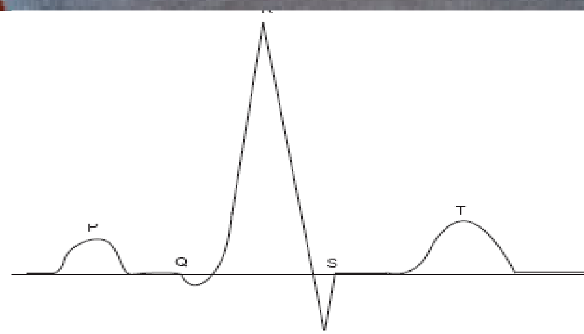
Real-World Sensors Feeding into Databases and Transaction Processors



HPTS Conference **October 2009**

Wayne Duquaine / Grandview Systems
E-mail: grandvu@sonic.net

Overview

- Outline
  - » Trends feeding into this explosion of real-time data
  - » Definitions
  - » Couple Case Studies

- Only 10% of the World's data has been digitized.
  Most of the "real world" has not been digitized.

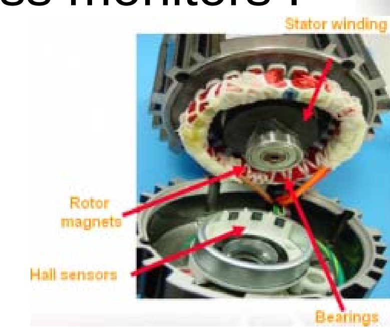
- For the cost of a tank of gas, you can now monitor almost anything.

Trends: Everything is becoming Intelligent

- Previously, Sensors and Real-Time Data were primarily used in Process Control and Industrial/Factory Control:
  - » Expensive Programmable Logic Controllers (PLCs)
  - » Usually existed as islands of automation (poorly integrated)

- Now, <u>everything is becoming intelligent and has sensors</u>:
  - » Intelligent Motors - Variable Frequency Drives (VFDs)
  - » Intelligent Power Supplies - Switch Mode Power Supplies (SMPS)
  - » Intelligent Lighting - DALI, DMX512, Bacnet
  - » Automated Electrical/Gas/Oil Metering - AMI
  - » PDAs and Intelligent Phones (iPhone, …)
  - » Digital Thermometers
  - » Even smart electric toothbrushes with wireless monitors !



- All these devices are starting to create <u>streams of useful data</u> (sans toothbrushes)

Trends: Rise of Cheap Pervasive Computing

- Total People on the Planet 6.77 billion
- Total MCUs installed on the Planet > 25.0 billion
  - » Total MPU + MCU Market (2002) 6.2 Billion units
  - » Microprocessors MPUs (Intel x86) 0.2 Billion
  - » Microcontrollers MCUs (8/16/32 bit) 6.0 Billion
  - » Annual MPU and MCU unit shipments have almost doubled since 2002
- Total Sensors installed on the Planet > 5.0 billion
- Total WiFi devices on the Planet > 1.0 billion
- When you have <u>4 times more Computers than People</u> on the planet, by definition <u>you have pervasive computing</u>

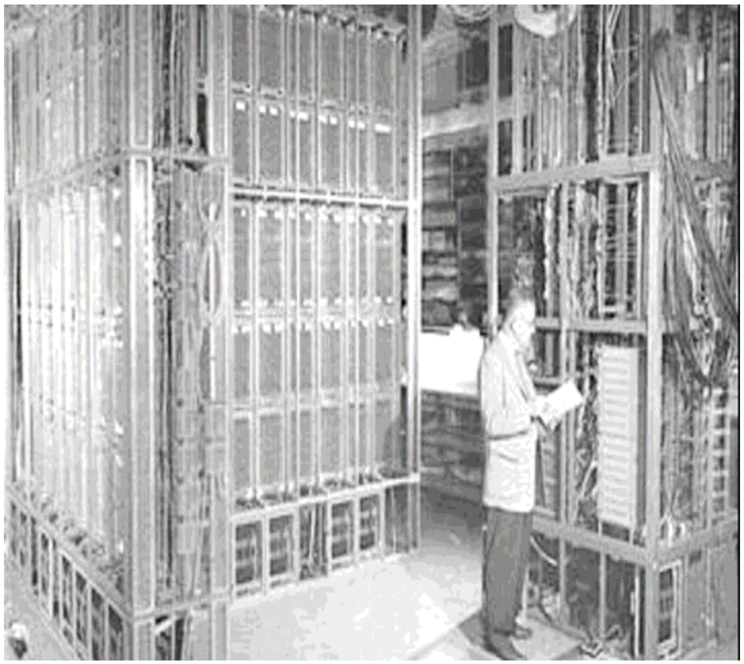
Trends: Real-Time Data is Becoming Dirt Cheap

- The cost to monitor and control real-time data I/O is dropping precipitously.

	Approx Cost	**(Example chips)**
» 8-bit MCU with built in ASK/FSK wireless	$ 0.92	(rfPIC12F3639)
» 8-bit MCU with built-in Ethernet	3.36	(PIC18F66J60)
» 16-bit MCU with built-in Zigbee	3.90	(TI cc430)
» 32-bit MCU with built in Ethernet (as much horsepower as a 3081 mainframe)	7.97	(LMI LM3S6911)
» Standalone (Wired) Ethernet Chip	2.60	(ENC424J600)
» Standalone WiFi Ethernet Chip	5.00	(ZG2100)
» Accelerometer Sensor	5.00-150.00	(sensitivity)
» Temperature Sensor	1.00-200.00	(“)

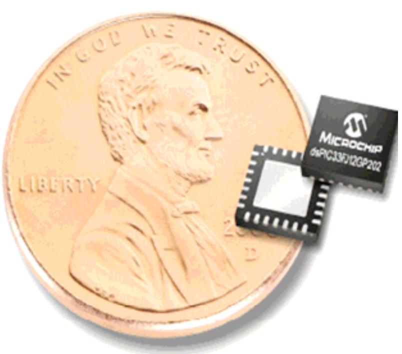
- Most MCUs now have TCP software stacks.
  - » Stacks typically 24-54KB memory footprint and will fit in a $ 3.00 128KB MCU

You Can Now Put A Roomful of Computers and Sensors into Your Pocket



TI eZ430-Chronos Wireless

- More Memory, More CPU Horsepower (25 MIPs)
- Built in Pressure Sensor, Accelerometer, Temperature
- Built in WiFi (915 MHz) to remotely send/rcv data
- Fully programmable (C, assembler)
- And it also keeps time
- <u>Your wristwatch really can submit transactions to CICS</u> !



dsPIC33FJ12GP 16-bit microcontroller has as much horsepower as a VAX (40MIPs), can handle 16+ sensors, and is 1/8 the size of a penny

Challenges of Real-Time Sensor Data

- Typically is a 5-10 fold increase in amount of data to process

- A lot of this data needs to be massaged and aggregated.
  To off-load this work, Micro-Brokers often inserted at the edge of the network
  - » Micro-Broker sits near the sensor nodes and pre-processes their results.
  - » It then collects and forwards the results to the main "back-ends".
  - » Helps reduce traffic on backbone network.

- Aggregates and trends are often more important than single, individual readings:
  - » Trend of power usage (Up or Down)
  - » Trend of temperature (HVAC)
  - » Trend of structural strain (see case study)

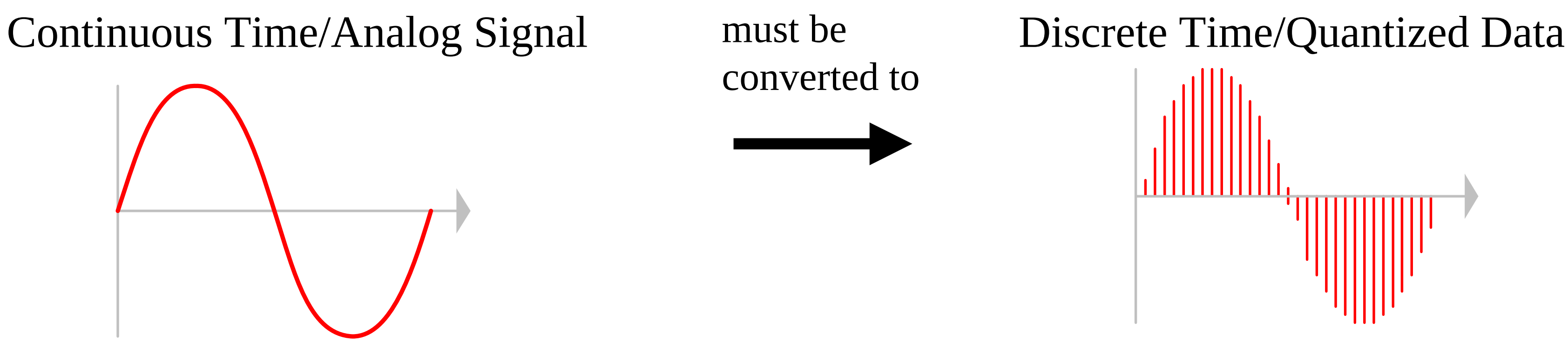
Definitions: Sensor/ Actuator

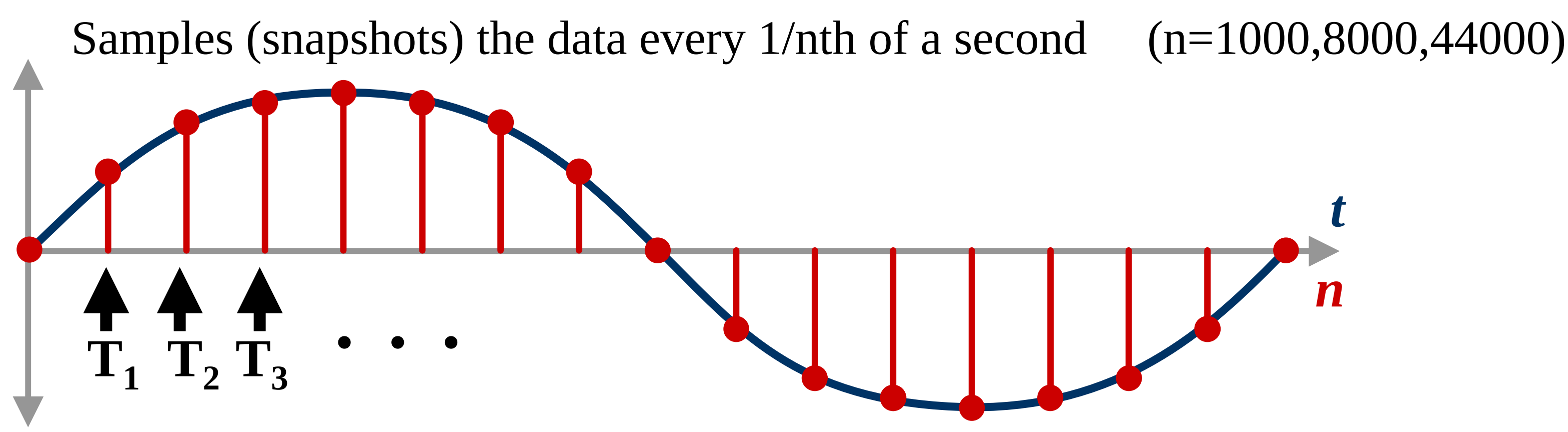
- Sensor = input device that reads a physical value, and outputs a voltage or current to represent the quantity.
  - » Temperature
  - » Accelerometer – measure tilt, acceleration, vibration, …
  - » Proximity Sensor
  - » PIR Sensor (Movement - Building Lighting, Automatic Doors)
  - » Humidity …
  - » Flow
  - » Position / Location (Motor Control, Motion Control, …)
  - » Gas
  - » Light

- Actuator = output device that performs a physical action
  - » Current – increase/decrease current (Motor Control, Power Control)
  - » Pneumatic – open/close air controlled valves (Process Control)
  - » Voltage – turning on a light
  - » Sound Alarms

Sensor Processing – 30,000 Foot View

The Real World is Analog





The computer then goes wandering through those quantized data points to find what it needs. Normally this is done via mathematical filtering algorithms (FIR or IIR).

* Graphics source: Microchip

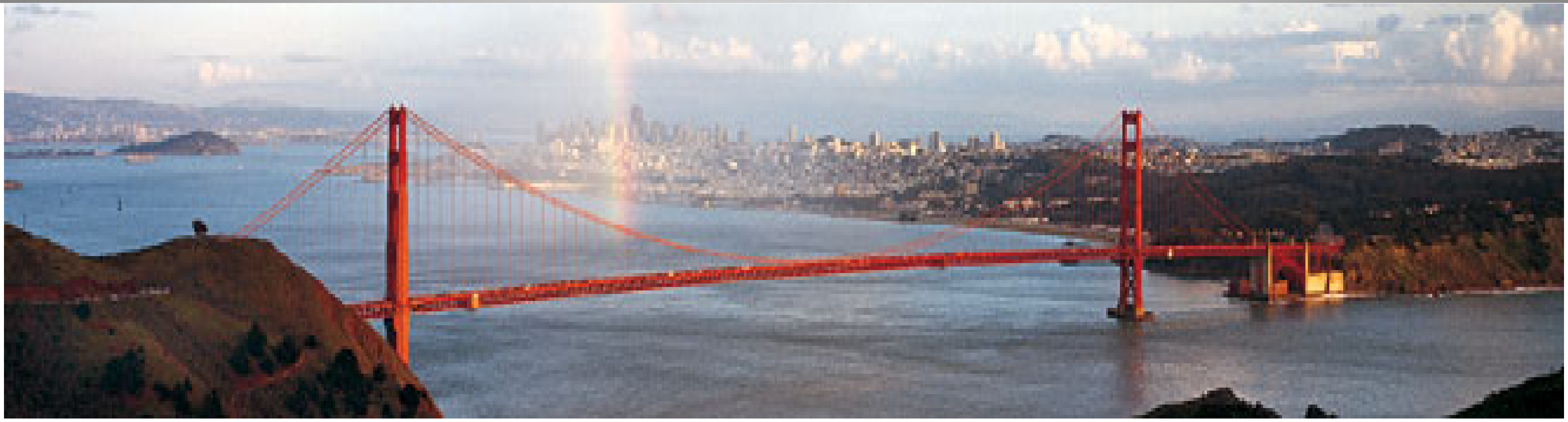
Typical Sensor Data

- Just reams of numbers and time indexes

0x01, 0x33, 0x33, 0x93, 0x11, 0x11, 0x00, 0x91, 0x11, 0x11, 0x19, 0x00, 0x0a, 0xba, 0xdb, 0xae,
0xab, 0xda, 0xaa, 0xa0, 0x82, 0x55, 0x35, 0x34, 0x33, 0x34, 0x32, 0x22, 0x35, 0x24, 0x33, 0x43,
0x32, 0x09, 0xfd, 0xdc, 0xdb, 0xcc, 0xbb, 0xab, 0x99, 0x81, 0x25, 0x35, 0x42, 0x42, 0x32, 0x12,
0x00, 0x89, 0x99, 0x9a, 0x88, 0x12, 0x44, 0x53, 0x43, 0x21, 0x0a, 0xde, 0xcc, 0xcb, 0xbb, 0xca,
0x98, 0x80, 0x23, 0x53, 0x53, 0x33, 0x33, 0x21, 0x08, 0xaa, 0xcb, 0xca, 0xba, 0xa9, 0x00, 0x25,
0x35, 0x43, 0x52, 0x33, 0x32, 0x18, 0x9d, 0xdc, 0xcc, 0xbc, 0xba, 0xba, 0x99, 0x00, 0x23, 0x54,
0x33, 0x43, 0x23, 0x11, 0x00, 0x88, 0x99, 0x98, 0x88, 0x13, 0x45, 0x43, 0x44, 0x22, 0x10, 0x9d,
0xec, 0xcc, 0xbb, 0xca, 0xaa, 0x90, 0x01, 0x35, 0x43, 0x43, 0x32, 0x21, 0x10, 0x88, 0x9a, 0x9b,
0xa9, 0xa8, 0x00, 0x35, 0x44, 0x43, 0x43, 0x41, 0x10, 0xbd, 0xdd, 0xbc, 0xcb, 0xab, 0xaa, 0x80,
0x14, 0x43, 0x53, 0x33, 0x22, 0x11, 0x88, 0x9a, 0x9b, 0xab, 0xaa, 0xa8, 0x82, 0x44, 0x54, 0x43,
0x33, 0x32, 0x0a, 0xde, 0xcc, 0xcc, 0xab, 0xbb, 0xa9, 0x01, 0x35, 0x44, 0x33, 0x33, 0x11, 0x08,
0x99, 0x9a, 0x9a, 0xaa, 0xaa, 0x98, 0x12, 0x64, 0x53, 0x35, 0x22, 0x20, 0x9b, 0xdd, 0xdb, 0xcc,
0xba, 0xb9, 0x98, 0x13, 0x55, 0x33, 0x42, 0x11, 0x18, 0x89, 0xa9, 0xaa, 0x99, 0x89, 0x89, 0x01,
0x35, 0x45, 0x34, 0x33, 0x22, 0x18, 0xbe, 0xdd, 0xbc, 0xcb, 0xab, 0xa9, 0x00, 0x35, 0x43, 0x52,
0x22, 0x00, 0x89, 0xaa, 0xaa, 0xa9, 0x98, 0x00, 0x10, 0x24, 0x44, 0x44, 0x33, 0x42, 0x10, 0x8b,
0xec, 0xdc, 0xbc, 0xba, 0xb9, 0x80, 0x14, 0x44, 0x42, 0x32, 0x11, 0x89, 0xab, 0xcb, 0xa9, 0x98,
0x22, 0x34, 0x33, 0x33, 0x63, 0x34, 0x33, 0x31, 0x0b, 0xfd, 0xcc, 0xbd, 0xab, 0x9a, 0x80, 0x24,

. . .

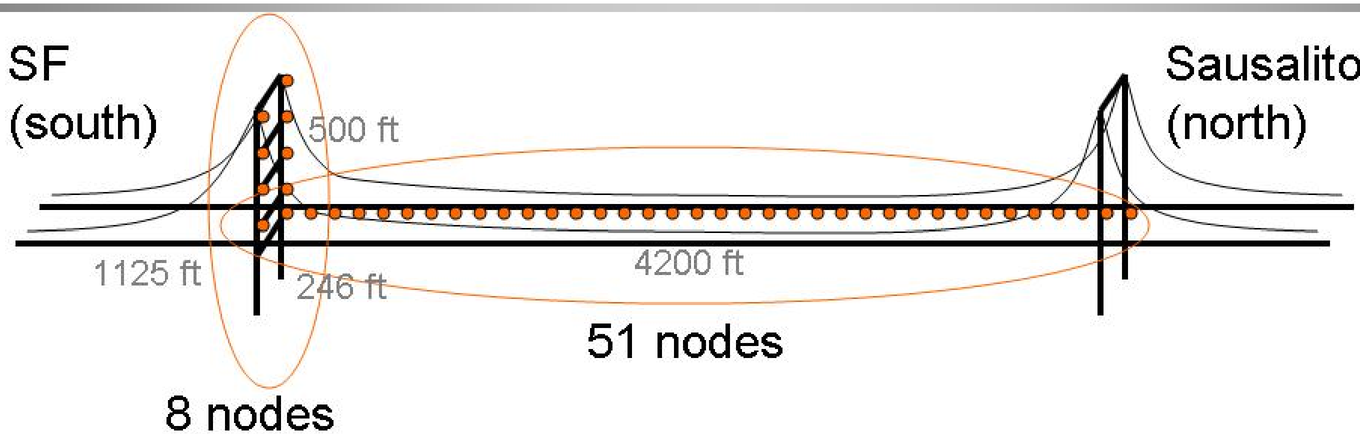
Case Study: Golden Gate Bridge



- Application - Structural Health Management (SHM)
  - » Monitor bridge vibrations to detect anomalies.
  - » X-raying by humans is not foolproof

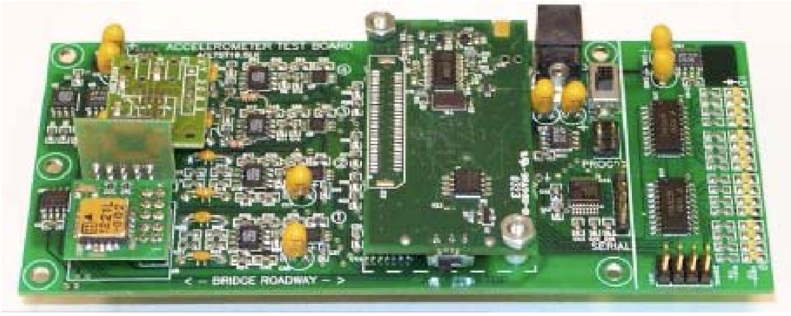
Rusting on the Bridge

Case Study: Golden Gate Bridge - 2



- 59 nodes (motes) along the bridge, using WiFi data transmission. Data forwarded to PC Server which performs data reduction.

- 6,000 readings per minute   100 pkts per second/sensor   (14 bytes/packet)



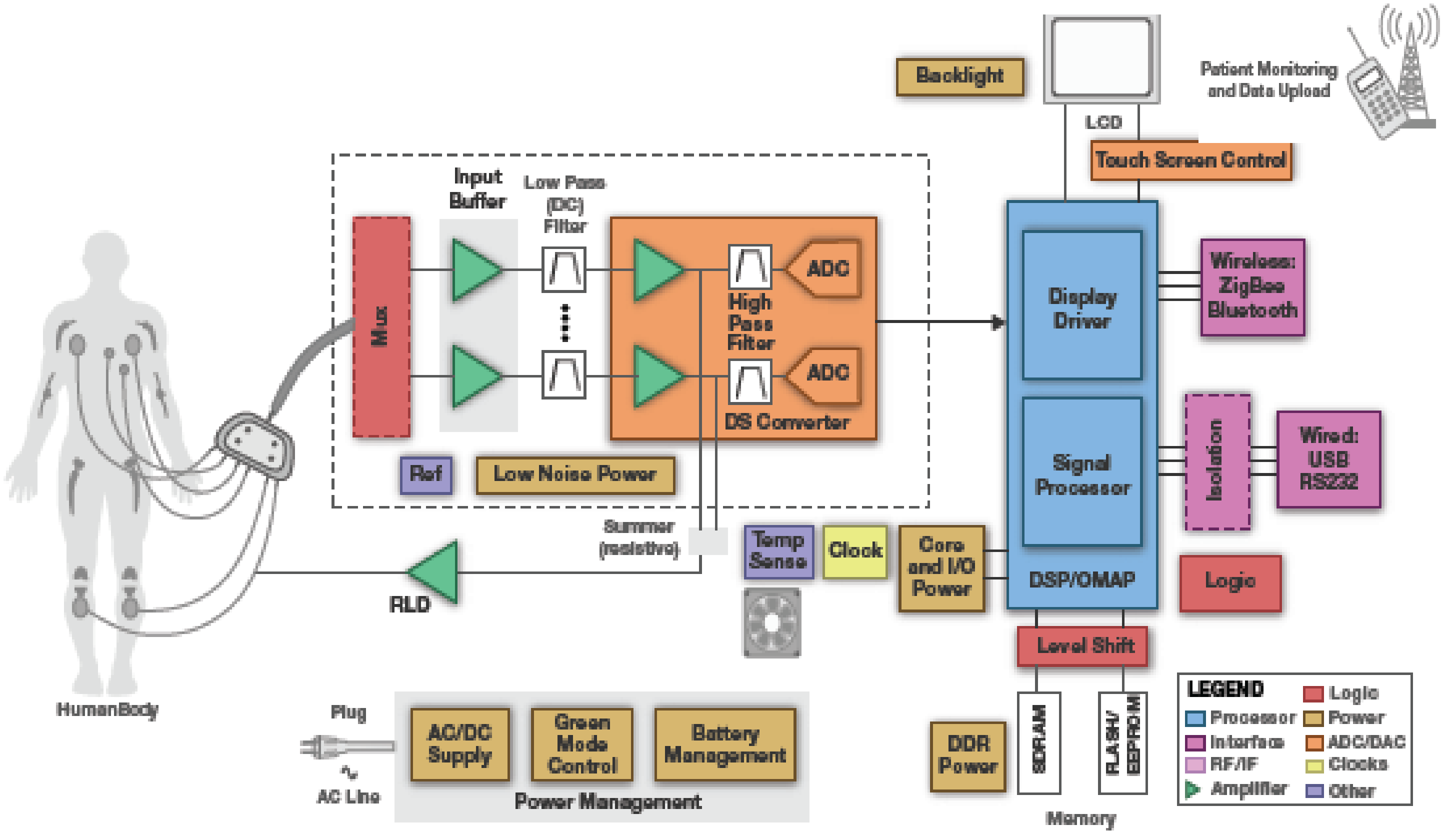
Mica2 motes (@ $ 150)
running TinyOS (RTOS) interfacing to
Accelerometer and Temperature sensors

Case Study: Golden Gate Bridge - 3

- PC Server acts as a network edge “Micro-Broker”

- <u>Raw data is not usable “as is”</u>
  - » Raw data needs to be corrected and cleaned up
    - – Accelerometer readings drift with temperature
    - – Temperatures at the bridge can vary +/- 40 degrees
  - » Raw data needs to be filtered to smooth out temporary “bumps”
    - – Multiple heavy trucks entering bridge at same time
    - – Low level quakes (2.0-3.0 Richter) typically several times per week
    - – Need to apply “low pass” filtering (in Digital Signal-ese) to raw data inputs
  - » <u>Results</u> can <u>then</u> be <u>forwarded to</u> be tracked in <u>database</u>, and to be displayed to operators

Case Study: Medical EKG - 1

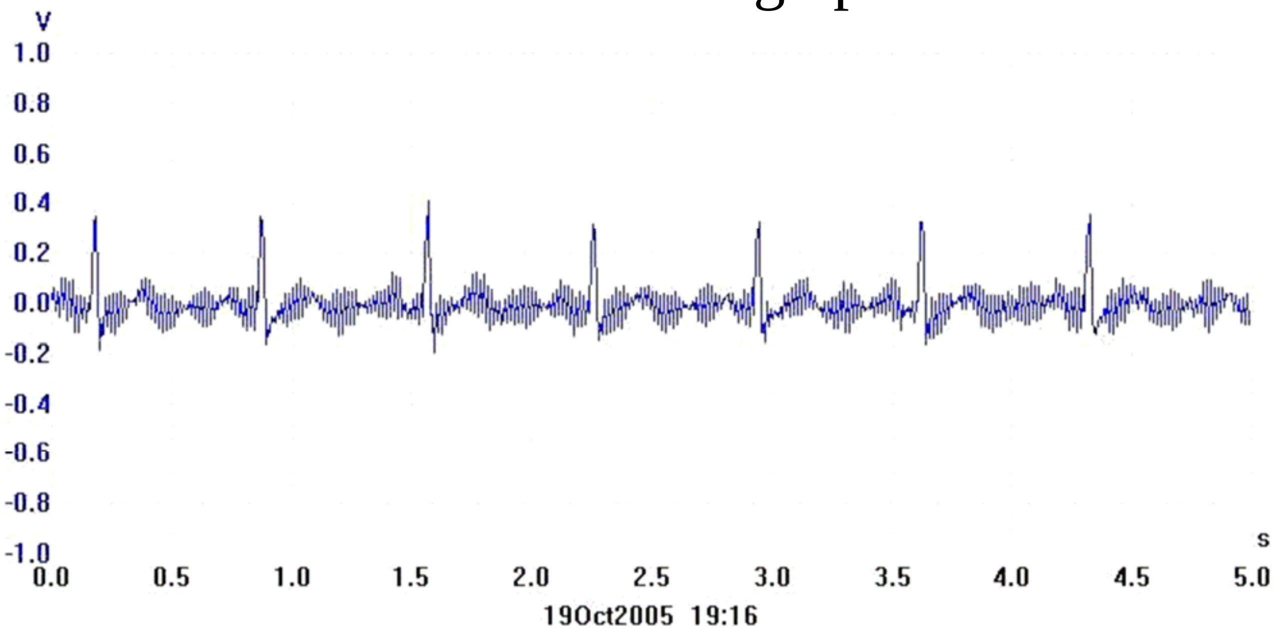


- Electrocardiogram (ECG/EKG) records the voltage generated by the heart, in order to create a pumping action.
- It is a 1 milli-volt impulse that spreads across the heart with a distinctive waveform, to cause a contraction of the heart muscle.
- Patient normally has 3-4 paired electrical leads attached to chest and legs

* Graphics source: Texas Instruments

Case Study: Medical EKG - 2

Cleaning up the Data



EKG recording with typical 60 cycle AC line noise

512 sps = approx 2 ms

- The voltage is typically sampled at 512 sps (samples per second) in order to properly capture the critical 20 millisecond "QRS" part of the heartbeat.
- Human bodies are large antennas, that pick up lots of 60 Hz AC line noise.
- The first step is to clean up the data by filtering out the 60 Hz AC noise.

* Graphics source: Texas Instruments

Case Study: Medical EKG - 3

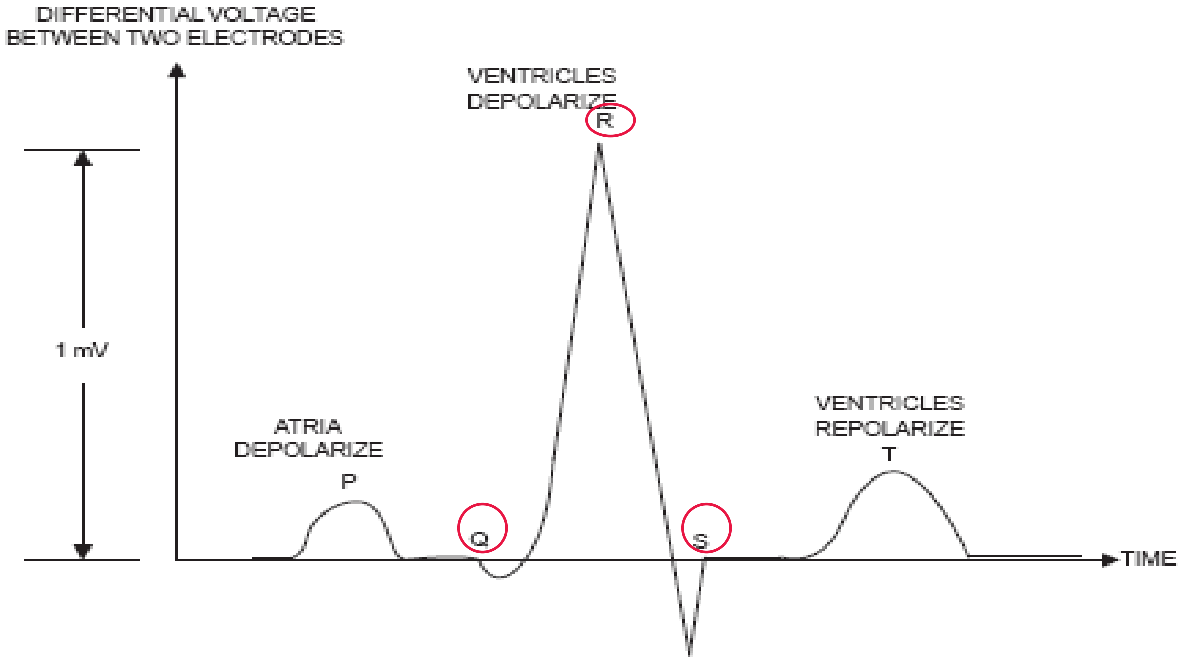


Figure 1. A Typical EKG Waveform

* Graphics source: Texas Instruments

- The "QRS" part of the heartbeat is crucial, since that indicates the overall firing order of the heart. If a patient's QRS signal differs significantly from the norm, it usually indicates a heart disorder.
- To capture the QRS data, and subsequently calculate the associated heart rate, the sampled data points are passed through a (FIR) filter, and the matching data bytes are picked off for processing.
- There are 6-8 leads (2 bytes each), at 512 sps = 512 packets/second of 16-18 bytes

Case Study: Medical EKG - 4

- For historical purposes, it would be ideal to save the EKG data to a database, in order to allow trend-analysis over time, etc
- But, what is the best way to save the data into a database, especially since it is multiple channels, each feeding in at 512 updates/second of binary data ?
- Save it as Binary BLOB data with and XML descriptor off to the side ?
- This style (multiple related inputs, 500-8000 samples/sec) is a repeating pattern in a LOT of Sensor apps, and gets even more complex when doing Image (Ultra-sound, …) apps

Typical EKG Transaction Record

ECG Header

00	80	00	80	00	Packet Number	Heart Rate	Lead Status (Low)	Lead Status (High)

ECG Data

Current Channel Low 8 Bits	Current Channel High 8 Bits

* Graphics source: Texas Instruments

Long Term Trends: Why do we care

- Our databases and transaction systems will have to accommodate the additional needs that intelligent sensor-based systems require:
  - » Sensors often need to be paired with calibration values, which are then used to process incoming Raw data into calibrated form
  - » Sensor data often needs to be aggregated and used in time series type of analysis
  - » Output data is often a series of values (not just a single number) in order to drive analog outputs
  - » Data often needs to be converted into graphical form, including trending and correlation (HMI)

- Projection: over time, Machine-to-Machine communication will dominate all other traffic. It will also be a primary driver (and receiver) of DBMS/TXP interactions

Long-Term Trends

- Increasingly our databases will be interfacing to analog-based data, and will have to adapt to it
  - » Multi-Channel Data (multiple sensors related to each other)
  - » Incoming data rates of 100-8000 updates per second, per device
  - » Acoustic Data (Heartbeats, Ocean Soundings, …)
  - » Image Data (Ultra-Sound, Google Maps, xxxx)
  - » This data normally uses mathematical filters (FIR,IIR) to process the data

- Analysis of Sensor data has some similarities to Business Intelligence Analytics, - but on steroids

- Your wristwatch will talk to your Database, feeding it sensor data.

- Your Database will reside in the PDA / SmartPhone in your Pocket.