킹베디드

1. 스마트 사회란?

“컴퓨터가 보이지 않게 내재되어 네트워크로 연결되어 있고 언제 어디서나 접속이 가능한 환경”

2. 임베디드(Embedded)

“사물에 IC 칩이나 마이크로 프로세서나 마이크로 컨트롤러 등을 내장하는 것을 임베디드(Embeded)라고 한다.”

2.1. 임베디드 시스템 정의

내장된 마이크로 프로세서나 마이크로 컨트롤러가 시스템을 구동하여 그 장비가 해야 하는 **특정한 기능을 수행하도록 프로그램이 내장되어 있는 시스템**이다.

2.2. 임베디드 시스템 특징

- **특정한 기능에 부합하는 최적화 설계가 가능**
- 소형, 경량, 저전력
- 열악한 환경속에의 안정성

2.3. 시스템 제약사항

Small Size, Low Weight
Low Power
Harsh environment
Safety
Cost sensitivity

2.4. 임베디드 시스템과 데스크탑 비교

	Desktop	“Soft” Embedded	“Hard” Embedded/Real-Time
Opertating System	Windows 9x/NT(32bit)	Posix-compliant RTOS Windows NT/CE 2.0 (Interrupt response time >> 20μsec)	Posix-compliant RTOS (TORNAFO, pSOS, QNX, etc) (16-32bit) Interrupt responce times 5-20μsec

“Hard” Embedded는 요청 후 응답시간이 1ms(10^-3) 보다 작다.
데스크 탑과 임베디드의 반응속도 차이
데스크탑 : 1초
임베디드 : 20마이크로세컨드 5 ~ 20, 실시간(Realtime)중요!!!
ex) 신호등 - 교통마비됨

2.5. 임베디드 하드웨어 설계 과정

- **요구기능과 성능 분석**
  어떤 용도로 **어떤 기능**을 갖는 시스템인지 또한 **어느 정도의 성능**을 가져야 하는지 분석
- **하드웨어와 소프트웨어 규격 결정**(저장용량 결정)
  분석된 결과에 따라 **하드웨어와 소프트웨어의 사양**이 결정
- **하드웨어 모듈결정**(Memory)
  결정된 사양에 따라 **프로세서, 메모리 및 주변장치**를 선정
- **회로도 설계**

프로세서, 메모리 및 주변장치 등의 배치를 나타내는
회로도 작성 (Block Diagram)

- **PCB 설계**
  실제 부품을 바탕으로 **물리적으로 각 신호를 연결**
- **하드웨어를 조립하고 시험**
  PCB를 제조하고 하드웨어가 **정상적으로 동작하는지 검사**

3. 마이크로 프로세서와 마이크로 컨트롤러의 차이점

마이크로 프로세서	마이크로 컨트롤러
컴퓨터의 CPU에 해당 주로 범용 시스템 **고성능**	주로 Embedded System에 이용 상대적으로 **저성능**
처리장치만 가지고 있기 때문에 메모리, I/O장치가 필요하다 회로크기, 비용, 전력모소가 크다.	내부에 메모리와 I/O를 가지고 있다. 따라서 회로크기가 작을 뿐만아니라 **저전력**이다. 내부통신을 사용, 빠른처리속도를 가진다.
적은수의 register->주로 메모리작업	**많은 수의 register->프로그래밍이 용이**

4. 라즈베리 파이와 아두이노의 차이점

4.1. 하드웨어

	아두이노	라즈베리파이
Clock speed	**16MHz**	**700MHz**
Processor	**마이크로 컨트롤러**	**마이크로 프로세서**
Multi-tasking	No	Yes
Flash memory	**32KB**	**micro sd card**
Operating system	**None**	**linux, distributions**

4.2. 소프트웨어

	아두이노	라즈베리파이
운영체제	toolkit	pedora , ahchlinux, raspian, android os, firefox, os
개발환경	독자 IDE	Eclipse
프로그래밍언어	Android C	Python
라이브러리	Android Library	Linux 표준 library
기타	H/W중심	SBC

5. 메모리 맵 방식 vs I/O 맵 방식

5.1. 메모리 맵 방식

대부분의 임베디드 시스템에서는 입출력 장치의 **주소공간을 별도로 할당하지 않고 메모리의 일부를 입출력 장치 공간으로 할당하여 사용한다.**
주소가 전체 시스템의 메모리 맵에서 일부가 할당되어 사용된다.

예를들어 주소 0xF0001000에 LED 제어기가 있고, 이때 LED의 주소는 0x100이다. 이 주소에 0x100을 기록하면 LED가 켜진다.

Address	Device
…	DIP
	LED
	..

5.2. I/O맵 방식

입출력 장치 전용의 주소공간을 할당하여 사용한다.
입출력 장치를 액세스하기 위해 **in 또는 out과 같은 별도의 명령**을 사용

in		out	
0x6	LED	0x7	FAN
(주소)	주소에 있는 input device		

5.3. 메모리 맵 방식 말고 I/O방식을 사용하는 이유

소프트웨어 개발자가 하드웨어에 연결된 주소를 알기 위해서

6. 시스템 온 칩(System-on-a-Chip, SoC)

하드웨어 로직 뿐만 아니라 프로세서 ,롬, 컨트롤러, 주변장치의 회로를 하나의 Chip에 집적화하는 기술
CPU, 메모리, DSP, 등을 하나의 칩으로 만드는 기술

7. FPGA(Field Programmable Gate Array)

VHDL을 이용하여 하드웨어 칩을 구현한다.
프로그램으로 하드웨어를 구현한다.

7.1. VHDL(Very high speed integrated circuits Hardware Description Language)

하드웨어를 표현하는 언어

- 사용자가 사용, 이해하기 쉬운 프로그래밍 언어
- 설계시간의 단축
- 복잡한 하드웨어도 VHDL을 이용해서 비교적 간단하게 설계 가능

7.2. VHDL 배경 및 장점

소프트웨어를 이용한 알고리즘 속도 향상의 한계가 있어서 등장했다.
반복 알고리즘 하드웨어로 구현하여 소프트웨어보다 빠른 속도로 계산 결과를 얻을 수 있다.

8. 미들웨어와 플랫폼

미들웨어

- 하드웨어와 소프트웨어 사이에 있는 것
- 응용에서 필요한 공통기능(라이브러리, 코덱 등)을 모두 가지고 있는 것

플랫폼

- 응용이 동작할 수 있도록 하는 기반이다.
- 기차역이 비유해서 설명

9. 범용 OS와 실시간 OS의 차이점

범용 OS
모든 타스크(Task)를 공평하게 처리

실시간 OS
우선권이 높은 타스크부터 처리
시간 제약성, 신뢰성 등을 일반 운영체제 보다 중요시 한다.
일반적으로 한가지 목적에 최적화 되어있다.

10. 리눅스

10.1. 장점

- 사용자 층이 넓어 오류 수정이 빠르고 안정성이 우수하다.
- 기능성과 확장성이 우수하다.

10.2. 단점

- 범용 OS로 설계되어 Real-Time을 지원하지 못한다.

11. 임베디드 리눅스

낮은 성능의 프로세서와 작은 크기의 메모리를 가진 내장형 시스템용으로 개발되었다.
임베디드 시스템을 위해 경량화된 리눅스
임베디드 시스템에 포팅(Porting)된 리눅스

포팅(Porting)
어떤 프로그램을 특정 플랫폼에서 동작될 수 있도록 프로그래밍 하는 기술

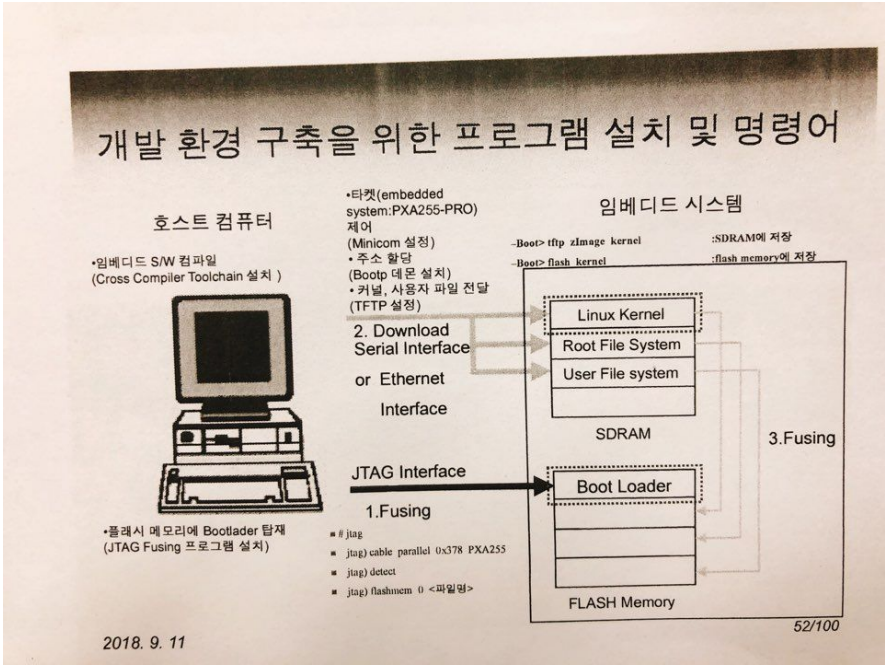
11.1. 장점

- Open Source
- 다양한 CPU Platform 지원
- 개발 비용이 비교적 저렴

11.2. 단점

- 커널크기 크다
- 완전하지 못한 Real Time
- 개발 환경의 부족
- 불투명한 안정성

12. Host system, Embedded System 연결 명령어 및 절차



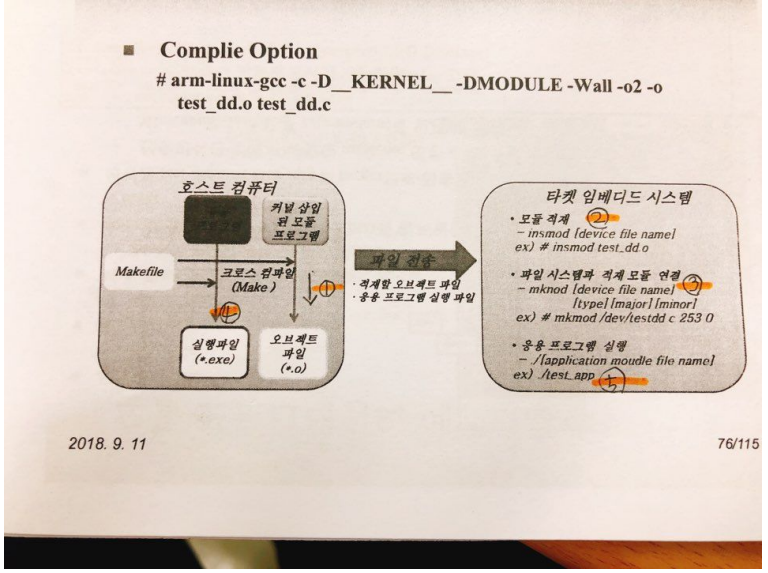
1) Cross Compiler Toolchain 설치를 통해 임베디드 프로그램 컴파일을 한다.
2) JTAG Interface 를 통해 Boot Loader를 Flash Memory 에 탑재
3) Serial Interface 을 통해 Minicom을 설정하여 원격 접속 및 제어를 한다.
4) 주소를 할당하고(Bootp 데몬), 파일 전송 tftp 통신을 하여 SDRAM 에 운영체제 및 파일을 전달한다.
5) SDRAM에 저장된 zImage (커널 Image file)을 Flash Memory 에 저장한다.

※ JTAG : 플래시 메모리에 부트로드 탑재, 소프트 웨어를 주입할 수 있도록 한다.
※ BootLoader : 하드웨어 보드 초기화, 커널 부트 관련 기능, 타겟 보드 상태 검사

1) 호스트 컴퓨터와 임베디드 시스템 연결
2) 임베디드 프로그램 컴파일(Cross Compile Toolchain 설치)
3) 플래시 메모리에 부트로드 탑재(JTAG Fusing 프로그램 설치)
4) 원격 접속 및 제어(Minicom 설정)
5) 임베디드 운영체제 및 사용자 파일 전달
   - 주소 할당(Bootp 데몬)
   - 파일 전송(TFTP 설정)

Fusing : JTAG 이용해 Boot Loader 작동
Download : 다운받아서 운영체계 설치 후 하드디스크로 옮김
Fusing : 램에서 메모리로 옮기는 과정

13. Cross Compile 과정

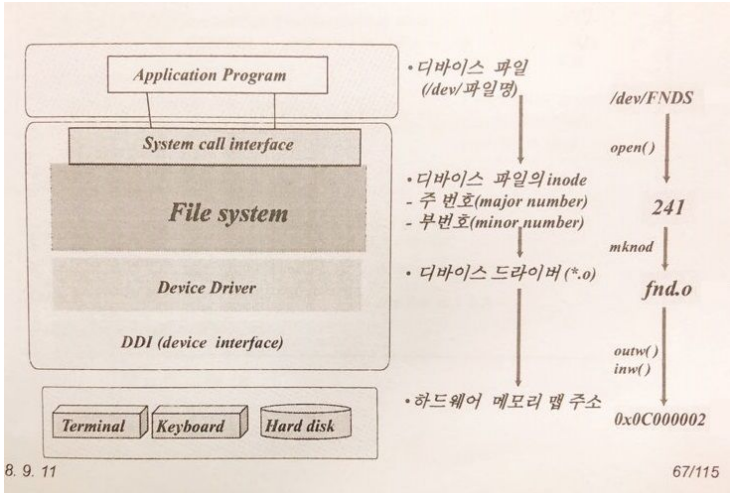


Cross Compile
PC 에서 임베디드 시스템에 맞는 환경으로 컴파일 한다

① Make 명령어로 커널 삽입된 모듈 프로그램 (드라이브) 를 test_dd.o 의 형태로 만든다.
② 모듈적재(드라이브 설치)를 위해 insmod 명령어를 사용한다.
③ 파일시스템과 적재 모듈을 연결한다. mknod 명령어를 사용한다. (하드웨어, 소프트웨어(운영체제) 연결)

※ mknod [device file name] [type] [major] [minor] : 파일 시스템과 디바이스 주소 연결 ex) mknod /dev/testdd c 253 0

④ 실행파일 만듬 (.exe)
⑤ 실행파일을 옮긴뒤 실행한다.(파일전송 프로토콜을 사용)



1) PC에서 임베디드 디바이스 드라이브 프로그램을 크로스컴파일 해준다. 파일 주소(241), 드라이브 이름, 메모리 맵 주소가 들어감
2) 임베디드에 FND 모듈 적재(설치)
3) 파일 시스템과 디바이스 주소 연결(mknod 명령 사용)
4) 어플리케이션 프로그램을 크로스 컴파일해준다. (드라이브 이름이 들어감)
5) 임베디드에서 실행한다.

14. 용어

☆ Cache Memory : 명령어가 저장됨
☆ System : 어떤 목적으로 조직을 만들고, 구성원이 있으며 각자 역할이 있고, 서로 연결되어 있는 체계
☆ calm technology : 말안해도 작동하는 기술 ex) 알람, 가로등