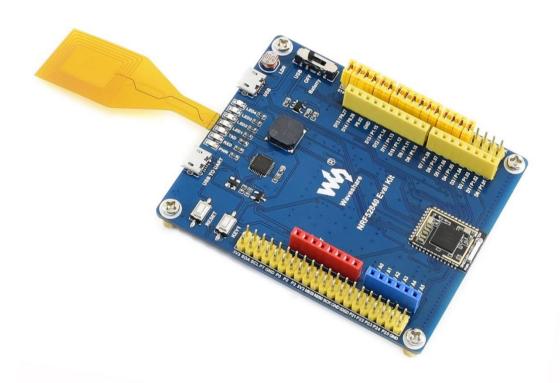
N







CONTENT

Chapter 1. Overview	5
what's on Board	5
Chapter 2. Setup Developing Environment	7
Install Keil and related Pack	7
Install SEGGER Embedded Studio for ARM 4.10a	7
Serial driver install	7
Install Jlink driver	8
Install nRFx-Command-Line-Tools	9
Install APP	9
Peripherals demo	10
Hardware connection	10
Download samples without Softdevice	11
Download Samples with Softdevoce	13
Chapter 3. Creating new project	15
Project template	15
Create a new project	16
Chapter 4. Controling LED	20
Codes	20
Hardware Connection	21
Chapter 5. UART	23
Codes	23

Hardware	27
Chapter 6. SPI	29
Loopback testing	29
Drive OLED	
Hardware	
Chapter 7. I2C	
Scanning Slave devices	
Read data from BME280	
Read data from MPU6050	
Hardware	
Chapter 8. NFC	41
Codes	41
Hardware	43
Chapter 9. Bluetooth	44
Preparation	44
Install softdevice	44
Bluetooth serial transparent	44
Bluetooth against losing	47
Bluetooth Beacon (Support WeChat Shake)	49
Wireless Mouse	52
Wireless Keyboard	53
CSCS	55

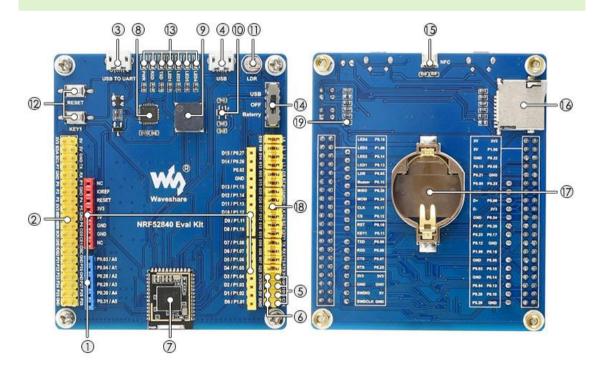
Bluetooth Master- Cardiotachometer	56
Master-Slaver-BLE Relay	
Chapter 10. MESH	62

CHAPTER 1. OVERVIEW

NRF52840 Eval Kit is a development/evaluation kit based on nRF52840, features Arduino / Raspberry Pi connectivity, and several common peripherals.

It supports the new generation Bluetooth 5.0, compared to Bluetooth 4.0, the data rate is twice, up to 2Mbps, the transfer distance is quadruple. Based on the ARM Cortex-M4F architecture, the CPU frequency achieves up to 64MHz. Bluetooth MESH¹ and 2.4GHz radio are also available, that means higher performance, and more suitable applications.

Thanks to the onboard Arduino / Raspberry Pi compatible headers, both Arduino shields and Raspberry Pi HATs can be attached at the same time, make it easy to use more expansion resources.



WHAT' S ON BOARD

¹ At least three NRF52849 modules are required for Bluetooth MESH networking

WAVESHARE

- Arduino headers: for connecting Arduino shields
- 2. Raspberry Pi GPIO header: for

connecting Raspberry Pi HATs

- 3. USB TO UART interface
- 4. USB port: the USB port of nRF52840
- 5. 4PIN SWD debugging interface:

for connecting ARM Debugger to

program / debug

6. 3.3V/5V power input/output:

provides power output OR powered from external power supply

- 7. Core52840: nRF52840 core module
- 8. CP2102: USB TO UART converter
- 9. Buzzer
- **10. RT9193-33:** 3.3V voltage regulator
- 11. Optical sensor
- **12. Keys:** RESET and USER KEY
- 13. LEDs

PWR: power indicator

RXD/TXD: nRF52840 UART RX/TX

indicator

LED1~LED4: USER LEDs

14. Power switch

USB: powered from USB connection

Battery: powered from CR2032

battery

15. NFC antenna connector

- 16. TF card slot
- 17. CR2032 battery holder

18. Jumpers

Short the jumpers: connect the

onboard peripherals to I/Os in

example code

Open the jumpers: connect to

custom I/Os via jumper wires

19. Current testing pads: 0Ω resistor is

soldered by default, remove the

resistor to test current of the board

CHAPTER 2. SETUP DEVELOPING ENVIRONMENT

Generally, we use nRFgo Studio software to programming NRF52840, however,

because of the compatibility of JLink driver, which is the driver of programmer of nRF

series. Herein we use CLI to setup developing environment for nRF52840 instead of

nRFgo Studio.

INSTALL KEIL AND RELATED PACK

Download Keil from its website and install. Note that Keil requires license,

otherwise it can only be used to compile codes which is smaller than 256kB.

After installing, you should also install related Packs. The NRF52840 pack was

already included in resources², run NordicSemiconductor.nRF_DeviceFamilyPack.8.17.0

to install it.

INSTALL SEGGER EMBEDDED STUDIO FOR ARM 4.10A

This tool is only required for Bluetooth MESH examples. You can skip this part if you do not use MESH.

You can download the package from resources and install it without license.

SERIAL DRIVER INSTALL

The Evaluation board has integrated CP2102 module, which is used for TTL to USB converting. To properly use serial port, you need to install CP2102 driver in your PC. If you have installed it before, skip this part.

The CP2102 driver is included in resources, choose one according to your OS.

² You can find them on Waveshare Wiki: <u>https://www.waveshare.com/wiki/NRF52840_Eval_Kit</u>

- Double-click installer "CPCP210xVCPInstaller_xxx" to run it
- Clock next to finish installing

After installing, you can find that a new serial device is recognized under

Computer->Manager->Devices Manager-> COM and LPT after you connect the

evaluation board to PC.

INSTALL JLINK DRIVER

JLink programmer is used to programming NRF52840. It can also be used to erase

NRF52840Flash and upload Bluetooth Softdevices³ with nRF5x-Command-Line-Tools.

We recommend you use V6.22 version or above, otherwise, the driver may cannot be

recognized. The JLink driver is included in resources, you can search and download it.

If you have installed Keil before JLink driver, don't forget to check the option to

cover it as below:

		IDLL V6.00h in "C:\Program Files\IAR Systems\Embedded Workbench	h 7.5\ARM\bin'')	
(eil MDK V5	i.18a (DLL V5.12f in "C:\	Kei_v5\ARM\Segger")		
Select All	Select None			
nes s ve	you would like to replace	d kept in the same folder, allowing manual "undo".		

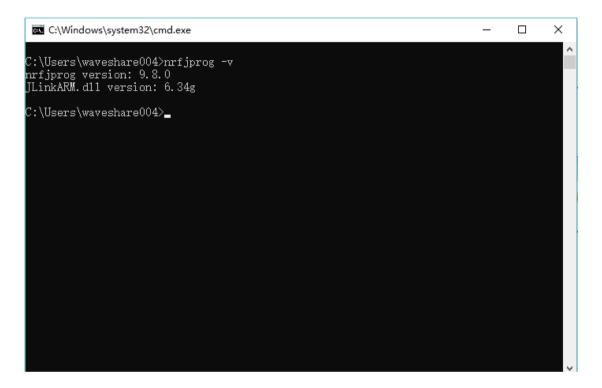
³ Softdevices are protocol stacks for nRF series

INSTALL NRFX-COMMAND-LINE-TOOLS

The tool can be downloaded from resources. Download and install it. After

installing, open console, and input nrfjprog -v to check the tools. It should be as below

if you install the tool successfully.



INSTALL APP

App should be installed in mobile phones for debugging.

nRF Connect: Common debugging tool of mobile phone. With this APP, use can get the original data. nRF Connect can display RSSI curve, supports multiply slave devices, has high compatibility and is commonly used.

nRF Toolbox: BLE tool. Toolbox is generally used for part of experiments, it has GUI like thermometer, cardiotachometer. However, it can only be used for some of experiments and it is not much stable.

PERIPHERALS DEMO

Finish the developing environment setup, you now can try with some demo. Herein

using the newest SDK(SDK 15.2), which support Bluetooth 5.0

There are examples in SDK, which can be divided to two kinds, one of them can

run without Softdevice and another are based on Softdevice. The demo which does

not need Softdevice generally located in

nRF5_SDK_15.2.0_9412b96\examples\peripheral\

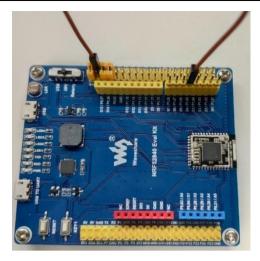
	blinky	2018/9/8 14:47	文件夹
	blinky_freertos	2018/9/8 14:48	文件夹
	blinky_rtc_freertos	2018/9/8 14:48	文件夹
	blinky_systick	2018/9/8 14:49	文件夹
	bsp	2018/9/8 14:49	文件夹
	cli	2018/9/8 14:50	文件夹
	csense	2018/9/8 14:51	文件夹
	csense_drv	2018/9/8 14:52	文件夹
	experimental_cli_libuarte	2018/9/8 12:06	文件夹
	experimental_libuarte	2018/9/8 12:06	文件夹
	fatfs	2018/9/8 14:53	文件夹
	flash_fds	2018/9/8 12:06	文件夹
	flash_fstorage	2018/9/8 12:06	文件夹
	flashwrite	2018/9/8 14:53	文件夹
	fpu_fft	2018/9/8 14:54	文件夹
	gfx	2018/9/8 14:56	文件夹
	gpiote	2018/9/8 14:56	文件夹
	i2s	2018/9/8 14:57	文件夹
	led_softblink	2018/9/8 14:57	文件夹
	low_power_pwm	2018/9/8 14:57	文件夹
	lpcomp	2018/9/8 14:58	文件夹
	nrfx_spim	2018/9/8 12:06	文件夹
_			

HARDWARE CONNECTION

- Connect J-Link programmer to NRF52840 Eval kit and PC
- Connect NRF52840 Eval Kit to PC or other external power adapter
- Pull the jumper of LED3, and wire L3 to P1.05⁴

⁴ LED3 isn' t compatible with official SDK, so we should re-connect it when test Blinky sample





DOWNLOAD SAMPLES WITHOUT SOFTDEVICE

We take Blinky as example, learn how to use peripherals examples of SDK. Click

blinky->pca10056->blank->arm5_no_packs->blink_pca10056.uvoptx to open the

project. Compiling it:



Generally, it can be compiled successfully without error or warning. After

compiling, you should configure it:



Click Debug, choose J-Links/J-Trace Cortex as debugger. Debug->Settings set

Port to SW

NRF52840 Eval Kit User Manual

WUAVESHARE

🔣 Options for Target 'nrf52840_xxaa'	×
Device Target Output Listing User C/C++ A	Asm Linker Debug Utilities
C Use Simulator <u>with restrictions</u> Settings	C Use J-LINK / J-TRACE Cortex Settings
Image: Initialization File: Image:	Load Application at Startup Run to main() Initialization File:
Restore Debug Session Settings	Restore Debug Session Settings
Breakpoints Toolbox	Breakpoints Toolbox
Watch Windows & Performance Analyzer	Watch Windows
Memory Display Viswer	Very Memory Display Very System Viewer
CPU DLL: Parameter:	Driver DLL: Parameter:
	SARMCM3.DLL -MPU
Dialog DLL: Parameter:	Dialog DLL: Parameter:
	TCM.DLL pCM4
Warn if outdated Executable is loaded	Warn if outdated Executable is loaded
Manage Component Vie	swer Description Files
OK Car	cel Defaults Melp

rtex JLink/JTrace Target Driver Setup			:
ebug Trace Flash Download J-Link / J-Trace Adapter	SW Device		
SN: 20160605	IDCODE	Device Name	Move
Device: J-Link ARM-OB STM32	SWDI Ox2BA01477	ARM CoreSight SW-DP	Up
HW : V7.00 dll : V6.30h			
			Down
	Automatic Detection	ID CODE:	
Port: Max			
SW 🔹 5 MHz 💌	C Manual Configuration	Device Name:	
Auto Clk	Add Delete Up	odate IR len:	
Connect & Reset Options Connect: Normal Reset: No Reset: No Reset after Connect	ormal 👻 🔽 🤇	he Options Download C Cache <u>C</u> ode Verify Co Cache <u>M</u> emory Downloa	de Download
Interface TCP/IP		1	Misc
Interface TCP/IP © USB © TCP/IP IP-Address	•		/lisc JLink Info
© USB ○ TCP/IP Network S	9 Port (A	Auto: Autodetect	JLink Info
Scan Network S IP-Address	Port (A	Auto:	
© USB ○ TCP/IP Network S	Port (A	Auto: Autodetect	JLink Info

Then click Flash Download, check the options as below and confirm by clicking

OK.

NRF52840 Eval Kit User Manual

Download Function LOAD C Erase Full Chi C Erase Sectors	P. I Program F Verify	Start: 0x20000000 Size: 0x2000	
O Do not Erase	✓ Reset an	nd Run	
Programming Algorithm	,		
Description	Device Size	Device Type Address Range	
nRF52xxx nRF52xxx UICR	2M 4k	On-chip Flash 00000000H - 001FFFFH On-chip Flash 10001000H - 10001FFFH	
		Start: Size:	
	Ac	d Remove	
	Ac	d Remove	
	Ac	ld Remove	
	Ac	id Remove	

DOWNLOAD SAMPLES WITH SOFTDEVOCE

Different with samples like the blinky, you should first download softdevices,

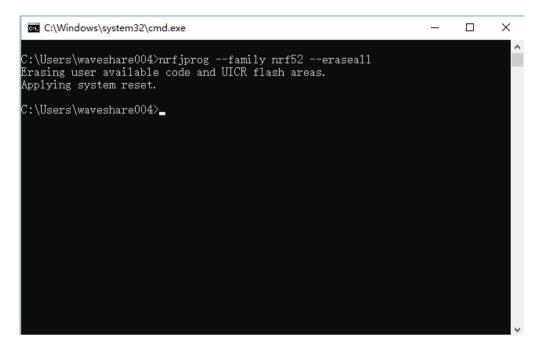
which Bluetooth demo codes are based on. Herein we take app_uart as example to

show you how to program such demos.

• First, we need to erase NRF52840 flash: Open CMD, connect NRF52840 to PC and using ARM Debugger(JLink) for programming. Type command to erase NRF52840

WAVESHARE

as below:

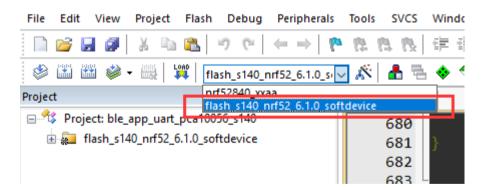


• Download Softdevice. Click

nRF5_SDK_15.2.0_9412b96->examples->ble_peripheral->ble_app_uart->pca10056

->s140->arm5_no_packs-> ble_app_uart_pca10056_s140.uvprojx to open the

project. Choose flash s140 nrf52 6.1.0 softdevice:



• Download the application code like blinky sample.

CHAPTER 3. CREATING NEW PROJECT

SDK should be used for NRF52840 developing, we take common SDK⁵ as example

to show you how to create a project.

The SDK used herein is SDK V15.2, there is also template for reference:

nRF5_SDK_15.2.0_9412b96\examples\ble_peripheral\ble_app_template\pca10056\s140

\arm5_no_packs\ble_app_template_pca10056_s140.uvprojx

PROJECT TEMPLATE

Generally, the project should include:



• DRIVER: Drivers of hardware peripherals are saved in this directory, like driver of

MPU6050 created by users

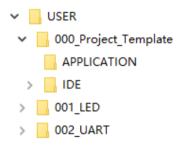
• SDK: SDK files copied from official SDK, like:



⁵ There are specific SDK used for Zigbee, Thread and Mesh sevices

There are so many files included in SDK, and we only use some of them for certain project. Most of libraries are included in components, external contains third party libraries like freertos library and so on. Old drivers are saved to folder integration and modules contains the new drivers.

- TEMP: Intermediate files generated while compiling. The files in this folder can be deleted.
- USER: User files



Generally, files in folders SDK, DRIVER and TEMP are same for different project, what

we need to do is create user files and save them to USER.

CREATE A NEW PROJECT

Create a directory to save project files and copy libraries files to it.

You can google about how to create a new Keil project. Herein we take about

some notices when creating new project.

• The drivers of user peripherals should be saved in DRIVER folder

	Part-1-Peripheral > DRIVER				
│ 名称	修改日期	类型	大小		
BME280	2018/11/29 14:58	文件夹			
📄 iic.c	2018/11/23 19:36	C Source File	4 KB		
b iic.h	2018/11/23 19:37	C++ Header file	2 KB		
📄 led.c	2018/10/26 17:40	C Source File	1 KB		
ы led.h	2018/10/26 17:34	C++ Header file	1 KB		
📄 три6050.с	2018/11/10 16:47	C Source File	13 KB		
<u>ы</u> mpu6050.h	2018/11/10 14:14	C++ Header file	15 KB		
oledfont.c	2018/11/7 15:25	C Source File	41 KB		
b oledfont.h	2018/11/7 15:29	C++ Header file	1 KB		
📄 spi.c	2018/11/7 11:39	C Source File	1 KB		
ы spi.h	2018/11/7 11:39	C++ Header file	1 KB		
📄 ssd1306.c	2018/11/23 19:38	C Source File	8 KB		
b ssd1306.h	2018/11/23 19:38	C++ Header file	1 KB		
uart.c	2018/10/29 14:49	C Source File	1 KB		
▶ uart.h	2018/10/29 14:49	C++ Header file	1 KB		

• The projects created should be saved under USER, and please notice the project

name

	Part-1-Pe	ripheral > US	SER
│ 名称	修改日期	类型	大小
000_Project_Template	2018/11/29 15:09	文件夹	
001_LED	2018/11/29 15:09	文件夹	
002_UART	2018/11/29 15:09	文件夹	
003_SPI(Example)	2018/11/29 15:09	文件夹	
004_SPI(OLED)	2018/11/29 15:09	文件夹	
005_I2C(Scan Device)	2018/11/29 15:09	文件夹	

• The application files are usually saved under the path: USER/[Project

file]/APPLICATION/

Part-1-Peripheral > USER > 000_Project_Template > APPLICATION			
│ 名称	修改日期	类型	大小
3D_Cube.c	2018/11/10 18:15	C Source File	5 KB
🖻 3D_Cube.h	2018/11/10 18:15	C++ Header file	1 KB
h include.h	2018/11/10 14:11	C++ Header file	2 KB
📄 main.c	2018/11/27 19:08	C Source File	2 KB
mymath.c	2018/11/10 18:02	C Source File	8 KB
🖻 mymath.h	2018/11/10 18:02	C++ Header file	1 KB
🖻 sdk_config.h	2018/9/8 15:31	C++ Header file	64 KB

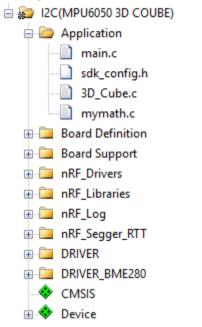
• The Keil project file is saved in IDE folder

>	Part-1-Peripheral →	USER > 000_Proje	ct_Template → IDE
│ 名称	修改日期	类型	大小
Listings	2018/11/29 15:09	文件夹	
RTE	2018/11/29 15:09	文件夹	
JLinkLog.txt	2018/11/29 15:34	文本文档	6 KB
📓 JLinkSettings.ini	2018/11/16 15:13	配置设置	1 KB
📄 Project_Template.uvguix.huangrui	2018/11/29 15:34	HUANGRUIMIN	88 KB
Project_Template.uvoptx	2018/11/29 15:34	UVOPTX 文件	14 KB
🞇 Project_Template.uvprojx	2018/11/27 19:11	礣ision5 Project	23 KB

• Save OBJ files to TEMP folder

🕅 Options for Target 'Template'	Х
Device Target Output Listing User C/C++ Asm Linker Debug Utilities	
Select Folder for Objects Name of Executable: 000_Project-Template	
Create Executable:	
✓ Debug Information □ Create Batch File	
Create HEX File	
✓ Browse Information	
C Create Library:\\TEMP\000_Project-Template.lib	
OK Cancel Defaults Help	

- You had better set the project tree same as official projects.
 - Project: I2C(MPU6050 3D COUBE)



• Notice the PACK version you use, make sure it is same as official examples

	Packs	device package	
Pack	Selection Version	Description	
ARM::AMP	excluded \lor	Software components for inter processor communication (Asymmetric Multi	
ARM::CMSIS	fixed v 4.5.0	CMSIS (Cortex Microcontroller Software Interface Standard)	
5.3.0			
4.5.0		Select 4.5 and set the version fixed	
4.3.0			
ARM::CMSIS-Driver	excluded \checkmark	CMSIS Drivers for external devices	
Hitex::CMSIS_RTOS_Tutorial	excluded 🗸	An Introduction to using CMSIS RTOS for Cortex-M Microcontrollers	
Keil::ARM_Compiler	excluded 🗸	Keil ARM Compiler extensions for ARM Compiler 5 and ARM Compiler 6	
Keil::MDK-Middleware	excluded 🗸	Middleware for Keil MDK-Professional and MDK-Plus	
Keil::STM32F0xx_DFP	excluded \lor	STMicroelectronics STM32F0 Series Device Support, Drivers and Examples	
Keil::STM32F1xx_DFP	excluded \checkmark	STMicroelectronics STM32F1 Series Device Support, Drivers and Examples	
Keil::STM32F4xx_DFP	excluded \checkmark	STMicroelectronics STM32F4 Series Device Support, Drivers and Examples	
Keil::STM32L4xx_DFP	excluded \lor	STMicroelectronics STM32L4 Series Device Support, Drivers and Examples	
Keil::STM32NUCLEO_BSP	excluded \checkmark	STMicroelectronics Nucleo Boards Support and Examples	
-NordicSemiconductor::nRE_DeviceFam	nil fixed ¥ 8.17.0	Nordic Semiconductor nRF ARM devices Device Family Pack.	
	✓ 3		
8.9.0		Selects 8.17.0 and sets the version fixed	

For more details, please refer to official examples.

CHAPTER 4. CONTROLING LED

In this chapter, we will instruct how to configure GPIO and control LED. The

examples used herein is 001_LED.

CODES

Open led.h files, you can find that the GPIO and related control functions are defined

in this header file:

```
#ifndef __LED_H__
#define __LED_H__
#include "nrf.h"
#include "nrf_gpio.h"
#define LED0 NRF_GPIO_PIN_MAP(0,13)
#define LED1 NRF_GPIO_PIN_MAP(0,14)
#define LED2 NRF_GPIO_PIN_MAP(0,19)
#define LED3 NRF_GPIO_PIN_MAP(0,16)
void LED_On(uint32_t led_number);
void LED_Off(uint32_t led_number);
#endif
```

Functions defined in header file will be realized in led.c file:

```
#include "led.h"
void LED_On(uint32_t led_number)
{
    nrf_gpio_cfg_output(led_number);
    nrf_gpio_pin_clear(led_number);
}
void LED_Off(uint32_t led_number)
{
    nrf_gpio_cfg_output(led_number);
    nrf_gpio_pin_set(led_number);
}
```

LED_On(led_number): Turn the LED on

LED_Off(led_number): Turn the LED off

And finally, the functions are called in main file to keep turning four LEDs on for

500ms then turning them off for 500ms.

```
int main(void)
{
   while (true)
   {
      LED On (LED0);
      LED On (LED1);
      LED On (LED2);
      LED On (LED3);
      nrf delay ms(500);
      LED Off(LED0);
      LED Off(LED1);
      LED Off(LED2);
      LED Off(LED3);
      nrf delay ms(500);
   }
}
```

HARDWARE CONNECTION

According to the codes above we know that, the four LEDs controlled should be

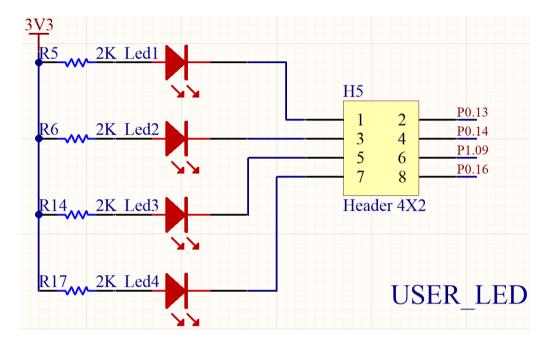
connected to P0.13, P0, 14, P0.19, P0.16 separately as table:

UART	GPIO
LED1	P0.13
LED2	P0.14
LED3	P1.09
LED4	P0.16

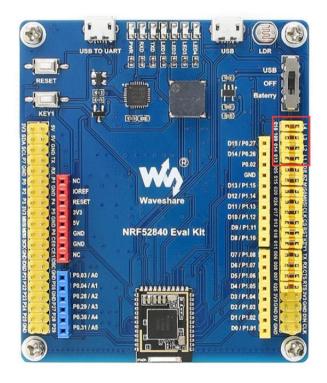
In official SDK, LED3 is connected to P0.15 which is not compatible with official

examples. If you want to use the official SDK and LED3, you need to change it to P1.09.

The schematic of Eval board is as below:



Header 4x2 are jumpers on board:



If you use official SDK, don't forget to pull L3 jumper and wire it to P0.15 (Do not require if you use 001_LED).

CHAPTER 5. UART

In this chapter, we describe about how to use UART interface of NRF52840. The example used herein is 003_UART. With the example, data can be sent from NRF52840 to UART interface, and LED on eval board will turn on/off when receive corresponding data from UART interface.

CODES

```
uart.h
#ifndef UART H
#define _UART_H_
#include "nrf.h"
#include "nrf gpio.h"
#include "app uart.h"
#include "nrf uart.h"
#define RX PIN NUMBER NRF GPIO PIN MAP(0,8)
#define TX PIN NUMBER NRF GPIO PIN MAP(0,6)
#define CTS PIN NUMBER NRF GPIO PIN MAP(0,7)
#define RTS PIN NUMBER NRF GPIO PIN MAP(0,5)
#define UART HWFC APP UART FLOW CONTROL DISABLED
#define MAX TEST DATA BYTES
                              (15U)
#define UART TX BUF SIZE 256
#define UART RX BUF SIZE 256
extern app_uart_comm_params_t comm_params;
void uart error handle(app uart evt t * p event);
#endif
uart.c:
#include "uart.h"
```

```
app uart comm params t comm params =
£
   RX PIN NUMBER,
  TX PIN NUMBER,
  RTS PIN NUMBER,
   CTS PIN NUMBER,
   UART HWFC,
  false,
   NRF UART BAUDRATE 115200
};
void uart error handle(app uart evt t * p event)
{
   if (p event->evt type == APP UART COMMUNICATION ERROR)
   {
      APP ERROR HANDLER (p event->data.error communication);
   }
   else if (p event->evt type == APP UART FIFO ERROR)
   {
      APP_ERROR_HANDLER(p_event->data.error_code);
   }
}
main.c:
#include "nrf.h"
#include <stdio.h>
#include <stdint.h>
#include <stdbool.h>
#include "app uart.h"
#include "app error.h"
#include "nrf delay.h"
```

```
#include "nrf uart.h"
#include "nrf uarte.h"
```

#include "led.h" #include "uart.h"

/**

```
* @brief Function for main application entry.
```

```
*/
int main(void)
ł
```

uint32 t err code;

WAVESHARE

```
APP UART FIFO INIT (& comm params, UART RX BUF SIZE, UART TX BUF SI
ZE,uart error handle,APP IRQ PRIORITY LOWEST,err code);
APP ERROR CHECK (err code);
printf("\r\nUART example started.\r\n");
while (true)
{
   uint8 t cr;
   //Get Data from the port!
   while (app_uart_get(&cr) != NRF_SUCCESS);
   //Then put data to the port
   while (app uart put(cr) != NRF SUCCESS);
   switch (cr)
   {
       case '1':
       £
          printf("\r\n LED0 ON \r\n");
          LED On (LED0);
          break;
       }
       case '2':
       {
          printf("\r\n LED0 OFF \r\n");
          LED Off(LED0);
          break;
       }
       case '3':
       {
          printf("\r\n LED1 ON \r\n");
          LED On (LED1);
         break;
       }
       case '4':
       {
          printf("\r\n LED1 OFF \r\n");
          LED_Off(LED1);
          break;
       }
       case '5':
```

```
{
             printf("\r\n LED2 ON \r\n");
             LED On (LED2);
             break;
          }
          case '6':
          {
             printf("\r\n LED2 OFF \r\n");
             LED Off(LED2);
             break;
          }
          case '7':
          {
             printf("\r\n LED3 ON \r\n");
             LED_On(LED3);
             break;
          }
          case '8':
          {
             printf("\r\n LED3 OFF \r\n");
             LED Off(LED3);
             break;
          }
          case 'q':
          {
             printf("\r\n Exit! \r\n");
             while (true);
          }
          case 'Q':
          {
             printf("\r\n Exit! \r\n");
             while (true);
          }
          default:
          {
             printf("\r\nPlease input correct command\r\n");
             break;
          }
      }
   }
}
```

The serial is redirected in the codes, for using printf function to print data to uart

interface. In the main.c, serial port will keep receiving data, and if corresponding data

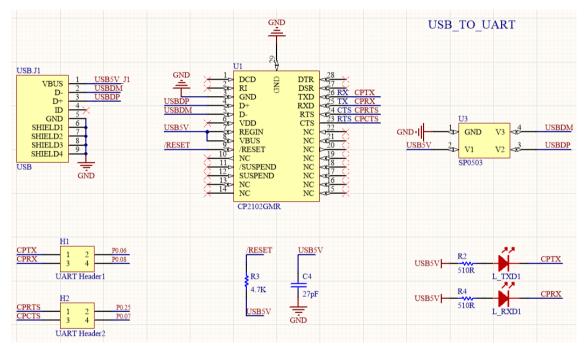
are received, LEDs will behavior according to it.

When 1 is received, LED1 is on, and turned off if receiving 2. Turn LED2 on if 3 and

turn off if 4 similarly.

HARDWARE

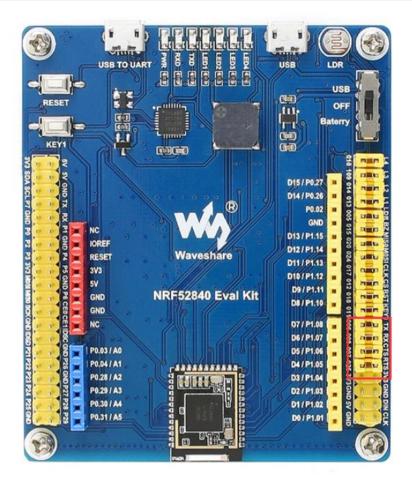
The schematic of serial part is as below:



H1 and H2 are jumpers of UART pins. CP2102 is accessed to NRF52840 only when

jumpers H1 and H2 are inserted.

UART	GPIO
СРТХ	P0.06
CPRX	P0.08
CPRTS	P0.25
СРСТЅ	P0.07



[Note] In official SDK, CPRTS is defined as P0.05, however, CPRTS is connected to

XCOM V2.0		-	
UART example started.	<u>^</u>	串口选择	
LEDO ON		COM14:USB-	-SERIAL \sim
LEDO OFF		波特率	115200 ~
LED1 ON 4		停止位	1 ~
LED1 OFF		数据位	8 ~
		奇偶校验	无~
		串口操作	● 关闭串口
		保存窗口	清除接收
		16进制星	誌□ 白底黑字
		RTS	DTR
	\sim	□ 时间戳(以换行回车断帧)
单条发送 多条发送 协议传输 帮助			
	5	5	发送新行
	6	6	
2 2	7	7	关联数字键盘
3	8	8	自动循环发送
4 4		9 周	期: 1000 ms
首页上一页	下一页 尾页	-	导入导出条目
	R:83 CTS=0 DSR=0 DCD=0	当前时间 11:5	6:33 .::

P0.25 on NRF52840 Eval kit. You need to modify the definition to P0.25 when using.

CHAPTER 6. SPI

Examples used in this chapter: 004_SPI(Example), 005_SPI(OLED)

In this chapter, we will describe about how to use SPI interface of NRF52840, do

loopback testing and try to drive a 0.96inch OLED module.

LOOPBACK TESTING

```
main.c
```

```
#include "nrf drv spi.h"
#include "nrf gpio.h"
#include "app util platform.h"
#include "nrf delay.h"
#include "app error.h"
#include <string.h>
#include "nrf log.h"
#include "nrf log ctrl.h"
#include "nrf log default backends.h"
#include "spi.h"
int main(void)
{
   APP ERROR CHECK (NRF LOG INIT (NULL));
  NRF LOG DEFAULT BACKENDS INIT();
   nrf drv spi config t spi config = NRF DRV SPI DEFAULT CONFIG;
   spi config.ss pin = CS PIN;
   spi config.miso pin = MISO PIN;
   spi config.mosi pin = MOSI PIN;
   spi config.sck pin = SCK PIN;
   /*
   Doesn't print what to send or what received in SPI
   */
   //APP ERROR CHECK(nrf drv spi init(&spi, &spi config, NULL,
NULL));
   /*
   Print what to send or what received in SPI
   */
   APP ERROR CHECK (nrf drv spi init (&spi, &spi config,
spi event handler, NULL));
```

```
NRF LOG INFO ("SPI example started.");
   NRF LOG FLUSH();
   while (1)
   ł
      memset(m rx buf, 0, m length);
      spi xfer done = false;
      APP ERROR CHECK (nrf drv spi transfer (&spi, m tx buf,
m_length, m_rx buf, m length));
      while (!spi xfer done)
      {
          WFE();
      }
      NRF LOG FLUSH();
      nrf_delay_ms(500);
   }
}
```

DRIVE OLED

When you drive OLED via SPI interface, you should notice that:

1. Do not use interrupt callback function when initializing SPI

```
/*
Doesn't print what to send or what received in SPI
*/
//APP_ERROR_CHECK(nrf_drv_spi_init(&spi, &spi_config, NULL,
NULL));
/*
Print what to send or what received in SPI
*/
APP_ERROR_CHECK(nrf_drv_spi_init(&spi, &spi_config,
spi event handler, NULL));
```

2. If you must to add interrupt codes, add delay to it.

```
void Single_Command(unsigned char cmd)
{
    nrf_gpio_pin_clear(DC);
    //if SPI is busy Frequently,delay here
    nrf_delay_us(30);
    spi_write(&cmd, sizeof(cmd));
}
```

3. The most basic function of OLED driver is written and read functions as below:

```
write
void Single_Command(unsigned char cmd)
{
    nrf_gpio_pin_clear(DC);
    //if SPI is busy Frequently,delay here
    //nrf_delay_us(30);
    spi_write(&cmd, sizeof(cmd));
```

read

}

```
void Single_Data(unsigned char dt)
{
    nrf_gpio_pin_set(DC);
    //if SPI is busy Frequently,delay here
    //nrf_delay_us(30);
    spi_write(&dt, sizeof(dt));
}
```

HARDWARE

In hardware, the SPI interface of NRF52840 are pinout to Arduino compatible

interface.

USB5V	3V3	U4				
R23 NC	R24 NC			SCL/D15 SDA/D14 AREF GND	17 16 15 14	P0.27 P0.26 P0.02 GND
3V3	3 IOREF 30 P0.18 29 20 USB5V 20 20 20 20	31 30 RESET 3V3 5 6 7		SCK/D13 MISO/D12 PWM/MOSI/D11 PWM/CS/D10 PWM/D9 D8	$ \begin{array}{r} 13 \\ 12 \\ 11 \\ 10 \\ 9 \\ 8 \end{array} $	P1.15 P1.14 P1.13 P1.12 P1.11 P1.10
GND	P0.03 22 P0.04 22 P0.28 2 P0.29 20 P0.30 19 P0.31 18	4 GND 4 PWR_VIN 3 A0 2 A1 1 A2 2 A3 4 A4	Arduino Boa	D7 PWM/D6 PWM/D5 D4 PWM/D3 D2 rd TX/D1 RX/D0	$ \begin{array}{r} 7 \\ 6 \\ 5 \\ 4 \\ 3 \\ 2 \\ 1 \\ 32 \\ \end{array} $	P1.08 P1.07 P1.06 P1.05 P1.04 P1.03 P1.02 P1.01
		Arduino				

[Note] The pins used in example herein are different with official SDK, if you use

SPI	GPIO
SPI_CS	P1.12
SPI_MOSI	P1.13
SPI_MISO	P1.14
SPI_SCK	P1.15

official SDK with NRF52840 Eval Kit, don't forget to modify the pins

Loopback testing

Data will be sent from MOSI and back from the MISO.

XCOM V2.0		
<pre><info> app: 4E 6F 72 64 69 63 <info> app: Transfer completed. <info> app: Received:</info></info></info></pre>	Nor di c	^
<info> app: 4E 6F 72 64 69 63 <info> app: Transfer completed.</info></info>	Nor di c	
<pre>(info) app: 4E 6F 72 64 69 63 (info) app: Transfer completed.</pre>	Nor di c	
(info) app: 4E 6F 72 64 69 63 (info) app: Transfer completed.	Nor di c	
(info) app: Received: (info) app: 4E 6F 72 64 69 63 (info) app: Transfer completed.	Nor di c	
(info) app: Received: (info) app: 4E 6F 72 64 69 63 (info) app: Transfer completed.	Nor di c	
info> app: Received: info> app: 4E 6F 72 64 69 63 info> app: Transfer completed.	Nor di c	
(info) app: Received; (info) app: 4E 6F 72 64 69 63 (info) app: Transfer completed.	Nor di c	
(info) app: Received: (info) app: 4E 6F 72 64 69 63	Nordic	

Drive OLED

The OLED will display image as below:

Full &	9 %	a m
27:	:56	SUN
TUSIC .	TEND	PHONE

CHAPTER 7. I2C

In this chapter, we will learn about how to use the I2C interface of NRF52840. There

are three examples, which are used here: 006 I2C(Scan Device), 007 I2C(BME280),

```
008_I2C(MPU6050 3D COUBE)
```

SCANNING SLAVE DEVICES

It keeps scanning the slave device address of those devices which are connected to

I2C bus.

```
for(address=1; address<=TWI_ADDRESSES; address++)
{
    err_code = nrf_drv_twi_rx(&m_twi, address, &sample_data,
    sizeof(sample_data));
    if (err_code == NRF_SUCCESS)
    {
        detected_device = true;
        device_address = address;
        NRF_LOG_INFO("TWI device detected at address 0x%x.",
        address);
        NRF_LOG_FLUSH();
    }
}</pre>
```

The code will scan slave devices address connected and print them.

READ DATA FROM BME280

BME280 is a module from Bosch that supports sense environmental temperature,

humidity and barometric pressure.

Write/read function from Bosch: https://github.com/BoschSensortec/BME280 driver

```
int8_t user_i2c_read(uint8_t dev_id, uint8_t reg_addr, uint8_t
*reg data, uint16 t len)
```

```
{
  int8 t rslt = 0; /* Return 0 for Success, non-zero for failure
*/
  /*
   * The parameter dev id can be used as a variable to store the
I2C address of the device
   */
  /*
   * Data on the bus should be like
   * |-----|
   * | I2C action | Data
                             1
   * |-----|
   * | Start | -
* | Write | (reg_addr)
                            1
   * | Stop
             | -
                             1
             | -
   * | Start
                            * | Read | (reg_data[0]) |
   * | Read
             | (....)
                             | (reg_data[len - 1]) |
   * | Read
   * | Stop | -
                      |
   * |-----|
   */
  return rslt;
}
int8 t user i2c write (uint8 t dev id, uint8 t reg addr, uint8 t
*reg data, uint16 t len)
{
  int8 t rslt = 0; /* Return 0 for Success, non-zero for failure
*/
  /*
  * The parameter dev id can be used as a variable to store the
I2C address of the device
   */
  /*
   * Data on the bus should be like
   * |-----|
   * | I2C action | Data
                             - I
   * |-----|
   * | Start | -
                            * | Write | (reg_addr) |
* | Write | (reg_data[0]) |
                              * | Write
             | (....)
                             * | Write | (reg data[len - 1]) |
   * | Stop
             | - |
```

```
* |-----|
*/
return rslt;
}
```

Working with NRF52840, you can use its library to realize the write/read function :

```
int8 t user i2c read(u8 dev id, u8 reg addr, u8 *reg data, u16 len)
{
   ret code t err code =
nrf drv twi tx(&m twi,dev id,&reg addr,1,false);
   APP ERROR CHECK (err code);
   err code = nrf drv twi rx(&m twi,dev id,reg data,len);
   APP ERROR CHECK (err code);
   return err code;
}
#define MAX WRITE LENGTH 200
int8 t user i2c write(u8 dev id, u8 reg addr, u8 *reg data, u16
len)
{
   ret_code_t err_code;
   uint8 t write data[MAX WRITE LENGTH];
   if(len>MAX WRITE LENGTH-1)
   {
      err code = 1;
      return err code;
   }
   write data[0] = reg_addr;
   memcpy(&write data[1],reg data,len);
   err code =
nrf drv twi tx(&m twi,dev id,write data,len+1,false);
   APP ERROR CHECK (err code);
   return err code;
```

}

BME280 Initialization in main.c

```
struct bme280 dev dev;
int8 t rslt = BME280 OK;
dev.dev id = BME280 I2C ADDR SEC;
dev.intf = BME280 I2C INTF;
dev.read = user i2c read;
dev.write = user i2c write;
dev.delay ms = user delay ms;
//before you init bme280, you can choose to do a selftest
rslt = bme280 crc selftest(&dev);
if(rslt == 0)
{
   NRF LOG INFO("BME280 self test pass\r\n");
   NRF LOG FLUSH();
}
rslt = bme280 init(&dev);
if(rslt == 0)
{
   NRF LOG INFO("Init Success\r\n");
   NRF_LOG_FLUSH();
}
else
{
   NRF LOG INFO("Init Fail, Please Check your address or the wire
you connected!!!\r\n");
   NRF LOG FLUSH();
   while(1);
}
/*
   Using normal mode to read the data
*/
stream sensor data normal mode(&dev);
```

bme280_crc_selftest(): Self testing function

stream_sensor_data_normal_mode(&dev): Read data of bme280 and calculate. Use

bme280_get_sensor_data() and print_sensor_data() to read and print_sensor_data().

READ DATA FROM MPU6050

This example is run to read MPU6050 data via I2C bus.

```
u8 IIC Write 1Byte (u8 SlaveAddress,u8 REG Address,u8 REG data)
{
   ret code t err code = user i2c write(SlaveAddress, REG Address,
&REG data, 1);
   return err code;
}
u8 IIC Read 1Byte(u8 SlaveAddress,u8 REG Address,u8 *REG data)
{
   ret code t err code = user i2c read(SlaveAddress, REG Address,
REG data, 1);
   return err code;
}
u8 IIC Write nByte (u8 SlaveAddress, u8 REG Address, u8 len, u8
*buf)
{
   ret code t err code = user i2c write (SlaveAddress, REG Address,
buf, len);
   return err code;
}
u8 IIC Read nByte(u8 SlaveAddress, u8 REG Address, u8 len, u8 *buf)
{
   ret code t err code = user i2c read(SlaveAddress, REG Address,
buf, len);
   return err code;
}
```

The basic functions are user_i2c_read() and user_i2c_write().



HARDWARE

USB5V	3V3	U4		
₹R23 NC	R24 NC		SCL/D15 SDA/D14 AREF GND	17 P0.27 16 P0.26 15 P0.02 14 GND
3V3	31 IOREF 30 P0.18 29 28 28 USB5V 27 26 25 24 24 P0.03 23 P0.04 22 P0.28 21 P0.29 20 P0.30 19 P0.31 18	3130RESET3V35VGNDGNDPWR_VINA0A1A2A3A4A5Arduino	SCK/D13 MISO/D12 /MOSI/D11 /M/CS/D10 PWM/D9 D8 D7 PWM/D6 PWM/D5 D4 PWM/D3 D2 TX/D1 RX/D0	13 P1.15 12 P1.14 11 P1.13 10 P1.12 9 P1.11 8 P1.10 7 P1.08 6 P1.07 5 P1.06 4 P1.05 3 P1.04 2 P1.03 1 P1.02 32 P1.01
	12C		GPIO	
	SCL_PI	N	P0.27	
SDA_PIN			P0.26	

In hardware, the I2C interface are pinout to the Arduino compatible header.

The I2C pins can be re-configured to others by modifying the codes:

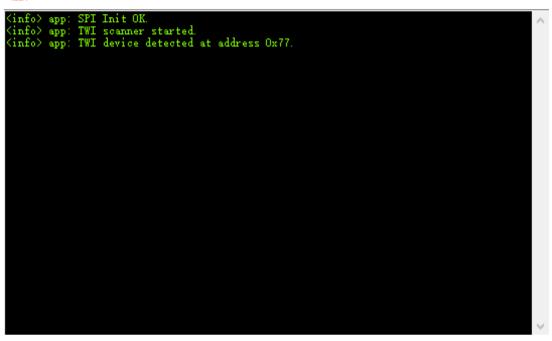
```
#define SDA_PIN NRF_GPIO_PIN_MAP(0,26)
#define SCL_PIN NRF_GPIO_PIN_MAP(0,27)
```

[Note] Don' t forget to change hardware connection if you re-configure the pins.

After running codes, the data will be printed to serial port and display the I2C address

in OLED as below:

XCOM V2.0





• Downloading 007_12C(BME280) codes, data will be printed to OLED as below:



• Downloading 008_I2C(MPU6050 3D COUBE), Data will be printed as below(The 3D

figure display is not stable):





CHAPTER 8. NFC

CODES

This example is used to starting mobile APP.

The name of Android Phone APP:

```
static const uint8_t m_android_package_name[] =
{'n', 'o', '.', 'n', 'o', 'r', 'd', 'i', 'c', 's',
'e', 'm', 'i', '.', 'a', 'n', 'd', 'r', 'o', 'i',
'd', '.', 'n', 'r', 'f', 't', 'o', 'o', 'l', 'b',
'o', 'x'};
```

The name of Windows Phone APP:

```
static const uint8_t m_windows_application_id[] =
{'{', 'e', 'l', '2', 'd', '2', 'd', 'a', '7', '-',
'4', '8', '8', '5', '-', '4', '0', '0', 'f', '-',
'b', 'c', 'd', '4', '-', '6', 'c', 'b', 'd', '5',
'b', '8', 'c', 'f', '6', '2', 'c', '}';
```

Initializing callback function, which will be execute when NFC objects are detected to

turn on LED0.

```
static void nfc callback(void * p context, nfc t2t event t event,
const uint8 t * p data, size t data length)
{
   (void) p context;
   switch (event)
   {
      case NFC_T2T_EVENT_FIELD_ON:
          bsp board led on (BSP BOARD LED 0);
          break;
       case NFC T2T EVENT FIELD OFF:
          bsp board led off (BSP BOARD LED 0);
          break;
      default:
          break;
   }
}
```

In main.c file, the process to start NFC are that:

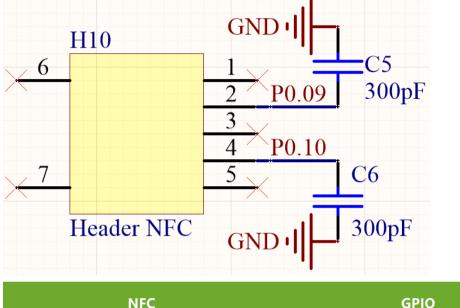
- 1. Set interrupt callback function and initialize NFC:
- 2. Encode NFC information
- 3. Load NFC information to NFC end

4. Start NRF

```
int main(void)
{
   uint32 t len;
   uint32 t err code;
   //Init log
   log init();
   //Init led
   bsp board init(BSP INIT LEDS);
   //set NFC with a callback
   err_code = nfc_t2t_setup(nfc_callback, NULL);
   APP ERROR CHECK (err code);
   //provide available buffer size for encoding function
   len = sizeof(m ndef msg buf);
   //Encode launchapp message into buffer
   err code = nfc launchapp msg encode(m android package name,
                                  sizeof(m android package name),
                                  m windows application id,
                                  sizeof(m windows application id),
                                  m ndef msg buf,
                                  &len);
   APP ERROR CHECK (err code);
   //Set created message as the NFC payload
   err code = nfc t2t payload set(m ndef msg buf, len);
   APP ERROR CHECK (err code);
   //Start sensing NFC field
   err_code = nfc_t2t_emulation_start();
   APP ERROR CHECK (err code);
   while (1)
   {
      NRF LOG FLUSH();
      ___WFE();
   }
}
```



HARDWARE



The pins for NFC are compatible with official⁶ SDK

NFC	GPIO
NFC_1_PIN	P0.09
NFC_2_PIN	P0.10

Testing:

Install nRF Toolbox APP in your phone. Connect NFC coils and download the codes to

eval board. LED1 will be on if you close NFC side of phone to the NFC coils, and turn

off when taking away

⁶ Nordic: <u>https://www.nordicsemi.com/</u>

CHAPTER 9. BLUETOOTH

The examples used in this chapter are: Part-2-Bluetooth-Slave-Device, Part-3-

Bluetooth-Master-Device, Part-4-Bluetooth-Master&Slave-Device.

PREPARATION

The software required:

- MDK5.25
- Jlink driver
- nRFx-Command-Line-Tools
- nRF Toolbox (mobile APPP)
- Mobile phone which supports BLE

INSTALL SOFTDEVICE

To run the examples, you need to first install Softdevice.

- Erase Flash
- Install Softdevice

BLUETOOTH SERIAL TRANSPARENT

The example located in Bluetooth-Slave-Device\USER

• Open SlaveDevice-Bluetooth-To-UART(APP) example, compile and download to

NRF52840 eval kit.

Expected Result:

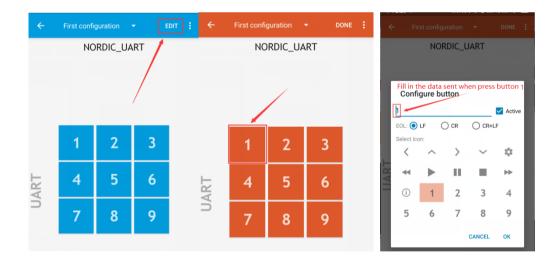
• LED1 is blinking when broadcast after downloading the code.

Open nRFToolbox software in your phone and enter UART, Connect Nodic_UART

device

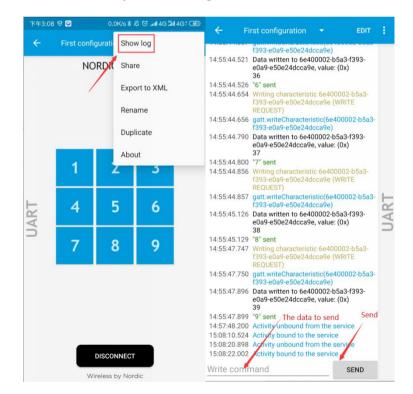
≡ nRF Toolt	хоо	:	÷		
CGMS CGMS HRM KSC	BGM BGM CSC CSC HTM HTM TEMPLATE	BPM DFU DFU PROXIMITY UART	UART	Select device: BONDED DEVICES: 小愛音箱-4536 7C:49:EB:70:4C:33 WQ-20151126QGHA DC:85:DE:B3:42:06 AVAILABLE DEVICES: n/a 45:E1:C8:FA:93:23 Nordic_UART D7:4B:27:E8:10:1C n/a 65:D4:82:F9:A5:05 n/a 66:E3:23:62:4B:4E n/a 6B:41:6D:85:21:4E	• (• • •

- After connecting, LED1 is on
- You can edit the figures and try to send:



÷	First confi	guration	~ E	DIT 🚦	XCOM V2.0	
	NC	ORDIC_UA	RT		123456789	\wedge
	1	2	3			
	<u> </u>					
RT	4	5	6			
UART						
	7	8	9			
	(Click 123456	5789			
						\sim

• You can also send data directly on Show log:





COM1 115200 bps, 8N1, RTS/CTS S	ettings Clear <u>A</u> bout		Creating service Binding to the service
Local RTS DTR Remote CTS DS	R RI CD ERR BREAK		Service started
		15:21:15.898	Connecting
Velcome to WaveShare.com		15:21:15.905	<pre>gatt = device.connectGatt(autoConnect = false)</pre>
			Activity bound to the service
		15:21:16.007	[Callback] Connection state changed with status: 0 and new state: 2 (CONNECTED)
		15:21:16.014	Connected to D7:4B:27:E8:10:1C
			Discovering Services
			gatt.discoverServices()
			Services Discovered
			Primary service found
		15:21:16.793	gatt.setCharacteristicNotification(6e4 00003-b5a3-f393-e0a9-e50e24dcca9e, true)
		15:21:16.803	Enabling notifications for 6e400003- b5a3-f393-e0a9-e50e24dcca9e
		15:21:16.805	gatt.writeDescriptor(00002902- 0000-1000-8000-00805f9b34fb, value=0x01-00)
		15:21:16.824	Data written to descr. 00002902-0000- 1000-8000-00805f9b34fb, value: (0x) 01-00
		15:21:16.826	Notifications enabled
		15:21:23.641	Notification received from 6e400003-b5a3-f393-e0a9- e50e24dcca9e, value: (0x) 57-65-6C-63-6F-6D-65-20-74-6F-20-57-6 1-76-65-53-68-61-72-65-2E-63-6F-6D-0D
/elcome to WaveShare.com		15:21:23.647	"Welcome to WaveShare.com " received

BLUETOOTH AGAINST LOSING

Download:

• Open the project SlaveDevice-Bluetooth-Proximity (APP), which located in Part-2-

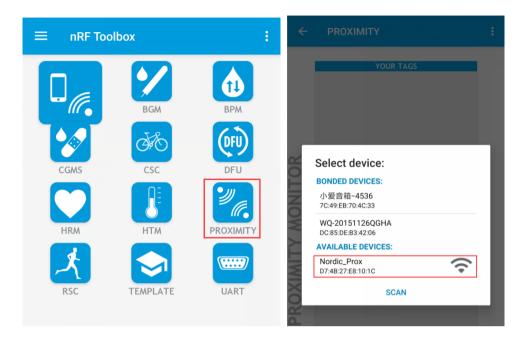
Bluetooth-Slave-Device\USER

• Compile and download to eval board

Expected result:

- LED1 is lighted in when broadcasting after downloading
- Turn on Bluetooth service of your phone

• Open nRF Toolbox and clock UART. Connect Nordic_UART device



• After connecting, LED1 will light on. The Toolbox will display as below:



- You can click the indicator button on APP to control P0.15 pin of Eval Kit board. When testing, you can connect P0.15 to one LED to check the status. Click Delete button to delete device.
- Press button S1 of Eval Kit board, music will be played in your phone, and

stopped if S1 is pressed again.

• The software also prompt if you take the phone away from the Eval Kit board.

÷	PROXIMITY		:
	YOUR TAGS	*	
	D7:4B:27:E8:10:1C		
~			
MON	nRF Toolbox Nordic_Prox is getting away!		
Σ		ОК	
WIX			

BLUETOOTH BEACON (SUPPORT WECHAT SHAKE)

Beacon is that build a small information base station by using BLE, it can be used in

Indoor navigation, mobile payment, store shopping, Object tracking and so on.

Beacon is not a standard built by Bluetooth SIG

Download:

Open the project SlaveDevice-Bluetooth-BEACON(WeChat) which located in Part-

2-Bluetooth-Slave-Device\USER

• Compile and Download to Eval Kit Board

[Note] The example is different with official examples(line 74-77)

APP_COMPANY_IDENTIFER change to:0X004c

APP_MAJOR_VALUE change to:0x00,0x0A

APP_MINOR_VALUE change to:0x00,0x07

APP_BEACON_UUID change to:0xFD,0xA5,0x06,0x93,\

0xA4,0xE2,0x4F,0xB1,\

0xAF,0xCF,0xC6,0xEB,\

Expected result:

- Open WeChat, and open Shake, there are not Nearby devices detect
- Download example and run it, LED1 will light on
- Open Shake again, Nearby device is detected



• Click Nearby and try to shake



[Note] The Web page detected is configurable, for more details about it you

can refer to official guide of WeChat: https://zb.weixin.qq.com/intro.xhtml

Attributes	Value
UUID	FDA50693-A4E2-4FB1-AFCF-C6EB07647825
Major	10
Minor	7

The testing ID used in the example is the open ID of WeChat

WIRELESS MOUSE

HOGP profile (HID over GATT Profile) can be used to release wireless mouse function.

HOGP use basic protocol GATT of BLE to release interaction between HID host and

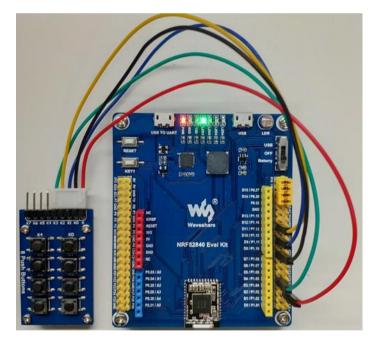
Device.

Download:

- Open the project SlaveDevice_bluetooth-HIDS_MOUSE(APP),
- Compile and download to Eval Kit board.

Expected result:

- Eval Kit board broadcast and LED1 lights on
- Connect P0.11, P0.12, P0.24, P0.25 to buttons



• Cursor will move when press buttons.

Pin	KEY	功能
P0.11	KEY1(left)	Mouse move left for 5 pixels

NRF52840 Eval Kit User Manual		
P0.12	KEY2(up)	Mouse move up for 5 pixels
P0.24	KEY3(right)	Mouse move right for 5 pixels
P0.25	KEY4(down)	Mouse move down for 5 pixels
ୟ 6.7KB/s ୬ ୟ ପି ଲା ଲୋ ବ୍ ⊞େ 1 ◯		
12 03 Sat, January 19 Hazy 20'		
FastMiui Apr Store Security Tools		
Weather Gallery Music Lock		
Themes Mi Cloud Portable hotspot		
S 🛛 O		

- If your PC support BLE, you can also used to control cursor on PC
- If you want to use the joystick of Arduino Accessory Shield, you need to modify

the definition of keys in pca10056.h

#define	BUTTON_1	NRF_GPIO_PIN_MAP(0,29)
#define	BUTTON_2	NRF_GPIO_PIN_MAP(0,4)
#define	BUTTON_3	NRF_GPIO_PIN_MAP(0,28)
#define	BUTTON_4	NRF_GPIO_PIN_MAP(0,31)

WIRELESS KEYBOARD

HOGP profile (HID over GATT Profile) can be used to release wireless keyboard

function. HOGP use basic protocol GATT of BLE to release interaction between HID

host and Device.

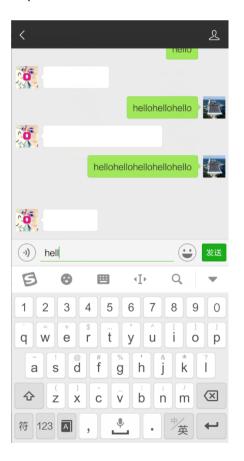
Download:

- Open the project SlaveDevice_bluetooth-HIDS_KEYBOARD(APP),
- Compile and download to Eval Kit board.

Expected result:

- Eval Kit board broadcast and LED1 is blinking.
- Open Bluetooth of your Phone, search and connect Nordic_keyboard device. After connecting, LED1 is on
- Then you can try to input text

Press KEY1 on board to input string "hello". Connect P0.15 to LED, and you can find that LED is turned off if Caps Lock is enabled, and the LED is turned on when Caps Lock is disabled.



CSCS

NRF52840 will send seed, rate, riding distance, total distance and gear ratio data to

phone via BLE.

The data is generated by software (simulated), which is based on Cycling Speed and

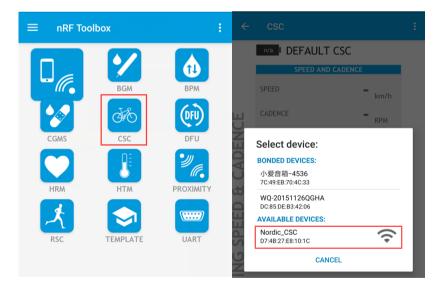
Cadence profile.

Download:

- Open the project Bluetooth-CSCS(APP),
- Compile and download to Eval Kit board.

Expected result:

- Eval Kit board is broadcasting and LED1 blinking
- Open nRF Toolbox APP, click CSC(Cycling Speed and Cadence)
- Click Connect to scan and connect to Nordic_CSC device



• After connecting, corresponding data are printed to APP

÷	CSC		:			
	^{81%} I NORDIC	_CSC				
	SPEED AND CADENCE					
	SPEED	16.0 _{km/h}				
Ц	CADENCE	34 _{RPM}				
EN	DISTANCE	1.30 _{km}				
CAD	TOTAL DISTANCE	1.38 _{km}				
t	GEAR RATIO	3.3				
CYCLING SPEED & CADENCE		DNNECT by Nordic				

BLUETOOTH MASTER- CARDIOTACHOMETER

You need two NRF52840 Eval Kit for this example, one is used as master, and another

is slaver.

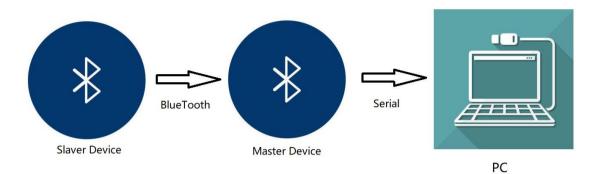
Download:

- Open project Master-Link-To-Slave-By-HRS(Master), compile and download it to one Eval Kit board.
- Open project Master-Link-To-Slave-By-HRS(Slaver), compile and download it to another Eval Kit board.

Both projects can be found in Part-3-Bluetooth-Master-Device\USER

Expected result:

• The communication between two Eval Kit boards is as below:



• After downloading, the master will auto-link to slaver, LED1 are on if they connect

successfully.

• Slaver will send data to Master, which are transferred to serial port and printed in

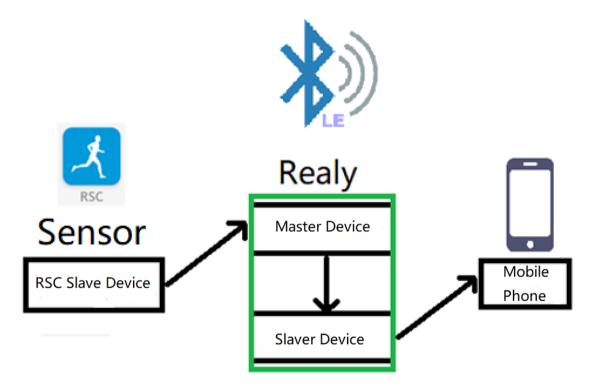
XCOM V2.0	_		×
<pre>Cinfo> app: Heart Rate collector example started. Cinfo> app: Starting scan. Cinfo> app: Connected Cinfo> app: Connected Cinfo> app: Data length for connection 0x0 updated to 251. Cinfo> app: Data length for connection 0x0 changed to 247. Cinfo> app: Data length for connection secured: role: Central, conn_handle: 0, procedure: Encryption Cinfo> app: Hart ATI MTU on connection secured: role: Central, conn_handle: 0, procedure: Encryption Cinfo> app: Battery Level Read as 97 %. Cinfo> app: Heart Rate = 180. Cinfo> app: Heart Rate = 180. Cinfo> app: Battery Level received 99 %. Cinfo> app: Heart Rate = 180. Cinfo> app: Heart R</pre>	串口选择 COM4:USB- 波特率 停止位 数据位 奇偶校验 串口操作 保存窗口 □ 16进制数	SERIAL 115200 1 8 天 (● 打开	 ✓ ✓
<pre>(info> app: Heart Rate = 170. (info> app: Battery Level received 98 %. (info> app: Heart Rate = 180. ************************************</pre>	□ RTS □ 时间戳(DTR (以换行回车 发送 清除发 停止发 oenedy.c	≡断帧) ś 送
	当前时间 15:4	41:28	.:

MASTER-SLAVER-BLE RELAY

With this example, Eval Kit board can work as master and slaver at the same time.

- Eval Kit board auto connect to another slaver (RSCS) when work as master
- Eval Kit board will connect to phone when work as slaver
- With BLE Relay example, Eval Kit can receiver data from RSCS slaver and send it to

mobile phone as below:



To complete this example, you require two Eval Kit boards, one is work as Sensor(RSCS slaver) and another is Relay (Master-Slaver)

Download:

- Open project Master&Slaver-Relay(Master), compile and download to Relay board.
- Open project Master&Slaver-Relay(Slaver), compile and download to Sensor board

【Note】	The slaver project is RSC project, it also can be HRM project	
--------	---	--

Peripheral	Re]	Lay		Central
Heart Rate > Sensor	Collector	Heart -> Rate Sensor	>	Collector
++ ++ Running > Speed Sensor ++	Running Speed - Collector +	and Running -> Speed Sensor		++

Expected result:

• Power on Relay board, it begins to scan devices(master) and broadcasts(slaver).

LED1 and LED3 are on

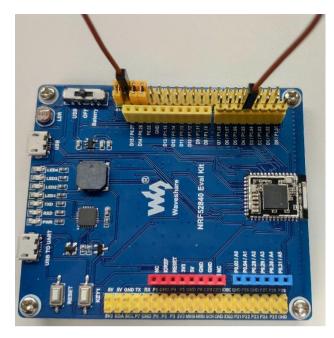
• Power on Sensor board, it begin to broadcast and LED1 is blinking, LED1 stay on if

Sensor board connected to Master successfully

Indicators of Relay Board

Indicators	Description
LED1	On: Master is scanning (it keeps scanning even Slaver is connected)
LED2	On: Master has connected to slaver (slaver: sensor board)常
LED3	On: Slaver is broadcasting
LED4	On: Slaver is connected master (master: mobile phone)

[Note] The pin of LED3 is not compatible with SDK, you need to wire it to P1.05 manually.



Indicator of Sensor board:

Indicator	Description
LED1	Blinking: Broadcasting
	On: Connected

- Open nRF Toolbox on phone, click RSC and Connect to nRF Relay device.
- After connecting, LED2 of Relay board is turned off and LED4 turns on. Data are

sent to phone.

WUAVESHARE NRF52840 Eval Kit User Manual har ≡ nRF Toolbox DEFAULT RSC n/a I NRF RELAY BGM BPM SPEED AND CADENCE SPEED 12.6_{km/h} - _____ OFU DFU CADENCE 80_{SPM} CADENCE CSC 2 DISTANCE 8_ l Select device: BONDED DEVICES: TOTAL DISTANCE n/a 小爱音箱-4536 7C:49:EB:70:4C:33 STEPS 11 പ WQ-20151126QGHA DC:85:DE:B3:42:06 **RUNNING SPEED** UART TEMPLATE AVAILABLE DEVICES: RUNNING nRF Relay D7:48:27:E8:10:1C ŝ SCAN DISCONNECT Wireless by Nordic Wireless by Nordic

XCOM V2.0

<pre>(info) app: Relay examp.</pre>	le started.	~
(info) app: Fast advert		
(info) app: Central con		
	find HRS or RSC on conn_handle OxO	
	dler: Connection secured: role: Central, conn_handle: 0,	
procedure: Encryption		
	dler: Connection secured: role: Central, conn_handle: 0,	
procedure: Encryption	aler: connection scoules, rose, central, conn_nanaec, c)	
	ed and Cadence service discovered on conn_handle 0x0	
	dler: Connection secured: role: Central, conn_handle: 0,	
procedure: Encryption	der, connection scontes, rore, central, conn_nandre, c)	
	= 1280	
	= 896	
(info) app: Speed		
(info) app: Speed		
	= 512	
	= 896	
	= 1280	
	= 1664	
	= 1280	
	= 896	
	= 512	
	= 128	
	= 512	
	= 896	
	= 1280	
	= 898	\sim

CHAPTER 10. MESH⁷

This chapter we describe some of the basic concepts of the Bluetooth Mesh network using Nordic's nRF5 SDK for Mesh.

The project we used herein is light-switch example from Nordic's MESH SDK, for more details about it, you can refer to official Documents.

The light-switch example demonstrates the major parts of the mesh network

ecosystem. it consists of three minor examples:

- Light switch server: a minimalistic server that implements a generic OnOff server model, which is used to receive the state data and control the state of LED1 on the board
- Light Switch Client: A minimalistic client that implements four instances of a Generic OnOff client model. When a user presses any of the buttons, an OnOff Set message is sent out to the configured destination address.
- Mesh Provisioner: A simple static provisioner implementation. This provisioner provisions all the nodes in one mesh network. Additionally, the provisioner also configures key bindings and publication and subscription settings of mesh model instances on these nodes to enable them to talk to each other.

These three examples will be referred to as the server, the client and the provisioner respectively.

The four buttons are used to initiate certain actions and four LEDs are used to reflect the status of actions as below:

⁷ You can directly refer to Nordic website:

- Server:
 - During provisioning process:
 - LED3 and 4 blinking: Device identification active
 - LED1 to 4: Blink four times to indicate provisioning process is

completed

- After provisioning and configuration is over:
 - LED1: Reflects the value of OnOff state on the server
 - LED ON: Value of the OnOff state is 1 (true)
 - LED OFF: Value of the OnOff state is 0 (false).
- Client:
 - During provisioning process:
 - LED3 and 4 blinking: Device identification active
 - LED1 to 4: Blink four times to indicate provisioning process is completed
 - After provisioning and configuration is over, buttons on the client are used to send OnOff Set message to the servers:
 - Button1: Send a message to the odd group(address: 0xC003) to turn on LED1
 - Button2: Send a message to the odd group (address: 0xC003) to turn off LED1
 - Button3: Send a message to the even group (address: 0xC002) to

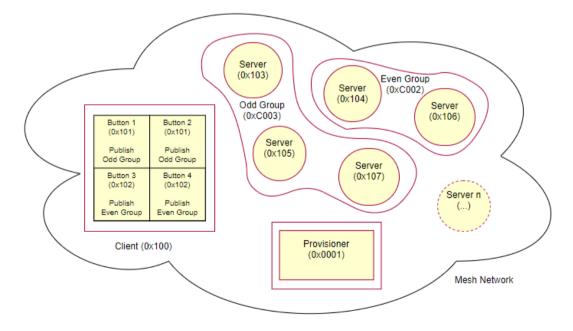
turn on LED1

- Button 4: Send a message to the even group (address: 0xC002) to turn off LED1.
- Provisioner:
 - Button1: Start provisioning
 - LED1: Reflects the state of the provisioning.
 - LED ON: provisioning of the node is in progress
 - LED OFF: No ongoing provisioning process
 - LED2: Reflects the state of the configuration
 - LED ON: Configuration of the node is in process
 - LED OFF: No ongoing configuration process

The following figure gives the overall view of the mesh network that will be set up

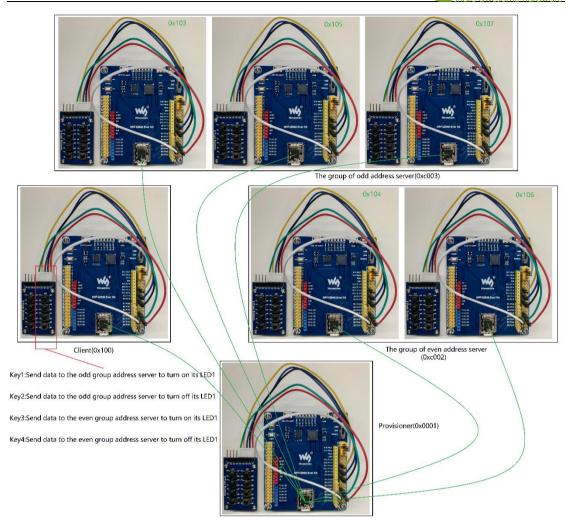
in this example. Numbers in parentheses indicate the addresses that are assigned to

these nodes by the provisioner.



The Mesh is as below:





Hardware requirement:

it required at least three NRF52840 Eval Kit:

- One Eval Kit board for server
- One Eval Kit board for client
- One Eval Kit board for provisioner

Software requirement:

- 1. nRF Mesh SDK: nrf5_SDK_for_Mesh_v2.2.0_src, download and extract the SDK
- 2. BLE SDK: nRF5_SDK_15.0.0_a53641a, download and extract the SDK
- 3. nrfjprog: This tool is used to erase Flash, can be installed by nRFx Command Line

Tools for Windows

4. SEGGER Embedded Studio for ARM 4.10a (The MESH project is built by this tool)

[Note] All the resources should be extracted to the same path, for linked by

compiler. Otherwise, the project is failed to compile.

```
.
+-- nrf5_sdk_for_mesh/
+-- nRF5_SDK_15.0.0_a53641a/
```

Download:

- Connect NRF52840 Eval Kit board to PC. Decide which board you want to use as client and which one as provisioner
- Open projects with SEGGER Embedded Studio for ARM 4.10a
 - Provisioner:

MESH\nrf5_SDK_for_Mesh_v2.2.0_src\examples\light_switch\provisioner\

light_switch_provisioner_nrf52840_xxAA_s140_6_0_0.emProject

Server:

MESH\nrf5_SDK_for_Mesh_v2.2.0_src\examples\light_switch\proxy_server\ligh

t_switch_proxy_server_nrf52840_xxAA_s140_6_0_0.emProject

Client:

 $\label{eq:MESH} MESH\nrf5_SDK_for_Mesh_v2.2.0_src\examples\light_switch\proxy_client\light \\$

_switch_proxy_client_nrf52840_xxAA_s140_6_0_0.emProject



• Compile projects

→ → nr 0.0K → nrs are met: nrs are met: above copyright not: → → nrs 0.1K 0.1K → Build and Debug Ctrl+F7 above copyright not:	File Edi	it View	v Search	Navigate	Project	Bu	ild Debug Target Tools Window Help		
Rebuild light_switch_provisioner_nrf52840_xxAA_s140_6.0.0 Alt+F7 Project Items Code Data Clean light_switch_provisioner_nrf52840_xxAA_s140_6.0.0 Alt+F7 Project Items Code Data Clean light_switch_provisioner_nrf52840_xxAA_s140_6.0.0 Alt+F7 Project Items Code Data Alt Alt D <thd< th=""> D D</thd<>	Project Exp	plorer			main.c	÷۳	Build light_switch_provisioner_nrf52840_xxAA_s140_6.0.0	F7	
Project Items Code Data A	Debug	_		_				Alt+F7	
Project retring Code Code Shift+F7 P mm 37K 14K P mm 03K 02K P mm 0.0K Cancel Build Solution Alt+Shift+F7 P mm 0.0K P Build and Debug Ctrl+T, F5 P mm 0.0K P Build and Debug Ctrl+T, Ctrl+F5 P mm 0.0K P P P P P P mm 0.0K P			-				Clean light_switch_provisioner_nrf52840_xxAA_s140_6.0.0		
> m 3.7.5 1.4K 1.4K > m 0.3.K 0.2K P > m 0.3.K 0.0K P m m <				Data		×	Build Solution	Shift+F7	A
▷ ♣ nr 2.7K 1.2K vith or without models ▷ ♣ nr 0.0K Compile main.c Ctrl+F7 ▷ ♣ nr 0.1K 0.1K 0.1K ▷ ♣ sin 0.4K 0.0K Build and Debug Ctrl+T, F5 Bearer 17.7K 10.8K Build and Run Ctrl+T, Ctrl+F5 ▷ ♣ ad 0.3K 0.0K Imid Configurations Standard Ctrl+ ded into a Nordic ctrl or a software upper t notice, this list documentation and/or ▷ ♣ ad 0.1K 0.0K Imid Configurations Statch Build Parallel and Unity Bui							Rebuild Solution	Alt+Shift+F7	
> Anf 0.0K ns are met: > Anf 0.0K Cmple main.c Ctrl+F7 > Anity 0.1K 0.1K Bearer Ctrl+T, Ctrl+F5 > Anity 0.0K Unid and Debug Ctrl+T, Ctrl+F5 Another Ctrl+T, Ctrl+F5 > And 0.3K 0.0K Image: Ctrl+E, Ctrl+F7 Another Ctrl+T, Ctrl+F5 > And 0.3K 0.0K Ctrl+E, Ctrl+F7 Another Ctrl+T, Ctrl+F5 > And 0.3K 0.0K Ctrl+E, Ctrl+F7 Another Ctrl+T, Ctrl+F5 > And 0.3K 0.0K Ctrl>E, Ctrl+E, Ctrl+T Another Ctrl+T, Ctrl+F5 > And 0.1K 0.0K Ctrl+E, Ctrl+F7 Another Ctrl+T, Ctrl+F5 > And 0.1K 0.0K Ctrl>E, Ctrl+E, Ctrl+T Another Ctrl+T, Ctrl+F5 > And 0.1K 0.0K Another Ctrl+E, Ctrl+T Another Ctrl+E, Ctrl+T > Another Ctrl>E, Ctrl+E, Ctrl+E, Ctrl+E, Ctrl+E, Ctrl+F7 Free Ctrl+E, Ctrl+F7 Another Ctrl+E, Ctrl+F7 Another Ctrl+E, Ctrl+F7 > Another Ctrl>E, Ctrl+E, Ctrl+E, Ctrl+E,							Clean Solution		with or without modific
> > > > Score pile main.c Ctrl+F7 > > > > > Score pile main.c Ctrl+F7 > > > > > Score pile main.c Ctrl+F7 > > > > > Score pile main.c Ctrl+F7 > > > > > Score pile main.c Ctrl+T, F5 above copyright not: > > > > > > Score pile main.c Ctrl+T, F5 above copyright not: > > > > > > > > > >									
> in 0.4K 0.0K 0.0K in and 0.00g Clink, F3 > in 0.4K 0.0K in and 0.0kg Clink, F3 In 5 ° ° > in 0.4K 0.0K in 0.0K in 0.0K in 0.0K in 0.0K > in 0.4K 0.0K in 0.0K in 0.0K in 0.0K in 0.0K in 0.0K > in 0.4K 0.0K in 0.0K in 0.0K in 0.0K in 0.0K in 0.0K > in 0.4K 0.0K in 0.0K in 0.0K in 0.0K in 0.0K in 0.0K > in 0.1K 0.0K in 0.0K in 0.0K in 0.0K in 0.0K in 0.0K in 0.0K > in 0.1K 0.0K in 0.0K in 0.0K in 0.0K in 0.0K in 0.0K > in 0.1K 0.0K in 0.0K in 0.0K in 0.0K in 0.0K in 0.0K in 0.0K > in 0.0K in 0.0K <td< td=""><td></td><td></td><td>2.3K</td><td>1.8K</td><td></td><td>Compile main.c</td><td></td></td<>			2.3K	1.8K			Compile main.c		
▷ 월 sin 0.4K 0.0K ⓐ Bearer (7.7K) (0.8K) ▷ 월 ad 0.3K 0.0K ▷ 월 ad 0.2K 0.0K ▷ 월 ad 0.2K 0.0K ▷ 월 ad 0.2K 0.0K ▷ 월 ad 0.1K 0.0K ▷ 월 ad 0.1K 0.0K ▷ 월 be 1.3K 0.0K ▷ 월 br 0.4K 0.0K ▷ 월 fift 0.1K 0.0K ▷ 월 ga 0.3K 0.0K ▷ 월 ga 0.3K 0.0K	⊳	🔒 rtt	0.1K	0.1K			Build and Debug	Ctrl+T_E5	above copyright notice,
ad 0.2K 0.0K 10 Image: Cancel Build Ctrl+. ded into a Nordic ct or a software up t notice, this list b b ad 0.1K 0.0K Image: Cancel Build Ctrl+. ded into a Nordic ct or a software up t notice, this list b b ad 0.1K 0.0K Image: Cancel Build Configurations the notice, this list b b b filt 0.0K Image: Cancel Build Configuration Image: Cancel Build Configuration Image: Cancel Build Configuration b b b filt 0.0K Image: Cancel Build Configuration Image: Cancel Build Configuration Image: Cancel Build Configuration Image: Cancel Build Configuration b b filt 0.1K 0.0K Image: Cancel Build Configuration Image							2		
> ad 0.2K 0.0K 10 radiance build Curles Curles <t< td=""><td>_</td><td></td><td></td><td></td><td></td><td>-</td><td></td><td>cui+i, cui+i s</td><td></td></t<>	_					-		cui+i, cui+i s	
> Ad 0.1K 0.0K Image: Build Configurations total a Structurate duration and total a Structurate duration a Structurate durate duration a Structurate durate durat					10	滔	Cancel Build	Ctrl+.	
▷ 🔓 ad 1.5K 0.0K Set Active Build Configuration ↓ ▷ ⓑ be 1.3K 0.0K Batch Build ↓ ▷ ⓑ br 0.4K 0.0K Batch Build ↓ ▷ ⓑ filt 0.1K 0.0K Parallel and Unity Build ↓ ▷ ⓑ ga 0.3K 0.0K Show Build Log Ctrl+B, Ctrl+T r the names of its products derived free						.~.	Puild Configurations		ict or a software update
▷ ▲ be 1.3K 0.0K Batch Build adocumentation and/or ▷ ▲ br 0.4K 0.0K Batch Build adocumentation and/or ▷ ▲ filt 0.1K 0.0K Parallel and Unity Build adocumentation and/or ▷ ▲ ga 0.3K 0.0K Show Build Log Ctrl+B, Ctrl+T						4.02			
▷ br 0.4K 0.0K Batch Build , , , , , , , , , , , , , , , , , , ,							-		
▷ 월 ga 0.3K 0.0K or the names of its ▷ 월 me 0.7K 0.0K Show Build Log Ctrl+B, Ctrl+T			0.4K	0.0K				•	
▷ 월 ga 0.3K 0.0K Show Build Log Ctrl+B, Ctrl+T products derived free	\triangleright	🔒 filt	0.1K	0.0K			Parallel and Unity Build	•	a the second of the
							Show Build Log	Ctrl+B Ctrl+T	
kan ac 0.4K 0.0K ★ software without specific prior written permission.	⊳	🎦 me				*		,	

• Download projects

light_switch_provisioner_nrf52840_xxAA_s140_6.0.0 - SEGGER Embedded Studio for ARM V4.10a (64-bit) - Non-Commercial License

File	Edit	View	Search	Navigate	Project	Bui	ld Debug	Tai	get Tools Window Help		
Proje	ct Explo	rer		×	main.c			3	Connect J-Link	Ctrl+T, C	1
		_						*>	Disconnect J-Link	Ctrl+T, D	1
t De	ebug		•		$(\leftarrow \rightarrow$			3	Reconnect J-Link	Ctrl+T, E	
Projec	t Items		Code	Data ^			Copyright		Attach Debugger	Ctrl+T, H	
	Þ 🗖		3.7K 0.3K	1.4K 0.2K		*	All right		Reset J-Link	Ctrl+T, S	
			2.7K	1.2K		*	Redistri	-	Download light_switch_provisioner_nrf52840_xxAA_s140_6.0.0	Ctrl+T, L	thout mo
	⊳ 占] nrf	0.0K				are perm	J _A	Verify light_switch_provisioner_nrf52840_xxAA_s140_6.0.0	Ctrl+T, V	•
	▷ 🛃		2.3K 0.1K	1.8K 0.1K		*			Erase All	Ctrl+T, K	
			0.4K	0.0K			1. Redist list of (curri, k	right not
4	🔁 Be	arer	[7.7K]	[0.8K]		*	1150 01 1		Download File	•	
	Þ 👌		0.3K	0.0K	10	*	2. Redist		Verify File	•	Nordic
			0.2K 0.1K	0.0K		*	Semico	μZ	Start Cycle Counter		ftware up
	⊳ 6		1.5K	0.0K		*	such p condit		Pause Cycle Counter		this list
	⊳ 🛓		1.3K	0.0K		*	mater:	-72	Zero Cycle Counter	Ctrl+T, Z	ion and/o
	▷ 🛃		0.4K 0.1K	0.0K		*		r 🗗	Target Connection Properties		
	⊳ E		0.1K	0.0K		*	3. Neith			n . cne name	s of its

• You can also try to debug

👁 light_switch_provisioner_nrf52840_xxAA_s140_6.0.0 - SEGGER Embedded Studio for ARM V4.10a (64-bit) - Non-Commercial License (Stopped)

Project Explorer			×	main.c		Go	F5	
C Debug	- 1	(Accession)			н	Break	Ctrl+.	
Cebug			0	\$3 G		Stop	Shift+F5	
Project Items	Code	Data	^		+	Restart	Ctrl+Shift+F5	EVEL_INFO, "App key ", m_nw_:
b b main.c	3.7K	1.4	ĸ				7.2.20	L_INFO, "Press Button 1 to st
B mesh_softc	0.3K	0.2	ĸ	• •	9	Toggle Breakpoint	F9	
b 🛃 node_setup	2.7K	1.2	ĸ			Breakpoints	•	
h f nrf_mesh_v	0.0K				-			-
Provisioner	2.3K	1.8	К		•1	Step Into	F11	
Int_input.c	0.1K	0.1	к		Ç⊒	Step Over	F10	put_handler, RTT_INPUT_POLL_M
b Simple_hal.	0.4K	0.0		▶ 590	ςΞ	Step Out	Shift+F11	());
A 🔄 Bearer 12 files	[7.7K]	[0.8]		•	+1	Run To Cursor	Ctrl+F10	L_INFO, " <start> \n");</start>
ad_listener.	0.3K	0.0			81	Auto Step	Alt+F11	
ad_type_filt	0.2K	0.0		8	G =	Auto step	ALTETT	
adv_packet	0.1K	0.0			d.	Show Next Statement	Alt+*	
advertiser.c	1.5K	0.0		3	fI	Set Next Statement	Shift+F10	
bearer_han	1.3K	0.0	K		•=	Set Hext Statement	Shine+110	

Running example:

When the flashing is complete, the script executes a reset operation to start the example applications.

After the reset, the provisioner waits for user input. Follow these steps to see the mesh network in actions:

- 1. Press Button1 on the provisioner board to start the provisioning process
 - a) The provisioner first provisions and configures the client and assigns the address 0x100 to the client node
 - b) The two instances of the OnOff client models are instantiated on separate secondary elements. For this reason, they get consecutive addresses starting with 0x101
 - c) Finally, the provisioner picks up the available devices at random, assigns them consecutive addresses, and adds them to odd and even groups
 - d) Observe the LED status on the provisioner, client and server boards.
 - e) Wait until LED1 on the provisioner board remains ON steadily for a few seconds, which indicates that all available boards have been provisioned and configured
 - Press buttons on the client board to change the state of LED1 on the server boards:
 - i. Press Button1 on the client board to turn ON LED1 on all servers with ODD address

- ii. Press Button2 on the client board to turn OFF LED1 on all servers with ODD addresses
- iii. Press Button3 on the client board to turn ON LED1 on all servers with EVEN address
- iv. Press Button4 on the client board to turn OFF LED1 on all servers

with EVEN address

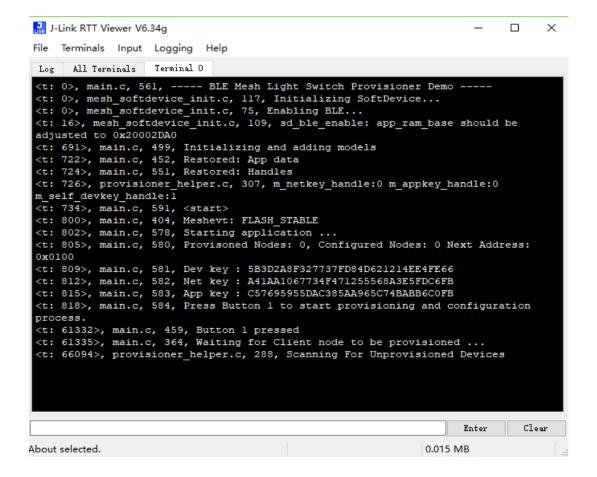
g) Press Button1 on the servers to locally change the state of LED1 and

observe that the client receivers the status message from the

corresponding server containing the new state value.

h) You can monitor the RTT logs with J-Link RTT viewer by connecting

provisioner and client.



User Phone as provisioner

Besides the NRF52840 board, you can also use your phone as provisioner.

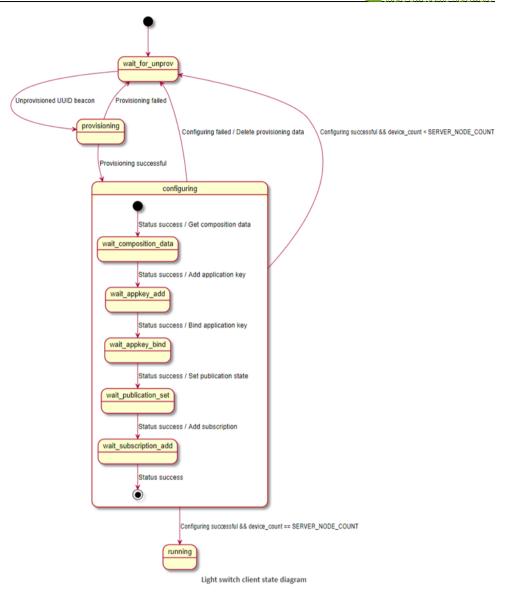
- 1. Download client and server project to Eval Kit board respectively.
- 2. Install nRF Mesh APP to your phone to work as a provisioner.
- 3. Connect to client and server
- 4. Configure client and server
- 5. Press Button1 on client board turns ON LED1 of server
- 6. Press Button2 on client board turns OFF LED1 of server

Provisioner:

The provisioner configures a network in a fixed, predefined way. it is implemented as a multi-layered state machine due to the asynchronous nature of the provisioning and configuration process.

The provisioner first provisions and configures a client device with a known URI hash. Then configures the servers and networking.





Client:

The light switch client implements a Generic OnOff client. Together with light switch server and mesh provisioner, it is part of the light switch example network demonstration, in which it has a provisioner role.

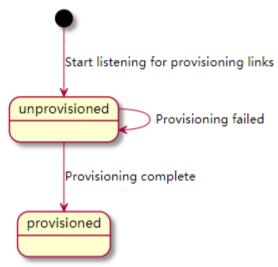
Server:

The light switch server is a Generic OnOff server that has a provisionee role in the light switch example network demonstration, which is also composed of light switch

client and mesh provisioner. There can be one or more servers in this network, for

example light bulbs.

The provisioner configures this server model instance to communicate with the client model.



State diagram for the Light switch server