

MATRIX | EBLOCKS 2

Bluetooth Communications



CP1795

MATRIX
www.matrixtsl.com

Copyright © 2018 Matrix Technology Solutions Limited

CP1795

Bluetooth Communication

Course notes

Contents

1	Introduction	6	4.2.2	Bluetooth layers	16
1.1	Structure of these notes	6	4.2.3	Hardware details	17
1.2	Learning outcomes	6	4.2.4	Profiles	19
1.3	Using this manual	7	5	Discovery	20
1.3.1	An introduction to the Practical implementation sections	7	5.1	Theory: Finding other Bluetooth devices	20
2	What do we mean by Bluetooth?	8	5.1.1	The Inquiry commands	20
2.1	The RN4678 Bluetooth module	8	5.1.2	Additional Inquiry parameters	20
2.2	Command mode	8	5.1.3	Discovery example	21
2.3	Hardware and software used in the course	8	5.2	Exercise 1: Discovering Bluetooth devices	22
2.3.1	E-blocks2 parts	8	5.2.1	Introduction	22
2.3.2	Flowcode	9	5.2.2	Objectives	22
2.4	Additional useful devices	9	5.2.3	Pre-requisites	22
2.4.1	Onboard Bluetooth/USB dongle	9	5.2.4	Hardware/Software requirements	22
3	Getting started	10	5.2.5	Exercise information	22
3.1	Setting up the hardware	10	5.2.6	Learning outcome	23
3.1.1	E-blocks2 setup	10	5.2.7	Additional tasks	23
3.2	Introduction to Flowcode	11	5.3	Practical implementation: Discovering Bluetooth devices	23
3.2.1	Starting out in Flowcode	11	5.3.1	Discovering and discoverable	23
3.3	The Bluetooth component	11	5.3.2	Planning the program	24
3.3.1	Properties and pin connection	12	5.3.3	Required macros	24
3.3.2	Bluetooth component macros	12	5.3.4	Initialising	25
3.4	Testing the hardware	12			
4	Bluetooth theory and background	13			
4.1	Introduction to Bluetooth	13			
4.1.1	Brief history	13			
4.1.2	Bluetooth concepts	13			
4.1.3	Bluetooth advantages	14			
4.1.4	Bluetooth disadvantages	14			
4.2	Protocols and the OSI model	14			
4.2.1	Application	16			

6.3.4 Running and demonstrating the program33

7 Connecting Bluetooth devices34

7.1 Theory: Connecting – Addresses34

7.2 Exercise 3: Connecting to a device35

7.2.1 Introduction35

7.2.2 Objectives35

7.2.3 Pre-requisites35

7.2.4 Hardware/Software requirements35

7.2.5 Exercise information35

7.2.6 Learning outcome35

7.2.7 Further work35

7.3 Practical implementation: Connecting36

7.3.1 Resetting the systems36

8 Passkeys and Connecting37

8.1 Theory: Passkeys and Connecting37

8.1.1 Sending the Passkey command37

8.1.2 Initiating pairing and connection37

8.2 Exercise 4: Passkeys and Connecting38

8.2.1 Introduction38

8.2.2 Objectives38

8.2.3 Pre-requisites38

8.2.4 Hardware/Software requirements38

8.2.5 Exercise information38

8.2.6 Learning outcome38

8.2.7 Further work38

8.3 Practical implementation: Passkeys and Connecting39

8.3.1 The basic program39

8.3.2 Advanced features: Choosing what device to connect to40

8.3.3 Resetting the systems40

9 Checking responses41

5.3.5 Sending a command25

5.3.6 Checking for responses26

5.3.7 Running the complete program27

6 Discoverability28

6.1 Theory: Discoverability28

6.1.1 Configuring for start up28

6.1.2 Using Set Commands28

6.1.3 Using Get Commands28

6.1.4 Using Action Commands28

6.1.5 Making a device discoverable29

6.1.6 Enter command mode29

6.1.7 Wait for the command to execute29

6.1.8 Baud rate and settings29

6.1.9 Authentication mode29

6.1.10 Slave mode29

6.1.11 Activating the updated configuration29

6.1.12 Wait for reboot and leave command mode29

6.1.13 A summary of the basic configuration set up commands29

6.2 Exercise 2: Discoverability30

6.2.1 Introduction30

6.2.2 Objectives30

6.2.3 Pre-requisites30

6.2.4 Hardware/Software requirements30

6.2.5 Exercise information30

6.2.6 Learning outcome30

6.2.7 Additional tasks30

6.3 Practical implementation: Discoverability31

6.3.1 Continuing development31

6.3.2 Configuring the Bluetooth device31

6.3.3 Creating the program32

9.1 Theory: Checking responses	41	11.2 Exercise 7: Trust and Security	50
9.1.1 Solicited and unsolicited responses	41	11.2.1 Introduction	50
9.1.2 Response handling macros	41	11.2.2 Objectives	50
9.2 Exercise 5: Checking responses	43	11.2.3 Pre-requisites	50
9.2.1 Introduction	43	11.2.4 Hardware/Software requirements	50
9.2.2 Objectives	43	11.2.5 Exercise information	50
9.2.3 Pre-requisites	43	11.2.6 Learning outcome	50
9.2.4 Hardware/Software requirements	43	11.3 Practical implementation: Trust and Security	51
9.2.5 Exercise information	43	11.3.1 General objectives	51
9.2.6 Learning outcome	43	11.3.2 Additional features	51
9.3 Practical implementation: Checking responses	44	12 Project design principles	52
9.3.1 Using the WaitForStringValue macro	44	12.1 Project – Simple remote control	52
9.3.2 Error checking methodology	44	12.1.1 Extending the project	52
10 Introduction to Bluetooth Low Energy (BLE)	45	12.2 Project - Medical datalogger	52
10.1 Bluetooth Low Energy (BLE) modes of operation	45	12.2.1 Extending the project	52
10.2 Exercise 6: BLE GAP transparent UART serial data service	46		
10.2.1 Introduction	46		
10.2.2 Objectives	46		
10.2.3 Pre-requisites	46		
10.2.4 Hardware/Software requirements	46		
10.2.5 Exercise information	46		
10.2.6 Learning outcome	46		
10.3 Practical implementation: BLE serial data service	47		
10.3.1 BLE mode	47		
10.3.2 Secure Simple Pairing	47		
10.3.3 Activating settings	47		
11 Trust and Security	48		
11.1 Theory: Trust and Security	48		
11.1.1 Security in general	48		
11.1.2 Security modes	49		
11.1.3 Authentication	49		
11.1.4 Trust	49		

Introduction

Structure of these notes

These notes are set out as follows:

Part 1: General introduction to Bluetooth and the Bluetooth communication kit

- Getting started – introduction to the hardware and software

Bluetooth history and general overview

Part 2: The course

A series of progressive exercises to take students through the concepts and practice required in establishing Bluetooth communications.

The chapters are broken down into 3 sections for each chapter.

- Theory section
- Exercise description

Practical notes

Part 3: References and Appendix

Ancillary chapters providing reference information.

Includes:

- Reference section

A “How to” Programming reference section

Learning outcomes

These teacher’s notes are designed to introduce the concepts and strategies required for practical Bluetooth communications. In completing the exercises in this course students will learn about the following:

- How one Bluetooth device discovers another Bluetooth device and the options concerning discoverability
- How Bluetooth devices pair and set up a communications channel
- How data of various kinds is transferred between Bluetooth devices

How trust and security are handled by Bluetooth.

These notes will not address the radio frequency characteristics and low level transmission characteristics and protocols of Bluetooth. These are all handled by an off the shelf Bluetooth module which shields users from such issues.

These notes are structured into a number of sections that first take you through setting up, configuring and testing the hardware and software into the background of Bluetooth and then into a series of Exercises and examples that take the student through the workings of Bluetooth.

The exercises should be carried out using Flowcode V10 or later, a graphical programming language. The Flowcode Bluetooth component is designed to allow students to learn about Bluetooth without getting bogged down with the problems of programming in C or a lower level language.

Using this manual

The main part of this manual is structured around a three part approach:

- **Theory.** The first part of a chapter introduces the topic at hand and discusses the theory behind it, explains the commands used and the general sequences and strategies required.
- **Exercise.** The second element is an exercise. The exercise is given here so that the aims and objectives are understood and borne in mind whilst reading through the next section.

Practical implementation. The third section discusses the practical implementation of the exercise. Items discussed here include Flowcode macros used, Flowcode strategies and any other bits of information or advice needed for implementing the exercise objectives.

It is intended that students should first have the theory explained to them in the form of a lecture or handout. Students can then be given the exercise. Supervisors have a choice whether the practical implementation notes are handed out or not.

We would suggest that for each exercise the students are given the Exercise sheet(s) and the Practical implementation notes. Students should build the programs in Flowcode. Initial Practical implementation notes are quite detailed and provide a good deal of information on how the program should be constructed. Later Exercises are not provided with such detailed Practical implementation notes: the student must then use his/her knowledge to complete the tasks detailed in the Exercise.

The Bluetooth module datasheet is a key part of the Bluetooth solution: students will be expected to look up the meaning of the commands and functions of the Bluetooth module. Accordingly students should be given a copy of the Bluetooth module user guide which is provided in the CP1795 Resources zip file.

For most of the exercises a complete set of solution programs are provided. These take the form of two programs – one for each Bluetooth node. Note that these examples will require the changing any device IDs used within the program.

An introduction to the Practical implementation sections

This first practical section is in the form of a worked example with all the practical information provided and an example program built. Later practical sections will contain the additional information required to accomplish the exercise objectives, and will contain code snippets for new procedures and macros, but will not necessarily include full example programs. The students are assumed at this stage to be competent enough to be able to create the programs given the practical section information supplied.

It is intended that wherever possible the programs created by the student for the previous exercise are re-used for the current exercise. This illustrates both the evolution of the program as new steps are added, and to show the growing complexity and program flow of a full working system. This approach also provides students with core code that they are already familiar with. However, the Student programs may need to be assessed at various points to ensure the code is adequate for the current project, and modifications or comments made as appropriate to steer student programs through the course as a whole. Such modifications may take the form of adapting parts of programs to macros to unclutter parts of the program, or for monitored and assessed code rewrites to implement element in a more organized and efficient manner.

What do we mean by Bluetooth?

Bluetooth can be accessed at a number of levels. Bluetooth is both a communications technology, and a communications strategy. As a communications technology Bluetooth transforms data from an application to radio signals and back again. As a communications strategy Bluetooth is about discovering and linking to other Bluetooth devices, and accessing features on those devices.

This course will concentrate on the communications strategies of Bluetooth. This is when the end user will experience Bluetooth technologies, such as sending data between devices.

The Bluetooth communications technology is generally handled automatically by the Bluetooth devices. Unless one is actively designing Bluetooth chips the lower communications layers will remain hidden from most end users. Theory and background has been provided to show how Bluetooth handles communications.

The RN4678 Bluetooth module

The device used on the Matrix Bluetooth BL0170 board is a Microchip RN4678 device. The device is a self-contained Bluetooth module that can be communicated with using a set of commands over a serial UART. This device allows users to connect to and use other Bluetooth devices at the application level. Using this device allows the course to concentrate of the strategies and sequences of communicating between Bluetooth devices without needlessly delving into the lower technology levels.

Command mode

RN4678 operates in two modes: Data mode (default) and Command mode. When

RN4678 is connected to another device and in Data mode, RN4678 acts as a data pipe:

anything received from UART is passed to the connected peer device through SPP if

connected to a Bluetooth Classic device, or via a private GATT service if connected to

a BLE device. When data is received from the peer device from SPP for Bluetooth

Classic or UART Transparent for BLE, such data outputs directly to UART.

RN4678 is configured or controlled, or both by setting it into Command mode and

executing ASCII commands over UART.

..

[Hardware and software used in the course](#)

The Bluetooth course makes use of the following Hardware and Software:

E-blocks2 parts

The E-Blocks2 training kit will contain two complete kits of the following components:

- BL0011 PIC programmer or BL0055 Arduino programmer
- BL0170 Bluetooth board

BL0169 LCD Display

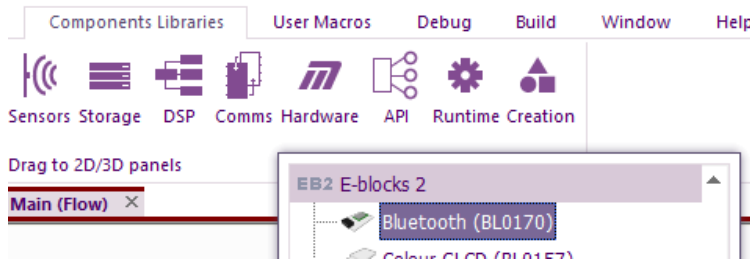
And one each of

- BL0145 Switch board
- BL0167 LED board

Flowcode

The example programs used in this course require Flowcode V10 or later.

For Flowcode V10, the **Bluetooth BL0170** component can be found under **E-blocks2** in the **Hardware** section of the **Component Libraries** toolbar:



All example and exercise program files supplied in the CP1795 Resources zip file have been created for Flowcode V10 or later.

This course assumes a degree of familiarity with Flowcode. If necessary, time should be spent using the tutorials and the Flowcode course to familiarize the student with using Flowcode.

Additional useful devices

The following devices are not supplied with the Bluetooth solution, however if they are available, they can be used in conjunction with the Bluetooth solution

Onboard Bluetooth/USB dongle

For communications to and from the PC a Bluetooth module is needed. Some recent laptops and PCs are already wired for Bluetooth. Most current PCs and Laptops though will require a Bluetooth module that plugs into a spare USB port and allow the computer to function as a Bluetooth device.

Note that it is not necessary to have a Bluetooth module on your PC to complete all exercises.

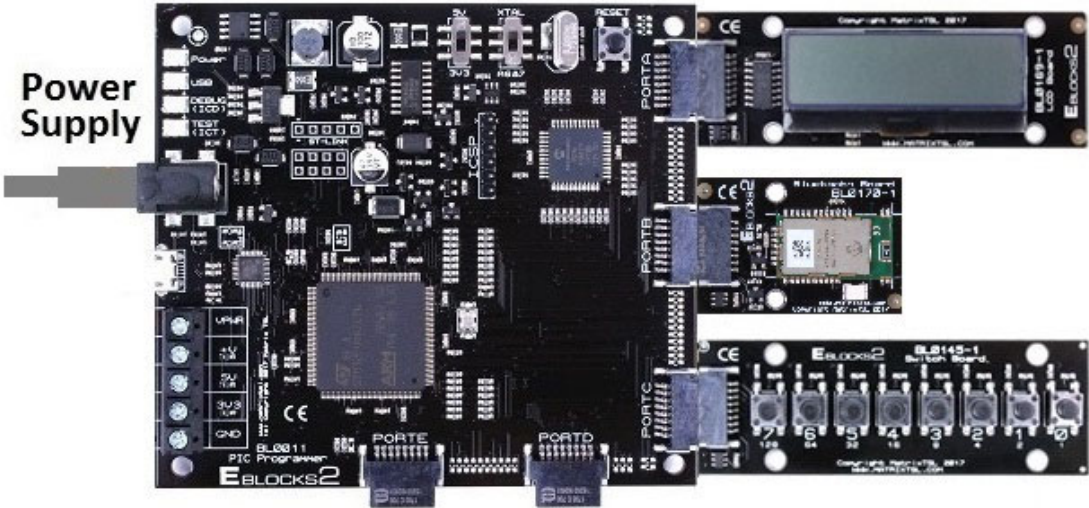
Getting started

Setting up the hardware

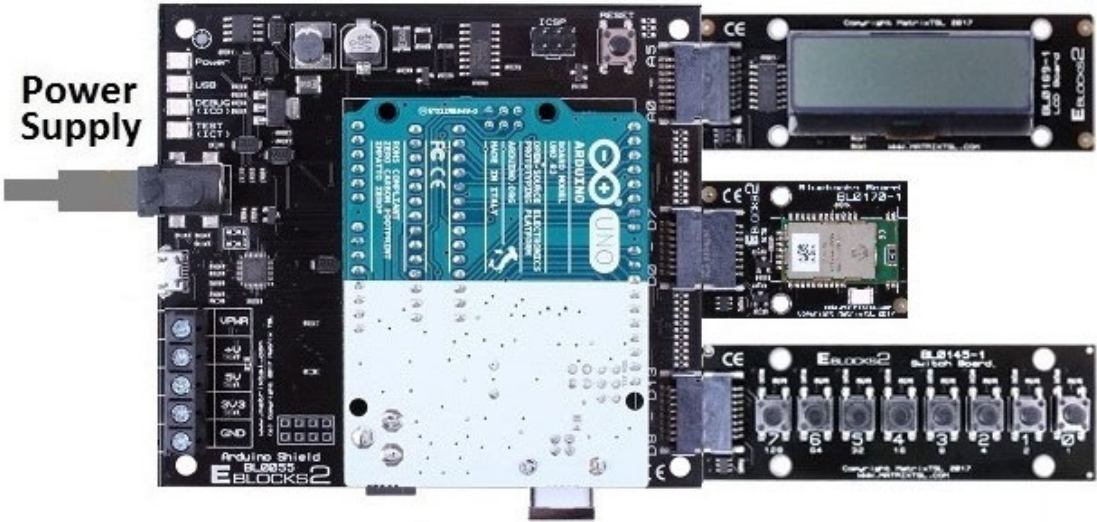
E-blocks2 setup

The two sets of E-blocks2 boards should be connected as shown below.

Some exercises use a switch board on Node B only, the active node, as shown.



PORT	BL0011 PIC Programmer
PORT A	BL0169 LCD Board
PORT B	BL0170 Bluetooth Board
PORT C	Some projects use BL0145 Switch Board or BL0167 LED board



PORT	BL0055 Arduino Programmer
PORT A0-5	BL0169 LCD Board
PORT D0-7	BL0170 Bluetooth Board
PORT D8-13	Some projects use BL0145 Switch Board or BL0167 LED board

Introduction to Flowcode

Flowcode is a flowcharting system for microcontrollers. Generally, microcontrollers are programmed in C (a hard language to learn) or Assembly language (an even harder language to program in). Teaching microcontrollers generally required a large investment of time in order to skill up students in the basics of the required languages before they were able to tackle systems such as Bluetooth.

Flowcode is a much simpler and more intuitive method of creating programs. Based on standard flowchart symbols and using drag and drop icons, a working program can be built in minutes. Icons can be configured using dialogs that remove the chance of syntax errors or invalid options being selected. Flowcharts can be visually followed and tracked through allowing users to see the wider picture as well as the single element. Most introductory courses on programming use, or recommend creating, flowcharts as a precursor to writing your final code, due to their ability to break down the program flow in a clear and understandable manner. With Flowcode that IS writing your program.

A range of components can be used with programs that range from basic LEDs and Switches through to full communications systems such as CAN, TCP/IP and of course Bluetooth. Once again dialogs are used to remove the possibilities of syntax errors or incorrectly put together function calls. Macro icons allow the user access to component functions allowing users to perform tasks as varied as lighting a single LED to sending text to an LCD display, to checking for incoming messages on a Bluetooth system.

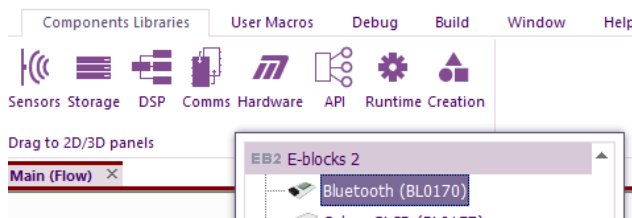
The system is designed to be easy to understand and use, but with the depth and flexibility required by today's technical, educational and industrial market places.

Starting out in Flowcode

This course assumes a degree of familiarity with Flowcode. New users to Flowcode should consult the Flowcode tutorials and introductory course prior to using Flowcode with this course, these can be found on the www.flowcode.co.uk website. Check the Matrix TSL website www.matrixtsl.com and the Matrix Learning Centre for course details and links.

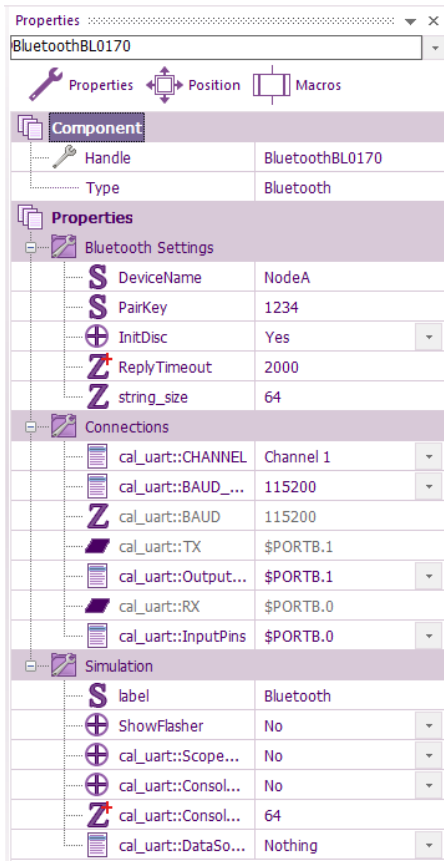
The Bluetooth component

For Flowcode V10, the **Bluetooth BL0170** component can be found under **E-blocks2** in the **Hardware** section of the **Component Libraries** toolbar:



Add this component to the System Panel

Properties and pin connection



The Bluetooth BL0170 component is based on the RN4678 device and has a number of properties and connections that can be set by the user.

These include the device name, pair key and whether the device is discoverable or not. These settings are applied when the device is initialised.

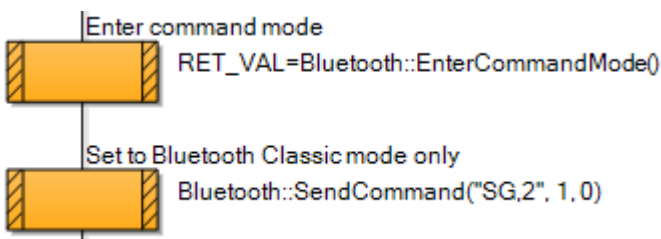
The serial UART connection details to the device are also configured here.

Bluetooth component macros

The Bluetooth component uses a number of macros (or functions) to help program the Bluetooth device. Full details of the macros, their parameters and their uses can be found in the Bluetooth component help file.

The Teaching exercises section of the course will introduce and explain the use of the macros as they are needed during the exercises.

Such as the example below that forces the Bluetooth device into command mode then issues a command.



Testing the hardware

The Bluetooth training kits can be tested, and their Bluetooth device addresses determined, by use of the included Flowcode test program named BT_TEST.fcfx

This can be found in the CP1795 Resources zip.

Connect the boards as shown in section 3.1 and apply power. Connect each kit in turn to the host PC running Flowcode via the USB cable and download the program.

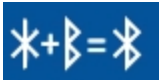
On successful testing the LCD will display the device address of the attached Bluetooth module. This can be noted and used as the connection address in subsequent exercises.

Introduction to Bluetooth

Brief history

“King Harald had this monument made for Gorm, his father, and Thyri his mother. That same Harald who won all of Denmark and Norway, and made the Danes Christian.” So says the Jelling Runestone. This Harald is none other than the legendary Viking king Harald Bluetooth after which Bluetooth technology is named. Bluetooth started off as nothing more than a project name, something to call the technology under development. But the name stuck. And it is a fitting name. Just as the Viking age Bluetooth brought disparate nations together, gave them a new creed and lead them on to greatness so does the modern day Bluetooth bring disparate hardware devices together, give them a new protocol and expand their capabilities. Today’s monument to Bluetooth though would not be runes carved on stone, but be the flurry of electronic signals as the world communicates via Bluetooth.

N.B. The Bluetooth logo contains the runes H and B for Harald Bluetooth:



Bluetooth was introduced in 1998 by the Bluetooth Special Interest Group, a federation of companies involved in communications, industry and business technologies. The variety and number of companies involved has helped to make Bluetooth successful and widely adopted by a number of different systems.

The original Bluetooth specification version 1.1 (1.0 and 1.0B being too restricted technology wise for major development) was immediately successful. The specification was updated to version 1.2 in 2003 with added features such as higher speeds and adaptive frequency hopping. In 2004 the version 2.0 + Enhanced Data rate (EDR) specification was released. V2.0+EDR is faster, uses less power and allows better communications between multiple devices. Bluetooth continues to grow and adapt to the rapidly changing telecommunications need of the modern marketplace.

Bluetooth concepts

- Bluetooth is designed to enable secure wireless data transfer between hardware devices.
- By the use of a protocol data can be passed between the devices in a standard manner. This simplifies coding and testing as propriety code does not need creating.
- Bluetooth is fully specified which allows manufacturers to create Bluetooth compatible hardware without requiring them to design and implement propriety high level protocols. This also allows for network simplification as there are no problems with competing propriety protocols or specifications.
- Communication is automatic once two Bluetooth devices are within range. This means users do not need to run through set up procedures to start communicating.

Bluetooth devices can be made secure and non-discoverable helping to eliminate security threats from automatic communication. You get to decide if you want to be on the network or not.

Bluetooth is essentially a wireless communications network with the ability to automatically discover and communicate with new devices that come within range. This allows two key benefits to the user.

Bluetooth advantages

Cables are not required.

No spaghetti cabling running all over the place. No unsightly messes. Nothing to trip over or damage. And for uses such as automotive technology no wiring looms to install and maintain/repair/replace.

No physical constraints due to positioning of cables or network points, therefore greater freedom of movement for devices.

Communication is distance based so devices can be mobile.

Devices can move from network to network without requiring infrastructure

Automatic discovery of other devices within range.

No need for physical changes, such as new network points, to be made to accommodate new devices. Therefore, ease of use and ease of upgrading.

Bluetooth security features allow you to set the device to respond to or ignore other devices putting you in charge not the network.

New devices can be added simply by coming within range so no need to edit or change settings or anything to add new devices.

Bluetooth disadvantages

Complexity and reliability

Cables may not be portable, but they are simple and reliable. Complexity, as in Bluetooth systems, brings with it increased risk of failure.

The more complex a system is, like Bluetooth, the harder it can be to find and fix faults.

Battery life

Mobile devices are limited by battery life and recharging facilities.

Loss of power during communications can occur.

Cables may be standardized but plug sockets and voltage levels are not. When traveling a power socket adapter may be required.

Costs

Bluetooth devices are often more expensive than conventional fixed solutions such as a cable.

An initial outlay may be required to convert from legacy equipment to Bluetooth solutions.

Higher repair costs.

Speed

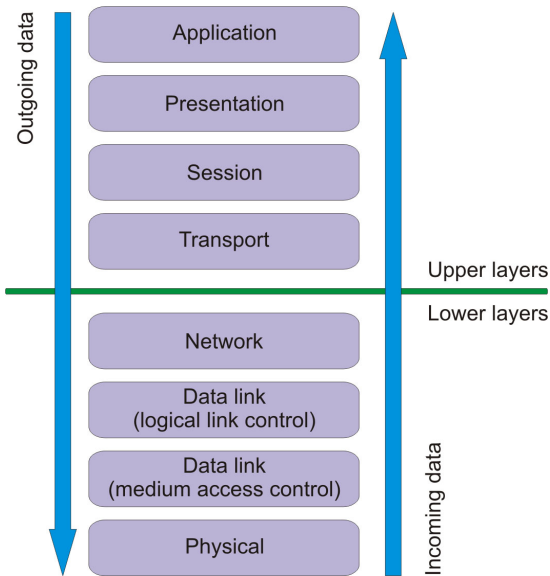
Bluetooth is slower than equivalent fixed solutions such as a direct cable.

Bluetooth communications speed may be restricted by either baud rate considerations, or by limitations of the data transfer speeds of the Bluetooth protocol.

Protocols and the OSI model

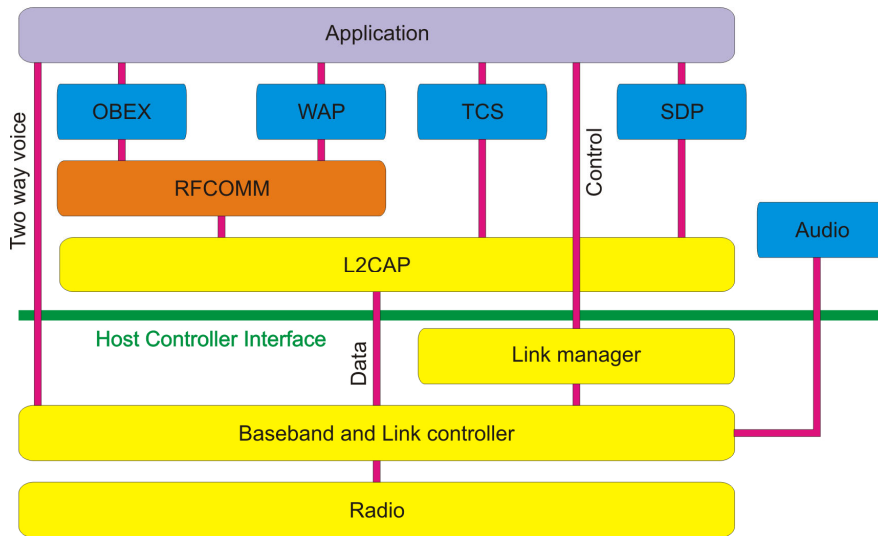
Bluetooth works in layers based on the OSI model. The OSI model is a sequence of layers used to define how data moves from the application to the physical sending of signals. This process is repeated in reverse upon reception of data to change from the signals back to application data. Each layer addresses fundamental areas of the communications processes. By abstracting out the different parts of the communications process into OSI model based layers, developers can implement their systems as a series of tasks each covering one or more OSI layers. Standards can be implemented for specific tasks or layers creating a base for developers to work on.

The basic OSI model diagram can be seen below:



The OSI model is used to aid in separating out the various elements

The Bluetooth layers can be seen below:



The basic process is simple: raw data at the top needs to be converted into bits and bytes that can be transmitted via radio signal and built back up into the raw data the application at the other end can work with.

Transmitting involves breaking the data into chunks that can be sent and wrapping the chunks of data with extra bits of information that can be used by that particular layer to check for things such as errors or missing data. A single piece of data from an application may get broken down into a number of different chunks each of which is surrounded by several different wrappers.

Receiving involves a pass-the-parcel process with each layer unwrapping the top covering and passing the packets up to the layer specified in that wrapper. At each step other functions can occur such as error checking and assembling groups of packets into a larger packet that will be passed on up once complete. Finally the packet will reach the application and the data can be displayed to the user (or put through the speaker if audio etc.).

Not all layers need to be used. Some packets can skip several layers. The process depends on the type of data being sent.

Application

The application is the element that is creating or using the data; be that raw data, audio signals or textual information. This is the level that we users communicate with the system. If we used the Post office as an analogy for Bluetooth the application would be the words that we write or read.

Data input by us in whatever form, spoken or typed etc. is converted into a form suitable for transmission, be that ASCII data, or audio format data. The data can then be sent down the pipeline to be transmitted.

When data is received it is in a format that the application can understand. All the collating of data packets and re-assembly has been done. This is just raw data ready to be used.

Bluetooth layers

OBEX/WAP/TCS/SDP/AUDIO

These are protocols for data. Rather than use custom data models it is good practice to make use of existing protocols and standards. This allows you to communicate with other applications via the same standard protocols. Data is converted to the protocol format ready to be passed on down, or is converted from the protocol to the raw data used by the application. By using such common protocols programming tasks are simplified and code is made easier to transfer to other devices.

Some protocols, such as SDP and TCS, can be used to help configure Bluetooth operations. SDP for instance allows applications to query the Bluetooth device as to what services it can provide.

The protocols commonly used include:

OBEX – **O**bject **E**xchange

WAP – **W**ireless **A**ccess **P**rotocol

TCS – **T**elephony **C**ontrol **S**pecification

SDP – **S**ervice **D**iscovery **P**rotocol

AUDIO – direct audio signal as used for headphones

RFCOMM

RFCOMM, **R**adio **F**requency **C**OMMunication, is a special protocol. RFCOMM enables users to use a standard serial COM port connection rather than a Bluetooth radio connection. These are virtual COM ports, not physical COM ports. RFCOMM tricks the PC into believing its virtual COM ports do exist in the same way as the real hardware ones. But when it receives data it quietly sends it to the Bluetooth radio link. The end application is none the wiser; to it it's just a serial port it sent to. When it receives data it is just data from a serial COM port, the Bluetooth side is totally hidden from it.

The great advantage of this is that the myriad of existing serial COM port technologies can be used directly with no need of special adapters or brand new software. This can have considerable effect on the cost and effort involved in upgrading or developing products that currently use Serial COM port technology. The best Post Office analogy for RFCOMM is email. It's a letter-style communication but is not a letter. Emails are in the same format as a letter, we write and receive them, but the process of sending them is not handled by the Post Office. Similarly the process of sending data in RFCOMM serial communications is not really serial communications even if it appears to be to the host controller or PC.

RFCOMM and the RN4678 module

The RN4678 module sits between the RFCOMM layer and the application layer. The module handles the actual RFCOMM side of things, but requires the commands as outlined in the command set to be sent to it so it not quite at the application level. This allows programmers

to concentrate on the communications strategy whilst leaving the RFCOMM specifics to the RN4678 Bluetooth module.

L2CAP

The **Logical Link Controller** and **Adaptation Protocol** controls and coordinates data flow creating virtual channels, sessions and file transfers. L2CAP is also the main packet assembly areas breaking down the data into the individual packets ready to be sent to the Baseband layer for preparation for transmission. On the return trip the L2CAP is the primary reassembly area collating all the individual packets and reassembling the data from them. In our Bluetooth Post Office this layer would be the pages of the letter, and the envelope it comes in.

Host Controller interface

The Host Controller Interface links the PC or system to the Bluetooth hardware. Elements of the HCI include device drivers required to run the Bluetooth hardware and its interface. Where the Bluetooth hardware meets the data handling hardware is where the HCI resides.

Link manager

Link manager is a low level language for configuring and controlling the links.

Given the sophistication of Bluetooth transmissions with adaptive frequency hopping and a wide range of available frequencies, the Link manager is a very useful tool to have. Link manager aids at the network and data link levels. Link manager is like the Zip code or Post code that the Post Office uses to automate much of its sorting.

Baseband and Link controller

The Baseband and Link controller, corresponding to the Data link level of the OSI model, is a packet controller. It assembles data sent down to it into data packets ready for transmission by the radio layer. The contents of the data and who sent it are irrelevant. All that matters is that the Baseband layer can assemble it into data packets for transmission.

Data passed up to it from the Radio layer has its packet tags examined and is then shunted upwards to the appropriate layer. The Baseband layer does not care what the data contains, just who to deliver it too. In our Post Office analogy this layer is the primary sorting office and the address on the letter.

Radio

The Radio layer is the Physical layer, the lowest level available in Bluetooth. At this level the only real concern is with the actual radio signal. A radio connection is made and the data sent across. The data packets assembled by the Baseband are sent out, and incoming data packets received and passed on up to the Baseband layer.

Hardware details

Bluetooth hardware uses a number of common hardware and software features in its communications that are useful to know.

Piconets

A Bluetooth device can be part of a network of up to 8 Bluetooth devices. The network is formed of one device selected to be the Master device and up to seven other Slave devices. This mini network is called a Piconet. Slaves can only send to the Master device not the other Slaves. The Master communicates to the Slaves in a 'Round robin' system talking to each Slave device in turn. So signals can be passed to another Slave device via the Master device, but not directly.

Forthcoming Bluetooth devices will have the ability to expand this by communicating between Piconets. One device will be the Master in one ring and a Slave in another allowing communication between the two rings.

Paging and inquiry

All Bluetooth devices have the ability to become Master or Slave devices. The Master device in a Piconet is the one that initiated contact, the Slaves those that responded. Contact is

initiated by sending out Inquiry signals to determine what devices are nearby. At the same time devices are listening for contact from another device. If a device is found it can be paged to establish contact and a Piconet either established, or joined if one already exists. If a Master exists as is the case when joining a Piconet then the new device will be a Slave. If no Piconet is already in existence then the device that initiated contact becomes the Master device. Which one it is depends on who received the inquiry signal. As devices can join or leave the network at will mechanisms exist to replace a master device that leaves the Piconet.

Radio specification

Bluetooth operates in the 2.45GHz ISM radio band. The band is split into 79 different channels which Bluetooth hops between at 1600 times a second (Specification 1.1 and 1.2). Later Specification 2 Bluetooth devices can be up to 3 times faster. Bluetooth hops frequencies to prevent both interference from or to other systems on that same frequency (e.g. a baby monitor), and to prevent Bluetooth devices hogging one particular channel and thus jamming it for other devices. Obviously the system needs to know which channel to change to. This is done at the radio level with the transmitter and receiver automatically keeping pace with the frequency hops of the other devices in the network. Signals sent via the radio level contain embedded data about the frequency hop pattern the device is using that the other devices can use to work out which to go to next. Fortunately it is all done automatically far down in the layers and so is not something you need to configure or program for with the Bluetooth E-blocks2 board.

Due to the low power of the radio signal (about 1 milliwatt) the range is limited to about 30 feet, but this is more than enough for most Bluetooth devices.

Audio transfer

Due to its telecommunications roots Audio signals are a major part of many Bluetooth devices.

In order to simplify Audio transfer, Bluetooth can fast track digitized audio signals through the system to the audio output on the receiving device. Audio data can be sent via the normal methods, and applications such as a Bluetooth MP3 player would generally use the normal Application to Application route as it has better data integrity, and buffering data at the receiving end will handle most latency issues. For direct communication such as a telephone though the direct route is preferable as data quality – i.e. noise, is less of an issue than latency – i.e. waiting to hear what was said as the data takes time to work through the system. Signals are fed down to the lower layers directly, processed and compressed at the Baseband level and sent to the receiver which sends the data direct from its Baseband level to the audio output device.

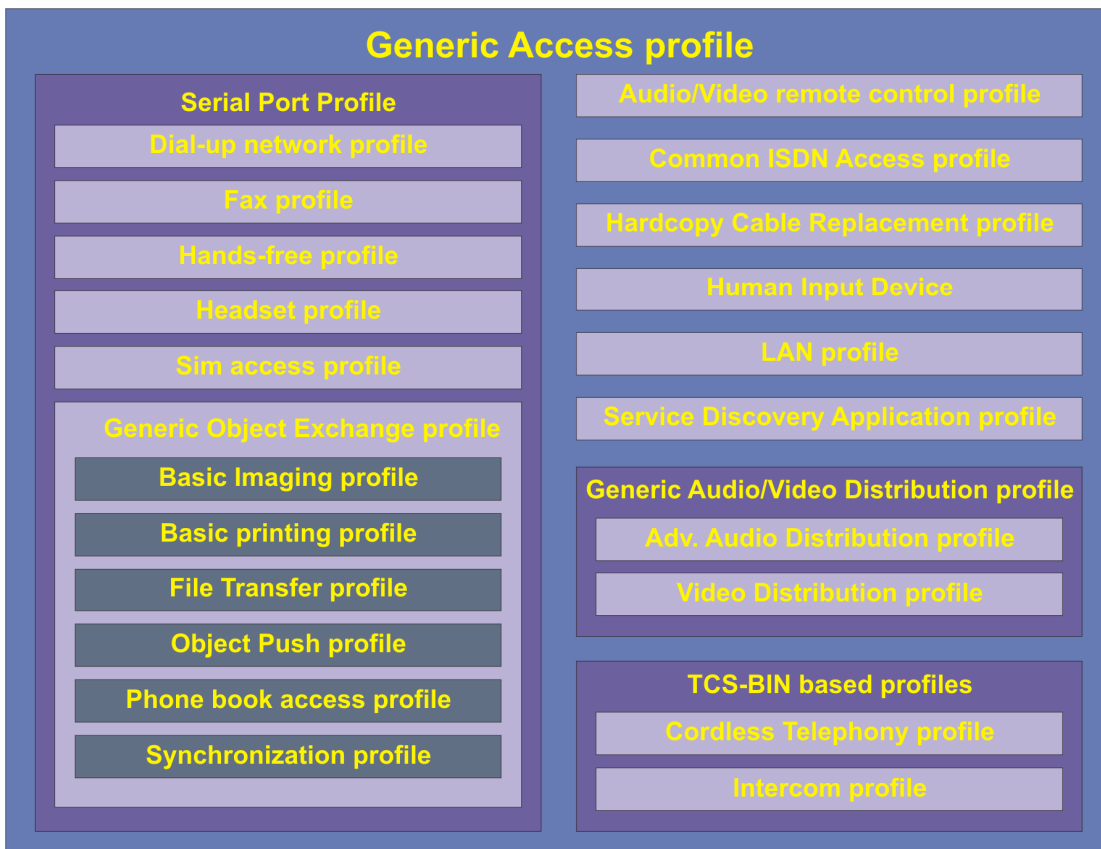
Authentication and encryption

Bluetooth is a radio device. This makes it inherently unsecure as any device within transmission range can listen in to the signal. Therefore security is a prime concern for Bluetooth.

The primary security level is pairing. Devices need to be paired up which requires the address and pass key for the device being paired with. Access to the address can be blocked by devices, thus making pairing impossible, and the Pass key is not accessible by communications so must be known for the device to pair. In practice this will be a PIN number supplied with the product. So to pair a phone and a headset the phone you will need to enter the headsets PIN number into the phone when prompted. No PIN – No communication.

Once paired communications can begin. Bluetooth devices can be set to both encrypt and authenticate communications, thus making sure that data is from an approved device, and is encoded making listening in fruitless. Any code can be cracked, but the Bluetooth encryption and close range needed for listening in make it a very secure system.

Profiles



Profiles are code based implementations for use with standard hardware structures such as headsets or mobile phones. Conforming to a profile allows the hardware to make use of features and behaviors appropriate for that type of device. For instance using the Headset profile allows other units to communicate to the Bluetooth device using standard calls that are part of the Headset implementation. The Headset profile will contain a number of functions that are also in a Telephone implementation, but many of the Telephone functions will not be in the Headset profile. This is because the Headset profile is a subset of the Telephone profile

The layering of profiles allows higher level profiles to use the same features as lower level ones. The Fax profile for instance can use the File Transfer features that are part of the File Transfer profile. It can also use the Synchronization profile features as well as this too is also a sub set of the Generic Object Exchange profile that is a sub set of the Serial Port profile that the FAX profile resides in.

Part 2: Teaching exercises

Discovery

Theory: Finding other Bluetooth devices

To be able a Bluetooth device to communicate using Bluetooth there needs to be another device to communicate to. With Bluetooth this can be any device within communications range. The first step to establishing communications is to find that other device. The RN4678 module has inquiry commands, “F,<number>” command that can be used to inquire about other devices in range. The “F” commands send a signal that is picked up by any Bluetooth device within range. These Bluetooth devices can then elect to respond to this inquiry to let the inquiring device know that they are out there. Devices respond to an inquiry by sending their device address that can then be used later on to contact the receiving Bluetooth system.

The Inquiry commands

Full details of the Inquiry commands can be found in the “RN4678 Bluetooth® 4.0 Dual Mode Module User’s Guide”. This is included with the resource files. Whenever a new command is introduced refer to the command set document to gain an understanding of the new command. Building up knowledge of the RN4678 commands set is an important body of knowledge. The “F,0” or “F,4” commands causes the Bluetooth module to transmit an Inquiry signal to which other Bluetooth devices can respond. The “F,4” command returns device information without the device name. We will use this command to reduce the amount of information we need to process and display. The optional <devclass> parameter is used to filter which devices to check for and will be dealt with later. For now, the base “F” command can be used.

Invoking the “F,4” command will cause us to receive the following responses:

- 123456789012 – The 12 digit Hexadecimal address of the Bluetooth device answering the inquiry. Note that not all devices within range may answer. Bluetooth devices can be set to ignore the Inquiry command. This will be covered next chapter.
- Inquiry complete – Received when the Inquiry is complete (default 10 seconds)

Err – An error has occurred

A number of responses can be received: one address response from each device answering the inquiry, and a confirmation message once all have been received. Checking can be done and if it is not an error response then the address can be displayed.

Additional Inquiry parameters

The Inquiry commands can take optional additional parameters that can be used to filter responses.

F,<0-5>,<value 2>

Where <0 - 5> is the Inquiry mode index

and <value 2> depends on the value of the Inquiry mode

Example: **F,2,001F00** (expects 3 bytes), will scan the default time (Set by **SL** command) using the COD 0x1F00

Note that Inquiry mode indexes 0 to 4 only applies to Bluetooth Classic and only Inquiry mode index 5 allows a BLE scan

Discovery example

There are 4 Bluetooth devices within range, named Blue1-Blue4. The devices are set up as follows:

Device	Address	Configuration
Blue1	00809872F3D4	Accepts Inquiry commands
Blue2	00809864DD44	Accepts Inquiry commands
Blue3	00809894E620	Does not accept Inquiry commands
Blue4	00809894E5D5	Accepts Inquiry commands

Device Blue1 transmits the inquiry command.

The following responses are received:

- 00809864DD44
- 00809894E5D5

Exercise 1: Discovering Bluetooth devices

Introduction

To be able to communicate to a Bluetooth device it is necessary to know that the device is there. The first step in Bluetooth communications is therefore to inquire as to what devices are present, and to be able to identify them so that communications can be established with them.

Objectives

- Develop a program that performs a Bluetooth device Inquiry and displays the addresses of any devices found on the LCD display.
- Display a completion message once all the responses have been received.

Pre-requisites

- An understanding of standard Flowcode Components and icons, such as Decision icons and the LCD component and macros.
- An understanding of sending Bluetooth commands (See the Bluetooth Component Help file in Flowcode for details on the macros involved)
- An understanding of receiving and retrieving message data (See the Bluetooth Component Help file for details on the macros involved, and an outline of the relevant strategies)

Hardware/Software requirements

The following items of hardware are required:

- Bluetooth solution for the Exercise program.
- 1 or more additional Bluetooth devices for demonstrating the program. Note: The second Bluetooth board from the Bluetooth solution can be used for this Exercise. The program BT_EX1_NODE_A.FCFX can be used to set up the board.
- Set up Flowcode and the hardware components as specified in the Getting starting section.

Exercise information

The Inquiry command is "F,4", see the command set in the RN4678 User Guide document for details on the RN4678 commands.

The expected responses are:

- A 12 digit device hexadecimal address from devices that respond.
- An error message.

"Found x" once the Inquiry is complete.

The objectives can be broken down into the following:

Tasks

- Send the Inquiry command.
- Check for responses.
- Display the device address of any device that responds.
- Display the completion message once the inquiry is complete.

Learning outcome

Primary learning outcomes for this exercise are:

- Understanding basic Bluetooth device commands
- Understanding the discovery process
- Creating and sending commands
- Checking for and retrieving responses

Checking for specific responses

Additional tasks

The following are additional tasks that can be implemented to add extra features or improved functionality to the basic program:

- Implement your own count of how many addresses were found.
- Perform basic error checking of the return values from the Flowcode component
- Check for an error response.
- Retrieve and display the device name(s) using command "F,0" rather than "F,4"

Practical implementation: Discovering Bluetooth devices

Discovering and discoverable

Note that the programs created for Exercise 1 needs to be used in conjunction with the program created in Exercise 2 to form a discoverable and discovering pair of boards. Using the two exercises as a linked pair allows students to explore discovering and discoverability using the same program from Exercise 1 in both Exercises, giving them continuity and the ability to see the earlier Exercise in action.

The two programs from Exercise 1 and Exercise 2 will form the core part of most of the subsequent exercises, additional commands being added in as the course progresses. As such a continuity approach can be taken where the final programs from previous Exercises can often be used as the starting point for the next. Using this approach allows students to see how the process flows and grows as they progress through the exercises.

The program students need to write in Exercise 1 is to discover any other Bluetooth devices in range. This necessitates a program in the corresponding Bluetooth device being active and that Bluetooth device being discoverable.

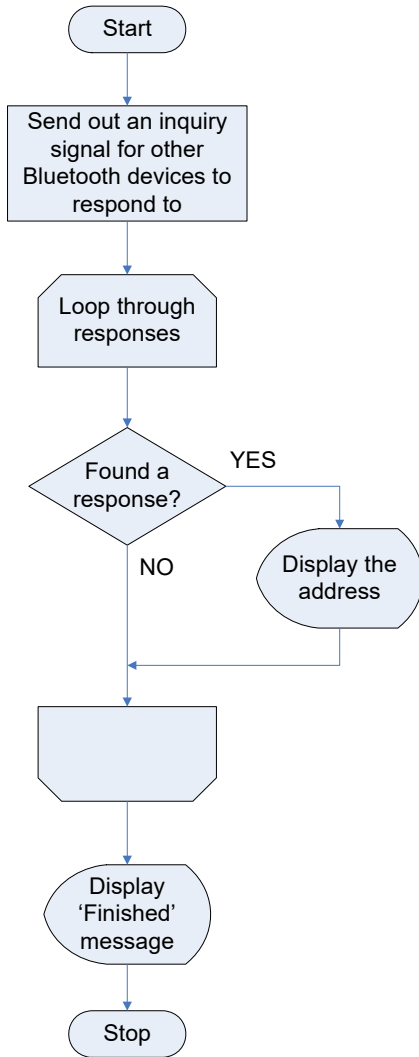
The Flowcode BL0170 RN4678 component has an Initialise function that sets up the Bluetooth device according to the settings of the component properties. Example exercise 1 node A sets the device to discoverable without any need of additional code in the program.

This allows the student to get started with a discoverable device before moving on to learn about the device commands.

Planning the program

Before writing a program there needs to be an idea of what the program objectives are, and an outline of how this will be implemented. Below is a rudimentary plane for an Inquiry program.

The plan is to send out the inquiry signal, then to loop through checking for any responses, displaying the address of any device that responds, and the completion message once the inquiry is complete.



Hardware requirements to achieve the plan include a Bluetooth module and a display device that is capable of displaying the entire 12 digit address of the responding devices. Here we will be using the BL0169 LCD display.

Software requirements include:

- Sending an Inquiry signal
- Checking for responses
- Obtaining the response data

Displaying the data.

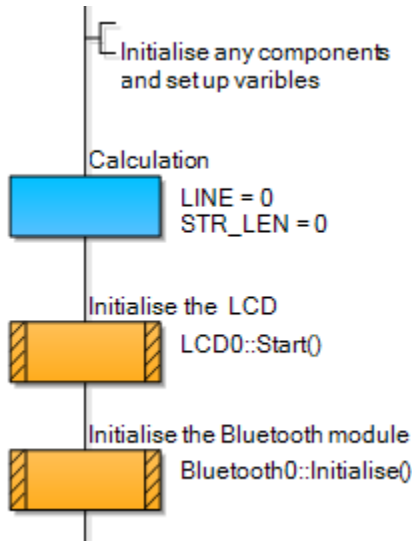
Required macros

Assuming the student has a basic familiarity with Flowcode and basic components such as the LCD display then only Bluetooth specific macros need to be covered.

The following Bluetooth component macros will be needed for the program:

- Initialise
- EnterCommandMode
- SendCommand
- ReceiveByte

Initialising



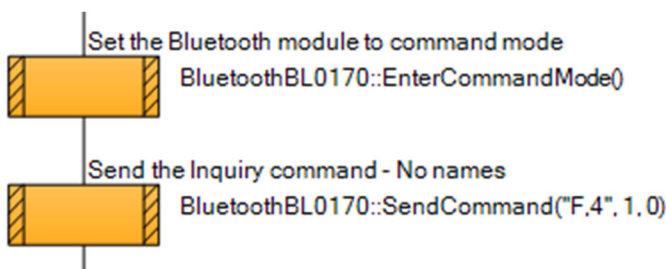
The basic program flow is to start off by Initialising any components and setting up any default variable values. Add in an LCD Start macro, and a Bluetooth Initialise macro.

The Bluetooth Initialise macro is required for all programs that use the Bluetooth component. The best place to add initialisation macros is right at the start of the program.

Add a calculation icon and enter default variable values. This can be updated whilst constructing the program to add in any new variables that are needed. Setting default values is good practice to avoid errors due to erroneous values.

Once all the components and variables are set up the first task can be considered, that of sending the Inquiry command.

Sending a command

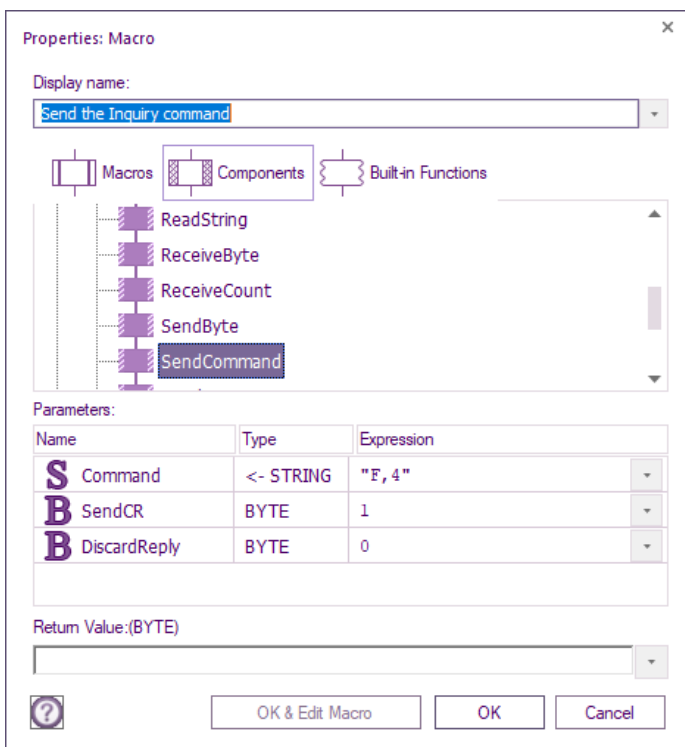


Before a command can be sent to the Bluetooth module, it needs to be put into command mode with the EnterCommandMode function of the component.

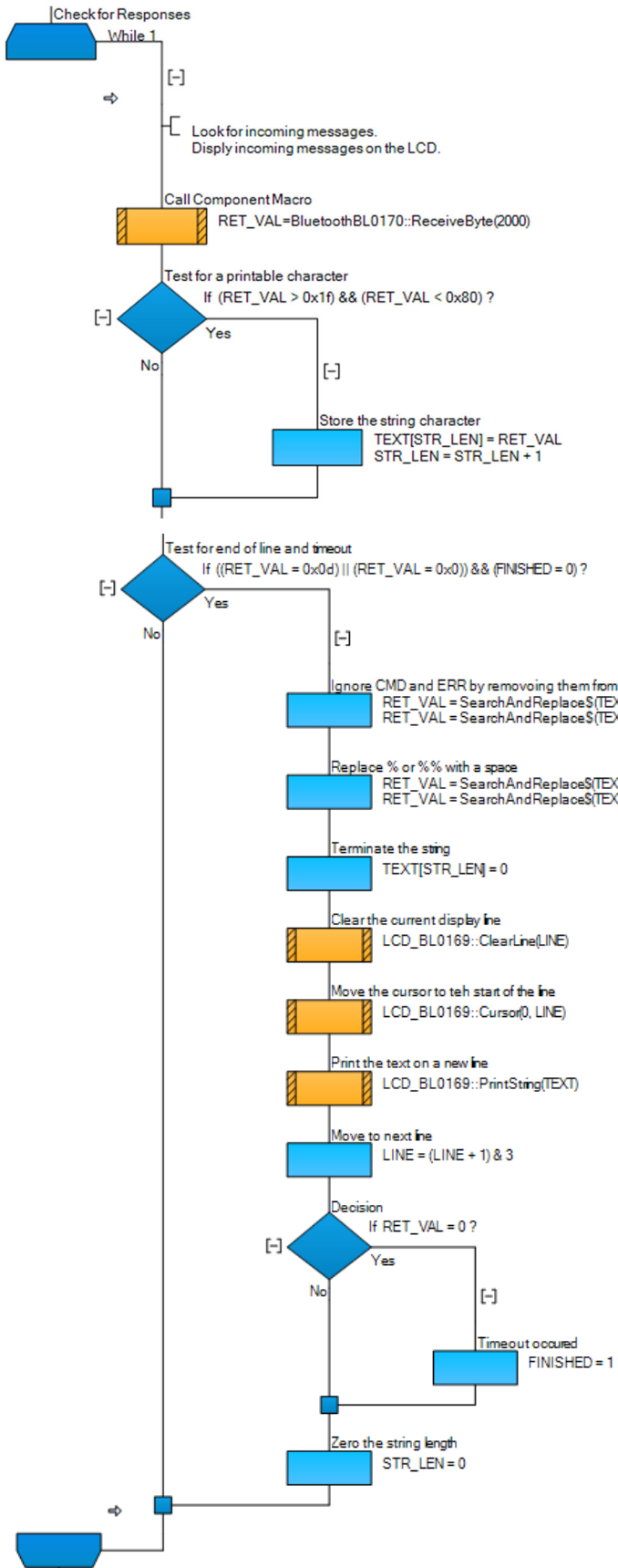
The inquiry command can now be sent using the SendCommand function. Enter "F,4" as the Command and set SendCR to value 1 and DiscardReply to 0.

This function returns a value of 1 if the command is successfully sent. But the return value will not be checked in this example to avoid making this first program over complex. However in general it is good practice to use the return value for basic error checking to ensure commands have been sent.

The LCD PrintString can be used to display progress messages, such as "Starting inquiry".



Checking for responses



Once the inquiry has been sent the next step is to listen for responses.

There can be a number of responses, so a loop structure is needed to go through them all, which is done by entering a continuous “while 1” loop.

A call to ReceiveByte will return characters as they are received from the Bluetooth module. We place this character into the RET_VAL variable.

We then test this variable for being a printable character (see an ASCII table for details). If it is printable, we store it in a temporary string buffer named TEXT.

Text strings received from the Bluetooth module usually end (are terminated) with a carriage return character, which is represented by the number 0x0d, as expressed in hexadecimal notation.

When we receive this character we then display the received string that is contained in our TEXT buffer.

The PrintString function expects the text string to end (terminated) by a NULL or 0 value character, so we first insert one into the end of our TEXT buffer.

If a NULL is returned then the print function will not be accessed, until a new printable character is detected.

You will also notice in this example Flowchart that we first compare the TEXT buffer with expected replies that we do not want to display, in which case we remove the unwanted replies from TEXT buffer.

The variable STR_LEN keeps track of our incoming text length, so reset the value to 0 before going back into the

Running the complete program FINISHED

The complete program can be found in example BT_EX1_NODE_B.FCFX.

This program is downloaded to the exercise target processor board with a BL0170 Bluetooth board attached to Port B in the case of BL0011 PIC processor.

In addition, at least one other Bluetooth device needs to be active and discoverable. This can be the other Bluetooth solution board, or a third party Bluetooth device.

Be warned though that not all Bluetooth devices will respond. When demonstrating the program it may be prudent to test the Bluetooth devices prior to using them for the demonstration to check if they can be discovered or not.

For example, if you have a Bluetooth-enabled mobile phone you will need to turn its Bluetooth on and set the phone's visibility so that it is not hidden.

Compile and download the program to the microcontroller on one of the Bluetooth boards. Briefly unplug the power, plug the power back in and press the Reset switch on the Programmer board. This ensures that the Bluetooth module resets and restarts correctly.

The LCD will display progress messages and the 12 digit address of any devices found, ending with a completion message of the number of devices found.

Discoverability

Theory: Discoverability

The inquiry command covered in the previous chapter is sent to all Bluetooth devices. However, not all devices will respond to the Inquiry signal. This is because Bluetooth devices can be set to hide themselves from other devices. Devices can also set themselves to be discoverable, in which case they will respond to the Inquiry command.

Configuring for start up

Bluetooth devices have a range of configuration options that cover everything from Passkeys and encryption to baud rate and the number of rings before answering. Defaults are generally used so that devices will normally work 'out of the box'. However it is good practice to configure devices with the correct settings upon start up to ensure the device is operating as desired.

In the RN4678 module these options can be configured with the various commands, detailed in the RN4678 Bluetooth® 4.0 Dual Mode Module User's Guide. These commands can be grouped into Set Commands, Get Commands and Action Commands.

Using Set Commands

Set commands change RN4678 configurations and take effect after rebooting via R,1 command, hard reset or power cycle. Some setting commands only apply to Bluetooth Classic operation or BLE. Set commands have parameters and include such things as:

- Bluetooth mode of operation
- Authentication method
- Set to factory defaults
- Device class and ID
- Inquiry scan window
- Device name
- Security pin code
- Data streaming reliability
- Remote addresses
- Service name
- UART baud rate
- Enable or disable command prompt

All configuration changes made by Set commands remain in Non-Volatile Memory (NVM) and survive the power cycle. Any configuration changes must take effect after a reboot.

Using Get Commands

The Get commands retrieve and display the stored information of the device. Most of these commands do not have any additional parameters. These commands return setting values and device status information.

Using Action Commands

Action commands perform actions such as inquiry scans, connecting, and entering or exiting Command mode. Some Action Commands only apply to Bluetooth Classic or BLE.

All Action commands take effect immediately but have no effect after power cycling.

Making a device discoverable

For this exercise we are interested in making the device answer automatically and to be discoverable and connectable, and for us to be able to send data to it.

To do this, the sequence of events and command settings are as follows:

Enter command mode

The Flowcode RN4678 component has a function that implements this named "EnterCommandMode". This sends the required sequence of \$\$\$ characters to the device.

Wait for the command to execute

Flowing each command request we are required to wait for the confirmation and readiness of the device to accept a new command. The device prompts for a command with the text string CMD> . So we use the Flowcode RN4678 function WaitForStringValue("CMD> ") to wait until the device is ready for a new command.

Baud rate and settings

There is a command, SU,<Baud Rate Index>, that sets the serial command baud rate. However, this is factory set to 115,200, so should be left at this default value for the exercises in this course.

Authentication mode

For the initial exercises we set the authentication mode to "Just Works" mode. This mode works without any request to display or input any security pin. The command for this is **SA,2**

Slave mode

When some slave mode commands are run, the Bluetooth automatically enables "Slave" mode .

Activating the updated configuration

Now that we have used Set commands to change parameters, we are required to do a reboot of the device. Any changes to the device configuration using the Set commands do not take effect until the device is rebooted. We do this with the **R,1** Action command.

Wait for reboot and leave command mode

Following the reboot we exit command mode by using the Flowcode RN4678 function LeaveCommandMode.

A summary of the basic configuration set up commands

A simple set of commands to make the device discoverable are:

Command	Description
SA,2	Secure Simple Pairing (SSP) "Just Works" mode. This mode works without any request to display or input any security pin.
R,1	Reboot the device to action the new configuration.

Introduction

- For a device to be found it needs to be discoverable. A Bluetooth device that is discoverable is able to respond to an Inquiry command from another Bluetooth device.

Objectives

- Develop a program that configures the Bluetooth device to be discoverable. This is the complimentary exercise to Exercise 1.

Pre-requisites

- An understanding of the Discovery process as detailed in Exercise 1. In particular the ability to be able to recognize an Inquiry command.
 - An understanding of receiving and retrieving message data (See the Bluetooth Component Help file for details on the macros involved, and an outline of the relevant strategies)
 - Hardware/Software requirements
 - The following items of hardware are required:
 - Bluetooth solution for the Exercise program.
 - 1 or more additional Bluetooth devices for demonstrating the program. Note: The second Bluetooth board from the Bluetooth solution can be used for this Exercise. The program from Exercise 1 can be used to send the Inquiry command,
- Exercise information

The objectives can be broken down into the following:

- Tasks
- Send the configuration commands.
- Check for messages.
- Display any messages.
- Control structures:
- Checking for and displaying responses.
- Learning outcome
- Primary learning outcomes for this exercise are:
- Understanding the discovery process.
- An understanding of how to configure the RN4678 device.
- An understanding of Set and Action commands.
- Checking for and retrieving responses.
- Additional tasks
- Referring in the Bluetooth module manual command settings and make the Bluetooth module non-discoverable.
- What implications does this have for the person using the equipment after you?
- Using the programs in Exercise 1 and 2 make a note of the Bluetooth addresses of each of the Bluetooth modules – you will need these in the next exercise.

Practical implementation: Discoverability

Exercise 2 accompanies Exercise 1 in that having shown how Discovering devices work in Exercise 1 the next step is to show how to make devices discoverable. In addition, configuration methods and the use of Set and Action commands are discussed.

It is important to stress that the commands used here are particular to the specific module used (the Microchip RN4678 device) so are feature of the RN4678 chip and not an inherent part of Bluetooth. Different chips may have different command structures and registers, or even use none at all. However, the general principles are the same.

Before starting to connect the USB lead to the second Bluetooth system – you can leave the program from Exercise 1 in the first Bluetooth system and we will use it to test if the program created here works.

Continuing development

The Students programs created for Exercise 1 can be used in conjunction with the program created in this exercise to form a discoverable and discovering pair of boards.

The two programs from Exercise 1 and Exercise 2 will form the core part of most of the subsequent exercises, additional commands being added in as the course progresses. As such a continuity approach can be taken where the final program from previous exercises can often be used as the starting point for the next.

Configuring the Bluetooth device

For this Exercise the Bluetooth device needs to be configured to be discoverable and connectable. We will not be connecting to it in this particular Exercise; however we may as well set up this basic configuration now.

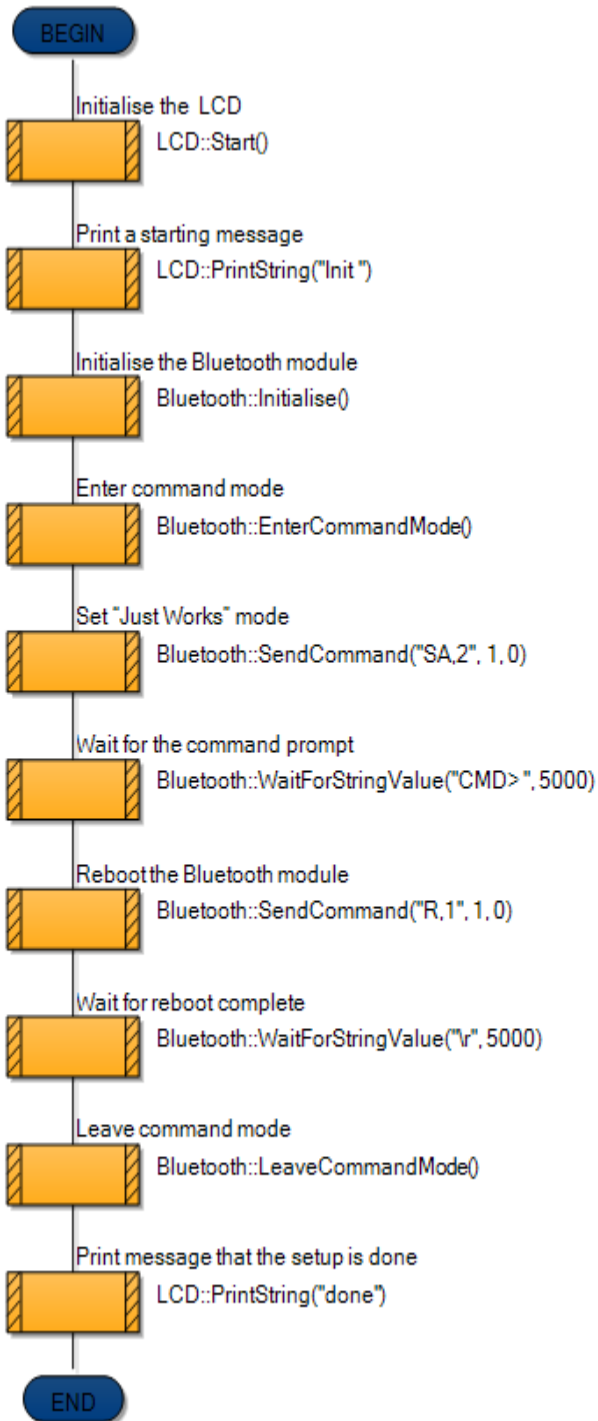
For this exercise use the following configuration commands:

SA,2

R,1

These commands will set the device up to answer immediately, and to be both discoverable and connectable.

Creating the program



The *SendCommand* and *WaitForStringValue* macros, detailed previously, are used to setup the Bluetooth module as in the example shown here.

Running and demonstrating the program

The program built here can be used with the Inquiry program built in Exercise 1 Node B.

The Node B program from Exercise 1 can be used in one Bluetooth board, and this Exercise 2 program can be used with as Node A in a second Bluetooth board.

- Create the program, noting as you do the 12 digit device address of the Bluetooth board that will be used with the Exercise 2 program.
- Download and run Exercise 2 on the Bluetooth board whose address you have noted.
- Download and run the Inquiry program created in Exercise 1 to the other board.
- Monitor the LCD Display on the board to be discovered. When the Inquiry command is sent it will be displayed on the LCD.

Monitor the LCD on the Inquiry program board. When the discoverable board is sent the Inquiry command it will respond by returning its 12 bit address (the number you noted down earlier). Check the LCD display to see that the address is displayed.

Connecting Bluetooth devices

Theory: Connecting – Addresses

As no physical connection exists between two devices all communications are capable of being picked up by any other Bluetooth device. An immediate problem is how to communicate with one particular device and not inadvertently communicate with other devices.

All Bluetooth Classic devices have an address (MAC address), a unique 12 digit hexadecimal number. This address is the same number that is returned in response to an inquiry command. This address can be either retrieved via an inquiry command, or can be stored in the program if a specific device address is to be used and is known in advance.

If a Bluetooth device is connectable and the address is known, then other Bluetooth devices can initiate connection to that device. Generally, devices will be discoverable and connectable so that the address can be determined from an Inquiry command. However, it is possible for devices to be connectable, but not discoverable. For instance, a set of phones that are designed to communicate only with each other may be set to be connectable but not discoverable. The pair of phones have the address of each phone in memory so that a discovery command is not needed, and the phone is ready to start communicating with its partner device.

The basic connection command is:

```
C,<bt_addr>
```

Where <bt_addr> is the 12 digit hexadecimal address of the device to initiate connection with.

The Bluetooth Classic MAC address of the RN4678 modules are set in the factory and cannot be changed.

Exercise 3: Connecting to a device

Introduction

If the address of a Bluetooth device is known, and the device has been configured to be connectable, then that device can be connected to.

Objectives

- Develop a program in one Bluetooth system (node B) that connects to another Bluetooth system (node A), and displays the reply to show that the connection has succeeded.

Develop a program in one Bluetooth system (node A) that allows another Bluetooth system (node B) to connect to it. This program is supplied for you in BT_EX3_Node_A.fcfx.

Pre-requisites

An understanding of the Discovery process as detailed in Exercise 1 and Exercise 2.

Hardware/Software requirements

The following items of hardware are required:

- Bluetooth solution for the Exercise program.

1 or more additional Bluetooth devices for demonstrating the program. Note: The second Bluetooth board from the Bluetooth solution can be used for this Exercise. Download the program BT_EX3_Node_A.fcfx into this second system.

Exercise information

The Initiate Connection command is:

Command	Description
C,<bt_addr>	Where <bt_addr> is the address of the second Bluetooth board, node A.

Set up two programs: one in node B to connect to a second Bluetooth device in node A. When you issue the connection command the receiving node will send the acknowledgement. Display all messages on the LCD display of the nodes.

Learning outcome

Primary learning outcomes for this exercise are:

Connecting to another Bluetooth device.

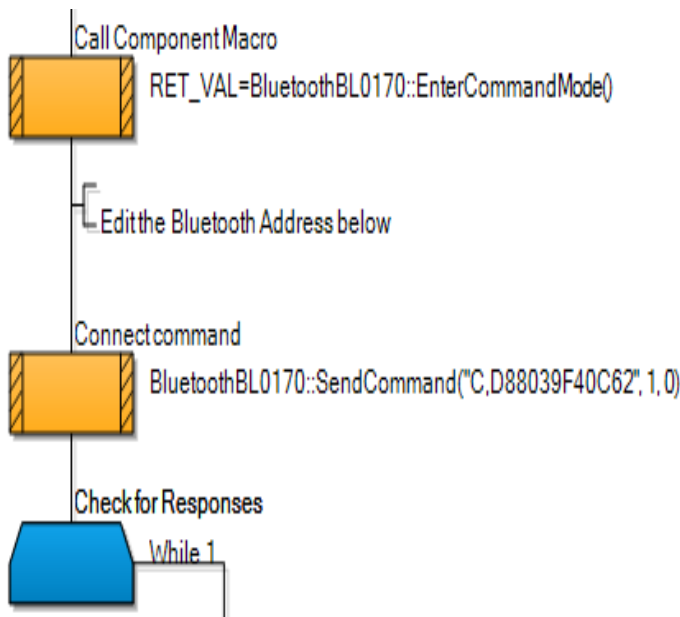
Further work

The LCD display only has 20 characters per line. Devise an extension to the code in BT_EX3_Node_A.FCFX so that any text overflow is displayed on the next line of the display.

Like previous exercises students actually need to develop two programs – one in the Bluetooth device Initialising the communication, and the second in the device receiving the initial communication.

This exercise is a quick and simple one from the workload point of view, but has a number of technical pitfalls to be aware of.

1) Device address. The example program BT_EX3_Node_B.FCFX uses the address of a Bluetooth device used for testing here at Matrix. This will need to be edited to match the address of the board that is to be used with the example program. For this you should use the address you found in Exercise 2 and enter it as a parameter to the *SendCommand* as below.



Be clear about which is the sending node and which is the receiving node and what their addresses are.

- 2) The Bluetooth device to be connected to needs to have been configured to be connectable. Once again the program BT_EX3_Node_A.FCFX has been set up with this issue in mind.
- 3) Once a connection has been made there needs to some way of verifying to ourselves that the connection has been made. The program BT_EX3_Node_A.FCFX is set up to display any messages sent, so we can simply send a message and check that it is received by the other device. The program you write should display the returning message from node A.

No error checking is done at present. The program assumes that the connection will succeed, which may not always be the case. A better method would be to use responses to error check the process. Responses and error checking will be dealt with in a later chapter.

Resetting the systems

The Bluetooth module is not equipped with a reset button. The only mechanism of issuing a reset is to remove power. If you have programmed the Bluetooth module to carry out some activity then pressing the reset button on the processor board will not necessarily reset the Bluetooth module.

For this reason, when developing pairs of programs, it **may** be necessary to remove power from the system and reboot the Bluetooth modules.

Theory: Passkeys and Connecting

Bluetooth communicates via radio which is an inherently unsecure transmission medium. Any device within transmission range can receive the signals sent to any other device. Using unique address solves the problem of specifying which device you intend to communicate to, but other methods are needed to prevent unauthorized access to a device.

The basic security system used with Bluetooth is called Pairing. Pairing is when two devices connect to each other using both a device address and the device's secret "link key" known as the Passkey. The Passkey cannot be retrieved from another device in the same way the address can. To be able to connect to a device you need to know the Passkey number for that device.

Sending the Passkey command

To set up a device so that it can be paired with a Passkey needs to be set up for the device.

The setting command to set up a Passkey is:

```
SP,<passkey>
```

Where *<passkey>* is the Passkey PIN number. The Passkey is a 4 digit number (for legacy pin code pairing, a 6-digit pin is used for SSP authentication in BLE). Generally, Passkey PIN numbers are supplied with documentation that accompanies a particular device. Device PIN numbers should be kept safe, just like you would with bank card PIN numbers.

Initiating pairing and connection

Once the passkey has been defined the data connection can take place by initiating the Connection command as follows:

```
C,<bt_addr>
```

Where *<bt_addr>* is the address of the Bluetooth Classic device to be connected to.

In many instances, such as when pairing a mobile phone with a headset, the Passkey will be in the product documentation and the program will prompt you to enter the Passkey.

Exercise 4: Passkeys and Connecting

Introduction

Devices can be paired with each other for communications. Pairing requires the address of the device to pair with, and a Passkey value to establish contact. Here you will need to develop two programs – one that establishes what the pass key is in a system, and the other that connects to that system and sends data to it.

Objectives

- Develop programs for two Bluetooth systems that allow pairing to take place. Develop a program for Node A that assigns a Passkey of “1234”, and make it discoverable. Develop routines that show any data sent to the node on the LCD. Develop a program for node B that Pairs with node A using Node A’s address and Passkey.
- Once connection is established a simple communication between the systems that shows communication is taking place – i.e. a counter on the sending count data as a single byte which is also displayed on the receiving node.

Pre-requisites

- An understanding of the connection process as detailed in Exercise 3.

An understanding of creating, sending and receiving commands as detailed in Exercises 1 and 2.

Hardware/Software requirements

The following items of hardware are required:

- Both Bluetooth solutions are required for this Exercise.
- Solution 1, node A, will be connected to.

Solution 2, node B, will initiate the connection.

Exercise information

Command	Description
SP,<passkey>	Where <passkey> is the passkey value of the device to be paired with.
C,<bt_addr>	Where <bt_addr> is the address of the second Bluetooth board. If the connection command is successful a CONNECT <bd_addr> response will be returned.

Once a connection has been established loop through and send the numbers 0-9 to be displayed on the other device.

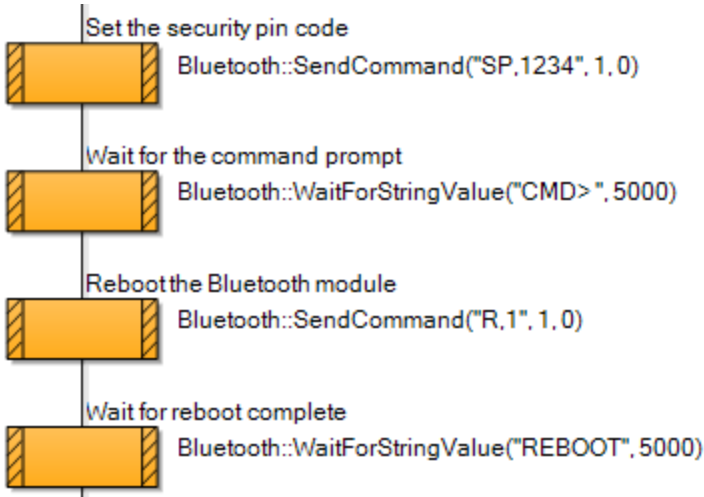
Learning outcome

- Understanding of the connecting process.
- Understanding of the role and security implications of the Passkey.

Further work

- Using a keypad and an array variable, develop two programs that allow the passkey of both the receiving system to be set, and that allows the passkey used by the transmitting system to be set.

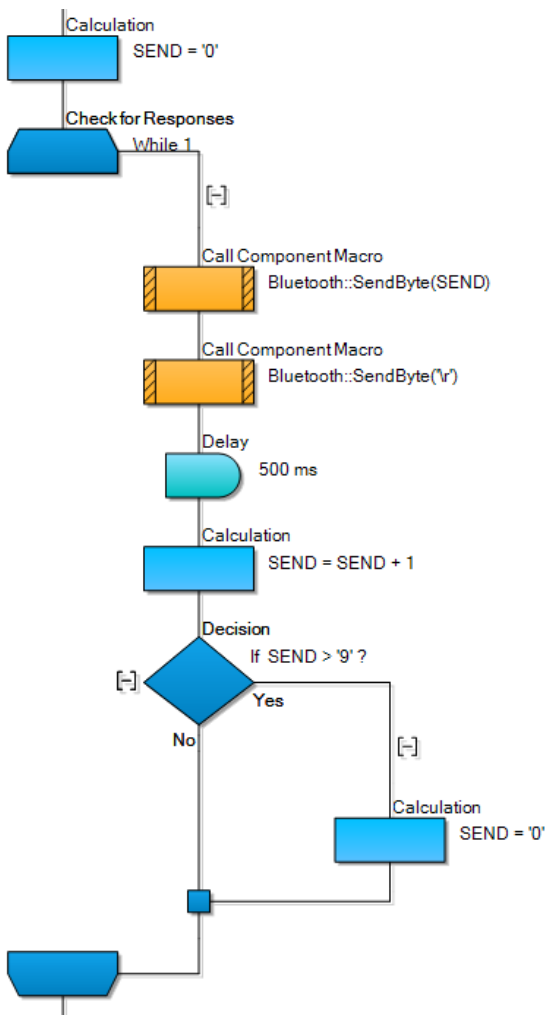
The basic program



Pairing is a key process in Bluetooth operations. The passkey needs to be configured, and as with all “set” commands, is actioned after a device reboot, which can be done with the “R,1” command.

If the details are known in advance then they can be simply entered into the program and the commands built and sent. If the passkey and address are not known in advance then they need to be established.

The Bluetooth device address can be obtained by downloading the BT_TEST Flowcode program. You may wish to note down the addresses for the various boards.



To demonstrate the connection is working a simple 0-9 repeating signal can be sent to be displayed on the receiving device.

Device addresses can be retrieved using the Inquiry command. However knowing if we need to connect device 007284972989 or 00234869030 is another matter. The Inquiry command can be extended to retrieve extra information above and beyond the 12 digit address. The command "F,0" retrieves not only the device address, but the friendly name of the device as well. The friendly name is a device description string that is easier for users to understand than the device address.

For example:

```
012345678912, "NodeA"
```

```
012345678913, "NodeB"
```

Displaying the friendly address can help users select the correct device out of a list. The practicalities of displaying the friendly name are a simple modification to the display response code from Example 1. Adding in the command space the first 14 characters can be displayed on the top line of the LCD, and the rest on the bottom line by moving the cursor when the first 14 characters have been displayed.

In addition we can add the 12 address characters into an array, storing the current address for further use. This can be done at the same time as we are printing out the characters. Add the first 12 characters to the current address.

Obtaining the passkey is more difficult as this it cannot be retrieved via a command. For the examples used in this course the Passkey "1234" is used throughout. In practice a passkey may need to be obtained from the device documentation and either entered in the program code, or entered into the program manually.

Resetting the systems

The Bluetooth board BL0170 is not equipped with a reset button. The only mechanism of issuing a reset is to remove power or issue the reboot command (R,1). If you have programmed the Bluetooth module to carry out some activity then pressing the reset button on the programmer board will not necessarily reset the Bluetooth module.

For this reason when developing pairs of programs it may be necessary to remove power from the system and reboot the Bluetooth modules.

For this exercise we recommend that you remove power from both systems each time you download a program. Then power up the receiving system, press reset and give it a few seconds to set up. Then power up the transmitting system, press reset and give it a few seconds to set up.

Checking responses

Theory: Checking responses

In some of the earlier exercises we send settings or commands and assume they will take place. A more useful method would be to monitor the result from the Flowcode component macro and additionally check the response from the process.

Most of the Bluetooth component macros have a return value that indicates if the request was successful or not. This return value can be tested and used to determine what should be done next, either continue with the next step of the program or skip to the end and abort.

When a command is sent to the Bluetooth device a response is generally sent as a reply to that command. For many commands and occasions the response will be an 'OK' message. However many commands have specific responses. The Discover command for instance sends addresses as responses, together with a process completion message.

Solicited and unsolicited responses

When a response is the direct result of sending a command, it is a solicited response i.e. a response that is expected. For instance, sending an SP,1234 command will result in a solicited response of "OK".

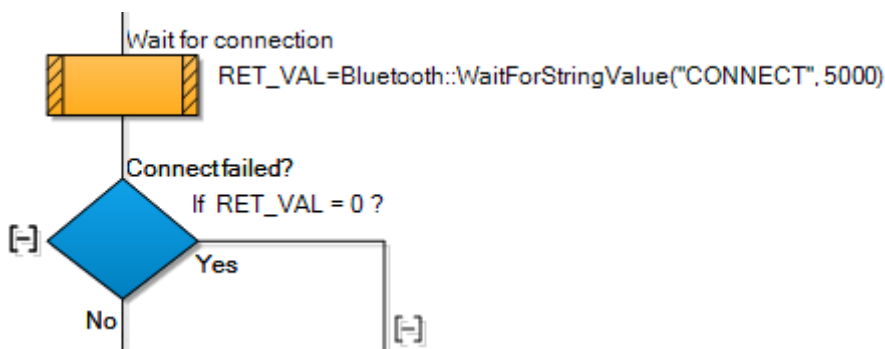
Sometimes a response is sent that is not in response to a specific command. This is an 'unsolicited' response. Unsolicited response can be sent at various times to indicate either a need for further interaction, or to provide additional feedback. For example the CONNECT responses sent during the establishment of communications to inform the other device of the current status of the operation.

Response handling macros

There are some Bluetooth component macros available to help with handling responses:

WaitForStringValue

This pauses program flow until a specific response message is received or the timeout period expires. For example, here we wait for the "CONNECT" message for a maximum of 5 seconds (5,000 mS).



These component macros enable us to read and thereby test or display any text that is received from the Bluetooth module.

For example, we can create a global MESSAGE text buffer and use the *ReadString* macro to read the reply into this buffer. *ReadString* has the buffer size (32 bytes long in this example) as a parameter.

The screenshot shows a development environment with two main panels. On the left is the 'Project Explorer' showing a table of global variables. On the right is a flowchart titled 'Display Responses'.

Expression	Value
Globals	
Constants	
Variables	
B CHAR	n/a
B CNT	n/a
B ERROR_VAL	n/a
B FINISHED	n/a
B INDEX	n/a
S MESSAGE[32]	n/a
B RET_VAL	n/a
B STR_LEN	n/a
Locals [DISP_RESP]	
Parameters	


```

graph TD
    BEGIN([BEGIN]) --> Note[Look for incoming messages.  
Display incoming messages on the LCD.]
    Note --> Goto[Go to next line]
    Goto --> Cursor[LCD_BL0169::Cursor(0, 1)]
    Cursor --> Read[Call Component Macro  
MESSAGE=BluetoothBL0170::ReadString(32)]
    Read --> Print[Call Component Macro  
LCD_BL0169::PrintString(MESSAGE)]
    Print --> END([END])
    
```

Exercise 5: Checking responses

Introduction

When the Bluetooth device receives a command it returns a response. By checking these responses it is possible to both monitor for any errors that occur and to wait for expected responses before continuing a sequence of commands.

Objectives

- Develop programs for two Bluetooth systems that allow connection to take place. Develop a program for Node A that assigns a Passkey of “1234”, and make it discoverable. Develop routines that show any data sent to the node on the LCD. You can use the program from Exercise 4 for this.
- Develop a program for node B that connects with node A using Node A’s address.

Report any errors that occur on node B

Pre-requisites

- An understanding of the connection process as detailed in Exercise 4.

An understanding of the WaitForStringValue macro as detailed on the previous page and in the Bluetooth component help tooltips.

Hardware/Software requirements

The following items of hardware are required:

- Both Bluetooth sets are required for this Exercise.
- Set 1, node A, will await connection.

Set 2, node B, will initiate connection.

Exercise information

Command	Description
SP,<passkey>	Where <passkey> is the passkey value of the device to be paired with.
C,<bt_addr>	Where <bt_addr> is the address of the second Bluetooth board. If the connection command is successful a CONNECT <bd_addr> response will be returned.

Learning outcome

Primary learning outcomes for this exercise are:

- Command and response sequences.
- Response types.
- Error checking and sequence monitoring.
- Unsolicited responses.

Practical implementation: Checking responses

Two important bits of information to note down are the Bluetooth Device addresses, and the Passkeys. For this exercise it is assumed that the general Passkey “1234” will be used. However the Bluetooth devices will need to be modified accordingly.

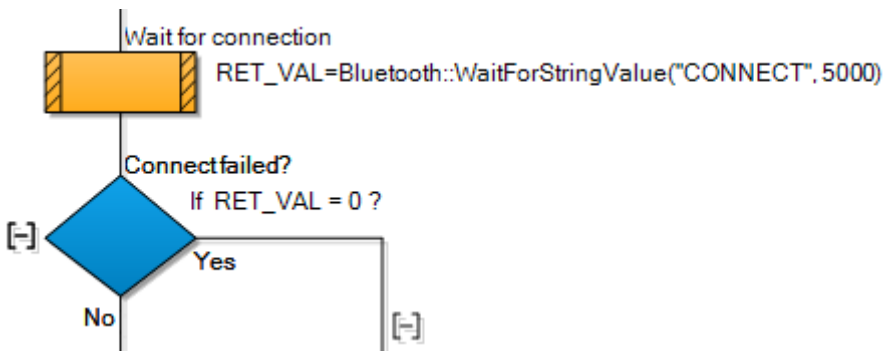
The task requires two programs, one to initiate connection, and one to display data sent to the paired device. The programs developed in Exercise 4 can be used as the base of Exercise 5 allowing the student to see how their code progresses.

The display program only needs to display data sent, so the display program from Exercise 4 can be used as is.

The main focus will be with expanding the program to include response checking.

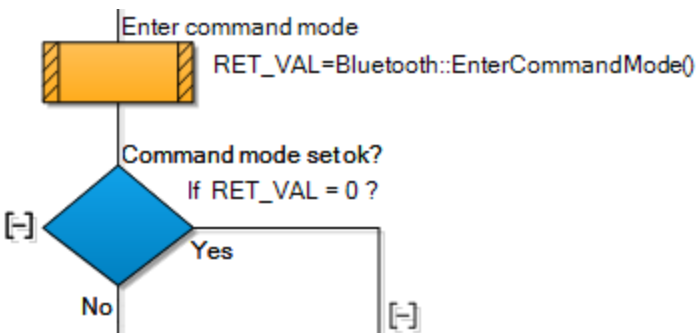
Using the WaitForStringValue macro

This pauses program flow until a specific response message is received or the timeout period expires. For example, here we wait for the “CONNECT” message for a maximum of 5 seconds (5,000 mS).



Error checking methodology

A basic error checking methodology should be adopted where commands are checked for errors. A failure indicates that there is a problem with either trying to send the command or the completion of the command.



For example, here we test the return value to check that command mode was successfully entered.

Once a command has been sent the *WaitForStringValue* can be used to check that the expected reply was received.

Introduction to Bluetooth Low Energy (BLE)

10.1 Bluetooth Low Energy (BLE) modes of operation

The Microchip RN4678 Bluetooth Dual Mode module used with this training course supports both Bluetooth Classic and Bluetooth Low Energy (BLE) communication.

Under Bluetooth Classic, RN4678 implements standard Serial Port Profile (SPP) that supports serial data transfer between two Bluetooth Classic devices.

In addition to SPP for Bluetooth Classic connectivity, the RN4678 has a private Generic Attribute Profile (GATT) service for serial data transfer between two BLE devices.

This BLE data streaming service provided in the RN4678 is named "Transparent UART". Therefore, the RN4678 is a Dual mode Bluetooth module, which supports both Bluetooth Classic and BLE serial data connectivity.

The module operates in two modes: Data mode (default) and Command mode.

When the module is "connected" to another device and in Data mode, it acts as a data pipe.

That is, anything received from UART is passed to the connected peer device through SPP if connected to a Bluetooth Classic device, or via a private GATT service if connected to a BLE device. When data is received from the peer device from SPP for Bluetooth Classic or UART Transparent for BLE, such data outputs directly to UART.

The Bluetooth module is configured and controlled by setting it into Command mode and executing ASCII text commands over the UART.

It has couple of operating modes that the user can set using the command SM:

Default mode (SM,0) - in which other Bluetooth devices can discover and connect to the module. Outbound connections can be initiated in this mode.

Auto reconnect mode (SM,6) - In this mode, the module attempts to connect to the remote device that matches the stored remote address. To set the remote address, use command SR.

All configuration changes made by Set commands remain in Non-Volatile Memory (NVM) and survive the power cycle. Any configuration changes only take effect after a reboot.

All Action commands take effect immediately but have no effect after power cycling.

Exercise 6: BLE GAP transparent UART serial data service

The RN4678 can initiate BLE connection in Generic Access Profile (GAP) Central mode to another BLE device supporting the "Transparent UART" service.

Objectives

- Develop a program that connects to a second Bluetooth device.
- Send serial data to the second device
- Print received messages and data onto the node LCD

Pre-requisites

- You should use the programs you have previously developed as a starting point.
- An understanding of the inquiry and discovery process
- Know the device address of the module to be connected to
- For devices that are BLE only, use the "F,5" command to inquire of their address

Hardware/Software requirements

The following items of hardware are required:

- Both Bluetooth solutions are required for this Exercise.
- Solution 1, node A will be connected to

Solution 2, node B will know the address of node A and initiate connection to it

Exercise information

Command	Description
SG,1	Set the module to Bluetooth Low Energy 4.0 mode
SA,2	Secure Simple Pairing (SSP) "Just Works" mode. This mode works without any request to display or input any secu-
R,1	Reboot the device to action the new configuration.
F,5	Inquire BLE devices in the neighborhood
C,<0,1>,<MAC address>	Attempt a connection with remote device, where the first parameter indicates the address type that can be found in the inquiry result (0 for public address and 1 for private address)

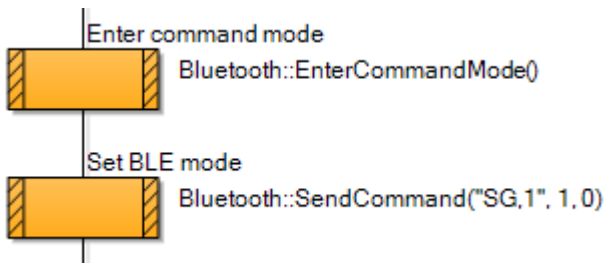
Learning outcome

Primary learning outcomes for this exercise are:

- Configuring the RN4678 module to operate in BLE mode
- Configuring the RN4678 module to operate in SSP "Just works" mode
- Connecting to a second Bluetooth BLE module

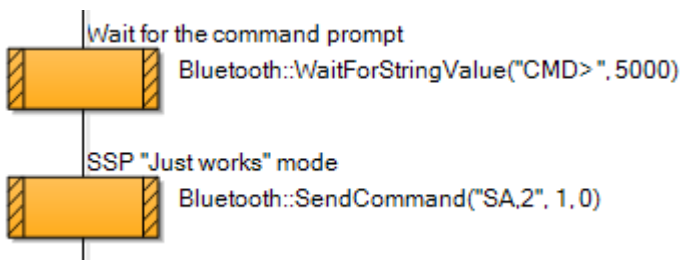
BLE mode

Set the module into BLE mode with the "SG,1" command



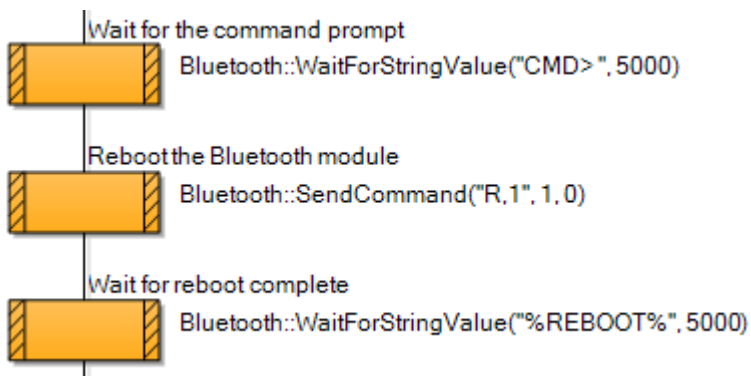
Secure Simple Pairing

Set the module to SSP "Just Works" mode with the "SA,2" command



Activating settings

Reboot the module to implement settings with the "R,1" command



For node A, display of responses to the LCD as previous exercises.

For node B, send the BLE connection command and sending of serial data as previous exercises.

Theory: Trust and Security

Radio communications are inherently vulnerable. Any device capable of receiving radio signal from the same frequency bands can listen in to the signal. Radio communications companies have over the years put considerable effort into ways to send signals securely. And just as much effort has been devoted at times to breaking that security. Enigma and Bletchley Park may be the two most famous names from the age of communications warfare, but their descendants still live on. Sadly the most significant security threat is not from Allied intelligence, but from the more prosaic but insidious hackers and virus writers that plague the internet. Also whilst the French resistance may not be monitoring your signals maybe your business rivals would like to? Details of contacts, deals, offers and prices are priceless ammunition to the cut and thrust of the board room, and the humble phone, with its array of Bluetooth add-ons, is right in the front line.

Security in general

So how does Bluetooth go about ensuring that you are secure?

There are three complementary systems in place.

- Hardware
- PIN numbers

Authentication, Encryption and Trust

At the hardware level Bluetooth employs frequency hopping techniques that leap from frequency to frequency. Listening to one frequency will get you just minute parts of the message. Trying to listen into a full conversation requires you to know the frequencies to hop too, information that is not shared to those devices not included in the communication.

PIN numbers are often used by devices such as Mobile phones and Headsets etc. The PIN number is supplied with a device and is required to be entered to be checked by the program.

Note: This is not the same as the Passkey that is passed programmatically, but a User entered PIN Number. PIN numbers are generally used to inform the device that you are the legitimate owner to help prevent thieves from using the device. PIN numbers should not be stored with the device for security reasons, just like you would not leave your PIN number with your bank cards.

At the program and device level the communications can be set to be Authenticated and the device can be recorded as a trusted device

Security modes

The RN4678 module supports encryption and authentication with security modes from 1 to 4. The definition of security mode are as follows:

- Security mode 1: Passkey Confirm
- Security mode 2: Just Works
- Security mode 3: Passkey Entry
- Security mode 4: Legacy Pin Code

For Bluetooth Classic, all security modes are supported.

For BLE, only security modes 1 to 3 are supported.

For security modes 1 and 3, by default, a random 6-digit security pin is generated and displayed at one end of the connection and the other end must enter the security pin. Optionally, if both connection ends are RN4678, it is possible to fix the 6-digit security pin for BLE by supplying 6-digit pin to command SP instead of 4-digit pin for legacy Pin Code mode.

Authentication

The RN4678 module Set Authentication command sets the authentication method when a remote device attempts to connect. Once a remote device exchanges pin codes with the RN4678 device, a link key is stored for future use. The device automatically and permanently stores up to four peer devices in flash memory using the First-In, First-Out (FIFO) method.

A 4-digit security pin code is used for legacy pin code pairing, while the 6-digit pin is used for SSP authentication in BLE if a fixed pin is desirable. The fixed 6-digit pin code in BLE is not supported by many of the Bluetooth Low Energy implementations.

Trust

When two Bluetooth devices have instigated pairing and established communications between themselves they are said to be in a trusted pair. Once a trusted pair has been established they can communicate with each other without requiring discovery or authentication. As this will obvious speed up the connection and communication processes there are a number of commands and option available to control recording which devices you trust.

Trusted Devices are cached and stored in a list. The Trusted Devices List is stored in Non Volatile Memory allowing it to be retained during power off. The cache on the other hand is temporary listing only the last device to pair and gain trust.

Exercise 7: Trust and Security

Introduction

Good communication is both secure and efficient. Security is required to block unauthorized communications. However security must not be too intrusive, and devices that are known to be trusted need to be able to communicate with ease.

Objectives

Create a program that makes an authenticated connection to another Bluetooth device.

Give the device three options controlled by Switches on Port A:

- List devices currently in the Trusted Devices List
- Add the current device into the Trusted Devices List
- Remove the current device from the Trusted Devices List.
- Use the program to demonstrate use of the Trusted Devices List.

Pre-requisites

- An understanding of the paired connection process as detailed in Exercise 4.
- An understanding of responses as detailed in Exercise 5.

Hardware/Software requirements

The following items of hardware are required:

- Both Bluetooth solutions are required for this Exercise.
- Node A will be used as the device to add or remove from the Trusted Devices list.
- Node B will be used to implement to objectives.

Exercise information

Command	Description
SP,<6 digit code>	Set the security pin code
SA,1	Secure Simple Pairing (SSP) mode
Y	Display device addresses in the paired list with index from 1 to 8
C,<1-8>,<MAC address>	Attempt a connection with remote device
K,1	Disconnect
U,< 1-8>	Remove a device from the paired device list
U,Z	Remove all devices from the paired device list

For details of the commands please see the Command reference of the Module User Guide.

Learning Outcome

Primary learning outcomes for this exercise are:

Principles of Security including:

- Authentication
- Trusted devices

Using the Trusted Devices list including:

- Adding Devices
- Removing Devices
- Listing Devices

Practical implementation: Trust and Security

General objectives

The node A program will act as the passive remote device to be paired and connected with. Therefore, this node is set to BLE Slave Mode with the “SG,1” and “SM,0” commands and the security pin code is set with the “SP,123456” command. The device then requires a reboot with the “R,1” command to configure these settings.

To help identify this Bluetooth node we use the “GB” command to get and then display the device address.

The node B is the active program and is set to BLE master mode with the “SG,1” and “SM,1” commands. Secure Simple Pairing (SSP) is set with the “SA,1” command and the pin code set with the “SP,123456” command.

The main program should then enter a continuous loop which inputs the switches state and respond to any inputs, sending the appropriate command as needed.

For this exercise use Switch 0 as ‘List Trusted Devices’, Switch 1 as ‘Add device and connect’, Switch 2 as ‘Disconnect’ and Switch 3 as ‘Remove Device’.

The commands that need to be sent are:

Switch	Command	Description
Switch 0	Y	Get and display the trusted device list
Switch 1	C,<1-8>,<MAC address>	Pair and connect with a device, add to the list
Switch 2	K,1	Disconnect from the remote device
Switch 3	N,<1-8>	Remove the remote device address from the list

A simple way of determining if the Add and Remove Device from the Trusted Device List worked is to list the devices and look for the device address.

The LCD can be used to show not only the Trusted Device List, but to show reply messages for the other options.

Additional features

As an optional extra the base program can be extended to add additional devices and also to clear the whole Trusted Devices list. These further command details can be found in the

“RN4678 Bluetooth® 4.0 Dual Mode Module User’s Guide”.

Project design principles

Once students have become familiar with Bluetooth and have completed the exercises in this course the next stage is to use that knowledge to design and implement a project.

Design considerations are discussed below, but no code or instructions are given. They can be used as a starting point for creating the project objectives and specifications.

Project – Simple remote control

A simple use of data transfer could be to control a device wirelessly. This can be demonstrated by the use of a switches board attached to a processor and a Bluetooth module setup as a master to transmit the switch data to a second processor with a Bluetooth module setup as a slave that receives the data and repeats this on an attached LED board.

Extending the project

Consider how the amount of data transfers could be reduced to a minimum.

Project - Medical datalogger

One useful task for Bluetooth would be medical datalogging. Patients need monitoring but wires and plugs tie the patient to the bed and create extra work for staff. Should a patient need to go for tests, or even to the bathroom then equipment needs unplugged and moved, or it needs to be mobile. And what is more mobile than Bluetooth? For patients, the ability to move around gives them greater freedom and independence. For staff the system would provide 24 hour datalogging, and ease of patient movement. Also data alerting can be used allowing staff to be alerted to problems as they occur.

Such dataloggers would also aid in medical tests, such as in sports related datalogging where traditional wired sensors would just get in the way.

The first thing to consider for the remote datalogger is what data do you wish to log?

For a simple datalogger temperature could be monitored. Alternatively you could add other sensors to the mix such as heart rate and respiration monitors.

Next you need to consider how frequently the data needs to be updated. For something like temperature it could be every 5 or 10 minutes. For something more critical such as monitoring heart rates you may need readings every few second. After deciding how frequently to send data you need to decide upon a strategy for sending the data. Keep continuous contact? Contact and send the data? Collect data for a while, and then batch send the data?

Next, are there any special cases to consider? Is a sudden rise in the heart rate important? What about temperature? Should we be warned with flashing lights and alarm bells? Should it set our beepers ringing so we know there is a problem. Or is a simple LED alert and a message on an LCD display enough for us? Do we need to know now or in 5 minutes at the next scheduled data broadcast?

Extending the project

Two obvious problems exist with the medical monitors compared to fixed monitors that restrict movement.

- What happens when the patient walks out of network range?

Where is the patient when the alarm goes off?

How can these problems be addressed?

Some of these issues are too big for addressing in a project, but would show how they have considered if written up by the student in an 'Extended/Additional features' section of the project report.

