CSE 237A
Interfacing, Sensors & Control

Tajana Simunic Rosing
Department of Computer Science and Engineering
University of California, San Diego.
Hardware platform architecture
Sensors & Actuators
Embedded System Hardware
Sensors and Actuators

- **Sensors**
  - Capture physical stimulus
  - Convert it to electrical signal

- **Actuators**
  - Create physical stimulus
  - Given electrical signals
  - Examples:
    - Pneumatic systems, IR, thermal, motors, MEMS
Example: Acceleration Sensor

- MEMS device
- Acceleration
  - Mass displaced from center
  - Resistance of wires changes
  - Model change as acceleration

Courtesy & ©: S. Bütgenbach, TU Braunschweig
Charge-coupled devices (CCD)

Based on charge transfer to next pixel cell
Example: Biometrical Sensors

Example: Fingerprint sensor (© Siemens, VDE):

Matrix of 256 x 256 elements

Voltage ~ Distance

Resistance computed.

Integrated into ID mouse.
Wireless Sensor Network

Data Distribution Network

PDA

Mobile and Stationary Operations

PC

Storage

Animal Monitoring

Precipitation

Temperature, humidity

HPWREN

3d ultrasonic anemometer

Solar radiation

Ship Monitoring

Weather station

In-flight camera

Stationary camera

Seismic

Precipitation

Solar radiation

3d ultrasonic anemometer

Animal Monitoring

PDA

Mobile and Stationary Operations

PC

Storage

Notebook Cellular Phone

Data Distribution Network
Actuators

Microsystems motor

(© MCNC)
Stepper Motor

- Stepper motor: rotates fixed number of degrees when given a “step” signal.
- In contrast, DC motor just rotates when power applied.
- Rotation achieved by applying specific voltage sequence to coils
- Controller greatly simplifies this
Secure Hardware

- Security needed for communication and storage
- Demand for special equipment for cryptographic keys
- To resist side-channel attacks
  - Measurements of the supply current
  - Electromagnetic radiation
- Special mechanisms for physical protection
  - Shielding, sensors for detecting tampering with the modules
- Logical security using cryptographic methods.
- Smart cards: special case of secure hardware
  - Have to run with a very small amount of energy.
Embedded System Hardware
Interfacing Sensors and Actuators

A/D converter
sample-and-hold

sensors

environment

information processing

D/A converter

display

actuators
Output

- Output devices of embedded systems include:
  - Displays, audio
  - Electro-mechanical devices: these influence the environment through motors and other electro-mechanical equipment. May require analog output.

- Naming convention:
Digital-to-Analog (D/A) Converters

Various types, can be quite simple, e.g.:
Embedded System Hardware
Interfacing Sensors and Actuators
Sample and Hold

\( V_e \) is a mapping \( \mathbb{R} \rightarrow \mathbb{R} \)

Digital information processing;

- Discrete time; sample and hold-devices.

Ideally: width of clock pulse \( \rightarrow 0 \)
Analog-to-digital converters

Proportionality

Analog to digital

Digital to analog
Quantization Noise

- quantization noise = approx - real signal

* [http://www.beis.de/Elektronik/DeltaSigma/DeltaSigma.html]
Frequency spectrum of sampled signal

- Let $X_c()$: frequency spectrum of the continuous signal, cut-off frequency $\Omega_N = 2\pi f_N$
- Let $\Omega_s = 2\pi f_s$: sampling frequency
- Let $X_s$: frequency spectrum of the sampled signal
- $X_s$ consists of multiple copies of $X_c$, separated by $\Omega_s$

Cannot be distinguished in sampled signal
Aliasing

- If $\Omega_s < \Omega_N/2$
- copies of spectrum overlap

No problem for signal reconstruction if this is avoided.
Nyquist theorem

Analog input to sample-and-hold can be precisely reconstructed from its output, provided that sampling proceeds at $\geq$ double of the highest frequency found in the input voltage [Nyquist 1928, Shannon, 1949]

Does not capture effect of value quantization: Quantization noise prevents precise reconstruction.
Putting it all together: Control Systems
Control System

- **Objective**: output tracks a reference even in the presence of measurement noise, model error and disturbances

- **Metrics**
  - Stability - Output remains bounded
  - Performance - How well an output tracks the reference
  - Disturbance rejection
  - Robustness - Ability to tolerate modeling error of the plant

- E.g: Cruise control, Thermostat, Aircraft altitude control
Performance

- **Rise time**
  - Time it takes from 10% to 90%

- **Peak time**

- **Overshoot**
  - Percentage by which Peak exceed final value

- **Settling time**
  - Time it takes to reach 1% of final value
Open-Loop Control Systems

- **Plant** - Physical system to be controlled
  - Car, plane, disk, heater,…
- **Actuator** - Device to control the plant
  - Throttle, wing flap, disk motor,…
- **Controller** - Designed product to control the plant
- **Output** - Aspect of the physical system we are interested in
  - Speed, disk location, temperature
- **Reference** - Value we want to see at output
  - Desired speed, desired location, desired temperature
- **Disturbance** - Uncontrollable input to the plant imposed by environment
  - Wind, bumping the disk drive, door opening
Closed Loop Control Systems

- **Sensor**
  - Measure the output
- **Error detector**
- **Feedback control systems**
- **Minimize tracking error**
Controller Design: PID

- Combine Proportional, Integral, and Derivative control to change Manipulated Variable (MV)
  - Use P to control disturbance
  - Use D to control the speed of reduction in error
  - Use I to ensure steady state convergence and convergence rate
- Does not guarantee optimality or stability, is not adaptive

\[
MV(t) = P_{out} + I_{out} + D_{out}
\]
Controller design - P

- Make a change to the output that is proportional to the current error
  - $P_{out}$: Proportional term of output
  - $K_p$: Proportional gain
  - $e$: Error

- P controller will not settle at the final target value
  - retains error that depends on $K_p$ and the process gain

- System can become unstable when $K_p$ is too high

\[ P_{out} = K_p e(t) \]
Controller Design: Integral

- Proportional to both the magnitude of the error and the duration of the error
- Large $K_i$ eliminates steady state errors faster but can cause overshoot

$$I_{out} = K_i \int_0^t e(\tau) \, d\tau$$
Controller Design: Derivative

- Reduces the magnitude of the overshoot produced by the integral component and improves the combined controller-process stability
- Large $K_d$ decreases the overshoot but amplifies the noise in the signal

$$D_{out} = K_d \frac{de}{dt}$$
Controller Design: PID

- Tuning the controller

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Rise Time</th>
<th>Overshoot</th>
<th>Settling Time</th>
<th>S.S. Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_p$</td>
<td>Decrease</td>
<td>Increase</td>
<td>Small Change</td>
<td>Decrease</td>
</tr>
<tr>
<td>$K_i$</td>
<td>Decrease</td>
<td>Increase</td>
<td>Increase</td>
<td>Eliminate</td>
</tr>
<tr>
<td>$K_d$</td>
<td>Small</td>
<td>Decrease</td>
<td>Decrease</td>
<td>None</td>
</tr>
</tbody>
</table>

$$u(t) = MV(t) = K_p e(t) + K_i \int_0^t e(\tau) \, d\tau + K_d \frac{de}{dt}$$
Interfacing
Hardware platform architecture
Bus

- Means for transferring bits
  - wired or wireless

- Bus
  - Connectivity scheme (serial, etc.)
  - Protocol
    - Ports
    - Single function (data bus) or complex protocol (addr, data, ctrl)
    - Timing diagrams
    - Arbitration scheme, error detection/correction etc.
Basic protocol concepts

- Actor: master initiates, servant (slave) respond
- Direction: sender, receiver
- Addresses: special kind of data
  - Specifies a location in memory, a peripheral, or a register within a peripheral
- Time multiplexing
  - Share a single set of wires for multiple pieces of data
  - Saves wires at expense of time

![Time-multiplexed data transfer diagram]
Control methods

1. Master asserts *req* to receive data
2. Servant puts data on bus **within time** $t_{\text{access}}$
3. Master receives data and deasserts *req*
4. Servant ready for next request

**Strobe protocol**

1. Master asserts *req* to receive data
2. Servant puts data on bus **and asserts ack**
3. Master receives data and deasserts *req*
4. Servant ready for next request

**Handshake protocol**
Microprocessor interfacing

- Port-based I/O (parallel I/O)
  - Processor has one or more N-bit ports
  - Processor’s software reads and writes a port just like a register

- Bus-based I/O
  - Processor has address, data and control ports that form a single bus
  - Communication protocol is built into the processor
  - A single instruction carries out the read or write protocol on the bus
Types of IO

Two ways to talk to peripherals

- Memory-mapped I/O
  - Peripheral registers occupy addresses in the same address space as memory

- Standard I/O (I/O-mapped I/O)
  - Additional pin (M/IO) on bus indicates whether a memory or peripheral access
Programming I/O

- Instructions for I/O:
  - special-purpose I/O;
  - memory-mapped load/store instructions.
- Intel x86 provides `in`, `out` instructions.
- Most CPUs use memory-mapped I/O.
- I/O instructions do not preclude memory-mapped I/O.
ARM memory-mapped I/O

- Define location for device:
  
  DEV1 EQU 0x1000

- Read/write code:

  LDR r1,#DEV1 ; set up device adrs
  LDR r0,[r1] ; read DEV1
  LDR r0,#8 ; set up value to write
  STR r0,[r1] ; write value to device
Peripheral intermittently receives data, which must be serviced by the processor
- CPU can *poll* to check for data
  - Busy/wait is very inefficient.
  - Hard to do simultaneous I/O.
- Peripheral can *interrupt* the processor when it has data

Interrupt-driven I/O
- Extra pin: if Int is 1, CPU jumps to ISR
- “polling” of the interrupt pin is built-into the hardware, so no extra time taken!
Priorities and vectors

- **Priorities** - what interrupt gets CPU first.
  - **Masking**: interrupt with priority lower than current priority is not recognized until pending interrupt is complete.
  - **Non-maskable interrupt (NMI)**: highest-priority, never masked; often used for power-down.

- **Vectors**
  - what code is called for each type of interrupt.

- Mechanisms are orthogonal: most CPUs provide both.
Generic interrupt mechanism

Assume priority selection is handled before this point.

- CPU acknowledges request.
- Device sends vector.
- CPU calls handler.
- Software processes request.
- CPU restores state to foreground program.

Diagram:

- Intr. priority > curr. priority?
  - Y: ack
    - Y: call table[vector]
    - N: timeout?
  - N: continue execution
- intr?
  - N: ignore
  - Y: bus error
- timeout?
  - Y: continue execution
  - N: vecotor?
Sources of interrupt overhead

- Handler execution time.
- Interrupt mechanism overhead.
- Register save/restore.
- Pipeline-related penalties.
- Cache-related penalties.
Direct memory access

- **Buffering**
  - Temporarily storing data in memory before processing

- **Microprocessor could handle this with ISR**
  - Storing and restoring microprocessor state inefficient
  - Regular program must wait

- **DMA controller more efficient**
  - Separate single-purpose processor
  - Microprocessor relinquishes control of system bus to DMA controller
  - Microprocessor can meanwhile execute its regular program
Peripheral to memory with DMA

1(a): μP is executing its main program. It has already configured the DMA ctrl registers

1(b): P1 receives input data in a register with address 0x8000.
Peripheral to memory with DMA

2: P1 asserts \textit{req} to request servicing by DMA ctrl.

3: DMA ctrl asserts \textit{Dreq} to request control of system bus

\begin{itemize}
  \item Program memory
    \begin{itemize}
    \item \textit{No ISR needed!}
    \item \textit{Main program}
      \begin{itemize}
      \item 100: instruction
      \item 101: instruction
      \end{itemize}
    \end{itemize}
  \end{itemize}

\begin{itemize}
  \item \textit{μP}
  \item Data memory
    \begin{itemize}
    \item 0x0000 \textit{ack}
    \item 0x0001 \textit{req}
    \item 0x8000 \textit{Dack}
    \item 0x8000
    \end{itemize}
  \end{itemize}

\begin{itemize}
  \item System bus
  \item \textit{P1}
\end{itemize}
Peripheral to memory with DMA

4: After executing instruction 100, \( \mu P \) sees \( Dreq \) asserted, releases the system bus, asserts \( Dack \), and resumes execution, \( \mu P \) stalls only if it needs the system bus to continue executing.
Peripheral to memory with DMA

5: DMA ctrl (a) asserts ack, (b) reads data from 0x8000, and (c) writes that data to 0x0001.

(Meanwhile, processor still executing if not stalled!)
Peripheral to memory with DMA

6: DMA de-asserts \textit{Dreq} and \textit{ack} completing the handshake with P1.

Program memory

\textit{No ISR needed!}

\ldots

Main program

\ldots

100: instruction

101: instruction

\begin{itemize}
\item Dreq
\item ack
\end{itemize}
Arbitration: Priority arbiter

- Determine which of multiple peripherals gets service first from single resource (e.g., microprocessor, DMA)
  - Daisy chain
  - Network oriented

- Priority arbiter
  - Single-purpose processor
  - Peripherals communicate with arbiter
Error detection and correction

- Error detection: ability of receiver to detect errors during transmission
- Error correction: ability of receiver and transmitter to cooperate to correct problem
- Parity: extra bit sent with word used for error detection
  - Even/Odd parity: data word plus parity bit contains even/odd number of 1’s
  - Always detects single bit errors, but not all burst bit errors
- Checksum: extra word sent with data packet of multiple words
  - e.g., extra word contains XOR sum of all data words in packet
Communication Protocols
Communication Protocols

Layering
- Lower levels provide services to higher level
- Easier to design
- Physical layer
  - Lowest level in hierarchy
  - Medium to carry data from one actor (device or node) to another

Protocols
- Serial
- Parallel
- Wireless
Serial communication

- Single data wire – transmit one bit at a time
- Higher data throughput with long distances
  - Less average capacitance, so more bits per unit of time
- Complex protocol and interfacing logic
  - Sender needs to decompose word into bits
  - Receiver needs to recompose bits into word
  - Control signals often on the same wire -> increasing protocol complexity
Serial protocol: 8251 UART

- Universal asynchronous receiver transmitter
- Takes parallel data and transmits serially
- 8251 functions are integrated into standard PC interface chip.
Serial communication

- **Parameters:**
  - Baud (bit) rate.
  - Number of bits per character.
  - Parity/no parity.
  - Even/odd parity.
  - Length of stop bit (1, 1.5, 2 bits).

![Diagram of serial communication with time, start, bit 0, bit 1, ..., bit n-1, stop, and a no character at the start]
Serial protocols: $I^2C$

- $I^2C$ (Inter-IC)
  - Two-wire serial bus protocol developed by Philips Semiconductors nearly 20 years ago
  - Enables peripheral ICs to communicate using simple communication hardware
    - appropriate for peripherals where simplicity and low manufacturing cost are more important than speed
  - Data transfer rates up to 100 kbits/s and 7-bit addressing possible in normal mode
  - 3.4 Mbits/s and 10-bit addressing in fast-mode
  - Common devices capable of interfacing to $I^2C$ bus:
    - EPROMS, Flash, and some RAM memory, real-time clocks, watchdog timers, and microcontrollers
Serial protocols: FireWire

FireWire (a.k.a. I-Link, Lynx, IEEE 1394)
- High-performance serial bus developed by Apple Computer Inc.
- Designed for interfacing independent electronic components
- Data transfer rates from 12.5 to 400 Mbits/s, 64-bit addressing
- Plug-and-play capabilities
- Packet-based layered design structure
- Applications using FireWire include:
  - disk drives, printers, scanners, cameras
- Capable of supporting a LAN similar to Ethernet
  - 64-bit address:
    - 10 bits for network ids, 1023 subnetworks
    - 6 bits for node ids, each subnetwork can have 63 nodes
    - 48 bits for memory address, each node can have 281 terabytes of distinct locations
Serial protocols: USB

- USB (Universal Serial Bus)
  - Easier connection between PC and peripherals
  - USB 1.1 has 2 data rates:
    - 12 Mbps for increased bandwidth devices
    - 1.5 Mbps for lower-speed devices (joysticks, game pads)
  - USB 2.0 runs at 480 Mbps; proposals for USB 3.0 up to 3 Gbps
  - Tiered star topology can be used
    - One USB device (hub) connected to PC
    - Up to 127 USB devices can be connected to hub
  - USB host controller
    - Manages and controls bandwidth and driver software required by each peripheral
    - Dynamically allocates power downstream according to devices connected/disconnected
Parallel communication

- Multiple data, control, and power wires
  - One bit per wire
- High data throughput with short distances
- Typically used when connecting devices on same IC or same circuit board
  - Bus must be kept short
    - Long parallel wires result in high capacitance values which requires more time to charge/discharge
    - Data misalignment between wires increases as length increases
- Higher cost, bulky
Parallel protocols: PCI Bus

- PCI Bus (Peripheral Component Interconnect)
  - High performance bus designed by Intel in the 1990’s
  - Interconnects chips, expansion boards, CPU memory subsystems
  - Data transfer rates of 127.2 to 508.6 Mbits/s and 32 bit addressing, extended to 64 bit addressing
  - Synchronous bus architecture
  - Multiplexed data/address lines

- PCI express – serial protocol
  - 250 Mb/sec – 1 Gb/sec
Parallel protocols: ARM Bus

- ARM Bus
  - Designed and used internally by ARM Corporation
  - Interfaces with ARM line of processors
  - Many IC design companies have own bus protocol
  - Data transfer rate is a function of clock speed
  - 32-bit addressing
Real-time Comm. Requirements

- Real-time behavior
- Efficient, economical  
  (e.g. centralized power supply)
- Appropriate bandwidth and communication delay
- Robustness
- Fault tolerance
- Maintainability
- Diagnosability
- Security
- Safety
Electrical robustness

- Single-ended vs. differential signals

Voltage at input of Op-Amp positive $\rightarrow '1';$ otherwise $\rightarrow '0'$

Combined with twisted pairs; Most noise added to both wires.
Differential signals: Evaluation

- **Advantages:**
  - Subtraction removes most of the noise
  - Changes of voltage levels have no effect
  - Reduced importance of ground wiring
  - Higher speed

- **Disadvantages:**
  - Requires negative voltages
  - Increased number of wires and connectors

- **Applications:**
  - USB, FireWire, ISDN
  - Ethernet (STP/UTP CAT 5/6 cables)
  - differential SCSI
  - High-quality analog audio signals (XLR)
Real-time behavior

- Carrier-sense multiple-access/collision-detection (CSMA/CD, Standard Ethernet)
  - No guaranteed response time.

- Alternatives:
  - token rings, token busses
  - Carrier-sense multiple-access/collision-avoidance (CSMA/CA)
    - WLAN techniques with request preceding transmission
    - Each partner gets an ID (priority). After each bus transfer, all partners try setting their ID on the bus; partners detecting higher ID disconnect themselves from the bus. Highest priority partner gets guaranteed response time; others only if they are given a chance.
Time division multiple access

Each assigned a fixed time slot:

- Master sends sync
- Some waiting time
- Each slave transmits in its time slot
- Variations (truncating unused slots, several slots per slave) exist

http://www.ece.cmu.edu/~koopman/jtdma/jtdma.html#classical
Advantages of TDMA-busses over priority-driven schemes

- Provides QoS guarantees in networks on chips
- TDMA resources support temporal composability, by separating resource access of different subsystems
- TDMA resources have a very deterministic timing behavior
- Can be made fault tolerant
- Support error detection
- Support error contention, i.e. a faulty subsystem does not affect the correct behavior of the remaining system
- Often applied for single processor scheduling to enable composable and hierarchical scheduling.
- Example: ARM AMBA-bus

Sensor/actuator busses

Sensor/actuator busses: Real-time behavior very important; different techniques:

- Many wires
- Fewer wires
- Expensive & flexible
Field busses: Profibus

- More powerful/expensive than sensor interfaces; mostly serial. Emphasis on transmission of small number of bytes.

- Examples: **Process Field Bus (Profibus)**
  - Designed for factory and process automation.
  - Focus on **safety**; comprehensive protocol mechanisms.
  - 20% market share for field busses.
  - **Token passing.**
  - $\leq 93.75$ kbit/s (1200 m); 1500 kbits/s (200m); 12 Mbit/s (100m)
  - Integration with Ethernet via Profinet.

[http://www.profibus.com/]
Controller area network (CAN)

- Designed by Bosch and Intel in 1981;
- Originally designed for cars; now used also for:
  - elevator controllers, copiers, telescopes, production-line control systems, and medical instruments
- differential signaling with twisted pairs
- coding of signals similar to that of serial (RS-232) lines of PCs, with modifications for differential signaling.
- arbitration using CSMA/CA
- throughput between 10kbit/s and 1 Mbit/s
- low and high-priority signals
  - maximum latency of 134 µs for high priority signals

www.can.bosch.com
Aircraft communication history

- Until recently, there wasn’t a strong need for networking inside an aircraft.
- When digital technologies were introduced, the communication was limited to digital data link.
- The introduction of digital technologies happened in the "control platform" area not in the "information" area.

Source: Thales
Digital transmission

- **Process Control**
  - sampling and **data transmission**
    - **data**: the digital value of an analog parameter (e.g. speed; height, attitude,....)
    - **transmission**: no response is expected

- **Information systems**
  - based of **information exchange**
    - **information**: a complex set of abstract values (e.g. digital map, flight plan, list of passenger duty free purchase, failure log book....)
    - **exchange**: a response is generally expected, at least to indicated that the information is received.
    - This "complex set of abstract values" give many bytes of data. This is one reason that calls for **higher speed data link**

Source: Thales
Process control requirements

- **Process control** data requires the transmission to be done with a minimum bounded delay.
  - The stability of the flight relies on this transmission
  - Time, integrity and availability are the key drivers.

- Aeronautical response: **ARINC 429** protocol
  - no common shared resource (limited risk of common failure)
    - one source, one line, several receivers
  - the transmitter does not need to know who receives data
  - no time synchronization between the transmitter and receiver
    - common shared time
ARINC 429 overview

- Developed by Aeronautical Radio, Incorporated (ARINC)
- Commonly used standard for the aircraft
- Electrical and data format standard for a 2-wire serial bus with one sender and many listeners.
- Each data is individually identified (by a label) and sent
Information system requirements

- Ensure that the information is transmitted without any error.
  - Data needs to be acknowledged
  - Messages can be sent again in case of error
- Past aircraft uses A429 but added acknowledgement.
**ARINC 629**

- Multi-transmitter protocol where many units share the same bus. It is a further development of [ARINC 429](#) especially designed for the [Boeing 777](#).

- Based on "waiting room" protocol:
  - Each node is assigned a unique number of mini slots that must elapse, with silence on the channel, before data transmission begins.

- **Three (groups of) time-out parameters:**
  - SG — synchronization gap controlling access to the waiting room
  - TGi — terminal gap, the personal time-out of node I
  - TI — transmit interval preventing monopolization of channel
  - TI > SG > max{TGi }

---

*Diagram showing the time-out parameters.*
TTP (Time-Triggered Protocol)

TTP – more than just a protocol
- Network protocol
- Operating system scheduling philosophy
- Fault tolerance approach

Time-Triggered approach
- Stable time base
- Simple to implement the usual stuff
- Cyclic schedules

Source: Dr. Insup Lee
TTP versions

- **TTP/A** (Automotive Class A = soft real time)
  - A scaled-down version of TTP
  - A cheaper master/slave variant
    - Distributed master slave is expensive

- **TTP/C** (Automotive Class C = hard real time)
  - A full version of TTP
  - A fault-tolerant distributed variant
Protocol Layer in TTP/A

1. Application
2. Mapping of TTP-messages to application relevant data elements
3. Communication Network Interface
4. Message Checking and Error Detection
5. Serial Communication Interface
6. Bus Driver
7. Transmission Medium
Polling

- **Operation**
  - Master polls the other nodes (slaves)
  - Non-master nodes transmit messages when they are polled
  - Inter-slave communication through the master
Polling Tradeoffs

- **Advantage**
  - Simple protocol to implement
  - Historically very popular
  - Bounded latency for real-time applications

- **Disadvantage**
  - Single point of failure from centralized master
  - Polling consumes bandwidth
  - Network size is fixed during installation (or master must discover nodes during reconfiguration)
TTP/C

- A time-triggered communication protocol for safety-critical (fault-tolerant) distributed real-time control systems
- Based on a TDMA (Time Division Multiple Access) media access strategy
- Based on clock synchronization
- Fail Silence
  - A subsystem is fail-silent if it either produces correct results or no results at all, i.e., it is quiet in case it cannot deliver the correct service
<table>
<thead>
<tr>
<th>Layer</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host Layer</strong></td>
<td>Application software in host</td>
</tr>
<tr>
<td><strong>FTU CNI</strong></td>
<td>Fault tolerance unit communication network interface</td>
</tr>
<tr>
<td><strong>FTU Layer</strong></td>
<td>FTU Membership</td>
</tr>
<tr>
<td><strong>Basic CNI</strong></td>
<td>Redundancy Management</td>
</tr>
<tr>
<td><strong>RM Layer</strong></td>
<td>SRU Membership</td>
</tr>
<tr>
<td><strong>SRU Layer</strong></td>
<td>Store the data fields of the received frames</td>
</tr>
<tr>
<td><strong>Data Link/Physical Layer</strong></td>
<td>Provide the means to exchange frames between the nodes</td>
</tr>
</tbody>
</table>
Structure of TTP/C System

CNI: Communication Network Interface
TTP: TTP Communication Controller
FTU: Fault Tolerant Unit
(a) Two active nodes, two shadow nodes

(b) Three active nodes with one shadow node (Triple modular Redundancy)

(c) Two active nodes without a shadow node
- Controller to run protocol
- DPRAM (dual ported RAM)  
  - Used for memory-mapped network interface
- BG (Bus Guard)  
  - Hardware watchdog to ensure “fail silent”
- HW must use highly accurate time sources  
  - Even dual redundant crystal oscillators as used for Boeing 777
Cycle in TTP/C

- **TDMA Cycle**
  - One FTU sends results twice
  - Then next FTU sends some results
  - And so on, until back to the next message from the first FTU

- **Cluster Cycle**
  - Cluster cycle involves scheduling all messages and tasks
TTP/C Frame

- I-Frames used for initialization
- N-Frames used for normal messages
Pros and Cons of TTP

- **Advantages**
  - Simple protocol to implement
  - Deterministic response time
  - No wasted time for master polling messages

- **Disadvantages**
  - Single point of failure from the bus master
  - Wasted bandwidth when some nodes are idle
  - Stable clocks
  - Fixed network size during installation
# TTP/A vs. TTP/C

<table>
<thead>
<tr>
<th>Service</th>
<th>TTP/A</th>
<th>TTP/C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock Synchronization</td>
<td>Central Multimaster</td>
<td>Distributed, Fault-Tolerant</td>
</tr>
<tr>
<td>Mode Switches</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Communication Error Detection</td>
<td>Parity</td>
<td>16/24 bit CRC</td>
</tr>
<tr>
<td>Membership Service</td>
<td>simple</td>
<td>full</td>
</tr>
<tr>
<td>External Clock Synchronization</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Time-Redundant Transmission</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Duplex Nodes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Duplex Channels</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Redundancy Management</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Shadow Node</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>
FlexRay

- Developed by consortium: BMW, Ford, Bosch, Daimler-Chrysler, etc.; Specified in SDL
- Combination TTP and the Byteflight’03 protocols.
  - Improved error tolerance and time-determinism
  - Meets requirements with transfer rates >> CAN std.
  
  High data rate can be achieved:
  - initially targeted for ~ 10Mbit/sec;
  - design allows much higher data rates
- TDMA (Time Division Multiple Access) protocol:
  Fixed time slot with exclusive access to the bus
- Cycle subdivided into a static and a dynamic segment.
TDMA in FlexRay

- Exclusive bus access enabled for short time in each case.
- Dynamic segment for transmission of variable length information.
- Fixed priorities in dynamic segment: Minislots for each potential sender.
- Bandwidth used only when it is actually needed.

[Diagram showing TDMA slots and minislots for Nodes A to F]
Time intervals in Flexray

- **Microtick** ($\mu t$) = Clock period in partners, may differ between partners
- **Macrotick** ($mt$) = Basic unit of time, synchronized between partners ($=r_i \times \mu t$, $r_i$ varies between partners $i$)
- **Slot** = Interval allocated per sender in static segment ($=p \times mt$, $p$: fixed (configurable))
- **Minislot** = Interval allocated per sender in dynamic segment ($=q \times mt$, $q$: variable)
  Short minislot if no transmission needed; starts after previous minislot.
- **Cycle** = Static segment + dynamic segment + network idle time
Structure of Flexray networks

- Bus guardian protects the system against failing processors, e.g. so-called “babbling idiots”
Other field busses

- **LIN**: low cost bus for interfacing sensors/actuators in the automotive domain
- **MOST**: Multimedia bus for the automotive domain
- **MAP**: designed for car factories.
- **EIB**: designed for smart buildings.
  - CSMA/CA; low data rate.
- **IEEE 488**: Designed for laboratory equipment.
  - Attempts to use standard Ethernet. However, timing predictability remains a serious issue.
Event vs. time triggered

- **Event Triggered (ET):**
  - Computation/communication triggered by an external event
  - Events are generated by (primarily) state changes in the environment
  - Efficient — only do things when they need to be done; rest and save energy/cpu time/bandwidth/... Otherwise
  - High peak-load if multiple events happen at once
  - Hard to analyze due to asynchronous nature of events

- **Time Triggered (TT):**
  - Computation/communication triggered by progress of a system clock
  - Events happen according to a fixed schedule Inefficient — do things periodically, whether needed or not
  - Enhanced analizability due to easily characterizable load, predictable interaction sequences, bus use, etc.
Comparison of protocols
Wireless communication

- Infrared (IR)
  - Frequencies just below visible light spectrum
  - Diode emits infrared light to generate signal
  - Infrared transistor detects signal
  - Cheap to build but need line of sight, limited range
  - Data transfer rate of 9.6 kbps and 4 Mbps

- Radio frequency (RF)
  - Electromagnetic wave frequencies in radio spectrum
  - Analog circuitry and antenna needed on both sides
  - Line of sight not needed, transmitter power determines range
Bluetooth, Wibree, Zigbee

Bluetooth
- IEEE 802.15.1
- Developed and licensed by the Bluetooth Special Interest Group (SIG)

Wibree
- Being adopted into Bluetooth specification
- New name: Bluetooth Low Energy Technology

ZigBee
- IEEE 802.15.4
- Maintained and published by the ZigBee Alliance
## Side By Side

<table>
<thead>
<tr>
<th></th>
<th>Bluetooth</th>
<th>Wibree</th>
<th>ZigBee</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Band</strong></td>
<td>2.4GHz</td>
<td>2.4GHz</td>
<td>2.4GHz, 868MHz, 915MHz</td>
</tr>
<tr>
<td><strong>Antenna/HW</strong></td>
<td>Shared</td>
<td></td>
<td>Independent</td>
</tr>
<tr>
<td><strong>Power</strong></td>
<td>100 mW</td>
<td>~10 mW</td>
<td>30 mW</td>
</tr>
<tr>
<td><strong>Target Battery</strong></td>
<td>Days – months</td>
<td>1-2 years</td>
<td>6 months – 2 years</td>
</tr>
<tr>
<td><strong>Life</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Range</strong></td>
<td>10-30 m</td>
<td>10 m</td>
<td>10-75 m</td>
</tr>
<tr>
<td><strong>Data Rate</strong></td>
<td>1-3 Mbps</td>
<td>1 Mbps</td>
<td>25-250 Kbps</td>
</tr>
</tbody>
</table>
## Side By Side, continued

<table>
<thead>
<tr>
<th></th>
<th>Bluetooth</th>
<th>Wibree</th>
<th>ZigBee</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Component Cost</strong></td>
<td>$3</td>
<td>Bluetooth + 20¢</td>
<td>$2</td>
</tr>
<tr>
<td><strong>Network Topologies</strong></td>
<td>Ad hoc, point to point, star</td>
<td>Ad hoc, point to point, star</td>
<td>Mesh, ad hoc, star</td>
</tr>
<tr>
<td><strong>Security</strong></td>
<td>128-bit encryption</td>
<td>128-bit encryption</td>
<td>128-bit encryption</td>
</tr>
<tr>
<td><strong>Time to Wake and Transmit</strong></td>
<td>3s</td>
<td>TBA</td>
<td>15ms</td>
</tr>
</tbody>
</table>
Wireless Protocols: 802.11

- IEEE 802.11
  - Standard for wireless LANs
  - Specifies parameters for PHY and MAC layers of network
    - PHY layer
      - handles transmission of data between nodes
      - data transfer rates up to 54Mbps (108 Mbps in 802.11a)
      - operates in 2.4 / 5 GHz frequency band (RF)
    - MAC layer
      - medium access control layer
      - protocol responsible for maintaining order in shared medium
      - collision avoidance/detection
IO Summary

- General-purpose processors
  - Port-based or bus-based I/O
  - I/O addressing: Memory mapped I/O or Standard I/O
  - Interrupt handling: fixed or vectored
  - Direct memory access

- Arbitration
  - Priority arbiter (fixed/rotating) or daisy chain

- Bus hierarchy

- Advanced communication
  - Parallel vs. serial, wires vs. wireless, error detection/correction, layering
  - Serial protocols: I²C, CAN, FireWire, and USB; Parallel: PCI and ARM.
  - Wireless protocols: IrDA, Bluetooth, and IEEE 802.11.