WIRELESS COMMUNICATION
Conceptual Framework

Serial
over USB
(virtual com port)
Conceptual Framework

Mechanisms and Enclosures
- Actuators
- Sensors

Circuits

Micro-Controller
- Embedded Code

PC or Phone
- Code

Wireless Serial?
XBee Radios

- 2.4GHz, up to 250kbps
- 3.3V
- Optimistic: 100m range. Reasonable inside: 20m
- On-board A2D, GPIO
- Simple serial AT command set on top of 802.15.4 (Zigbee)
- ~ 50mA draw
- Downside : $20-30/each
Simple case: Serial cable replacement

UART pins

Serial over USB

Micro-Controller

Embedded Code

PC or Phone

Code
More Flexible: Peer-to-peer

Requires firmware configuration of PAN ID; Radio ID for addressing
Cheaper alternative for point-to-point

- 2.4GHz
- 1, 2Mbps data rate
- Low Power:
  - 11.3mA TX at 0dBm output power
  - 12.3mA RX at 2Mbps air data rate
- Configurable frequency, logical channel
- SPI interface for configuration and data I/O
- “MultiCeiver” to receive from 6 logical channels
- $3 at single units
## Wiring up a nRF24L01

### Diagram
![Diagram of nRF24L01 wiring](image)

### Table

<table>
<thead>
<tr>
<th>Nordic</th>
<th>Function</th>
<th>KL25Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 GND</td>
<td></td>
<td>GND</td>
</tr>
<tr>
<td>2 VCC</td>
<td></td>
<td>3.3V</td>
</tr>
<tr>
<td>3 CE</td>
<td>Chip Enable</td>
<td>D9 (PTD5)</td>
</tr>
<tr>
<td>4 CSN</td>
<td>SPI Chip Select Not (pull low to initiate SPI cmd)</td>
<td>D10 (PTD0)</td>
</tr>
<tr>
<td>5 SCK</td>
<td>SPI Clock</td>
<td>PTC5</td>
</tr>
<tr>
<td>6 MOSI</td>
<td>Master Out Slave In</td>
<td>D11 (PTD2)</td>
</tr>
<tr>
<td>7 MISO</td>
<td>Master In Slave Out</td>
<td>D12 (PTD3)</td>
</tr>
<tr>
<td>8 IRQ</td>
<td>Interrupt</td>
<td>D8 (PTA13)</td>
</tr>
</tbody>
</table>

Other wireless approaches

• Many other RF transceiver available
• Infrared
• Audio
Smule Sonic Lighter

- **Speaker**: put near mic to ignite
- **Flints**: strike to ignite
- **Valve**: hold on both phones to ignite
- **Flame**: tilt/touch to play with fire, double tap to see map of flames
- **Mic**: blow to extinguish, put near speaker to ignite

Smule Sonic Lighter

IR Sniffing

Example from Tom Igoe, Making Things Talk
Infrared Example: NEC Protocol

Pulse Distance Modulation

- Logical "1" duration: 560μs
- Logical "0" duration: 560μs
- Time interval between logical "1" and "0": 1.12ms
- Time interval between logical "0" and next logical "1": 2.25ms
Infrared Example: NEC Protocol

8 bit address, 8 bit command
Transmitted twice for error checking
Initial gain control burst

Diagram showing the NEC protocol with address and command bits represented.
GETTING ON THE NET
Conceptual Framework

Micro-Controller
Embedded Code

PC or Phone
Code
Conceptual Framework

Micro-Controller

Embedded Code

Internet

Ethernet/Wifi?
Ethernet on 32bit ARM: Feasible.

- FRDM K64F (ARM Cortex M4, $35)
- Seeed Arch Pro (ARM Cortex M3, $40)
- Mbed LPC1768 (ARM Cortex M3, $35)
Ethernet on Microcontroller Gotchas

• Received packets or even outgoing messages may not fit into RAM
  (Forget about nice XML parsing – JSON maybe OK)

• Single thread, slow as server

• Relatively easy to connect when you’re in control of the network, harder when someone else is running infrastructure (MAC filtering)
A Frequent Workaround

Micro-Controller

PC Server

Internet

Remote Client

Serial
or local wireless

Ethernet/Wifi
Serving Data

• Most likely, people will consume data through HTML in a Browser

• Standard Web 1.0 model: Client requests URL, Server constructs page using server-side logic, sends HTML+JS back to client
Client

Server

GET index.html

Client
Client
GET index.html
Server
Node, PHP, ...

index.html
GET index.html

Microcontroller

Sensor value is: 42
Some Frequent Workarounds

e.g., Amazon EC2, Hosted DB (Xively, iDigi)

Micro-Controller

PC Gateway

Remote Server

Remote Client

Internet

Serial, Xbee, ...

Ethernet/Wifi
Xively, iDigi

- Mostly focused on timeseries sensor data
- Simple HTTP Post APIs to save data to their DB
- Web APIs to query, or embed ready-made widgets
GET index.html

Client

GET index.html

Server

Ruby, PHP, …

Microcontroller

index.html

<p>Sensor value is: 42</p>

Updates after page is sent to client are lost
WEB SOCKETS / SOCKET.IO
GET index.html
GET index.html

index.html:

```html
<script src="socket.io.js" />
</script>
```
GET index.html

Client

GET socket.io.js

Server

index.html

index.html:

<script src="socket.io.js"></script>
GET index.html

index.html

GET socket.io.js

Socket.io.js

<script ...

index.html:

<script src="socket.io.js">
</script>
Socket.io.js

Client

GET index.html

index.html

GET socket.io.js

bi-directional pipe

Server

index.html:

<script src="socket.io.js"></script>
Client

GET index.html

index.html

GET socket.io.js

bi-directional pipe

Server

Microcontroller

Socket.io.js

<table>
<thead>
<tr>
<th>index.html:</th>
</tr>
</thead>
<tbody>
<tr>
<td>New sensor value: 43</td>
</tr>
</tbody>
</table>
WIRELESS COMMUNICATION
WiFi

CC3000 ~$30

Mbed LPC1347 with CC3000 on board ~$55
CC3000 as Client

Micro-Controller → CC3000 → Remote Server

Internet

SPI

802.11b/g
WiFi gotchas

• Power hungry (350mA TX)

• Authentication:
  – WEP, WPA, WPA2 personal/enterprise … managing authentication can be messy
  – Worse: Browser-based authentication (e.g., AirBears!) impossible or very hard.

• General ethernet: Firewalls, NAT etc. : May be hard to serve to the outside world
WIFI CHAT DEMO

• See the demo
• Your turn: follow instructions on wiki