

TinyBT

Bluetooth for TinyOS

Bluetooth in short

- **Cable replacement**
 - Driven by phone vendors
 - Focus on interoperability
- **Bluetooth**
 - Free 2.4 GHz (ISM) band
 - Time Division Multiplexing (TDM)
 - at the radio level
 - Connection oriented
 - Separate channels via frequency hopping
 - Requires “discovery”
 - What is the impact on performance?
 - Multihop capabilities

Bluetooth and Sensor Networks

- **Promises**

- Off-the-shelf radio
 - Available
 - Promise to be cheap
- High speed
- Separated channels

- **Challenges**

- Stack is complex
 - Strip down to constraints of sensor nodes?
- Radio is a "Bluetooth module"
 - embeds front-end radio, baseband and MAC layer
 - Are standard Bluetooth physical layer and MAC layer adapted to the sensor network regime?

Our Approach

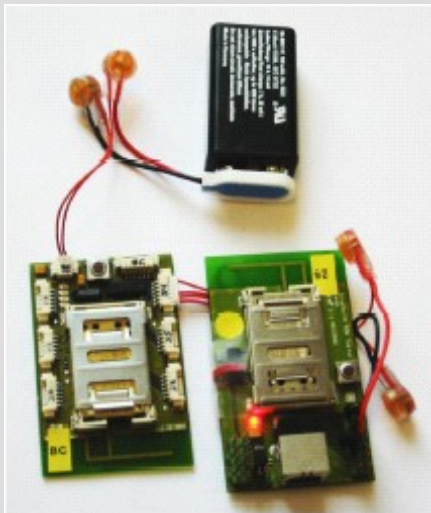
Pragmatic Approach

BTNodes from ETH Zurich

- Atmega 128 7.32 MHz
- 128 KiB flash
- Dual-radio
 - Ericsson ROK 101 007

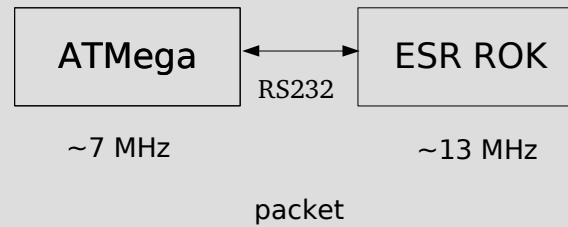
Port of TinyOS to BTNodes

- Development of TinyBT
- Self-Assembly Procedure
- App. using Radio-level TDM
 - TinyDB on top of TinyT
- Performance Evaluation
 - Intrinsic properties
 - Prototype properties

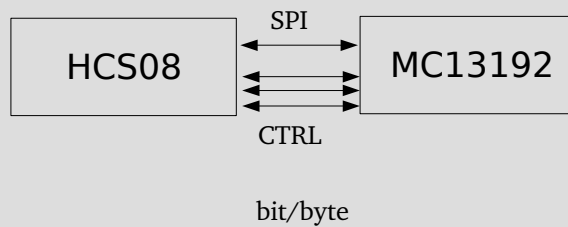


Bluetooth Hardware

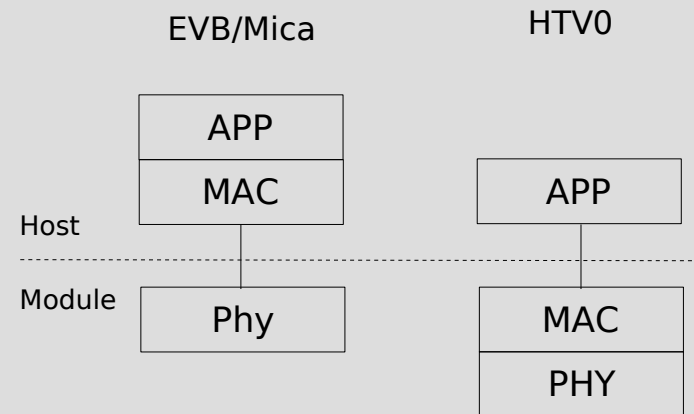
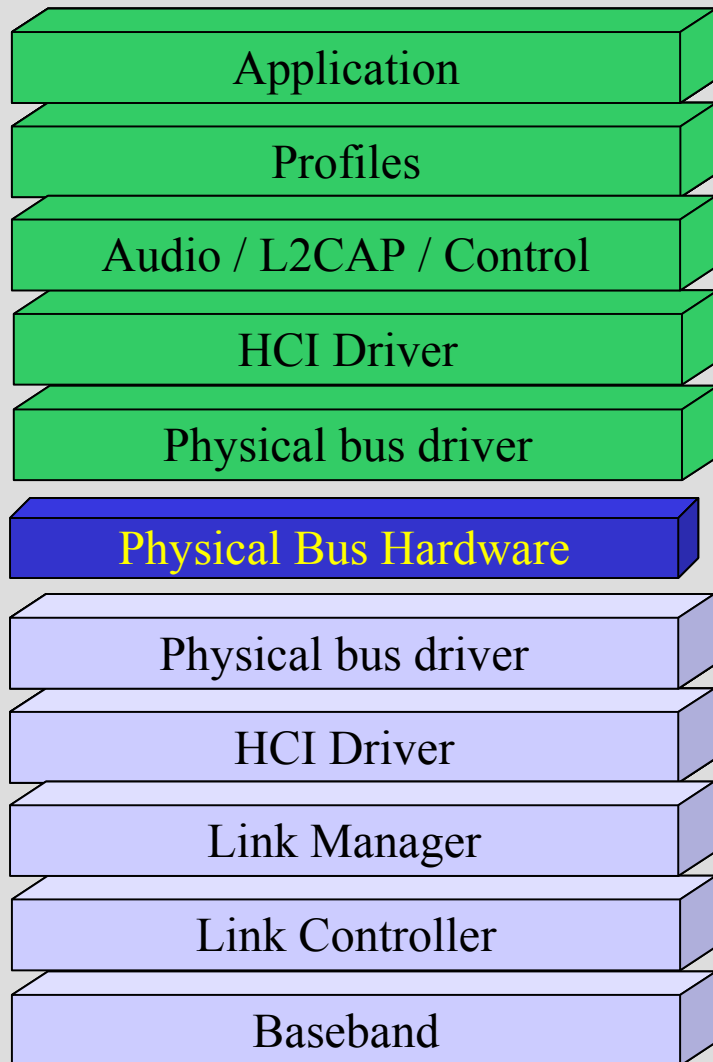
BTNode2_2



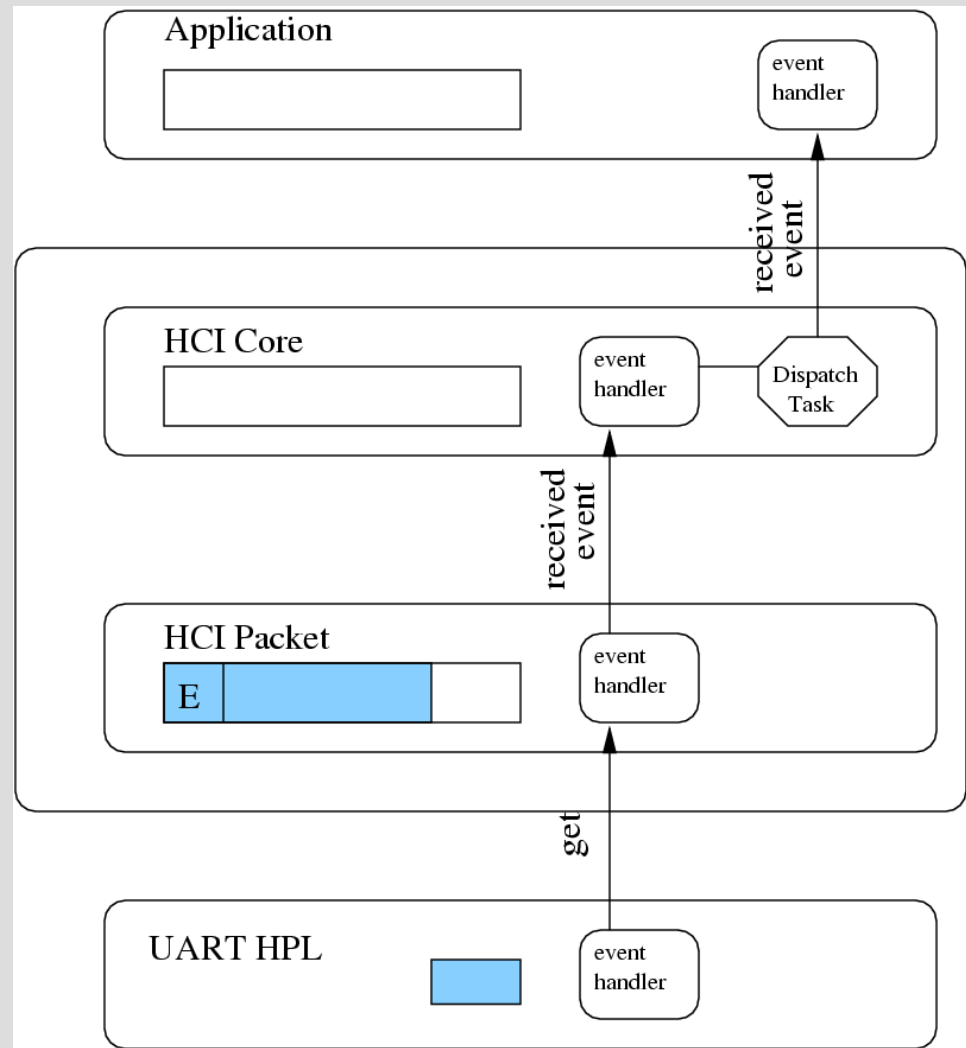
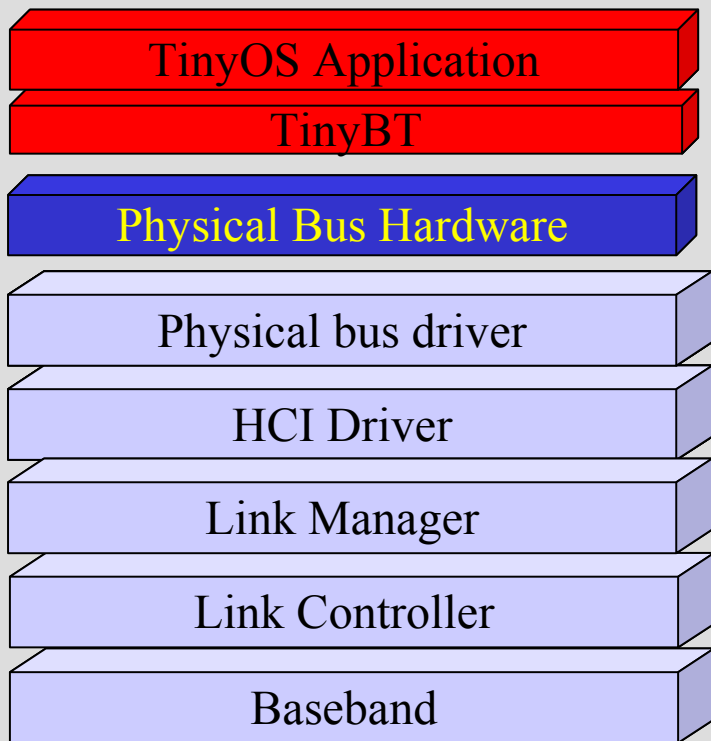
EVB/Mica/HTV0



Bluetooth stack

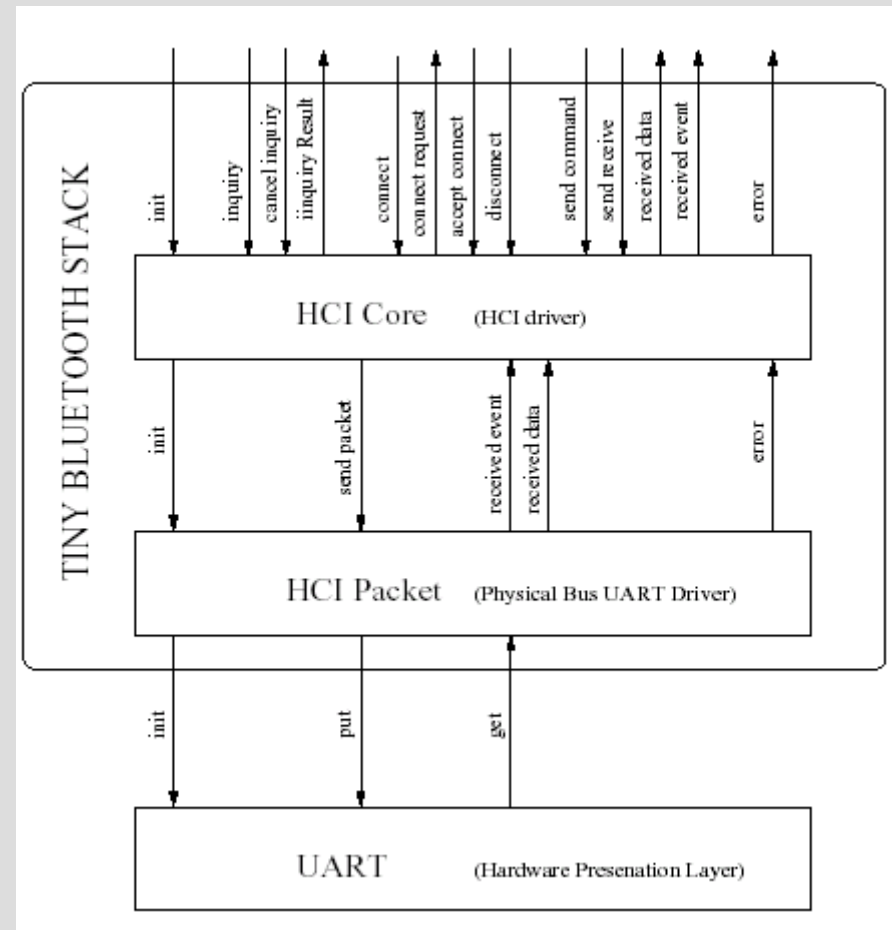


TinyBT Stack



TinyBT Stack

- Asynchronous Programming Model
 - HCI mapped onto TinyOS events and commands
 - UART events decoupled from HCI events
- Buffer Trading
 - Buffers swapped between modules
 - Generic Packet type casted into specific packet depending on event/command
- Interesting information encapsulated inside Bluetooth module



Example

```
result_t send_pkt(inquiry_info* remote_parms) {
    result_t res;
    create_conn_pkt *conn_create = (create_conn_pkt *) get_buf(free_pkts);
    if (conn_create==NULL) panic();

    // Setup start/end pointers
    rst_send_pkt((gen_pkt*) conn_create);
    conn_create->start = &conn_create->cp;

    memcpy(&(conn_create->cp.bdaddr), &remote_parms->bdaddr,
           sizeof.bdaddr_t));

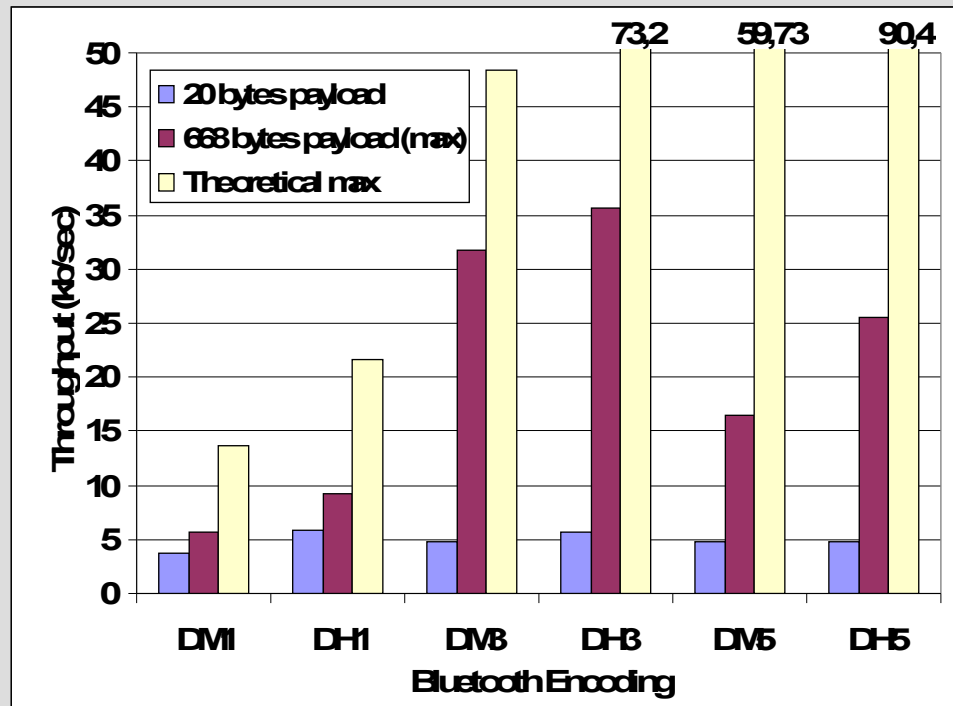
    conn_create->cp.pkt_type      = 0x8; // Packet type: DM1
    conn_create->cp.role_switch  = 0x1; // Role switch capable
    conn_create->cp.pscan_rep_mode = remote_parms->pscan_rep_mode;
    conn_create->cp.pscan_mode   = remote_parms->pscan_mode;
    conn_create->cp.clock_offset  = remote_parms->clock_offset;

    res = call Bluetooth.postCreateConn(conn_create);
    if (res==FAIL) free_buf(free_pkts, (gen_pkt*) conn_create);
    return res;
}
```

Code Size Breakdown

Description	code	bss	data
Support & TinyOS core	1180		
UART 0 & Interrupts	346	4	
UART1 & Interrupts	292	5	
hciPacket0	604	155	
hciPacket1	588	155	
hciCore0	1624	159	
hciCore1	1590	159	
Assembly Component	4796	1021	16
Total	11020	1658	16

Throughput



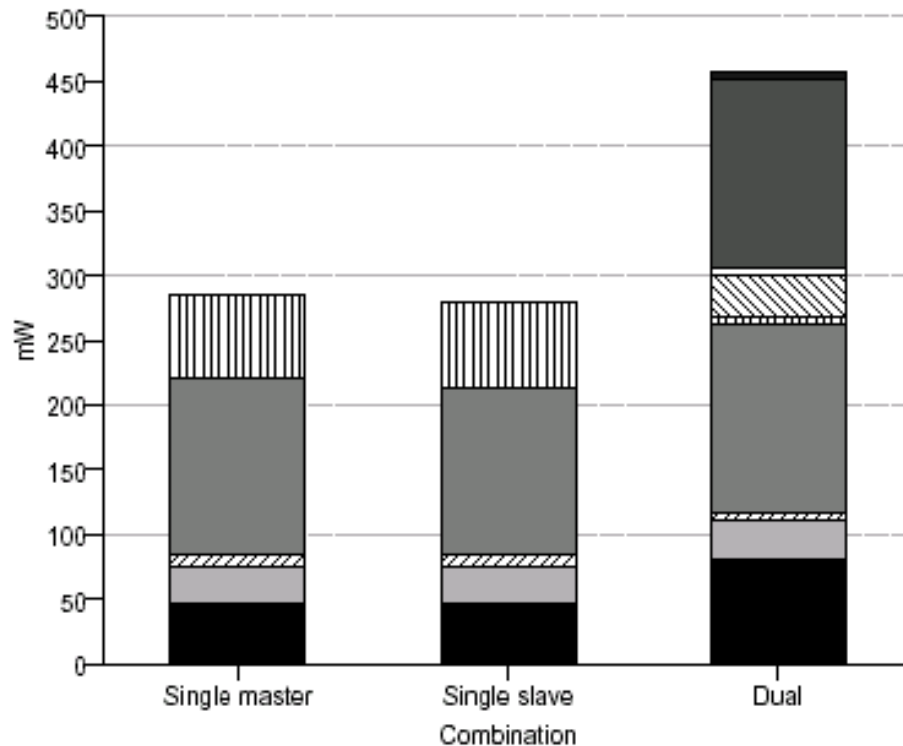
- Point-to-point throughput is high!
- The performance we achieve is far from the theoretical max
 - UART limit is 45 KiB/s
 - Junk sent by Bluetooth module
- Slave-to-master and master-to-slave throughput are similar
- Throughput degrades for Multipoint connections

- DM and DH are two encoding schemes
- DM offers a lower error rate.
- 1, 3 and 5 corresponds to the number of consecutive slots during which slaves and masters communicate.

	1	2	3
Aggregate	38.1	25.4	19.3
Per Slave	38.1	12.7	6.4

Energy Usage Breakdown

Maintaining connections is very expensive



- 50mW when idle and 250 mW when communicating
- Berkeley's mica motes: 10 mW when idle and 160 mW when communicating

Different sleep modes!

Conclusion

- **Intrinsic Bluetooth Properties**
 - It is feasible to develop a Bluetooth stack for TinyOS devices
 - Encapsulation within Bluetooth module hurts
 - Frequency hopping hurts (40 sec period for sniff mode)
 - Inquiry, connection establishment is slow
- **Better engineering might improve**
 - Scatternet support
 - Cost of connection maintenance
 - Throughput
 - max
 - decrease on point-to-multipoint
- **Code available in TinyOS contrib directory**