임베디드 엔지니어 교과서, 백상호 (250211)

CH.09. 사물 인터넷/인공지능 시대의 임베디드 소프트웨어 개발

p. 241 ~ 274

섹션41, 산업혁명과 임베디드 기술과의 관계성

스터디에서 다룰만한 내용이 아니라고 봄

교양 수준

그래서 DX까지 발표에서 생략함

1차 산업혁명

- 핵심기술 - 증기기관
- 대표 제품/서비스 - 철도
- 공장생산 - 증기엔진/기계생산

2차 산업혁명

- 핵심기술 - 석유/전기
- 대표 제품/서비스 - 자동차
- 공장생산 - 대량생산

3차 산업혁명

- 핵심기술 - 컴퓨터/인터넷
- 대표 제품/서비스 - PC
- 공장생산 - 자동화
    - FA - Factory Automation
        - 컴퓨터에 의해 제어되는 로봇에 의해 제조라인에서 중요한 역할을 담당
    - PA - Process Automation

        - 기체나 액체를 정제하는 플랜트의 제어를 자동화
    - BA - Building Automation
        - 건물의 공기조절, 엘리베이터, 방범 자동화
    - DA - Distribution Automation
        - 상품의 물류와 반송의 제어 자동화

4차 산업혁명

- 핵심기술 - 사물인터넷/인공지능
- 대표 제품/서비스 - 자동운전
- 공장생산 - 스마트 공장

1차 산업혁명의 의의

직접적인 관계는 없지만, 하드웨어 기술이 발전함

기계장치를 만드는 기술은 오래전부터 있었지만,

1차 산업혁명에 의해 더 복잡한 기술이 생기고 복제되었으며,

증기기관에 의해 장시간의 연속동작이 가능해짐

2차 산업혁명의 의의

에너지원으로 전기/석유를 사용하게 됨

대량생산

화학/철강

인쇄술 진전, 기계식 인쇄기

3차 산업혁명의 의의

임베디드 시스템의 탄생

인터넷, 컴퓨터

공장 자동화

- FA - Factory Automation
    - 컴퓨터에 의해 제어되는 로봇에 의해 제조라인에서 중요한 역할을 담당
- PA - Process Automation
    - 기체나 액체를 정제하는 플랜트의 제어를 자동화

- BA - Building Automation
    - 건물의 공기조절, 엘리베이터, 방범 자동화
- DA - Distribution Automation
    - 상품의 물류와 반송의 제어 자동화

4차 산업혁명의 의의

Industry4.0 라고도 불림

무인화

- FA에서 스마트팩토리 등의 간소화가 추진됨
- 자동차 분야는 자율주행

사람이 생각한 절차와 규칙 => 방대한 데이터를 기반으로 컴퓨터가 스스로 생각한 절차와 규칙

DX와 임베디드 시스템

Digital Transformation

"사회"에서 "장치"를 사용

"장치"는 "데이터"를 생성

생성한 "데이터"는 "클라우드"로 전송

"클라우드"로 모은 데이터는 "빅데이터"

"빅데이터"를 기반으로 "사회"에 가치를 제공

섹션42, DX 시대의 임베디드 시스템 개발

개발기술은 변하지 않지만, 흐름과 소비기간이 달라짐

아키텍처로는 클라우드 지식, 조작계에 활용하는 스마트폰 태플릿 등의 UI 디자인에 대한 지식이 추가로 필요하게 됨

기능배치의 변화

기존

- 특정 기능을 수행하는 전용장비
- 네트워크에 연결되기도 하지만 간단한 접속

현대

- 클라우드에서 정보를 수집 => 데이터 분석 => 가치창출
- 유지보수 부품 및 소모품을 효율적으로 제공
- 이상감지 및 예측 => 사용자 편의성 상승

클라우드로 데이터를 수집/분석함으로써 가치를 창출하는 2가지 유형

1. **고성능 및 소형화된 제품**

   스마트폰에서 가능한 기능은 임베디드 시스템에서라면 더 작고 고성능이어야 함

2. **데이터로부터 도출되는 어드바이스**

   독립형 임베디드 시스템에서 제공한 정보보다 유용한 것을 제공해야 함

예시, 만보계

10만원이 넘는 만보계도 존재

노년층을 대상으로 하여 스마트폰보다 편리한 사용을 추구

전용 웹사이트에서 데이터 축적, 분석, 어드바이스 등의 서비스 이용 가능

고성능 마이크로 컴퓨터가 필요없음 <= 데이터를 클라우드에서 처리하면 장치의 스팩에서 충족하지 않아도 됨

활동량 측정기 (스마트 워치를 말하는 듯 함)

센서, 최소한의 조작 시스템, 스마트폰과의 PAN 접속기능을 구비

스마트폰은 WAN을 통해 클라우드에 접속하며, 조작과 UI는 여기서 제공

결국 장치의 비용절감이 가능해짐

> WAN
>
> Wide Area Network
>
> 인터넷이 가장큰 WAN임

> PAN
>
> Personal Area Network
>
> 사용자의 개인 기기를 중심으로 구성된 소규모 네트워크
>
> 스마트폰과 스마트워치 간의 Bluetooth 연결 등

에지 컴퓨팅

소비자용 장치는 PAN을 통하면 되지만 산업용장비로서의 전용장치는 클라우드에 접속하기 위해 WAN에 대한 엑세스 기능이 필요함

각각의 장치가 직접 WAN에서 클라우드에 접속하면 문제발생

- 비용과 소비전력 상승
- 클라우드로의 접속 및 데이터가 집중되어 처리가 지연됨

에지 시스템

- 대량의 소형장치를 묶어서 데이터를 정리하고, 효율적으로 클라우드 등의 센터에 보고
- 이러면 빠른 응답이 가능, WAN 회선의 트래픽 감소
- 머신러닝을 클라우드에서 처리하지 않고 에지 컴퓨터로 처리하는 경우가 많아지고 있음
    - 인식판단이 빨라짐

웹서비스의 문턱은 낮아지고 있지만, 임베디드 시스템의 제공은 이럴수 없음

- 장치의 프로토타입, 양산, 출하, 유통의 과정이 필요하기 때문
- 라이프사이클이 짧아지고 있음
- 리드타임도 짧아지고 있음 (상품을 기획~제품출시 까지의 기간)

프로토타입

- 기능 및 성능평가를 위해 만들어짐
- 제품평가를 목적으로 함
- 사양이 거의 결정된 상황에 만들어짐
- 평가 결과에 따라 각종 수정이 이루어지고 제품으로 출하됨

PoC

Proof of Concept

- 사양을 결정하기 전, 콘셉트를 결정할때 작성됨
- 예비 사용자에서 피드백을 얻을 목적으로 하여
    - 세부적인 개선점이 아니라,
    - 보다 큰 콘셉트와 제품의 방향성을 판단할 재료를 얻음

- PoC의 목적은 상품인 임베디드 시스템이 비즈니스로서 성립하는지를 확인하는 것
- PoC가 시장에서 받아들여질지를 테스트하고, 그것의 결과 데이터로부터 콘셉트의 좋고 나쁨을 파악함
- 현재 방향과 다른 방법의 아이디어를 추출하고 기록
- 최근은 프로토타입보다 PoC를 만드는것이 주목받고 있음

린 스타트업

Lean Startup

- 새로운 상품이나 서비스의 위험을 줄이면서 실현해 나가는 방법론
- 일종의 PoC인 **MVP**를 구축함
  - **Minimum Viable Product**
  - 아이디어를 바탕으로 상품이나 서비스를 구축
- 최소한의 실용성을 지닌 제품(=MVP)을 만들고, 사용자로부터 체험 피드백을 받아 실용 정도를 측정함
- 측정을 통해 향후 액션을 생각함
  - 아이디어 추려내기, 기능 추가 등을 통해
  - 지금까지와는 다른 것들을 구축하고 다시 측정함
- 피드백 루프시간 최소화가 중요함
  - 웹과는 다르게 자동적인 측정이 힘들기 때문

섹션43, 임베디드 엔지니어의 학습방법

임베디드 기술의 진화는 비교적 느리다고 알려져 있음

사실이지만, 거짓이기도 함

> 사실
>
> - 미션 크리티컬 or 고성능이 아닌, 저가의 임베디드 시스템에서는 오래된 기술을 사용하고 있음
> - 신뢰성의 관점에서는 옛날의 기술이 실적이 풍푸하므로 오작동을 줄일 수 있음

> - 오래되어 저렴하고 대량입수가 가능한 부품을 사용하는게 이득인 상황이 있음

> 거짓
>
> - 사물인터넷, 인공지능을 사용하는 임베디드 시스템은 최신 기술을 사용함

- 임베디드 엔지니어들은 최신 기술을 따라잡으며 업무를 진행해야함
- 그러나 최신 기술도 기반기술이 있고, 이것을 응용하여 최신기술이 탄생함
- 새로운 기술을 두려워하지 말고, 새로운 기술을 사용한다는 기쁨을 즐길것

수파리 이론

- **수** : 스승의 가르침을 준수
- **파** : 잘 연마한 기술을 응용하여 개인에 맞는 독창적인 방식을 창출
- **리** : 기존의 것에 연연하지 않고 스승을 떠나 보다 성장한 단계
- 신입 엔지니어는 선배의 노하우와 일하는 방식을 확실하게 마스터할것
- 아류의 기술은 나중의 성장에 방해가 될 수 있음
- 단, 배속된 현장의 방식이 항상 올바르다는 보장은 없음

표준적인 개발방법 배우기

- 연구회 및 세미나 수강
- 자격시험 응시
  - 필요성을 느끼지 못하더라도, 기술을 체계적으로 배울 수 있다는 장점이 있음
  - 누적된 기반과 체계를 파악하고 기출문제를 풀어가며 기술을 폭넓게 이해할 수 있음
  - 기업에따라 자격보유로
    - 기술수당 or 합격 축하금이 지급되기도 함
    - 승진조건
    - 우대조건

정보를 제공하면 다른정보도 수집됨

- 사내에서 개선을 위한 제안 및 작업 사례로 정보공유를 도모하고 작업표준에 반영
- 사외로는 윗선에 허락을 얻고 깃허브에 공유하고 IT 커뮤니티에 기여
- 기술자 커뮤니티 참여도 추천

섹션44, IDE 설치

- 아두이노 IDE 설치
- 이미 저번에 다 함

섹션45, Yocto 빌드환경의 준비

Yocto

https://docs.yoctoproject.org/

- 하드웨어 아키텍처와 관계없이, 임베디드 제품 및 기타 대상 환경을 위한,

  맞춤형 Linux 기반 시스템을 개발자가 만드는 데 도움이 되는

  오픈 소스 협업 프로젝트
- 사용자 정의, 확장 가능, 유연한 빌드 시스템을 제공함
- 그 자체로 리눅스 배포판은 아님
- **Poky**
  - OpenEmbedded 기반의 참조 빌드 시스템
- **dash**
  - 데비안과 우분투에서 사용되는 가볍고 빠른 셸
- **Linux image**
  - Linux 운영 체제의 컴파일된 버전, 커널, 루트 파일 시스템, 부트로더와 같은 부분을 포함하는 바이너리 파일

설치과정

dash 설치

```
sudo dpkg-reconfigure dash
```

- 디폴트 쉘로 설정하지 말것

```
$ sudo apt-get update
```

- 책에서 설명하는 thud는 yocto 2.6 버전의 코드명이며 python2를 사용함
- 이는 ubuntu 22.04에서 지원하지 않으므로 전부 무시하고 yocto4, python3 기준으로 자체적으로 작성한 명령어를 따름

설치

- 설치 명령어 본문 - Python2 때문에 실패함

```
$ sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib build-essential chrpath libsdl1.2-dev xterm cmake subversion coreutils unzip texi2html texinfo docbook-utils fop gawk python-pysqlite2 make gcc xsltproc g++ desktop-file-utils libgl1-mesa-dev libglu1-mesa-dev autoconf automake groff libtool libxml-parser-perl
```

- 명령 분석
- **Yocto, 임베디드 Linux 개발, 교차 컴파일** 에 필요한 다양한 빌드 도구와 종속성을 설치

1. **필수 빌드 도구**

패키지	목적
gawk	GNU awk, 텍스트 처리에 사용됨.
wget	외부 파일을 다운로드합니다.
git-core	Yocto 소스를 가져오기 위한 버전 제어 시스템.
diffstat	패치 파일의 차이점을 요약합니다.
unzip	보관 자료를 추출합니다 .zip.
texinfo	GNU 문서 형식을 처리합니다.
gcc-multilib	32비트 및 64비트 바이너리를 빌드할 수 있습니다.
build-essential	gcc, g++, make, 및 필수 개발 도구를 설치합니다 .
chrpath	런타임 라이브러리 검색 경로를 수정합니다.
coreutils	핵심 Unix 명령(ls, cat, cp, 등)을 제공합니다.

2. **추가 컴파일 및 디버깅 도구**

패키지	목적
xterm	디버깅을 위한 터미널 에뮬레이터.
cmake	크로스 플랫폼 빌드 시스템(Make의 대안)
subversion	버전 제어 시스템(Git과 유사하지만 더 오래됨).
texi2html	Texinfo 문서를 HTML로 변환합니다.
docbook-utils	DocBook XML/SGML 문서를 처리합니다.
fop	XML-PDF 문서 프로세서.
xsltproc	XML 스타일시트를 처리합니다.

3. **Python 및 스크립팅 종속성**

패키지	목적
python-pysqlite2	Python 2에 대한 SQLite 데이터베이스 지원(더 이상 지원되지 않으며 필요하지 않을 수 있음).
make	표준 빌드 시스템 도구.
gcc	C 컴파일러.
g++	C++ 컴파일러.

4. **그래픽 및 GUI 라이브러리(X11, Mesa, OpenGL용)**

패키지	목적
desktop-file-utils	.desktopGUI 앱의 파일을 처리합니다 .
libgl1-mesa-dev	OpenGL 개발 헤더.
libglu1-mesa-dev	GLU(OpenGL Utility) 개발 헤더.

5. **Autotools 및 빌드 구성 도구**

패키지	목적
autoconf	스크립트를 생성합니다 configure.
automake	파일을 생성합니다 Makefile.in.
groff	매뉴얼 페이지를 포맷하는 데 사용됩니다.
libtool	공유 라이브러리 구축을 담당합니다.
libxml-parser-perl	Perl을 사용하여 XML 파일을 구문 분석합니다(일부 빌드 도구에 대한 종속성).

- **개선판 설치명령어**

> Python2 대신 Python3 사용

```
$ sudo apt-get install gawk wget git diffstat unzip texinfo gcc-multilib b
uild-essential chrpath libsdl1.2-dev xterm cmake python3 python3-pip
python3-pexpect python3-git python3-jinja2 python3-subunit subversi
on coreutils texi2html texinfo docbook-utils fop xsltproc desktop-file-uti
ls libgl1-mesa-dev libglu1-mesa-dev autoconf automake groff libtool lib
xml-parser-perl
```

- PC가 64비트 환경일 때, 다음의 파일도 설치함

  교과서 - 구버전

  ```
  sudo apt-get install libncurses5-dev
  sudo apt-get install lib32z1
  sudo apt-get install lib32ncurses5 <= 우분투 22.04에서 지원되지 않음음
  sudo apt-get install ncurses—dev
  ```

  개선

  ```
  sudo apt-get update
  sudo apt install -y lz4 liblz4-tool zstd
  sudo apt-get install libncurses5-dev lib32z1 lib32ncurses-dev ncurses-
  dev
  ```

프록시 설정

- 회사에 들어가서 프록시 네트워크를 사용할 경우를 대비하여 이 설정이 필요한듯
- 많은 회사, 대학 또는 기업 네트워크에서는 모든 발신 트래픽이 프록시 서버를 통과하도록 요구하는 보안 정책을 시행하고 있음
- 적절한 프록시 설정이 없으면 Yocto 빌드 프로세스가 소스 코드, 패키지 또는 종속성을 다운로드하려고 할 때 실패할 수 있음

```
$ sudo vi ~/etc/apt/apt.conf
```

- 넣을 내용

```
Acquire::ftp::proxy "<ftp://ID:PASSWORD@proxy> 서버 명:8080/";
Acquire::http::proxy "<http://ID:PASSWORD@proxy> 서버 명:8080/";
Acquire::https::proxy "<https://ID:PASSWORD@proxy> 서버 명:8080/";
```

```
$ vi ~/.wgetrc
```

- 넣을 내용

```
http—proxy=http://ID:PASSWORD@proxy 서버 명:8080/
https—proxy=http://ID:PASSWORD@proxy 서버 명:8080/
ftp—proxy=http://ID:PASSWORD@proxy 서버 명:8080/
```

```
$ vi ~/.bashrc
```

- 넣을 내용

```
export http—proxy=http://ID:PASSWORD@proxy 서버 *8080/
export https—proxy=http://ID:PASSWORD@proxy 서버 *8080/
export ftp—proxy=http://ID:PASSWORD@proxy 서버 명:8080/
```

git을 이용하여 프록시 서버를 통과하기

```
$ sudo apt-get install corkscrew
$ sudo vi ~/usr/local/bin/git-proxy
```

- 넣을내용

```
#!/bin/sh
exec /usr/bin/corkscrew [proxy 서버 명] 8080 $1 $2 ~/.proxy-auth
```

```
$ sudo chmod 755 ~/usr/local/bin/git-proxy
$ vi ~/.bashrc
```

- 넣을 내용

```
export GIT_PROXY_COMMAND=/usr/local/bin/git-proxy
```

```
$ vi ~/.proxy-auth
```

- 넣을 내용

```
ID:PASSWORD
```

섹션46, 라즈베리 파이3의 Yocto 환경 구축

- STM32F411RE의 리눅스 이미지를 만들어보려 했으나 다음의 이유로 불가능
  - Linux에 필요한 MMU(메모리 관리 장치)가 없음
  - 전체 Linux가 아닌 베어 메탈 펌웨어(RTOS 또는 독립형 C 코드)용으로 설계됨
  - Yocto 는 STM32MP1 과 같은 MMU 지원 프로세서에 사용되며 STM32F4에는 사용되지 않음

디렉토리 구성

- sources 이하의 디렉토리는 아래에서 클론함

```
|~/yocto
|--build
|----conf
|--downloads
|--sources
|----meta-openembedded
|----meta-raspberrypi
|----poky
```

라즈베리 파이 3의 환경을 다운로드

```
mkdir -p ~/yocto && cd ~/yocto
git clone git://git.yoctoproject.org/poky.git -b kirkstone
git clone git://git.openembedded.org/meta-openembedded -b kirkstone
git clone git://git.yoctoproject.org/meta-raspberrypi -b kirkstone
```

빌드전에 Yocto의 레이어를 구성

```
$ source sources/poky/oe-init-build-env
bitbake-layers add-layer ../sources/meta-openembedded/meta-oe
bitbake-layers add-layer ../sources/meta-openembedded/meta-python
bitbake-layers add-layer ../sources/meta-raspberrypi
```

환경파일 편집

```
$ vi ~/yocto/build/conf/local.conf
```

- 넣을 내용

```
# Target Machine (Raspberry Pi 3)
MACHINE = "raspberrypi3"
# Download directory (stores fetched source files)
DL_DIR ?= "${TOPDIR}/../downloads"
# Enable systemd (modern init system)
DISTRO_FEATURES:append = " systemd virtualization"
VIRTUAL-RUNTIME_init_manager = "systemd"
DISTRO_FEATURES_BACKFILL_CONSIDERED = "sysvinit"
VIRTUAL-RUNTIME_initscripts = ""
# Graphics Support (Enable VC4 for Weston)
# MACHINE_FEATURES:append = " vc4graphics"
# Enable Serial Debugging
ENABLE_UART = "1"
```

빌드

- CUI의 이미지를 작성

```
$ cd ~/yocto/build
$ bitbake core-image-base
```

- 빌드하는 중간에 멈춘적이 있으면 빌드가 되지 않으므로 아래를 입력

```
$ pkill -9 bitbake
```

```
$ bitbake core-image-base
```

SD 카드의 데이터

- 빌드가 완료되면 라즈베리 파이3용 SD카드 이미지가 만들어짐

```
$ cd ~/yocto/build/tmp/deploy/images/raspberrypi3/
$ ls core-image-base-raspberrypi3.rpi-sdimg
```