



저작자표시-동일조건변경허락 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.
- 이차적 저작물을 작성할 수 있습니다.
- 이 저작물을 영리 목적으로 이용할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



동일조건변경허락. 귀하가 이 저작물을 개작, 변형 또는 가공했을 경우에는, 이 저작물과 동일한 이용허락조건하에서만 배포할 수 있습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

석사학위논문

비전문가의 로봇 창작을 지원하는
스마트 기기 기반의
통합 개발 및 제어 환경 설계



2013년

HANSUNG
UNIVERSITY

한성대학교 대학원

정보통신공학과

정보통신공학전공

김진성

석사학위논문
지도교수 조혜경

비전문가의 로봇 창작을 지원하는
스마트 기기 기반의
통합 개발 및 제어 환경 설계

Designing a Smart Device Based
Integrated Development & Control Environment
for Supporting Non-Expert Robot Creators

2012년 12월 일

한성대학교 대학원

정보통신공학과

정보통신공학전공

김진성

석사학위논문
지도교수 조혜경

비전문가의 로봇 창작을 지원하는
스마트 기기 기반의
통합 개발 및 제어 환경 설계

Designing a Smart Device Based
Integrated Development & Control Environment
for Supporting Non-Expert Robot Creators

위 논문을 공학 석사학위 논문으로 제출함

2012년 12월 일

한성대학교 대학원

정보통신공학과

정보통신공학전공

김진성

김진성의 공학 석사학위논문을 인준함

2012년 12월 일



심사위원장 _____인

심 사 위 원 _____인

심 사 위 원 _____인

국 문 초 록

비전문가의 로봇 창작을 지원하는 스마트 기기 기반의 통합 개발 및 제어 환경 설계

한성대학교 대학원
정보통신공학과
정보통신공학전공
김 진 성

본 논문에서는 저가 로봇키트(robot kit)의 기능성 및 활용성과 연관 콘텐츠를 획기적으로 향상시키기 위한 방안으로, 스마트 기기 기반의 로봇용 통합 개발 및 제어 환경을 제안한다. 먼저 로봇의 기능을 향상시키기 위해 스마트 기기를 로봇 제어기로 확장시켜 터치스크린을 로봇의 개발 및 제어를 위한 인터페이스 장치로 활용하고 음성인식, TTS(text to speech) 등과 같은 스마트 기기의 고급기능을 로봇의 입출력 장치로 활용할 수 있도록 지원한다.

또한, 저가 로봇키트 사용자의 대다수가 프로그래밍을 배우지 않은 어린이지만 스마트 기기 사용에 익숙한 디지털 네이티브(digital natives)라는 점, 콘텐츠의 공유와 확산을 위해서 하드웨어 의존성을 최소화하여야 한다는 점을 고려하여 다음의 요구사항을 만족하는 개발환경을 설계한다. 첫째, 프로그래밍 및 디버깅, 실행의 과정을 매우 직관적이고 단순하게 한다. 둘째, 다양한 OS 기반의 스마트 기기를 지원한다. 셋째, 특정 로봇 제어기에 제한되지 않고 다양한 로봇 제품을 용이하게 수용할 수 있는 기반을 마련한다.

이를 위해, 본 논문에서는 먼저, 로봇의 제어구조에 관한 선행연구들과

비전문가들을 대상으로 한 기존의 로봇 프로그래밍 인터페이스들을 분석하여 초보자에게 적합한 프로그래밍 방법과 직관적인 사용자 인터페이스를 고안한다. 다음으로는 스마트 기기가 로봇의 실시간 모니터링 및 제어에 활용될 수 있도록 로봇 제어를 제로 클라이언트(zero-client)로 연동하는 제어 및 통신 프로토콜을 설계하는데, 특히, 현재 상품화되어 있는 다양한 로봇 제어를 수용할 수 있는 방안을 연구하여 제안한다. 마지막으로, 스마트 기기 상에서 로봇 프로그래밍 및 디버깅환경과 실시간 제어 환경이 안정적으로 동작될 수 있도록 전체 시스템의 아키텍처를 설계하고 구현한다.

제안된 통합 개발·제어 환경을 통해 저가 로봇키트들의 기능이 향상되고 쉽게 제어될 수 있음에 따라, 교육 현장에서 수학, 과학 및 기술 교육 목적에 충실하게 활용됨은 물론, 일반인들이 창의적인 아이디어를 표현하고 공유하는 창작의 도구로도 활용될 수 있을 것으로 기대된다.

【주요어】 교육용 로봇, 스마트 로봇, 스마트 기기, Reactive Robot System, Educational Programming Language, Visual Programming Language, Behavior-Based Robotics, Teleo-Reactive Sequences, Robot Integrated Development & Control Environment

목 차

제 1 장 서 론	1
제 1 절 연구의 배경	1
제 2 절 연구의 목적	3
1. 저가형 교구로봇의 기능 확장	3
2. 직관적인 로봇 프로그래밍 환경의 개발	4
제 3 절 문제의 정의	6
제 2 장 선행 연구	9
제 1 절 로봇 제어 구조(robot control architecture)	9
1. 계획 로봇(deliberative robots)	9
2. 반응 로봇 구조(reactive robot architecture)	11
3. 혼합 구조(hybrid architecture)	13
4. T-R 프로그램(Teleo-Reactive Programs)	15
5. 로봇 제어 구조의 제한과 시사점	17
제 2 절 비전문가용 로봇 프로그래밍 환경 분석	18
1. 레고 NXT-G (LEGO NXT-G)	18
2. 피코 블록(Pico Blocks)	20
3. 로보이드 스튜디오(Roboid Studio)	22
4. 로보플러스 (RoboPlus)	24
5. 비전문가용 로봇 프로그래밍 환경의 제한과 시사점	26
제 3 장 시스템 설계	27
제 1 절 전체 시스템 구성	27

제 2 절 소프트웨어 아키텍처	29
제 3 절 교육용 로봇 프로그래밍 언어의 설계	34
1. T-R 시퀀스와 직관적인 행동 편집	34
2. 절차적 표현을 위한 전역 변수	36
제 4 절 다양한 제품 수용을 위한 제어 프로토콜	38
제 5 절 가상로봇	41
 제 4 장 구현 및 결과	 43
제 1 절 구현 환경	43
제 2 절 사용자 인터페이스 구현	45
제 3 절 구현 결과	49
 제 5 장 결 론	 56
 【참고문헌】	 58
 ABSTRACT	 61

【 표 목 차 】

[표 1-1] 본 논문의 기대효과	8
[표 2-1] 비전문가용 로봇 프로그래밍 환경 비교	26
[표 4-1] 지원 가능한 운영체제 및 실행 장치	43
[표 4-2] 테스트 로봇의 지원 기능	44
[표 4-3] 가상센서, 조건, 행동의 편집에 사용하는 조작행위 정의	45
[표 4-4] 교육용 로봇의 본 논문 적용 전/후 기능비교	55



【 그림 목 차 】

<그림 2-1> 감지-계획-행동(SPA)의 구조(R. Brooks 1986)	9
<그림 2-2> 일반적인 계층적 계획의 예	10
<그림 2-3> 행동기반 로봇(BBR)의 구조 (R. Brooks, 1986)	11
<그림 2-4> AuRA 구조의 서브시스템 5개 (Robin R. Murphy, 2000)	13
<그림 2-5> 회로망에서 T-R 시퀀스 구현 (N. Nilsson, 1994)	16
<그림 2-6> NXT-G의 GUI(Graphic User Interface)	19
<그림 2-7> NXT-G의 센서블록(좌)과 모터블록(우)	19
<그림 2-8> Pico Blocks의 GUI	20
<그림 2-9> Pico Blocks의 연산 블록, 흐름 제어 블록	21
<그림 2-10> 로보이드 스튜디오의 타임라인 모션 에디터	22
<그림 2-11> 로보이드 스튜디오의 컴포우저(Composer)	23
<그림 2-12> 로보이드 스튜디오의 데이터-비주얼라이저와 에뮬레이터	23
<그림 2-13> 로보플러스의 테스트 GUI	24
<그림 2-14> 로보플러스의 매니저	25
<그림 2-15> 로보플러스의 모션에디터	25
<그림 3-1> 기존의 교육용 로봇 시스템	27
<그림 3-2> 본 논문의 교육용 로봇 시스템	28
<그림 3-3> 소프트웨어 아키텍처(Software Architecture)	30
<그림 3-4> 스마트 기기와 연동하여 확장된 로봇(Expanded Robot) ...	32
<그림 3-5> 기초조건과 복합조건의 데이터 구조	34
<그림 3-6> 단위행동(action)과 행동(behavior)의 데이터 구조	35
<그림 3-7> 교육용 로봇 제품의 블루투스 리모컨 프로토콜	38
<그림 3-8> 2바이트 유효 패킷 분할의 예	39
<그림 3-9> 다종의 교육용 로봇 플랫폼 수용	40
<그림 3-10> 가상 로봇과 실제 로봇의 비동기 갱신	42
<그림 4-1> 전체 GUI 화면 구성	46
<그림 4-2> 이진 값(binary data) 형식의 모니터	46

<그림 4-3> ADC 값 형식의 모니터링	46
<그림 4-4> 가상 센서	47
<그림 4-5> 단위 조건 생성	47
<그림 4-6> 단위 조건의 조합	48
<그림 4-7> 타임라인 동작 에디터	48
<그림 4-8> 본 논문에서 제시한 프로그래밍 과정	49
<그림 4-9> 어보이더 로봇(STEM-kit)	50
<그림 4-10> 본 시스템으로 구현한 어보이더 프로그램의 예	50
<그림 4-11> 라인트레이서(School-kit)	51
<그림 4-12> 본 시스템으로 구현한 라인트레이서 프로그램의 예	52
<그림 4-13> 순차적인 행동실행의 예	53
<그림 4-14> 순차적인 임무수행의 예	53



제 1 장 서 론

제 1 절 연구의 배경

최근 교구용 로봇(또는 교육용 로봇)을 이용한 로봇 활용 교육(또는 창작형 로봇학습)이 학습자의 창의력을 향상시킬 뿐만 아니라 과학, 기술, 공학, 수학능력에 긍정적인 영향을 미친다는 연구결과가 발표되고 있다(송정범 & 이태욱, 2011 ; 전윤주, 2010 ; 박응식 & 문성환, 2009 ; 김경현 외 7인, 2009). 교구용 로봇이란 전문적 로봇기술 교육목적에 활용되는 로봇을 말하며 일반교과 및 창의력 교육이 주된 목적인 로봇을 뜻한다(김경현 외 7인, 2009). 교구로봇활용수업의 효과분석 연구 보고서에 따르면 최근 약 2000여 개의 학교에서 방과 후 및 특기적성 활동 시간을 활용한 로봇 교육이 실시되고 있으며, 지식경제부에서는 방과 후 학교 수업용 로봇을 전국 100여개 학교에 지원하고 체계적인 로봇교육 방안을 마련하여한다고 밝힌 바 있어 교구용 로봇의 보급이 가속화 될 것으로 예상했다(김경현 외 7인, 2009). 어린이의 성장 과정 중에 로봇과 같은 기술을 자연스럽게 노출시키므로 창작 활동에 대한 흥미와 보다 높은 집중을 유도할 수 있다(한정혜, 2011 ; 이강봉, 2011). 창작형 로봇학습이란 학습자가 상상한 로봇을 직접 제작하는 활동 외에 이를 동작하기 위한 로봇 프로그래밍 학습 과정을 포함한다(한정혜, 2011 ; 이강봉, 2011). 이는 일반적인 프로그래밍 교육(PC를 이용한 프로그래밍)과 달리 학습자의 의도가 실제 공간상에 존재하는 로봇에 반영되기 때문에 프로그래밍을 학습하는데 있어서 보다 높은 흥미와 의욕을 유도할 수 있으며 다양한 사고를 하도록 유도할 수 있는 장점이 있다(박응식 & 문성환, 2009 ; 김경현 외 7인, 2009 ; 한정혜, 2011 ; 이강봉, 2011).

그러나 이러한 많은 장점에도 로봇이 실제 교육에 활용되기에는 다음과 같은 어려움이 있다. 첫 번째, 교육용 로봇의 시장구조상 가격 경쟁력 확보를 위해 하드웨어가 극히 제한적이다. 학습자는 정밀하지 않은 하드웨어

를 사용하여 발생하는 오차를 감안한 프로그램을 작성해야하는 어려움이 있을 뿐만 아니라 디스플레이장치가 미흡하여 프로그래밍에 필요한 정보를 얻기도 쉽지 않고 표현에도 한계가 있다. 아직 충분한 개념이 서지 않은 어린 학생들이 사용하게 되므로 오차와 한계로 인해 과학기술에 대한 선입견이 생기지 않도록 주의해야한다.

두 번째, 교육용 로봇은 제품마다 각기 다른 프로그래밍 방식과 프로그래밍 환경에서 동작하기 때문에 교수자가 만든 교수안이 재사용되기 어렵다(박동희, 2009). 최근에 정부 사업을 통하여 로봇 활용 교육의 효과적인 확산을 위한 교수안이 도출되고 있다. 실제로 로봇 소양 교육과정 연구 보고서(조혜경 외 9인, 2011)에 따르면 지식경제부가 지원하는 2010년 시범사업의 광운대학교 컨소시엄이 총 100차시 분량의 교수안을 도출하였고 ‘미니멀리즘 기반 UCR(user create robot) 기술개발’ 과제에서 한성대학교와 한국교육학술정보원이 총 200차시 분량의 교수안을 도출하였다. 교육용 로봇 제품의 프로그래밍 환경이 호환성을 보장한다면 도출된 교수안의 활용도가 높아질 뿐만 아니라 신제품 로봇에도 교수안을 재활용할 수 있을 것이다.

세 번째, 현재 제품화된 교육용 로봇 프로그래밍 환경은 많은 수의 명령어와 직관적이지 않은 인터페이스로 구현되어있어 일반 사용자가 짧은 시간 내에 사용법을 습득하기 어렵다. 창작형 로봇학습을 수행하려면 로봇 프로그래밍 교육이 선행되어야하지만 실제 교육현장에서 프로그래밍 교육을 다룰 시간의 확보가 힘들다(조혜경 외 9인, 2011). 국내 교육용 로봇 프로그래밍 환경의 대부분이 컴퓨터언어(c-language)의 문법체계를 따르기 때문에 프로그래밍에 대한 기초지식이 없는 학습자가 사용하기 어렵다. 또한 로봇 프로그래밍 환경은 일반적으로 널리 사용되지 않은 프로그램이기 때문에 사용자 경험(user experience)이 통용되어있지 않아 직관적인 사용자 인터페이스(user interface)의 설계가 필요하다.

제 2 절 연구의 목적

본 논문의 목적은 저가형 교구로봇의 기능을 개선하기 위한 방안을 연구하고 프로그래밍에 대한 전문지식이 없는 사용자도 쉽게 사용할 수 있고 다양한 교육용 로봇 하드웨어를 수용할 수 있는 프로그래밍 환경을 개발하여 교육용 로봇이 과학, 공학, 기술, 수학 교육에서 활용될 수 있도록 지원하는데 있다. 로봇을 프로그래밍 기술을 익히는 매체로 사용하는 것보다는 프로그래밍 기술에 익숙하지 않은 초보자가 로봇을 통해 자신의 아이디어를 표현할 수 있도록 지원하는데 초점을 둔다.

1. 저가형 교구로봇의 기능 확장

로봇은 일반적으로 임무(task)를 완료하기 위해 주위 환경에 대한 변화를 지각하고 상황을 판단하여 임무 수행에 유리한 방향으로 동작을 수행하는데 이는 인식(perception), 인지(cognition), 행동(action)으로 이루어진 로봇 기술에 기반을 둔다. 이러한 기술은 일반적인 교육용 로봇에도 적용되어 있지만 교육용 로봇의 하드웨어는 성능이 떨어지고 다양하지 못하기 때문에 원하는 임무를 제대로 수행하기 어려우며 표현에 제약이 따른다. 교육용 로봇은 입/출력 포트의 개수가 제한되어있으며 입력 센서의 경우 해상도가 낮아 정밀하지 못하다. 특히 교육용 로봇의 센서로 가장 많이 사용되는 적외선센서(infrared ray sensor)는 소자의 특성상 주위 환경에 따라 측정값의 차이를 보이는데 이는 센서데이터의 모니터링 장치가 없는 교육용 로봇으로 프로그래밍하는 사용자로 하여금 혼란을 갖게 한다. 사용자가 자신이 작성한 프로그램논리의 옳고 그름을 판단할 수 있도록 가장 먼저 센서데이터의 신뢰성을 보장하는 방안이 필요하며 그 다음으로 사용자 스스로가 자신이 저작한 프로그램논리를 검증할 수 있도록 디버깅 장치를 제공해주어야 한다.

일반적인 프로그램의 디버깅은 프로그램의 실행 자취를 한 단계씩 추적

하며 그 결과를 확인하는 방식으로 이루어지며 주로 구문오류나 논리오류를 해결하기 위한 과정이다. 특히 프로그래밍 언어를 이제 막 배우기 시작하는 초보자는 구문오류로 인한 디버깅에 많은 시간을 소비하게 된다. 그래서 대부분의 초보자용 로봇 프로그래밍 환경은 간단한 규칙 기반의 시스템(rule-based system)으로 이루어져있어 많은 양의 구문오류를 미연에 방지한다. 규칙 기반의 시스템에서 로봇 프로그램은 실행 결과가 로봇의 출력장치로 표현되고 이를 직접 눈으로 확인하여 예측이 가능하므로 디버깅 기능은 눈에 보이지 않는 입력장치나 변수를 확인하는 기능에 초점을 맞춰야한다. 또한 디버깅 방법이 프로그램의 실행 자취를 한 단계씩 추적하는 것이 아니라 실행과정을 그대로 드러내어 직관적으로 확인할 수 있도록 해야 한다.

본 논문에서는 프로그래밍 활동을 활성화하기 위해 저가형 센서의 모니터링 기능을 개발하고 학습자 스스로가 프로그래밍에 필요한 정보를 습득하고 프로그래밍 활동을 진행할 수 있도록 지원한다. 또한 사용자가 작성한 프로그램에서 발생한 논리오류를 스스로 해결할 수 있도록 입력장치나 변수를 실시간으로 확인할 수 있는 디버깅 기능을 개발한다. 나아가 교육용 로봇의 부족한 입/출력장치를 보완할 수 있는 방안을 제시하여 표현력을 높인다.

2. 직관적인 로봇 프로그래밍 환경의 개발

일반적인 프로그래밍 명령어는 사용자에게 친숙하지 않을뿐더러 문자기반의 명령어를 사용하기 때문에 직관적인 인터페이스를 표현하기 적합하지 않다. 반면에 비주얼 프로그래밍 언어(VPL : visual programming language)는 일반적인 프로그래밍 언어에서 쓰이는 문자기반의 명령어를 대신하여 시각적 요소로 이루어진 명령어를 사용하여 프로그램을 만들 수 있다. 비주얼 프로그래밍 언어는 2차원 이상의 공간상에 아이콘(icon)이나 다이어그램(diagram), 상자, 화살표 등으로 이루어진 시각적 요소를 결합

하여 프로그램 로직을 표현한다(The free dictionary, n.d.; Wikipedia, 2012). 비주얼 프로그래밍 언어의 특성상 초보자도 아이콘기반의 프로그래밍 문법만 익히면 간단한 프로그램을 작성할 수 있다.

하지만 아무리 쉬운 프로그래밍 방법을 제안하여도 많은 양의 명령어를 사용한다면 사용자의 학습흥미를 방해할 여지가 있기 때문에 교육용 로봇을 제어하는 프로그래밍 언어는 초보자가 학습하기 쉽도록 최소한의 명령어만 사용하는 전략(mini-language)을 사용하는 것이 적합하다(박동희, 2009).

로봇의 프로그램은 지속적으로 변화하는 실제 환경 속에서 동작하기 때문에 예외 상황이 빈번하게 발생되는데 이를 컴퓨터에서 흔히 사용해오던 관습적인 프로그래밍 방식으로 처리하면 모든 예외상황에 대한 예측과 대응을 수많은 인터럽트를 통해 구현해야하므로 매우 복잡해진다(Nils J. Nilsson, 1994). 때문에 로봇의 예외상황을 단순하게 처리할 수 있는 프로그래밍 방식이 필요하며 이를 고려한 비주얼 프로그래밍 체계의 교육용 로봇 프로그래밍 환경을 개발하여서 프로그래밍 명령어 체계나 문법에 대한 선행학습이 없어도 짧은 시간 내에 습득하여 사용할 수 있도록 지원한다.

제 3 절 문제의 정의

본 논문은 앞서 언급한대로 교육용 로봇의 미흡한 기능을 보완하고 직관적인 로봇 프로그래밍 방법을 제안하여 교육용 로봇의 활용성을 높일 수 있는 방안을 제시하는데 있다. 이러한 목적을 해결하기 위해서는 다음과 같은 연구문제가 고려되어야한다.

첫 번째, 어떻게 적은 비용으로 교육용 로봇의 기능을 개선할 것인가?

두 번째, 초보자가 사용할 수 있는 프로그래밍 방식을 어떻게 구현할 것인가?

세 번째, 초보자가 쉽게 디버깅할 수 있게 하려면 어떻게 정보를 제공하는 것이 효과적인가?

네 번째, 어떤 방법으로 다수의 교육용 로봇 제품을 수용할 것인가?

위의 문제를 해결하기 위하여 이미 대중화된 스마트폰(smart-phone)이나 태블릿PC(tablet-pc)와 교육용 로봇을 결합한다. 스마트폰이나 태블릿 PC는 큰 디스플레이 기능을 비롯하여 다양한 센서 장치와 블루투스(bluetooth) 통신, 기타 고급기능이 강점인데 이는 교육용 로봇이 가지는 기능적 단점을 해결할 수 있다. 대부분의 교육용 로봇은 리모트 컨트롤(remote control)을 지원하기 위해서 블루투스 통신 인터페이스를 갖추고 있는데 이를 활용한다면 추가적인 하드웨어 개발을 하지 않고 스마트 기기와 통신할 수 있다. 기존에 로봇제어기가 담당하던 로봇의 두뇌역할을 스마트 기기가 대신하고 로봇제어기는 I/O 보드(input/output board)의 역할을 맡게 하여 펌웨어(firmware)를 변경하지 않고 다양한 로봇 플랫폼을 수용할 기반을 마련할 수 있다. 또한 스마트 기기 덕에 강화된 디스플레이 기능을 활용하여 PC가 없는 환경에서도 로봇 프로그래밍이 가능해진다.

본 논문의 연구문제를 해결하기 위하여 다음과 같이 연구 목표를 세운다.

첫 번째, 사용자가 가진 스마트 기기를 로봇의 제어기로 확장하여 저가 제어기에 없었던 고급 기능 지원한다. 사용자가 프로그래밍에 필요한 정보

를 스마트 기기의 디스플레이장치를 통해 얻을 수 있도록 지원하고 스마트 기기의 고급센서와 가상센서를 로봇 센서와 동일하게 사용가능하도록 하여 저가의 교육용 로봇에 부족한 입력 장치를 지원한다. 또한 사운드파일재생, TTS 등 스마트 기기의 사운드 장치를 연계한 로봇 콘텐츠 제작이 가능하도록 기능을 제공한다.

두 번째, 직관적인 프로그래밍 인터페이스를 제공하여 기술소양이 낮은 사용자를 수용한다. “A조건이 만족되면 B행동을 한다.”는 단순한 반응형 행동 기반의 제어 구조로 구성하여 프로그래밍 지식이 없는 사용자도 직관적으로 논리설계가 가능하도록 지원한다. 로봇을 동작시키기 위한 명령어를 익히지 않고 논리연산만으로 조건설계가 가능하도록 조건생성 인터페이스를 제공하며 타임라인 기반의 행동 편집기능을 통해 LED나 모터 등 다수 출력장치들을 쉽게 동기화하는 시각적 인터페이스를 제공한다.

세 번째, 항시 로봇의 내부를 모니터할 수 있는 기능을 제공하여 디버깅이 어려운 로봇 프로그래밍에서 사용자 스스로가 문제를 찾을 수 있도록 지원한다. 로봇의 모니터는 눈에 보이지 않는 센서 변화를 시각화한 센서 모니터와 사용자가 정의한 조건이 올바르게 동작하는지 확인할 수 있는 조건모니터, 하드웨어 이상에 따른 오작동을 확인할 수 있는 장치모니터로 구분하여 문제의 원인을 쉽게 파악할 수 있도록 지원한다. 또한 편집하거나 실행할 때 모니터를 확인할 수 있도록 하여서 언제든지 디버깅이 가능하도록 개발한다.

네 번째, 로봇 제어기를 I/O board로 사용하고 가상의 로봇 모델을 정의하여 기존에 상용화된 제품을 수용한다. 상용화된 제품들이 사용하는 리모컨용 블루투스 모듈로도 제어가 가능한 통신 프로토콜을 구현하여 추가로 하드웨어를 구입하지 않아도 되게 한다. 그리고 로봇 제품마다 다른 사양을 반영할 수 있는 로봇 사양 명세서를 지정하고 사용자 임의로 조립한 로봇의 정보를 얻을 수 있는 사용자 조립 명세서를 지정한다. 이 두 가지 명세서를 기반으로 가상의 로봇모델을 정의하고 실제 로봇과 동기화시 이를 거쳐서 진행하여 사양에 따른 성능을 최적화한다. 위와 같은 연구 목표를 통해 [표 1-1]과 같은 결과를 기대할 수 있다.

[표 1-1] 본 논문의 기대효과

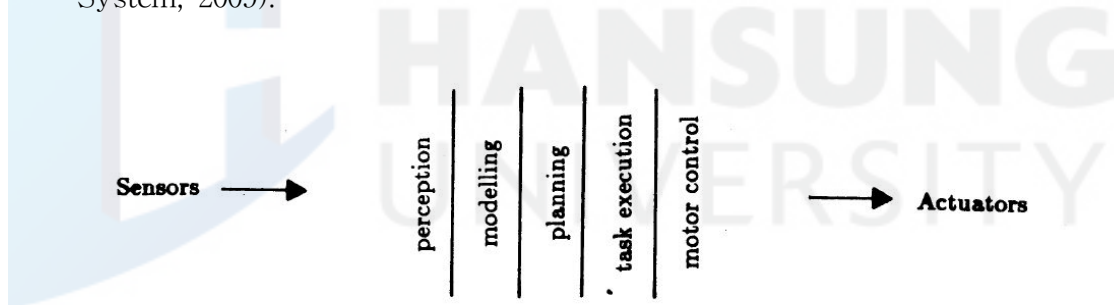
분류	As is	To be
센서 값 확인	센서 모니터용 프로그램을 작성하거나 매니저 모드를 실행한다.	언제든지 로봇의 센서 값을 확인한다.
프로그램 작성	명령어나 문법체계를 익히면 프로그램을 작성할 수 있다.	조건-행동 규칙을 익히면 프로그램 작성이 가능하다.
프로그램 수정	수정된 프로그램을 새로 이식해야한다.	수정된 프로그램을 즉시 실행해볼 수 있다.
오작동 (환경변화)	센서 값에 대한 디버깅을 통해서 문제를 찾아야한다.	환경변화에 따른 센서 값 변화가 모니터에 그대로 드러난다.
오작동 (고장)	프로그램 디버깅과 장치 점검을 모두 해봐야 알 수 있다.	로봇 모니터를 통해서 문제를 찾을 수 있다.
표현장치 가짓수	제품화된 장치만 사용할 수 있다.	스마트 기기의 센서와 가상센서 등을 추가로 사용할 수 있다.
프로그램 작성시간	작성해야할 프로그램의 양이 많아 소요시간이 길다.	작성해야할 프로그램의 양이 적어 소요시간이 짧다.
사용법 습득시간	명령어나 문법체계를 익히는 시간이 필요하다.	조금만 사용해보면 짧은 시간 내에 습득한다.

제 2 장 선행 연구

제 1 절 로봇 제어 구조(robot control architecture)

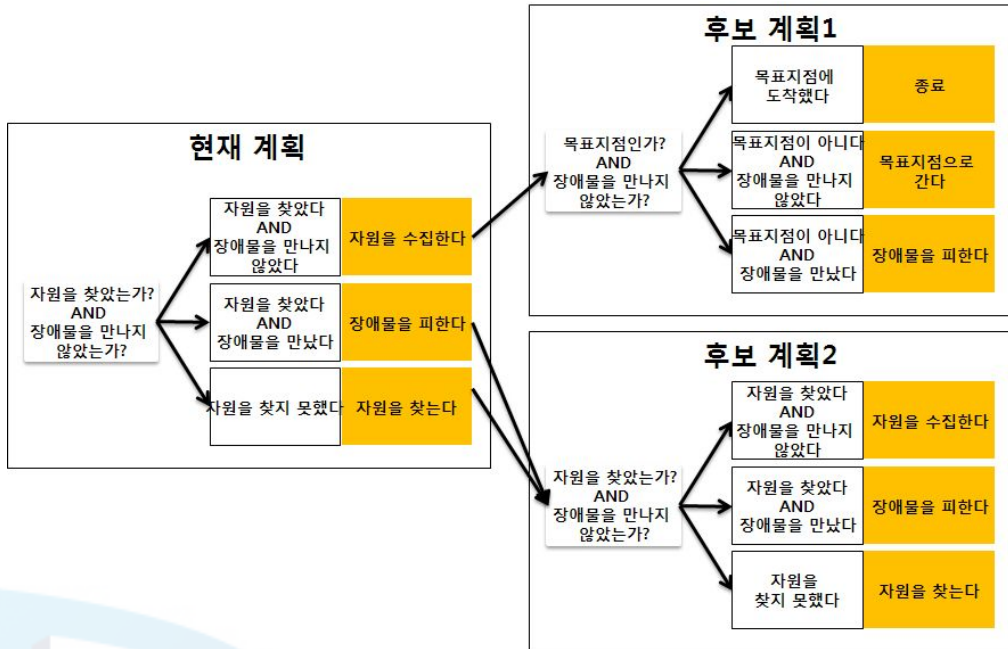
1. 계획 로봇(deliberative robots)

지능형 로봇은 지속적으로 변화하고 예측하기 힘든 환경 속에서 동작한다. 이러한 이유로 로봇이 동작하고 있는 동안 계속해서 주변 환경을 센싱(sensing)하여 이를 로봇의 동작에 반영하는 이론들이 등장했다. Shakey가 1960년대에 제안한 감지-계획-행동(SPA : Sense Plan Action)은 <그림 2-1>과 같이 지속적으로 감지-계획-행동을 순환하여 임무의 달성을 위해 행동계획을 수정하는 방법으로 계획 로봇이다(Robot Operating System, 2009).



<그림 2-1> 감지-계획-행동(SPA)의 구조(R. Brooks 1986)

이렇게 센서로부터 얻은 자료(data)를 바탕으로 새로운 행동 계획을 세우기 때문에 대체적으로 계층적 구조(hierarchical architecture)를 갖는다. 어떤 행동을 수행하면 그 결과에 따라 새로운 계획을 수립하기 때문에 <그림 2-2>처럼 새로운 분기점(branch)이 등장한다. 이는 컴퓨터 공학의 계층적 과업 네트워크(HTN : Hierarchical Task Network)와 유사한 방법으로 빈번하게 변화되는 모의 전투 환경에서 가상군의 자율적 행위를 위한 자동계획 기법에도 사용되었다(최대희 외 4인, 2011).

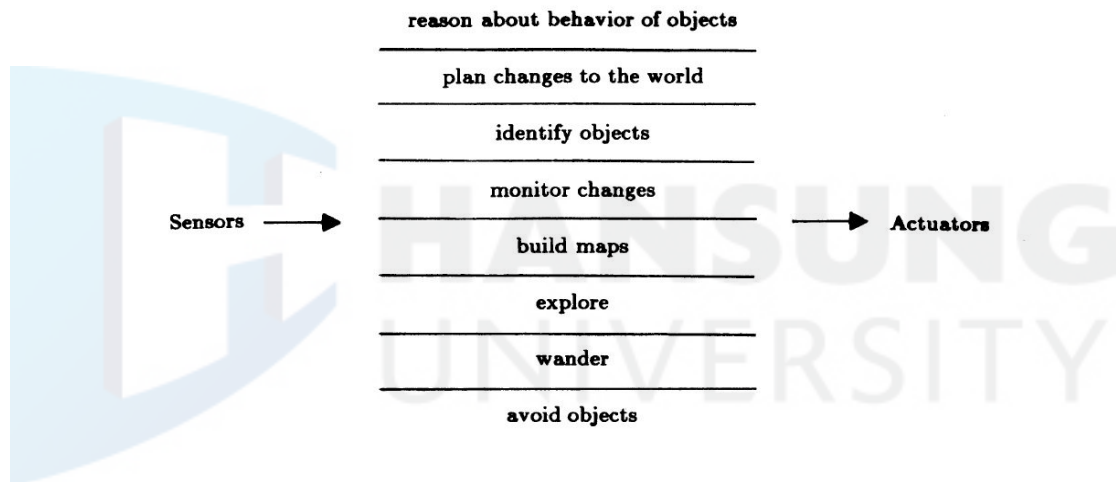


<그림 2-2> 일반적인 계층적 계획의 예

이러한 계층적 계획 구조는 실행순서(procedure)가 요구되는 행동을 표현하기에 효과적이며 복잡도가 높은 행동을 구현할 수 있는 장점이 있지만 예외상황이 발생할 때마다의 새로운 계획을 수립해야하고 분기점이 늘어나기 때문에 충분한 자원을 요구한다는 단점을 지닌다.

2. 반응 로봇 구조(reactive robot architecture)

앞서 언급한 계층적 구조의 단점을 해결하여 보다 환경변화에 대응하기 유리한 반응 로봇 구조가 등장하였다. Brooks가 1986년도에 제안한 포섭 구조(subsumption architecture)나 1991년도에 제안한 행동기반의 로봇공학(BBR : behavior-based robotics)은 지속적으로 환경을 감지하고 이를 즉각적으로 행동에 반영하는 단순한 구조로 계획과정이 생략된 반응 로봇 구조 이다(R. Brooks, 1986 ; R. Brooks, 1991).



<그림 2-3> 행동기반 로봇(BBR)의 구조 (R. Brooks, 1986)

반응 로봇 구조는 지능이 낮은 곤충이 환경에서 즉각적으로 반응하는 것과 같이 감지에 대한 반응을 계획과정을 거치지 않고 수행한다. <그림 2-3>과 같이 행동기반의 로봇공학에서는 계획(plan)이 생략된 감지(sense)와 행위(action)의 결합을 행동(behavior)으로 규정하며 로봇의 임무를 이러한 행동단위로 분해하여 병렬로 동작한다. 포섭구조는 행동기반로봇에 속하는데 이동과 같은 기본적인 행동을 가장 낮은 계층에 배치하고 기본적인 행동이 동작하는 중에 환경과의 상호작용으로 감지된 신호를 상위 계층에서 감시(monitoring)하여 임무수행에 유리한 행동이 유발되도록 설

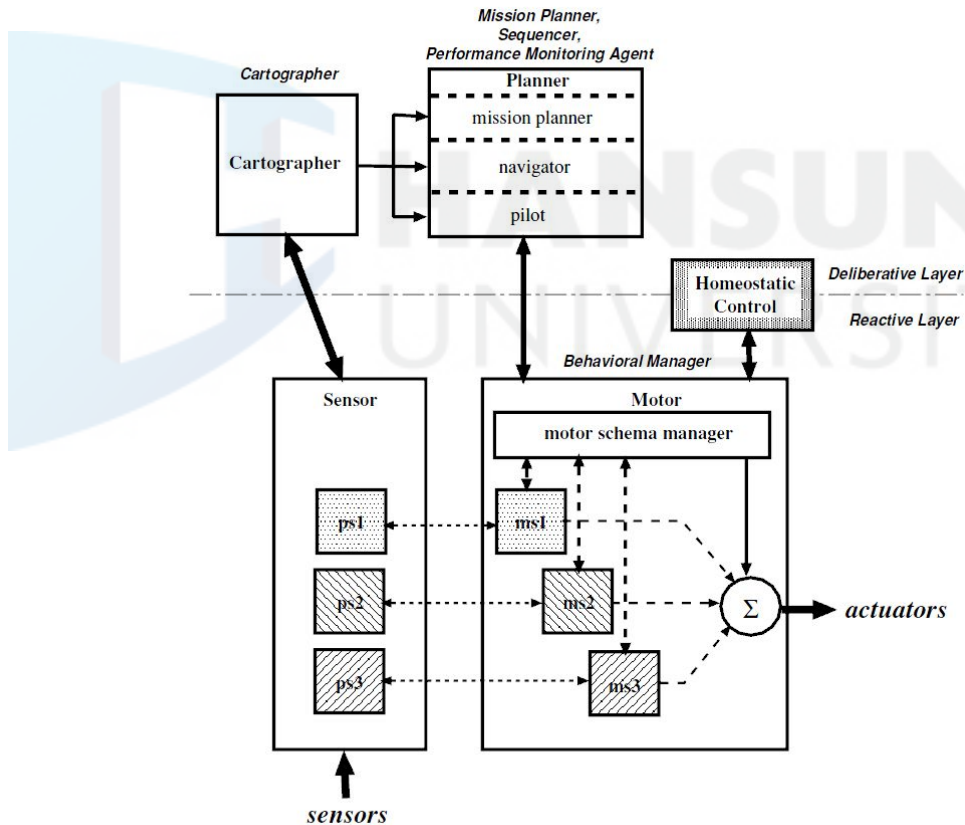
계한다. 또한 설계에 따라 상위계층의 출력신호가 하위계층의 감지신호 유입을 억압(suppression)하거나 하위계층에서 나오는 출력신호를 억제(inhibition)하여 임무수행에 불필요한 행동을 차단한다(R. Brooks, 1986 ; R. Brooks, 1991).

포섭 구조는 계층 간에 독립성을 보장하므로 계층을 쌓으므로 변형이 가능할 뿐만 아니라 반응성과 동시성이 보장되고 계획과정이 생략되어 정보처리의 병목 현상이 일어나지 않는 장점을 갖지만 복잡한 임무를 수행해야할 경우 포섭관계를 설계하기 어렵다는 한계점을 보인다.



3. 혼합 구조(hybrid architecture)

최근에는 계획로봇과 반응로봇의 단점을 해결하고자 계획(deliberation)과 반응(reaction)을 병렬적으로 처리하는 혼합 구조(hybrid architecture)에 대한 연구가 활발히 이루어지고 있다. 대표적으로 Ron Arkin의 AuRA(Autonomous Robot Architecture)가 있다. AuRA는 가장 오래된 혼합 구조로 스키마 이론(schema theory)에 근거한 5개의 서브시스템(subsystem)으로 구성된 객체 지향 클래스를 기반으로 한다. 다음 <그림 2-4>는 AuRA 구조의 5개의 서브시스템을 나타낸다.



<그림 2-4> AuRA 구조의 서브시스템 5개 (Robin R. Murphy, 2000)

서브시스템 중 계획자(planner) 서브시스템과 지도제작자(cartographer) 서브시스템은 전반적인 로봇의 계획을 담당한다. 계획자와 지도 제작자는 서로 상호작용하며 임무수행에 필요한 길을 따라가거나 잃어버린 길을 찾는다. 지도 계획자는 계획자에게 선형을 바탕으로 제작한 지도(map)를 제공하고 계획자는 이를 바탕으로 현재 임무의 일부를 완수하기 위한 행동 리스트를 작성하여 모터(motor) 서브시스템에 전송한다. 센서 서브시스템과 모터 서브시스템은 로봇의 반응 임무를 담당하는데, 이는 기초적인 지각 스키마(perceptual schema)와 모터 스키마(motor schema)의 유한상태 기계(FSM : finite state machine)로 조직화되며 모터 스키마 관리자를 통해 서로의 정보를 공유할 수 있다. 이 두 서브시스템은 일반적인 상황에서는 계획자에 의해 작성된 행동 리스트를 토대로 동작하지만 로봇의 고장이 발생할 수 있는 예외상황에는 항상성 제어(homeostatic control) 서브시스템에 의해 수정된 행동을 수행한다. 항상성 제어는 계획자 서브시스템에 별도의 모듈을 추가하는 것이 아닌 내부적 필요로 인한 잠재적인 욕구에 의해 이루어진다. 이는 포텐셜 필드(potential fields)로 표현된 행동의 출력과 잠재적 욕구에 의한 이득의 벡터 곱으로 간단하게 계산된다(Robin R. Murphy, 2000). AuRA는 독립적인 구조로 모듈화가 용이하고 변형이 쉬워 다양한 이론을 적용하기에 적합하다. 하지만 구현 비용이나 구현 과정이 복잡하다는 단점을 갖는다.

4. T-R 프로그램(Teleo-Reactive Programs)

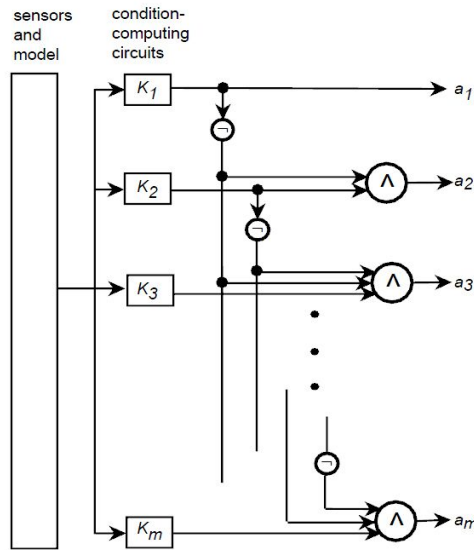
앞서 언급한 구조들과 다른 관점으로 회로 이론을 바탕으로 Nillsson이 제안한 T-R 프로그램을 분석한다(N. Nilsson, 1994). T-R 프로그램에서 T-R 시퀀스(Teleo-Reactive Sequences)는 임무의 중요도에 따라 상위에 배치하는 방법을 말한다. T-R 시퀀스는 겉으로 보기에 관습적인 프로그래밍(conventional programming)에서 흔히 사용하는 생성 규칙(production rule) 같이 $K_1 \rightarrow a_1$ 인 평평한 조건-행동 규칙(condition-action rules)을 기반으로 하지만 약간의 차이가 있다.

$$\begin{array}{lcl} K_1 & \rightarrow & a_1 \\ K_2 & \rightarrow & a_2 \\ & \dots & \\ K_i & \rightarrow & a_i \\ & \dots & \\ K_m & \rightarrow & a_m \end{array}$$

T-R 시퀀스는 마치 스위치가 눌려지면 동작하는 아날로그 전자회로처럼 스위치가 꺼지면 동작도 멈춘다. 즉 조건이 행동의 수행을 시작하는 역할을 하면서 동시에 행동의 수행을 유지하는 역할을 한다. 일반적인 프로그램에서는 함수의 시작조건이 만족하면 그 함수가 리턴(return)될 때까지 실행되는 반면에 T-R 프로그램에서는 함수를 호출한 시작조건이 유지되지 않으면 함수실행이 중단된다. 이는 예외상황에 대한 예측이 어려운 실제 환경에서 현재 실행중인 함수를 중단할 상황을 결정하기에 적합하다.

T-R 시퀀스에서 조건들은 관습적인 프로그래밍에서 우선순위(priority)와 흡사한 개념의 순서를 갖는데 상위레벨 시퀀스(high-level sequence)의 조건이 만족하면 하위레벨 시퀀스(low-level sequence)의 조건들은 무시된다. 실제로 하위레벨 시퀀스의 조건이 만족되기 위해서는 상위레벨 시퀀스가 만족하지 않는다는 전제가 있어야하므로 우선순위를 갖는 프로그램처럼 동작한다. <그림 2-5>는 T-R 시퀀스를 전기회로망으로 표현한 것인데 사용자가 정의한 조건 K 가 n 번째 시퀀스를 갖는다면 K_n 으로 표시할

수 있으며 이와 대응하는 행동을 a_n 으로 표시할 수 있다.



<그림 2-5> 회로망에서 T-R 시퀀스 구현

(N. Nilsson, 1994)

따라서 사용자는 조건과 행동함수 쌍을 궁극적인 임무 순으로 나열하면 된다. 자원을 수집하는 예를 들어보면 다음과 같다.

$see_resource \rightarrow collect_resource$

$on_trail \rightarrow follow_trail$

$T \rightarrow wander$

최상위레벨의 조건 $see_resource$ 가 만족하면 $collect_resource$ 를 수행하고 하위레벨의 조건-행동 쌍은 무시되며 $see_resource$ 가 만족하지 않으면 on_trail 조건이 유효하게 된다. $see_resource \rightarrow collect_resource$ 규칙에서 예기치 않은 상황이 발생하면 $on_trail \rightarrow follow_trail$ 규칙으로 대처할 수 있다. 이는 매우 직관적이며 간단한 계획만으로도 단순한 임무를 수행할 수 있지만 여러 로봇의 협동이 필요한 복잡한 임무를 수행하기에는 부적합하다(S. Coffey & K. L. Clark, 2006).

5. 로봇 제어 구조의 제한과 시사점

앞서 살펴본 계획 구조는 실행순서가 요구되는 행동을 표현하기에 직관적이지만 계획과정을 처리하기위한 충분한 자원이 필요하고 사용자가 예외상황에 대해 모두 고려해야하므로 설계할 양이 많아질 수 있어 비전문가를 대상으로 하는 교구용 로봇에 적합하지 않다. 반응 로봇 구조는 행동계층 간에 독립성을 보장하여 반응성과 동시성이 보장되어 계획 구조에 비해 예외상황처리가 효과적이지만 임무수행을 위해 행동 간의 포섭관계를 설계하기가 어렵다는 한계점을 보인다. 혼합 구조는 앞서 언급한 계획 구조와 반응 로봇 구조의 단점을 모두 해결하였지만 사용자가 직접 구현하기에 복잡하며 많은 양의 자원이 필요하기 때문에 저가형 교구 로봇의 성격에 맞지 않다. T-R 시퀀스는 앞선 구조에 비해 매우 직관적이며 간단한 계획만으로도 단순한 임무를 수행할 수 있을 뿐만 아니라 예외상황처리에도 효과적이다. 하지만 다수 임무의 절차적인 수행이 필요한 복잡한 상황을 구현할 수 없다.

본 논문은 비전문가가 짧은 시간 내에 습득하여 사용할 수 있는 프로그래밍 환경을 개발하기 위해 복잡한 임무수행에 유리한 구조보다는 단순한 임무를 직관적으로 설계하기 유리한 T-R 시퀀스를 기반으로 프로그래밍 환경을 설계한다. T-R 시퀀스를 기반으로 절차적인 표현의 일부를 수용할 수 있는 방안을 마련한다면 비전문가도 예외상황대처를 보다 쉽게 할 수 있는 환경을 개발할 수 있을 것으로 판단된다.

제 2 절 비전문가용 로봇 프로그래밍 환경 분석

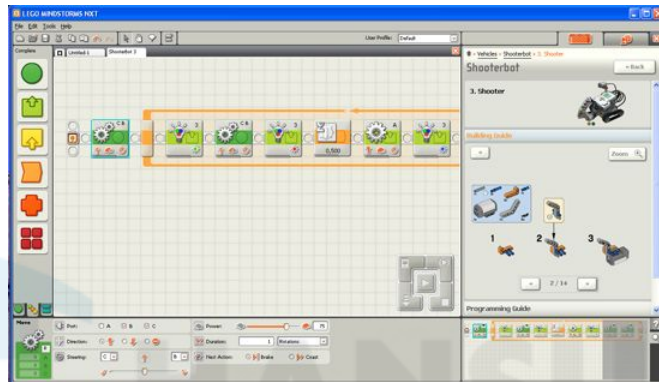
본 절에서는 본 논문의 주제인 교육용 로봇의 고급 기능지원과 직관적인 프로그래밍 환경을 개발하기 위해 기존에 비전문가를 대상으로 개발되어 상용화된 교육용 로봇 프로그래밍 환경을 살펴본다. 교육용 로봇의 프로그래밍이 가능한 환경으로는 멀티미디어 프로그래밍 환경에 로봇 프로그래밍 기능이 추가되어 로봇과 멀티미디어를 융합한 형태의 콘텐츠를 저작할 수 있는 환경과 로봇만으로 콘텐츠를 저작할 수 있는 로봇전용 프로그래밍 환경이 존재한다. 전자의 경우에는 프로그래밍에 대한 포괄적인 활동을 모두 다루기 때문에 명령어의 개수가 많고 복잡하며 후자의 경우에는 로봇 제어 활동만 다루기 때문에 상대적으로 명령어의 개수가 적고 단순하다. 본 논문에서는 프로그래밍을 처음 접하는 어린 학생을 대상으로 한 프로그래밍 환경을 개발하는 것이 목표이기 때문에 로봇과 멀티미디어를 융합한 형태의 콘텐츠를 저작할 수 있는 환경은 다루지 않고 보다 짧은 시간 내에 습득할 수 있는 로봇 전용 콘텐츠를 저작할 수 있는 프로그래밍 환경을 조사하여 장단점을 분석한다.

1. 레고 NXT-G (LEGO NXT-G)

NXT-G는 제어기와 입출력 장치를 갖춘 조립식 블록완구 “마인드 스톰 NXT(Mind storm NXT)”의 프로그래밍 환경이다. NXT-G는 사용자 스스로 환경에 적응할 수 있도록 다양한 예제, 사용설명서, 비디오 설명을 제공하며 레고(LEGO) 블록과 호환성을 갖춘 로봇을 사용하기 때문에 사용층이 두터워 정보를 쉽게 얻을 수 있다(STEM centric, 2012 ; NXT Programs, 2007). 또한 비전문가도 사용할 수 있도록 <그림 2-6>과 같이 아이콘 기반의 블록형 GUI(Graphic User Interface)를 제공하며 순서도를 작성하는 형태로 프로그래밍을 진행할 수 있다(홍기천, 2009).

NXT-G에서 제공하는 장치제어용 명령어 블록은 모터블록, 소리블록,

초음파센서블록, 빛센서블록 등이 있으며 그 형태는 <그림 2-7>과 같다. 사용된 장치제어용 명령어 블록에 선택적으로 기능을 덧붙일 수 있으며 이때마다 블록위에 작은 아이콘이 추가된다. 텍스트로 표현된 명령어를 사용하지 않고 블록의 모양과 추가된 아이콘만으로 사용자가 작성한 프로그램의 해석이 가능하다. 하지만 여러 가지 정보를 표현하기 위해 조밀한 아이콘을 다수 사용하기 때문에 복잡한 프로그램을 작성할수록 가독성이 떨어진다.



<그림 2-6> NXT-G의 GUI(Graphic User Interface)

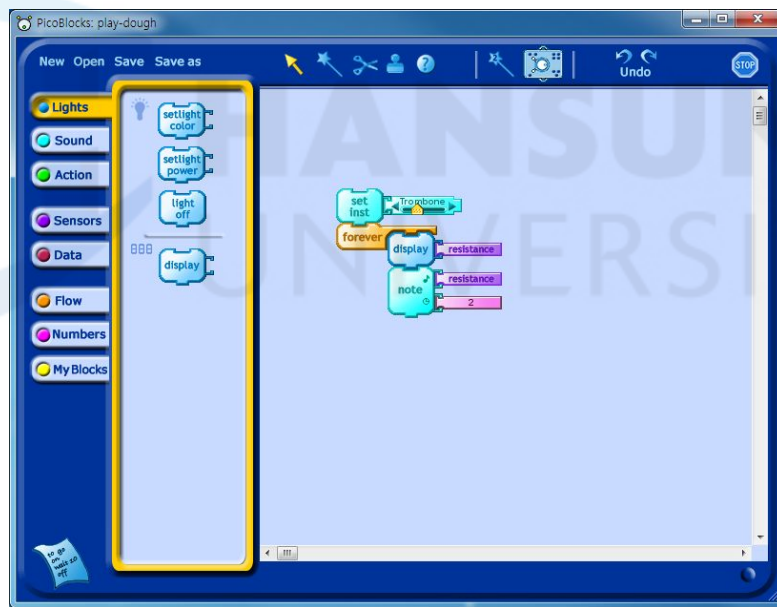


<그림 2-7> NXT-G의 센서블록(좌)과 모터블록(우)

NXT-G는 장치를 제어하기 위한 명령어 블록 외에 간단한 분기(branch)를 만들 수 있는 스위치블록과 다양한 종류의 반복문을 제공한다. 기본적인 문법은 컴퓨터언어와 흡사하여 프로그램의 시작과 끝이 존재하는 절차 프로그램(procedure program)의 형식을 떠나 초보자가 하기 쉽도록 구문 에러를 방지하기 위해 레고블록을 끼워 맞추는 디자인을 사용한다. 로봇 실행 중 디버깅할 수 있는 환경을 별도로 제공하지 않는 대신에 다양한 구문 예제를 제공한다.

2. 피코 블록(Pico Blocks)

피코 블록은 교육용 로봇인 피코 크리켓(Pico Cricket)을 프로그래밍하기 위한 인터페이스로 프로그래밍의 원리를 반영하여 설계된 EPL(Educational Programming Language)이다. 피코 크리켓은 Papert의 구성주의적 교육 철학을 기반으로 MIT 미디어랩과 레고(LEGO), 플레이풀 인벤션 컴퍼니(Playful Invention Company)의 공동 연구결과로 2006년에 출시되었으며 철사와 실, 종이, 재활용 용품을 활용하여 만든 창작물에 생명력을 가미하는 활동을 통해 프로그래밍의 원리와 로봇 원리를 학습할 수 있다(송정범 & 이태욱, 2008 ; Pico Cricket, 2006). <그림 2-8>을 보면 알 수 있듯이 피코 블록은 문자와 색상을 기반으로 명령어를 구분한다.

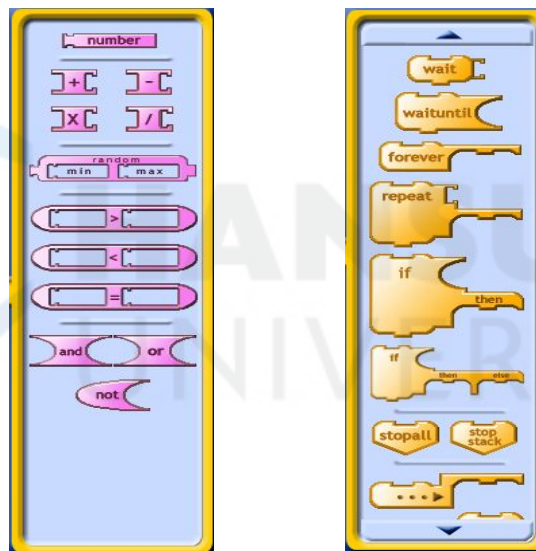


<그림 2-8> Pico Blocks의 GUI

EPL은 초보자를 대상으로 한 프로그래밍 교육을 목적으로 갖기 때문에 VPL과 흡사한 시각적 형상 명령어를 기반으로 하며 컴퓨터 프로그래밍 언어의 원리를 많이 반영한 형태이다. 때문에 피코 블록은 <그림 2-9>와 같이 컴퓨터프로그래밍의 조건 분기, 반복과 같은 흐름제어문법이나 사칙

연산, 비교연산, 논리연산과 같은 연산자에 대한 명령어도 지원한다. 다른 교육용 로봇 프로그래밍 환경보다 많은 수의 명령어를 지원함에도 블록 결합 인터페이스가 직관적이어서 사용법이 간단하다.

피코 블록의 명령어 블록은 프로그래밍 문법의 표현을 블록과 블록의 결합부로 나타내어 퍼즐 블록을 결합하듯이 실습하다보면 문법이 익혀지도록 설계되어있다. 또한 명령어 블록의 기능을 표현하기위한 용도로 새로이 상징화된 아이콘을 사용하지 않고 일반적인 연산자나 단어를 그대로 채용하여 가독성을 높였다. 피코블록은 로봇을 이용하여 간단한 발명을 하는데 목적이 맞춰져있어 정해진 개수의 부품만 사용할 수 있다.



<그림 2-9> Pico Blocks의 연산 블록, 흐름 제어 블록

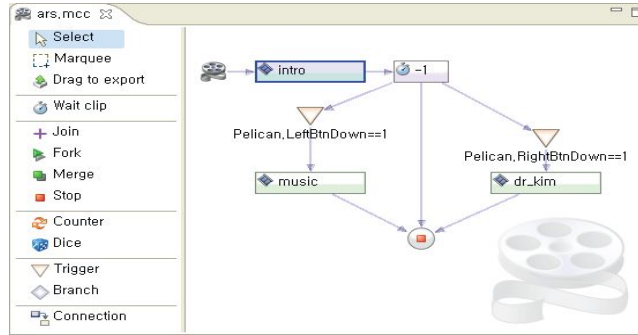
3. 로보이드 스튜디오(Roboid Studio)

로보이드 스튜디오는 로보메이션(ROBOMATION)에서 지식경제부 과제로 만들어진 로봇 전용 개발 플랫폼으로 썬-클라이언트(thin-client) 방식으로 동작하기 때문에 호환성이 높다. 로보이드 스튜디오는 크게 비전문가를 대상으로 하는 콘텐츠 제작기능과 전문가를 대상으로 하는 컴포넌트 개발 플랫폼으로 나뉜다(최정호, 2012 ; Roboid Studio, 2010). 본 논문은 비전문가를 대상으로 한 교육용 로봇 프로그래밍 환경을 개발하는 것이 목표이므로 전문가를 대상으로 한 컴포넌트 개발 플랫폼에 대한 분석하지 않는다.

로보이드 스튜디오는 이클립스(eclipse) 3.5.1을 기반으로 만들어져 있고 이클립스를 기반으로 동작하기 때문에 이클립스의 인터페이스를 그대로 따른다(Roboid Studio, 2010). 콘텐츠 제작기능은 <그림 2-10>과 같은 타임라인 모션 에디터에서 등록된 로봇 모델의 여러 출력장치를 동기화하여 모션클립(motion-clip)을 생성하는 과정과 <그림 2-11>과 같은 콘텐츠 컴포저(composer)에서 모션클립의 실행 흐름을 제어하는 과정을 포함한다(Roboid Studio, 2010).

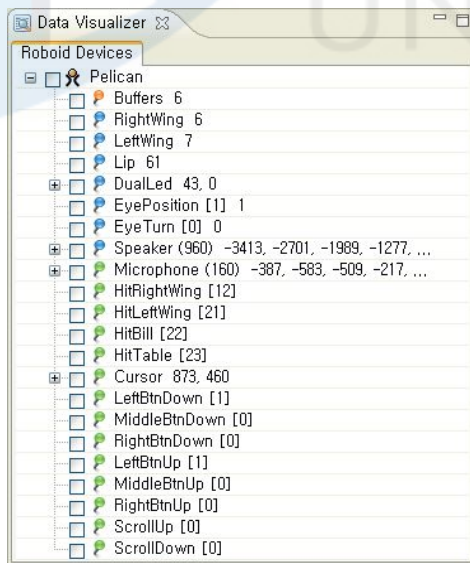


<그림 2-10> 로보이드 스튜디오의 타임라인 모션 에디터



<그림 2-11> 로보이드 스튜디오의 컴포저(Composer)

이외에 <그림 2-12>와 같이 다양한 부가 기능을 제공한다. 로봇 장치의 상태를 확인할 수 있는 데이터-비주얼라이저(data-visualizer) 기능을 제공하여 사용자가 로봇의 현재 상태를 확인할 수 있고 가상으로 콘텐츠를 구동해 볼 수 있는 에뮬레이터(emulator)를 제공하여 로봇이 없어도 콘텐츠를 제작할 수 있다. 반면 이클립스의 인터페이스를 그대로 따르기 때문에 비전문가가 사용하기에는 메뉴가 많고 복잡하며 로봇 모델을 등록하는 과정 또한 어려워 로봇 모델이 제공되는 제품이 주로 사용되고 있다.

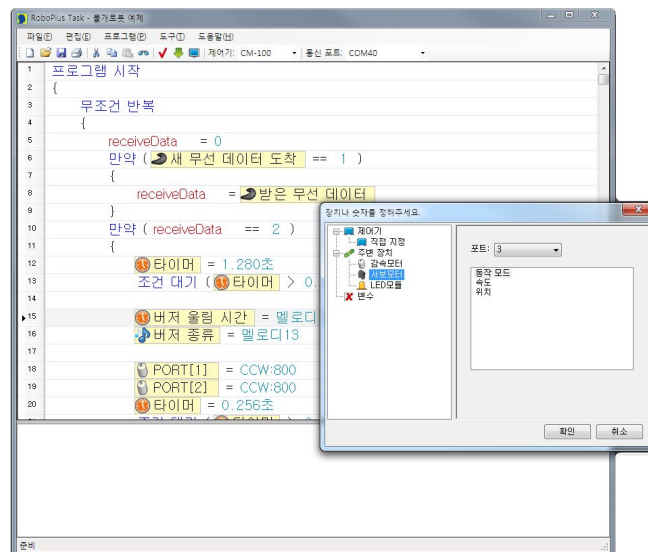


<그림 2-12> 로보이드 스튜디오의 데이터-비주얼라이저(좌)와 에뮬레이터(우)

4. 로보플러스 (RoboPlus)

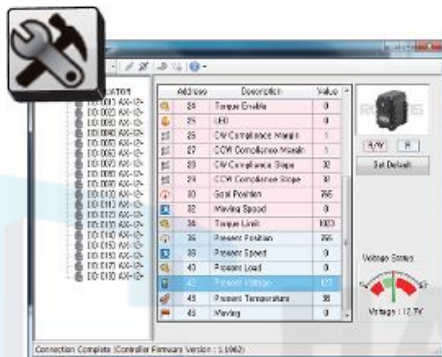
로보플러스는 국내의 로봇 솔루션 업체인 로보티즈(ROBOTIS)에서 개발한 로봇 전용 프로그래밍 환경으로 로보티즈에서 개발된 교육용 로봇 키트부터 연구용 플랫폼까지 모두 지원한다. 로보플러스는 컴퓨터언어(c-language)와 흡사한 문법과 명령어를 사용하여 로봇을 프로그래밍하는 ‘태스크(Task)’와 로봇의 센서 및 구동장치를 점검하는 ‘매니저(Manager)’, 다수의 액추에이터(actuators)를 동기화하여 로봇의 모션을 편집하는 ‘모션 에디터(Motion Editor)’로 구성된다. 태스크는 제어기 펌웨어에 가상머신(virtual machine)을 이식하여 IL(intermediate language)을 구동하는 방식으로 동작하기 때문에 확장성이 좋다(Robotis, 2010).

태스크의 문법과 명령어는 사용자의 프로그래밍 능력 향상을 목표로 <그림 2-13>과 같이 컴퓨터언어와 매우 흡사한 문법체계로 설계되어있어 복잡한 로봇의 프로그래밍이 가능하다. 하지만 초보자가 사용하기에는 명령어의 개수가 많고 구문오류에 대한 대비가 되어있지 않아 전문교육과정을 거쳐야 사용이 가능하다. 태스크는 구문오류를 체크하는 기능을 제공하지만 장치에 대한 디버깅 기능을 제공하지는 않는다.

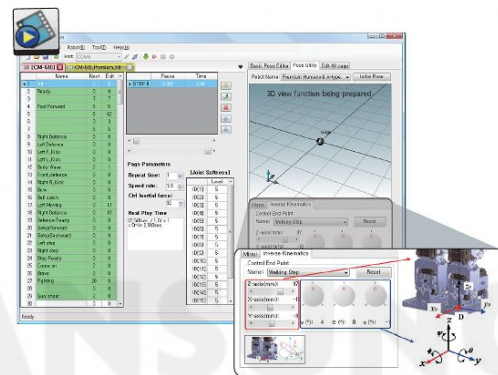


<그림 2-13> 로보플러스의 태스크 GUI

반면에 매니저와 모션에디터와 같은 부가적인 기능을 제공하여 사용자의 프로그래밍활동을 돕는다. <그림 2-14>에 나타난 매니저는 사용자가 스스로 로봇을 점검할 수 있도록 로봇의 장치를 모니터링할 수 있는 기능을 제공하는데 이를 활용하면 우회적인 방법으로 프로그램에 필요한 정보를 습득할 수 있다. 또한 휴머노이드(humanoid)와 같이 다수의 서보모터로 조립된 로봇을 프로그래밍할 경우, <그림 2-15>의 모션에디터를 사용하여 프로그래밍에서 호출할 수 있는 복잡한 모션을 설계할 수 있다.



<그림 2-14> 로보플러스의 매니저



<그림 2-15> 로보플러스의 모션에디터

5. 비전문가용 로봇 프로그래밍 환경의 제한과 시사점

앞서 살펴본 레고 NXT-G와 피코 블록은 초보자가 빠른 시간 내에 사용법을 습득할 수 있도록 명령어의 양을 최소화하였고 블록 명령어를 결합하는 방식으로 프로그래밍 인터페이스를 제공하여 보다 직관적인 절차 프로그래밍이 가능했다. 하지만 하나의 장치를 한번 제어하기위해서 반드시 하나의 블록명령을 사용해야하기 때문에 다수의 모터를 사용하는 고급 로봇을 제어하기에 적합하지 않다. 또한 한 가지의 하드웨어만 전용으로 프로그래밍할 수 있어 호환성이 떨어졌다.

반면에 로보이드 스튜디오와 로보플러스는 다양한 명령어와 모션에디터와 같은 부가적인 기능을 제공하여 다수의 모터를 사용하는 고급 로봇의 프로그래밍을 지원하였지만 초보자 스스로가 사용법을 습득하기에 어려운 프로그래밍 인터페이스를 제공하였다. 하지만 호환이 가능한 로봇이 초급 로봇에서부터 고급로봇까지 다수 존재했다.

비전문가용 로봇 프로그래밍 환경의 분석을 [표 2-1]로 나타낼 수 있으며 이를 바탕으로 본 논문에서는 명령어의 양을 최소화하여 사용법 습득을 빠르게 할 수 있는 직관적인 프로그래밍 인터페이스를 연구하고 나아가 다수의 모터를 사용하는 고급로봇의 효율적인 프로그래밍을 지원하는 모션에디터를 구현한다. 또한 전문적인 디버거 보다는 데이터 뷰어나 장치 매니저와 같이 간접적으로 문제해결을 도울 수 있는 직관적인 모니터의 설계가 필요하다.

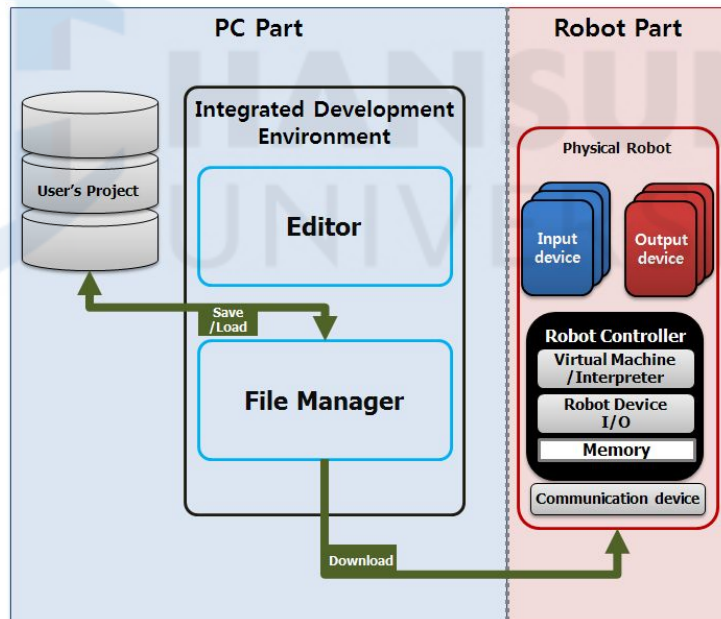
[표 2-1] 비전문가용 로봇 프로그래밍 환경 비교

	사용 난이도	하드웨어 호환성	가독성	부가 기능			
				모션 에디터	장치 매니저	에플 레이터	데이터 뷰어
NXT-G	쉬움	하	하	X	X	X	X
Pico Cricket	쉬움	하	상	X	X	X	X
Roboid Studio	어려움	중	중	O	X	O	O
RoboPlus	어려움	중	상	O	O	O	X

제 3 장 시스템 설계

제 1 절 전체 시스템 구성

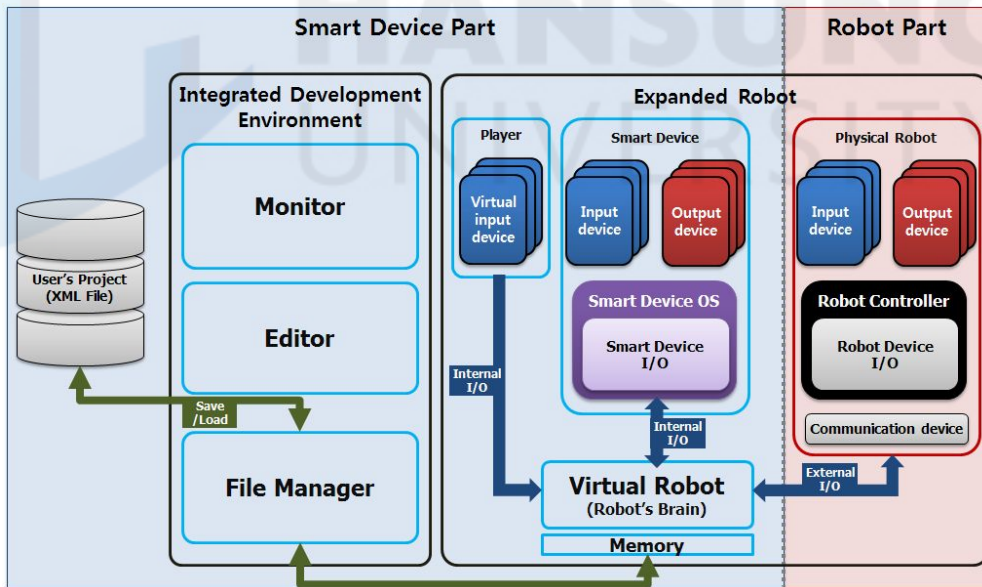
기존의 교육용 로봇에서 프로그래밍 활동은 <그림 3-1>과 같이 사용자가 PC에서 실행되는 통합개발환경을 활용하여 로봇의 프로그램을 작성하고 로봇제어기가 해석할 수 있는 형태로 변환시켜 이식하는 과정으로 이루어진다. 이러한 방법은 사용자가 자신이 작성한 프로그램을 수정하는 과정이 번거로우며 테스트를 수행할시 로봇의 내부 상태를 알기 어려워 로봇 창작 활동을 방해하는 요인이었다. 또한 제품마다 제어기가 프로그램을 해석하는 방식이 다양하기 때문에 각기 다른 개발환경을 통합하기 어려웠다.



<그림 3-1> 기존의 교육용 로봇 시스템

따라서 본 논문에서는 로봇 제어기의 의존성을 줄여 개발환경을 통합하고 나아가 보다 쉽게 프로그램을 수정하고 테스트할 수 있도록 스마트 기기를 활용하고자 한다. 스마트 기기는 교육용 로봇 제어기에 비해 월등한

중앙처리장치(CPU)를 갖고 있어 로봇의 두뇌역할을 하기에 적합한 것처럼 보이지만 로봇의 모터나 기타 입출력장치를 제어하기에는 하드웨어 인터페이스를 갖추고 있지 않다. 따라서 스마트 기기가 로봇의 제어기 역할을 맡게 되면 로봇의 입출력장치와 인터페이스를 하는 I/O 보드의 역할을 다른 기기가 대신해야하는데 이때 이 역할을 기존의 로봇 제어기에 맡기면 기존의 로봇 부품을 그대로 사용할 수 있다. 이는 <그림 3-2>과 같이 표현할 수 있다. 기존의 로봇 제어기가 이 역할을 하기에 기능적으로 가장 좋은 장치라고는 볼 수 없지만 기업에서 부담할 추가적인 개발 비용이나 새로이 로봇을 구입해야하는 사용자의 입장을 고려해볼 때 좋은 선택이라 생각한다. 그리고 제안한 시스템이 특정 하드웨어에 특화되지 않고 상용화된 제품들을 최대한 수용하는 방향으로 개발된다면 제품 간의 소프트웨어 호환성을 보장할 수 있을뿐더러 사용자의 하드웨어 선택폭이 다양해질 수 있다.

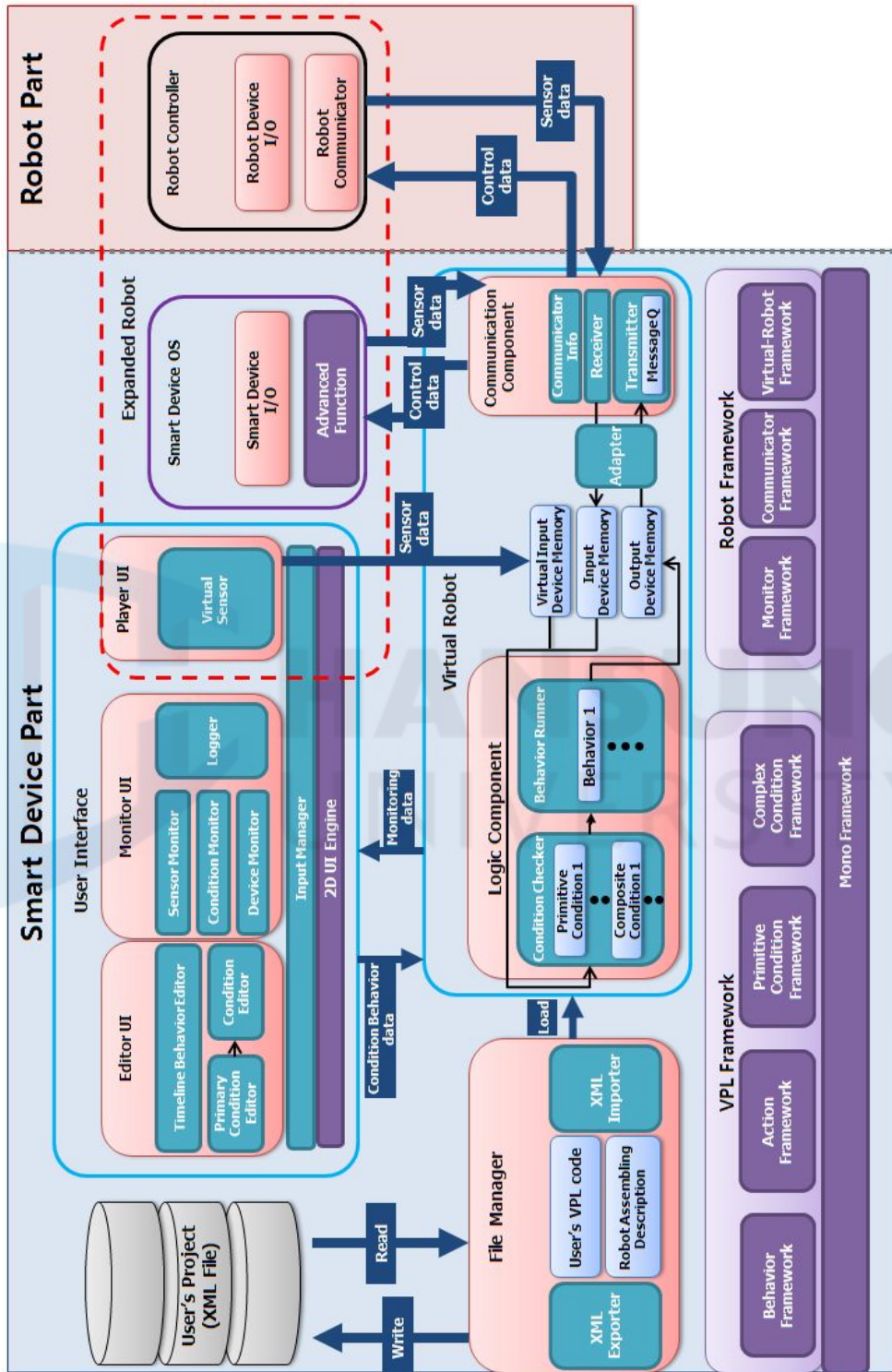


<그림 3-2> 본 논문의 교육용 로봇 시스템

제 2 절 소프트웨어 아키텍처

본 시스템의 소프트웨어는 실행되는 위치에 따라 스마트 기기 파트(smart device part)와 로봇 파트(robot part)로 나누어 <그림 3-3>와같이 설계한다. 스마트 기기 파트는 프로그래밍 사용자 인터페이스(Programming User Interface)와 가상로봇(Virtual Robot), 파일매니저(File Manager)로 구성한다. 사용자의 편집의도에 따라 다양한 변형이 일어날 수 있고 사용자와의 직접적인 상호작용이 일어나는 영역을 프로그래밍 사용자 인터페이스로 구분하여 설계하고 사용자의 프로젝트를 해석하고 실제 로봇과 동기화를 담당하는 영역을 가상로봇으로 구분하여 설계한다. 또한 사용자가 저작한 콘텐츠를 공유가 용이한 파일의 형태로 변환할 수 있도록 파일매니저를 설계한다.

먼저 프로그래밍 사용자 인터페이스는 에디터 UI(Editor UI)와 모니터 UI(Monitor UI), 플레이어 UI(Player UI)로 구성된다. 에디터 UI는 사용자가 프로그래밍을 기초 조건(primitive condition), 복합 조건(composite condition), 행동(behavior) 단위로 나누어 편집할 수 있도록 설계하여 사용자가 프로젝트를 단계적으로 설계할 수 있도록 유도하고 프로그램의 수정이 용이하도록 지원한다. 모니터 UI는 센서모니터(Sensor Monitor), 조건모니터(Condition Monitor), 장치모니터(Device Monitor), 로거(Logger)로 구성되어 로봇의 오작동에 대한 디버깅을 사용자가 직관적으로 진행할 수 있도록 지원한다. 센서모니터와 조건모니터, 장치모니터는 프로그램을 편집하는 중에도 매 프레임(frame)마다 가상로봇과 동기화되어 사용자에게 프로그램 편집에 필요한 센서 정보와 조건 상태 정보, 로봇의 상태정보를 제공한다. 반면에 로거는 프로그램이 실행되는 중에만 작동하며 로봇에 내려진 모든 명령을 로깅(logging)하는 역할을 한다. 플레이어 UI는 사용자가 화면상에 가상의 센서를 추가하여 사용할 수 있는 가상 센서 기능을 제공하여 로봇의 부족한 입력장치를 지원한다. 프로그래밍 사용자 인터페이스는 언제든지 모드전환 스위치를 눌러 편집모드와 실행모드를 변경할 수 있기 때문에 짧은 과정으로 수정과 테스트가 가능하다.

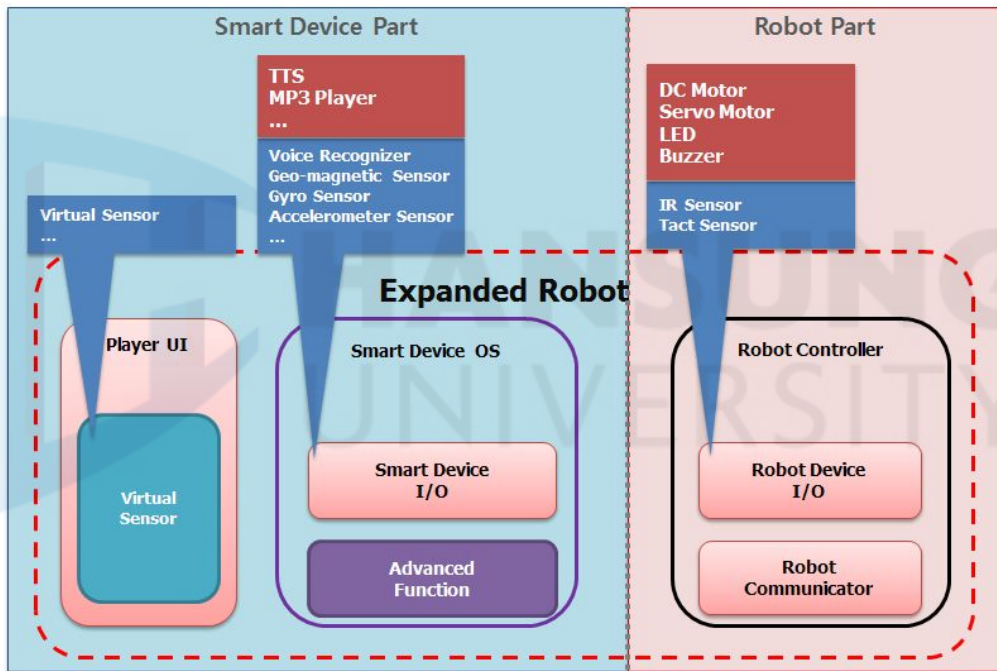


〈그림 3-3〉 소프트웨어 아키텍처(Software Architecture)

가상로봇은 물리적 로봇과 동기화를 수행하는 역할과 사용자가 작성한 프로그램의 논리를 검사하여 실행하는 역할을 한다. 가상로봇의 커뮤니케이션 컴포넌트는 앞서 언급한대로 실제 로봇의 구동을 위해 로봇 커뮤니케이터와 동기화를 진행하거나 스마트 기기의 고급기능을 활용하기 위해 스마트 기기 OS와 통신한다. 프로그램의 논리를 검사와 실행을 담당하는 로직 컴포넌트(Logic Component)는 조건 검사자(Condition Checker)와 행동 실행기(Behavior Runner)로 구성되어 사용자가 작성한 프로그램에 따라 가상로봇을 제어한다. 조건 검사자는 조건 검사 주기마다 $sensor1 > 30$ 와 같은 기초 조건들의 목록을 일괄적으로 검사한 후 결과를 조건 모니터에 갱신한다. 기초 조건에 대한 검사가 완료되면 기초 조건들의 논리 연산으로 정의된 복합 조건들에 대한 검사를 일괄적으로 시행한다. 복합 조건이 만족할 경우 이와 대응된 행동이 행동 실행기에 후보행동(candidate behavior)으로 추천된다. 행동 실행기는 현재 실행중인 행동이 존재할 경우 후보행동과 우선순위를 경쟁하여 높은 우선순위를 가진 행동을 실행한다.

로봇 파트는 로봇 제어기에 이식된 로봇 장치 I/O(Robot device I/O)와 로봇 커뮤니케이터(Robot Communicator)로 구성되는데 이는 로봇 제품에서 지원하는 프로그래밍 소프트웨어를 통해 쉽게 구현할 수 있다. 로봇 파트는 단순히 스마트 기기에서 하달한 명령을 수행하는 역할과 로봇 센서로 취득한 정보를 스마트 기기에 제공하는 역할을 한다. 이러한 과정에서 로봇 장치 I/O는 로봇 센서로 취득한 미가공 데이터(raw data)를 다운샘플링(down-sampling)하여 로봇 커뮤니케이터에 넘겨주는 역할과 로봇 커뮤니케이터로부터 넘겨받은 패킷을 해석하여 액추에이터에 전달하는 역할만을 담당하며, 실제로 스마트 기기와의 통신은 가상로봇 내에 위치한 커뮤니케이션 컴포넌트(Communication Component)와 로봇 커뮤니케이터 사이의 외부 연결(external connection)을 통해 이루어진다. 이때 외부 연결은 로봇 하드웨어에서 지원하는 통신방법에 따라 다양한 방법을 적용할 수 있는데, 대부분의 교육용 로봇은 리모트 컨트롤을 위한 블루투스 장치를 지원하므로 이를 사용하는 것이 적합하다.

본 소프트웨어는 교육용 로봇의 기능을 추가로 지원하기 위해서 스마트 기기의 고급기능을 <그림 3-4>와 같이 물리적 로봇(physical robot)의 부품처럼 활용한다. TTS나 음성인식, 고급 센서 등 로봇에서 활용될 수 있는 스마트 기기의 고급기능은 대부분 스마트 기기의 OS를 통해 제어할 수 있다. 스마트 기기의 고급기능을 제어할 때에도 로봇을 제어할 때와 같이 가상로봇 내의 위치한 커뮤니케이션 컴포넌트에 의해서 이루어지지만 스마트 기기의 OS와 본 시스템은 물리적으로 같은 공간 내에 위치하므로 별도의 통신 기기를 사용하지는 않는다.



<그림 3-4> 스마트 기기와 연동하여 확장된 로봇(Expanded Robot)

본 시스템을 사용하여 저작된 사용자 콘텐츠는 기존의 교육용 로봇처럼 제어기에 이식할 필요 없이 스마트 기기에서 바로 실행되므로 로봇에 프로그램을 다운로드하는 기능은 구현하지 않는다. 따라서 사용자 콘텐츠를 파일 형태로 저장하면 프로젝트 파일을 공유하는 것만으로 사용자간에 콘텐츠 공유가 가능해진다. 파일매니저(File Manager)는 사용자 프로젝트를

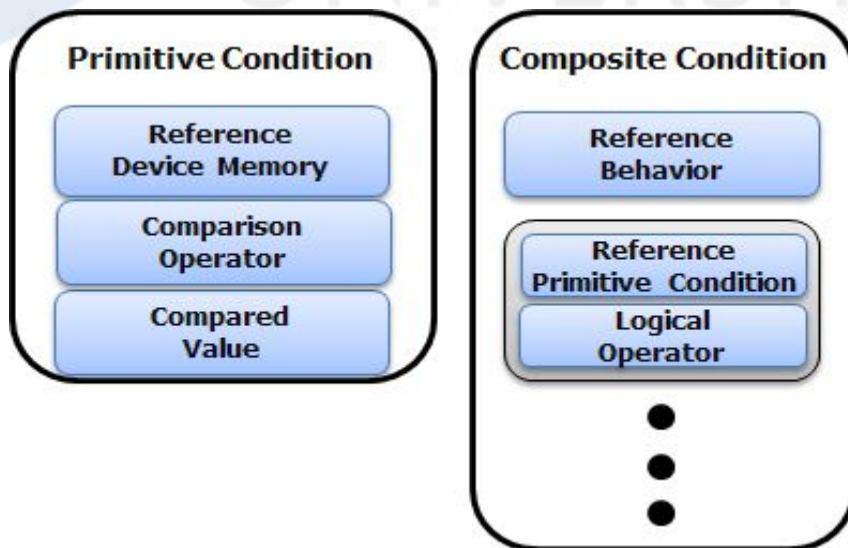
XML 형식으로 변환하여 스마트 기기의 파일 시스템(file system)에 저장하는 역할을 한다. 이때 사용자 프로젝트에는 사용자가 작성한 프로그램코드(VPL code)와 사용자 조립 명세서가 포함된다.

본 시스템은 프레임워크를 기반으로 동작하는데 이는 향후 새로운 명령어나 새로운 로봇이 추가되어도 사용자가 저장한 콘텐츠에 영향을 미치지 않기 위해 설계된 VPL 프레임워크와 다양한 로봇을 지원하기 위해 설계된 로봇 프레임워크로 구성된다. 이러한 프레임워크뿐만 아니라 스마트 기기 파트 전체는 유니티3D(Unity3D)라는 상용화된 엔진을 기반으로 구현하는데 이는 여러 가지의 장점을 갖는다. 유니티3D는 3차원 게임개발 엔진으로 마이크로소프트(Microsoft) 닷넷(.NET) C#의 막강한 API(Application Programming Interface)를 이기종의 하드웨어에서 지원하기 위해 오픈소스화한 모노프로젝트(Mono project)를 기반으로 한다. 유니티3D를 활용하면 엔진에서 제공하는 3차원 그래픽 기능의 변형으로 2차원의 사용자 인터페이스를 표현할 수 있고 모노프로젝트의 API를 활용한 빠른 개발이 가능할 뿐만 아니라 윈도우즈(Windows), 아이오에스(iOS), 안드로이드(Android) 등 다양한 OS(Operating System)에 대해서 호환성이 보장된 개발환경을 개발할 수 있게 된다. 기존의 로봇 프로그래밍 환경은 PC전용으로 개발되어 실습환경이 제한되었지만 본 논문에서 제안한 방법은 OS에 대한 호환성을 보장하므로 실습환경의 제한을 극복할 수 있다.

제 3 절 교육용 로봇 프로그래밍 언어의 설계

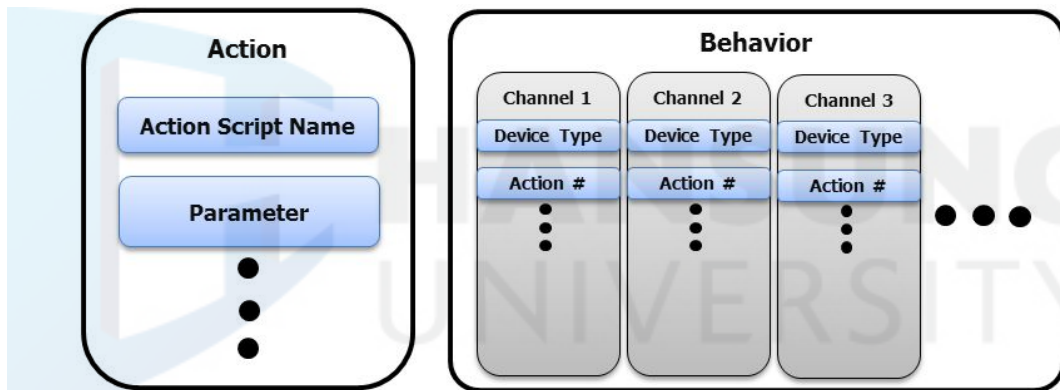
1. T-R 시퀀스와 직관적인 행동 편집

본 논문에서 제안하는 교육용 로봇의 프로그래밍 언어는 Nilsson이 제안한 T-R 시퀀스를 기반으로 한다. T-R 시퀀스는 일반적인 생산 규칙 (production-rule)과 흡사하게 동작하지만 조건이 만족하지 않는 즉시 행동을 멈추고 하위 우선순위의 조건을 검사하여 예외상황에 대비가 제대로 되어있지 않아도 오동작을 막을 수 있다. 때문에 로봇의 예외상황을 모두 예측하기 어려운 초급자가 사용하기에 적합하다고 판단된다. 사용자가 단계적으로 조건을 설계할 수 있는 인터페이스를 제공하고 고급 행동 편집 기능을 제공하면 T-R 시퀀스만으로 반응형 행동 기반의 로봇을 만들 수 있다. 본 논문에서 제안하는 프로그래밍 언어는 조건을 편집하는 과정을 두 단계로 나누어 진행할 수 있도록 구현한다. 먼저 $sensor1 > 10$ 과 같이 단순한 조건을 기초 조건으로 정의한 후 기초 조건과 논리연산자를 조합하여 보다 복잡한 조건을 정의한다. 이러한 두 조건은 <그림 3-5>와 같은 데이터 구조를 갖는다.



<그림 3-5> 기초조건과 복합조건 데이터 구조

절차 프로그램을 사용하는 환경에서 초급자가 로봇의 다양한 장치를 사용하는 행동을 설계하는 것은 쉽지 않다. 절차 프로그램은 명령어를 넣는 순서에 따라 실행되기 때문에 여러 장치간의 동기화가 어려우며 복잡한 행동을 구현하는데 있어서 시간 예측이 힘들다. 때문에 초급자가 고급 행동을 저작하기 위해서는 로보이드 컴포넌트의 타임라인 에디터나 로보플러스의 모션에디터와 같은 고급기능이 지원되어야한다. 본 논문에서 타임라인 기반의 행동 에디터를 구현하기 위해 <그림 3-6>와 같은 데이터 구조로 단위행동(action)과 행동을 설계한다. 행동은 본 시스템에서 제공하는 단위행동을 시간적으로 나열한 목록으로 정의되며 사용하는 장치의 수만큼 타임라인 채널을 갖는다.



<그림 3-6> 단위행동(action)과 행동(behavior)의 데이터 구조

2. 절차적 표현을 위한 전역 변수

본 논문에서 제안하는 교육용 로봇의 프로그래밍 언어는 레고 NXT-G와 같이 단순한 아이콘 명령어를 사용하는 VPL 체계로 구현하지만 레고 NXT-G와 다르게 절차프로그래밍이 아닌 반응 중심의 T-R 시퀀스를 기반으로 설계하기 때문에 절차적인 프로그램을 작성하기에 적합하지 않다. 또한 T-R 시퀀스는 단순한 반응 중심의 예제를 구현하기에는 직관적이지만 계층적 구조를 표현하기 위해서는 지역화 개념을 추가해야 한다. 따라서 본 논문에서는 T-R 시퀀스에 두 가지의 전역변수 개념을 추가하여 자주 사용하는 절차프로그램과 계층적 구조를 표현한다.

흔히 사용하는 절차 프로그램의 예는 $K_1 \rightarrow b_1 \rightarrow b_2$ 와 같이 K_1 이 만족하여 b_1 이 수행된 후 연이어 b_2 를 수행하는 상황이다. 이는 T-R 시퀀스에서 행동을 설계할 때 b_1 과 b_2 이외에 $b_1 \rightarrow b_2$ 를 새로운 행동 b_3 로 정의하여 $K_1 \rightarrow b_3$ 과 같이 해결할 수 있지만 이러한 해결법은 사용자에게 많은 행동을 작성해야 하는 부담을 준다. 따라서 보다 근본적인 대안이 필요한데 가장 최근에 종료된 행동을 기록하는 전역변수를 추가하여 해결할 수 있다. 전역변수 “가장 최근에 종료된 행동”은 사용자에게 의해 추가되거나 값 변경이 일어날 수 없으며 반드시 시스템에 의해서 동작하고 1개만 존재할 수 있다. 앞서 제시한 전역변수 “가장 최근에 종료된 행동”을 활용하여 $K_1 \rightarrow b_1 \rightarrow b_2$ 를 제표현하면 다음과 같은 T-R 시퀀스로 표현할 수 있다.

$$K_2 \rightarrow b_2$$

$$K_1 \rightarrow b_1$$

여기서 K_2 는 “가장 최근에 종료된 행동이 b_1 이다.”로 정의한다. 이와 같은 방법으로 행동이 연이어지는 절차 프로그램의 일부를 표현할 수 있다.

일반적으로 많이 사용하는 계층적 구조는 로봇이 두 가지 이상의 임무를 절차적으로 수행해야 할 때 사용된다. 두 가지의 절차적인 임무를 갖는 “자원을 수집하여 집으로 돌아오는 로봇”을 예를 두 번의 T-R 시퀀스로 표현하면 다음과 같다.

자원을 찾았다 → 자원을 수집한다
 장애물을 만났다 → 장애물을 피한다
 $T \rightarrow$ 돌아다닌다

집을 찾았다 → 집으로 돌아간다
 장애물을 만났다 → 장애물을 피한다
 $T \rightarrow$ 돌아다닌다

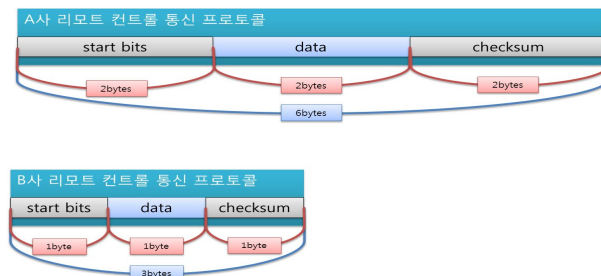
자원을 수집하는 임무가 완료되면 자원을 수집하는 프로그램이 종료되고 집으로 돌아가는 임무의 프로그램이 실행된다. 이와 같은 프로그램을 하나의 T-R 시퀀스로 표현하기 위해서는 T-R 프로그램과 같이 지역함수(local function) 개념이 등장해야하는데 이는 초급자에게 혼란을 안겨준다. 따라서 본 논문에서는 지역함수의 개념을 사용자가 이해하기 쉬운 “현재 계획(current plan)”이라는 메타포(metaphor)를 사용하여 일부만 제공하고 자한다. “현재계획”은 앞서 언급한 “가장 최근에 종료된 행동”과 같이 전역변수이며 사용자에게 의해 추가될 수 없고 반드시 1개만 존재한다. 하지만 “가장 최근에 종료된 행동”변수는 “가장 최근에 종료된 행동”변수와 다르게 사용자가 값을 변경할 수 있으며 어떤 행동이 종료될 때를 하나의 조건으로 사용하여 계획을 변경할 수 있다. 앞서 예로 들었던 “자원을 수집하여 집으로 돌아오는 로봇”을 “계획”개념을 추가한 한 번의 T-R 시퀀스로 표현하면 다음과 같다.

현재계획이 B이다 and 집을 찾았다 → 집으로 돌아간다
 현재계획이 B이다 and 장애물을 만났다 → 장애물을 피한다
 현재계획이 계획B이다 → 돌아다닌다
 자원을 찾았다 → 자원을 수집한다 after 현재계획을 B로 변경한다
 장애물을 만났다 → 장애물을 피한다
 $T \rightarrow$ 돌아다닌다

제 4 절 다양한 제품 수용을 위한 제어 프로토콜

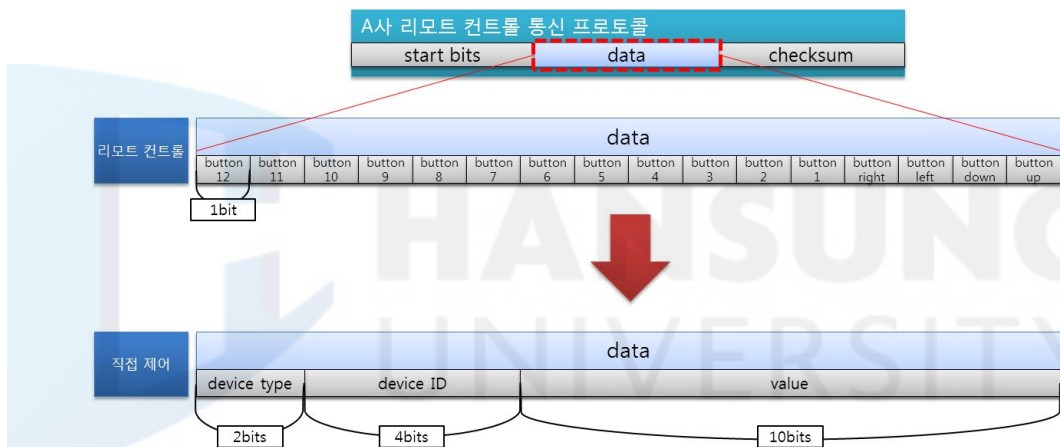
본 시스템이 다양한 제품의 교육용 로봇 하드웨어를 지원하려면 스마트 기기와 로봇간의 통신 프로토콜을 표준화하여 상용화된 로봇에 모두 적용하거나, 기존에 각 제품들이 사용하던 프로토콜을 모두 수용해야한다. 통신 프로토콜을 표준화하는 방법은 이상적인 방법이지만 기준을 정하는데 있어서 시장의 반발을 살 수 있을뿐더러 실제 적용까지 많은 시간이 필요하다. 또한 프로토콜이 바뀌게 되므로 기존에 만들어진 콘텐츠 체계가 무너질 수 있다. 반면 기존에 제품들이 사용하던 프로토콜을 모두 수용하는 방법은 기존에 이미 만들어진 콘텐츠 체계에 영향을 미치지 않는다는 장점이 있지만 수용할 제품이 추가될 때마다 개발에 소요되는 시간이 추가로 요구된다. 두 가지 방법 중 기존의 콘텐츠 체계에 영향을 미치지 않기 위해서 기존의 제품들이 사용하던 프로토콜을 모두 수용하는 방법을 선택하고 수용 제품이 추가될 때마다 필요한 개발시간을 줄이기 위하여 제어 프로토콜을 지정한다.

대부분의 교육용 로봇 제품들은 리모트 컨트롤을 지원하기위해 블루투스 장치를 지원하며 대부분의 스마트 기기도 블루투스 통신 장치를 내장하고 있다. 따라서 본 시스템이 로봇 제어에 블루투스 통신을 사용하게 되면 추가로 통신 장치를 개발하지 않아도 된다. 또한 교육용 로봇 제품들의 블루투스 통신 프로토콜은 대개 리모트 컨트롤에 적합하도록 설계되어있어 유효데이터의 범위가 비슷한데 이는 <그림 3-7> 과 같다. 유효데이터의 크기에 따라 제품을 분류하면 유효데이터 범위가 제품들은 같은 제어 프로토콜을 사용할 수 있어 개발 시간을 줄일 수 있다.



<그림 3-7> 교육용 로봇 제품의 블루투스 리모컨 프로토콜

로봇의 제어기에는 스마트 기기에서 전송한 데이터를 해석하여 로봇을 제어하기 위한 제어 프로토콜이 이식되어야 한다. 제어 프로토콜은 스마트 기기로부터 전송받은 패킷을 해석하여 제어하고자 하는 장치 종류(type)와 장치 식별자(ID), 값(value)을 알아내기 위한 규약과 환경으로부터 취득한 센서 데이터를 스마트 기기가 해석할 수 있는 패킷으로 만드는 규약을 포함한다. 또한 이러한 규약대로 만들어진 패킷을 전송하는데 사용할 리모트 컨트롤용 통신 프로토콜도 포함한다. 장치 1개를 제어하기 위해 필요한 정보량은 유효 패킷의 크기로 제한되며 만약 이를 넘을 경우에는 제어 값의 해상도(resolution)을 낮추거나 수용가능한 장치의 수를 줄여야 한다.



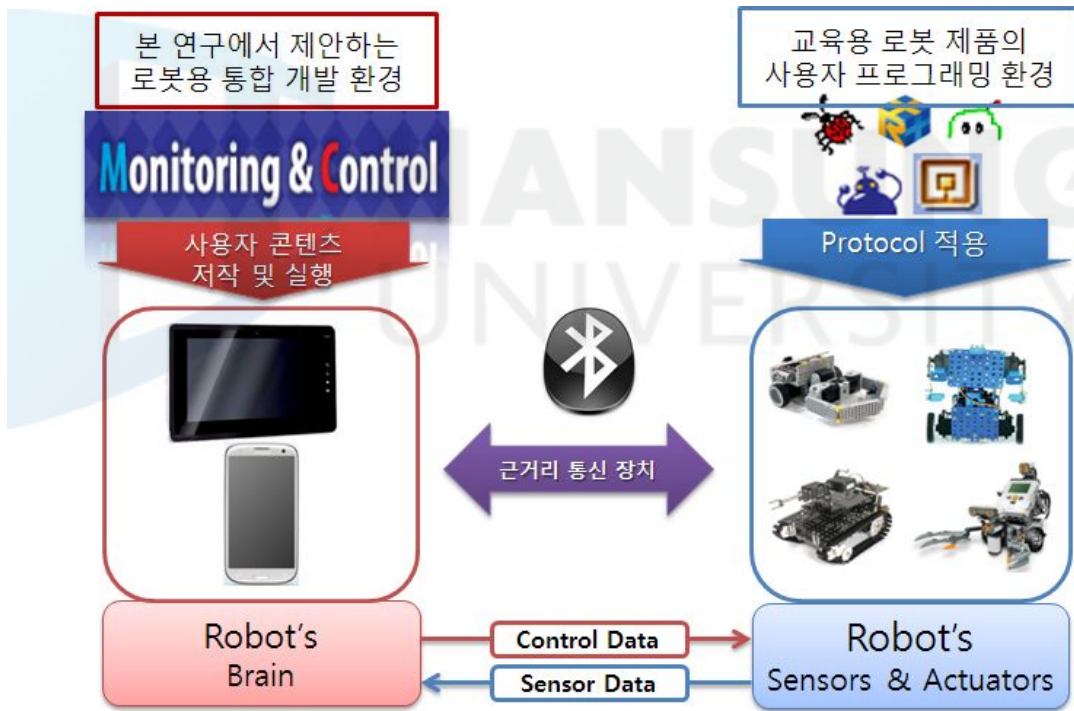
<그림 3-8> 2바이트 유효 패킷 분할의 예

<그림 3-8>은 리모트 컨트롤용 통신 프로토콜의 유효 패킷을 분할하여 사용하는 예시이다. 먼저 리모컨 제어에 사용할 경우에는 동시에 여러 버튼이 눌렸는지 확인하기 위하여 유효 패킷을 1비트(bit) 단위로 나누어 리모컨의 각 버튼에 대응시킨다. 이러한 방법은 동시에 여러 명령을 내릴 수 있다는 장점을 지니지만 수용할 수 있는 명령의 가짓수가 유효 패킷의 비트 수로 제한된다는 단점을 갖는다.

때문에 로봇을 직접 제어하기 위해 이를 변형하면 한 번의 유효 패킷에 하나의 장치만을 제어할 수 있는 명령만 담아 패킷의 크기를 확보해야 한다. 이때 유효 패킷은 장치의 종류와 장치의 식별자, 값을 담기 위해 3개의

필드로 분할된다. 식별자 필드는 원하는 장치를 구분하기 위해서 반드시 필요하고 값 필드는 명령의 정도(level)를 세분화하기 위해 필요하다. 종류 필드는 장치의 종류에 따라 값을 처리하기 위한 방법이 달라지기 때문에 포함돼야한다.

이와 같은 방법은 <그림 3-9>와 같이 제품이 기존에 사용하던 프로그래밍 방식을 그대로 사용하여 구현될 수 있기 때문에 이미 상품화된 제어기 내에 이식되어있는 펌웨어를 수정하지 않고도 적용할 수 있다. 펌웨어를 수정하지 않아도 된다는 점은 기존에 활성화되어있던 콘텐츠 체계에 영향을 미치지 않기 때문에 기업입장에서 큰 위험부담을 갖지 않고 시도해볼 수 있다(김진성 & 조혜경, 2011).



<그림 3-9> 다종의 교육용 로봇 플랫폼 수용

제 5 절 가상로봇

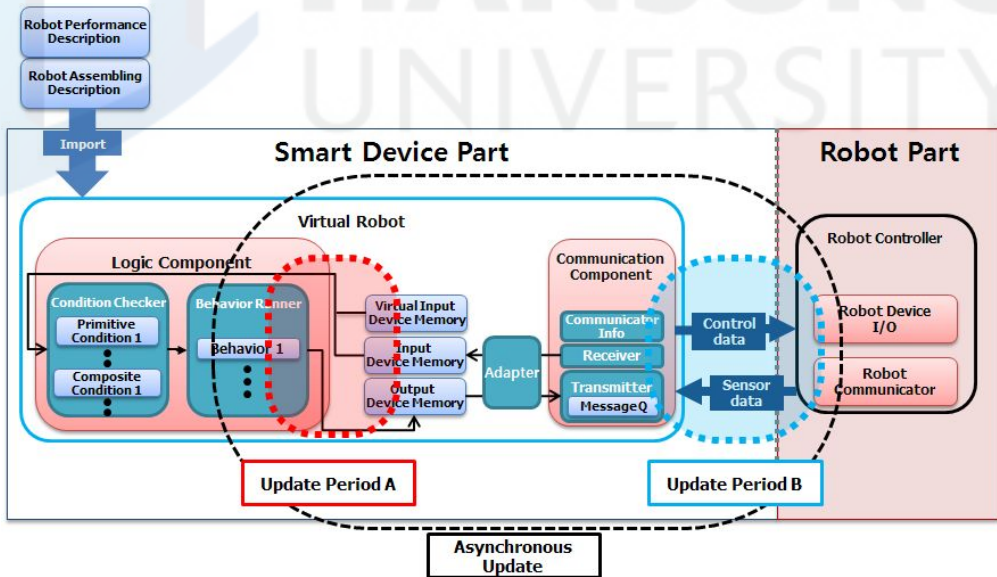
교육용 로봇 프로그래밍 환경은 일반적인 로봇 프로그래밍 환경과 다르게 사용자가 선택한 로봇 제품과 조립에 사용한 부품에 따라 다양한 형태로 변할 수 있기 때문에 이를 수용할 수 있는 준비가 필요하다. 사용자가 창작한 로봇을 프로그래밍할 수 있도록 지원하려면 프로그래밍 환경에서 창작에 사용된 로봇부품에 대한 편집만 가능하도록 제한을 두어야 한다. 이로써 사용자가 조립에 사용하지 않은 부품에 대한 행동을 정의하지 못하게 막을 수 있고 사용한 부품에 대해서만 모니터링 기능을 제공할 수 있어서 혼돈되는 상황을 사전에 방지할 수 있다.

사용자가 창작한 로봇에 맞춰진 기능을 제공하려면 사용자가 조립에 사용한 제품의 종류와 조립한 형태에 대한 정보가 필요하다. 이러한 정보는 로봇 사양 명세서(Robot Performance Description)와 로봇 조립 명세서(Robot Assembling Description)를 작성하므로 취득할 수 있다. 로봇 사양 명세서는 제품의 종류에 따라 제어에 필요한 변수의 종류와 값의 범위, 제어 해상도, 장치 주소값 등을 기술한 형태이다. 이는 본 시스템이 새로운 제품을 수용하고자할 때 반드시 필요하다. 로봇 조립 명세서는 사용자가 조립에 사용한 제품의 종류와 부품의 개수, 포트연결에 대한 정보를 담고 있다.

이 두 가지 명세서를 기반으로 생성된 가상로봇은 실제 조립된 로봇과 실행중인 사용자 프로그램의 차이를 줄이는 역할을 한다. 가상 로봇은 제품의 하드웨어 성능을 바탕으로 사용자가 선택할 수 있는 편집 범위를 제한하고 사용자가 편집한 값을 실제 사용될 데이터의 형태로 변환하여 장치별 기억 공간(Device Memory)에 저장한다. 장치별 기억 공간에 저장된 데이터는 통신 컴포넌트에 의해 주기적으로 실제 로봇에 전송되어 실시간으로 갱신된다. 이때 두 개의 유동적인 갱신 주기(update period)를 사용하여 갱신의 안정성을 보장한다. <그림 3-10>에 나타난 갱신 주기A는 현재 실행되고 있는 행동을 장치별 기억공간에 갱신하기 위한 주기이며 이는 프로그램을 구동하는 스마트 기기의 중앙처리장치(Central Processing Unit)

영향을 받는다. 갱신 주기B는 블루투스 장치를 활용한 가상로봇의 장치별 기억공간의 데이터를 실제 로봇장치에 갱신하기위한 주기이며 대부분의 교육용 로봇이 UART(Universal Asynchronous Receiver and Transmitter) 통신을 사용한다. 이는 비동기 통신이며 블루투스 모듈의 속도와 이를 처리하는 로봇 제어기의 중앙처리장치 영향을 받는다. 두 번의 갱신 주기 사이에는 장치별 기억 공간이 걸쳐있는데 이를 통해 현재 실행되는 행동과 실제 로봇 장치 사이에 비동기 갱신(asynchronous update)이 이루어진다.

현재 실행되는 행동에 따라 로봇 장치를 자주 갱신할 필요가 발생할 수 있는데 이때 갱신 주기를 높이지 않으면 메시지 큐가 넘치는 상황이 발생할 수 있다. 또한 갱신이 불필요한 경우에는 갱신주기를 낮춰 배터리 소모를 줄여야한다. 유동적으로 갱신주기를 조절하기위해서 가상로봇의 커뮤니케이션 컴포넌트와 장치별 기억 공간사이에 이를 조절해주는 어댑터를 둔다.



<그림 3-10> 가상 로봇과 실제 로봇의 비동기 갱신

제 4 장 구현 및 결과

제 1 절 구현 환경

본 시스템은 다양한 운영체제와 장치를 지원하기 위하여 멀티 플랫폼을 지원하는 유니티3D(Unity3D) 엔진을 기반으로 개발하였으며 전반적인 프로그래밍은 C# 언어를 기반으로 개발하였다. 지원 가능한 운영체제와 실행 장치는 [표 4-1]과 같다. 윈도우즈를 사용하는 PC의 경우 USB 2.0을 지원하는 경우에 모두 실행 가능했으며 안드로이드를 사용하는 스마트 기기의 경우 OS버전 2.2이상의 블루투스2.0 통신을 지원하는 장치에서 사용 가능했다. 하지만 iOS를 사용하는 스마트 기기의 경우에는 국내 교육용 로봇의 블루투스 장치와의 통신을 허용하지 않아 로봇동작이 불가능했다. 이는 향후에 해당 블루투스 장치를 허가하거나 iOS에서 인증하는 보안칩을 지원할 경우 사용가능할 것으로 보인다.

[표 4-1] 지원 가능한 운영체제 및 실행 장치

운영체제	실행 장치 (확인된 장비)	사용자 콘텐츠 편집 유무	사용자 콘텐츠 실행 유무
Windows XP(32bits/64bits) 이상 버전	데스크톱 PC 노트북	O	O
Android 2.2 이상 버전	삼성 갤럭시 탭 10.1 삼성 갤럭시 노트 10.1 삼성 갤럭시 S 삼성 갤럭시 S2 삼성 갤럭시 S3 삼성 갤럭시 플레이어 LG 옵티머스 G LG 옵티머스 LTE 2	O	O
iOS 4.0 이상 버전	아이폰 4 아이폰 4S 아이패드 아이패드2	O	X

본 시스템을 적용해본 교육용 로봇 하드웨어는 로보티즈(ROBOTIS)의 STEM-kit과 로보로보(ROBOROBO)사의 School-kit이며 [표 4-2]와 같은 기능을 지원한다. 이 두 제품은 초등학생 및 중학생을 주 대상으로 개발된 로봇으로 블루투스2.0 통신을 지원하여 본 시스템의 적용이 가능하였다. STEM-kit에서 제공하는 모터는 마이크로제어 장치(MCU : Micro Controller Unit)를 내장하고 있어 각도와 속도를 제어할 수 있을 뿐만 아니라 모터의 현재 상태를 읽을 수 있어 본 시스템에서 제공하는 장치 모니터링 기능을 사용할 수 있었다. 반면에 School-kit에서 제공하는 모터는 아날로그 서보모터로 모터의 현재 상태를 읽을 수 없어서 장치 모니터링 기능이 제한됐다. 또한 STEM-kit의 통신 프로토콜은 유효패킷의 크기가 2바이트라서 제어 값의 해상도를 8~10비트 정도로 넉넉히 할당할 수 있었던 반면에 School-kit의 통신 프로토콜은 유효패킷의 크기가 1바이트라서 제어 값의 해상도를 비교적 낮게 할당해야했다. 이처럼 본 시스템은 블루투스2.0 통신이 가능한 교육용 로봇 제품군을 수용할 수 있지만 하드웨어 성능에 따라 일부 기능이 제한될 수 있다.

[표 4-2] 테스트 로봇의 지원 기능

제품명	제품별 지원 기능								
	모터 속도 제어	모터 각도 제어	모터 속도 읽기	모터 각도 읽기	LED 제어	이진 센서	ADC 센서	블루 투스 2.0	유효 패킷 크기
STEM-kit	O	O	O	O	O	O	O	O	2bytes
School-kit	O	O	X	X	O	O	X	O	1bytes

제 2 절 사용자 인터페이스 구현

본 시스템은 터치스크린 입력을 사용하는 스마트 기기와 마우스, 키보드 입력을 사용하는 PC에서 일관성 있는 조작법을 제공하기 위하여 기본적인 조작행위를 [표 4-3]과 같이 정의하였다.

[표 4-3] 가상센서, 조건, 행동의 편집에 사용하는 조작행위 정의

조작 행위	마우스-클릭(click)	손가락-터치(touch)
선택	대상 1회 클릭 (click)	대상 1회 탭 (tap)
추가	특정영역 1회 클릭 (click)	특정영역 1회 탭 (tap)
삭제	대상을 특정영역으로 드래그 & 드롭 (drag & drop)	대상을 특정영역으로 드래그 & 드롭 (drag & drop)
이동	대상 드래그 & 드롭(drag & drop)	대상 드래그 & 드롭(drag & drop)
수정	대상 1.5초 누르고 있기 (press)	대상 1.5초 누르고 있기 (press)
값 편집	슬라이드 바(slide bar) 밀기	슬라이드 바(slide bar) 밀기

앞서 정의한 기본적인 조작행위를 기반으로 사용자 인터페이스를 <그림 4-1>와 같이 구현하였다. 전체적인 사용자 인터페이스는 프로그래밍에 필요한 모든 정보를 한눈에 확인할 수 있도록 배치하였으며 프로그래밍의 단계별로 영역을 분할했다.

<그림 4-1>에서 ❶은 로봇의 환경 설정 기능으로 이를 통해 사용자가 저장한 로봇의 정보를 읽어드리고 로봇과 스마트 기기의 통신을 연결할 수 있도록 구현하였다. <그림 4-1>에서 ❷은 프로젝트 관리 기능으로 사용자가 저장한 콘텐츠를 저장하고 불러오는 기능과 실행하는 기능을 담당한다. <그림 4-1>에서 로봇 프로그래밍에 필요한 영역은 ❸, ❹, ❺, ❻로 구성했다.

먼저 <그림 4-1>의 ❸은 모니터 영역으로 사용한 로봇의 센서와 장치의 모니터링 기능을 하도록 구현했다. 모니터링 영역은 실시간으로 로봇과 동기화하여 사용자가 프로그램을 편집하는 중에도 로봇의 상태를 확인할

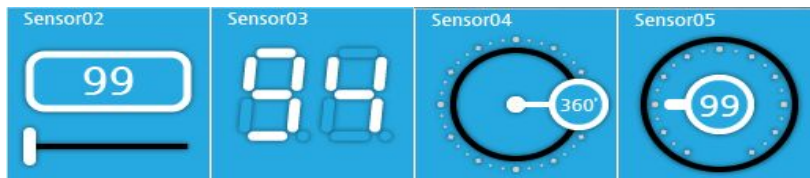
수 있도록 돕는다. 초보자의 눈높이에 맞는 다양한 모양의 모니터를 제공하여 흥미를 유발했다. 모니터는 측정하는 물리량의 특성에 따라 이진 센서(binary sensor)와 ADC(analog-to-digital converter)센서로 구분되며 <그림 4-2>, <그림 4-3>과 같이 다양한 모양으로 표현된다. 가상 로봇에 의해 일정주기마다 로봇의 장치상태를 갱신하여 모니터 특성에 맞게 스케일링(scaling)을 한 뒤 모니터 아이콘이나 수치로 시각화된다.



<그림 4-1> 전체 GUI 화면 구성

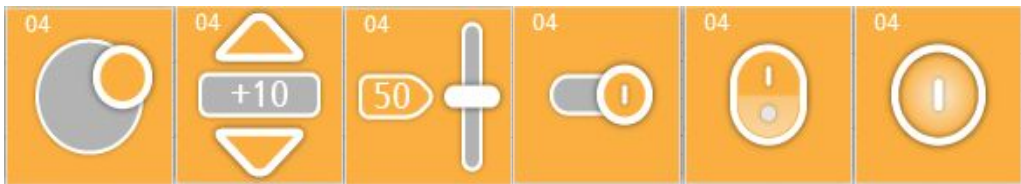


<그림 4-2> 이진 값(binary data) 형식의 모니터



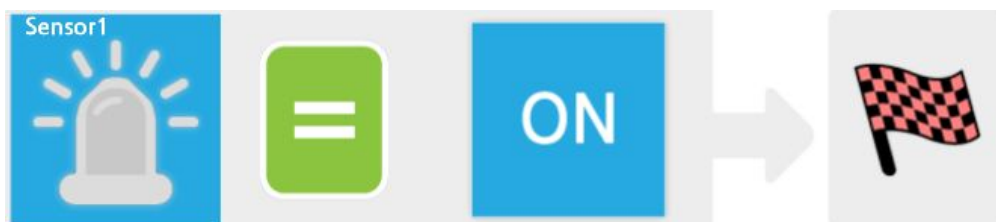
<그림 4-3> ADC 값(Analog-to-Digital Conversion data) 형식의 모니터링

<그림 4-1>의 ④는 가상 센서 영역으로 로봇의 센서 포트가 부족할 경우 가상의 센서를 추가할 수 있으며 단순한 리모컨 버튼으로도 사용될 수 있다. 사용자가 추가한 가상 센서는 로봇의 센서처럼 기초 조건을 생성하는데 사용할 수 있으며 <그림 4-4>와 같이 다양한 가상 센서를 지원한다.



<그림 4-4> 가상 센서

<그림 4-1>의 ⑤는 사용자의 프로그램 로직을 구현하기 위한 기초 조건을 생성하고 조건을 모니터하는 영역이다. 단위 조건은 <그림 4-5>와 같이 비교할 대상을 지정하고 비교연산자(comparison operator)와 비교 값을 지정한 후 조건을 상징화할 대표 깃발을 선택하면 생성되며 조건 모니터에 추가된다. 본 시스템에서 기초 조건은 깃발이라는 메타포로 표현되며 각 깃발은 사용자가 지정한 색상으로 식별한다. 기초 조건이 만족하지 않으면 깃발의 색상이 반투명하게 표현되며 만족하면 깃발의 색상이 올바르게 표현된다. 추가된 기초조건은 논리 조합을 통해 복합조건으로 표현될 수 있으며 사용자가 만든 복잡조건은 로봇을 동작시킬 특정 상황을 의미하게 된다. 기초조건이 올바르게 만들어져있는 상황이라면 <그림 4-6>과 같이 기초조건과 논리연산자(logical operator)만으로 복합조건을 생성할 수 있다.



<그림 4-5> 단위 조건 생성



<그림 4-6> 단위 조건의 조합

<그림 4-1>의 ⑥은 사용자에게 의해 생성된 로봇의 행동을 우선순위의 순서로 나열되는 영역이다. 사용자는 앞서 생성한 복합조건과 대응시킬 행동을 <그림 4-7>과 같은 타임라인 행동 편집기를 활용하여 생성할 수 있다. 타임라인 행동 편집기에서의 편집은 로봇을 제어할 수 있는 단위시간을 결정한 후 이 단위시간에 맞춰서 순차적으로 실행할 단위 명령어 블록을 나열하는 것으로 진행된다. 타임라인 행동 편집기에서 각 가로축은 로봇의 출력장치별 채널(channel)을 의미하며 세로축은 단위시간을 나타낸다. 따라서 사용자는 세로축에 맞춰서 각 출력장치의 출력 값을 지정하며 행동을 편집할 수 있다.

단위시간 : 300 ms										
	1	2	3	4	5	6	7	8	9	10
모터1	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON
모터2	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON
서보1	90°	80°	70°	80°	90°	90°	100°	110°	100°	90°
서보2	180°	170°	160°	170°	180°	180°	170°	160°	170°	180°
LED1	ON	WAIT	OFF	ON	OFF	ON	WAIT	WAIT	WAIT	OFF
LED2	OFF	OFF	WAIT	WAIT	WAIT	WAIT	WAIT	WAIT	WAIT	WAIT

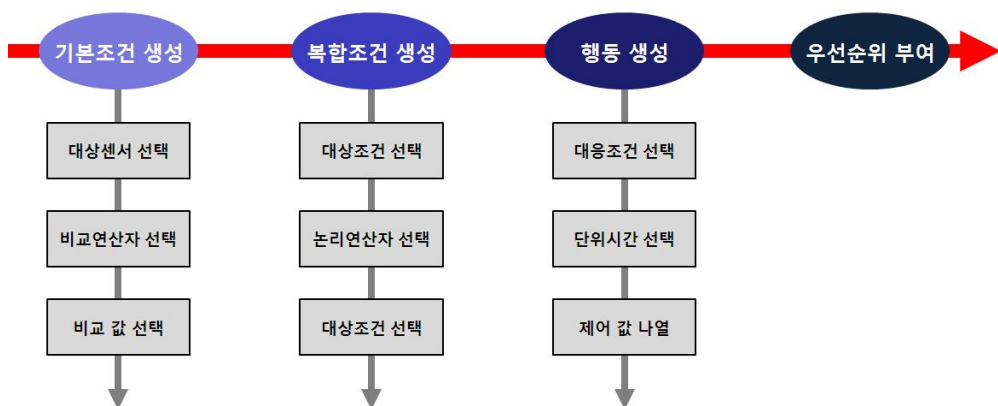
<그림 4-7> 타임라인 동작 에디터

앞서 언급한 방법으로 생성된 행동은 실행되는 동안 <그림 4-1>의 ⑦과 같은 로거에 로그(log)를 남기게 되는데 사용자는 이를 통해 자신의 프로그램이 올바르게 동작하고 있는지 확인할 수 있다.

제 3 절 구 현 결 과

사용자가 본 시스템을 활용하여 로봇 프로그램을 작성 하는 과정은 프로그램 단위인 기초조건, 복합조건, 행동을 순차적으로 생성하고 생성된 행동에 우선순위를 부여하는 과정으로 진행되며 이는 <그림 4-8>과 같다. 가장 먼저 기초조건 생성 과정을 수행하는데 이과정은 비교할 대상 센서를 선택하고 비교연산자와 비교 값을 선택하는 과정으로 진행된다. 다음으로 복합조건을 생성하는 과정을 진행하는데 이 과정에서는 앞서 생성한 기초조건 중에 논리 연산을 취할 대상 조건들을 선택하고 논리연산자를 선택하여 조합한다. 그리고 나서 앞서 생성한 복합조건 중에서 행동의 실행 조건으로 사용할 대응조건을 선택하고 타임라인의 단위시간을 결정한 후 타임라인의 시간 순으로 제어 값을 나열하여 행동을 생성한다. 마지막으로 이렇게 생성된 행동에 우선순위를 부여한다.

초급자가 앞서 설명한 프로그래밍 과정을 진행하기 위해서 사전에 숙지해야할 지식은 비교연산, 논리연산, 제어 값 이해, 우선순위 정도로 매우 낮은 수준이다. 또한 프로그램의 각 단위를 생성하는 과정은 3번의 선택만으로 수행되기 때문에 매우 간단하다. 하지만 이렇게 간단한 과정만으로 사용자가 만들고자하는 콘텐츠를 저작할 수 있는지 검증할 필요가 있다.

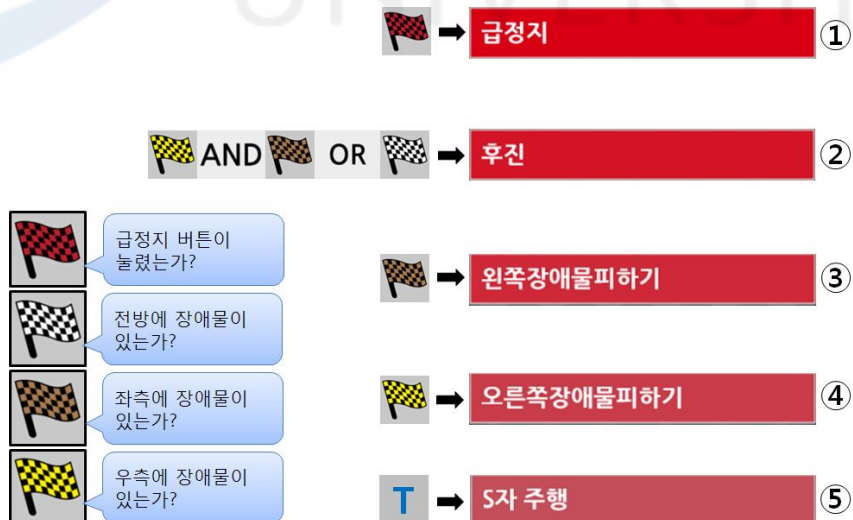


<그림 4-8> 본 논문에서 제시한 프로그래밍 과정

먼저 본 논문으로 구현된 시스템이 로봇실행에 적합한지 판단하고자 실제 교육환경에서 많이 활용하는 예제 중에 환경변화의 감지가 필요한 반응형 로봇 예제를 선정하였다. 첫 번째 예제는 이동로봇의 가장 기본적인 예제인 “어보이더(avoider)”로 특정 목적지 없이 자율적으로 주행하다가 장애물을 만나면 스스로 피하는 로봇이다. 이 예제는 로보티즈사의 STEM-kit을 활용하여 로봇을 제작한 후 본 논문으로 구현된 시스템을 활용하여 프로그래밍을 진행하였다. 로봇의 외형은 <그림 4-9>과 같으며 전방, 좌측, 우측에 장애물 감지 센서가 위치하여 해당 방향에서 장애물을 발견하면 반대방향으로 고개를 돌리며 방향을 틀어 이동한다.



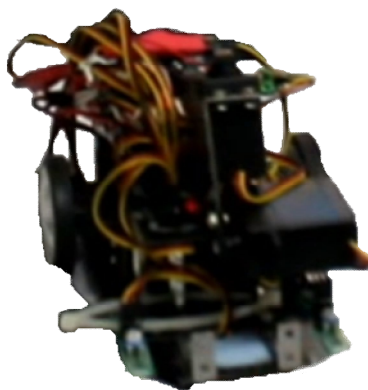
<그림 4-9> 어보이더 로봇(STEM-kit)



<그림 4-10> 본 시스템으로 구현한 어보이더 프로그램의 예

어보이더 예제는 본 시스템을 통해 <그림 4-10>와 같이 구현할 수 있었다. 어보이더 로봇의 행동은 응급한 예외상황을 대처하기 위한 행동 순으로 배치되어 가장 아래에는 기본적인 행동이 배치된다. 이와 같이 단순한 편집으로 어보이더의 프로그램을 작성할 수 있었다. 모니터를 통해 로봇의 현재 상태를 확인할 수 있었고 이를 통해 보다 효율적인 조건설계가 가능했다. 또한 가상 센서를 통해 구현한 급정지 버튼이 로봇의 자율주행 중에도 정상적으로 작동하는 것을 확인할 수 있었다.

두 번째 예제는 “라인트레이서(line-tracer)”로 바닥에 그려진 검정 선을 따라 움직이는 로봇이다. 라인트레이서는 실제로 많은 경진대회가 열리고 있으며 초등학생 및 중학생을 대상으로 하는 대부분의 교육용 로봇 제품에서 단골로 등장하는 예제이다. 이 예제는 로보로보사의 School-kit을 활용하여 로봇을 제작한 후 마찬가지로 본 논문으로 구현된 시스템을 활용하여 프로그래밍을 진행하였다. 로봇의 외형은 <그림 4-11>과 같으며 이동형 로봇의 형태로 제작하였다. 라인트레이서는 주행 중 전면 하단에 라인(line)을 감지하기 위해 배치된 두개의 센서를 통해 라인을 감지하며 라인을 벗어나지 않도록 이동방향을 조율하는 형태로 동작한다. 이 예제의 프로그램은 본 시스템을 통해 <그림 4-12>과 같이 프로그래밍할 수 있었다.



<그림 4-11> 라인트레이서(School-kit)



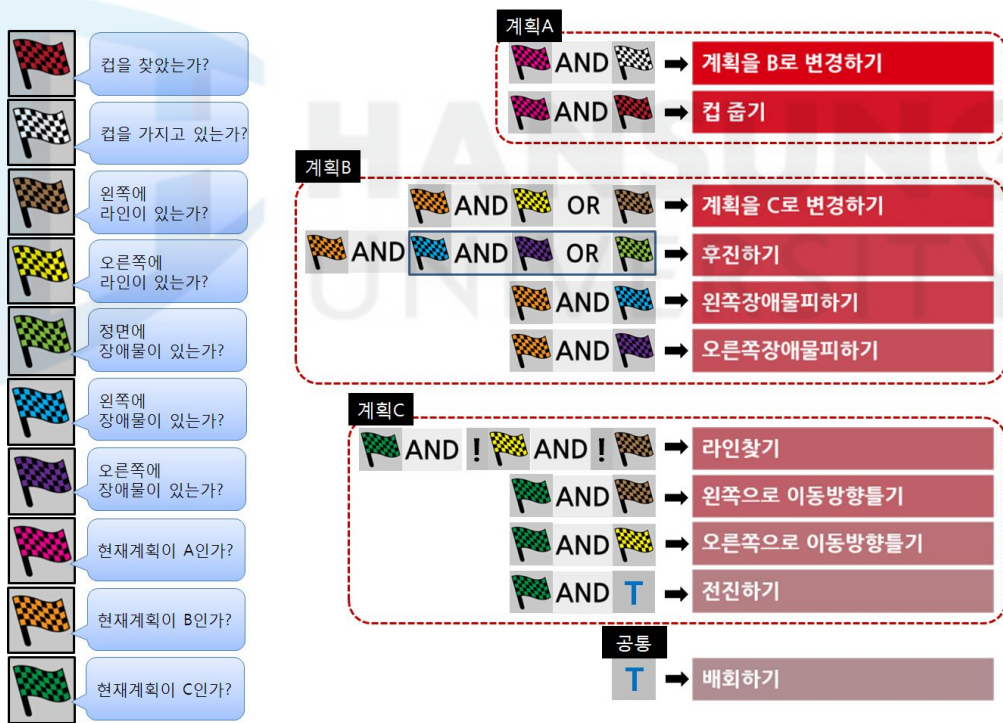
<그림 4-12> 본 시스템으로 구현한 라인트레이서 프로그램의 예

라인트레이서의 행동은 어보이더와 마찬가지로 응급한 예외상황을 대처하기 위한 행동 순으로 배치됐다. 두개의 기초조건 “왼쪽에 라인이 있는가?”와 “오른쪽에 라인이 있는가?”를 한 번씩 재활용하여 비교적 간단히 어보이더 프로그램을 작성할 수 있었고 실행시에 로봇의 이동속도가 빠르더라도 라인을 놓치지 않고 따라가는 것을 확인할 수 있었다.

앞서 두 가지의 반응형 로봇 예제를 구현해보므로 환경변화의 감지가 필요한 상황에서 본 시스템이 올바르게 동작하는 것을 확인하였다. 앞서 언급한 두 가지 예제는 환경변화에 대한 반응이 올바르게 작동하는지를 확인하기 위하여 비교적 단순한 구문을 선택한 것이었기 때문에 본 시스템에 “가장 최근에 종료된 행동”변수와 “현재계획”변수를 추가하여 절차적인 프로그램을 표현할 수 있는지는 확인하지 못하였다. 이를 검증하기 위하여 추가로 두 가지 상황에 대해 표현해보았다. 첫 번째로 “가장 최근에 종료된 행동”변수를 활용하여 로봇이 다수의 행동을 순차적으로 실행하는 상황을 표현해보았다. <그림 4-13>는 “가장 최근에 종료된 행동”을 활용하여 전진-좌회전-킥 줍기 행동을 연이어 수행하는 예제이다. “전진”, “좌회전”, “킥 줍기”는 각각 독립적인 행동이지만 선행되어야 할 행동이 있는 경우 “가장 최근에 종료된 행동”이라는 변수를 활용하면 특정 행동이 종료된 이후를 행동의 필요조건으로 사용할 수 있다. 이러한 방법으로 다수의 행동을 순차적으로 실행하는 로봇을 표현할 수 있었다.



<그림 4-13> 순차적인 행동실행의 예



<그림 4-14> 순차적인 임무수행의 예

두 번째로 “현재계획”변수를 활용하여 로봇이 다수의 임무를 순차적으로 수행해야하는 상황을 표현해보았다. 이러한 상황은 계층적 구조로 해결할 수 있는데 본 시스템에서는 “현재계획”이라는 변수를 사용하여 계층적 구조를 표현할 수 있다. <그림 4-14>은 “현재계획”이라는 변수를 활용하여 컵을 줍고 라인을 찾은 후에 라인을 따라 목적지로 돌아오는 예제이다. 이 예제는 모든 행동이 유일한 우선순위를 할당받아 경쟁하는 것처럼 보이지만 “현재계획”에 대한 조건이 AND로 복합되어있어 계층적으로 동작하게 된다. “현재계획”이 계획A라면 “현재계획이 B인가?”와 “현재계획이 C인가?”라는 조건이 불만족하기 때문에 해당 조건이 만족해야 실행되는 행동들은 절대로 실행될 수 없게 된다. 단, “컵을 가지고 있는가?”라는 조건이 만족하게 되면 현재계획이 B로 변경되어 다음 임무를 수행하게 된다. 이러한 방식으로 다수의 임무를 순차적으로 수행하는 로봇을 표현할 수 있었다.

본 논문의 목적은 초보자가 사용가능한 교육용 로봇 프로그래밍 환경을 개발하고 나아가 이미 상용화된 교육용 로봇의 기능을 향상 시키는 것이다. 본 논문은 교육용 로봇과 스마트 기기가 결합한 형태의 확장된 로봇(expanded robot)을 제안하였으며 이를 통해 사용자는 교육용 로봇에서 스마트 기기의 고급기능을 사용할 수 있게 되었다. [표 4-4]는 기존에 상용화된 교육용 로봇과 본 논문에서 제안한 방법으로 확장된 로봇의 기능을 비교한 표이다. 기존의 교육용 로봇에 본 논문에서 제안한 방법을 적용할 경우 TTS, 오디오파일 재생기능을 사용할 수 있게 되었으며 센서 및 조건에 대한 모니터링을 지원받을 수 있게 되었다. 또한 기존 로봇에서 제공하던 센서 외에 가속도센서, 자이로센서, 지자기센서를 사용할 수 있게 되었다. 하지만 장치 모니터링 기능의 경우, 디지털 서보모터의 필드 값을 읽어야하기 때문에 아날로그 서보모터를 사용하는 제품에서는 제한된다.

[표 4-4] 교육용 로봇의 본 논문 적용 전/후 기능비교

기능 항목	본 논문 적용 전 (STEM-kit)	본 논문 적용 후 (STEM-kit)	본 논문 적용 전 (School-kit)	본 논문 적용 후 (School-kit)
TTS (Text to Speech)	X	O	X	O
오디오 파일재생	X	O	X	O
센서 모니터링	△ (별도 프로그램 제공)	O	X	O
조건 모니터링	X	O	X	O
장치 모니터링	△ (별도 프로그램 제공)	O	X	X
접촉 센서	O	O	O	O
적외선 센서	O	O	O	O
가속도 센서	△ (추가 구입하여 사용 가능)	O	X	O
자이로 센서	△ (추가 구입하여 사용 가능)	O	X	O
지자기 센서	X	O	X	O

제 5 장 결 론

본 논문에서는 교육용 로봇에 스마트 기기를 결합한 형태의 확장된 로봇을 제안하여 교육용 로봇에 고급기능을 지원하였고 초보자도 사용가능한 직관적인 프로그래밍 환경과 프로그래밍 방식을 구현하였다. 기존에 로봇의 제어기가 담당하던 두뇌 역할을 보다 월등한 성능을 가진 스마트 기기에 위임하였고 로봇의 제어기는 I/O보드의 역할만 담당하게 하여 제어기에 사용자가 작성한 프로그램을 이식할 필요가 없어졌다. 따라서 사용자에게 다양한 제품의 교육용 로봇을 하나의 프로그래밍 환경에서 편집할 수 있도록 지원하였으며 프로그램 편집 과정을 단축시킬 수 있게 되었다.

또한 초급자가 프로그램을 보다 쉽고 체계적으로 설계할 수 있도록 프로그램의 설계 단위를 기초조건, 복합조건, 행동으로 구성하였다. 이는 단순하고 직관적인 T-R 시퀀스를 기반으로 동작하며, “가장 최근에 종료된 행동”과 “현재계획”이라는 변수를 추가로 도입하여 일부 절차적인 프로그램도 구현할 수 있게 하였다. 그리고 스마트 기기의 디스플레이장치를 로봇 전용 프로그래밍 및 모니터링 도구로 사용하여 사용자가 프로그램을 작성하거나 테스트하는 과정 중에 로봇의 장치, 센서, 조건을 즉시 확인할 수 있게 하였다. 모니터링 도구가 개발됨에 따라 로봇 디버깅이 서툰 초보자가 로봇의 문제를 직관적으로 해결할 수 있게 되었다.

끝으로 기존의 교육용 로봇이 갖고 있던 기능적 제한을 교육용 로봇과 스마트 기기를 결합하므로 해결할 수 있었다. 교육용 로봇에 스마트 기기의 다양한 센서와 부가적인 고급기능들이 더해져 로봇으로 표현할 수 있는 가짓수가 늘어났다.

교육용 로봇의 한계점을 보완한 본 시스템을 통해 기존에 로봇을 어려워 하던 학생들도 보다 쉽게 로봇을 활용하여 자신의 생각을 표현할 수 있을 것으로 보인다. 본 시스템을 활용한 교육으로 로봇을 동작시키면서 많은 학생들이 학습에 흥미를 가질 것으로 기대하며, 짧은 시간에도 사용법 습득이 가능해졌기 때문에 수학이나 과학 및 기술 교육 목적에 더욱 충실하게 활용될 수 있을 것으로 기대된다.

현재는 제품마다 통신 속도나 유효 패킷 크기에 한계가 있어 부품의 수를 제한하였지만 앞으로 최적의 통신 방법을 연구한다면 많은 부품을 사용하는 다관절 로봇의 프로그래밍도 가능할 것으로 보인다. 나아가 블루투스 외에 와이파이(Wi-Fi)와 같은 다른 무선 통신망을 지원하도록 연구한다면 교육용 로봇 외에 보다 더 다양한 로봇을 지원할 수 있을 것으로 보인다.



【참고문헌】

1. 국내문헌

- 김경현, 손충기, 정미경, 백제은, 김진숙, 장시준, 류은영, 김영애(2009), 『교구로봇활용수업의 효과분석 연구』, 한국교육학술정보원, 연구보고 CR 2009-27
- 김진성, 조혜경(2012), “교육용 로봇의 모니터링 및 컨트롤을 위한 스마트 기기 기반 저작 환경 개발”, 대한전기학회, 『대한전기학회 학술대회 논문집』, pp.1365-1366
- 박동희(2009), “이토이 기반의 교육용 로봇 제어 프로그래밍을 위한 IDE 개발”, 석사학위논문, 고려대학교 대학원
- 박웅식, 문성환(2009), “초등학생의 논리적 사고력 신장을 위한 로봇 교육 프로그램 개발 및 적용”, 석사학위논문, 서울교육대학교 교육대학원
- 송정범, 이태욱(2008), “피코 크리켓(Pico Cricket)을 활용한 프로그래밍 교육이 문제해결력에 미치는 효과”, 『實科教育研究』, Vol.14, No.4, pp.243-258
- _____ (2011), “교육용 로봇을 활용한 STEM 통합교육이 학업성취, 교과태도에 미치는 효과”, 『한국정보교육학회논문지』, Vol.15, No.1, pp.11-22
- 이강봉(2011년 8월 2일), “로봇 교육이 어린이 흥미를 높인다.”, 『사이언스타임즈』
- 전윤주(2010), “실과에서 로봇 프로그래밍 학습이 초등학생의 창의력에 미치는 효과”, 석사학위논문, 한국교원대학교 대학원
- 조혜경, 김미량, 박광현, 신승용, 박정호, 배상용, 정성희, 김진성, 채경화, 김영애(2011), 『로봇 소양 교육과정 개발 연구』, 한국교육학술정보원

보원, 연구보고 KR 2011-5

최대회, 조준호, 김익현, 박정찬, 정성훈(2011), “위게임에서 가상군의 자율적 행위를 위한 자동계획 기법”, 『한국컴퓨터정보학회논문지』, Vol.16, No.9, pp.11-18

최정호(2012), “로보이드 프레임워크를 사용한 HRI 로봇의 즉흥 연주를 위한 동적 모션 생성 로보이드 컴포넌트의 개발”, 석사학위논문, 한양대학교 대학원

한정혜(2011년 6월 30일), “선진국의 연구 사례를 통한 창의적 미래학교의 로봇기반학습(r-Learning) 정책 방향”, 교육정책네트워크 정보센터, 교육정책포럼

홍기천(2009), “레고 NXT 로봇을 활용한 예비교사의 프로그래밍 언어 수업 방안”, 『한국정보교육학회논문지』 Vol.13, No.1, pp.71-78

Roboid Studio(2010), <http://www.roboidstudio.org>

Robotis support(2010), <http://support.robotis.com>

2. 국외문헌

- Nilsson, Nils J.(1994), “Teleo-Reactive Programs for Agent Control”,
Journal of Artificial Intelligence Research, Vol.1, pp.139-158
- Coffey, S., Clark, K.L.(2006), “A Hybrid, Teleo-Reactive Architecture
for Robot Control”, In: *Proceedings of the Second International
Workshop on Multi-Agent Robotic Systems*, MARS
- Brooks, R.(1986), “A Robust Layered Control System for a Mobile
Robot”, *IEEE Journal of Robotics and Automation*, Vol.2, No.1,
pp.14-23, March
- Brooks, R.(1991), “Intelligence Without Reason”, In: *Proceedings of the
12th International Joint Conference on Artificial Intelligence*,
pp.569-595
- Murphy, Robin R.(2000), *Introduction to AI Robotics*, Cambridge, MA,
USA: The MIT Press
- STEM centric(2012), <http://www.stemcentric.com>
- NXT Programs(2007), <http://www.nxtprograms.com>
- Pico Cricket(2006), <http://www.picocricket.com>
- The free dictionary(n.d.), <http://encyclopedia2.thefreedictionary.com>,
“visual programming language”
- Wikipedia(2012), <http://en.wikipedia.org>, “visual programming language”
- Robot Operating System(2009), <http://www.ros.org>, “sense plan act”

ABSTRACT

Designing a Smart Device Based Integrated Development & Control Environment for Supporting Non-Expert Robot Creators

Kim, Jin Sung

Major in Information &

Communications Engineering,

Dept. of Information &

Communications Engineering,

Graduate School, Hansung University

In order to efficiently enhance the functionality and the utilizability of cost-effective educational robot kits and their related contents, this thesis suggests a smart device-based integrated development and control environment for robots. First, in order to empower useful and advanced functions to robots, smart devices are employed as an expanded robot controller. Then the touchscreen can be used as an interface unit and advanced features of smart devices, such as voice recognition and TTS, can be available as if they were a part of robot's embedded functions.

In an aspect of utilizability, followings are considered: The largest population of robot users are young children who cannot deal with programming languages but are digital natives enjoying smart devices. Hardware dependancy must be minimized for ease of contents sharing

and spreading. Consequently, the following requirements on the system can be derived: First, the programming, debugging and execution process must be very intuitive and simple. Second, the system must support for smart devices with different OS platforms. Third, the system must be able to be connected with various robot products not restricted to a specific robot controller.

To meet the requirements from above two aspects, first, this thesis suggests an intuitive user interface and an appropriate programming method for non-experts based on the analysis of previous works on robot control architectures and already existing robot programming interfaces. Next, a type of zero-client communication and control protocol is designed so that the smart device can monitor and control the robot in real-time. Especially, the author focuses on how to employ various commercialized robot controllers without major modification on their firmwares. Finally, this thesis presents the integrated overall system architecture which enables unified operation of the robot programming and debugging environment and the real-time control environment together within a single application.

As cost-effective robot kits become advanced and easier to use owing to the suggested system, it is expected that they can be a creation tool for people who want to realize and share their creative ideas, not to mention their usefulness as tools for educational programs in mathematics, science and technology.

【keywords】 Educational Robots, Smart Robots, Smart Devices, Reactive Robots, Educational Programming Language, Visual Programming Language, Behavior-Based Robotics, Teleo-Reactive Sequences, Robot Integrated Development & Control Environment