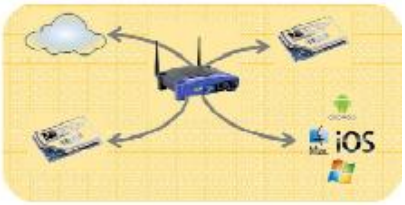
Name: ..... Student ID: .....
Name: ..... Student ID: .....

Laboratory 11
Interfacing to Wireless Modules

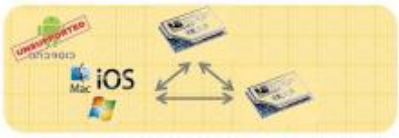
1. Introducing Wi-Fi

Wi-Fi is a local area wireless technology that allows an electronic device to exchange data or connect to the internet using 2.4 GHz UHF and 5 GHz SHF radio waves. Many devices can use Wi-Fi, e.g., personal computers, smartphones, digital cameras and digital audio players. These can connect to a network resource such as the Internet via a wireless network access point. Such an access point (or hotspot) has a range of about 20 meters indoors and a greater range outdoors. Hotspot coverage can comprise an area as small as a single room with walls that block radio waves, or as large as many square kilometres achieved by using multiple overlapping access points. Wi-Fi can be less secure than wired connections (such as Ethernet) because an intruder does not need a physical connection. Web pages that use SSL are secure but unencrypted internet access can easily be detected by intruders. Because of this, Wi-Fi has adopted various encryption technologies. The early encryption WEP, proved easy to break. Higher quality protocols (WPA, WPA2) were added later. An optional feature called Wi-Fi Protected Setup (WPS), had a serious flaw that allowed an attacker to recover the router's password.

- **Infrastructure**
  - Client nodes communicate via an access point
  - Most common, like connecting your PC to a home network



- **Adhoc**
  - Point-to-Point connection
  - Every node connected to every other node
  - Android unsupported



- **Soft AP**
  - Module looks like an Access point
  - AP module is central coordinator
  - Basic networkmanagement
  - DHCP, Routing, Gateway redirection

Fig.1. Wi-Fi networking modes

2. RN-171-EK evaluation board

The RN-171-EK evaluation board is a field-ready, Wi-Fi Alliance certified, 802.11 prototyping platform. The board has the flexibility to connect directly to PCs via a standard USB interface or to embedded processors through the TTL UART interface. The RN-171-EK contains a battery boost circuit that makes the RN-171-EK perfect for battery powered applications such as sensors, data acquisition systems, controllers, etc. The module incorporates full TCP/IP stack, real-time clock, and supports the FTP client, DHCP, DNS, and HTML client protocols. The module supports ad hoc and infrastructure networking modes. The analog sensor interface provides direct connections to read the sensors' acceleration and other analog data without requiring an external microprocessor. Once the configuration is set up, the module can automatically scan to find an access point, associate, authenticate, and connect over any Wi-Fi network. Additionally, the module can automatically wake up, send data to a remote host, and go back to a low-power sleep state.

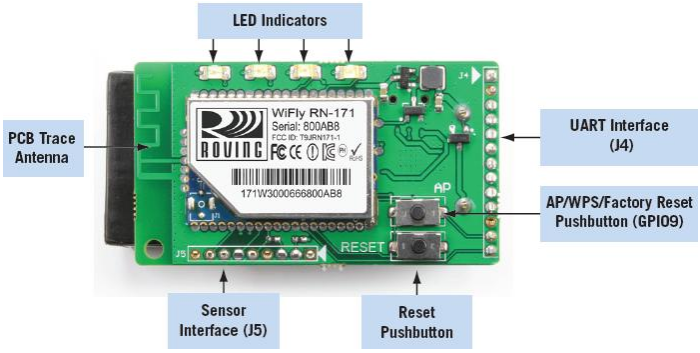


Fig.2. Roving Networks RN-171-EK Evaluation Board

3. Basic Startup Procedure

3.1 Connect Evaluation Board

- Connect battery
- Connect board to your computer and use PuTTY or any Terminal Emulators. Green LED blinks slowly
- Use device manager to find the COM port

3.2 Launch Command Mode

- Open assigned COM port 9600 baud, 8 bits, No Parity, 1 stop bit
- Type **$$$** Module responds with **<CMD>**
- Review & Reset Configuration
- Check configuration & firmware version: **get ver**
- Perform factory reset:
  factory RESET (factory R)
- **reboot**

3.3 Search for wireless networks

- Type **$$$**
- Type **scan**

3.4 Auto-Join a network with persistent configuration

- **set wlan ssid <String>**
- **set wlan phrase <String>**
- **set wlan join 1**
- **save**
- **reboot**

List of command and its description

Command	Default	Description
set apmode beacon <*value*>	102	Sets the apmode beacon interval in milliseconds.
set apmode link monitor <*value*>	3600	This command is used in soft AP mode to detect if the individual client devices are active and in range of the module.
set apmode passphrase <*string*>	NULL	This command sets the soft AP mode passphrase to be used for WPA2-AES encryption.
set apmode probe <*value*>	5	Sets the apmode probe timeout in seconds (apmode mode only).
set apmode reboot <*value*>	0	Sets the reboot timer.
set apmode ssid <*string*>	NULL	This command sets the soft AP mode network name (SSID) to be broadcast where <*string*> is the SSID.
set broadcast address <*address*>	255.255.255.255	Sets the address to which the UDP hello/heartbeat message is sent.
set broadcast backup <*address*>	0.0.0.0	Sets the secondary broadcast backup address.
set broadcast interval <*mask*>	7	Sets the interval (in seconds) at which the hello/heartbeat UDP message is sent.
set broadcast port <*value*>	55555	Sets the port to which the UDP hello/heartbeat message is sent.
set broadcast remote <*port*>	0	Sets the secondary broadcast port.
set comm $ <*char*>	$	Sets character used to enter command mode to <*char*>.
set comm close <*string*>	*CLOS*	Sets the ASCI string that is sent to the local UART when the TCP port is closed.
set comm idle <*value*>	0	Sets the idle timer value in seconds.
set comm match <*value*> | <*hex*>	0	Sets the match character in hex or decimal.
set comm open <*string*>	*OPEN*	Sets the ASCI string that is sent to the local UART when the TCP port is opened.
set comm remote <*string*>	*HELLO*	Sets the ASCI string that is sent to the remote TCP client when the TCP port is opened.
set comm size <*value*>	64	Sets the flush size in bytes.
set comm time <*value*>	5	Sets the flush timer.
set dhcp lease <*value*>	86400	Sets the soft AP mode DHCP lease time in seconds.
set dns address <*address*>	0.0.0.0	Sets the IP address of the DNS sever.
set dns backup <*string*>	rn.microchip.com	Sets the name of the backup host for TCP/IP connections to <*string*>.
set dns name <*string*>	server1	Sets the name of the host for TCP/IP connections to <*string*>.
set ftp addr <*address*>	0.0.0.0	Sets the FTP server's IP address of the FTP server.
set ftp dir <*string*>	public	Sets the starting directory on the FTP server.

set ftp filename <*filename*>	See description	Sets the name of the file that is transferred when issuing the **ftp u** command, where <*filename*> is the firmware image. Firmware version 4.0 default is wifly3-<*version*>.img (RN131) wifly7-<*version*>.img (RN171). Firmware prior to 4.0 default is wifly-GSX-<*version*>.img (RN131) wifly-EZX-<*version*>.img (RN171).
set ftp pass <*string*>	Pass123	Sets the password for accessing the FTP server.
set ftp mode <*mask*>	0x0	Sets the ftp mode, where <*mask*> indicates active or passive mode. Default is passive.
set ftp remote <*value*>	21	Sets the FTP server's remote port number.
set ftp time <*value*>	200	Sets the FTP timeout value, where <*value*> is a decimal number that is five times the number of seconds required.
set ftp user <*string*>	roving	Sets the user name for accessing the FTP server.
set ip address <*address*>	0.0.0.0	Sets the WiFly module's IP address.
set ip backup <*address*>	0.0.0.0	Sets a secondary host IP address.
set ip dhcp <*value*>	1	Enables/disables DHCP mode.
set ip flags <*mask*>	0x7	Sets the TCP/IP functions.
set ip gateway <*address*>	0.0.0.0	Sets the gateway IP address.
set ip host <*address*>	0.0.0.0	Sets the remote host's IP address.
set ip localport <*value*>	2000	Sets the local port number.
set ip netmask <*address*>	255.255.255.0	Sets the network mask.
set ip protocol <*flag*>	2	Sets the IP protocol.
set ip remote <*value*>	2000	Sets the remote host port number.
set ip tcp-mode <*mask*>	0x0	Controls the TCP connect timers, DNS preferences, and remote configuration options.
set opt average <*value*>	5	Sets the number of RSSI samples used to calculate the running RSSI average.
set opt deviceid <*string*>	WiFly-*XXX*	Sets the configurable device ID, where *XXX* is GSX for the RN131 and EZX for the RN171.
set opt format <*flag*>	0x00	Sets the HTTP client/web server information.
set opt jointmr <*value*>	1000	Sets the join timer, which is the length of time (in ms) the join function waits for the access point to complete the association process.
set opt replace <*char*>	$ (0x24)	Sets the replacement character you use to indicate spaces in the SSID and passphrases, where <*char*> is a single character.
set opt password <*string*>	"" (no password required)	Sets the TCP connection password.
set opt signal <*value*>	0	Configures the threshold level for the RSSI value in infrastructure mode.
set q power <*value*>	0	Automatically turns on the sensor power.
set q sensor <*mask*>	0	Specifies which sensor pins to sample when sending data using the UDP broadcast packet or the HTTP auto sample function.
set sys autoconn <*value*>	0	Sets the auto-connect timer in TCP mode.
set sys autosleep <*value*>	0	Sets the auto-sleep timer in UDP mode.
set sys iofunc <*mask*>	0x0	Sets the I/O port alternate functions.
set sys launch_string <*string*>	web_app	Sets the application to launch when GPIO9 is high after power up.

set sys mask <*mask*>	0x20F0 (RN131) 0x21F0 (RN171)	Sets the I/O port direction.
set sys printlvl <*value*>	0x1	Controls the debug print messages printed by the WiFly module on the UART.
set sys output <*mask*> <*mask*>	None	sets the output GPIO pins high or low. The optional <*mask*> sets a subset of the pins.
set sys sleep <*value*>	0	Sets the sleep timer.
set sys trigger <*flag*> *or* <*mask*>	0x1	With this parameter setting, the module wakes from sleep state using the sensor input 0, 1, 2, and 3.
set sys value <*mask*>	0x0	Sets the default value of the GPIO pins' outputs upon power-up.
set sys wake <*value*>	0	Sets the automatic wake timer in seconds.
set time address <*address*>	64.90.182.55	Sets the time server address.
set time enable <*value*>	0	Tells the module how often to fetch the time from the specified SNTP time server in minutes.
set time port <*value*>	123	Sets the time server port number.
set time raw <*value*>	None	Allows you to set the RTC raw value from the console in seconds.
set uart baud <*value*>	9600	Sets the UART baud rate, where <*value*> is 2400, 4800, 9600, 19200, 38400, 57600, 115200, or 230400.
set uart flow <*value*>	0	Sets the flow control mode and parity.
set uart instant <*value*>	Not applicable	Immediately changes the baud rate, where <*value*> is 2400, 4800, 9600, 19200, 38400, 57600, 115200, or 230400.
set uart mode <*mask*>	0	Sets the UART mode register.
set uart raw <*value*>	Not applicable	Sets a raw UART value.
set uart tx <*value*>	Not applicable	Disables or enables the UART's TX pin (GPIO10), where <*value*> is 1 or 0.
set wlan auth <*value*>	0	Sets the authentication mode.
set wlan channel <*value*> <*flag*>	0	Sets the WLAN channel, where <*value*> is a decimal number from 1 to 13 representing a fixed channel and <*flag*> is the optional character **i** (meaning immediate).
set wlan ext_antenna <*value*>	0	Determines which antenna is active, where <*value*> is 0 (use the chip antenna) or 1 (use the U.FL connector).
set wlan fmon <value>	3600	Sets the soft AP mode link monitor timeout threshold for the associated client device.
set wlan id <*string*>	–	Reserved for future use.
set wlan hide <*value*>	0	Hides the WEP key and WPA passphrase, where <*value*> is 0 or 1.
set wlan join <*value*>	1|0	Sets the policy for automatically associating with network access points.
set wlan key <*value*>	Not applicable	Sets the 128-bit WEP key, where <*value*> is EXACTLY 26 ASCII chars (13 bytes) in hex without the preceding 0x.
set wlan linkmon <*value*>	0 (disabled)	Sets the link monitor timeout threshold, where <*value*> is a decimal number representing the number of failed scans before the module declares **AP is Lost** and de-authenticates.
set wlan mask <*mask*>	0x1FFF (all channels)	Sets the WLAN channel mask, which is used for scanning channels with auto-join policy 1 or 2).
set wlan phrase <*string*>	rubygirl	Sets the passphrase for WPA and WPA2 security modes.
set wlan number <*value*>	0	Sets the WEP key number.

set wlan rate <*value*>	12	Sets the wireless data rate.
set wlan ssid <*string*>	roving1	Sets the SSID with which the module associates.
set wlan tx <*value*>	0	Sets the Wi-Fi transmit power, where <*value*> is a decimal number from 1 to 12 that corresponds to 1 to 12 dBm.
set wlan user <*string*>	–	Reserved for future use.

TABLE D-2: GET COMMANDS

Command	Description
get apmode	Displays the Soft AP mode settings.
get broadcast	Displays the broadcast UPD address, port, and interval.
get com	Displays the communication settings.
get dns	Displays the DNS settings.
get everything	Displays all of the configuration settings, which is useful for debugging.
get ftp	Displays the FTP settings.
get ip <*char*>	Displays the IP address and port number settings, where <*char*> is the optional parameter **a**. Using <*char*> returns the current IP address.
get mac	Displays the device's MAC address.
get option	Displays the optional settings such as the device ID.
get sys	Displays the system settings, sleep and wake timers, etc.
get time	Displays the time server UDP address and port number.
get wlan	Displays the SSID, channel, and other WLAN settings.
get uart	Displays the UART settings.
ver	Displays the firmware version.

TABLE D-3: STATUS COMMANDS

Command	Description
show battery	Displays current battery voltage, and is only applicable to Microchip's battery-powered products such as the RN370 and temperature sensors (ISENSOR-CB).
show connection	Displays the connection status in the hex format 8<*XYZ*>.
show io	Displays the GPIO pins' level status in the hex format 8<*ABC*>.
show net <*char*>	Displays the current network status, association, authentication, etc., where <*char*> is the optional parameter **n**. Using the **n** parameter displays only the MAC address of the access point with which the module is currently associated.
show q <*value*>	Displays the value of the analog interface pin, where <*value*> is 0 to 7.
show q 0x1<*mask*>	Displays multiple analog interface values simultaneously.
show rssi	Displays the last received signal strength.
show stats	Displays the current statistics, packet RX/TX counters, etc.
show time	Displays the number of seconds since the module was last powered up or rebooted.

TABLE D-4: ACTION COMMANDS

Command	Description
$$$	Use this command to enter Command mode.
apmode <*bssid*> <*channel*>	Creates a soft AP network.
close	Disconnects a TCP connection.
exit	Exits command mode.

factory RESET	Loads the factory defaults into the module's RAM and writes the settings to the standard configuration file. You must type the word **RESET** in capital letters.
join <*string*>	Instructs the WiFly module to join the network indicated by <*string*>.
join # <*value*>	Use this command to join a network that is shown in the scan list, where <*value*> is the entry number listed for the network in the scan list.
leave	Disconnects the module from the access point to which it is currently associated.
lookup <*string*>	Causes the module to perform a DNS query for host name <*string*>.
open <*address*> <*value*>	Opens a TCP connection to <*address*>, where <*value*> is the port number.
ping <*string*> <*value*>	Pings a remote host, where <*string*> is a parameter setting and <*value*> is the number of pings. The default is 1 packet.
reboot	Forces the module to reboot (similar to a power cycle).
run	Runs an application using ASCII commands.
scan <*value*> <*char*>	Performs an active probe scan of access points on all 13 channels. The default is 200 ms/channel.
sleep	Puts the module to sleep.
time	Sets the real-time clock by synchronizing with the time server specified with the time server (**set time**) parameters.

4. Experiment

1. Configure two WiFi modules according to procedure in topic 3. Set a Smart Phone to be a WiFI Hot spot. Use command **show net** to show its current network status

2. Communicate between two WiFi modues that shares the same SSID by using command
`open <LAB PARTNER IP> 2000`

3. Connect a Module to a Xpresso boards RN-171EK using Asynchronous Port. Then use command **`get everything`**. Show the result on your console

```
Serial async_port(P0_11, P0_10);    // Tx, Rx
int main() {
      char buff[1024];
      async_port.baud(9600); //set baud rate to 9600, 8 bits, no parity
      while (1){
        async_port.printf("$$$");
        if (async_port.readable()==1) //is there a character to be read?
            int recd_val=async_port.scanf("%s", buff);
        TFT.printf("%s", buff);
      }
}
```

RN-171 UART Interface	Xpresso pin
Pin 1 3.3VDD	VDD
Pin 2 GND	GND
Pin3 UART RX	P0_11 TX
Pin4 UART TX	P0_10 RX

5. Introduction to Bluetooth Smart

Bluetooth low energy or **Bluetooth LE**, marketed as **Bluetooth Smart** provides wireless data links between such devices as mobile phones, computers, wireless audio headsets and systems requiring the use of remote sensors. Bluetooth Smart's main characteristics can be summarised as follows:

Technical Specification	Classic Bluetooth technology	Bluetooth Smart technology
Distance/Range (theoretical max.)	100 m (330 ft)	<100 m (<330 ft)
Over the air data rate	1–3 Mbit/s	1 Mbit/s
Application throughput	0.7–2.1 Mbit/s	0.27 Mbit/s
Active slaves	7	Not defined; implementation dependent
Security	56/128-bit and application layer user defined	128-bit AES with Counter Mode CBC-MAC and application layer user defined
Robustness	Adaptive fast frequency hopping, FEC, fast ACK	Adaptive frequency hopping, Lazy Acknowledgement, 24-bit CRC, 32-bit Message Integrity Check
Latency (from a non-connected state)	Typically 100 ms	6 ms
Total time to send data (det.battery life)	100 ms	3 ms
Voice capable	Yes	No
Network topology	Scatternet	Scatternet
Power consumption	1 W as the reference	0.01 to 0.5 W (depending on use case)
Peak current consumption	<30 mA	<15 mA
Service discovery	Yes	Yes
Profile concept	Yes	Yes

The Bluetooth standards dictate that when Bluetooth devices detect one another, they determine automatically whether they need to interact with each other. Each device has a Media Access Control (MAC) address which communicating devices can recognise and initialise interaction if required. Bluetooth systems in contact with each other form a *piconet*. Bluetooth has many valuable applications where wires are intrusive, expensive or difficult to install. Recent enhancements have enabled streaming of high quality audio data and increased range, so the opportunities and applications for Bluetooth are continuously growing.

6. Nordic nRF51822

The nRF51822-mKIT is a low cost ARM mbed enabled development board for Bluetooth® Smart designs with the nRF51822 system-on-a-chip (SoC). The Nordic nRF51822 combines a Bluetooth v4.1-compliant 2.4GHz multiprotocol radio and an ARM® Cortex®-M0 processor on a single chip optimized for ultra-low power operation. It has 31 GPIO's accessible on pin header. Its peripheral interface includes Analog In, SPI, I2C, and UART. The kit gives access to all GPIO pins via pin headers and incorporates a coin-cell battery holde. Bluetooth Smart is quickly becoming a key communication component for IoT devices and it's already supported in modern smartphones and tablets. It is designed for enabling short-range wireless connectivity to things like coin cell-powered accessories. This opens the door to things like Appcessories and a whole host of applications for interacting and configuring devices, where you can embed a Bluetooth Smart chip and bring your own device (BYOD).

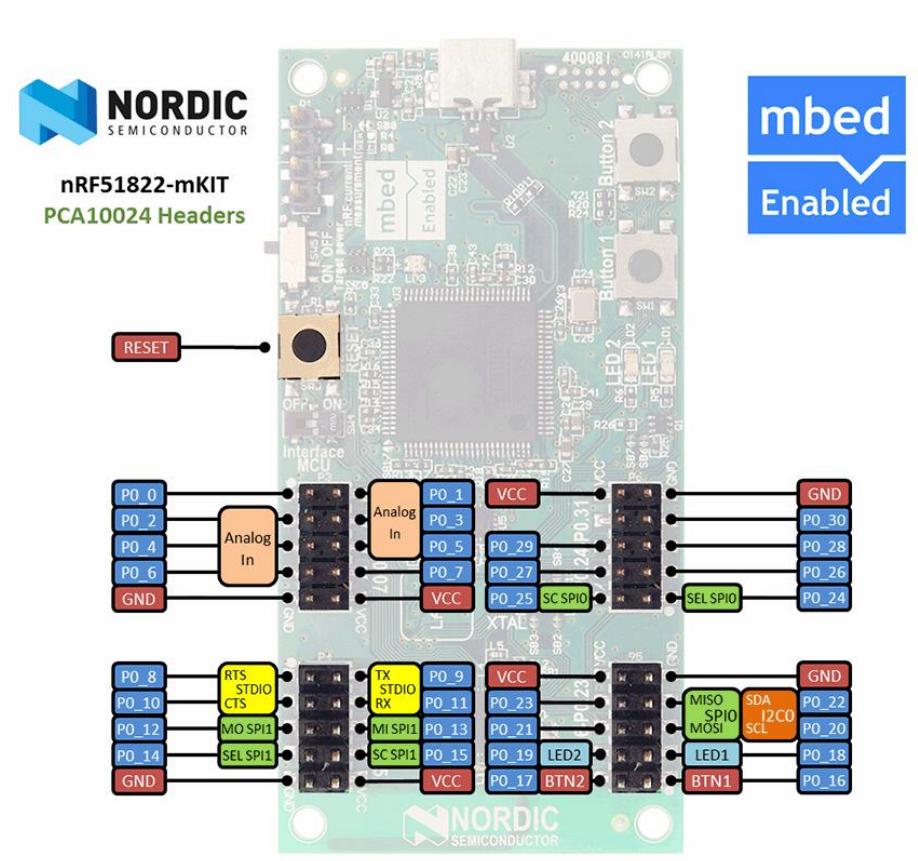
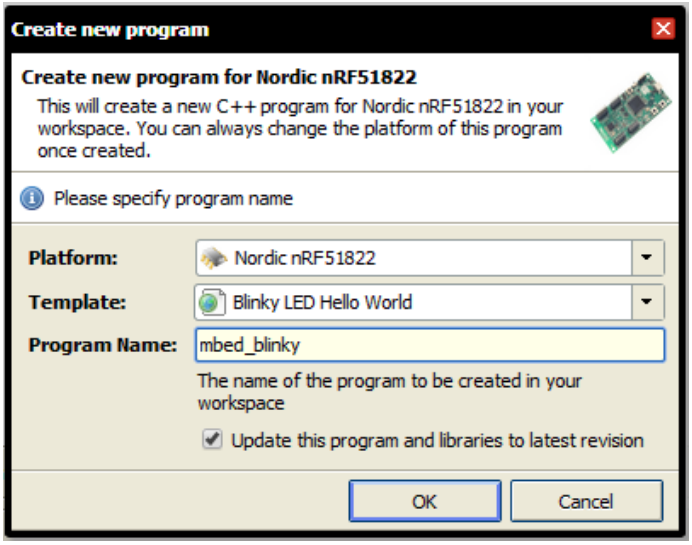


Fig. 1. nRF51822 development board. It shows a map of GPIO pin headers and its alternate function

7. Interfacing to a Bluetooth module

In the mbed platform, its tool chain and development system are online accessible through ARM mbed developer site (developer.mbed.org). To develop an application, we first write a program in the online workspace, compile it online and then download a .hex file and flash to the mbed board.

1) Go to http://developer.mbed.org/ and signup
2) Plug in the development board. Your PC will recognize it as a **mbed drive**. Click on mbed.htm, it will open a new page for platforms Nordic
3) On the right side of this new page click on "Open mbed Compiler" to create new program



Click OK. Now you have add a program named mbed_blinky into your workspace.
4) Compile your program by click at "Compile" icon of the IDE
5) Once it finishes compiling, the compiler output (.hex file) will be automatically downloaded to your PC.
To program the board, simply copy (drag and drop) the .hex file to the mbed drive. Note that the LED3 is blinking very fast during the programming. It then turn on without blinking after finishing the programming.
6) Run the program by pressing the "RESET" button (see Fig.1). Note that when running the program LED1 is blinking.

You can create new program by click on "New" icon. You can also import programs and libraries by click on "Import" icon.

8. Experiment

A good application to start with is the BLE_HeartRate example, not too complicated, and it includes data transfer which is normally required for any application.

Program this example to your nRF51822 mbed enabled kit by doing the following:

1. Go to the `http://developer.mbed.org/teams/Bluetooth-Low-Energy/code/BLE_HeartRate/` then click Import this program.
2. Click through the options and make sure the BLE_HeartRate project is selected.
3. Click Compile. A .hex file should download in your browser.
4. Have your nRF51822 mbed kit connected; it should appear as a USB mass storage device. Drag or copy the file to this device. It programs and you are done on this side.

The easiest way to test the application is to use a phone app, Nordic Semiconductor has several examples for iOS and Android `http://www.nordicsemi.com/eng/Products/nRFready-Demo-APPS.`

To connect with nRF Toolbox do the following:

1. Start the nRF Toolbox app.
2. Click the Heart/HRM icon.
3. Click Connect.
4. Look for Nordic_HRM (the default name) in the list. Touch it to select.
5. Observe the data in the graph.