Figure 13.1 Possible Organization of an Embedded System
Figure 13.2  Simplified View of Major Elements of a Multicore Microcomputer
Arduino Uno
Figure 13.3 Typical Microcontroller Chip Elements
Atmega 328 Pinout
Figure 13.4 Host-Target Environment

Host
- Cross-platform development environment

Target
- Kernel
- Root file system
- Boot loader
Figure 13.5  Kernel Compilation

Kernel Source
- From open source or hardware vendor

Kernel Configuration
- Defined according to target hardware and system requirements

Cross Compiler
- Executable on host system

Kernel image
- Executable on target system; ready to be started by boot loader on target system
Figure 13.6  Size of Linux Kernel (shown in GZIP-compressed file size)
Raspberry Pi

SoC - System on a Chip

ARM Processor
512 MB Memory
80 MHz Clock
Figure 13.7 μcLinux/μclibc Software Architecture
module TimerM {
    provides {
        interface StdControl;
        interface Timer;
    }
    uses interface Clock as Clk;
} ...

class TimerC {
    provides {
        interface StdControl;
        interface Timer;
    }
}

implementation {
    components TimerM, HWClock;
    StdControl = TimerM.StdControl;
    Timer = TimerM.Timer;
    TimerM.Clk -> HWClock.Clock;
}

Figure 13.9 Example Component and Configuration
(a) Simplified view of the Surge Application

(b) Top-level Surge Configuration

LED = light-emitting diode
ADC = analog-to-digital converter

Figure 13.10 Example TinyOS Application
unsigned char buffer_empty = true;
cyg_mutex_t mut_cond_var;
cyg_cond_t cond_var;

void thread_a( cyg_addrword_t index )
{
    while ( 1 ) { // run this thread forever
        // acquire data into the buffer ...
        // there is data in the buffer now
        buffer_empty = false;
        cyg_mutex_lock( &mut_cond_var );
        cyg_cond_signal( &cond_var );
        cyg_mutex_unlock( &mut_cond_var );
    }
}

void thread_b( cyg_addrword_t index )
{
    while ( 1 ) { // run this thread forever
        cyg_mutex_lock( &mut_cond_var );
        while ( buffer_empty == true ) cyg_cond_wait( &cond_var );
        // get the buffer data ...
        // set flag to indicate the data in the buffer has been processed
        buffer_empty = true;
        cyg_mutex_unlock( &mut_cond_var );
        // process the data in the buffer
    }
}