lwBT – a lightweight Bluetooth Stack for lwIP

Introduction

In recent years, the interest of connecting small devices, even wireless ones, to an existing IP network such as the global Internet has increased. In order to be able to communicate over the Internet, an implementation of the TCP/IP protocol stack is needed. One such implementation is lwIP, an open source implementation of the TCP/IP protocol suite that was originally written by Adam Dunkels of the Swedish Institute of Computer Science. It is suitable for use in embedded systems with tens of kilobytes of free RAM and room for around 40 kilobytes of code ROM.

lwBT was created to transport IP data from lwIP over Bluetooth. It's an implementation of the upper layers of the Bluetooth protocol suite and makes use of lwIPs packet buffering, memory handling, debug messaging, error messages, portability, configurability and shares its coding style principles. lwBT can be used with or without a HCI interface to gain access to the lower layer implementations of the Bluetooth stack that normally run on a separate host controller. The current implementation is written for the LAN Access Point (LAP) and Dial-Up Networking (DUN) profiles with the Personal Area Networking (PAN) profile to be added in the future.

Interfacing lwBT

lwBTs different protocols need to be initialized and controlled in order to setup and tear down a Bluetooth connection. Sample control applications have been created to demonstrate how this can be done for a system with a Host Controller that acts as a Data Terminal and a LAN Access Point according to the LAP and DUN profile.

The lwBT control application interface

The lwBT interface allows the control application program to integrate with the Bluetooth code. Program execution is event based by having callback functions being called from within the Bluetooth code. The Bluetooth code and the control application program both run in the same thread.

Callbacks

Program execution is driven by callbacks. Each callback is an ordinary C function that is called from within the Bluetooth code. Every callback function is passed the current Bluetooth protocol connection state as an argument. Also, in order to be able to keep program specific state, the callback functions are called with a program specified argument that is independent of the connection state.
The functions for setting the application connection state are:

- void hci_arg(struct hci_pcb *pcb, void *arg)
- void l2cap_arg(struct l2cap_pcb *pcb, void *arg)
- void sdp_arg(struct sdp_pcb *pcb, void *arg)
- void rfcomm_arg(struct rfcomm_pcb *pcb, void *arg)
- void ppp_arg(struct ppp_pcb *pcb, void *arg)

They specify the program specific state that should be passed to all other callback functions. The "pcb" argument is the current connection control block, and the "arg" argument is the argument that will be passed to the callbacks.

The lwBT system interface

From the system's standpoint, lwBT consists of 11 C functions, lwbt_memp_init(), phybusif_init(), hci_init(), l2cap_init(), sdp_init(), rfcomm_init(), ppp_init(), l2cap_tmr(), rfcomm_tmr(), ppp_tmr() and phybusif_input(). The init functions are used to initialize the lwBT stack and are called during system startup. The function phybusif_input() is called when the upper layers have received ACL data, and the tmr functions are called once per second. It is the responsibility of the system to call these lwBT functions.

The lwBT device driver interface

If you plan to use lwBT in a host less system your lower layer protocols need to support these two functions called by L2CAP:

- lp_connect_req() – to cause the Link Manager to create a connection to the Bluetooth device with the address specified by the command parameters.
- lp_acl_write() – to send data on a ACL link.

To interface with L2CAP, these three functions can be used:

- lp_connect_cfm() – to confirm the request to establish a lower layer (Baseband) connection.
- lp_disconnect_ind() – to indicate that the lower protocol (Baseband) has been shut down by LMP commands or a timeout event.
- l2cap_input() – this function will process the L2CAP packet and forward it to the upper layers of the lwBT and lwIP stacks if necessary.

If your system includes a Host and a Host Controller your physical bus interface should support this function called by HCI:

- phybusif_output() – to send commands or data to the Host Controller.

To interface with HCI, these two functions can be used:

- hci_event_input() – handle incoming HCI event packets.
- hci_acl_input() – handle incoming ACL data packets.

The lwBT periodic timer interface

The periodic timer is used for driving all lwBT internal timer events such as packet retransmissions. When the periodic timer fires, typically once per second, the lwBT tmr functions should be called.

Protocol implementations

lwBT uses the four protocols in the Bluetooth LAP and DUN profile: L2CAP, SDP, RFCOMM and PPP. If wanted, it also uses the Host Controller Interface (HCI) and IP Network Address Translator (NAT).

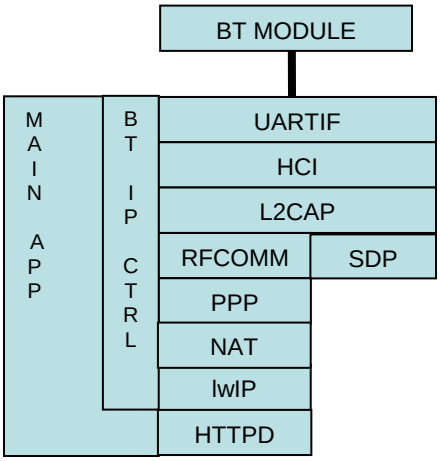

Figure 1 lwBT stack with a HCI interface, a physical bus interface, NAT, lwIP and a HTTP server

Logical Link Control and Adaptation Protocol (L2CAP)

The L2CAP layer code in lwBT has three responsibilities: higher level protocol multiplexing, packet segmentation and reassembly, and the conveying of quality of service information. It is simplified by not implementing connection less data channels.

Service Discovery Protocol (SDP)

Provides a Client/Server model that allow the application to search other remote devices for services and to register services of it own that can be searched.

RFCOMM

RFCOMM is a subset of the ETSI TS 07.10 standard with some Bluetooth-specific adaptations. The RFCOMM layer code in lwBT has the responsibility to emulate the serial cable line settings and status of an RS-232 serial port. It provides multiple concurrent connections by relying on L2CAP to handle multiplexing over single

connections, and to provide connections to multiple devices. It is simplified by only including the multiplexer commands necessary to control such connections.

Point to Point Protocol (PPP)

PPP is used to carry packets from the higher IP layer across the RFCOMM serial port emulation layer. It provides a Link Control Protocol (LCP) for establishing, configuring, and testing the data-link connection. It is simplified by assuming that we will only use the Internet Protocol Control Protocol (IPCP) to encapsulate network layer protocol information over point-to-point links.

Host Controller Interface (HCI)

The HCI layer code in lwBT acts a command interface to the baseband controller and link manager, and gives access to hardware status and control registers. It is simplified by only implementing commands and events necessary to setup and control an ACL link. It can easily be expanded to with more commands and event handling to fit the needs of the application.

Configuring lwBT

The configuration for lwBT is kept in a single .h-file called lwbtopts.h. This file contains configuration options that are project specific such as RFCOMM port number, PPP local address pool range, the maximum number of simultaneous connections and parameters individual to each protocol in the lwBT suite.

Architecture specific functions

The PPP and RFCOMM protocols in lwBT need help to calculate the frame check sequence (FCS). This is implemented in fcs.c and includes four support functions for calculating FCS-8 and FCS-16. They are split from the protocol implementation to make it easier to optimize the them for a specific processor architecture. The lwBT distribution contains sample C implementations of the support functions.

The fcs16_crc_check() and fcs16_crc_calc() calculates or checks the FCS-16 using a 512 byte lookup table.

The fcs8_crc_check() and fcs8_crc_calc() calculates or checks the FCS-8 using a 256-byte lookup table.

Example applications

The lwBT distribution includes two sample control applications, bt_ip_dt.c, and bt_ip_lap.c that initializes a host controller and connects to a network as a Data Terminal (DT) through a DUN or LAP enabled device. The sample LAP application then proceeds

to initialize the access point which will provide LAN access through the established DT connection. If a DT connection attempt fails or an existing DT connection is closed, it will redo a search of devices within range and attempt to connect to them. The application only need to call bt_ip_start() to startup both lwIP and lwBT. After startup is complete the sample control application setup the lwIP application, such as a web server or a sensor. The LAP control application now becomes discoverable to other devices that wish to gain LAN access.

Bibliography

A. Dunkels. lwIP - a lightweight TCP/IP stack.
Web page. 2002-10-14.
URL: http://www.sics.se/~adam/lwip/

Bluetooth Special Interest Group. Bluetooth Core, Specification of the Bluetooth System, Version 1.1,
February 2001

Bluetooth Special Interest Group. Bluetooth Profiles, Specification of the Bluetooth System, Version 1.1,
February 2001

Simpson, W., Editor, "The Point-to-Point Protocol (PPP)", RFC 1661, Daydreamer,
July 1994.

Simpson, W., Editor, "PPP in HDLC-like framing", RFC 1661, Daydreamer,
July 1994.

Simpson, W., Editor, "The PPP Internet Protocol Control Protocol (IPCP)", RFC 1332, Daydreamer,
May 1992.