자율무인잠수정을 위한 임무 언어 개발*

김방현○ 이필엽 심형원 전봉환 이판묵
한국해양연구원
{bhkim, powermanz, hwshim, bhjeon, pmlee}@moeri.re.kr

Development of Mission Language for Autonomous Underwater Vehicle

Banghyun Kim○ Fillyoub Lee Hyungwon Sim Bonghuan Jun Panmook Lee
Korea Ocean Research & Development Institute

요 약

자율무인잠수정은 탐사 목적에 따라 다양한 임무를 수행해야 하며, 임무에 따라 자율무인잠수정 행동의 유형과 순서는 달라질 수 있다. 그러나 대부분의 자율무인잠수정은 한정된 임무에 대하여 프로그램 내부에 고정된 행동 유형으로 동작하며, 다른 유형의 임무를 수행해야 할 경우에는 프로그램을 수정해야 하는 문제점이 있다. 따라서 본 연구에서는 자율무인잠수정이 수행할 수 있는 다양한 임무를 명시할 수 있는 임무 언어를 개발하였다. 이 임무 언어는 명령어의 실행 순서를 제어할 수 있는 제어문과 자율무인잠수정의 행동을 지정하거나 자율무인잠수정의 상태를 입출력 할 수 있는 명령어, 그리고 변수 정의를 제공하기 때문에, 사용자가 자율무인잠수정의 임무를 자유롭게 표현하는 것이 가능하다. 임무 언어로 작성된 임무 파일은 전용 어셈블러에 의해 이진 형식의 실행이미지로 변환된 후에, 자율무인잠수정 내장 소프트웨어 내부의 가상기계 기억장치에 적재되어 실행된다. 실행이미지를 가상기계에서 해석하고 실행하는데 필요한 시스템의 자원을 최소화하기 위하여 임무 언어는 자율무인잠수정의 임무를 표현하기 위한 필수적인 부분만을 고려하여 설계되었으며, 문법은 ARM v5 어셈블리와 유사한 형태이다. 개발된 임무 언어는 한국해양연구원에서 개발한 이심이100 자율무인잠수정에 적용되었으며, 이후 개발할 6,000m급의 이심이6000 자율무인잠수정에도 사용될 예정이다.

1. 서론

무인잠수정(UUV: unmanned underwater vehicle)은 사람이 탑승하지 않기 때문에 유인잠수정에 비해 안전성 문제에서 자유로워 1990년대 이후 해저탐사의 주역이 되고 있다. 무인잠수정은 사용자와 연결 방식에 따라서 크게 원격제어 무인잠수정(remotely operated vehicle: 이하 ROV라 함)과 자율무인잠수정(autonomous underwater vehicle: 이하 AUV라 함)으로 분류된다. ROV는 전력 케이블 및 데이터 통신 케이블이 수면 또는 지상에 있는 장비와 연결되어 있어서 사용자가 원격으로 제어하여 수중에서 임무(mission)를 수행할 수 있는 수중탐사장비이다. 반면에 AUV는 케이블이 연결되어 있지 않은 독립 탐사장비로, 배터리를 전원으로 사용하고 탐사 임무는 AUV가 수중에 진수되기 전에 AUV에 탑재되는 내장 소프트웨어(embedded software)로 작성되어야 한다.

AUV의 임무는 보통 AUV 행동의 유형과 순서로 표현된다. 군사용 또는 상업용으로 개발된 AUV는 임무가 고정되어 있기 때문에, 내장 소프트웨어 내부에 임무가 고정적으로 명시되어 있다. 그러나 이러한 형태의 내장 소프트웨어에서는 임무가 변경될 경우에는 소프트웨어를 다시 작성해야 하는 문제점이 있다. 따라서 소프트웨어 내부에 임무를 명시하지 않고, 내장 소프트웨어 외부에 독립적으로 임무를 표현할 수 있는 임무 언어(mission language)가 필요하다. 특히 연구 목적의 AUV의 경우에는 다양한 임무를 수행해야 하는 경우가 많기 때문에 임무 언어가 유용하게 사용될 수 있다.

본 연구에서는 한국해양연구원에서 해양환경 및 자원을 탐사하기 위해 개발한 이심이(ISiMI: integrated submersible for intelligent mission implementation)[1] AUV의 다양한 임무를 간결하게 명시하고 수행할 수 있는 TML(tiny mission language)을 개발하였다. TML은 명령어의 실행 순서를 제어할 수 있는 제어문과 AUV의 행동을 지정할 수 있는 명령어, 그리고 변수 정의를 제공하기 때문에 사용자가 AUV의 임무를 자유롭게 표현하는 것이 가능하다. 사용자가 텍스트 형태로 작성한 TML 임무 파일은 TML 어셈블러에 의해 이진 형식의 실행이미지로 변환된 후에, AUV 내장 소프트웨어 내부의 TML 가상기계(virtual machine) 기억장치에 적재되어 실행된다. TML은 실행이미지를 TML 가상기계에서 해석하고 실행하는데 필요한 시스템의 자원을 최소화하

* 본 연구는 국토해양부에서 지원한 “차세대 심해용 무인잠수정 개발” 사업의 일환으로 수행되었음.

```
{
 Start                    : Waypoint_Navigation_Order
 Destination_Latitude     : 40.0000 Degrees
 Destination_Longitude    : -74.0000 Degrees
 Transit_Mode             : Steer_to_Point
 Transit_Depth            : 10 Meters
 Transit_Speed_In_Water   : 5.0 Knots
 Use_SSS                  : False

 Start                    : Survey_Order
 TopLeft_Latitude         : 40.0000 Degrees
 TopLeft_Longitude        : -74.0000 Degrees
 TopRight_Latitude        : 40.0000 Degrees
 TopRight_Longitude       : -73.9500 Degrees
 BottomLeft_Latitude      : 39.9500 Degrees
 BottomLeft_Longitude     : -74.0000 Degrees
 BottomRight_Latitude     : -39.9500 Degrees
 BottomRight_Longitude    : -73.9500 Degrees
 Survey_Depth             : 30 Meters
 Survey_Speed             : 3.5 Knots
 StartPoint               : TopLeft
}
```

그림 1. MPL의 예

기 위하여 AUV의 임무를 표현하기 위한 필수적인 부분만을 고려하여 설계되었으며, 문법은 ARM v5 어셈블리와 유사한 형태이다.

행동 기반 혼합형 제어 구조(behavior-based hybrid control architecture)로 이루어진 이심이 내장 소프트웨어에서 TML은 임무 파일의 실행 결과로 발생되는 행동의 입출력을 행동 조정기(behavior coordinator)로 전달하고, 이심이의 경로를 조정하는 것이다. 실제 이심이의 동작은 활성화된 행동들을 관리하는 행동 중재기에 의해 결정되어 수행된다.

논문의 구성은 2장에서 AUV의 임무 언어에 대하여 소개하고, 3장에서는 이심이 AUV의 내장 소프트웨어에 관하여 살펴본다. 4장에서는 이심이 AUV의 임무 언어인 TML에 관하여 설명하고, 마지막으로 5장에서는 결론을 맺는다.

2. AUV 임무 언어

아직까지는 AUV를 임무 수행을 위한 임무 언어에 대한 연구가 미비하다. 현재 진행된 대표적인 연구로는 PSU/ARL(Pennsylvania state university / applied research laboratory)의 MPL(mission programming language)[2], NUWC(naval undersea warfare center)의 CCL(common control language)[3], 그리고 NPS(naval postgraduate school)의 AVCL(autonomous vehicle control language)[4]이 있다.

MPL은 PSU에서 개발한 Seahorse[5] AUV를 위한 임무 언어로서 파싱을 위해 Lex와 Yacc를 이용하는 컴파일러 형태의 언어이다. MPL에서는 제어문과 변수를 사용할 수 있으며, 다단계로 정의된 임무 파일은 하이브리드 임무 제어기(hybrid mission controller)에 의하여 수

```
task mp3
 {
 (:= (init)
     (void)
     (= k 1)
     (= b 1)
     (= n 1)
     (= t 1)
     (= operator_set SEQUENCE))
 (:= (subtasks)
     (void)
 (= Complex box)
 (= Complex wc))
 }
 task box
 {
 (:= (init)
     (void)
     (= k 5)
     (= b 1)
     (= n 5)
     (= t 1)
     (= operator_set SEQUENCE))
 (:= (subtasks)
     (void)
 (= Maneuver CCL_MANEUVER_GOTO CCL_SURF_LOC_LAT_LNG 41.555933
    71.339067 CCL_SURFACE 0 CCL_CONSTANT_DEPTH_PATH CCL_SPEED_MAX 0 )
 (= Maneuver CCL_MANEUVER_GOTO CCL_SURF_LOC_LAT_LNG 41.557000
    71.339067 CCL_SURFACE 0 CCL_CONSTANT_DEPTH_PATH CCL_SPEED_MAX 0 )
 (= Maneuver CCL_MANEUVER_GOTO CCL_SURF_LOC_LAT_LNG 41.557000
    71.330000 CCL_SURFACE 0 CCL_CONSTANT_DEPTH_PATH CCL_SPEED_MAX 0 )
 (= Maneuver CCL_MANEUVER_GOTO CCL_SURF_LOC_LAT_LNG 41.555933
    71.330000 CCL_SURFACE 0 CCL_CONSTANT_DEPTH_PATH CCL_SPEED_MAX 0 )
 (= Maneuver CCL_MANEUVER_GOTO CCL_SURF_LOC_LAT_LNG 41.555933
    71.339067 CCL_SURFACE 0 CCL_CONSTANT_DEPTH_PATH CCL_SPEED_MAX 0 ))
 }
 task wc
 {
 (:= (init)
     (void)
     (= k 1)
     (= b 1)
     (= n 1)
     (= t 1)
     (= operator_set SEQUENCE))
 (:= (subtasks)
     (void)
 (= Maneuver CCL_MANEUVER_MAINTAIN_POSITION CCL_MAINTPOS_LAT_LNG
    41.555933 71.339067 CCL_FALSE 0 CCL_TIME_RELATIVE 10 CCL_TIME_MIN ))
 }
```

그림 2. CCL 임무파일의 예

행된다. 경로를 관리하기 위하여 MPL은 직렬 순서 큐(sequential order queue)와 인터럽트 기반의 순서 큐(interrupt- driven order queue)를 사용한다. MPL 시스템은 구현이 복잡하고 사용자가 임무 파일을 작성하기 어렵다는 단점이 있다. 그림 1은 MPL로 작성된 임무의 일부분을 보여준다.

CCL은 통신과 작업 조정을 수행하는 자율 에이전트에 사용되는 간결하고 견고한 언어이다. CCL로 작성된 임무 파일은 컴파일러에 의해 중간 단계의 바이트 형으로 컴파일 된 후에 AUV로 다운로드 된다. CCL의 문법은 LISP 언어 및 C 언어와 유사하다. CCL은 상호 협조하는 여러 대의 AUV 임무를 지원하지만, 제어문은 지원하지 않는다. CCL은 AUSI(autonomous undersea systems institute)에서 개발한 SAUV(solar-powered AUV)[6]와 NUWC(naval undersea warfare center)에서 개발한 MARV(mid-sized autonomous reconfigurable vehicle)[7]에 사용되었다. 그림 2는 CCL로 작성된 임무 파일을 보여준다.

AVCL은 XML(extended markup language) 기반의 문맥 자유 문법(context-free grammar: CFG)를 사용하는 인터프리터 언어로 많은 종류의 AUV를 지원한다. AVCL이 사용될 수 있는 AUV로는 NPS에서 개발한

```
<UUVCommandScript/>
    <Position>
        <XYPosition x="0.0" y="0.0"/>
        <Depth value="0.0"/>
    </Position>
    <Thrusters value="false"/>
    <Waypoint>
        <XYPosition x="100.0" y="100.0"/>
        <Depth value="45"/>
        <SetPropeller>
            <AllPropellers value="100.0"/>
        </SetPropeller>
    </Waypoint>
    <Waypoint>
        <XYPosition x="500.0" y="100.0"/>
        <!-- use prior depth! -->
    </Waypoint>
    <Waypoint>
        <XYPosition x="500.0" y="200.0"/>
        <Depth value="25"/>
    </Waypoint>
    <Waypoint>
        <XYPosition x="0.0" y="0.0"/>
    </Waypoint>
    <MakeDepth value="0.0"/>
    <Quit/>
</UUVCommandScript>
```

그림 3. AVCL의 예

ARIES(acoustic radio interactive exploratory server)[8] AUV 및 Pheonix AUV[9], PSU의 Seahorse AUV[5], WHOI(Woods Hole oceanographic institution)에서 개발한 REMUS(remote environmental monitoring units)[10] AUV가 있다. AVCL은 XML을 이용하기 때문에 제어문을 지원하지 않으며, 실행속도가 느린 문제점이 있다. 그림 3은 AVCL로 작성된 임무 파일의 예이다.

지금까지 소개한 임무 언어에서 MPL만이 일반적인 컴퓨터 언어가 지원하는 제어문과 변수를 지원한다. 그러나 MPL은 많은 기능을 제공하기 위한 복잡한 구조 때문에 구현하기가 쉽지 않을 뿐만 아니라 사용자가 MPL로 임무를 작성하기 어렵다. 반면에 본 연구에서 개발한 TML은 제어문과 변수를 지원하면서도 구현이 쉽우며, 가상기계 위에서 코드화된 실행이미지를 실행시키기 때문에 인터프리터 형태의 언어에 비하여 실행 속도가 빠르다. 또한 TML은 간결한 형태의 어셈블리 문법과 소수의 명령어를 제공하기 때문에 사용자가 TML 임무 파일을 쉽게 작성할 수 있다.

3. 이심이 내장 소프트웨어 구조

AUV 내장 소프트웨어의 자율 제어 기술은 AUV의 활용에 있어 운용자의 개입을 최소화하는 한편, 임무 성공률 및 반복성, 그리고 회수율을 높이는 데 필수적인 기술이다. 또한, 이를 통해 AUV의 효율성을 높일 있으므로 한 번의 진수로 장시간동안 여러 가지 임무 수행이 가능해 진다. 그러나 자율 제어 기술은 필수 기술 중 가

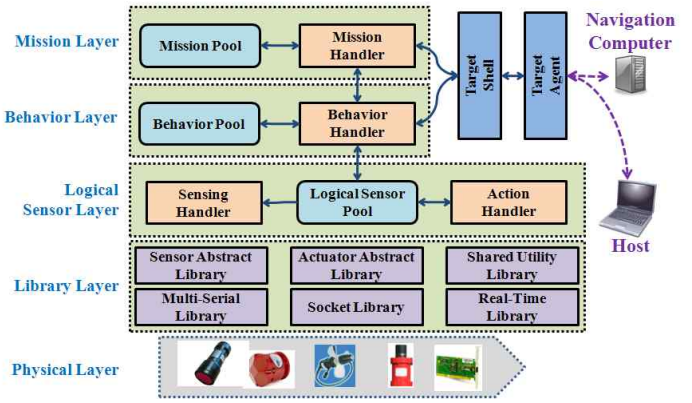

그림 4. 이심이 내장 소프트웨어 구조

장 개발이 더딘 기술 중의 하나로 분류된다[11].

자율 제어 기술 개발과 관련된 연구는 대부분 AUV의 제어 시스템의 구조(control architecture)와 연관되어 진행되어 왔다. 제어 구조는 일반적으로 제어 입력 산출 방법에 따라 숙고형(deliberative)과 반응형(reactive) 혹은 행동 기반(behavior-based) 제어 구조로 나뉘며, 형태에 따라 직렬형(hierarchical)과 병렬형(parallel) 구조, 그리고 혼합형(hybrid) 구조로 나눌 수 있다. 일반적으로 숙고형의 제어 구조는 직렬형 구조에, 반응형의 제어 구조는 병렬형 구조에 적합하다고 알려져 있다[12]. 그러나 개별 구조가 갖는 단점으로 인해 현재는 혼합형 구조가 많이 사용되는 추세이며, 부정확한 수중 환경에서 임무 성공률을 높이기 위하여 학습 알고리즘이 적용되고 있다[13]. 혼합형 제어 구조는 대부분 임무 계획층(mission planning layer), 임무 수행층(mission execution layer), 하드웨어 계층(hardware layer)의 세 개의 계층으로 이루어지며 이 중 임무 수행층은 주로 반응형 제어 구조, 특히 행동 기반 제어 기법이 많이 사용된다[14].

이심이 내장 소프트웨어는 행동 기반 혼합형 제어 구조로 이루어져 있으며, 그림 4와 같이 임무 계층(mission layer), 행동 계층(behavior layer), 논리 센서 계층(logical sensor layer), 라이브러리 계층(library layer)으로 구성된다.

임무 계층은 임무 파일의 실행이미지를 해석하여 실행하며, 행동 계층은 임무 계층의 출력으로 만들어진 행동 풀(behavior pool)에서 AUV의 행동을 적절하게 선택하여 수행한다. 논리 센서 계층은 센서로부터의 입력 데이터ma와 구동기(actuator)로 출력되는 데이터를 논리 센서 풀(logical sensor pool)을 이용하여 관리하며, 기존의 센서 데이터를 가공하여 논리적인 센서 데이터를 생성하기도 한다. 라이브러리 계층은 센서나 구동기 하드웨어와 인터페이스를 담당하며, 직렬 통신이나 이더넷 통신

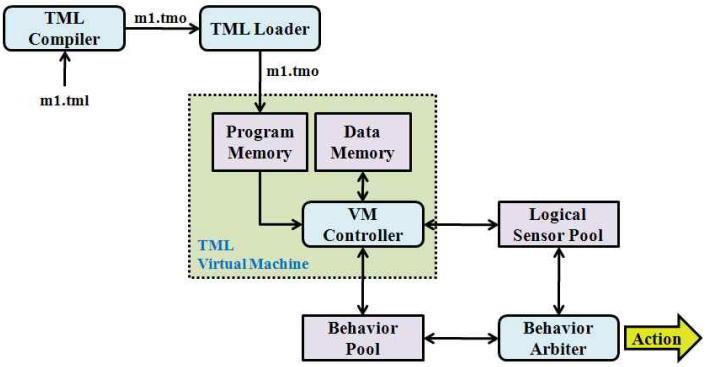

그림 5. TML 임무파일의 처리 시스템

을 수행한다. 응용프로그램 수준에서 실시간 처리를 지원하는 프로세스의 선점(preemption) 기능도 라이브러리 계층에 포함되어 있다. 이심이의 구동기는 수평 방향 운동을 제어하는 방향타(rudder)와 수직 방향 운동을 제어하는 승강타(stern), 그리고 프로펠러 추진기(thruster)가 있다.

타겟 쉘과 타겟 에이전트는 외부 시스템과의 인터페이스 역할을 수행하는데, 이심이 내부의 항법컴퓨터나 이심이의 원격 모니터링 및 제어를 담당하는 호스트 컴퓨터와 연결된다. 항법컴퓨터와는 이더넷 통신을 통하여 연결되며, 호스트 컴퓨터와는 수중 음향 통신을 통하여 연결된다. 이들 사이의 통신 메시지는 자체적으로 개발된 프로토콜인 ACP(AUV communication protocol)[15]를 이용한다. 타겟 쉘은 호스트 컴퓨터로부터 입력된 사용자의 명령을 CLI(command line interface) 기반으로 처리하며, 타겟 에이전트는 ACP에 따라 통신 메시지를 부호화하거나 복호화한다.

임무 계층의 임무 핸들러(mission handler)는 임무 언어를 처리하기 위한 TML 어셈블러와 TML 가상기계를 포함하고 있다.

4. TML(tiny mission language)

TML은 AUV의 임무를 정의하기 위한 언어로 한국해양연구원에서 개발한 자율무인잠수정인 이심이를 위하여 설계되었다. TML 명령어의 수는 이심이의 임무를 표현할 수 있는 필수적인 기능만을 지원할 수 있도록 최소화하였다. TML은 명령어의 실행 순서를 제어할 수 있는 제어문과 이심이의 행동과 경로를 지정할 수 있는 명령어, 그리고 변수 정의를 제공하기 때문에 사용자가 이심이의 임무를 자유롭게 표현하는 것이 가능하다. TML 변수는 32비트 크기로 16개까지 지정할 수 있으며, 가상기계 내부의 r0부터 r15까지의 레지스터에 저장된다. TML을 처리하기 위한 시스템은 그림 4의 임무 계층에 포함

표 1. TML 명령어

Op-code	Pseudo-code	Parameter	Operation
00000	ADC	reg1 reg2	reg1 = reg1 + reg2 + carry
00001	ADD	reg1 reg2	reg1 = reg1 + reg2
00010	AND	reg1 reg2	reg1 = reg1 & reg2
00011	BAC	BhID, cond	Add behavior with condition
00100	BHA	BhID, para	Add behavior
00101	BHD	BhID	Delete behavior
00110	CLL	label	PSH PC, PC = label
00111	CMP	reg1 reg2	reg1 - reg2
01000	DIV	reg1 reg2	reg1 = reg1 / reg2
01001	EOR	reg1 reg2	reg1 = reg1 ^ reg2
01010	JMP	label	PC = label
01011	LDI	reg1 val	reg1 = val
01100	LDS	reg1	reg1 = SR(status register)
01101	LSR	reg1 LS_ID	reg1 = LS[ID]
01110	LSW	reg1 LS_ID	LS[ID] = reg1
01111	MCH	mode	0:Emergency, 1:Manual, 2:Auto
10000	MOD	reg1 reg2	reg1 = reg1 % reg2
10001	MOV	reg1 reg2	reg1 = reg2
10010	MUL	reg1 reg2	reg1 = reg1 * reg2
10011	ORR	reg1 reg2	reg1 = reg1 | reg2
10100	POP	reg1	reg1 = STACK[SP], ++SP
10101	PSH	reg1	--SP, STACK[SP] = reg1
10110	RET		POP PC
10111	SBC	reg1 reg2	reg1 = reg1 - reg2 - carry
11000	STS	reg1	SR = reg1
11001	SUB	reg1 reg2	reg1 = reg1 - reg2
11010	SWP	reg1 reg2	reg1 ↔ reg2
11011	WPA	x y z	Add waypoint
11100	WPD		Delete current waypoint
~11111	Reserved		

되며, TML로 작성된 임무 파일은 그림 5와 같은 형태로 처리된다.

사용자가 텍스트 형태로 작성한 TML 임무 소스파일(.tml)은 TML 어셈블러에 의해 이진 형식의 임무 목적파일(.tmo)로 변환된다. 임무 목적파일은 TML 로더에 의해 TML 가상기계의 프로그램 기억장치에 적재된 후에, 가상기계의 PC(program counter)에 따라 순서대로 실행된다. 임무 목적파일의 실행 결과로 발생되는 행동을 인수와 함께 행동 풀에 추가하거나 등록된 행동을 삭제한다.

TML 가상기계는 CPU에 해당하는 가상기계 처리기(virtual machine controller)와 임무 목적파일이 적재되는 프로그램 기억장치, 그리고 16개의 32비트 레지스터와 임의 크기의 스택을 포함하는 데이터 기억장치로 구성된다. 스택의 크기는 임무 파일에서 지정할 수 있으며, 최대 크기는 64MB까지 지정이 가능하다.

TML 문법은 ARM v5 어셈블리와 유사하다. TML 명령어는 컴파일 후에 5비트의 명령어 코드(opcode)와 3비

표 2. TML의 조건코드

Code	Pseudo-code	Condition
000	EQ	equal
001	NE	not equal
010	GT	greater than
011	GE	greater than or equal
100	LT	less than
101	LE	less than or equal
110	CS	carry set
111		always

트의 조건코드(conditional code)로 변환되며, 오퍼랜드(operand)의 크기는 명령어의 종류에 따라 달라진다. 대부분의 오퍼랜드 크기는 16비트이기 때문에 BAC, BHA, LDI, WPA 명령어를 제외한 나머지 명령어들은 컴파일 후에 16비트의 이진 형식으로 변환된다. 표 1은 TML에서 정의된 명령어들을 보여준다. BAC, BHA, BHD 명령어는 행동 풀에 행동을 추가하거나 삭제하는 명령어이고, WPA와 WPD 명령어는 상대좌표로 경로를 추가하거나 삭제하는 명령어이다. MCH 명령어는 AUV의 모드를 변경하는 명령어이고, LSR과 LSW 명령어는 논리 센서 풀의 값을 읽거나 쓸 수 있는 명령어이다. POP과 PSH 명령어는 스택 연산을 수행하는 명령어이고, CLL과 RET 명령어는 함수 호출을 위해 사용되는 명령어이며, JMP 명령어는 TML 실행 순서를 변경할 수 있는 제어 명령어이다.

TML은 ARM v5와 유사한 조건 검사 방법을 제공한다. 표 1의 명령어 뒤에 접미어 형태로 표 2의 조건코드가 붙을 수 있으며, 명령어의 해석 전에 이 조건코드를 먼저 검사한다. 만약 조건코드의 결과 값이 거짓이라면 이 명령어는 수행하지 않고, PC 값은 다음 명령어를 가르키게 된다. 예를 들어, "ADDEQ r1 r2"의 경우에 이전 연산 결과가 0이라면 r1에 r2의 값을 더하여 저장하고, 그렇지 않다면 명령어를 실행하지 않는다.

TML은 환경변수를 설정하기 위하여 표 3과 같은 사용할 수 있다. '$base' 지시어는 임무의 기준 위치를 설정하며, '$home' 지시어는 모선의 위치를 설정한다. '$include' 지시어는 선언 위치에 지정된 외부 임무 파일을 삽입하며, '$stack' 지시어는 스택의 크기를 바이트 단위로 설정한다. 이 외에 TML은 JMP나 CLL 명령어의 목적지를 나타내기 위하여 레이블 지시어를 사용할 수 있다.

그림 6은 잔디 깍기 형태의 탐사 임무를 TML로 작성한 예제로, 심도 10m와 속도 1㎳를 유지하며 100m x 100m 지역을 'ㄹ'자 모양으로 3회 반복하여 운항하고 처음 위치로 되돌아오는 임무이다. BHA 명령어 뒤의 인자는 이심이 내장 소프트웨어에 정의된 이심이의 행동 이름을 나타내며, WPA는 경로를 상대좌표로 설정한 것을 의미한다. 'LSR r10 37'은 가상 센서 풀에서 ID가 37인 센서의 값을 읽어서 10번 레지스터에 저장하라는 명령어이며, 37번 논리 센서는 남아있는 경로의 수를 저장하고 있다. 논리 센서 풀은 이와 같이 실제 물리적인 센서의 값은 아니지만 이심이의 운영에 필요한 값들을 센서 형태로 지정할 수 있다. 구동기의 출력값에 따라서 가상적으로 이심이의 위치 이동을 할 수 있는 센서 시뮬레이션 모듈과 결합하여 TML 예제를 실험한 결과는 정상적인 동작을 보여주었다.

표 3. TML의 지시어

Directive	Parameter	Operation
$base	x y z	임무의 기준 위치
$home	x y z	모선 위치
$include	filename	지시어 위치에 외부 임무파일을 삽입
$stack	size	스택의 크기

```
# TML example: Lawn-mower type survey
$home 123.4 567.8 0
        BHA     BhEmergency
        BHA     BhKeepDepth 10
        BHA     BhKeepYaw 90
        BHA     BhKeepSpeed 1
        LDI     r11 1    # decrement value
        LDI     r12 3    # count = 3
L10:    WPA     0 0 10
        WPA     100 0 10
        WPA     100 50 10
        WPA     0 50 10
        WPA     100 100 10
        BHA     BhWpTrace
L20:    LSR     r10 37   # r10 = LS[37](remaining WP)
        CMP     r10 r11
        JMPGE   L20      # wait until # of is 1
        SUB     r12 r11
        JMPNE   L10      # repeat survey
        BHA     BhGoHome
```

그림 6. 잔디 깎기 형태의 탐사를 위한 TML 예제

5. 결론 및 향후 연구과제

TML은 간결한 임무 언어이지만 AUV의 임무를 표현하기에 충분한 기능을 갖고 있으며, 가상기계 위에서 코드화된 실행이미지를 실행시키기 때문에 인터프리터 형태의 언어에 비하여 실행 속도가 빠른 임무 언어이다. 다양한 형태의 AUV 임무를 TML로 작성하여 센서 시

뮬레이션 모듈과 결합하여 실험한 결과에 따르면, 정상적인 동작을 확인할 수 있었다.

한국해양연구원에서 개발한 AUV인 이심이는 현재 최대 수심 100m까지 운항이 가능한 이심이100 모델이 개발 완료되었으며, 이후 최대 수심 6,000m까지 가능한 이심이6000 모델을 개발할 예정이다. TML은 2010년도 하반기에 이심이100 모델과 이심이6000 모델에 적용하여 실해역에서 실험할 예정이다.

앞으로 TML을 사용하여 이심이의 운영을 위한 여러 실험을 진행하면서 보완해야할 내용이 발생할 것으로 예상된다. 그러한 부분들을 보완하여 TML을 다른 AUV 시스템에도 적용할 수 있는 범용 임무 언어로 완성해 나가면서, C 언어와 같은 고급 언어 형태의 임무 언어를 개발하여 TML과 연동시키는 것이 향후 연구과제이다.

참고문헌

[1] B. H. Jun, J. Y. Park, F. Y. Lee, P. M. Lee, C. M. Lee, K. Kim, Y. K. Lim, and J. H. Oh, "Development of the AUV 'ISiMI' and a free running test in an Ocean Engineering Basin," Ocean Engineering, Vol.36, No.1, pp.2-14, January 2009.

[2] G. Giger, L. Xue, S. Tangirala, and M. Kandemir, "High Level Mission Programming Support for Autonomous Underwater Vehicles," AUVSI's Unmanned Systems North America, Orlando, FL., 2006.

[3] C. N. Duarte, C. Buzzell, G. R. Martel, D. Crimmins, R. Komerska, S. Mupparapu, S. Chappell, D. R. Blidberg, and R. Nitzel. "A Common Control Language to Support Multiple Cooperating AUVS," 14th International Symposium on Unmanned Untethered Submersible Technology, August 2005.

[4] D. Davis, "Automated Parsing and Conversion of Vehicle-Specific Data into Autonomous Vehicle Control Language (AVCL) Using Context-Free Grammars and XML Data Binding," Proceedings of the 14th International Symposium on Unmanned Untethered Submersible Technology, Durham, NH., August 2005.

[5] http://www.arl.psu.edu/capabilities/at_suv.html.

[6] http://www.navsea.navy.mil/nuwc/newport/auvd/.

[7] J. Jalbert, J. Baker, J. Duchesney, P. Pietryka, W. Dalton, D.R. Blidberg, S.G. Chappell, R. Nitzel, and K. Holappa, "Solar-Powered Autonomous Underwater Vehicle Development," In Proceedings of the Thirteenth International Symposium on Unmanned Untethered Submersible Technology, August 2003.

[8] D. B. Marco, and A. J. Healey, "Current Developments in Underwater Vehicle Control and Navigation: The NPS ARIES AUV," OCEANS 2000 MTS/IEEE Conference and Exhibition, pp 1011-1016, 2000.

[9] D. Brutzman, M. Burns, M. Campbell, D. Davis, T. Healey, M. Holden, B. Leonhardt, D. Marco, D. McClarin, B. McGhee, and R. Whalen, "NPS Phoenix AUV Software integraion and in-water testing," Autonomous Underwater Vehicle Technology, Proceedings of the 1996 Symposium on AUV '96, pp. 99-108, Jun 1996.

[10] C. von Alt, , B. Allen, T. Austin, and R. Stokey, "Remote Environmental Measuring Units," Autonomous Underwater Vehicle's 94 Conference, 1994.

[11] US Navy, The Navy Unmanned Undersea Vehicle (UUV) Master Plan, 2004.

[12] A. Yavnai, "Architecture for an Autonomous Reconfigurable Intelligent Control System (ARICS)," Proceedings of the 1996 Symposium on Autonomous Underwater Vehicle Technology, pp 238-245, 1996.

[13] M. Carreras, J. Yuh, J. Batlle, and P. Ridao, "A Behavior-based Scheme using Reinforcement Learning for Autonomous Underwater Vehicles," IEEE Journal of Oceanic Engineering, Vol 30, pp 416-427, 2005.

[14] P. Ridao, J. Yuh, J. Batlle, and K. Sugihara, "On AUV Control Architecture," Proceedings 2000 IEEE/RSJ International Conf. on Intelligent Robots and Systems, 2000.

[15] 김방현, 전봉환, 최현택, 박종원, 임용곤, 이판묵, "자율무인잠수정을 위한 통신 프로토콜 설계", 선박해양기술 47권 1호, pp. 23-29, 2009. 7.