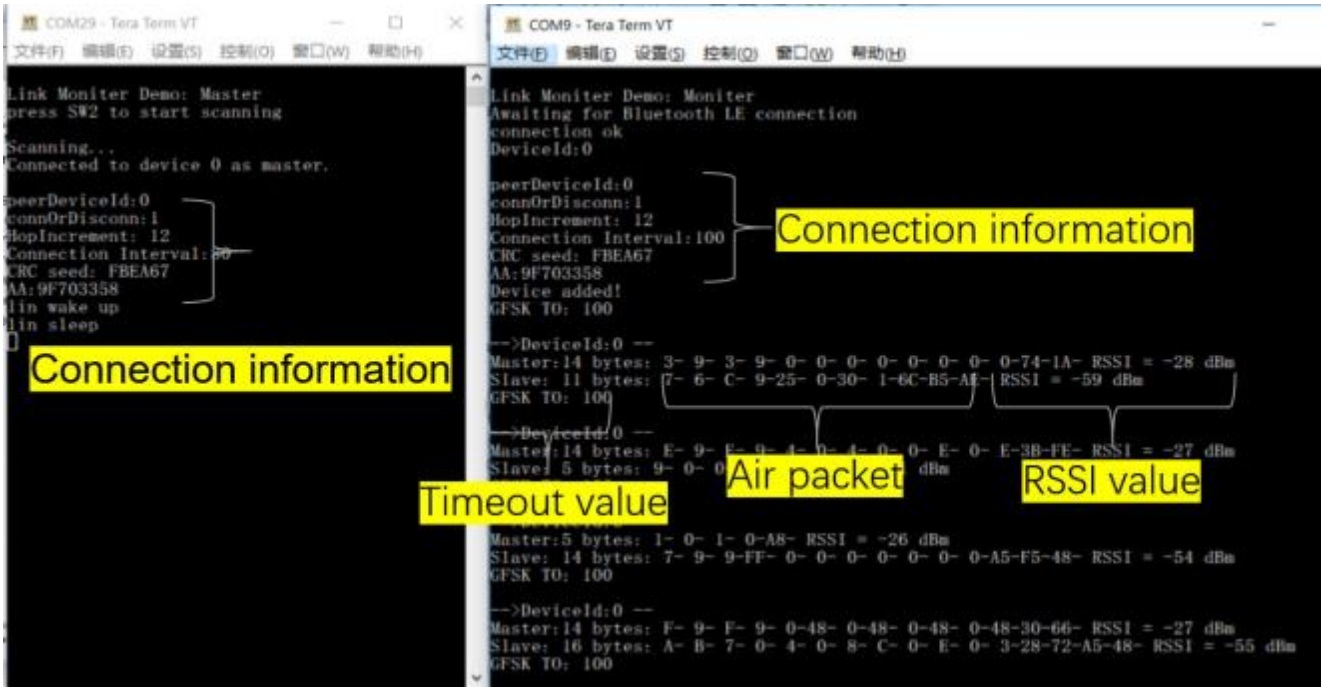


图 12. 测试结果

5 结论

通过GFSK，我们可以有效监控蓝牙LE的连接，并结合KW36/38的高精度RSSI检测能力。 通过KW36或KW38实现反中继攻击的测距是可行且有效的。

6 修订史

表 2. 修订史

版本号	日期	描述
0	07/2020	初稿
1	09/2020	• 更新了图1，图2，图3，图9和图12 • 增加了图5和图8

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer’s applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 09/2020
Document identifier: AN12872

arm