# Computer Networks

## ECE 376 Embedded Systems

## Lectrure #30

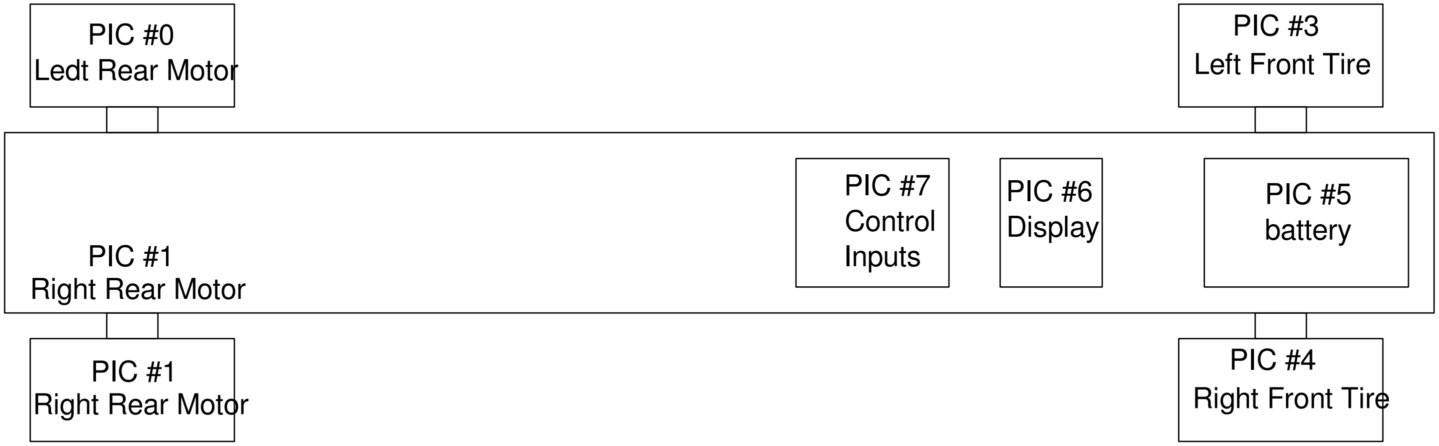
# Networks:

Problem:

- Allow multiple computers to talk to each other
- Set up a protocol so data can get through

Limitation

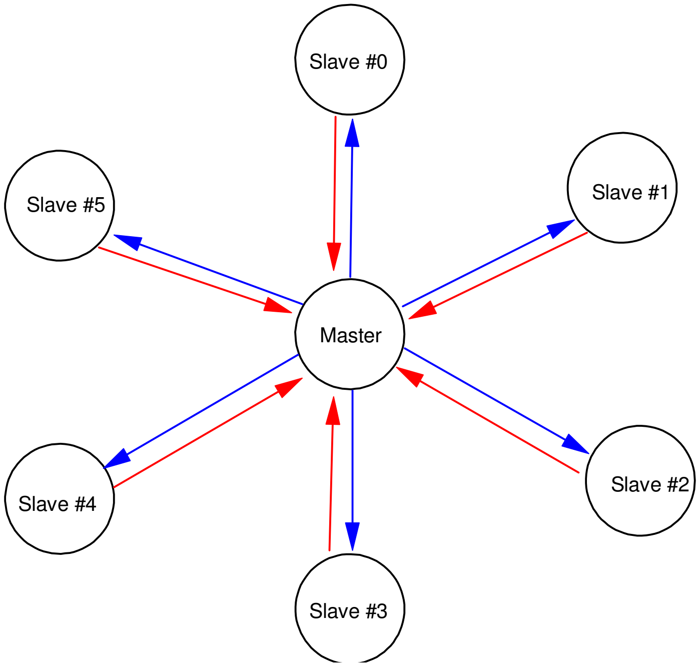
- Only one device can talk on a bus at a time



# Star Network (Master / Slave)

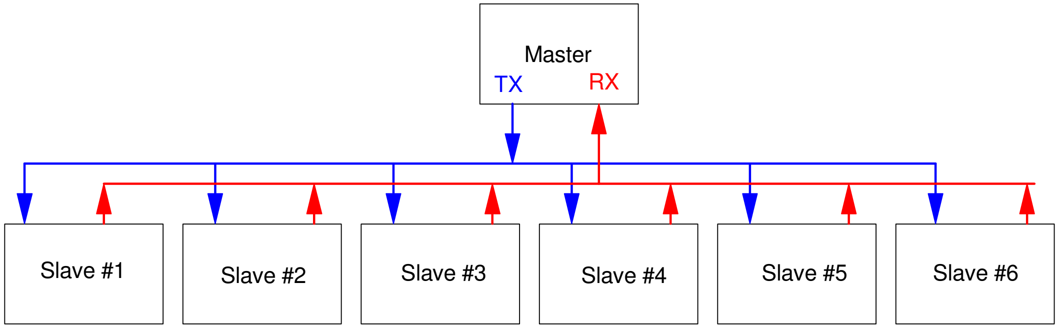
All communications comes from and goes through the master

- Master initiates data transfer
- All data pass through the master



Advantage:

- Easy to set up
- Easy to control
- Easy to scale up



## Star Network Example

Sensor XX:

- Read the voltage on RA0 every Tms
- Take N recordings
- Send data to the master

Message from Master (parsing)

```
S01T200N500
```

- Sensor #1
- T = 200ms (sampling rate)
- N = 500 (number of data points)

```
ID = ( MSG1[1] - 48) * 10
   + ( MSG1[2] - 48) * 1;
T  = ( MSG1[4] - 48) * 100
   + ( MSG1[5] - 48) * 10
   + ( MSG1[6] - 48) * 1;
N  = ( MSG1[8] - 48) * 100
   + ( MSG1[9] - 48) * 10
   + ( MSG1[10] - 48) * 1;

if(ID == 1) {
    for(i=0; i<N; i++) {
       VOLT = A2D_Read(0) * 0.488;
       SCI_Out(VOLT, 3, 2);
       SCI_CRLF();
       LCD_Move(0,0);  LCD_Out(i, 3, 0);
       Wait_ms(T);
       }
    }
```

# Star Network Example #2

Have 4 commands:

- Sxxx: Sensor xx wake up
- Txxx: Sample every xxx ms
- Nxxx: Take xxx samples
- G Start collecting data

Example:

```
T200
N500
S01
G
```

```
if(MSG[0] == 'T') {
   T  = ( MSG1[1] - 48) * 100
      + ( MSG1[2] - 48) * 10
      + ( MSG1[3] - 48) * 1;
      }
 if(MSG[0] == 'N') {
   N  = ( MSG1[1] - 48) * 100
      + ( MSG1[2] - 48) * 10
      + ( MSG1[3] - 48) * 1;
      }
 if(MSG[0] == 'S') {
   ID = ( MSG1[1] - 48) * 100
      + ( MSG1[2] - 48) * 10
      + ( MSG1[3] - 48) * 1;
      }

 if(MSG[0] == 'G') {
  if(ID == 1) {
    for(i=0; i<N; i++) {
      VOLT = A2D_Read(0) * 0.488;
      SCI_Out(VOLT, 3, 2);
      SCI_CRLF();
      LCD_Move(0,0);  LCD_Out(i, 3, 0);
      Wait_ms(T);
      }
   }
```

# ASCII Communications

Less efficient than binary

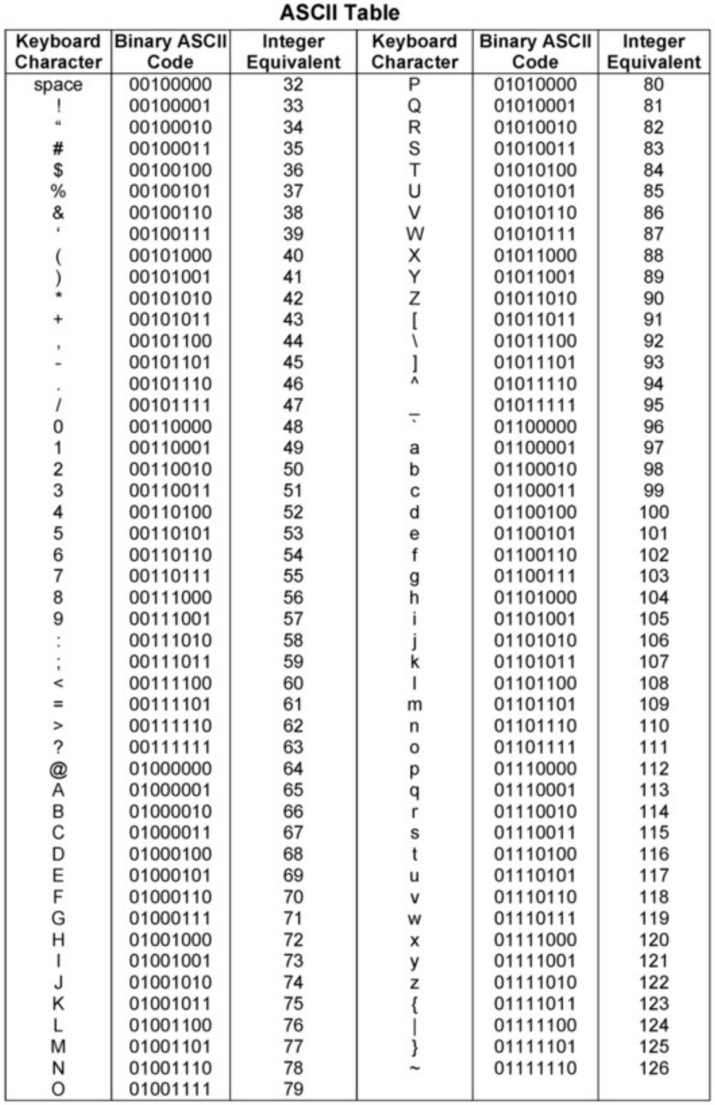
- 255 is three bytes

Allows you to watch the data

- helps in debugging

Allows you to use a computer and keyboard

- Play the role of the master
- Play the role of a sensor



ASCII Table

| Keyboard Character | Binary ASCII Code | Integer Equivalent | Keyboard Character | Binary ASCII Code | Integer Equivalent |
|---|---|---|---|---|---|
| space | 00100000 | 32 | P | 01010000 | 80 |
| ! | 00100001 | 33 | Q | 01010001 | 81 |
| “ | 00100010 | 34 | R | 01010010 | 82 |
| # | 00100011 | 35 | S | 01010011 | 83 |
| $ | 00100100 | 36 | T | 01010100 | 84 |
| % | 00100101 | 37 | U | 01010101 | 85 |
| & | 00100110 | 38 | V | 01010110 | 86 |
| ‘ | 00100111 | 39 | W | 01010111 | 87 |
| ( | 00101000 | 40 | X | 01011000 | 88 |
| ) | 00101001 | 41 | Y | 01011001 | 89 |
| * | 00101010 | 42 | Z | 01011010 | 90 |
| + | 00101011 | 43 | [ | 01011011 | 91 |
| , | 00101100 | 44 | \ | 01011100 | 92 |
| - | 00101101 | 45 | ] | 01011101 | 93 |
| . | 00101110 | 46 | ^ | 01011110 | 94 |
| / | 00101111 | 47 | _ | 01011111 | 95 |
| 0 | 00110000 | 48 | ` | 01100000 | 96 |
| 1 | 00110001 | 49 | a | 01100001 | 97 |
| 2 | 00110010 | 50 | b | 01100010 | 98 |
| 3 | 00110011 | 51 | c | 01100011 | 99 |
| 4 | 00110100 | 52 | d | 01100100 | 100 |
| 5 | 00110101 | 53 | e | 01100101 | 101 |
| 6 | 00110110 | 54 | f | 01100110 | 102 |
| 7 | 00110111 | 55 | g | 01100111 | 103 |
| 8 | 00111000 | 56 | h | 01101000 | 104 |
| 9 | 00111001 | 57 | i | 01101001 | 105 |
| : | 00111010 | 58 | j | 01101010 | 106 |
| ; | 00111011 | 59 | k | 01101011 | 107 |
| < | 00111100 | 60 | l | 01101100 | 108 |
| = | 00111101 | 61 | m | 01101101 | 109 |
| > | 00111110 | 62 | n | 01101110 | 110 |
| ? | 00111111 | 63 | o | 01101111 | 111 |
| @ | 01000000 | 64 | p | 01110000 | 112 |
| A | 01000001 | 65 | q | 01110001 | 113 |
| B | 01000010 | 66 | r | 01110010 | 114 |
| C | 01000011 | 67 | s | 01110011 | 115 |
| D | 01000100 | 68 | t | 01110100 | 116 |
| E | 01000101 | 69 | u | 01110101 | 117 |
| F | 01000110 | 70 | v | 01110110 | 118 |
| G | 01000111 | 71 | w | 01110111 | 119 |
| H | 01001000 | 72 | x | 01111000 | 120 |
| I | 01001001 | 73 | y | 01111001 | 121 |
| J | 01001010 | 74 | z | 01111010 | 122 |
| K | 01001011 | 75 | { | 01111011 | 123 |
| L | 01001100 | 76 | \| | 01111100 | 124 |
| M | 01001101 | 77 | } | 01111101 | 125 |
| N | 01001110 | 78 | ~ | 01111110 | 126 |
| O | 01001111 | 79 | | | |

# RFID Chips

Used for tracking package (ID)

Also can return information

- Temperature
- Vibration

Monitors what you are carrying on you when you leave the store

- Anti-Theft

## RFID Protocol

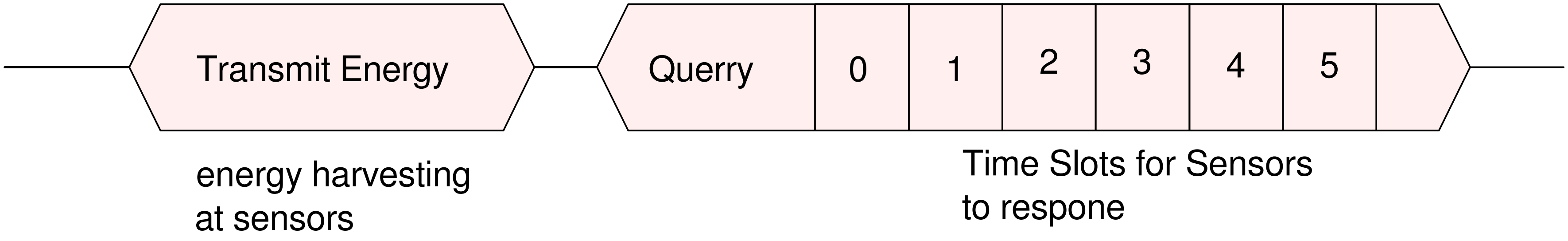
Master intiates conversation

- All sensors in range respond
- Only certain sensors in range respond

Sensors then respond

- Responses can be in one of N time slots
- Each sensor picks a time slot at random
- If there is a conflict, sensor tries again next time slot

It can take several querries to get data from all N sensors

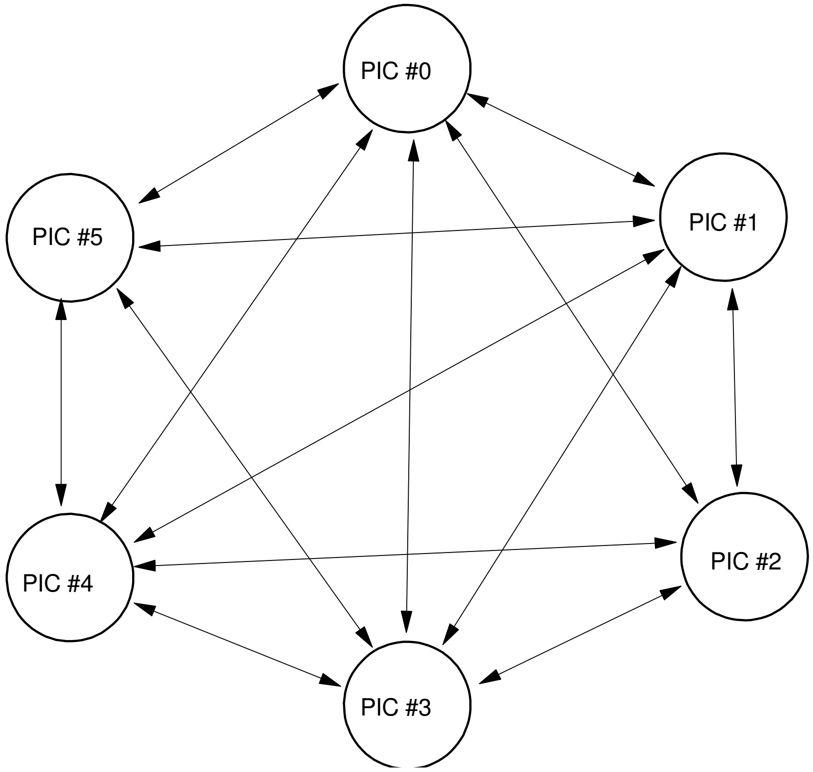

Transmit Energy
Querry
0
1
2
3
4
5
energy harvesting at sensors
Time Slots for Sensors to respone

## Ad-Hoc Network

Any sensor can talk to any other sensor

- Need to add more data to the header
- FxxTyyDzzzz
- From xx To yy Data zzzz

Need to determine what happens if two devices talk at the same time



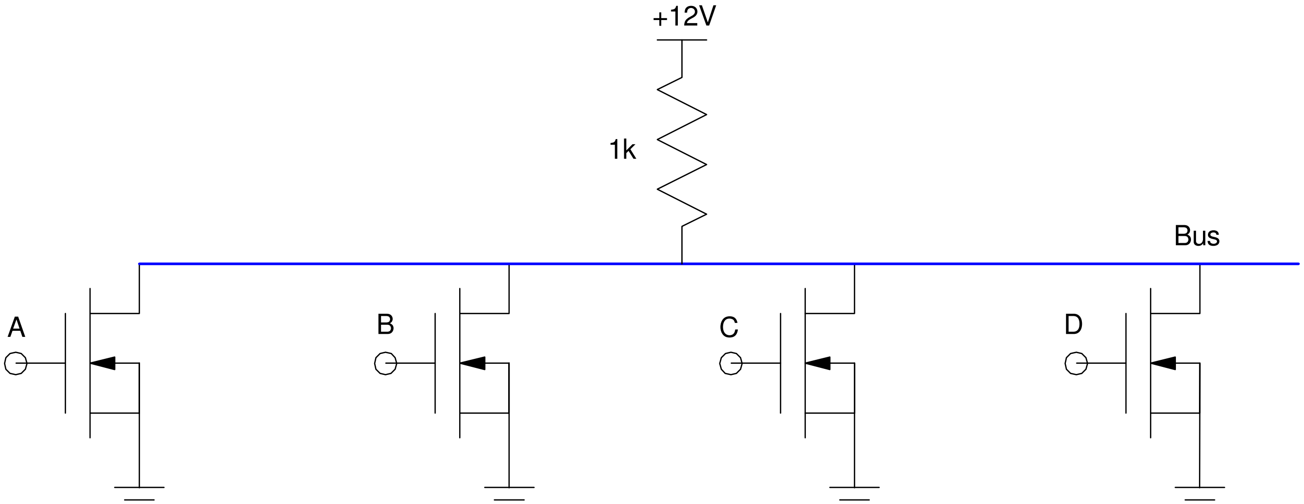
# CAN Bus Protocol

CAN is used in the automotive industry

- Power
- Ground
- Data

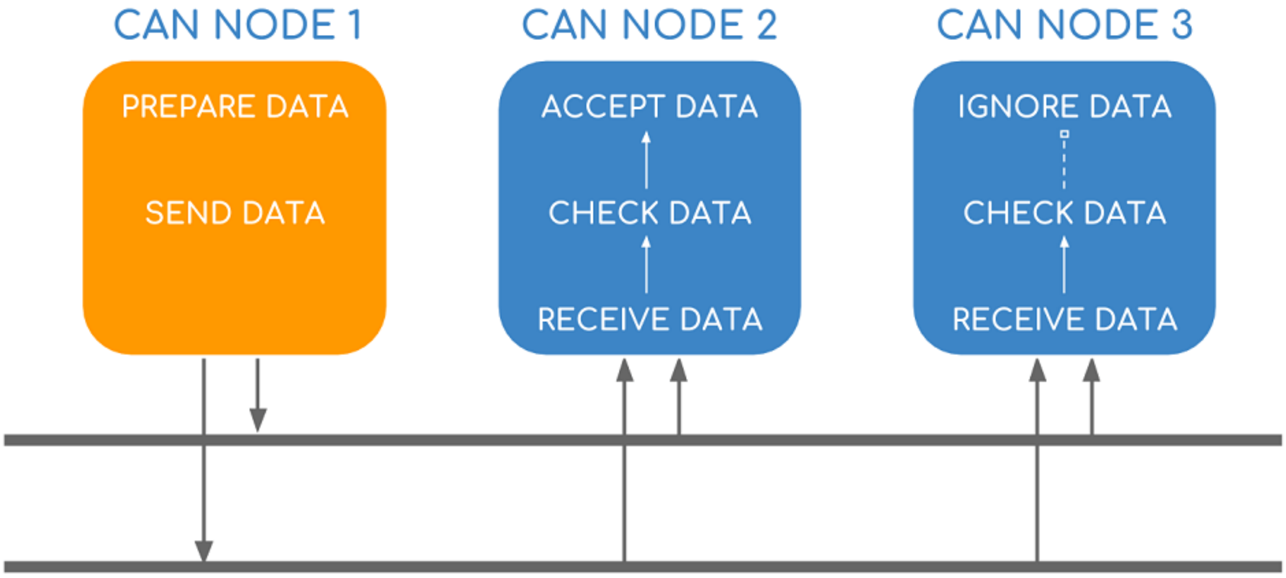
Zero-Priority Encoding

- Uses open-collector logic
- If anyone transmits a zero, zero wins



# CAN Example

| | | | |
|---|---|---|---|
| A | 1101 1011 | 1101 1011 | 1101 1011 |
| B | 1100 0101 | 1100 0101 | |
| C | 1011 1011 | | |
| | | | |
| Bus | 1011 1011 | 1100 0101 | 1101 1011 |
| | C gets through | B gets through | A gets through |



## Additional CAN details

All devices must be synchronized

- They need to know when to send each bit

Max number of consecutive 1's or 0's is three

- If you send 111 or 000, the next bit is a stuffing bit (0 or 1)
- Stuffing bits allow each device to see edges to resync their clocks

When receiving data,

- If you see 111 or 000, ignore the next bit
- It's a stuffing bit