ISSN:1226-7244 (Print)
ISSN:2288-243X (Online)
논문번호 18-02-04
24
j.inst.Korean.electr.electron.eng.Vol.22,No.2,250~257,June 2018
http://dx.doi.org/10.7471/ikeee.2018.22.2.250

BLE 비콘과 컴퓨터비전을 적용한 스마트 버스 시스템

Smart Bus System using BLE Beacon and Computer Vision

유 민 정*, 이 유 진*★

Minjung You*, Eugene Rhee*★

Abstract

In this paper, a smart bus system that automates public bus traffic payment by applying beacon and computer vision and provides bus route information, real-time location information, getting off alarm is proposed. By using the beacon to recognize busses near the stop and to board the bus to be boarded, this system automatically processes the payment when boarding by using the distance from the beacon and the information provided by the beacon and the face comparison. After the payment processing, the system provides the route information of the boarded bus and the real-time bus location information to the user, and when the user sets an alarm using these informations, the alarm is activated when the bus leaves the bus stop.

요 약

본 논문에서는 비콘과 컴퓨터비전을 적용하여 대중교통 버스 결제 자동화 및 탑승 후의 버스 노선 정보, 실시간 위치 정보 제공, 하차 알람, 하차 벨 작동 자동화를 구현한 스마트 버스 시스템을 제안한다. 비콘을 이용하여 정류소에 근접하는 버스들을 인식하고 탑승하고자 하는 버스를 탑승하게 될 경우 비콘과의 거리와 비콘이 제공하는 정보 및 얼굴 비교를 이용하여 탑승 시 결제를 자동으로 처리한다. 결제 처리 후, 탑승한 버스의 노선 정보와 실시간 버스 위치 정보를 사용자에게 제공하고 해당 정보를 이용하여 사용자가 알람을 설정하면 설정한 하차 정류소 이전 정류소를 버스가 출발할 시 알람이 작동시키고 버스의 하차 벨을 작동시키는 대중교통 버스의 전반적인 자동화 시스템을 구축하였다.

Key words : Bus System, Face Recognition, Beacon, Computer Vision, openAPI

* Dept. of Electronic Engineering, Sangmyung University

★ Corresponding author

E-mail: eugenerhee@smu.ac.kr, Tel: +82-41-550-5413

Manuscript received Jun. 11, 2018; revised Jun. 14, 2018; Accepted Jun. 29, 2018

Ⅰ. 서론

일반적으로 대중교통 버스에 탑승할 시 현금 또는 카드/휴대폰을 태그하여 요금을 지불한다. 즉, NFC 방식을 이용하게 되는데 이는 장치에 직접 접촉 또는 조작해야하기 때문에 미리 꺼내어 준비해야하는 번거로움과 태그 시 태그가 잘 동작되지 않아 뒷사람에게 피해를 주는 경우도 종종 발생한다 [1],[2].

탑승 후에는, 만일 탑승자가 초행길이여서 길에 익숙하지 않을 경우에 매 정류소를 지날 때마다 주의를 기울여야하는 번거로움이 발생하며, 초행길이 아닌 사용자라 하더라도 피곤하여 잠깐 졸았을 때, 또는 다른 무언가에 집중하다가 하차해야할 정류소를 놓치는 경우도 종종 발생한다.

하차 시에는, 버스 벨을 눌러 자신이 하차한다는 표현을 해야 하는데 가끔 사람이 너무 많아 벨을 누르기에 어려운 상황이 생기기도 하고 우연히 앉은 자리 근처에 벨이 없는 경우도 있다. 이런 경우 움직이는 버스에서 일어나 벨을 눌러야하는데 위험한 사고가 발생하기도 한다. 요금 지불에 있어서는 내릴 때 또한 미리 카드를 꺼내야 하는 불편함이 있으며, 종종 어떤 사용자는 본인이 내리지 않음에도 불구하고 하차 태그를 미리 하여 추가 요금을 회피하는 경우도 있다.

이러한 전반적인 불편함 개선과 탑승자에게 편의 제공 및 부정 태그 방지를 위해, 본 논문에서는 BLE Beacon과 Computer Vision을 이용한 스마트 버스 시스템을 제안한다.

Ⅱ. 본론

1. Beacon

Beacon은 기본적으로 스마트 기기의 정보 수신에서 사용된다. 블루투스4.0 버전 이상에서 제공되는 기술로 최대 70 m까지 통신이 가능하다 [3]. 구체적으로 사용되는 방식은 특정 공간에 입장하게 되면 그 공간에 대한 정보를 제공하는 방식, 쇼핑몰과 같은 곳에서의 할인 쿠폰 정보 제공, 센서들 간의 정보 교환 등의 방식으로 활용되고 있다.

Beacon은 한 가지 방식이 아닌 여러 방식으로 개발되었는데, 첫 번째는 iBeacon과 같은 Bluetooth 기반의 BLE를 활용하는 방식, 두 번째는 WiFi 및 GPS를 활용하는 방식, 세 번째는 저주파, 고주파 방식을 활용하는 방식, 마지막으로 네 번째는 빛을 송신/수신하여 통신하는 방식이다 [4].

본 논문에서는 BLE를 활용한 Beacon 방식을 채택하였다. BLE 기반 Beacon은 장치간의 정보 교환, 원격 제어 및 모니터링 등과 같은 분야에서 활용/연구되고 있다 [5]. 이 방식은 iBeacon이라고 불리며 Apple 사에서 처음 제안한 방식이며, 이는 iOS의 자체적인 위치 기반기술을 확장한 기술로 이를 이용해 특정위치와 단말기 사이의 대략적인 위치를 측정할 수 있다 [6]. 16진수 정보를 제공하며 이 정보를 단말에서 수신하여 사용한다. 이의 구성은 아래의 [표 1]과 같다.

Table 1. Beacon frame structure
표 1. 비콘 프레임 구조

Factor	Byte	Explanation
AD flags	3	first AD
AD header	2	second AD
Company ID	2	Apple = 0x4C00
Type ID	1	iBeacon = 0x02
Data Length	1	leftovers payload bytes
UUID	16	device ID
Major Number	2	Classify company, store ect.
Minor Number	2	Classify area
Tx Power	1	signal strength

보통 AD flags, AD header, Company ID, Type ID는 고정적인 값이며 이 이외의 값에 송신하고자 하는 정보를 넣는다.

가. iBeacon 기술 활용

앞서 [표 1]에서의 UUID, Major Number, Minor Number 값을 이용해 버스 고유 정보를 표현한다. 예를 들어 651번 버스의 Beacon 정보는 [그림 1]과 같다.

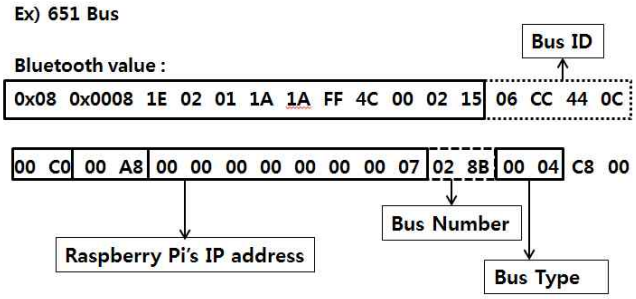

Fig. 1. Structure of Raspberry Pi(Bus) Beacon
그림 1. 라즈베리파이(버스) 비콘의 구조

UUID에 해당하는 부분은 Bus ID와 Raspberry Pi의 IP address 정보를 담고 있다. 우선 [그림 1]에서의 2번째 박스는 Bus ID정보를 담고 있으며 그 다음 3, 4, 5번째 박스는 IP Address 정보를 담고 있다. Major Number부분인 6번째 박스 부분은 버스 번호 정보를 담고 있고 7번째 박스 부분인 Minor Number부분은 버스 타입 정보를 담고 있다.

Bus ID는 이후 설명할 공공 데이터 포털 openAPI 정보에 따른 버스들의 고유 ID정보이며 버스 타입 또한 공공 규칙을 기본으로 따른다. 예를 들어 1은 공항 버스, 2는 마을 버스, 3은 간선 버스, 4는 지선 버스, 5는 순환 버스 등을 의미한다 [7],[8].

구조의 맨 마지막 부분은 거리 정보를 포함한다. 이 모든 정보를 App에서 받아들여 10진수로 변환하여 사용한다.

2. Face Recognition

버스에 여러 명이 탑승할 경우, 버스 내의 센서 입장에서는 A라는 사람이 센서를 통과하였는지, B라는 사람이 통과하였는지 구분하지 못한다. 따라서 이를 구분하기 위해 사람의 얼굴을 이용한 인증 과정을 거친다. 인증의 방법은 그 사람 자체를 인식하여 구분하는 것이 아닌 사진과 사진을 비교하여 그 속에 동일 인물이 존재하는지 만을 검증하여 센서를 통과한 탑승자가 누구인지 인증한다.

기본적으로는 각각의 사진에서 얼굴을 검출하고 그 얼굴을 분석하여 분석 값을 토대로 두 사진 값을 비교하는 것이다. 얼굴을 검출하기 위해서는 이미지의 Gradient를 분석해야한다. 이때 모든 픽셀을 기준으로 Gradient를 검출하는 것이 아닌 16×16과 같이 정사각형으로 검출한다. 그리고 이 검출된 값들을 여러 사진들과 비교하여, 즉 기계학습 시켜 얼굴의 패턴이라 생각되는 것을 얼굴이라 인식한다. 예를 들어 openCV의 경우 밝은 영역의 픽셀 평균값과 어두운 영역의 평균값의 차이를 구하여 이 값이 일정 값을 넘으면 얼굴이라 인식한다 [9],[10].

얼굴 인식 후에는 얼굴의 위치를 교정해야 한다. 예를 들어 측면으로 찍힌 얼굴과 정면으로 찍힌 얼굴을 비교한다면 눈, 코, 입 등의 위치가 다르기 때문에 컴퓨터는 동일 얼굴이 아니라고 인식되게 된다. 따라서 Face Landmark Estimation Algorithm을 이용하여 얼굴의 Landmark(눈 안쪽, 가장자리, 턱, 코 등)를 검출해야 한다. 이후 검출된 Landmark를 같은 위치에 위치시킨다. 예를 들어 코는 무조건 중앙에, 눈은 이미지의 상단 1/3지점에 위치시킨다 [11]-[13].

이러한 방식을 이용하여 학습하게 되고 학습에 따라 얼굴 비교 정확성이 올라가게 된다. Machine Learning을 이용하여 비교하는 방식, 즉 TensorFlow, CNN, dlib, PCA 등을 사용하여 직접 학습시켜 비교하는 방법이 있고 API(openCV, openFace, Kairos verify, MS Azure, Amazon rekognotion 등)를 이용하는 방법이 있는데, 여기서는 후자를 이용하여 Face Recognition을 진행하였다.

3. 자동 결제를 수행하는 스마트 버스 시스템

앞의 Beacon과 Face Recognition의 방식을 응용하여 대중교통 버스의 승/하차 자동 결제 시스템을 구현한다. 이는 Android App과 버스 내의 Raspberry Pi를 기반으로 동작한다.

가. 스마트 버스 시스템 User/Bus 부분 구성도

대중교통 자동화 시스템의 전반적인 구성에 대해 서술한다.

(1) 사용자의 대기 및 탑승

탑승과정 절차는 [그림 2]와 같다.

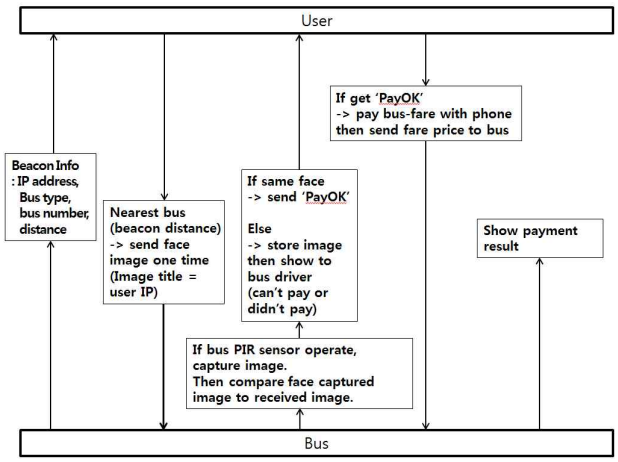

Fig. 2. Flow of Boarding Bus Process
그림 2. 탑승 과정 흐름

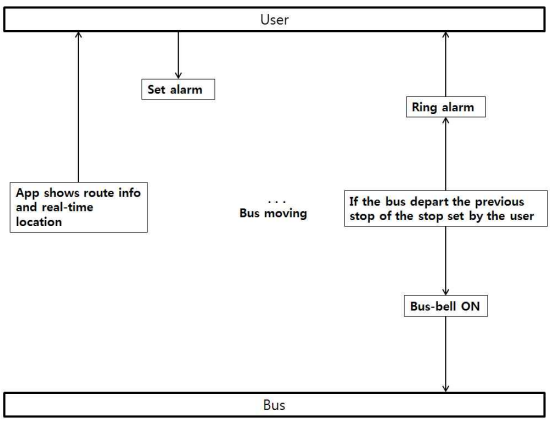

Fig. 3. System Flow after Boarding
그림 3. 탑승 후 시스템 흐름

우선, 버스는 자신들 고유의 정보를 Beacon을 이용해 뿌리고 있다. 사용자들은 정류소에서 버스를 기다리고 있으면 다가오는 버스들의 Beacon 정보를 수신 받게 된다. 이때 수신 받은 Beacon들 중 가장 가까운 Beacon의 IP정보를 이용하여 미리 App에 찍어 놓은 자신의 얼굴 이미지를 한번 전송한다. 버스는 전송받은 이미지를 접속한 IP Address를 파일 명으로 하여 Unpay Directory에 저장한다.

사용자는 본인이 탑승해야할 버스에 탑승한다. 탑승 시 버스에 있는 센서가 동작하게 되고 센서 동작 시 이미지를 캡처한다. 이때 캡처한 이미지와 사용자가 이전에 전송한 얼굴 이미지를 비교하여 두 이미지 속의 인물이 일치하는지 검사하고 만일 일치한다면 버스는 사용자의 App에 'PayOK'라는 메시지를 전송한다. 사용자가 'PayOK'라는 메시지를 수신하면 결제를 진행하고 결제 완료 시 결제 금액을 버스에 전송한다. 버스는 결제 금액을 수신하면 모니터에 결제 완료 사인을 보여준다. 만일 수신된 이미지들과 캡처한 이미지 중 일치하는 얼굴이 없을 경우 해당 캡처된 이미지는 별도로 저장되고 모니터에 미결제 인수로 카운트되며 미결제자 이미지는 버스 운전기사에게 제공된다.

(2) 탑승 후의 시스템

사용자가 결제를 완료한 후에 제공하는 시스템의 절차는 [그림 3]과 같다.

사용자의 App에는 탑승한 버스의 노선도와 버스의 현재 위치 정보가 출력된다. 위치 정보는 Real-Time으로 제공된다. 제공된 버스 노선도 중, 하차하고자 하는 정류소를 클릭하면 알람이 설정되고 버스가 설정된 정류소의 이전 정류소를 출발하게 되면 알람이 울리고 버스 하차 벨이 켜지게 된다.

(3) 사용자의 하차와 환승

사용자가 하차할 시, 그리고 만일 환승하게 될 경우의 절차는 [그림 4]와 같다.

하차시의 결제 과정은 승차 시의 결제과정과 동일하다. 사용자가 하차를 완료하면 App에는 사용자가 어디서 승차였고 어디서 하차하였는지에 대한 정보가 저장된다. 그리고 총 결제 금액과 저장된 정보를 출력한다. 만일 사용자가 환승을 해야 하는 경우는 App의 첫 페이지로 이동하고 더 이상 대중교통을 이용할 필요가 없으면 App을 종료시킨다.

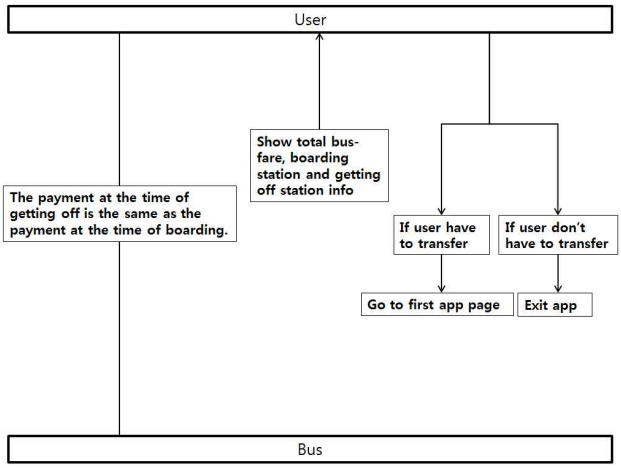

Fig. 4. System Flow after Getting off
그림 4. 하차 후 시스템 흐름

나. Android App의 구성 (User)

본 논문에서 제안하는 시스템은 Android App을 이용해 구현하였다 [14]. App의 전반적인 구성과 구성 시 사용한 상세 기술들은 다음과 같다. 우선 App의 작동 과정은 [그림 5]와 같다.

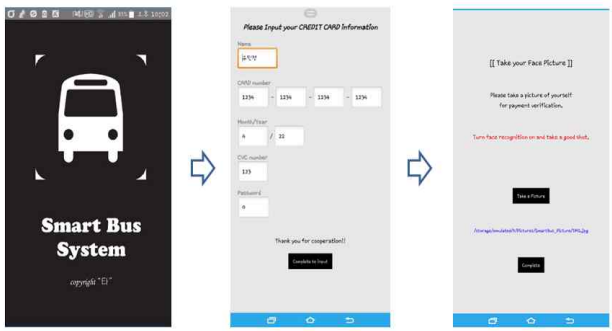

Fig. 5. First Flow of Android App
그림 5. 첫 번째 안드로이드 앱 흐름

App을 실행시키면 우선 결제할 카드 정보를 입력한다. 이 정보는 기본적으로 한번만 입력하면 App에 저장되어 다음번 실행 시에는 이 페이지를 뛰어넘게 된다. 다음으로는 본인의 얼굴 이미지를 저장한다. 'Take a Picture' 버튼을 클릭하게 되면 카메라로 연결되어 얼굴 사진을 찍을 수 있게 되고 찍은 후 저장되면 저장 경로를 출력해준다. 이미지는 카드 정보 입력 시와 마찬가지로 기본적으로 한번만 찍으면 되며 사진이 해당 경로에 저장되어 있을 경우 다음번 실행 시 이 페이지는 뛰어넘게 된다.

정보 입력을 모두 마치게 되면 사용자가 대기 정류소에서 보는 화면으로 넘어가게 된다. 이 화면은 [그림 6]과 같다.

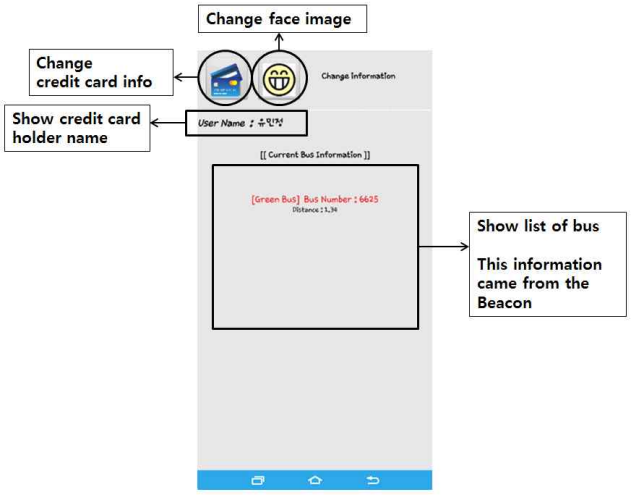

Fig. 6. Second Flow of Android App, Showing Bus Information
그림 6. 두 번째 안드로이드 앱 흐름, 버스 정보 출력

저장한 정보를 변경할 수 있는 2개의 이미지 버튼이 상단에 위치하며 각각의 버튼을 클릭하게 되면 앞서 정보를 입력했던 페이지로 이동한다. 이미지 버튼 바로 밑에는 카드 정보 입력 시 입력했던 카드 소유자명이 출력된다.

중앙에는 정류소에 접근하는 버스의 정보들, 버스 타입(Green Bus, Blue Bus, Red Bus 등), 버스 번호, 버스와의 대략적인 거리 정보가 출력된다. 여러 버스들이 정류소에 도착하면 도착한 버스들 모두가 출력된다.

이후 사용자가 버스 요금을 지불하게 되면 [그림 7]과 같은 메시지 박스가 뜨게 되고 'OK' 버튼을 클릭하게 되면 탑승한 버스의 번호, 버스 노선 정보와 버스의 현재 위치가 출력된다.

버스의 현재 위치는 5 초에 한 번씩 자동 갱신된다. 노선 정보는 정류소 ID, 정류소명, 정류소별 막차 시간 및 첫차 시간을 출력한다. 버스 아이콘은 버스의 차량 번호를 출력한다.

버스 아이콘은 움직이며 텍스트 색상이 파란색인 경우는 버스가 움직이고 있음을 의미하며 텍스트 색상이 빨강인 경우는 버스가 해당 정류소에 도착하여 잠시 멈춰있음을 의미한다.

사용자가 알람을 설정한 경우, 즉 하차하고자 하는 정류소를 선택한 경우 정류소 텍스트 색상은 보라색으로 바뀐다.

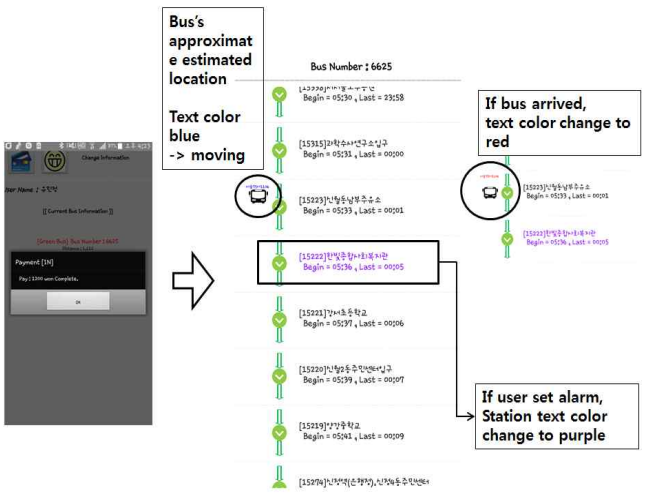

Fig. 7. Information of Bus Route, Location and Alarm Setting
그림 7. 버스 경로와 위치 정보 그리고 알람 설정

버스가 설정한 정류소의 이전 정류소를 출발하게 되면 알람이 5 초 간격으로 울리게 되고 알람 메시지 박스의 'OK' 버튼을 클릭하게 되면 버스 하차 벨이 켜진다.

사용자가 하차하게 되면 하차 요금을 결제하고 승차 정류소와 승차 시 지불한 요금, 하차 정류소와 하차 시 지불한 요금, 그리고 이들의 총 금액을 출력한다.

페이지 하단에는 사용자의 이전 탑승 정보, 어떤 버스를 탑승하였고 어디서 승차하고 하차하였는지에 대한 리스트 정보를 출력한다. 과정은 [그림 8]과 같다.

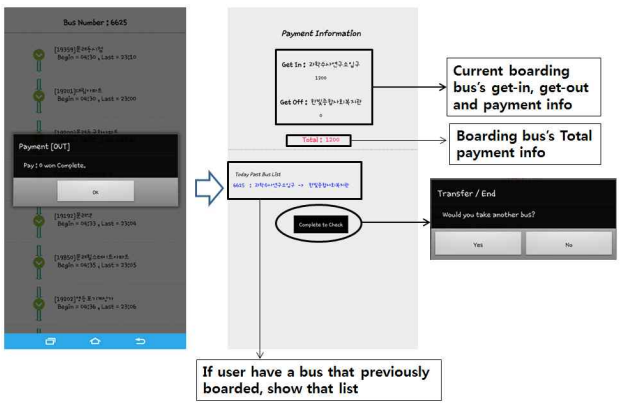

Fig. 8. Get-off-fare Payment, Show Payment and Previous Information
그림 8. 하차 요금 지불, 결제 정보와 이전 탑승정보 출력

이후 'Complete to Check' 버튼을 클릭하게 되면 메시지 박스가 뜨게 되고 'Yes' 클릭 시 App의 처음 화면(Beacon 리스트 출력 화면)으로 이동하고 'No' 클릭 시 App을 종료한다. 이때 App은 내부적으로 마지막 하차 시간과 버스 번호를 저장하여 환승 시 결제 금액 설정에 이용된다.

다. Raspberry Pi의 구성 (Bus)

각각의 버스들은 Raspberry Pi를 내장하고 운행하게 된다. 이것의 기능은 Beacon, TCP/IP 통신, Pi Camera를 이용한 캡처 기능, 센서 및 LED 제어, Face Recognition과 Compare Face 기능, 결제 정보 LCD 출력 등을 담당한다. Raspberry Pi3 B를 사용하였고 Pi Camera와 7 inch touch LCD, PIR 센서(NT0061), SMD RGB(E4-P30), 어댑터로 구성된다.

우선 Raspberry Pi는 기본적으로 Beacon 숙주이며 계속해서 정보를 내보낸다. 본인의 IP Address, 버스 타입, 버스 번호, 거리 정보가 이에 해당된다.

다음으로 Raspberry Pi에는 3개의 프로그램이 돌아가고 있다. 첫 번째는 Android App에서 이미지를 수신 받고 저장하는 기능을 담당하며, 두 번째는 PIR 센서 동작 시 캡처하고 그 캡처 이미지와 수신 받은 이미지의 얼굴을 비교하여 결과를 처리하는 기능, 마지막 세 번째는 App에서 오는 버스 하차 벨 시그널을 받아 벨을 동작시키는 기능을 수행한다. 이의 과정은 [그림 9]와 같다.

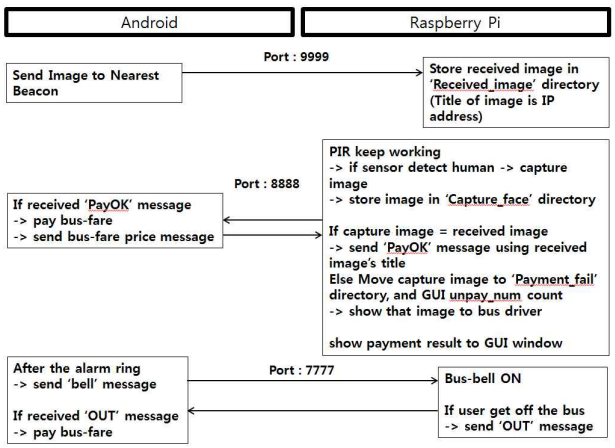

Fig. 9. Socket Communication between App and Raspberry Pi
그림 9. 라즈베리파이와 앱 간의 소켓 통신

3 가지 프로그램은 통신 포트 9999, 8888, 7777을 사용하며 TCP/IP 통신으로 이루어진다. TCP/IP 통신은 UDP와는 다르게 Hand Shaking을 제공하며 연결지향형이고 패킷의 신뢰성을 일정부분 제공하기 때문에 선택하였다 [15].

Port 9999는 java TCP 통신으로 이뤄지며 App이 클라이언트, Raspberry Pi가 서버 역할을 하고 이의 결과는 [그림 10]과 같다.

Fig. 10. Socket Communication Port 9999
그림 10. 포트 9999 소켓 통신

Port 8888은 Python TCP 통신으로 이루어진다. Raspberry Pi가 클라이언트, App이 서버 역할을 하며 Raspberry Pi의 경우 Face Compare Similarity에 따라 통신여부를 결정한다. 그리고 수신 받은 요금 정보와 Similarity의 결과 실패되었을 경우의 전체 인수 정보를 Python GUI를 통해 출력한다. Python GUI의 3가지 모드와 통신 결과는 [그림 11]과 같다.

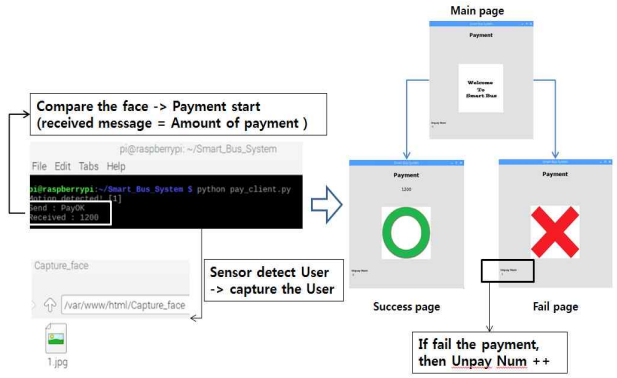

Fig. 11. Socket Communication Port 8888
그림 11. 포트 8888 소켓 통신

Port 7777은 C의 TCP 통신으로 이루어지며 App이 클라이언트, Raspberry Pi가 서버 역할을 한다. 통신 결과는 [그림 12]와 같다.

이러한 프로그램은 버스가 운행되는 동안 반복된다. 그리고 이미지정보는 하나의 정류소를 출발한 시점부터 다음 정류소 도착 시까지 유효하다.

Fig. 12. Socket Communication Port 7777
그림 12. 포트 7777 소켓 통신

라. openAPI 기반 버스노선 및 위치 정보

전체적인 시스템의 절반 부분은 openAPI가 담당하고 있다. App에서 제공하는 버스의 노선 정보, 실시간 위치 정보는 정부기관 openAPI에서 제공한 정보를 받아와 사용한다. 이들 정보는 App 내의 Thread에서 5 초 간격으로 돌아가며 버스 그림을 통하여 사용자가 인식하기 쉽도록 설계되었다.

사용된 openAPI는 www.data.go.kr에서 제공하는 정보인 서울특별시 노선정보 조회와 서울특별시 버스위치 정보 조회를 이용하였다 [7],[8]. 우선 버스 번호에 따른 노선 정보를 검색하고 노선 ID를 얻은 후 해당하는 노선 리스트를 출력한다. 그리고 버스 ID를 이용해 버스의 위치를 얻어 그려진 노선 정보와 비교해 해당 위치에 버스를 그린다.

III 결론

이 시스템을 이용함으로써 사용자의 편의가 증가하는 것뿐만 아니라 서론에서 언급되었 듯이 사용자의 부정 태그 또한 방지할 수 있다. 센서를 통과하여야 결재를 진행할 수 있기 때문이다. 일반적으로 현금으로 부정 결재하는 경우가 절반 이상을 차지하는 것은 사실이지만 하차 전 태그도 그리 낮지 않은 현실이다.

본 논문에서는 서울시 버스를 기준으로 스마트 버스 시스템을 모델링하였지만, 해당 버스 뿐만 아니라 여러 분야에서도 활용가능 할 것이라 판단된다. 예를 들어 매표소에서 표를 구매하기 위해 줄을 서야하는 번거로움과 대기 줄이 길 경우 시간 지연의 불편함도 발생한다. 이러한 불편함은

표를 확인하는 곳에 본 논문에서 제안하는 시스템을 이용하여 자동 결제를 진행하게 됨으로서 감소시킬 수 있다. 관광지뿐만 아니라 영화관, 놀이동산 등에서도 충분히 활용할 수 있다.

본 시스템을 확장시켜 주로 탑승하는 버스들과 하차 정류소 등의 정보를 저장하여 사용자가 따로 알람을 설정하지 않더라도 자동으로 설정되는 서비스를 제공할 수 있을 것이며, 카드 정보 입력 시 해당 카드 소유자와 입력을 진행하는 얼굴을 대조하여 해당 인물만이 카드 정보를 입력할 수 있는 서비스를 제공한다면, 즉 버스 탑승 시 얼굴 인증을 진행하기 때문에 부정 카드 이용을 상당 부분 감소시킬 수 있을 것으로 판단된다. 또한 더 나아가 해당 장소에서의 실시간 통계정보 수집과 이를 바탕으로 한 특정 서비스 제공에도 활용될 수 있을 것이다.

References

[1] H. S. Ryu, S. H. Park, and Y. B. Jang, “Communication Technology Trends between Terminals for Proximity Service,” *Inf. & Commun. Mag.*, Vol. 30, No. 12, pp. 97-104, November 2013.

[2] H. S Kang and I. S. Koo, "Beacon Node Based Localization Algorithm Using Received Signal Strength and Path Loss Calibration for Wireless Sensor Network," *The Journal of The Institute of Webcasting, Internet and Telecommunication*, Vol. 11, No. 1, 2011.

[3] Craig Gilchrist, *Learnig iBeacon*, PACKT, November 2014

[4] H. T. Kim, *On-demand beyond O2O*, e-bizbooks, 2016.

[5] K. Y. Kim, S. Y. Shin, K. S. Bae, and S. Chae, “Design and Implementation of NMEA 2000 based Universal Gateway,” *J. KICS*, Vol. 39, No. 2, pp. 191-198, 2014. DOI: 10.7840/kics.2014.39C.2.191

[6] M. S. Gast, B*uilding Applications with iBeacon*, O’REILLY, 2015

[7] Seoul Topis, “Seoul Bus Route Information Inquiry Service,” https://www.data.go.kr/dataset/15000193/openapi.do

[8] Seoul Topis, “Seoul Bus Location information Inquiry Service,” https://www.data.go.kr/dataset/15000332/openapi.do

[9] K. Fukunaga, "Introduction to Statistical Pattern Recognition," *Academic Press, Orlando*, FL., 1972.

[10] S. Shin, *Emgu CV Essentials*, PACKT, November 2013.

[11] G. W. Song, “Face Recognition Technology and Trend,” *Korea Multimedia Society*, Vol. 7, No. 2, pp. 1-8, 2016.

[12] R. Brunelli and T. Poggio, “Face Recognition : Feature versus Templates,” *IEEE Trans, Pattern Analysis and Machine Intelligence*, Vol. 15, No. 10, 1993. DOI: 10.1109/34.254061

[13] K. Swets and T. Poggio, “Example based Learning for View based Human Face Detection,” *IEEE Trans, PAMI*, 20, pp. 39-51, 1998. DOI: 10.1109/34.655648

[14] J. Woo and G. S. Park, *Android Programming Using Android Studio*, Hanbit Academy, 2016.

[15] J. F. Kurose and K. W. Ross, *Computer Networking a Top-down Approach*, Pearson, 2017.

BIOGRAPHY

Minjung You(Member)

2015~present : Undergraduate in Electronic Engineering, Sangmyung University.

Eugene Rhee(Member)

2001 : BS degree in Electronic Engineering, Hanyang University.
2003 : MS degree in Electronic Engineering, Hanyang University.
2010 : PhD degree in Electronic Engineering, Hanyang University.
2012~present : Assistant Professor, Sangmyung University.