Today: Wired embedded networks

- Characteristics and requirements
- Some embedded LANs
  - SPI
  - I2C
  - LIN
  - Ethernet

Next lecture: CAN bus

Then: 802.15.4 – wireless embedded network
Network from a High End Car
Embedded Networking

- In the non-embedded world TCP/IP over Ethernet, SONET, WiFi, 3G, etc. dominates

- No single embedded network or network protocol dominates
  - Why not?
Embedded vs. TCP/IP

- Many TCP/IP features unnecessary or undesirable in embedded networks
- In embedded networks...
  - Stream abstraction seldom used
    - Embedded networks more like UDP than TCP
    - Why?
  - Reliability of individual packets is important
    - As opposed to building reliability with retransmission
  - No support for fragmentation / reassembly
    - Why?
  - No slow-start and other congestion control
    - Why?
Which is better?
Characteristics and Requirements

- Determinism more important than latency
- Above a certain point throughput is irrelevant
- Prioritized network access is useful
- Security important only in some situations
- Resistance to interference may be important
- Reliability is often through redundancy
- Cost is a major factor
- Often master / slave instead of peer to peer
A Few Embedded Networks

- **Low-end**
  - SPI
  - I2C
  - LIN
  - RS-232

- **Medium-end**
  - CAN
  - MOST
  - USB

- **High-end**
  - Ethernet
  - IEEE-1394 (Firewire)
  - Myrinet
How do you choose one?

- Does it give the necessary guarantees in…
  - Error rate
  - Bandwidth
  - Delivery time – worst case and average case
  - Fault tolerance

- Is it affordable in…
  - PCB area
  - Pins
  - Power and energy
  - $$ for wiring, adapter, transceiver, SW licensing
  - Software resource consumption: RAM, ROM, CPU
  - Software integration and testing effort
Most Basic Embedded Network

◆ “Bit banged” network:
  ➢ Implemented almost entirely in software
  ➢ Only HW support is GPIO pins
  ➢ Send a bit by writing to output pin
  ➢ Receive a bit by polling a digital input pin

◆ Can implement an existing protocol or roll your own

◆ Advantages
  ➢ Cheap
  ➢ Flexible: Support many protocols w/o specific HW support

◆ Disadvantages
  ➢ Lots of development effort
  ➢ Imposes severe real-time requirements
  ➢ Fast CPU required to support high network speeds
SPI

◆ Serial Peripheral Interface
  Ø Say S-P-I or spy

◆ Characteristics:
  Ø Very local area – designed for communicating with other chips on the same PCB
    • NIC, DAC, flash memory, etc.
  Ø Full-duplex
  Ø Low / medium bandwidth
  Ø Master / slave

◆ Very many embedded systems use SPI but it is hidden from outside view

◆ Originally developed by Motorola
  Ø Now found on many MCUs
SPI Signals

- Four wires:
  - SCLK – clock
  - SS – slave select
  - MOSI – master-out / slave-in
  - MISO – master-in / slave-out

- Single master / single slave configuration:
Multiple Slaves

◆ Each slave has its own select line:

◆ Addressing lots of slaves requires lots of I/O pins on the master, or else a demultiplexer
CPOL and CPHA

◆ Clock polarity and clock phase
  ➢ Both are 1 bit
  ➢ Configurable via device registers

◆ Determine when:
  ➢ First data bit is driven
  ➢ Remaining data bits are driven
  ➢ Data is sampled

◆ Details are not that interesting...

◆ However: All nodes must agree on these or else SPI doesn’t work
SPI Transfer

1. Master selects a slave
2. Transfer begins at the next clock edge
3. Eight bits transferred in each direction
4. Master deselects the slave

◆ Typical use of SPI from the master side:
  1. Configure the SPI interface
  2. Write a byte into the SPI data register
     ➢ This implicitly starts a transfer
  3. Wait for transfer to finish by checking SPIF flag
  4. Read SPI status register and data register

◆ Contrast this with a bit-banged SPI
More SPI

◆ SPI is lacking:
  - Sophisticated addressing
  - Flow control
  - Acknowledgements
  - Error detection / correction

◆ Practical consequences:
  - Need to build your own higher-level protocols on top of SPI
  - SPI is great for streaming data between a master and a few slaves
  - Not so good as number of slaves increases
  - Not good when reliability of link might be an issue
I²C

- Say “I-squared C”
  - Short for IIC or Inter-IC bus
- Originally developed by Philips for communication inside a TV set
- Main characteristics:
  - Slow – generally limited to 400 Kbps
  - Max distance ~10 feet
    - Longer at slower speeds
  - Supports multiple masters
  - Higher-level bus than SPI
I2C Signals and Addressing

◆ Two wires:
  - SCL – serial clock
  - SDA – serial data
  - These are kept high by default

◆ Addressing:
  - Each slave has a 7-bit address
    - 16 addresses are reserved
    - One reserved address is for broadcast
    - At most 112 slaves can be on a bus
  - 10-bit extended addressing schemes exist and are supported by some I2C implementations
I2C Transaction

- **Master issues a START condition**
  - First pulls SDA low, then pulls SCL low
- **Master writes an address to the bus**
  - Plus a bit indicating whether it wants to read or write
  - Slaves that don’t match address don’t respond
  - A matching slave issues an ACK by pulling down SDA
- **Either master or slave transmits one byte**
  - Receiver issues an ACK
  - This step may repeat
- **Master issues a STOP condition**
  - First releases SCL, then releases SDA
  - At this point the bus is free for another transaction
Multiple-Master I2C

◆ One master issues a START
  ➢ All other masters are considered slaves for that transaction
  ➢ Other masters cannot use the bus until they see a STOP

◆ What happens if a master misses a START?
  ➢ When a master pulls a wire high, it must check that the wire actually goes high
  ➢ If not, then someone else is using it – need to back off until a STOP is seen
LIN Bus

- Very simple, slow bus for automotive applications
  - Master / slave, 20 Kbps maximum
  - Single wire
  - Can be efficiently implemented in software using existing UARTs, timers, etc.
    - Target cost $1 per node, vs. $2 per node for CAN
Ethernet

◆ Characteristics
  ➢ 1500-byte frames
  ➢ Usually full-duplex
  ➢ 48-bit addresses
  ➢ Much more complicated than SPI, I2C
  ➢ Often requires an off-chip Ethernet controller

◆ Can be used with or without TCP or UDP

◆ Hubs, switches, etc. support large networks

◆ Random exponential backoff has bad real-time properties
  ➢ No guarantees are possible under contention
Embedded TCP/IP

- This is increasing in importance
  - Remember that TCP/IP can run over any low-level transport
    - Even I2C or CAN
  - TCP/IP stacks for very small processors exist

- Drawbacks
  - TCP/IP is very generic – contains features that aren’t needed
  - TCP/IP targets WANs – makes many design tradeoffs that can be harmful in embedded situations

- Good usage: Car contains a web server that can be used to query mileage, etc.

- Bad usage: Engine controller and fuel injector talk using TCP/IP
Summary

◆ Embedded networks
  ➢ Usually packet based
  ➢ Usually accessed using low-level interfaces

◆ SPI, I2C
  ➢ Simple and cheap
  ➢ Often used for an MCU to talk to non-MCU devices

◆ CAN
  ➢ Real-time, fault tolerant LAN

◆ Ethernet
  ➢ More often used for communication between MCUs

◆ Subsequent lectures:
  ➢ CAN bus
  ➢ 802.15.4 – low-power wireless embedded networking