# 분산화된 센서와 엑츄에이터에 대한 로봇 소프트웨어 컴포넌트의 인터페이스 추상화

양광웅*, 원대희, 최무성, 김홍석, 이호길(한국생산기술연구원), 박준우(파인디지털)

# Robot software component interface abstractions for distributed sensor and actuator

KwangWoong Yang*, DaeHeui Won, Moosung Choi, Hongseok Kim, HoGil Lee (KITECH),
and Joon Woo Park(Fine Digital Inc.)

**ABSTRACT**

Robot is composed of various devices but, those are incompatible with each other and hardly developing reusable control software. We suggest standard abstract interface for robot software component to make portable device and reusable control software of robot. This assures uniform abstracted interface to the device driver software like sensor and actuator and, control program can be transparent operation over device. We can develop devices and control software separately and independently with this idea. This makes it possible to replace existing devices with new devices which have a improved performance.

**Key Words** : Component(컴포넌트), HAL, Abstraction(추상화), Interface(인터페이스), Robot(로봇)

## 1. 서론

로봇은 산업용, 극한작업용, 의료용, 오락용 그리고 인간에 가까운 휴머노이드와 같이 다양하게 존재한다. 또한 로봇은 다양한 분야에 여러 형태로 적용 가능하다. 로봇의 하드웨어 및 제어 소프트웨어는 점점 복잡해지고 있으며 다양한 센서와 엑츄에이터가 사용되고 있다. 로봇에 사용되는 소프트웨어가 복잡해짐에 따라, 여러 개발자가 로봇 개발에 참여하고 있으며, 다양한 목적의 플랫폼 상에서 다양한 언어로 소프트웨어가 작성된다. 우리는 이러한 다양성을 모두 포함할 수 있는 분산환경에서 사용 가능한 미들웨어를 필요로 하고, 로봇 소프트웨어에 투명성과 유연성을 부여하기 위해 소프트웨어 컴포넌터의 인터페이스 추상화가 필요하다.

본 연구에 앞서 우리는 CMR (Component based Modularized Robot)과 FMR (Function based Modularized Robot)을 개발하였다 [1-4]. 이 연구의 목적은 모듈화• 표준화 된 방법으로 로봇을 개발하여 새로운 로봇으로 발전 가능한 기반 소프트웨어/하드웨어 플랫폼의 구조를 연구하는 것이었다. 선행연구에서는 분산환경에서 사용 가능한 미들웨어가 개발되었고, 로봇 소프트웨어를 구성하는 각종 컴포넌트들-Vision, Navigation, Localization, MotorDriver, MotionDriver-이 개발되었다. 기존의 로봇 및 로봇 소프트웨어에서는 투명성과 유연성이 부족하다는 것이다. 예를 들자면, 로봇의 기능 향상을 위해 기존의 센서를 새로운 센서로 교체하거나, 새로운 센서를 추가하기가 어렵다. 또한 기존의 로봇에서 개발된 소프트웨어가 새로운 로봇에 적용되기 어렵다. 이러한 로봇 소프트웨어 구조는 이식에 많은 시간을 필요로 하고, 로봇의 소프트웨어 및 하드웨어 개발 양쪽 모두에 있어서 비효율적이다.

본 연구는 로봇 하드웨어로부터 로봇 소프트웨어를 분리하고 계층적인 소프트웨어 구조를 사용하여 기존 로봇 소프트웨어 개발의 문제점을 해결하고자 한다 [5,6]. 우리는 로봇 장치의 호환과 제어 소프트웨어의 재사용을 가능하도록 하기 위해 로봇 소프트웨어 컴포넌트의 표준화 된 추상 인터페이스를 계층적으로 구성하는 방안을 제시한다. 이 방법은 운영체제에서 장치를 추상화 하는 것과 유사하다.

소프트웨어 컴포넌트가 단일 플랫폼에서만 동작하기보다, 분산 네트워크상에 연결된 다수의 플랫폼에 분산될 때 여러 이점이 있다. 로봇은 필요한 센서와 엑츄에이터를 모두 로봇에 장착하게 되는데, 부피가 크거나 무거운 센서는 로봇에서 분리하여 설치할 수 있으며, 고가의 센서가 여러 대의 로봇에 탑재되어야 하는 경우는 하나의 센서를 다수의 로봇이 사용할 수 있도록 구성할 수

있다. 예를 들자면, 제어 소프트웨어가 설치된 데스크탑 PC로 엑츄에이터만 가지는 다수의 로봇을 제어할 수 있다. 이 때, 천정에 부착된 하나의 위치인식 센서로 다수의 로봇 위치를 추적할 수 있다 [7].

분산환경에서 사용 가능한 미들웨어는 CORBA와 DCOM이 있다 [8,9]. 하지만, CORBA와 DCOM은 범용적인 사용을 목적으로 개발된 소프트웨어라서 로봇에 사용하기에는 과다한 자원을 요구하고 Real-time성이 필요한 예측 가능한 동작을 보장받기 힘들다. 또한 CAN, RS-232같은 통신 미디어를 사용할 수 없으며, 이에 대한 하부 프로토콜을 개발하는 것이 어렵다. 본 연구를 위해 '퍼스널로봇 기반기술개발' 과제에서 개발된 MRSF(Modularized Robot Software Framework) 아키텍처를 사용하였다. 분산 환경에서 동작하는 MRSF 아키텍처는 다양한 개발 언어를 사용할 수 있고, 다양한 개발 플랫폼을 지원하며, 다양한 통신 미디어를 사용할 수 있다 [10-13].

## 2. 하드웨어 추상화 계층

하드웨어 추상화 계층(Hardware Abstraction Layer; HAL)은 운영체제나 가상머신과 같은 소프트웨어에서 일반적으로 사용되어온 개념이다. HAL은 운영체제로부터 프로세서 그리고 플랫폼 의존적인 부분을 분리시킴으로써 운영 체제가 다양한 프로세서 플랫폼과 호환되도록 한다. Linux와 같은 OS를 예로 들면, 이미 수많은 장치들이 사용가능하고 수많은 응용프로그램들이 개발되고 있다. 응용 프로그램 개발자는 하부 플랫폼에 무관하게 소프트웨어를 작성할 수 있다.

다양한 구성의 로봇들은 하드웨어에 있어 실질적인 차이가 있다. 예를 들어, 환경인식을 위해 시각을 이용하는 로봇과 촉각을 이용하는 로봇이 있고, 바퀴로 이동하는 로봇과 다리로 이동하는 로봇이 있다. 본 연구는 제어· 인식 소프트웨어와 하드웨어 사이에 HAL을 제공하고 인터페이스를 표준화 하는 것이다. HAL은 하부 센서와 엑츄에이터에 대한 균일한 추상을 제공하여, 그 하부 하드웨어는 제어· 인식 소프트웨어에 대하여 투명하게 동작한다. 이것은 로봇 제어 소프트웨어가 로봇의 하드웨어에 독립적으로 작성될 수 있게 한다. 그래서 한 로봇을 위해 개발된 제어 소프트웨어가 다른 로봇에 이식될 수 있다.

MRSF가 모든 분야의 로봇 소프트웨어를 포함할 수 없지만 투명하고 이식 가능한 구조를 가져야 한다. 본 연구에서 MRSF 아키텍처에 로봇 하드웨어로부터 제어 소프트웨어를 분리하고 계층적인 소프트웨어 구조를 사용 하도록 하는 HAL 명세를 추가하였다. HAL 명세는 로봇 소프트웨어와 하드웨어 간에 연결을 위한 소프트웨어 컴포넌트의 인터페이스를 정의하고, 하드웨어의 특성을 기술하기 위해 프로파일을 정의한다.

인터페이스는 장치의 추상화된 작업을 기술하는 명세이다. 인터페이스 명세를 기술하기 위해 속성(property)과 메소드(method)가 사용된다. 예를 들자면, 센서 어레이의 센서 개수는 속성으로 기술될 수 있으며, 센서를 읽으라는 명령은 메소드로 기술될 수 있다. 또한, 인터페이스는 소프트웨어 컴포넌트 간의 투명한 접근을 제공하는 서로간의 약속이다. 이런 약속을 정의할 때 C C++, Java와 같은 특정 언어에 의존하지 않는 인터페이스 정의 언어가 필요하다. MRSF 아키텍처에서는 MIDL(Module Interface Definition Language)을 사용한다. MIDL로 기술되는 인터페이스는 속성과 메소드의 묶음으로 이루어지고 인터페이스의 상속 기능을 사용할 수 있다. MIDL로 기술된 인터페이스는 MIDL 컴파일러를 통해 대상 언어에 적합한 스텁/스캐러튼(stub/skeleton) 코드로 변환된다. MIDL은 구현을 위한 언어가 아니며 소프트웨어 컴포넌트의 구현은 C, C++과 같은 프로그래밍 언어를 통하여 이루어진다. 현재 MIDL 컴파일러는 C, C++, RPL(Robot Programming Language), Java, Python, Visual Basic 언어로의 매핑을 지원한다.

로봇의 센서와 엑츄에이터 장치들은 기능에 따라 그룹화 된다. 그래서 유사한 작업을 수행하는 장치는 사용자 관점에서 동일한 장치로 취급되고 같은 클래스로 분류된다. 같은 클래스에 속하는 장치 드라이버나 소프트웨어 컴포넌트는 동일한 인터페이스를 가지지만, 내부 구조는 다를 수 있다. 장치 드라이버는 인터페이스를 구현한 소프트웨어 코드로서, 인터페이스에서 정의한 기능을 수행하기 위하여 장치를 적절하게 제어한다. 일반적으로 드라이버는 특정한 하드웨어에 종속되도록 제작된다. 하드웨어를 직접 제어· 감시하는 소프트웨어를 장치 드라이버로 구분하지만, 장치 드라이버도 소프트웨어 컴포넌트에 속한다.

같은 클래스에 속하는 장치 드라이버일지라도 서로 다른 특성이 있다. 이는 장치의 프로파일로 기술된다. 프로파일에는 장치의 다양한 특징들이 기술될 수 있다. 예를 들자면, 바퀴 직경, 축 길이, 크기와 높이, 무게 등이 프로파일에 기술된다. 로봇에 대하여 바퀴 직경의 변경이 이루어질 경우, 프로파일만의 변경으로 동일한 소프트웨어를 사용하여 로봇을 제어할 수 있다.

## 3. 표준 인터페이스로 추상

로봇의 기능은 작업, 이동, 인식, 처리 기능으로 나눌 수 있다. 로봇이 이동하기 위해서는 환경을 인식하여 장애물을 회피하고 로봇의 위치를 인식하여야 한다. 그리고 작업하기 위하여 사람의 팔과 같은 장치가 필요하다. 이동수단으로 바퀴를 이용하거나 동물과 같이 두 개 이상의 다리를

이용할 수 있다. 환경인식 센서는 절대위치를 파악, 물체와의 상대적 거리를 파악, 이동량을 감지를 위한 센서와, 동물의 시각과 같이 물체를 인식하고 색과 동작 탐지가 가능한 카메라가 있다. 본 연구에서는 위에서 예시한 로봇 장치들에서 장치를 적절하게 분류하고 적절한 인터페이스를 추출하여야 한다.

추상화된 장치에서 소프트웨어와 하드웨어의 경계를 나누는 것이 상당히 애매한 경우가 있다. 예를 들자면, 초음파 위성 수신기와 같이 로봇의 위치를 직접 읽을 수 있는 센서가 있고, 거리 측정에 사용되는 레이저 파인더 센서는 Markov localization 방법에 의해 위치 측정에도 사용할 수 있다. 로봇이 위치를 아는 것도 중요하지만 이동 중 장애물을 피하려면 장애물과의 거리도 알아야 한다. 그래서 센서 디바이스에 대한 추상 계층은 위치를 측정하는 층과 거리를 측정하는 2층으로 구성된다.

한 종류의 장치를 대상으로 인터페이스를 디자인 하는 것은 비교적 쉽지만, 다양한 종류의 장치를 모두 만족하도록 인터페이스를 일반화 시키는 것은 쉽지 않다. 본 연구에서 제시하는 인터페이스가 모든 장치에 대하여 적절하다고 할 수 없다. 인터페이스 추출을 위해, CMR과 FMR에서 사용한 모든 디바이스와 로봇 소프트웨어 컴포넌트를 도식화 하였다. 기능을 수행하는 디바이스 드라이버나 소프트웨어 컴포넌트를 직사각형으로 표시하고 이들간 데이터의 흐름을 화살표로 연결하였다. 그림 1은 로봇에 사용한 SLAM 소프트웨어 구조다. 그림 2는 SLAM 소프트웨어 내의 localization 컴포넌트의 세부 구조다. 그림 3은 SLAM 소프트웨어 내의 obstacle avoidance 컴포넌트의 세부 구조다. 지면 관계상 path planning과 map building 소프트웨어의 세부 구조를 싣지는 않는다.

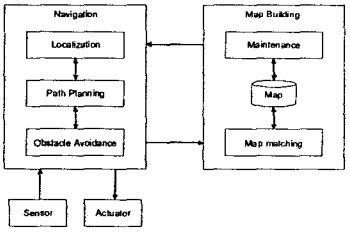


Fig. 1 Structure of SLAM software.

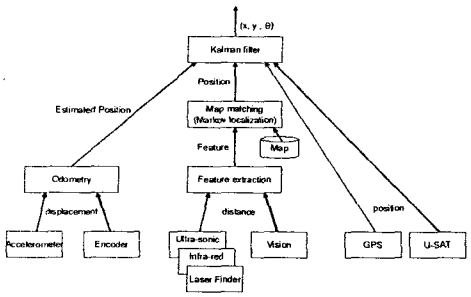


Fig. 2 Structure of localization software.

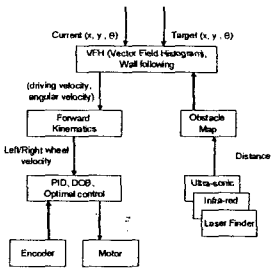


Fig. 3 Structure of obstacle avoidance software.

로봇에 사용중인 센서와 엑츄에이터를 기능과 특성에 따른 장치 클래스로 분류하고, 로봇 소프트웨어 블록간 데이터 흐름에서 유사한 데이터와 명령이 입출력 되는 블록을 같은 클래스에 속하는 컴포넌트로 분리하였다.

유사 장치들의 일반화된 기능에서 표준 추상 인터페이스를 정의하였다. 그리고 유사 장치의 각기 다른 특성을 기술하는 프로파일을 작성하였다 센서와 엑츄에이터에 대한 2층 구조의 소프트웨어 컴포넌트 클래스는 그림 4에서 보여진다.

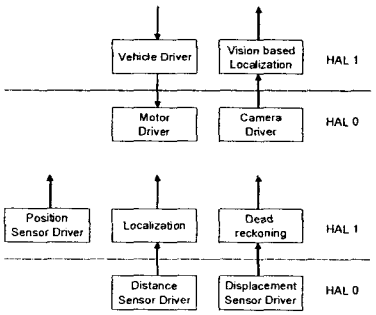


Fig. 4 Layered structure of software component class.

상용 장치들은 범용으로 사용될 수 있도록 만들어지기 때문에, 장치가 가져야 할 대부분의 기능을 포함하고 있다. 하지만 인터페이스의 구조가 본 연구에서 제안하는 구조와 일치하지 않기 때문에, 표준 인터페이스로 변환하기 위한 래핑(wrapping) 코드를 작성하여야 한다.

### 3.1 이동 장치

이동 장치의 인터페이스 추상에는 ㈜한울로보틱스의 싱크로(synchro) 방식의 모바일 모듈과 직접 제작한 차동형 모바일 모듈을 사용하였다. 차동형 모바일 모듈은 구성이 간단하여 모터와 감속기, 엔코더를 사용하여 직접 제작하여 사용하는 경우가 많다. 모터를 제어하기 위하여 Faulhaber MCDC2805 모션 제어기 혹은 직접 제작한 모션 제어기를 사용 하였다. 또한, 모바일 모듈의 제어 소프트웨어가 모션 제어기에 투명하게 접근하도록 하기 위하여 모션 제어기 인터페이스를 추상화 하였다. 그래서 이동 장치는 두 계층의 인터페이스로 구성된다. 모터의 위치와 속도를 직접 제어· 감시하는 모터 드라이버와 로봇의 위치와 속도를 제어하는 vehicle 드라이버로 구성된다.

(1) Motor Driver

모터와 엔코더는 PID 혹은 최적화 제어 등 다양한 폐루프(closed loop) 제어가 가능하다. 대부분의 상용 모터 제어기는 위치제어와 속도제어를 사용한다. 그리고 엔코더 값을 읽을 수 있다. Motor Driver는 이동 장치를 직접 제작하는 경우, 모터 디바이스에 투명한 이동 장치 드라이버를 제작할 수 있다.

모터 드라이버 인터페이스는 Initiate(), DefineHomePosition(), VelocityCtl(), RelativePositionCtl(), AbsolutePositionCtl() 메소드를 가지고, Position, Velocity, Acceleration 속성을 가진다.

모터의 프로파일에는 Operation Mode, Encoder Resolution, Position Limits, Minimum Velocity, Maximum Velocity, Acceleration 이 있다.

(2) Vehicle Driver

3차원 공간에서 강체의 위치는 latitude, longitude, altitude, roll, pitch and yaw 6개의 값으로 표현된다. 일반적으로 평면상에서 이동로봇의 위치는 3자유도의 $(x, y, \Theta)$로 표시된다. 터렛을 가지는 싱크로 드라이버 장치, 전방향 이동 장치는 이동방향과 헤딩(heading)을 구분하여 4자유도의 $(x, y, \Theta_1, \Theta_2)$로 표시되기도 한다.

많은 로봇 소프트웨어가 로봇을 이동시키기 위해 위치제어보다 속도제어를 사용한다. 이동 장치 드라이버는 상대 위치제어, 절대 위치제어, 속도제어가 가능하고, 바퀴의 엔코더(encoder) 값으로 위치를 추정(dead-reckoning)한다.

Vehicle 디바이스 인터페이스는 Initiate(), GoForward(), GotoRelativeXY(), GotoAbsoluteXY(), Stop(), TorqueFree(), Drive(), SetOdometry(), SetVelocity(), SetAcceleration() 메소드를 가지고, Running, Odometry, Velocity, Encoder 속성을 가진다.

프로파일에는 Maximum Velocity, Acceleration, StdDeviation, Offset, WheelDiameter, AxleLength, Maneuverability, DegreeOfFreedom 이 있다.

### 3.2 센서 장치

센서는 절대위치 센서, 상대거리 센서, 변위 센서로 분류한다. 상대위치 센서는 센서와 물체간 거리를 측정하지만, 절대 위치 센서는 하나의 기준으로부터 로봇의 위치 $(x, y)$ 혹은 로봇의 위치와 방향 $(x, y, \Theta)$ 값을 측정한다. 변위 센서는 로봇의 위치 변화량 $(\Delta x, \Delta y)$ 혹은 방향 변화량 $(\Delta \Theta)$ 을 감지한다.

모든 센서 클래스의 인터페이스는 Initiate(), ReadData() 메소드를 가지며, MeasurementCount, MeasurementData 속성을 가진다. 프로파일에는 LimitsMin, LimitsMax, Deviation, SpreadRadius, NumberOfMeasurement, Offset, Adjustment Polynominal, MaximumVariance, MaximumVarianceCount, AveragingCount이 있다.

(1) Position Sensor Driver

로봇의 위치 인식을 위해 GPS(Global Positioning System), PSD(Position Sensitive Detector), RFID(Radio Frequency Identification), 초음파 위성 시스템(U-SAT), 랜드마크 기반 비전(Landmark based Vision) 센서들을 사용한다. GPS, PSD, RFID, U-SAT 센서는 위치 $(x, y)$를 인식하지만 방향을 인식할 수 없다. 방향을 예측하기 위해서는 로봇이 움직여야 하며, odometry의 변화량과 센서에서 측정한 위치의 변화량으로 방향을 추정한다. 랜드마크 기반 비전 센서는 위치와 방향 $(x, y, \Theta)$을 인식한다.

PSD 디바이스를 사용하여 다수의 로봇에 대한 위치 검출이 가능하다. 다수의 로봇에서 방사되는 적외선은 천정의 PSD센서로 검출되고, PSD 디바이스는 로봇에게 위치를 알려주게 된다.

(2) Distance Sensor Driver

로봇과 장애물간 상대거리 측정을 위해 초음파 센서 어레이, 적외선 센서 어레이 또는 레이저 파인더(laser finder) 센서들을 사용한다. 로봇이 이동하기 위해서는 상대거리와 절대위치 모두를 필요로 한다. 장애물을 감지하기 위해서는 상대거리 센서가 있어야 하며, 로봇의 위치를 파악하기 위해서는 절대위치 센서가 있어야 한다. 하지만 상대거리 센서로부터 Markov localization을 통해 확률 기반의 절대위치를 계산해 낼 수 있다. Markov localization 컴포넌트는 Position sensor driver와 같은 클래스에 속한다.

(3) Displacement Sensor Driver

가속도 센서, 엔코더와 같이 이동량을 측정하는 센서는 로봇의 위치를 추측하기 위해 이동량을 적분한다. 그래서 이동 거리가 증가할수록 오차가 누적된다. 오차를 보정하기 위해서 절대위치 센서가 필요하다. 이동량과 절대위치를 보정하기 위하여 Kalman filter를 사용한다.

(4) Camera Driver

카메라는 저렴한 가격에 비하여 대량의 데이터를 입력 받을 수 있다. 하지만 우리가 필요로 하는 정보를 추출하기 위해서는 고성능 프로세서가 필요하다. 카메라에서 입력된 영상은 localization, 거리측정, 물체인식, 색인식, 동작감지 등 다양한 부분에 사용되고 있다. 영상은 320x240크기의 영상과, 640x480크기의 RGB24 포맷 영상을 입력

받을 수 있다.

## 4. 적용

분산화된 센서와 엑츄에이터에 대한 로봇 소프트웨어 컴포넌트의 인터페이스 추상이 로봇 소프트웨어와 장치 드라이버에 효과적으로 적용 가능한지 보기 위하여 로봇 소프트웨어 개발 시나리오를 진행하였다. 첫째, 로봇의 센서와 제어기를 PC에서 시뮬레이션 하면서 원격 PC에서 제어 소프트웨어를 개발 한다. 시뮬레이션은 로봇 하드웨어로는 불가능한 다양한 테스트를 할 수 있어 강인한 소프트웨어를 개발할 수 있다. 둘째, 시뮬레이터를 이용하여 제어 소프트웨어 개발이 어느 정도 완료되었다면, 시뮬레이터를 로봇 하드웨어와 디바이스 드라이버로 교체한다. 이는 제어 소프트웨어에서 연결 정보를 수정하면 된다. 로봇 하드웨어와 연결된 제어 소프트웨어는 시뮬레이터에서 예측하지 못한 문제들을 가지고 있다. 제어 소프트웨어는 여전히 원격 PC에서 개발되기 때문에 로봇에 탑재하여 개발하는 것보다 디버깅이 편리하다. 셋째, 하드웨어 테스트가 완료된 경우, 원격 PC에서 개발한 제어 소프트웨어를 로봇으로 탑재하여, 로봇이 독립적으로 기능을 수행할 수 있다. 마지막으로, 로봇에서 사용중인 센서와 엑츄에이터를 같은 클래스에 속하는 다른 종류로 바꾸어 제어 소프트웨어가 정상적으로 동작하는지 본다.

현재 사용중인 시뮬레이터는 RFID센서와 Camera Device를 제외한 Sensor Device와 Vehicle Device를 시뮬레이션 할 수 있다. 디바이스 클래스별로 시뮬레이터가 존재하며, 시뮬레이터에서 profile의 교체만으로 특성이 다른 다양한 센서들을 만들어낼 수 있다.

이동 장치의 드라이버 인터페이스 추상을 위해서 직접 제작한 차동형 모바일과 ㈜한울로보틱스에서 제작한 싱크로 드라이버 방식의 모바일을 교체하여 테스트 하였다. 후자는 4자유도의 ($x$, $y$, $\theta_1$, $\theta_2$)를 가지는데 제어 소프트웨어가 터릿(turret)의 방향을 제어하는 기능을 포함하고 있지 않아, 디바이스 드라이버를 수정하여 로봇의 이동 방향과 터릿의 방향을 일치하도록 하였다.

로봇 프레임은 되도록 많은 센서를 탈착 가능하도록 설계되어 있다. 테스트 시에는 센서를 하나씩 선택하여 테스트 하였다. RFID 센서의 경우 위치는 알 수 있지만 방향을 알 수 없다. RFID 태그의 감지 범위보다 정밀하게 로봇 위치를 인식하기 위하여 Markov localization 알고리즘을 사용하였고, 방향은 Kalman filter로 예측하였다. U-SAT 센서는 초음파 위성을 사용한 위치 측정 시스템으로 4개의 위성을 사용한다. 로봇이 특정한 장소로 진입하여 1개 이상의 위성이 보이지 않는 경우, 반사파로 인해 잘못된 위치를 출력하였다. 파티션과 책상 등으로 복잡한 실내환경에서는 사용하기가 곤란하였다.

디바이스 드라이버의 인터페이스는 MIDL로 정의되고, MIDL 컴파일러로 skeleton 코드를 생성한다. 센서와 vehicle의 디바이스 드라이버는 주로 C, C++ 언어를 사용하여 작성되었다.

로봇 소프트웨어에서 디바이스 드라이버를 사용하기 위해 디바이스 드라이버의 인터페이스를 MIDL 컴파일러로 컴파일 하여 stub 코드를 생성한다. 로봇 제어 소프트웨어는 C++, Java, Visual Basic 언어를 사용하여 작성되었다.

그림 5,6은 동일한 Markov Localization 소프트웨어로 레이저 파인더와 초음파 센서 어레이를 번갈아 사용하여 테스트 한 것이다. 녹색 십자가가 로봇의 위치를 표시한다. 파란색이 진할수록 로봇이 위치할 확률이 높다. 두 테스트 모두 비슷한 위치로 로봇을 이동시키며 Localization을 수행하였지만, 각 센서의 특징으로 결과가 다르게 출력됨을 볼 수 있다. 그림 5에서 로봇의 위치가 확률이 가장 높은 점과 어긋나 있는 것은 레이저 파인더가 로봇의 중심으로부터 오프셋이 있기 때문이다.

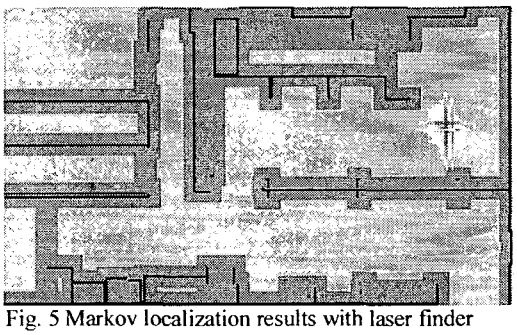

Fig. 5 Markov localization results with laser finder

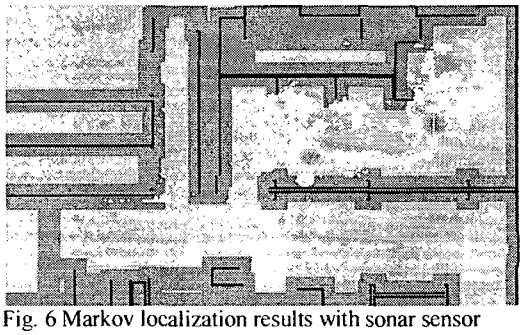

Fig. 6 Markov localization results with sonar sensor

## 5. 결론

본 연구는 로봇 소프트웨어가 분산된 하드웨어에 독립적으로 개발될 수 있도록 하는 디바이스 인터페이스 추상화에 대한 연구이다. 이를 위해 MRSF 아키텍처에 HAL 을 도입하였다. HAL 은 한 로봇을 위해 개발된 로봇 소프트웨어가 다른 로봇에 이식될 수 있게 해준다. 그리고 로봇

소프트웨어를 변경할 필요 없이 기존 로봇의 센서 또는 액추에이터가 교체될 수 있게 해준다.

우리는 로봇에 일반적으로 사용되는 여러 종류의 센서와 액추에이터에 대하여 일관되게 사용 가능한 인터페이스를 설계하고 구현하였지만, 이것이 모든 로봇에 적절하게 대응될 수 없다는 것을 알고 있다. 하나의 완전한 표준화된 인터페이스를 찾는 것은 불가능할 것이다. 하지만, 범용으로 유용하게 사용 가능한 추상화된 인터페이스를 만들어가야 할 필요가 있다.

## 후 기

본 연구는 차세대신기술개발사업의 퍼스널로봇 기반기술개발과제의 지원으로 수행되었습니다.

## 참고문헌

1. Sin-Wook Ryu, KwangWoog Yang, Hong-Seok Kim, Ho-Gil Lee, "Functionally Distributed Modular Robot System using Virtual Machine," *Proceedings of ICCAS*, pp.2330-2335, Oct. 16-19, 2002, Muju, Korea
2. S. G. Roh, S. M. Baek, D. H. Lee, K. H. Park, T. K. Moon, S.W. Ryew, J. Y. Kim, T. Y. Kuc, H. S. Kim, H. G. Lee, H. R. Choi, "Development of Personal Robot Platform : Approach for Modular Desing," *ICCAS*, pp. 2313-2318, October 2002.
3. S. G. Roh, K. H. Park, K. W. Yang, H. S. Kim, H. G. Lee, and H. R. Choi, "Development of Dynamically Reconfigurable Personal Robot," *ICRA*, pp.4023-4028, 2004.
4. S.G. Roh, K.H. Park, K.W. Yang, J.H. Park, H.S. Kim, H.G. Lee and H.R. Choi, "Dynamic Infrastructure for Personal Robot : DynI," *ICCAS*, pp. 2039 - 2044, 2003.
5. Richard T. Vaughan, Brian P. Gerkey and Andrew Howard, "On device abstractions for portable, reusable robot code," *Proceedings of IROS 2003*, Las Vegas, Nevada, October, 2003.
6. Brian P. Gerkey, Richard T. /Vaughan and Andrew Howard, "The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems," *Proceedings of ICAR 2003*, pp.317-323, June 30 – July 3, 2003, Coimbra, Portugal.
7. Brian P. Gerkey, Richard T. Vaughan, Kasper Stoy, Andrew Howard, Gaurav S. ?sukhatme and Maja J Mataric, "Most Valuable Player: A Robot Device Server for distributed Control," *Proceedings of IROS 2001*, pp. 1226-1231, Vailea. Hawaii, 29 Oct. - 3 Nov., 2001.
8. "Common Object Request Broker Architecture: Core Specification," OMG, March 2004.
9. Markus Horstmann and Mary Kirtland, "DCOM Architecture," MSDN, July 1997.
10. Gun Yoon, Hyoung Yuk Kim, Ju Sung Lee, Hong Seok Kim, Hong Seong Park, "Middleware Structure for Personal Robot," *ICCAS*, pp. 153-157, 2003. 6.
11. KwangWoong Yang, Hong-Seok Kim, Jaehyun Park, "A Virtual Machine for Modularized Personal Robot Controller," *Proceedings of ICCAS*, pp.2170-2173, Oct. 16-19, 2002, Muju, Korea.
12. 윤건, 김형육, 박홍성, "모듈 기반 퍼스널 로봇을 위한 미들웨어 구조," 제어.자동화.시스템공학 논문지, 제 10 권, 제 5 호, pp. 464-474, 2004. 5.
13. "표준형 로봇디자인센터(Robot Design Center)의 프레임워크 개발," 제 2 회 퍼스널로봇 기반기술 개발 Workshop, pp. 175-193, 2003.