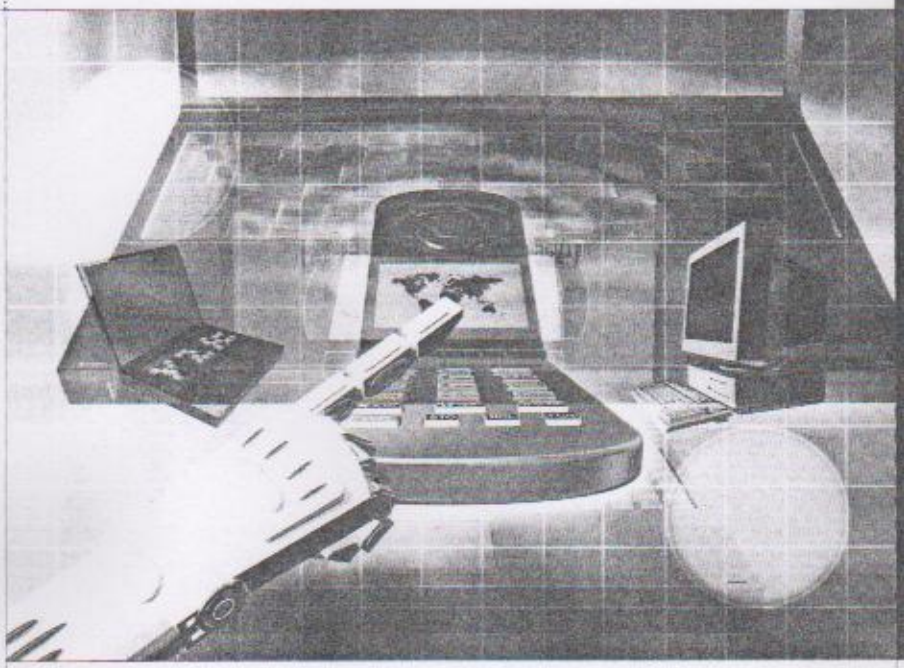
2006 한국모바일학회 추계학술대회

Society of Mobile Technology Fall Conference, 2006



- 일시 : 2006년 11월 23일(목) - 24일(금) 10:00~19:00
- 장소 : 군산대학교
- 주최 : (사)한국모바일학회(www.smt.or.kr)
- 주관 : 군산대학교
- 후원 : 한국인터넷진흥원, KT, 하나로통신, Neoshop, 하이버스, 군산대학교 임베디드 누리사업팀, 군산대학교 정보통신기술연구소

SMT Proceedings of SMT, 2006, Vol.3. No.2

목 차

■ 튜토리얼

■ 발표논문

오전세션 – Track A – 무선 통신 및 센서 네트워크 I (Room 1 : 13419)
(오전 10:30 ~ 11:15) 좌장: 배석찬 교수(군산대)

Break Time(11:15 ~ 11:25)
(오전 11:25 ~ 12:10) 배성한 교수 (세종사이버대)

오전세션 – Track B

– 무선 통신 및 센서 네트워크 II(Room 2 : 13420)

(오전 10:30 ~ 11:15) 좌장: 장경성 교수(초당대)

Break Time(11:15 ~ 11:25)

(오전 11:25 ~ 12:10) 좌장: 김영선 교수(대림대)

오전세션 – Track C

– 광대역 및 멀티미디어 전송(Room 3 : 13523)

(오전 10:30 ~ 11:15) 좌장: 권오병 교수(계원대)

Break Time(11:15 ~ 11:25)

(오전 11:25 ~ 12:10) 좌장: 정형원 교수(광운대)

오후세션 – Track A – 모바일 서비스 및 플랫폼 I(Room 1 : 13419)

(오후 15:00 ~ 15:45) 좌장: 이정식 교수(군산대)

Break Time(15:45 ~16:00)

(오후 16:00 ~ 16:45) 좌장: 지석근 교수(군산대)

오후세션 – Track B – 모바일 서비스 및 플랫폼 II(Room 2 : 13420)

(오후 15:00 ~ 15:45) 좌장: 권창희 교수(한세대)

Break Time(15:45 ~ 16:00)

(오후 16:00 ~ 16:45) 좌장: 김신택 교수(대림대)

오후세션 – Track C – 모바일 정보 서비스(Room 3 : 13523)

(오후 15:00 ~ 15:45) 좌장: 정구민 교수(국민대)

Break Time(15:45 ~ 16:00)

(오후 16:00 ~ 16:45) 좌장: 김영선 교수(대림대)

오후세션 – Track D – 모바일 정책 및 기술 동향(Room 4 : 13421)

(오후 15:00 ~ 15:45) 좌장: 신성윤(군산대)

Break Time(15:45 ~ 16:00)

(오후 16:00 ~ 16:45) 좌장: 권오병 교수(계원대)

■ 튜

• 유비

(오

Polling 메카니즘 기반의 임베디드 소프트웨어 시스템 개발

최제현, 김영철, 김경창*
홍익대학교 컴퓨터정보통신공학과, 컴퓨터공학과*
e-mail : jhchoi@selab.hongik.ac.kr

Development of Embedded Software System Based on Polling Mechanism

Je Hyun Choi, R. Youngchul Kim, KyungChang Kim*
Dept. of CIC and Dept. of CS*, Hongik University, Korea
e-mail : jhchoi@selab.hongik.ac.kr

요약문

본 논문은 한정된 자원을 가진 임베디드 시스템의 효율적 운영을 위한 개발에 초점을 두고 있다. 최근 들어 임베디드 시스템의 활용분야가 역시 점점 확대되고 있다. 우리는 임베디드 시스템에 소프트웨어 아키텍처 관점의 개발을 하고자 한다. 적용사례로 Polling 메카니즘 기반의 로봇 미로 찾기 시스템을 개발을 보인다.

키워드 : 임베디스 시스템, Polling 메카니즘

Abstract

This paper is focused on developing for efficient operation of the embedded system which has the limited resource. Recently practical fields of embedded system are widely being extended. We attempt to develop the embedded software system based on software architecture. this paper contains one example of mobile robot ladyrinth searching system with polling mechanism.

Key Words : Embedded System, Polling Mechanism

Ⅰ. 서론(연구배경)

산업용 전기 및 전자 분야의 경우, 주로 하드웨어 중심으로 제품 개발이 진행되므로, 상대적으로 소프트웨어 부분에 대한 체계적인 개발이 이루어 지지 않았다. 하지만 고기능 사양 및 다기능 제품에 대한 요구가 날로 증가 되므로써, 제품개발에 있어서 소프트웨어가 차지하는 비중은 점차 커지고 복잡해지고 있다. 또한 소프트웨어에 대한 생산성과 품질보증에 대한 요구가 늘어나고 있는 추세에 있다[1].

이는 비단 전기·전자 제품만 해당되는 것은 아

니며, 수많은 소프트웨어 시스템 역시 마찬가지이다.

이처럼 사용자들의 요구가 날로 증가하며, 소프트웨어 시스템이 복잡해짐에 따라 요구를 만족시키고 복잡한 시스템의 한정된 자원을 좀 더 효율적으로 사용하기 위해 이에 따른 여러 가지 다양한 소프트웨어 개발 메카니즘 역시 발전해 왔다.

본 논문에서는 전통적인 소프트웨어 개발 메카니즘인 Polling 메카니즘을 적용하여 미로 찾기 로봇 시스템을 개발하였다.

Ⅱ. 연구 절차

Polling 메카니즘 기반의 임베디드 소프트웨어 시스템 개발의 절차는 [그림 1]과 같다.

소프트웨어 개발 라이프 사이클 기법 중 폭포수 모델(Waterfall model)을 기본으로 하여 Polling 메카니즘 기반의 로봇 미로 찾기 시스템을 개발하고 기타 다른 메카니즘들과의 비교 및 분석을 통해 Polling 메카니즘이 갖는 장·단점에 대해 유추해 보았다.

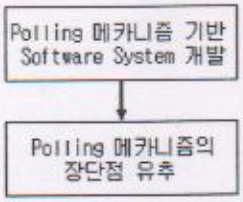


그림 1. 연구 절차

Ⅲ. 로봇 시스템 개발

Polling 메카니즘 기반의 미로 찾기 로봇 시스템 개발을 하드웨어적인 측면과 소프트웨어적인 측면에서 각각 살펴보면 다음과 같다.

1. 하드웨어 소개 및 제원

본 임베디드 소프트웨어 시스템 개발에 사용된 로봇은 기본적인 마이크로 제어 장치 및 관련 개발 제품을 다루는 Parallax.Inc사에서 생산한 보봇(Boe-Bot) 제품으로 대표적인 객체지향 언어인 Java를 통한 동작제어가 가능하다. Boe-Bot의 외형은 [그림 2]와 같다.

미로 찾기 로봇 시스템은 Java를 통해 로봇의 제어 프로그램을 작성하고 이 작성된 코드를 인터프리트하여 바이트코드로 변환한 뒤, 로봇에 전송한다. 전송된 파일은 메모리에 저장이 되고 작성된 제어 프로그램의 동작 원리에 맞게 작동을 하게 된다.

[표 1]은 Boe-Bot 시스템의 제원을 보인다.

그림 2. Boe-Bot

표 1. Boe-Bot의 제원

항 목	제 원
Module Footprint	24-pin DIP Module
Package Measurements(LxWxH)	1.2"x0.6"x0.4"(3.0x1.5x1.0cm)
Operating Enviroment	0° - 70°C (32° - 158°F)
Microcontroller	Ubicom SX48AC
RAM	32Kbytes
EEPROM	32Kbytes
Number of I/O pins	16
Voltage Supply	6 - 24 VDC (unregulated) -or- 5 VDC(regulated)
Voltage regulator current output	0 < Iout < 180mA
Corrent Consumption	60 mA / 13mA nap
Sink/Source Current per I/O	30 mA / 30mA
Sink/Source Current per module	60mA / 60mA per 8 I/O pins
Sink/Source Current per Bank Pins(0-7) and (8-15)	30 mA / 30 mA
Windows Editor/Debugger	Javelin Stamp IDE

2. 시스템 개발 메카니즘

기본적으로 Boe-Bot은 제어에 필요한 소프트웨어 시스템으로 Polling 메카니즘만을 지원하기 때문에 기타 다른 메카니즘, 예를 들면 Event Driven 메카니즘, Multi Thread 메카니즘 등은 소프트웨어 시스템으로는 동작이 불가능하다. 따라서, Pollig 메카니즘의 실제 로봇 미로 찾기 시스템을 구현한 뒤 Event Driven 메카니즘과 Multi Thread 메카니즘과의 비교를 통해 장·단점을 유추해 보았다.

2.1 Polling 메카니즘

Polling 메카니즘은 컨트롤러가 끊임없이 주변장치에 서비스가 필요한지 물어보는 소프트웨어 기법이다. 컨트롤러로 전송할 준비가 된 데이터를 가질 때, 주변장치는 플래그를 설정하고 컨트롤러는 다음 Poll에서 인지한다. 어떤 플래그들이 설정되었는가에 따라, 컨트롤러가 다른 소프트웨어 루틴으로 점프하여, 이와 같은 다수의 주변장치들이 연속적으로 Poll될 수 있다.

임베디드 시스템에서의 Polling 메카니즘은 [그림 3]과 같이 표현 할 수 있다[2].

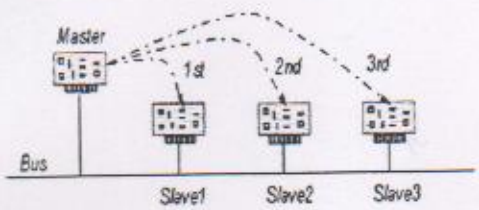


그림 3. Polling 메카니즘

이러한 Polling 메카니즘의 각 주변장치를 지속적으로 Poll 해야 함으로 프로세서 자원의 낭비를 야기할 수 있고, 오버헤드가 발생할 수 있으며, 전송 데이터의 응답시간이 길어질 수 있다는 단점이 있다.

2.2 Event Driven 메카니즘

Event Driven 메카니즘은 사용자 또는 시스템에 의해 어떠한 사건이 발생되면 시스템이 이 사건의 처리를 위한 작업을 수행하는 방식이다[3].

기존의 Polling 방식은 동기 전송/수신 파라다임에 의존한다. 예를 들어, Subsystem A는 정기적인 데이터 취득을 위해 Subsystem B에 연결되어 있어야하는데, 이로 인해 앞서 언급한 Polling 메카니즘의 단점과 같은 문제들이 야기될 수 있다.

따라서 이러한 문제를 해결하는 좋은 방법은 비동기 푸시 혹은 사건 주도의 파라다임을 허락하는 것이다. 즉, 접속을 유지할 필요 없이 Subsystem B는 변화가 감지되었을 때 Subsystem A에 메시지를 보낸다.

2.3 Multi Thread 방식

Multi Thread 방식 즉, 다중 스레드 방식은 CPU이용의 기본 단위가 되는 스레드를 다수의 제어 스레드를 통해 동시에 하나 이상의 작업을 동시에 처리하는 방식을 말한다. 하나의 응용 프로그램이 여러 개의 비슷한 작업을 수행해야 하는 상황에서 매우 유용하며, 응답성이 증가되고, 같은 주소 공간 내에 있는 다른 스레드와 자원을 공유할 수 있다는 장점을 갖는다[4].

Ⅳ. 개발 단계

본 Polling 메카니즘 기반의 임베디드 소프트웨어 시스템 개발에는 기본적으로 소프트웨어 개발 프로세스 기법 중 폭포수 모델을 적용하였다[5].

1. Use case diagram

Use case diagram은 시스템과 외부적 객체와의 상호작용을 표현하며, 요구사항 유도 단계에서 수행한다. 본 시스템의 Use case diagram은 [그림 4]와 같다.

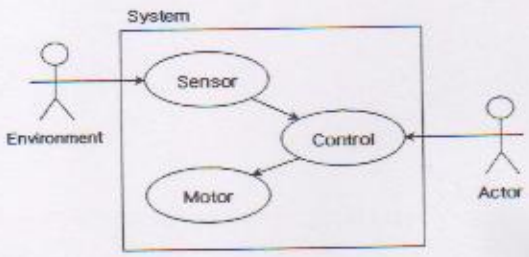


그림 4. Use case diagram

2. Class diagram

그림 5. Class diagram

클래스 다이어그램은 시스템의 정적인 구조를 나타내는 다이어그램이며, 요구사항 분석 단계에서 수행한다. 본 시스템의 클래스 다이어그램은 [그림 5]와 같다.

3. Sequence diagram

시퀀스 다이어그램은 사용자와 시스템 혹은 객체와 시스템 간의 동적인 행위를 기술하는 다이어그램이며, 설계 단계에서 수행한다. 시퀀스 다이어그램을 표현하면 [그림 6]과 같다.

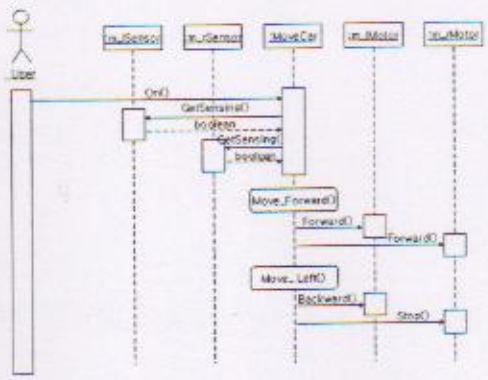


그림 6. Sequence diagram

4. State diagram

스테이트 다이어그램은 하나의 동적 객체의 상태 및 상태 변화를 기술한 다이어그램이며, 소프트웨어 개발 프로세스 중 설계 단계에서 수행한다. 본 시스템의 스테이트 다이어그램은 [그림 7]과 같다[6].

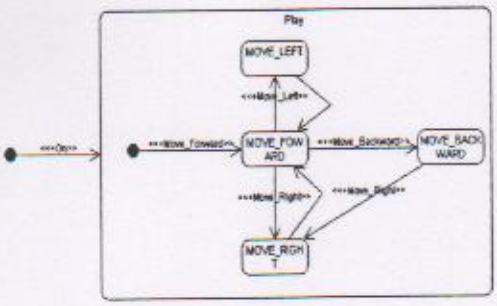


그림 7. State diagram

5. 소스 코드 구현 단계

본 미로 찾기 로봇의 제어에는 대표적인 객체지향 언어인 Java가 사용되었다. Java는 사용이 비교적 간단한 객체지향 언어로서 작성된 자바 프로그램을 중간언어 형태인 자바 바이트코드로 컴파일하고, 이렇게 생성된 자바 바이트코드를 자바 인터프리터가 해석함으로써, 자바 인터프리터와 런타임 시스템이 이식(porting)된 모든 플랫폼에서 자바 바이트코드를 직접 실행할 수 있다는 장점이 있다.

5.1 Motor 클래스

모터를 움직이기 위해서는 왼쪽과 오른쪽의 각 모터의 객체가 생성되어, 앞쪽으로 갈 경우 양쪽의 바퀴가 회전해야하고 오른쪽으로 갈 경우는 왼쪽 바퀴는 정지하고 오른쪽 바퀴는 역회전 해야한다. 그리고 뒤로 가야하는 경우에는 양쪽의 바퀴가 역회전해야 한다. [표 2]에서 바퀴의 구동 명령어는 PWM(속도, 전기적 신호)로 나타내며, 모터 클래스에서는 전진, 후진, 그리고 정지 메소드가 정의되어 있다.

각각의 모터에 대한 객체의 생성은 시스템의 전체적인 컨트롤을 담당하는 MoveCar 클래스에서 이루어진다.

표 2. Motor 클래스

```
import stamp.core.*;
class Motor
{
        //Association
        //Attribute
        protected int m_nPin;
        public final static int SPEED=30;
        public final static int STOP=173;
        public final static int PULSE=2304;
        protected PWM m_cMotor;
        protected char m_direction='s';
        protected char m_chPosition;
        //Function
        public void Forward() {
                //todo:write code
                if(m_direction != 'f')
                {
                        if(m_chPosition == 'l')
                                m_cMotor.update(STOP - SPEED,PULSE);
                        else
                                m_cMotor.update(STOP + SPEED,PULSE);
                        m_direction = 'f';
                }
        }
        public void Backward() {
                //todo:write code
                if(m_direction != 'b')
                {
                        if(m_chPosition == 'l')
                                m_cMotor.update(STOP + SPEED,PULSE);
                        else
                                m_cMotor.update(STOP - SPEED,PULSE);
                        m_direction = 'b';
                }
        }
        public void Stop() {
```

5.2 TouchSensor 클래스

양쪽의 터치 센서에서 들어오는 신호를 감지하여, 신호의 값을 통해 시스템의 이동 방향을 판단하기 위해 센서처리를 하는 클래스이다.

[표 3]에서 Setpin(핀번호) 메소드를 통하여 센서를 감지하는 핀을 초기화하며, GetSensing() 메소드를 통하여 센서의 감지 여부를 파악한다.

모터와 마찬가지로 각 센서에 대한 객체의 생성은 MoveCar 클래스에서 이루어진다.

표 3. TouchSensor 클래스

```
import stamp.core.*;
class TouchSensor
{
        //Association
        //Attributes
        protected int m_nPin;
        //Position
        public boolean GetSensing() {
                //Automatic code
                return CPU.readPin(m_nPin);
        }
        public void SetPin(int pin) {
                //Automatic code
                m_nPin = pin;
        }
};
```

5.3 MoveCar 클래스

전체적인 시스템의 컨트롤을 담당하는 클래스이며, 모터 및 센서의 초기화, 센서를 통해 감지된 신호를 통한 방향 설정 등의 역할을 담당한다.

센서의 신호를 통해 방향을 결정하는 메소드와 신호를 기반으로 전진, 후진, 좌회전, 우회전을 담당하는 각각의 메소드를 가지고 있으며, 전진의 경우에는 양쪽 모터 회전, 우회전일 경우에는 왼쪽 모터 정지 오른쪽 모터 역회전, 그리고 후진의 경우에는 양쪽 모터 역회전이 각 메소드가 포함하고 있는 내용이다.

표 4. MoveCar 클래스

```
public void Move_Forward(int timeCount) {
        //Automatic code
        m_lLed.Off();
        m_rLed.Off();
        m_lMotor.Forward();
        m_rMotor.Forward();
}
public void Move_Backward(int timeCount) {
        //Automatic code
        m_lLed.On();
        m_rLed.On();
        m_input = false;
        if(timeCount < DELAY)
        {
                m_lMotor.Backward();
                m_rMotor.Backward();
        }
        else if(timeCount < DELAY*2)
        {
                Move_Right(timeCount - DELAY);
        }
        else if(timeCount < DELAY*3)
                m_input = true;
}
public void Move_Left(int timeCount) {
        //Automatic code
        m_lLed.On();
        m_lMotor.Stop();
        m_rMotor.Backward();
        m_input = false;
```

6. 테스트

구현된 소프트웨어 시스템의 소스코드를 로봇에 전송하여 실제 미로 찾기 로봇 시스템의 요구사항에 맞게 동작하는지를 테스트 해보았다.

[그림 8]은 간단한 미로 트랙에서 시스템의 테스트를 거치는 과정이다.

테스트 결과 미로 찾기 로봇 시스템은 간단한 미로를 문제없이 통과함을 확인할 수 있었다.

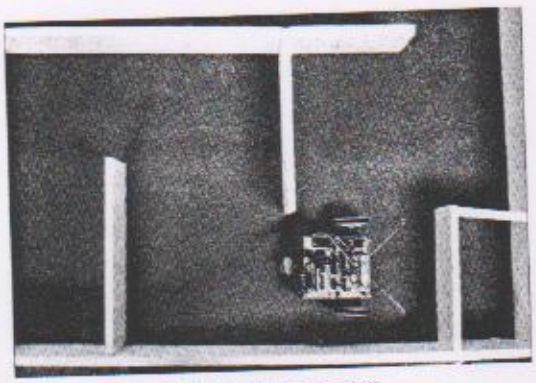


그림 8. 테스트 과정

V. 결론

본 논문은 Polling 메카니즘을 기반으로 소프트웨어 개발 프로세스 중 폭포수 기법을 적용하여, 로봇 미로 찾기 시스템을 개발하였다. 전통적인 임베디드 소프트웨어 시스템 개발 메카니즘인 Polling 메카니즘은 세 개의 CPU 명령 사이클로 구현이 가능하기 때문에 간단하고 기본 연산은 효율적이라는 장점이 있지만, 하지만 Event Driven 메카니즘과 Multi Thread 메카니즘과는 달리 각각의 주변장치를 지속적으로 Poll해야 하기 때문에 시스템 자원의 낭비를 야기할 수 있으며, 이로 인한 오베헤드가 발생할 수 있고 전송 데이터의 응답 시간이 길어 질 수 있다는 단점을 안고 있다. 이러한 Polling 메카니즘의 단점은 시스템이 복잡해질수록 더욱 크게 작용할 수 있기 때문에 최근의 복잡한 소프트웨어 시스템은 대부분 Event Driven 혹은 Multi Thread 메카니즘을 통하여 개발되고 있는 추세이다. 소프트웨어 시스템의 개발에서 자신의 시스템에 가장 적합한 개발 메카니즘을 찾는 것은 시스템 개발의 가장 초기 단계부터 고려해야 할 사항이며, 이를 통해 시스템의 목적에 맞는 효율적 수행 능력을 얻을 수 있을 것이다.

VI. 참고문헌

[1] Lisa Bronsword, Paul Clements, "A Case Study in Successful Product Line Development , CMU/SEI-96-TR-016, 1996

[2] Abraham Silberschatz etc, 조유근, 고건, 김영찬 공역 "Operating System Concepts", 홍릉과학출판사, pp.425-426, 2004.

[3] http://www.winoble.com/WiN_KR/html/bizm/EB_EventDrivenArch.jsp

[4] Abraham Silberschatz etc, 조유근, 고건, 김영찬 공역 "Operating System Concepts", 홍릉과학출판사, pp.125-127, 2004.

[5] Berned Bruegge etc, "Object Oriented Software Engineering", Prentice Hall, pp.629, 2004.

[6] Berned Bruegge etc, "Object Oriented Software Engineering", Prentice Hall, pp.29-65, 2004.

[7] Parallax.Inc, Javelin Stamp Manual Version 1.0, 2002.