Journal of the Korea Industrial Information Systems Research Vol. 19 No.5, Oct. 2014
http://dx.doi.org/10.9723/jksiis.2014.19.5.057

MSRDS 플랫폼에서 로봇 센서들의 성능 비교분석†

(Comparative Analysis of the Performance of Robot Sensors in the MSRDS Platform)

이 정 원[1], 정 종 인[2]
(Jeong-Won Lee and Jong-In Chung)

요 약 로봇 개발에서 하드웨어를 개발한 후에 테스트하면 많은 시행착오를 겪게 되어 많은 비용이 소요된다. 로봇이 현장과 동일한 로봇 시뮬레이션을 사용하여 로봇을 개발하면 소프트웨어 및 하드웨어의 병행 개발 및 테스트를 통해 개발결과를 예측할 수 있고 비용을 절감할 수 있다. 로봇 시뮬레이션 플랫폼인 마이크로소프트의 로보틱스 개발자 스튜디오(MSRDS)는 하드웨어 로봇이 없이도 기본적인 로봇 프로그래밍을 할 수 있는 시뮬레이션 로봇과 환경을 제공한다. 본 논문에서는 MSRDS에서 LRF센서, Bumper센서, IR센서, Sonar센서의 성능을 비교분석하기 위하여 미로 찾기를 수행한다. 센서의 성능을 분석하기 위하여 동일한 조건으로 실험한다. 4가지의 센서중에서 LRF센서가 주행시간, 방향전환횟수, 장애물 충돌횟수면에서 우수한 성능을 보인 반면에 범퍼센서는 가장 성능이 낮았다. IR센서와 Sonar센서는 방향 전환횟수면에서 LRF센서 보다는 낮은 성능을 보였다.

핵심주제어 : MSRDS, 로봇 시뮬레이션, SPL, 센서

Abstract MSRDS(Microsoft Robotics Developer Studio), the robot simulation platform provides the simulation robots and environments enabling to the basic robot programming without hardware robots. In this paper, we carry the maze escaping problems to compare and analyze the performance of LRF, bumper, IR, and sonar sensor with the same condition on MSRDS(Microsoft Robotics Developer Studio) environment. To evaluate the performance of sensors, we program the simulation environments with same conditions for all sensors. We could find that the LRF sensor had the highest performance and the bumper sensor has the lowest performance on the travel time, the number of turning, and the number of collisions. It was also confirmed that IR sensor and sonar sensor had lower performance than LRF sensor on the number of turning.

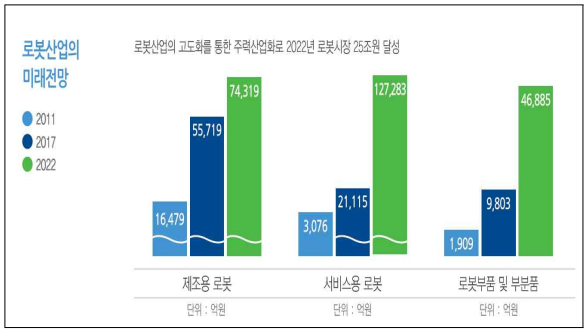
Key Words : MSRDS, Robot Simulation, SPL, Sensor

1. 서 론

한국 로봇산업진흥원은 로봇이 IT, BT등 최신 기술의 혁신적 발전을 통해 융.복합화되고 있고 저출산이나 고령화 문제, 안전에 대한 요구등 메가트랜드에 가장 부합된다고 하면서 1인 1로봇 시대가 올것이라고 내다봤다. 앞으로 로봇 산업은 복지, 교육, 문화 등 타 분야

† 이 논문은 2013년 공주대학교 학술연구지원사업의 연구비 지원에 의하여 연구되었음.
1) 공주대학교 대학원 컴퓨터교육전공, 제1저자
2) 공주대학교 컴퓨터교육과, 교신저자

로의 서비스 확대를 통해 로봇의 보편화가 현실화될 것으로 생각된다. 1978에 자동차 용접로봇의 국내 최초 도입을 통해 부터의 로봇 산업 태동기를 시작하여 2002년부터 차세대 성장 동력산업으로 선정하면서 지능형 로봇이 등장하고 로봇 정책을 본격화 하였다. 2008년부터 지능형로봇 개발 및 보급 촉진법이 제정되면서 로봇 정책을 본격화 하였다[1].



<Fig. 1> Prospect of Robot Industry

최근 초·중·고등학교 및 대학에서도 IT 및 공학 분야에 대한 학생들의 관심을 증대시키고, 학습효과를 높이기 위해 기존 수업 내용에 로봇을 적용하여 응용하도록 함으로써 학생들의 흥미를 유발하고 수업 참여도를 높이고자 많은 시도를 하고 있다. 특히 현업에서 개발 및 테스트를 위한 개발 로봇 등의 물리적인 로봇 구매의 제약과 비용적인 문제에 대한 대안으로서 각종 시뮬레이션 로봇을 활용하는 사례가 늘고 있으며, 이러한 시뮬레이션 로봇을 접목하는 시도는 학생들의 호기심을 자극하고 기존에 접하지 못했던 새로운 환경을 제공함으로써, 기대감과 함께 성취감을 느낄 수 있도록 하여, 의도하는 학습 효과에 긍정적으로 작용한다[2,3].

로봇 개발에서 하드웨어를 개발한 후에 테스트하면 많은 시행착오를 겪게 되어 많은 비용이 소요된다. 로봇이 사용되는 현장과 동일한 로봇 시뮬레이션을 사용하여 로봇을 개발하면 소프트웨어 및 하드웨어의 병행 개발 및 테스트를 통해 개발결과를 예측할 수 있고 비용을 절감할 수 있다[4,5].

전세계적으로 마이크로소프트사의 MSRDS[7], 에볼루션 로보틱스사의 ERSP, 윌로우 개러지사의 ROS, 일본 AIST의 RT(Robot Technology) 미들웨어인 OpenRTM-aist , 유럽의 OROCOS 등이 대표적인 로봇 플랫폼이다.

MSRDS는 로봇 머니퓰레이터의 공학적인 하드웨어 구현요소, 기구학(kinematics)과 같은 소프트웨어 구현요소를 처리할 수 있는 개발 환경과 시뮬레이션을 제공하고 있다. 공통의 메시지 규약과 모듈화된 서비스 기반의 연동기술을 제공하여 다양한 지능형 서비스요소, 인간친화형 기술을 개발하는데 있어서 다양한 요소를 효율적이며 쉽게 접목시킬 수 있는 개발 프레임워크를 제공하고 있다. 그러므로 MSRDS 플랫폼은 하드웨어 로봇이 없이도 기본적인 로봇 프로그래밍을 가능하게 하기 위한 시뮬레이션 로봇과 환경이 제공되며, 직접 시뮬레이션 로봇도 만들어 볼 수 있으며 우주선이나 잠수함과 같은 특이한 형태의 시뮬레이션 로봇도 만들어 볼 수 있다.

서비스 로봇 시장에서의 핵심 기술은 로봇 소프트웨어와 콘텐츠, IT 기술의 융합이라 할 수 있다. 이처럼 로봇 소프트웨어 기술의 급속한 발달과 더불어, 로봇은 산업제품 생산에 핵심적인 요소가 되고 있으며, 인간생활과 밀접한 서비스 로봇에 대한 필요성이 최근 증대하고 있다. 다양하고 고도화된 기능의 로봇 개발과 함께 작업 수행 능력이 증가하고 있으며, 작업 공간도 로봇의 이동성과 함께 점점 확장되고 있기 때문에, 이동작업 공간 내에서 로봇이 자기 위치를 파악하는 것은 작업 수행에 필수적인 요소 기능이라 할 수 있다. 이동로봇이 자기 현재 위치를 알기 위해서는 여러 가지 센서의 정보를 활용해야 한다[6].

본 논문에서는 시뮬레이션 툴 중에서 대표적인 로봇 소프트웨어 제품 중의 하나인 Microsoft의 로보틱스 개발자 스튜디어(MSRDS)의 SPL 플랫폼에서 제공하는 LRF센서, Bumper센서, IR센서, Sonar센서를 동일한 미로 탈출을 통해 각 센서의 성능을 비교·분석 하고자 한다.

2. 관련 연구

2.1 MSRDS SPL 플랫폼

현재 로봇 소프트웨어 프레임워크는 목적에 따라 대단히 많이 연구되고 있다. 이중에서 무료이면서 비교적 많이 알려진 MSRDS(Microsoft Robotics Developer Studio) 프레임워크를 이용한 연구를 진행하였다.

MSRDS는 현재의 로봇 개발 과정에서 겪고 있는 이식성 문제와 재사용성 문제를 해결하고, 로봇 종사자들

에게 보다 쉬운 접근성을 제공하기 위해 만들어진 개발 도구이다.

MSRDS는 그림 2와 같이 기본적으로 Windows 또는 Windows CE 계열의 운영체제 상에서 닷넷(.NET) 프레임워크를 기반으로 하여 동작하는 소프트웨어 플랫폼이다. 또한 기능적으로는 DSS(Decentralized System Service)와 CCR(Concurrency and Coordination Runtime)의 실행환경을 기반으로 한다[7,9]. 소프트웨어 구성에서는 서비스 기반의 아키텍처를 가지고 있으며 포트를 통하여 서비스 간의 통신이 이루어지는 전형적인 서비스 기반 소프트웨어 아키텍처를 기반으로 하고 있다. 따라서 개발자는 간단한 프로그램의 경우 VPL(Visual Programming Language)을 이용하여 응용 프로그램의 작성이 가능하지만, 전문적인 프로그램을 위해서는 SPL (Simple Programming Language), 닷넷(.NET) 프레임워크 환경에서의 C# 프로그램에 익숙하여야 하며 서비스 기반의 소프트웨어 구조를 이해하여야 프로그램 작성이 용이하다. 본 연구에서는 MSRDS SPL 스크립트 언어를 이용하여 각 센서들의 동작을 구현하였다. MSRDS의 SPL은 스크립트 기반의 다양한 시뮬레이션 툴을 제공해 주며, 개발자들에게 좀 더 강력한 시뮬레이션 환경을 개발하고 응용할 수 있도록 도움을 준다. SPL 스크립트 언어는 VPL과 연동되어 실행이 가능하며, C# 언어와도 연동하여 실행될 수 있도록 지원하고 있다.

MSRDS의 시뮬레이션 환경은 Ageia사의 PhysX 엔진을 이용하며, Microsoft XNA 프레임워크를 기반으로 구성 되었다. 시뮬레이터는 하드웨어 개발 이전에 시뮬레이션 로봇을 여러 개발자가 동시에 활용됨으로써 개발 기간을 단축하고, 코드의 생산성과 품질 향상에 이용 가능하다[10].

2.2 센서 비교

2.2.1 LRF 센서

레이저 레인지 파인더(LRF: Laser Range Finder) 센서는 연구용 로봇 플랫폼에서 가장 중요한 역할을 한다[11,12]. MSRDS 시뮬레이션에 포함되어 있는 LRF 센서는 SICK사의 LMS-200 모델을 시뮬레이션으로 구현해 놓은 센서이다. 20kHz~200kHz 정도의 주파수를 갖는 초음파를 발신하여 그것이 반사되어 되돌아오는 시간을 구하여 반사 물체까지의 거리를 탐지한다.

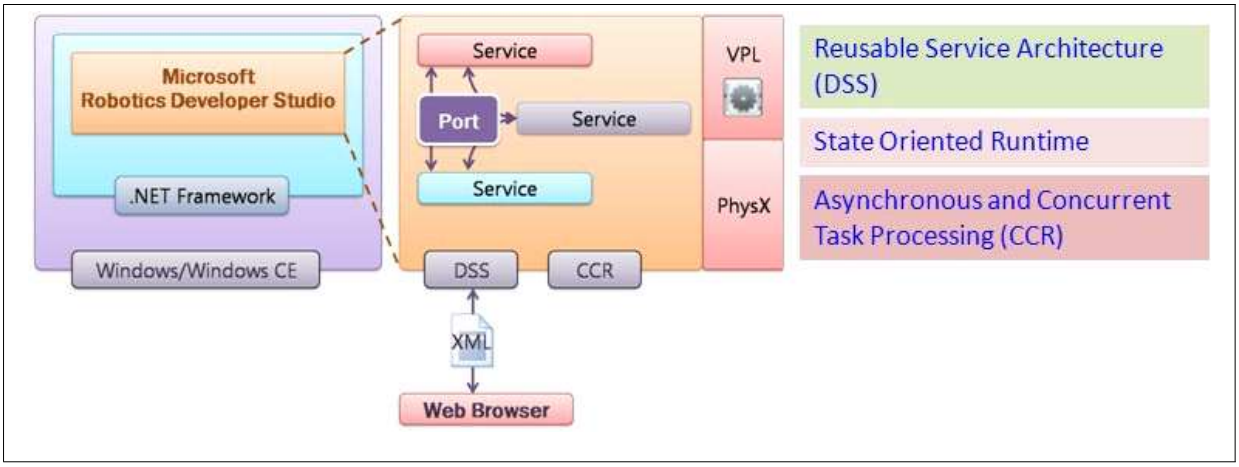
LRF 센서는 다음과 같은 특징을 가진다.

- 한 번에 180도 각도에 대한 레이저 스캐닝을 통해 거리를 측정할 수 있다.
- 측정된 데이터는 0.5도 간격으로 측정되며, 180도 각도에 대해 총 361개의 데이터가 측정된다.
- 최대 측정 거리는 제품마다 차이가 있으나 MSRDS에서 시뮬레이션 되는 모델은 8m까지 측정된다.

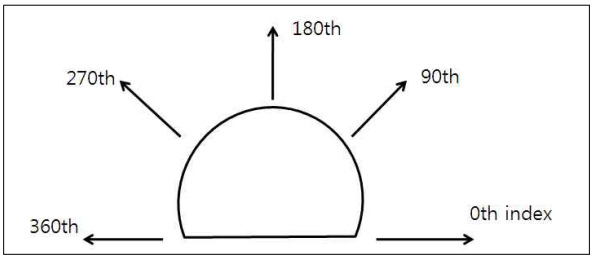
2.2.2 Bumper 센서

범퍼(Bumper) 센서는 실제로 존재하지는 않는다. 실제 현장에서 사용되는 센서는 접촉(Contact) 센서 또는 터치(Touch) 센서들이다.

MSRDS에서 범퍼센서는 터치(Touch) 센서와 동일하



<Fig. 2> Key Functions of MSRDS Architecture



<Fig. 3> Measuring Range of LRF Sensor

며 다른 대상과의 접촉을 인지하기 위해 사용되는 센서이다. 범퍼 센서는 대상이 접촉되었을 경우, True 값을 발생 시키고, 접촉 상태가 해제되었을 경우에는 False 값을 발생 시킨다. MSRDS 범퍼 센서에서는 하나의 범퍼 센서에 여러 개의 범퍼를 추가할 수 있도록 샘플을 제공하고 있으며, 범퍼 센서들을 그룹으로 관리 할 수 있기 때문에 여러 번 작업해야 할 일을 단순화 시켜 준다.

2.2.3 IR 센서

IR(Infrared Range)센서는 MSRDS 2008 R2에 새롭게 추가된 센서로서, 적외선 센서는 발광부와 수광부로 나누어진다. 발광부에서 나온 적외선이 물체에 반사되어 수광부에 얼마나 많은 양이 들어오느냐에 따라서 수광부에 들어오는 전압의 양이 변화하게 된다[13].
시뮬레이션 IR 센서는 다음과 같은 특징을 가진다.

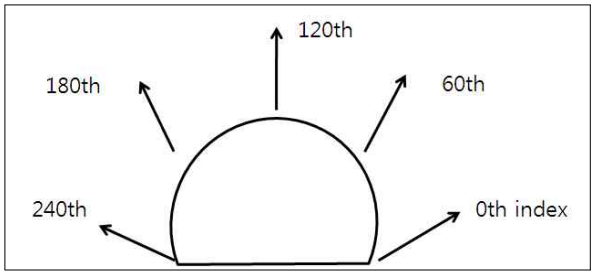
- 1m 이내의 장애물에 대한 거리를 측정한다.
- 한 방향에 대해서만 거리를 측정한다.
- 시뮬레이션 상의 IR 센서는 대상의 색상에 관계없이 거리만을 측정해 준다.

2.2.4 Sonar 센서

Sonar 센서는 초음파를 이용한 거리감지 센서로, 초음파를 만들어 송출하고 반사되어 돌아오는 시간차를 계산하면 거리계가 되고, 단지 수신량의 변화나 세기를 비교검출하면 물체의 유/무를 탐지하는 탐지기가 된다. 일반적으로 공기 중 물체 감지에 적용되는 Sonar 센서에서 사용되는 주파수는 9khz에서 50khz정도이며 이 범위의 주파수는 강력한 초음파 펄스를 발생하기 쉽고 지향특성을 얻기가 용이하기 때문이다.

Sonar 센서는 초음파의 절대값 보다는 초음파 존재의 유무, 초음파 펄스파면의 상대적 크기를 이용하는 경우가 많으며 LRF센서와 같이 넓은 범위를 단시간에 측정한다. 넓은 범위를 측정할 수 있기 때문에 어군 탐지기에 주로 사용된다. MSRDS 시뮬레이션에 포함되어 있는 Sonar센서는 다음과 같은 특징을 가진다.

- 한 번에 120도 각도에 대한 음파측정으로 거리를 측정할 수 있다.
- 측정된 데이터는 0.5도 간격으로 측정되며, 120도 각도에 대해 총 241개의 데이터가 측정된다.
- 최대 측정 거리는 500m까지 측정된다.



<Fig. 4> Measuring Range of Sonar Sensor

2.3 로봇 주행

SPL은 MSRDS의 시뮬레이션 환경을 스크립트로 구현할 수 있는 도구이다. SPL 스크립트 상에서 시뮬레이션 엔진을 실행하기 위하여 StartSimulationEngine 스크립트를 시용한다. 시뮬레이션 환경에 이미 구현되어 있는 로봇을 추가하거나 모터가 장착된 새로운 로봇을 추가할 수 있다. 로봇은 다양한 종류가 있으며 작동하는 메커니즘도 다양하다. 본 논문에서는 두 개의 메인 바퀴와 하나의 캐스트를 가지는 2축 모터 플랫폼을 사용하였다.

2축 모터를 갖는 로봇을 추가하기 위하여 AddDifferentialDriveEntity 스크립트를 사용한다. 2축 모터는 두 개의 모터를 갖는다. 각 모터의 파워 값을 제어함으로써 주행을 제어할 수 있다. 모터의 파워값의 범위는 -1.0~+1.0이며 최대 전진속도는 +1.0, 모터의 정지는 0, 최대 후진속도는 -1.0이다. 모터의 파워를 제어하기 위하여 왼쪽 및 오른쪽 바퀴의 파워값을 전달하는 SetDrivePower 스크립트를 사용한다.

SPL은 로봇을 제어하기 위한 다양한 스크립트를 제공한다.

Get(): 센서의 데이터를 읽음.
Go(power): power로 전진.
Go(left, right): 왼쪽바퀴에 left 파워 및 오른쪽 바퀴에 right 파워로 주행
GoTo(distance, power): power로 distance 거리만큼 이동

RotateDegrees(degrees, power): power로 degrees 각도 만큼 회전

SetDrivePower(left, right): 왼쪽바퀴에 left 파워 및 오른쪽 바퀴에 right 파워를 설정

Turn(degrees, power): power로 degrees 각도 만큼 회전

DriveDistance(distance, power): distance만큼 power로 주행

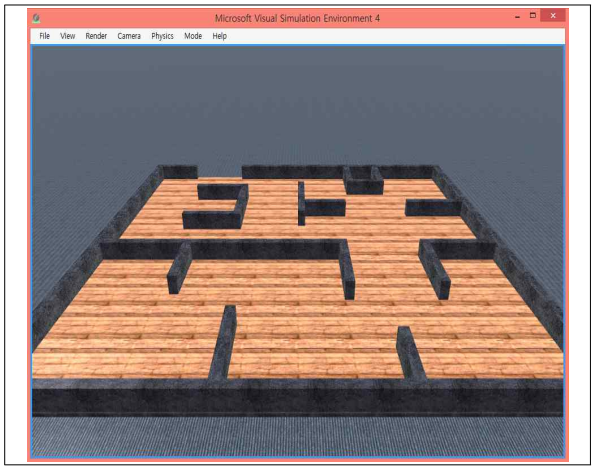
Wait n: n 밀리초만큼 수행을 지연

Pressed: 범퍼센서의 범퍼가 눌러졌는지 판단값 반환

3. 실험 설계

3.1 미로 생성

본 실험에서 사용할 미로를 생성하기 위하여 MSRDS VPL과 SPL으로 프로그래밍하여 사용자가 원하는 미로를 생성할 수 있다. SPL개발환경에서는 미로를 생성하기 위하여 Maze Explorer와 Maze Builder 서비스가 제공되며, 이와 관련한 전용 서비스(API)들이 존재한다.



<Fig. 5> Maze generated by Maze Builder

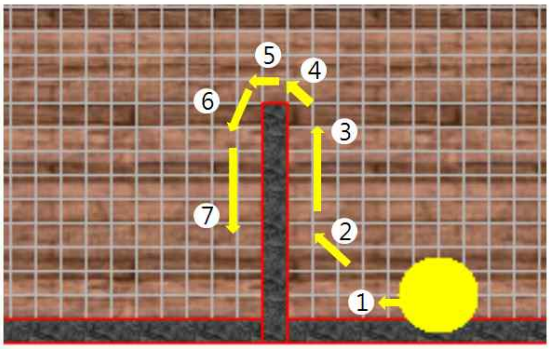
3.2 센서 로봇 주행 알고리즘

3.2.1 LRF센서

LRF 센서를 장착한 로봇의 주행 알고리즘은 다음과 같다.

알고리즘

```
// 알고리즘의 핵심 원리: 로봇이 미로의
   벽을 왼쪽에 끼고 따라간다.
if(미로의 벽과 로봇의 좌측거리가 0.15m보다 작거나 135도 방향의 거리가 0.3m작거나 전방 거리가 1m보다 작다)
{
현재의 방향보다 30도 오른쪽으로 회전한 후 직진
}
else
if(미로의 벽과 로봇의 좌측거리가 1.5m보다 크다)
{
현재의 방향보다 30도 왼쪽으로 회전한 후 직진
}
```



<Fig. 6> Running Method of Robot with LRF Sensor

코드

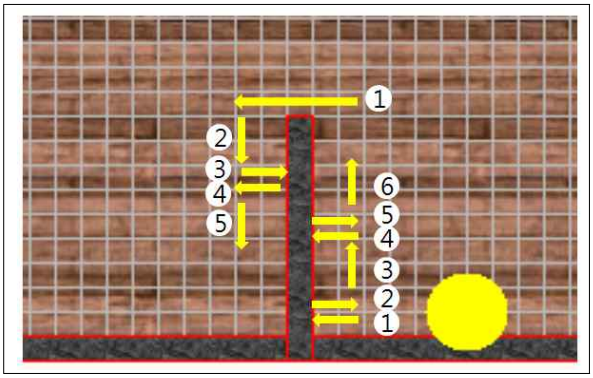
```
while (true)
{
  distance = lrf1.Get()
  d180 = distance[355]
  d135 = distance[270]
  d90 = distance[180]
  d0 = distance[1]
if (d180 < 150 || d135 < 300 || d90 < 1000)
{
  base1.Turn(-30, 0.2)
  SetDrivePower
    /TargetName:base1
    /LeftWheelPower:0.1
    /RightWheelPower:0.1
}
```

코드

```
else {
  if (d180 > 1500) {
    base1.Turn(30, 0.2)
    base1.GoTo(0.1, 0.3)
    SetDrivePower
        /TargetName:base1
        /LeftWheelPower:0.1
        /RightWheelPower:0.1
  }
}
```

3.2.2 범퍼센서

범퍼(Bumper) 센서를 장착한 로봇의 주행 알고리즘은 다음과 같다.



<Fig. 7> Running Method of Robot with Bumper Sensor

알고리즘

```
// 알고리즘의 핵심 원리: 로봇이 미로의 벽을
   왼쪽에 끼고 따라간다.


0.3m를 이동한다.
if(미로의 벽과 로봇이 충돌이 발생한다)
{
현재의 위치에서 0.3m를 후진한다.
오른쪽으로 90도 회전한다.
현재의 위치에서 0.3m 전진한다.
      if (미로의 벽과 로봇이 충동하지 않는다)
      {
      왼쪽으로 90도 회전한다.
      앞으로 0.3m 전진한다.
      }
}
```

코드

```
bmp_pressed = bumper1.Get()
if(bmp_pressed == true){
    base1.DriveDistance(0.3, -0.3)
    wait 1000
    base1.RotateDegrees(-90, 0.2)
    wait 1000
    base1.SetDrivePower(0.3, 0.3)
    wait 1000

bmp_pressed2 = bumper1.Get()
if(bmp_pressed2 == false){
    base1.RotateDegrees(90, 0.2)
    wait 1000
    base1.SetDrivePower(0.3, 0.3)
    wait 1000
  }
}
```

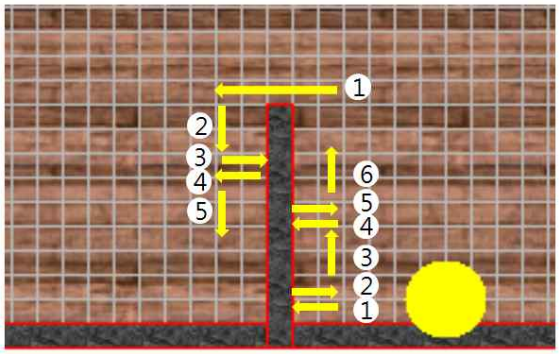
3.2.3 IR센서

IR 센서 로봇를 장착한 로봇의 주행 알고리즘은 전방의 거리를 측정하여 주행 및 회전하는 것을 제외하고는 범퍼센서의 알고리즘과 동일하다.

알고리즘

```
// 알고리즘의 핵심 원리: 로봇이 미로의 벽을
   왼쪽에 끼고 따라간다.


0.3m를 이동한다.
if(미로의 벽과 로봇의 거리가 0.5m보다 작다)
{
현재의 위치에서 0.3m를 후진한다.
오른쪽으로 90도 회전한다.
현재의 위치에서 0.3m 전진한다.
      if (미로의 벽과 로봇의 거리가 0.5m보다
크다)
      {
      왼쪽으로 90도 회전한다.
      앞으로 0.3m 전진한다.
      }
}
```



<Fig. 8> Running Method of Robot with IR Sensor

코드

```
ir_distance1 = ir1.Get()
if(ir_distance1 <= 0.5){
    base1.DriveDistance(0.1,- 0.3)
    wait 1000
    base1.RotateDegrees(-90, 0.2)
    wait 1000
    base1.SetDrivePower(0.2, 0.2)
    wait 1000
    ir_distance1 = ir1.Get()
  if(ir_distance1 > 0.5){
    base1.RotateDegrees(90, 0.2)
    wait 1000
    base1.SetDrivePower(0.3, 0.3)
    }
}
```

3.2.4 Sonar센서

Sonar 센서는 최대 전방 120도까지의 거리를 측정할 수 있기 때문에 Sonar센서를 로봇의 전방과 좌측에 장착하였으며 로봇의 주행 알고리즘은 LRF센서와 같은 원리로 주행한다.

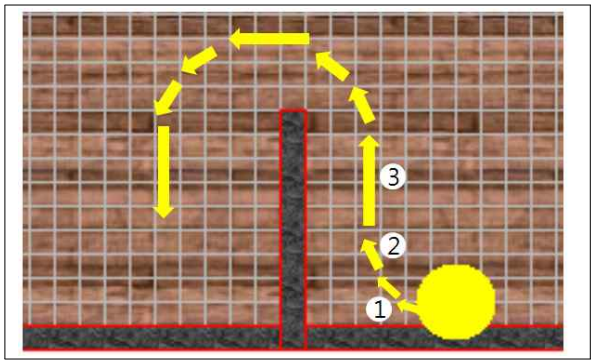
알고리즘

// 알고리즘의 핵심 원리: 로봇이 미로의 벽을 왼쪽에 끼고 따라간다.

if(미로의 벽과 로봇의 좌측거리가 0.15m보다 작거나 135도 방향의 거리가 0.3m작거나 전방 거리가 1m보다 작다)
{
현재의 방향보다 30도 오른쪽으로 회전한 후 직진
}

알고리즘

else
if(미로의 벽과 로봇의 좌측거리가 1.5m보다 크다)
{
현재의 방향보다 30도 왼쪽으로 회전한 후 직진
}



<Fig. 9> Running Method of Robot with Sonar Sensor

코드

```
while (true)
{
    distance1= sonar1.Get()
    distance2= sonar2.Get()
    d1_1 = distance1[240]
    d1_2 = distance1[180]
    d1_3 = distance1[120]
    d2_1 = distance2[240]
    d2_2 = distance2[180]
    d2_3 = distance2[120]

if (d2_2 < 0.150 || d1_1 < 0.300 || d1_2 < 1)
    {
      base1.Turn(-30, 0.2)
      base1.Go(0.1, 0.1)
} else    {
     if (d2_2 > 1.500)
       {
          base1.Turn(30, 0.2)
          base1.GoTo(0.1, 0.3)
          base1.Go(0.1, 0.1)
                  }
          }
}
```

4. 실험 결과

MSRDS의 SPL 플랫폼을 이용하여 동일한 미로에서 같은 주행속도로 각 센서들의 미로 탈출 실험을 통해 주행시간, 장애물 충돌횟수, 방향전환 횟수의 데이터를 측정하여 비교·분석하였다.

4.1 실험 조건

Windows 환경의 컴퓨터에서 동일한 실험 조건을 위해 동일한 미로에서 같은 주행속도로 각 센서 모두에 동일하게 적용하였으며, 센서의 특성을 고려하여 적용할 수 있는 알고리즘은 조금씩 차이가 있으나 회전 기준은 벽을 타고 왼쪽방향을 주행하는 공통 조건으로 실험하였으며, 각 센서별 10회씩 미로를 탈출하는 것을 실험하였다. 모든 로봇은 처음에 파워를 0.1로 설정하여 주행하다가 회전하게 될 경우에는 0.2의 파워로 회전한 후에 0.1m 거리를 0.3의 파워로 이동하는 동일한 실험 조건으로 진행하였다.

4.2 실험 항목

실험 항목은 장애물(미로) 탈출(주행) 시간, 장애물 충돌 횟수, 장애물(미로) 탈출을 위한 센서별 방향전환 횟수 에 대한 측정을 하였다.

- 장애물(미로) 탈출(주행)시간
  - 각 로봇이 미로를 탈출하는데 걸리는 총 시간을 의미한다.
- 방향전환 횟수
  - 각 로봇이 미로를 탈출하면서 장애물을 회피하기 위해 오른쪽 또는 왼쪽으로 방향을 전환한 총 횟수를 의미한다.
- 장애물 충돌 횟수
  - 각 로봇이 미로를 탈출하면서 장애물에 충돌한 총 횟수를 의미한다.

4.3 실험 데이터

4.3.1 LRF 센서

<Table 1> Measured Data of Robot with LRF sensor

	1	2	3	4	5	6	7	8	9	10
주행 시간 (분.초)	3.50	3.48	3.46	3.49	3.45	3.43	3.37	3.49	3.51	3.50
방향 전환 횟수	55	53	56	56	54	54	51	56	58	56
장애물 충돌 횟수	2	2	4	1	2	1	1	3	3	3

4.3.2 Bumper 센서

<Table 2> Measured Data of Robot with Bumper sensor

	1	2	3	4	5	6	7	8	9	10
주행 시간 (분.초)	24.18	23.56	23.51	23.58	24.07	23.50	23.48	24.12	23.53	23.57
방향 전환 횟수	333	333	333	333	333	333	333	333	333	333
장애물 충돌 횟수	167	167	167	167	167	167	167	167	167	167

4.3.3 IR 센서

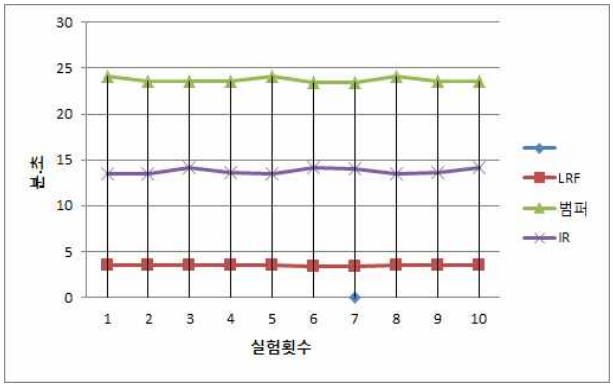
<Table 3> Measured Data of Robot with IR sensor

	1	2	3	4	5	6	7	8	9	10
주행 시간 (분.초)	13.46	13.42	14.18	13.57	13.51	14.11	14.02	13.47	13.56	14.17
방향 전환 횟수	193	191	218	201	195	207	203	193	198	210
장애물 충돌 횟수	12	5	10	21	13	14	8	9	8	12

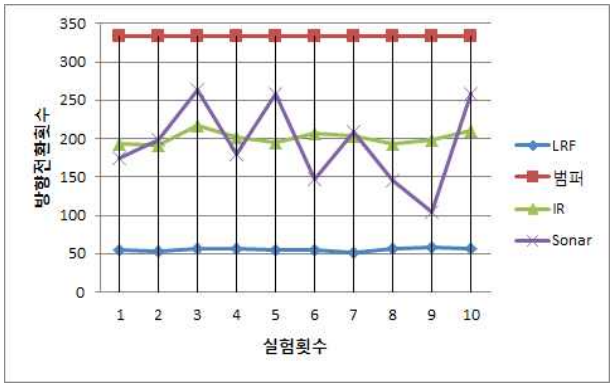
4.3.4 Sonar 센서

<Table 4> Measured Data of Robot with Sonar sensor

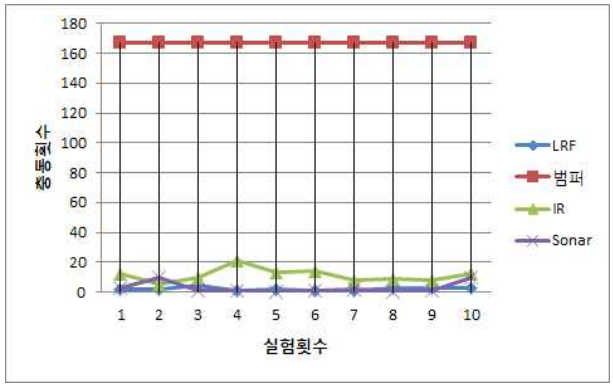
	1	2	3	4	5	6	7	8	9	10
주행 시간 (분.초)	5.41	5.28	7.10	6.12	7.05	4.56	5.41	4.42	4.00	7.48
방향 전환 횟수	175	198	263	179	258	147	209	145	104	258
장애물 충돌 횟수	3	10	1	1	0	1	2	0	1	10



<Fig. 10> Comparison of Travel Time



<Fig.11> Comparison of the number of Turning



<Fig.12> Comparison of the number of collision

4.4 센서 간 비교·분석 결과

실험을 위해 로봇에 장착한 LRF 센서, IR 센서, Sonar 센서, Bumper 센서들의 경우 아래와 같이 주행을 하는데 몇 가지 영향을 받는 요인들을 확인하였으며 시뮬레이션을 통한 자율 주행 시 센서에 영향을 미치는 요인을 분석하였다.

4.4.1 주행 속도

LRF 센서, IR 센서, Sonar 센서의 경우 각 센서들이 센서정보를 수집하기 이전에 빠른 속도로 동작하는 경

우에는 장애물 충돌 등이 발생하기 때문에, 센서 각각의 자체 측정 스케쥴(Schedule) 시간을 고려하여 자율주행 속도를 너무 빨리 하지 않도록 해야 하는 등의 매우 밀접한 관계가 있다.

범퍼(Bumper) 센서의 경우에는 주행 속도가 너무 빠를 경우 장애물과의 충돌력이 매우 커기지 때문에 충돌 이후 반사작용에 따라 로봇의 주행 방향이 흐트러지는 현상이 발생하였으며, 범퍼 센서를 동작시킬 경우의 주행 속도의 빠르기는 범퍼가 장애물과 충돌하는 힘의 에너지와 밀접한 관계가 있다. 이는 MSRDS 시뮬레이션은 물리적 현상을 그대로 반영하기 때문이다.

4.4.2 측정을 위한 출력 메모리

실험을 위해 센서를 Runtime 시에 Command창을 통해 실시간 센서정보를 출력하여 실험 데이터를 확인하는 과정에서 센서정보의 출력물이 많아지게 될 경우 센서의 자체 측정 스케쥴(Scheduler) 시간과 주행 속도에 민감하게 작용하여 원활한 주행을 확보하지 못하는 현상이 나타났으며, 이로 인해 장애물과의 충돌도 발생하는 센서들도 나타났다. 실제 센서 정보에 대한 출력물을 발생시키지 않을 경우에는 원활한 주행을 보여 준다.

4.4.3 Sonar센서의 데이터의 변화

각 센서마다 10회의 주행을 실시하여 주행시간, 방향전환횟수, 장애물 충돌횟수을 측정하였으나 Sonar센서의 방향전환횟수 및 주행시간의 편차가 다른 데이터 보다 크다. 방향전환횟수의 편차가 특히 큼을 확인하였다. Sonar센서는 전방 120도 각도에 대한 음파측정으로 거리를 측정할 수 있기 때문에 본 실험에서는 2개의 Sonar센서를 부착하였다. 두 개의 Sonar센서의 음파 간섭으로 인하여 이러한 결과가 나온 것으로 추측할 수 있다.

4.4.4 프로그램 작성 패턴

Procedure_SensorNotify 방법은 센서가 동작하고 있는지를 계속 정보수집(listening)을 하고 있다가 센서가 동작을 하면 해당 프로시저를 호출하므로 컴퓨터 자원을 많이 소비하므로 While Loop방법보다 센싱속도가 늦었다. 그러므로 본 실험에서는 While Loop방법을 채택하여 실험을 수행을 하였다.

4가지의 센서중에서 LRF센서가 주행시간, 방향전환횟수, 장애물 충돌횟수면에서 우수한 성능을 보였으며 반면에 범퍼센서는 가장 성능이 낮았다. IR센서와 Sonar센서는 방향 전환횟수면에서 LRF센서 보다는 낮은 성능을 보였다.

5. 결 론

동일한 환경에서 각 센서들을 실험한 결과 크게 두 가지로 구분되어지는 특징들이 나타났다. 첫 번째 특징으로는 측정 거리에 따라서 센서들을 구분할 수 있으며, 두 번째 특징으로는 측정 방향에 따라서 센서들을 구분할 수 있다. 또한 이 두 가지 특징을 기준으로 센서들을 분류할 때 장거리 측정과 다양한 범위의 방향을 동시에 측정 가능한 센서는 LRF센서라는 것을 확인할 수 있었으며, 짧은 시간과 광범위한 범위를 측정 할 수 있다는 점에서 LRF 측정 데이터에 대한 활용도가 높다는 것을 확인할 수 있었다.

일반적인 거리 센서 또는 단방향 센서와 레이저 거리 센서 분석

● 일반적인 거리 센서(IR 센서, Sonar 센서) 및 단방향 측정 센서(IR 센서, Bumper 센서)
- 근거리 또는 근접한 거리에 있는 장애물의 거리를 측정하기 위해 사용된다.
- 센서 별로 측정 가능한 거리 대역을 가지고 있기 때문에, 거리 측정 범위(Range) 별로 센서를 달리 적용해 주어야 한다.
- 한 방향에 대해서만 측정이 가능하기 때문에 여러 방향에 대해 거리를 측정할 경우, 여러 개의 센서를 원형으로 배치해야 한다.
- 정확도가 떨어지며, 노이즈가 작용할 수 있다.

● 레이저 거리 센서(LRF센서)
- 정확도가 매우 높다.
- 일반적인 거리 센서에 비해 장거리에 있는 장애물의 거리 측정이 가능하다.
- 한 방향이 아닌 특정 각도 범위 내에 있는 장애물의 거리를 한 번에 모두 측정 가능하다.

로봇 소프트웨어의 많은 부분이 공통된 기술임에도 불구하고 많은 기업에서 처음부터 개발하는 중복 투

자의 특징이 있고, 동일한 요구 사항이 존재함에도 불구하고 기능이 유사한 다른 로봇에 공유되지 않고 있으며, 다양한 요구 사항의 변화와 기술의 발전이 존재하는 로봇의 소프트웨어에 기능 추가가 이루어지지 않고, 성능도 향상되지 않고 있다. 이와 같은 문제점을 해결하기 위하여 로봇 소프트웨어 플랫폼은 로봇 산업의 생태계를 활성화 시키는 쪽으로 기능이 집중되어야 하며, 특히 독립적이고 표준화된 플랫폼을 활용하여 많은 사용자가 다양한 응용 프로그램을 작성하고 이와 같은 유용한 분석 데이터를 이용할 수 있는 생태계가 형성되어야 한다.

여러 로봇 소프트웨어 플랫폼에서 제공 되어지는 시뮬레이션 센서들의 구현 방식은 표준화 되지 않았지만, 센서들의 개념적 측면과 활용적 측면에서는 매우 유사하기에 각 센서들의 특징을 알고 장단점을 상호 보완하여 융합 한다면 좀 더 효율적인 로봇으로의 확장이 가능하다.

향후 연구에서는 여러 가지의 센서들이 미로의 특정한 부분에서 이상행동을 하는 것을 추적하는 연구가 필요할 것이다.

References

[1] H. Bae and Y. J. Kim, “Technic trend of Intelligent Robot”, Week Technic Trend, National IT Industry Promotion Agency, Vol. 1444, 2010.

[2] Y. J. Kim, “MSRDS Simulation Environment and External Interface”, Robor and Human, Korea Robotics Society, Vol. 7, No. 2, pp. 16-22, 2010.

[3] S, H. Cho, “The Effect of Robotics in Education based on STEAM”, Journal of Korea Robotics Society, Vol. 8, No.1, pp. 58-65, 2013.

[4] J. S. Park, “Discrete-Time Sliding Mode Control for Robot Manipulators”, Journal of The Korea Industrial Information System Society, Vol 16, No. 4, 2011.

[5] S. P. Kim, “Kinematic and dynamic analysis of a spherical three degree of freedom joint rehabilitation exercise equipment”, Journal of The Korea Industrial Information System Society, Vol. 14, No. 4, pp.16-29, 2009.

[6] M. Y. Kim, S. T. Ahn, H. S. Cho, “Bayesian Sensor Fusion of Monocular Vision and Laser Structured Light Sensor for Robust Localization of a Mobile Robot ”, Journal of Institute of Control Robotics and Systems, Vol. 16, No. 4, pp. 381-390, 2010.

[7] http://www.microsoft.com/robotics/

[8] B. W. Choi, “A Review and Outlook of Robotic Software Frameworks”, Journal of Korea Robotics Society, Vol. 5, No. 2, pp. 169-176, 2010.

[9] S. Y. Hong, “Intelligent Robotics Programming for SMART Creative Engineering”, BNC Education, 2012.

[10] J. I, Chung and Y. J. Kim, “MSRDS VPL- Robotics Programming for Reasoning Enhancement”, HongRung Pub. 2012.

[11] H. J. Kim, “Sensor Engineering”, HongRung Pub, 2010.

[12] J. O. Kim “Fundmental and Application of Sensors“, BokDoo Pub“, 2012.

[13] C. Y. Park, B. H. Leem, J. T. Ryu, “Development of Gas Sensor Modules and Sensor Calibration Systems”, Journal of The Korea Industrial Information System Society, Vol. 15, No. 2, pp. 83-90, 2010.

이 정 원 (Jeong-Won Lee)

- 공주대학교 컴퓨터교육전공 교육학석사
- 공주대학교 컴퓨터교육전공 박사과정

- 관심분야 : 정보시스템 성과, 교육과정 개발, 국가직무능력표준(NCS) 응용SW엔지니어링 활용패키지개발

정 종 인 (Jong-In Chung)

- 정회원
- 경북대학교 전자공학과(전산전공) 공학사
- 경북대학교 대학원 전자공학학과 (전산전공) 공학석사

- 서강대학교 전자계산학과 공학박사
- 미국 서던캘리포니아 대학교(USC) 박사후연수과정
- 공주대학교 사범대학 컴퓨터교육과 교수
- 공주대학교 과학영재교육원장, 원격교육연수원장
- 관심분야 : 정보보안, 로봇프로그래밍, 영재교육

논 문 접 수 일 : 2014년 07월 18일
1 차 수 정 완 료 일 : 2014년 08월 24일
2 차 수 정 완 료 일 : 2014년 09월 30일
게 재 확 정 일 : 2014년 10월 15일