

졸업 작품 2차 중간보고서

시각 장애인을 위한 장애물 감지 스마트 지팡이

- 라즈베리파이를 이용한 장애인 편의 솔루션 -



건국대학교 컴퓨터공학부

김민재 이규진 이종찬

지도교수 : 유준범 교수님 (인)

< 목 차 >

1. 개요	2
1.1. 선정 배경	2
1.2. 목적	3
2. 프로젝트 목표	4
3. 결과물 명세	5
3.1. 결과물 구성	5
3.1.1. 하드웨어 구성	5
3.2. 결과물 사양	5
3.2.1. 하드웨어 사양	5
3.2.2. 소프트웨어 사양	6
4. 개발 상세	7
4.1. 기술적 요구사항	7
4.1.1. 라즈베리파이 (Raspberry Pi)	7
4.1.2. 초음파센서 (Ultrasonic Sen)	7
4.1.3. 진동모듈	8
4.1.4. 블루투스 (Bluetooth)	8
4.2. 작동원리	10
4.2.1. 초음파	10
4.2.2. 진동 모듈	12
4.3. 개발 방법	13
4.3.1. Data Flow Diagram	14
4.3.2. Structure Chart	14
4.4. 용어 설명	16
4.4.1. GPIO	16
4.4.2. 커널 패닉	16
4.4.3. 페어링	16
4.4.4. OpenCV	16
5. 개발 현황	17
5.1 System Context Diagram	17
5.2 Data Flow Diagram	18
6. 구현 진행 현황	30
7. 개발 계획	30
8. 참고문헌	31

0. 프로젝트 이름

시각 장애인을 위한 장애물 감지 스마트 지팡이

1. 개요

1.1. 선정 배경

IT 기술의 발전과 여러 스마트 기기의 개발은 우리의 삶을 더 편리하고 윤택하게 하는데 도움을 주고 있다. 그에 따라 우리의 삶이 많이 바뀐 것은 사실이지만 우리 주위엔 아직 변화가 필요한 분야가 많이 남아있다. 그 중 한 분야로 우리는 장애인을 위한 스마트 기기 분야를 선정하였다. IT 기술의 혜택을 받음으로써 편리함의 극대화를 느낄 수 있는 사람들이야 말로 바로 장애인들이기 때문이다. 그리고 그 중에서도 우리는 시각장애인에 주목하였다. 2013년 보건복지부 통계에 따르면 전체 등록 장애인 약 250만명 중 시각장애인의 수는 약 10%인 25만명이 넘고 그 중 1~3급의 중증 시각장애인은 5만명에 달한다. 그런데 과연 그들은 IT 기술의 혜택을 얼마나 받고 있을까?

현재 삼성을 제외하고 모든 기업들이 시각장애인 안내견 사업에서 손을 이미 뗀 상태에다 삼성 마저 몇 년 전 그 규모를 대폭 축소하면서 안내견 공급에 차질이 생긴 상태이다. 안내견 마저 없다면 그들을 도울 수 있는 건 실질적으로 가까운 가족들이나 지인들 혹은 사회봉사자 같은 인력들뿐이다. 그 인력마저 없다면 단지 시각장애인용 흰 지팡이에 온 몸을 의지해야만 하는 신세가 된다. 그들은 지금 기술의 도움이 누구보다도 절실하게 필요한 상태다. 따라서 우리는 시각장애인들의 흰 지팡이에 IT 기술을 접목하여 새로운 가치를 이끌어내자는 의견을 모으게 되었다.



〈그림 2 시각장애인과 안내견〉

1.2. 목적

시각장애인들이 항상 입을 모아서 얘기하는 것이 앞이 보이지 않아 다칠 위험이 커서 밖을 나가기 겁이 난다는 것이다. 시각장애인을 지팡이는 무릎 아래에 있는 장애물에 대해서만 탐지가 가능하고, 무릎 위로는 어떠한 장애물이 오더라도 피할 수 없기 때문에 얼굴과 어깨 쪽을 다치는 일이 많다. 그래서 그들이 조금 더 편하게 다닐 수 있도록 도와주는 또 하나의 눈이 되어주고자 하였다.

따라서 우리는 시각장애인들이 항상 휴대하고 다니는 흰 지팡이에 초음파 센서를 장착한 라즈베리파이를 부착함으로써 전방의 장애물들의 정보를 실시간으로 이용자가 알 수 있도록 해주는 스마트 지팡이를 만들고자 한다. 또한 단지 전방의 장애물이 있는지 없는지의 여부뿐만 아니라 초음파 센서를 추가 장착하여 좌전방, 우전방, 상, 하 등 여러 정보를 받아와 그 결과를 가공하여 사용자에게 조금 더 유용한 정보를 전해주고자 한다.

그러나 시각장애인이라는 사용자의 특성상 디스플레이 같은 시각적 정보를 제공하는 것이 불가능하거나 제한적이므로 청각, 촉각을 이용하여 정보를 전달해야 한다. 따라서 진동 모터와 블루투스 이어폰을 통해서 사용자에게 시각에 대한 정보를 전달해 줄 것이다. 조작법 또한 쉽고 간편해야 함으로 최대한 버튼의 개수가 적고 어렵고 복잡한 조작을 필요로 하지 않아야 한다.

2. 프로젝트 목표

- a. 라즈베리파이와 초음파 센서를 이용하여 전방의 무릎 위 장애물을 인식
- b. 여러 개의 초음파 센서를 통해 받은 정보를 가공하여 알려주어 보행자의 상황 판단을 도움
- c. 탐지한 장애물의 거리, 방향, 높이 정보를 시각장애인이 구분할 수 있는 진동과 소리 정보로 제공
- d. 편리한 사용을 위한 블루투스 이어폰과의 연동 기능
- e. 시각장애인이 사용할 수 있는 쉬운 조작법
- f. 안전과 직결되므로 믿고 사용할 수 있는 높은 신뢰성
- e. 휴대용 기기임을 감안하여 휴대 가능한 가벼운 무게와 긴 전원 지속 시간

3. 결과물 명세


3.1. 결과물 구성

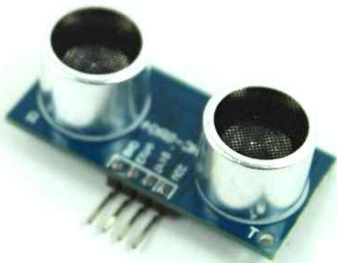
3.1.1. 하드웨어 구성


라즈베리파이2, 라즈베리파이 전용 I/O 키트, HC-SR94(초음파 센서), MB-0412V(진동모터), MicroSD카드 16GB, 5V 보조 충전 배터리, 지팡이 목업(mockup), Bluetooth CSR 4.0 Dongle, 블루투스 이어폰

3.2. 결과물 사양

3.2.1. 하드웨어 사양

	Raspberry Pi 2 B		
	Processor	Broadcom	BCM-2836
	Clock Speed	900	MHz
	Register Width	32-bit	
	RAM	1GB	
	GPIO Pins	40	
	I/O Current MAX	5-10mA	
	POWER	5V	600mW
	Operating System	Linux	& Others

	HC-SR04		
	Power Supply	5V DC	
	Quiescent Current	<2mA	
	Effectual Angle	<15°	
	Ranging Distance	20 ~ 5,000mm	
	Resolution	3 mm	
	Weight	15.00g	
	Size	2.0 * 4.3 * 1.5cm	

	MB-0412V		
	Rated Voltage	3V DC	
	Operating Voltage	2.2~3.6V DC	
	Rated Current	≤70mA	
	Rated Speed	10000±2000r/min	
	Stall Current	≤105mA	
	Starting Voltage	≤1.7V DC	
	Terminal Resistance	35±20%	
	Connection Form	Wire type	

	PiCamera		
	Resolution	2592 * 1944	
	QXIGA	15 frame/sec	
	1080p	30 frame/sec	
	VGA	90 frame/sec	
	S/N rated	36 dB	
	Dynamic Range	68 dB	
	Ribbon length	150mm	
	Weight	3.4g	

3.2.2. 소프트웨어 사양

3.2.2.1. 운영체제

Raspbian - Release date : 2015-05-05 (Kernel Version 3.18)

3.2.2.2. 언어

Python - 2.7x Version

3.2.2.3. GPIO 제어 API

WiringPi - 2.18 Version

3.2.2.4. 블루투스 제어 API

LightBlue - 0.4 Version

4. 개발 상세

4.1. 기술적 요구사항

4.1.1. 라즈베리파이 (Raspberry Pi)

라즈베리파이는 리눅스 커널 기반 운영체제를 사용한다. 라즈비안 (Raspbian) 이라는 라즈베리 파이에 최적화된 데비안 계열의 무료 운영체제가 제공되고 있다. 소형 컴퓨터인데도 불구하고 강력한 성능의 GPU를 기본적으로 내장하고 있으나 본 프로젝트에선 아웃풋으로 그래픽컬한 내용을 다루고 있지 않으므로 고려하지 않는다. 라즈베리파이에서는 라즈비안 이외에도 데비안, 우분투, FreeBSD 등의 운영체제가 제공되기도 한다.

본 프로젝트에서는 라즈베리파이에 특화된 운영체제인 라즈비안을 사용하여 안정적인 환경을 제공하며, 개발적인 측면에서는 프로젝트 개발 팀원들이 평소에 주로 사용하는 우분투(Ubuntu)와 같은 데비안 계열의 리눅스이므로 라즈비안을 선택하였다.



<그림 3 라즈베리파이>

4.1.2. 초음파센서 (Ultrasonic Sensor)

가까운 거리에 있는 물체 혹은 사람의 유/무, 거리측정, 속도 측정 등에 사용된다. 초음파 소자는 고유 진동에 전압이 가해지면 압전 효과에 의해 초음파를 발생시키게 된다. 초음파의 파장은 전파 속도를 주파수로 할당한 값과 같다. 전파의 속도는 $3 \times 10^8 \text{m/s}$ 이지만 공기 중에 전파되는 순간 음속을 바뀌게 되어 344m/s 의 속도를 가지게 된다. 이때 파장이 짧으므로 거리 방향과 정밀도가 높다.

초음파 센서 모듈을 포함하여 다른 모듈을 사용하기 위해서는 GPIO 보드를 사용해야하는데 그 보드를 제어하려면 기본적으로 커널 레벨에서 이루어져

야 하지만 모든 센서를 커널에서 제어하게 되면 개발 미숙으로 인해 지연 문제와 실제 동작 중 예상치 못한 예외로 커널 패닉(Kernel Panic)이 발생할 수 있다. 이런 관점에서 본다면 유저 레벨에서 GPIO를 제어 할 수 있는 라이브러리가 필요한데 이때 제공되는 라이브러리가 WiringPI 이다. 이 라이브러리는 현재 Python에도 제공되고 있으며 GPIO를 제어하는 라이브러리 중 가장 강력하다고 알려져 있어서 많은 사람들이 사용하고 있다. 이에 따라 관련 자료도 많아서 개발 시에 시간을 단축할 것으로 예상된다.



〈그림 4 초음파 센서〉

4.1.3. 진동모듈 (Vibrate Module)

진동모터는 전자기적 힘의 발생원리를 이용, 전기적 에너지를 기계적 진동으로 변환한 것이다. 즉 회전자의 무게 중심을 일방향으로 편심되게 구성해 회전시키면 회전 불균형인 진동이 발생하는 원리를 이용한 것으로 그 형태에 따라 바(bar)형과 편평형으로 분류된다. 최근에는 단순히 진동을 내는 기능 외에 스피커와 결합하는 등 기능상 변화를 주거나, 음악·게임·동영상 등에 요구되는 진동의 세기를 조절하는 기능을 부가하기도 하고, 스프링에 매달린 추가 상하 왕복운동에 의해 진동을 발생시키는 새로운 진동모터가 개발되기도 했다. 이때 특히 라즈베리파이 등 소형 컴퓨터에서 부착해서 사용할 수 있게 개발된 부품을 진동 모듈이라고 하고, 원리는 진동 모터와 같다.

진동 모듈도 GPIO 보드에 달리게 되는데 역시 마찬가지로 WiringPI로 제어할 수 있다. 초음파 센서와는 다르게 앞에서 밝혔듯이 이 장치는 출력장치이므로 값을 세팅하는 형식으로 제어해야 한다.

4.1.4. 블루투스 (Bluetooth)

모바일 기기에서 흔히 사용되는 근거리 무선 통신(PAN) 기술이다. 일종의 무선 USB같은 용도로 많이 사용된다. 2.45GHz 주파수 대역을 사용한다. 블루투스는 다양한 기기들이 안전하고 저렴한 비용으로 전 세계적으로 이용할 수 있는 무선 주파수를 이용해 서로 통신할 수 있게 한다. 블루투스는 유선 USB를 대체하는 개념이며, 와이파이(Wi-Fi)는 이더넷(Ethernet)을 대체하는 개념이다. 암호화에는 SAFER+를 사용한다. 장치끼리 믿음직한 연결을 성립하려면 키워드를 이용한 페어링(pairing)이 이루어지는데, 이 과정이 없는 경우도 있다.

블루투스 통신은 소켓 프로그래밍과 많이 닮아 있다. 특히 Python에서 라이브러리 형태로 제공되는 LightBlue를 사용한다면 Java에서 사용하던 소켓 프로그래밍 API 만큼 쉽게 통신을 사용 할 수 있다.

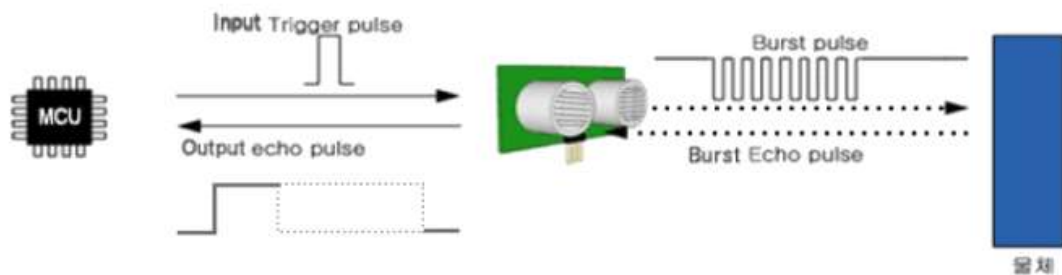


<그림 5 블루투스>

4.2. 작동원리

4.2.1. 초음파

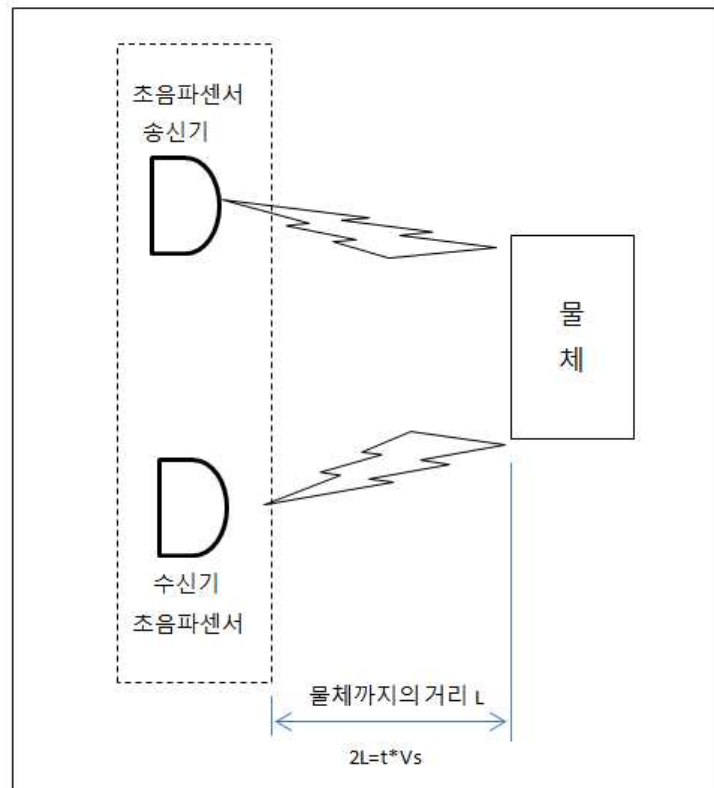
초음파 센서는 2개의 압전 소자로 되어 있는 것을 바이물, 1개의 압전 소자로 되어 있으면 유니물이라고 부른다. 이 압전 소자를 통해서 초음파를 발생 시키게 된다. 물체에서 반사된 음파가 센서로 다시 돌아오고, 그 음파가 입력 센서를 진동시켜 반사되어 돌아 왔다는 것을 표시하게 된다.



〈그림 6 초음파 센서 작동 원리 1〉

4.2.1.1. 송신용 초음파 센서 : 초음파를 출력한다는 것은 송신 소자에 공진 주파수와 동시에 주파수의 발진 전력을 공급하여야 하는데 공급하는 신호는 구형파 펄스이고 전압 진폭은 1~20V 정도입니다. 단 직류 성분이 없어야 하므로 콘덴서를 사용하여 직류성분을 제거하여 공급한다. 전압이 높으면 음압이 높게 되므로 멀리까지 보낼 수 있지만 약 10V에서 포화되어 10V이상 전압을 높여도 큰 효과는 없다.

4.2.1.2. 수신용 초음파 센서 : 초음파를 수신하면 그 출력 단자에 전압을 출력합니다. 수신 감도는 센서의 공진 주파수 근처가 송신기와 마찬가지로 40khz가 가장 좋으며 출력에 부하 저항을 연결하면 그 양단에 초음파의 강약에 따라서 40khz의 정현파가 나타나게 됩니다. 출력 전압은 거리에 따라 몇mV~수백mV정도가 되기 때문에 이것을 증폭해서 비교기 등에서 고감도로 검출해서 디지털 신호로 변환하여 사용한다.



〈그림 7 초음파 센서 작동 원리 2〉

4.2.1.3. 거리 측정 : 초음파센서의 송신측에서 짧은 시간동안 펄스를 출력하면 신호가 물체에 도달하여 반사되어 되돌아온 신호를 수신기의 초음파 센서에서 검출한다.

$$t = \frac{2 \times L(\text{물체와의 거리m})}{V_s(\text{음속m/s})}$$

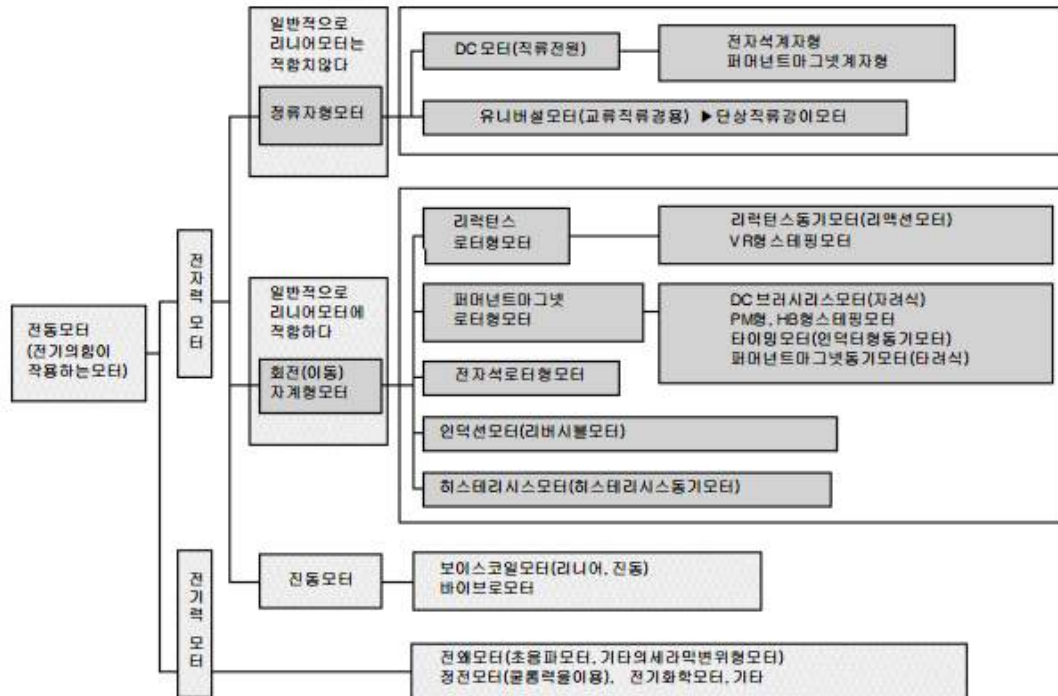
t: 신호가 되돌아 올때까지 걸리는 시간(s)

〈그림 8 초음파 센서 작동 원리 3〉

여기서 음속 $V_s = 331.5 + 0.6T$, T = 온도(℃)이며, 실내 온도를 25℃라고 하면 음속 $V_s = 340\text{m/s}$ 라고 하면 음이 1cm의 거리를 왕복하는데 걸리는 시간은 $t_c = 2 \times 0.01 / 340 = 58.824 \times 10^{-6}$ 이 된다.

4.2.2. 진동 모듈과 스피커(이어폰)

모터의 종류는 다음과 같이 알 수 있다. 진동 모터의 경우 전자력 모터로 분류된다.



<그림 9 모터의 종류>

위에서 밝힌바와 같이 진동은 소리를 제외하고 시각 장애인에게 제공 될 수 있는 가장 뚜렷하고 명확한 출력 방법이다. 즉, 가장 중요한 출력장치이므로 여러 가지 방법으로 사용자에게 신호를 주어야 한다. 예를 들어, 이용자 정면의 초음파 센서를 통해 감지되는 물체에 대해서 진동의 세기와 주기에 따라 사용자에게 정보를 제공한다. 세기가 작을수록 물체는 멀리 떨어져 있고, 크기가 클수록 물체가 가까이 있는 등의 정보를 제공한다. 또 블루투스로 연결된 이어폰을 통해 소리를 출력하는데, 사용자 좌전방, 우전방을 감지하는 초음파 센서로 통해 입력되는 물체의 정보에 대한 방향성을 제공한다. 예를 들어 왼쪽 아래에 물체가 있을 경우 왼쪽 이어폰에 낮은 음의 소리를 출력하고, 오른쪽 상단에 물체가 감지 될 경우 높은음의 소리를 오른쪽 이어폰에만 출력하는 그런 방식이다.

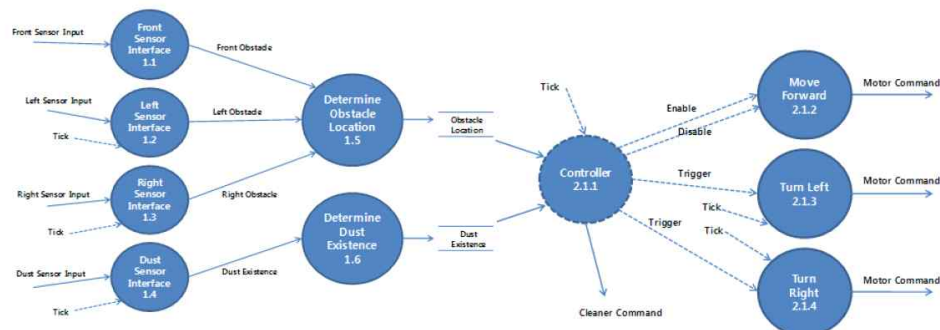
이때 진동 센서와 이어폰을 모두 사용하는 특별한 출력이 있는데, 앞이 완전한 벽이라 나갈 수 없는 경우엔 특별한 경고음을 출력하여 사용자가 벽이라는 것을 쉽게 인지하도록 하고, 방의 출구를 잘 찾을 수 없는 경우에는 출구의 방향을 알려주는 등의 정보를 제공 할 예정이다.

4.3. 개발 방법

본 프로젝트는 프로그램의 크기가 작고 데이터의 흐름이 중요한 부분을 차지하므로 절차적 개발 방법인 SASD(Structured Analysis and Structured Design)을 사용하여 개발한다. DFD(Data Flow Diagram)을 통해서 데이터의 전체적 흐름을 파악하고 Structure Chart를 통해 전체적인 프로그램의 구조를 알 수 있다. Divide and Conquer를 기반으로 시스템을 잘게 쪼개어 Top-Down 방식으로 프로그램을 구성한다. 최종적으로 프로그램의 질을 높이고 시스템의 위험을 줄이는 것이 목표이다. 무엇보다도 SASD는 프로그램의 Input과 Output이 명료하게 표현되므로 임베디드 시스템에 적합하므로 본 프로그램에 알맞은 방법론이라고 생각하였다.

4.3.1. Data Flow Diagram

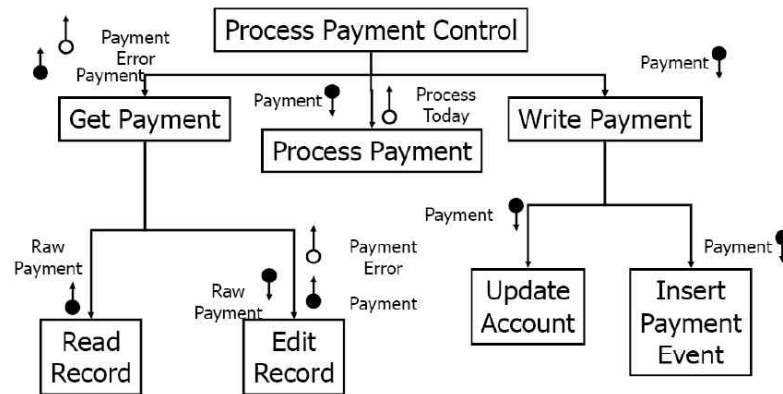
시스템의 자료 흐름을 나타내기 위해 사용한다. 자료가 입력되어서 출력되기까지의 과정을 한 눈에 알 수 있도록 해준다.



<그림 10 Data Flow Diagram>

4.3.2. Structure Chart

프로그램의 전체적인 구조를 알기 위해 Data Flow Diagram 작성 후 Structured Design 과정에서 수행한다.



<그림 11 Structure Chart>

4.4. 용어 설명

4.4.1. GPIO(General Purpose Input/Output)

일반적인 입력, 출력이 가능한 IO. 하나의 하드웨어 핀이 입력 혹은 출력을 수행한다. 하나의 핀이 Register를 통해서 Input으로 쓰일지 Output으로 쓰일지 결정되는 것이다. Input, Output 외에 High impedance(하이임피던스) 상태로 두어서 상대방의 출력을 그대로 받아들이는 상태로 설정할 수 있다.

4.4.2. 커널 패닉(Kernal Panic)

운영 체제가 치명적인 내부 오류를 감지하여 안전하게 복구가 불가능할 때 취하는 동작이다. 유닉스 계열 운영 체제에서 널리 쓰인다.

4.4.3. 페어링(Pairing)

블루투스를 사용하는 둘 이상의 기기에 대해서 서로를 등록하여 연결하는 작업. 최초 1회 페어링 후에는 자동으로 연결된다.

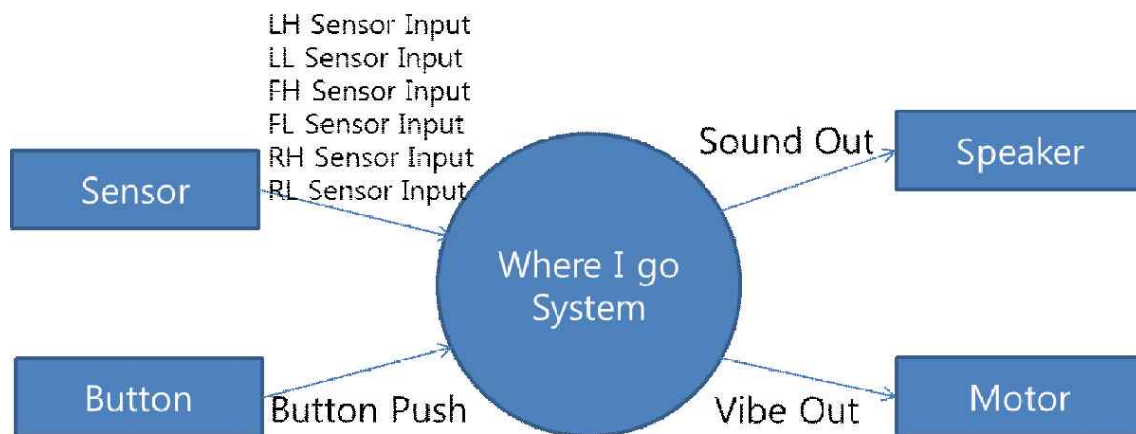
4.4.4. OpenCV(Open Computer Vision)

오픈 소스 컴퓨터 비전 C라이브러리이며 실시간 이미지 프로세싱에 중점을 둔 라이브러리이다.

5. 개발 현황

5.1. System Context Diagram

5.1.1. Basic System Context Diagram



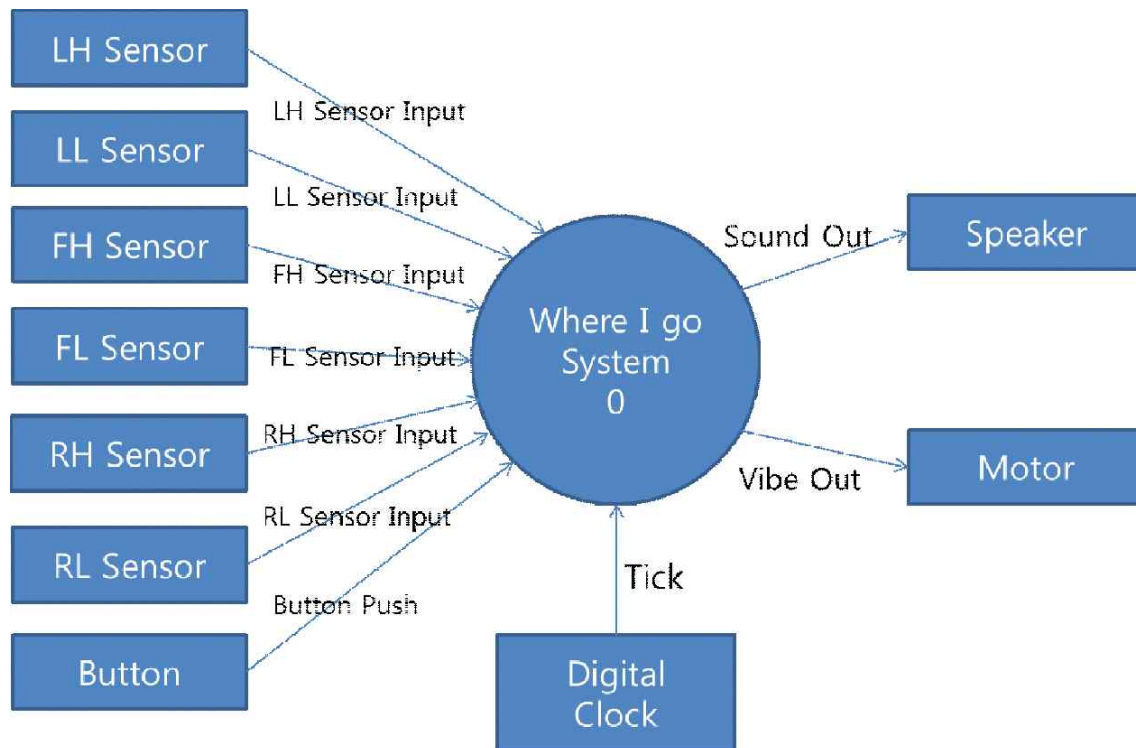
5.1.2. Event List

Input / Output Event	Description
LH Sensor Input	좌측 상단의 센서가 장애물을 정기적으로 감지한다.
LL Sensor Input	좌측 하단의 센서가 장애물을 정기적으로 감지한다.
FH Sensor Input	정면 상단의 센서가 장애물을 정기적으로 감지한다.
FL Sensor Input	정면 하단의 센서가 장애물을 정기적으로 감지한다.
RH Sensor Input	우측 상단의 센서가 장애물을 정기적으로 감지한다.
RL Sensor Input	우측 하단의 센서가 장애물을 정기적으로 감지한다.
Button Push	버튼 입력과 동시에 장애물을 감지되는 센서를 기억한다.
Sound Out	장애물이 감지된 센서에 따라 알맞은 소리를 출력한다.
Vibe Out	감지된 장애물의 거리에 따라 진동을 출력한다.

5.2. Data Flow Diagram

5.2.1. DFD Level 0

5.2.1.1. DFD



5.2.1.2. Process Specification

5.2.1.2.1. Process 0

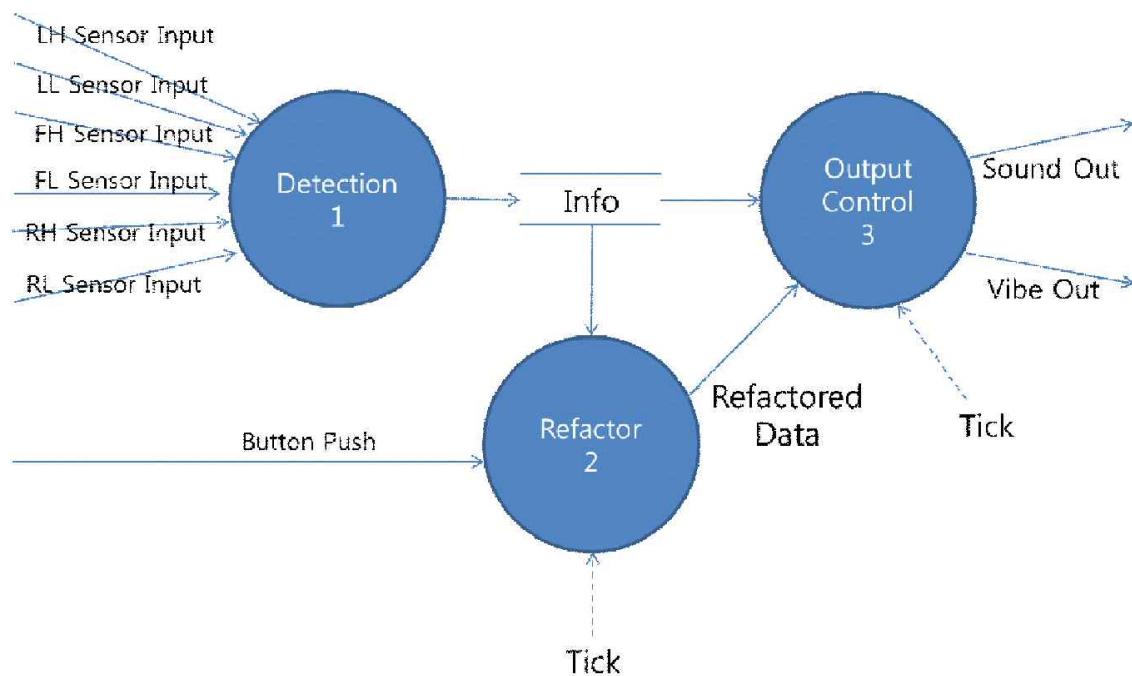
Reference Number	0
Name	Where I go System
Input	LH Sensor Input, LL Sensor Input, FH Sensor Input, FL Sensor Input, RH Sensor Input, RH Sensor Input, Button Push
Output	Sound Out, Vibe Out
Process Description	정기적으로 6개의 Sensor Input에서 입력 받은 값을 지정해 놓은 값에 따라 3단계로 변환해 주어서 Sound Out과 Vibe Out으로 보내준다. 또한 Button Push가 입력되면 입력 받은 값들을 조합하여 Sound Out으로 보내준다.

5.2.1.3. Data Dictionary

Input / Output Event	Description	Format / Type
LH Sensor Input	좌측 상단의 센서가 장애물을 정기적으로 감지한다.	Int , Periodic
LL Sensor Input	좌측 하단의 센서가 장애물을 정기적으로 감지한다.	Int , Periodic
FH Sensor Input	정면 상단의 센서가 장애물을 정기적으로 감지한다.	Int , Periodic
FL Sensor Input	정면 하단의 센서가 장애물을 정기적으로 감지한다.	Int , Periodic
RH Sensor Input	우측 상단의 센서가 장애물을 정기적으로 감지한다.	Int , Periodic
RL Sensor Input	우측 하단의 센서가 장애물을 정기적으로 감지한다.	Int , Periodic
Button Push	버튼 입력과 동시에 장애물을 감지되는 센서를 기억한다.	True / False , Interrupt
Sound Out	장애물이 감지된 센서에 따라 알맞은 소리를 출력한다.	Left / Right/ Both / Off
Vibe Out	감지된 장애물의 거리에 따라 진동을 출력한다.	Level 1, 2, 3 / Off

5.2.2. DFD Level 1

5.2.2.1. DFD



5.2.2.2. Process Specification

5.2.2.2.1. Process 1

Reference Number	1
Name	Detection
Input	LH Sensor Input, LL Sensor Input, FH Sensor Input, FL Sensor Input, RH Sensor Input, RH Sensor Input
Output	Info
Process Description	정기적으로 6개의 Sensor Input에서 장애물이 인식한 곳을 기억하고 장애물이 인식되었다면 거리 값에 따라 3단계로 나누어서 장애물이 나타난 곳과 거리 값을 Info로 보내준다.

5.2.2.2.2. Process 2

Reference Number	2
Name	Refactor
Input	Button Push, Info
Output	Refactor_Command
Process Description	Button Push가 입력되면 Info의 정보를 입력 받아서 8단계의 상황으로 분류를 한다.

5.2.2.2.3. Process 3

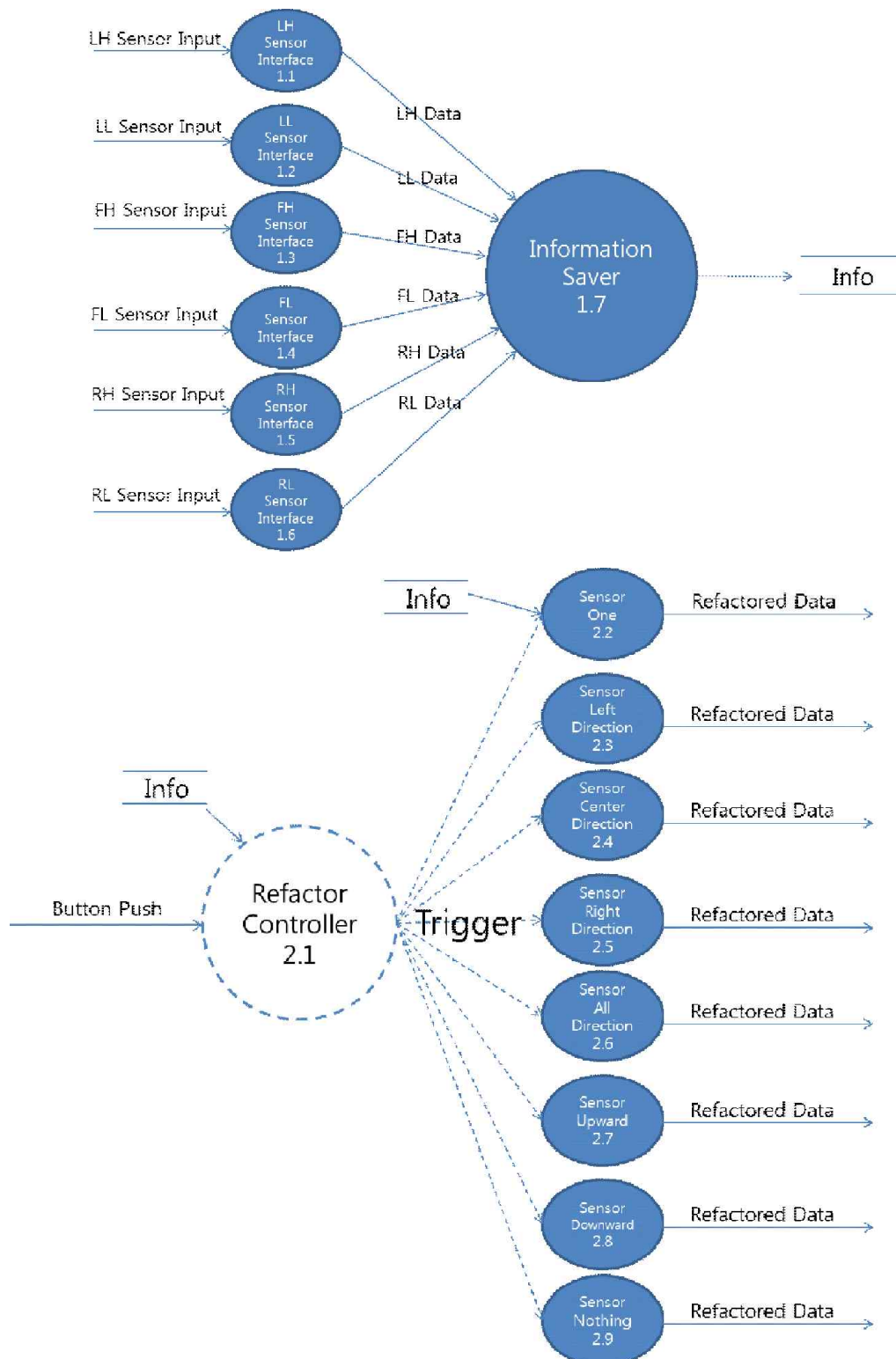
Reference Number	3
Name	Output Control
Input	Info, Refactored Command
Output	Sound Out, Vibe Out
Process Description	정기적으로 Info의 값에 따라 알맞은 Sound Out과 Vibe Out을 해주며 Refactor_Command가 입력 될 때는 모든 상황을 잠시 멈추고 Refactor_Command의 값에 따라서 조합에 맞는 Sound Out을 해준다.

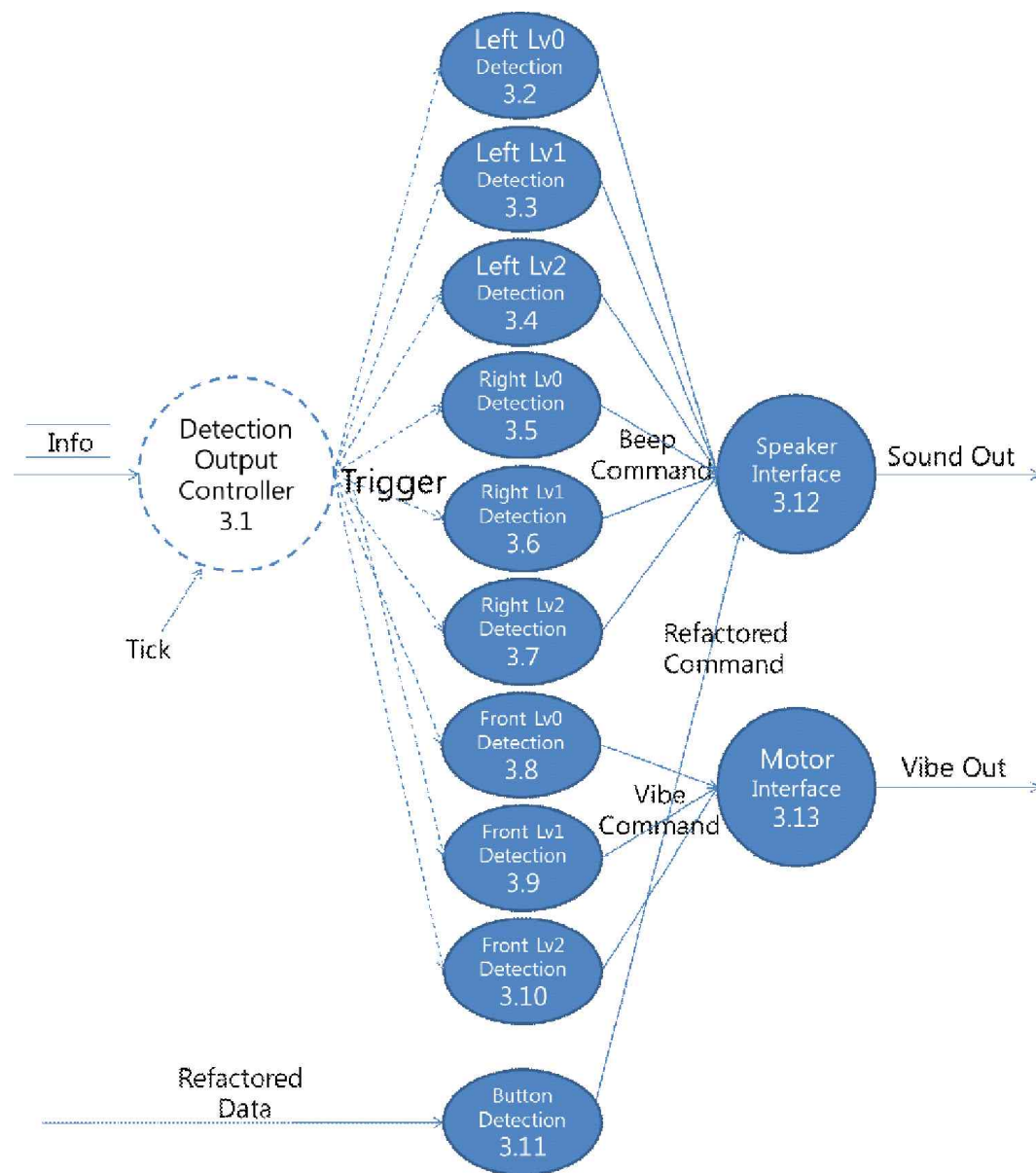
5.2.2.3. Data Dictionary

Input / Output Event	Description	Format / Type
Info	LH Sensor Input, LL Sensor Input, FH Sensor Input, FL Sensor Input, RH Sensor Input, RH Sensor Input중에서 장애물이 인지된 센서 이름과 장애물의 거리 단계를 Detection 1로 부터 전달받아 저장한다.	Dictionary, Periodic
Refactored Data	Refactor 2로 부터 가공된 7단계의 정보 중 하나를 입력 받는다.	Int, Interrupt

5.2.3. DFD Level 2

5.2.3.1. DFD





5.2.3.2. Process Specification

5.2.3.2.1. Process 1.1

Reference Number	1.1
Name	LH Sensor Interface
Input	LH Sensor Input
Output	LH Data
Process Description	정기적으로 LH Sensor로 부터 넘어오는 정보를 LH Data에 저장해 주는 역할을 한다.

5.2.3.2.2. Process 1.2

Reference Number	1.2
Name	LL Sensor Interface
Input	LL Sensor Input
Output	LL Data
Process Description	정기적으로 LL Sensor로 부터 넘어오는 정보를 LL Data에 저장해 주는 역할을 한다.

5.2.3.2.3. Process 1.3

Reference Number	1.3
Name	FH Sensor Interface
Input	FH Sensor Input
Output	FH Data
Process Description	정기적으로 FH Sensor로 부터 넘어오는 정보를 FH Data에 저장해 주는 역할을 한다.

5.2.3.2.4. Process 1.4

Reference Number	1.4
Name	FL Sensor Interface
Input	FL Sensor Input
Output	FL Data
Process Description	정기적으로 FL Sensor로 부터 넘어오는 정보를 FL Data에 저장해 주는 역할을 한다.

5.2.3.2.5. Process 1.5

Reference Number	1.5
Name	RH Sensor Interface
Input	RH Sensor Input
Output	RH Data
Process Description	정기적으로 RH Sensor로 부터 넘어오는 정보를 RH Data에 저장해주는 역할을 한다.

5.2.3.2.6. Process 1.6

Reference Number	1.6
Name	RL Sensor Interface
Input	RL Sensor Input
Output	RL Data
Process Description	정기적으로 RL Sensor로 부터 넘어오는 정보를 RL Data에 저장해주는 역할을 한다.

5.2.3.2.7. Process 1.7

Reference Number	1.7
Name	Information Saver
Input	LH Data, LL Data, FH Data, FL Data, RH Data, RL Data
Output	Info
Process Description	정기적으로 6개의 Data를 정리하여 하나의 정보로 만들어서 Info로 내보낸다.

5.2.3.2.8. Process 2.1

Reference Number	2.1
Name	Refactor Controller
Input	Button Push, Info
Output	Info, Trigger
Process Description	Button Push가 들어오면 실행되며 Info의 정보를 8개의 단계로 나누어 주는 역할을 한다.

5.2.3.2.9. Process 2.2

Reference Number	2.2
Name	Sensor One
Input	Info, Trigger
Output	Refactored Data
Process Description	Refactor Controller로부터 6개의 센서 중에 한 곳에서만 정보가 들어올 때 실행되며 정보를 Refactored Data로 보내준다.

5.2.3.2.10. Process 2.3

Reference Number	2.3
Name	Sensor Left Direction
Input	Trigger
Output	Refactored Data
Process Description	Refactor Controller로부터 6개의 센서 중에 왼쪽에서만 정보가 들어올 때 실행되며 정보를 Refactored Data로 보내준다.

5.2.3.2.11. Process 2.4

Reference Number	2.4
Name	Sensor Center Direction
Input	Trigger
Output	Refactored Data
Process Description	Refactor Controller로부터 6개의 센서 중에 정면에서만 정보가 들어올 때 실행되며 정보를 Refactored Data로 보내준다.

5.2.3.2.12. Process 2.5

Reference Number	2.5
Name	Sensor Right Direction
Input	Trigger
Output	Refactored Data
Process Description	Refactor Controller로부터 6개의 센서 중에 오른쪽에서만 정보가 들어올 때 실행되며 정보를 Refactored Data로 보내준다.

5.2.3.2.13. Process 2.6

Reference Number	2.6
Name	Sensor All Direction
Input	Trigger
Output	Refactored Data
Process Description	Refactor Controller로부터 6개의 센서 모두에서 정보가 들어올 때 실행되며 정보를 Refactored Data로 보내준다.

5.2.3.2.14. Process 2.7

Reference Number	2.7
Name	Sensor Upward
Input	Trigger
Output	Refactored Data
Process Description	Refactor Controller로부터 6개의 센서 중에 상단에서만 정보가 들어올 때 실행되며 정보를 Refactored Data로 보내준다.

5.2.3.2.15. Process 2.8

Reference Number	2.8
Name	Sensor Downward
Input	Trigger
Output	Refactored Data
Process Description	Refactor Controller로부터 6개의 센서 중에 하단에서만 정보가 들어올 때 실행되며 정보를 Refactored Data로 보내준다.

5.2.3.2.16. Process 2.9

Reference Number	2.9
Name	Sensor Nothing
Input	Trigger
Output	Refactored Data
Process Description	Refactor Controller로부터 6개의 센서 중에 아무런 정보가 들어오지 않을 때 실행되며 정보를 Refactored Data로 보내준다.

5.2.3.2.17. Process 3.1

Reference Number	3.1
Name	Detection Output Controller
Input	Info
Output	Trigger
Process Description	Info의 정보를 좌측, 정면, 우측 및 거리 3단계에 따라 9단계로 나누어서 Left Lv0 Detection, Left Lv1 Detection, Left Lv2 Detection, Right Lv0 Detection, Right Lv1 Detection, Right Lv2 Detection, Front Lv0 Detection, Front Lv1 Detection, Front Lv2 Detection으로 보내준다.

5.2.3.2.18. Process 3.2

Reference Number	3.2
Name	Left Lv0 Detection
Input	Trigger
Output	Beep Command
Process Description	좌측의 Level 0에 물체가 있다는 정보를 받아와서 소리 신호를 위한 정보로 변환하여 준다.

5.2.3.2.19. Process 3.3

Reference Number	3.3
Name	Left Lv1 Detection
Input	Trigger
Output	Beep Command
Process Description	좌측의 Level 1에 물체가 있다는 정보를 받아와서 소리 신호를 위한 정보로 변환하여 준다.

5.2.3.2.20. Process 3.4

Reference Number	3.4
Name	Left Lv2 Detection
Input	Trigger
Output	Beep Command
Process Description	좌측의 Level 2에 물체가 있다는 정보를 받아와서 소리 신호를 위한 정보로 변환하여 준다.

5.2.3.2.21. Process 3.5

Reference Number	3.5
Name	Right Lv0 Detection
Input	Trigger
Output	Beep Command
Process Description	우측의 Level 0에 물체가 있다는 정보를 받아와서 소리 신호를 위한 정보로 변환하여 준다.

5.2.3.2.22. Process 3.6

Reference Number	3.6
Name	Right Lv1 Detection
Input	Trigger
Output	Beep Command
Process Description	우측의 Level 1에 물체가 있다는 정보를 받아와서 소리 신호를 위한 정보로 변환하여 준다.

5.2.3.2.23. Process 3.7

Reference Number	3.7
Name	Right Lv2 Detection
Input	Trigger
Output	Beep Command
Process Description	우측의 Level 2에 물체가 있다는 정보를 받아와서 소리 신호를 위한 정보로 변환하여 준다.

5.2.3.2.24. Process 3.8

Reference Number	3.8
Name	Front Lv0 Detection
Input	Trigger
Output	Vibe Command
Process Description	정면의 Level 0에 물체가 있다는 정보를 받아와서 진동 신호를 위한 정보로 변환하여 준다.

5.2.3.2.25. Process 3.9

Reference Number	3.9
Name	Front Lv1 Detection
Input	Trigger
Output	Vibe Command
Process Description	정면의 Level 1에 물체가 있다는 정보를 받아와서 진동 신호를 위한 정보로 변환하여 준다.

5.2.3.2.26. Process 3.10

Reference Number	3.10
Name	Front Lv2 Detection
Input	Trigger
Output	Vibe Command
Process Description	정면의 Level 2에 물체가 있다는 정보를 받아와서 진동 신호를 위한 정보로 변환하여 준다.

5.2.3.2.27. Process 3.11

Reference Number	3.11
Name	Button Detection
Input	Refactored Data
Output	Refactored Command
Process Description	Refactored Data를 정리하여 Refactored Command로 변환하여 준다.

5.2.3.2.28. Process 3.12

Reference Number	3.12
Name	Speaker Interface
Input	Beep Command, Refactored Command
Output	Sound Out
Process Description	Beep Command와 Refactored Command에 따라 정해진 소리로 변환하여 소리를 출력해 준다.

5.2.3.2.29. Process 3.13

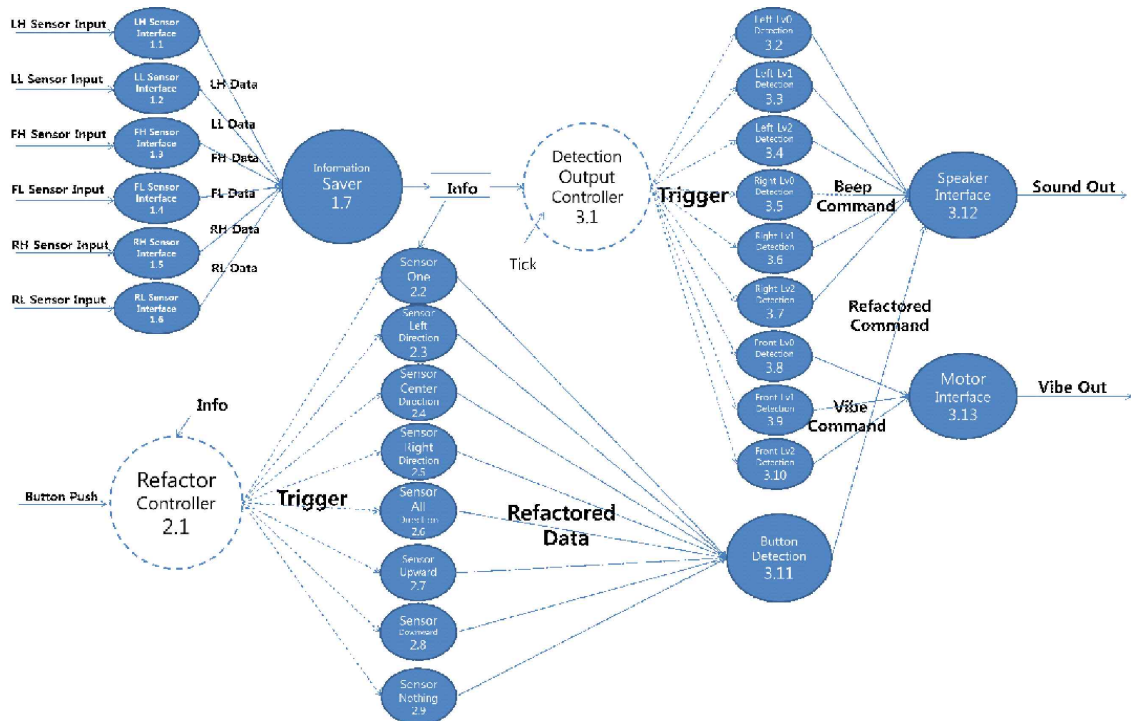
Reference Number	3.13
Name	Motor Interface
Input	Vibe Command
Output	Vibe Out
Process Description	Vibe Command에 따라 정해진 진동으로 변환해 주어서 Vibe Out 으로 보내내 준다.

5.2.3.3. Data Dictionary

Input / Output Event	Description	Format / Type
LH Data	좌측 상단의 센서의 정보를 정기적으로 Information Saver로 보내준다.	Int , Periodic
LL Data	좌측 하단의 센서의 정보를 정기적으로 Information Saver로 보내준다.	Int , Periodic
FH Data	정면 상단의 센서의 정보를 정기적으로 Information Saver로 보내준다.	Int , Periodic
FL Data	정면 하단의 센서의 정보를 정기적으로 Information Saver로 보내준다.	Int , Periodic
RH Data	우측 상단의 센서의 정보를 정기적으로 Information Saver로 보내준다.	Int , Periodic
RL Data	우측 하단의 센서의 정보를 정기적으로 Information Saver로 보내준다.	Int , Periodic

Input / Output Event	Description	Format / Type
Beep Command	Left Lv0 Detection, Left Lv1 Detection, Left Lv2 Detection, Right Lv0 Detection, Right Lv1 Detection, Right Lv2 Detection 으로부터 정보를 받아와서 Speaker Interface로 보내준다.	Dictionary , Periodic
Vibe Command	Front Lv0 Detection, Front Lv1 Detection, Front Lv2 Detection 으로부터 정보를 받아와서 Motor Interface로 보내준다.	Dictionary , Periodic

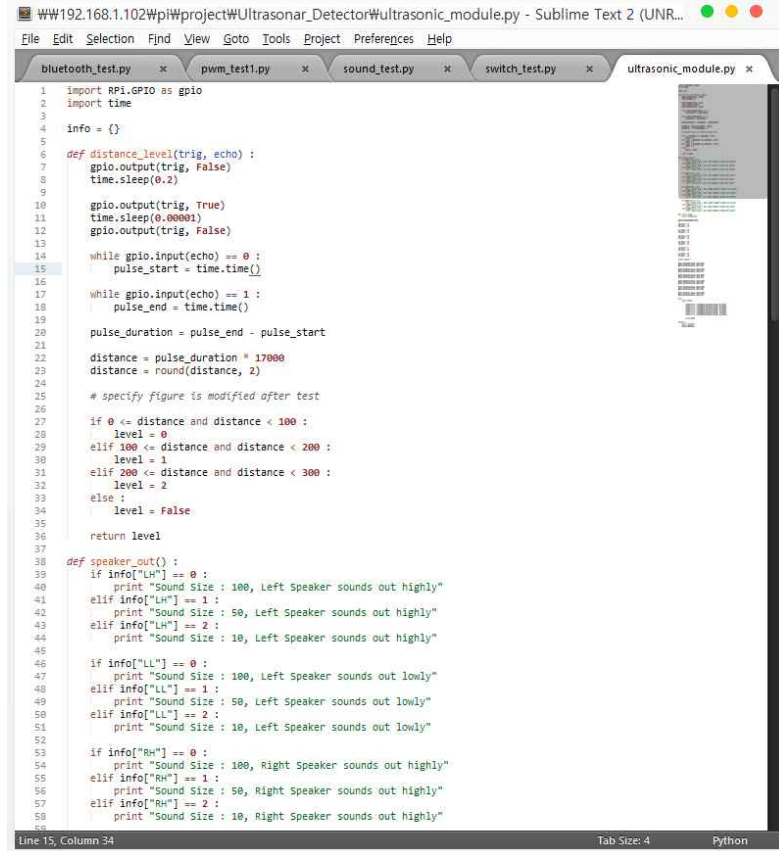
5.2.4. Overall DFD



6. 구현 진행 현황

6.1. 초음파 센서 및 블루투스

현재 초음파 센서와 블루투스 작동에 대한 구현을 완료함.



```
1 import RPi.GPIO as gpio
2 import time
3
4 info = {}
5
6 def distance_level(trig, echo):
7     gpio.output(trig, False)
8     time.sleep(0.2)
9
10    gpio.output(trig, True)
11    time.sleep(0.00001)
12    gpio.output(trig, False)
13
14    while gpio.input(echo) == 0:
15        pulse_start = time.time()
16
17    while gpio.input(echo) == 1:
18        pulse_end = time.time()
19
20    pulse_duration = pulse_end - pulse_start
21
22    distance = pulse_duration * 17000
23    distance = round(distance, 2)
24
25    # specify figure is modified after test
26
27    if 0 <= distance and distance < 100:
28        level = 0
29    elif 100 <= distance and distance < 200:
30        level = 1
31    elif 200 <= distance and distance < 300:
32        level = 2
33    else:
34        level = False
35
36    return level
37
38 def speaker_out():
39     if info["LH"] == 0:
40         print "Sound Size : 100, Left Speaker sounds out highly"
41     elif info["LH"] == 1:
42         print "Sound Size : 50, Left Speaker sounds out highly"
43     elif info["LH"] == 2:
44         print "Sound Size : 10, Left Speaker sounds out highly"
45
46     if info["LL"] == 0:
47         print "Sound Size : 100, Left Speaker sounds out lowly"
48     elif info["LL"] == 1:
49         print "Sound Size : 50, Left Speaker sounds out lowly"
50     elif info["LL"] == 2:
51         print "Sound Size : 10, Left Speaker sounds out lowly"
52
53     if info["RH"] == 0:
54         print "Sound Size : 100, Right Speaker sounds out highly"
55     elif info["RH"] == 1:
56         print "Sound Size : 50, Right Speaker sounds out highly"
57     elif info["RH"] == 2:
58         print "Sound Size : 10, Right Speaker sounds out highly"
59
60
```

6.2. 진동센서

현재 진동센서의 구현을 완료함.



```
1 import RPi.GPIO as GPIO
2 import time
3
4 print "PWM : Pulse Width Module Example"
5
6 GPIO.setmode(GPIO.BCM)
7 GPIO.setup(18, GPIO.OUT)
8
9 p = GPIO.PWM(18,1)
10
11 p.start(10)
12
13 raw_input("Press any key to change frequency (2Hz):")
14
15 p.ChangeFrequency(2)
16
17 raw_input("Press any key to change frequency (5Hz):")
18
19 p.ChangeFrequency(5)
20
21 raw_input("Press any key to quit:")
22 p.stop()
23
24 print "GPIO.cleanup()"
25 GPIO.cleanup()
26
```

7. 개발 계획

	주요내용	추진 일정			
		~10.23	~10.31	11.1~8	11.11~13
1	기능별 구현 완료				
2	디자인 완성				
3	Test & Debug				
4	졸업작품 전시회				

8. 참고문헌

시각장애인 보행편의를 위한 요구사항 분석 연구

www.kbufac.or.kr/DataTech/DownLoad/59?preFix=file&pos=1

Software Modeling & Analysis (SASD)

<http://dslab.konkuk.ac.kr/Class/2010/10SMA/Team%20Project/5/OOAD%20vs%20SASD.pdf>

Introduction to SASD

<http://dslab.konkuk.ac.kr/Class/2014/14SE/Lecture%20Note/Introduction%20to%20SASD.pdf>

GPIO

http://forum.falinux.com/zbxe/index.php?document_srl=438015&mid=hardware

커널 패닉

https://ko.wikipedia.org/wiki/%EC%BB%A4%EB%84%90_%ED%8C%A8%EB%8B%89

라즈비언 릴리즈 노트

http://downloads.raspberrypi.org/raspbian/release_notes.txt

WiringPI 릴리즈 노트

<https://git.drogon.net/?p=wiringPi;a=summary>

LightBlue 릴리즈 노트

<http://lightblue.sourceforge.net/>

라즈베리파이 설명

https://ko.wikipedia.org/wiki/%EB%9D%BC%EC%A6%88%EB%B2%A0%EB%A6%AC_%ED%8C%8C%EC%9D%B4

초음파 센서의 원리

<http://juke.tistory.com/m/post/194>

진동 모터의 원리

<http://www.newswire.co.kr/newsRead.php?no=288509>

블루투스의 용어 설명

<https://ko.wikipedia.org/wiki/%EB%B8%94%EB%A3%A8%ED%88%AC%EC%8A%A4>

라즈베리파이2 사양

<http://blog.naver.com/roboholic84/220363056151>

hc-sr04 사양

<http://deviceall.blog.me/220378824117>

MB-0412V 사양

<http://www.11st.co.kr/product/SellerProductDetail.tmall?method=getSellerProductDetail&prdNo=1085346644>

모터의 종류

https://www.google.co.kr/url?sa=t&rct=j&q=&esrc=s&source=web&cd=15&cad=rja&uact=8&ved=0CDYQFjAEOAo&url=http%3A%2F%2Ffile10.uf.tistory.com%2Fattach%2F273295465242AFCC26A25E&ei=g9OOVdvIEda68gWS hZ34Cw&usg=AFQjCNHv2psVKgSKI042e0DseEe_QyCl0g&sig2=4GcfvDKZWA B5d-NXx8FUgQ