**IoT 프로젝트 결과보고서**

조/작성자 : 7조

작성일자 : 2021.08.24

1. **개요**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ◦ 제 목 : SMART 버스 정류장  ◦ 개발 기간 : 2021.07.05 ~ 2021.08.25  ◦ 조원 및 담당역할 :   |  |  | | --- | --- | | **조원** | **역할** | | 김승현 | 앱을 통한 승차 알림 요청 구현  버스 이동 구현  실제 모형 및 회로 제작 | | 강걸 | LED- 조도 센서 측정 값에 따라 밝기 조절  환풍기- 적외선 센서로 ON/OFF 구현 | | 안호권 | 화재 경보기- 온도 센서와 파이어베이스를 연동하여 구현  에어컨- 적외선 센서로 에어컨 ON/OFF 구현 | | 임도현 | 대기자 유무 판단 – 적외선 센서로 구현  파트 별 코드 통합 및 파이버 베이스 연동 | | 정다솔 | 아두이노 – 파이어베이스간 통신 구현  버스정류소 대기자 알림 장치 및 버저 알림 구현  버스 대기자 알림 구현 | |

1. **개발 목표 달성 수준**

|  |
| --- |
| **2.1 개발 목표**  경기도 연구소 보고서에 따르면 현존하는 밀폐형 버스 정류장 서비스의 경우 다음과 같은 문제점들을 가지고 있다. 먼저 지금의 밀폐형 버스 정류장의 경우 통풍과 냉난방 관리가 어려웠다(불필요한 전력 낭비 발생). 또한 밀폐형 버스정류장을 활용하는 대기자의 경우 칸막이로 인한 시각 정보 제한으로 인하여 자신이 이용하고자 하는 버스를 제대로 식별하기 어려웠다. 이와 더불어 승객들이 대기 부스 안에 있는 기존의 밀폐형 버스 정류장의 특성상 늦은 밤에 버스 정류장에 사람이 대기하고 있을 경우, 버스기사가 대기자를 제대로 인지하지 못하여 불필요한 정차를 하거나, 대기자가 있음에도 불구하고 대기자를 지나치는 경우도 발생하였다. 이에 우리 조의 경우 IOT를 활용하여 현존하는 밀폐형 버스 정류장의 단점들을 해결할 수 있는 방안들을 마련했다. 프로젝트 개발 세부 목표는 다음과 같다.  **1.** **밀폐형 버스 정류장 내 적외선 센서를 활용한 대기자 유무 파악.**   -    **개발목표:** 기존 버스정류소의 불필요한 전력낭비 문제 해결   * 대기자의 유무에 맞춰 냉난방기, 통풍장치 및 조명장치의 ON/OFF 결정   **2.**  **조도센서와 온도센서를 활용한 효율적인 냉난방 및 조명장치 구동**   -    **개발목표:** 기존 버스정류소의 불필요한 전력낭비 문제 해결   * 외부 밝기를 상시 측정하여 내 외부 조명의 효율적 관리         -    대기부스 내 온도를 상시 측정하여 공공기관에서 제시하는 적정온도 수준으로              부스온도 유지  **3**.  **푸시 버튼 및 클라우드와 스마트폰 앱을 통한 버스-승객 간 통신**.         -  **개발목표:** 기존 버스정류소의 불필요한 정차 문제를 해결하기 위한 대기자 알림 장치 구현   * 앱 및 푸시 버튼을 활용하여 대기자가 이용하려는 버스 정보를 클라우드에 전송 * 클라우드에 데이터가 입력 시 해당 버스에 정차 신호 전달   **4.  버저 및 전광판을 통한 대기자 유무 알림**   * **개발목표:** 기존 밀폐형 버스정류의 제한적인 시인성 문제 해결 * 버스 정류소의 전광판을 활용한 대기자가 유무 알림 * 승강장 내 대기자가 타고자 하는 버스가 정류장에 도착했을 때, 버저를 통하여 대기자에게 알림        위의 핵심 개발 목표 4가지에 기반한 우리 조의 프로토타입은 다음과 같다.    <그림 1> Smart Bus Station 프로토타입  **[1] Device (WeMos & Sensors)**  Smart bus station을 구현하기 위한 구체적인 장비 및 센서들의 활용방안은 다음과 같다.  **1) 대기자 인식 (적외선 센서)**     - 적외선 센서를 통해 대기 공간 내 대기자 유무를 파악.  - 대기자가 있을 경우, 냉난방기 및 조명 장치 작동.  - 대기자가 일정 시간 이상 없을 경우, 환풍기 작동 및 조명, 냉난방 장치 절전.  **2) 온도 조절 및 화재 알림 (온도 센서)**  - 대기 공간 내 온도를 실시간으로 측정하여 냉난방기를 통해 적정 온도 유지.  - 일정 온도 초과 시 화재 알림 작동  **3) 조명 조절 (조도 센서)**  - 일출/일몰 및 주변 환경에 따른 밝기를 인식하여 내부 및 외부 조명 조절.  **4) 대기자 알림 (데이터 베이스)**  - 대기자가 타고자 하는 버스를 앱 및 푸시 버튼을 통해 클라우드에 전송.  - 클라우드의 데이터를 통해 정차가 필요한 버스에 정차 신호 전달.  - 추가적으로 대기 공간 내 대기자가 있음을 전광판을 통해 알림.  **5) 버스 도착 알림 (버저)**  - 버스가 정류장에 도착했을 때 버저를 통해 대기자에게 알림.  **[2] Cloud (Firebase)**     - 밀폐형 버스 정류장의 상태(온도, 조도, 대기자 유무)의 데이터를 수신.     - 버스 정류장의 냉난방 장치, 조명, 환풍기의 동작 여부 확인.     - 앱을 통해 대기자로부터 받은 신호를 대기자가 타고자 하는 버스에 전달.  **[3] Application (AppInventor)**     - 사용자의 핸드폰 위치센서를 통해 사용자의 위치를 확인하여, 일정 거리 내의 정류장  에 대해서만 승차 알림 요청 기능 활성화   * 선택한 정류장의 버스 목록에서 타고자 하는 버스를 선택하여 알림 요청 * 요청을 한 상태에서 다른 버스에 알림 요청 시, 기존 요청을 해제 * 이미 선택한 버스를 다시 선택 시, 요청 해제 * 선택한 버스가 이전 정류장을 출발하면 “잠시 후 도착” 알림 * 선택한 버스가 선택한 정류장에 도착하면 “도착 또는 출발” 알림 * 다수의 이용자가 동시에 사용 가능   **2.2 목표 달성 수준**  **1)현재 제품 및 기술 현황**  - 현재 존재하는 밀폐형 버스 정류장에는 버스 대기자의 유무를 고려하지 않아 통풍과 냉난방의 효율적인 관리가 어려움.  - 버스 기사가 대기자를 인식하기 어려워 불필요한 정차문제 발생.  - 대기자가 이용하고자 하는 버스 정보 획득이 어려움.  **2)기존 제품과의 차별점**   -  대기자 인식: 적외선 센서를 통해 대기자가 있을 경우에만 내부 시설 가동(냉난방, 환풍시설, 전광판 등)  - 냉난방: 대기 공간 내 온도 센서를 통한 냉난방 장치 자동 조절  - 조명: 조도 센서를 통한 대기 공간 내 조명 조절  - 버스 도착 알림: 버저를 통해 버스 도착 시 대기자에게 알림  - 대기자 유무 알림: Firebase 서버와 LED를 통해 버스에 대기자가 있음을 알림  **3)기대효과**  **-** 불필요한 에너지 낭비를 줄일 수 있을 것으로 예상.      -  버스 이용자들에게 쾌적한 환경을 제공하기에 고객만족도가 증가할 것으로 예상.      -  해당 시스템을 활용하면 불필요한 버스 정차가 줄기에 교통혼잡이 줄 것으로 기대되며 버스의 효율성이 향상될 것으로 예상.  **4)성취수준(에어컨 및 다른 세부 내용들이 추가적으로 필요할 것 같습니다)**   * 전반적으로 해당 시스템에 필수적인 요소들을 모형 형태로 구현 성공 * 화재경보기의 경우 온도센서를 활용하여 정류장 내 온도를 측정하여 파이어베이스에 전송하여 온도 값이 특정 값을 넘길 시 싸이렌 소리구현. * 버스 대기자 호출과 알림의 경우 LED센서를 사용하여 대기자 호출이 있는 경우 전 정류장을 지난 순간 LED로 알림. 해당 정류장을 지난 후에는 자동으로 대기자 호출 초기화 * 대기자가 인식된 이후 10분간은 버스 정류장의 시스템이 작동되게 함. 중간에 대기자가 인식되면 시간이 리셋 되어서 중간에 대기자가 다시 생겨도 작동이 멈추는 일은 없게 함. 이를 통해 기존의 버스 정류장에 비해 소모되는 전력을 줄임. * 에어컨 부분: 추가수정해야함 |

1. **팀 활동 과정**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **3.1 팀 활동 내역**  **김승현:** 보고서 작성과 앱 인벤터를 통한 버스 승차 알림 요청 시스템 구현, 가상의 버스 정류장과 버스 노선을 시스템 상으로 구현하여 프로젝트와 연동, D1 보드 및 여러 센서들을 사용하여 실제 회로 구현 및 정류장 모형 제작, 결과 발표를 위한 시연 영상 촬영  **강걸:** 보고서 작성과 조도센서를 이용한 LED 자동 점등 및 밝기 조절, 적외선 센서를 활용한 자동 환풍 기능 구현.  **안호권:** 보고서 작성과 온도센서와 파이어베이스를 연동한 화재경보시스템 구현. 적외선 센서를 활용한 에어컨 ON/OFF 구현.  **임도현:** 보고서 작성과 적외선 센서를 사용하여 대기자 유무시스템 구현하여 파이어베이스와 연동. 파트 별 코드 통합.  **정다솔:** 아두이노-파이어베이스간 통신을 통한 버스정류소 버저 알림, 버스 대기자 알림 구현. 보고서 작성 및 발표자료 작성  **3.2 프로젝트 활동 중 경험한 문제점**  **김승현:** Firebase를 통해 어플과 아두이노 간의 데이터 통신 중 데이터 타입의 불일치와 데이터 조작의 시간 지연에 의한 오류, 지도 상에 효과적으로 정류장과 해당 정류장의 버스 목록을 추가하는 방법, 앱에서 휴대폰의 위치센서를 활용하기 위한 권한을 얻어오지 못하는 어려움을 겪음.  **강걸:** 보다 효율적인 전력 사용을 위해 단순 LED 점등 뿐만 아니라 주변 밝기에 맞춰 조절하는 밝기를 맞추는 것에 대한 방법을 강구하는데 어려움을 겪음.  **안호권:** 프로젝트 개발 중 냉난방 장치 자동화를 위해 필수적인 리모컨의 IR신호 획득에 어려움을 겪음.  **임도현:** 프로젝트 개발 중 대기자를 인식하고 대기자가 없더라도 마지막 대기자가 인식된 순간 이후 일정 시간 동안에도 대기자가 있다는 신호를 부여하는데 어려움을 겪음. 코드를 통합하는 과정에서 아날로그 I/O 포트가 부족하여 2개의 보드를 활용하는데 어려움 겪음.  **정다솔:** 어플리케이션과 아두이노가 파이어베이스에서 같은 변수를 사용하도록 통합하는 과정에서 어려움을 겪음. 실시간으로 버스의 위치를 파악해 LED값와 대기자 데이터 값을 업데이트하는 데에 어려움을 겪음  **3.3 문제점 해결 과정**   |  |  |  | | --- | --- | --- | | **문제점** | **적용 해결 방법** | **해결 과정 및 결과** | | 리모컨 IR 센서 | -IR센서 측정 장비를 추가적으로 구매.  - 각종 라이브러리를 참고하여 리모컨의 적외선 신호값 을 측정하는 코드 학습,  - IR을 활용한 에어컨 자동화 알고리즘 학습. |  | | D1 보드 아날로그 I/O 부족 | - 보드를 추가적으로 구매  - D1 보드와 역할 분리 | UNO 보드를 추가적으로 구매하여 사용되는 아날로그 I/O 은 UNO 보드에서, 그 외의 부분은 기존의 D1 보드에서 작동하게 함 | | 어플리케이션과 아두이노 의 파이어베이스 통합 | - 아두이노에서는 대기자 호출 정보를 bool 타입으로 사용하지만 어플리케이션에서는 int 타입을 사용  - 파이어베이스에서 int 타입의 데이터를 string타입으로 받아오도록 설정 | - 아두이노와 파이어베이스 연동에서 어떤 타입의 변수가 사용 가능한지 파악  -  사용하는 변수를 모두 string 타입으로 통합  - 이를 통해 버스 대기자에 대한 정보 확인이 용의해졌으며 스위치를 통한 대기자 호출 취소 기능 구현 성공 | | 버스 위치 정보 실시간 수집 | - 전 정류장에 도착할 경우 버스 정류장 버저 및 버스 정류장과 버스의 led 알림 설정 필요  - 해당 정류장을 지나갈 경우 버스 대기자 값 초기화  - 해당 정류장을 지나갈 경우 버저 알림 종료 및 led 소등 | - 루프 과정에서 정류장의 값이 바뀌는 경우를 추적하여 그 경우에만 led 알림과 버저 알림 울리도록 설정  - 이를 통해 버저의 울림 횟수를 한 번으로 고정하고 전 정류장과 해당 정류장을 구분하여 led를 점등/소등하도록 구현 | | 센서 파트 통합에 따른 변수 통합 및 포트 분배 | - 동일한 변수를 받아서 작동해야 되는 부분과 아날로그 부분은 통합  - 화재 알림 시스템과 버스 알림 시스템은 버저 포트를 공유해야 해서 D1 보드와 UNO 보드의 I/O을 조절 | - D1 보드와 UNO 보드에서 버저는 UNO에 연결되어 있어 D1에서는 출력으로, UNO 보드에서는 입력 값으로 처리하며 D8 포트를 서로 연결하여 연동 | | 주변 조도에 따른 LED 밝기 변화 | -  조도센서 입력 값을 보다 세분화하여 LED의 밝기 조절 | - 적정 조도 센서 입력 값 구분  -LED 전압 변화를 통한 밝기 조절 | | 앱과 아두이노의 Firebase 데이터 조작에 대한 시간 지연에 의한 오류 | 앱과 아두이노가 동시에 수행하는 일(버스 도착 시 알림 요청 초기화)을 하나의 Device에서만 수행 | 앱과 정류장이 Firebase의 데이터를 초기화 시키는 대신, 버스가 일괄적으로 모든 요청을 초기화 하도록 하여 시간 지연의 문제 해결. | | 앱의 지도 상에 정류장과 그에 대한 데이터를 효과적으로 입력하는 방법 | 일일이 앱 인벤터의 Marker를 통해 정류장을 설정하는 대신, 모든 정류장의 데이터를 하나의 파일로 만들어 업로드 | 모든 정류장에 대한 데이터(이름, 위치, 버스 목록)를 “geojson” 파일로 만들어, 추후에 데이터의 변경 및 다른 정류장의 추가를 용이하게 함. | | 앱 인벤터에서 안드로이드의 위치 센서 사용 권한을 얻지 못함 | 앱 인벤터의 함수 외에 다른 방법을 사용하여 위치 센서 권한 획득 및 위치 정보 입력 | 앱 인벤터의 Web Viewer를 통해 Google Map을 실행, 해장 지도에서 “내 위치” 버튼을 통해 위치 센서 권한 및 사용자의 GPS 정보를 획득하면, 해당 권한과 데이터를 앱에서 활용. | |

1. **시스템 설계 및 상세설계**

|  |
| --- |
| **4.1 프로젝트 전체 구성도**    <그림 2>Smart Bus Station 전체 구성도  우리 조가 구상한 Smart Bus Station 프로젝트는 크게 고객, 버스정류장, 버스로 이루어진 세 가지 주체들의 상호작용으로 구현된다. 이에 해당 주체들은 IOT를 기반으로 위의 그림과 같이 정보를 주고받는다. 먼저 고객의 경우 자신들이 정차할 버스에 대한 정보를 앱과 버스 정류장의 대기 버튼으로 파이어베이스에 전송한다. 버스의 경우 파이어베이스에서 오는 정보를 바탕으로 대기자의 존재유무와 정차유무를 판단한다. 그리고 버스정류장의 경우 버스정류소에 설치된 각종 장비들이 획득한 정보 값을 파이어베이스에 전송하고, 코드에 입력된 값을 바탕으로 고객에게 쾌적한 환경을 제공한다. 우리 조는 해당 프로젝트를 모형으로 표현하였는데, 모형으로 나타낸 우리 조의 전체 구성도는 다음과 같다.  **4.2 기능별 구성도 및 흐름도**   Smart Bus Station의 구체적인 기능별 구성도 및 흐름도는 크게 앱과 아두이노에 설치된 각종 센서들로 나눌 수 있다. 먼저 앱부분을 살펴보면 다음과 같다.    <그림 3>앱 구동화면 1 <그림 4>앱 구동화면 2    <그림 5>앱 구동화면 3  앱 구동화면은 위의 <그림 3>, <그림 4>, <그림 5> 와 같고 앱의 작동 흐름도를 살펴보면 아래 <그림 6>으로 나타낼 수 있다.    <그림 6>버스 부분 Flow Chart    <그림 7>버스 호출 Flow Chart    <그림 8> 대기자 호출 Flow Chart  버스 정류장의 구성도를 살펴보면 디지털 신호를 I/O 받는 D1 보드와, 아날로그 신호를 I/O 받는 UNO 보드로 구성되어 있다. 적외선 센서, 온도 센서, 조도 센서는 UNO 보드에 연결되어 있으며 나머지 센서들은 D1 보드에 연결되어 있고 각각의 D8 포트로 IR\_State를 공유하며 연결된다.    <그림 9>D1 보드 <그림 10>UNO Nano 보드  각 센서 별 기능을 살펴보면 다음과 같다.   1. **온도센서**: 먼저 아두이노 우노 보드의 J8 포트에 설치된 온도센서를 활용하여 주변 온도에 대한 정보를 측정하여 파이어베이스에 전송한다. 그리고 해당 온도 값을 바탕으로 에어컨의 작동 및 화재발생을 감지한다. 2. **버저**: 아두이노 UNO 보드의 J6 포트에 연결된 버저를 활용하여 화재발생 시 싸이렌 소리가 울리도록 하고 버스 도착 알림을 D1 보드로부터 D8 포트로 전송을 받으면 알림이 울리도록 하였다. 3. **적외선 센서**: 아두이노 UNO 보드 J7에 연결된 적외선 센서를 바탕으로 센서에 대기자가 감지되면 감지된 시간으로부터 20분 동안은 대기자가 있다고 판별하게끔 하였고 bool 형태로 유무를 출력하였다. 4. **LED:** 대기자가 있을 경우 버스에 LED를 통해 알림이 가도록 구현하였다. 5. **조도 센서:** 조도 센서를 통해 변화하는 조도 값을 측정하고 이를 바탕으로 어두울 시에만 LED를 작동시켜 밝기를 조절하였다. 6. **IR 센서**: 에어컨 리모컨 신호를 해석 및 전송하여 에어컨을 제어하기 위한 센서 값을 추출하였다.   각 센서의 Flow Chart는 아래와 같다.    <그림 11> 온도, 버저, 적외선 센서부 Flow Chart    <그림 12> IR 센서 및 환풍기 Flow Chart    <그림 13> 조도 센서 Flow Chart |

1. **프로젝트 결과 평가**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **◦ 목표 기술 달성도**   |  |  |  | | --- | --- | --- | | **목표 기술** | **구현 기술** | **기술 달성도(%)** | | **화재 알림** | 온도 측정 및 싸이렌 소리 구현 | 90%. 전체적인 구동은 성공하였으나, 버저의 한계로 싸이렌 소리를 온전히 구현하진 못함. | | **에어컨 자동화 기술** | 적외선 센서와 연동하여 대기자 유무를 파악하였다. | 80%. 리모컨 센서를 해킹하여 ON/OFF 구동은 성공하였으나 온도와 습도 등 세부적인 조절은 구현하지 못함. | | **LED 자동 조절 기능** | 조도 센서를 통하여 주변 밝기를 측정하고 LED 단순 점등뿐만 아니라 밝기를 조절하도록 구현 | 100%. 조도 센서를 통해 밝기 값의 수치(버스 정류장 외부의 밝기)를 바탕으로 LED 2개를 조절하여 밝기 조절 | | **자동 환풍 기능** | 적외선 센서와 연동하여 대기자가 유무에 따라 자동으로 환풍을 하도록 구현 | 100%. 적외선 센서로부터 대기자 유무를 판단하고 이를 통해 대기자가 있거나 일정 시간 동안은 작동되게 구현 | | **대기자 알림 기능** | 버스 정류장의 스위치와 어플리케이션을 이용하여 버스에 대기자가 있음을 알리도록 구현 | 100%. 파이어베이스와 앱과정류장과의 연동을 통해 버튼을 눌러서 대기자가 신호를 보낼 수 있게 구현함. | | **버스 알림 기능** | 버스의 위치정보를 실시간으로 확인하여, 대기자가 있는 버스가 정류장에 도착할 경우 버저를 통해 알리도록 구현 | 100%. 파이어베이스와 앱과 정류장과의 연동을 통해 대기자가 있을 경우, 버스가 접근하면 정류장 내의 UNO 보드에 연결된 버저를 통해 알리도록 함 | | **사용자 위치에 기반한 승차 알림 요청 제한** | 핸드폰의 위치 센서를 통해 사용자의 GPS 정보를 불러오고, 사용자의 위치와 가까운 정류장에 대한 승차 알림 요청 기능만을 활성화  추가로 이미 승차 알림 요청을 전송한 정류장에서 일정 거리 이상 멀어질 경우, 메세지와 함께 해당 승차 알림 요청 자동 취소 | 100% 구현 | | **다수의 사용자의 앱의 동시 사용** | 특정 정류장의 특정 버스에 대한 승차 요청을 bool 형태가 아닌 int 형태로 하여, 앱 및 정류장의 스위치를 통한 승차 알림 요청 시 해당 int 데이터에 +1을 하여 업데이트. 이후 해당 데이터가 0이 아닌 경우에만 버스에 알림 활성화. | 100% 구현 | | **한 명이 하나의 승차 알림 만을 요청할 수 있도록 제한** | 앱을 통해 이미 승차 알림 요청을 한 상태에서 다른 버스에 승차 알림을 요청할 경우, Firebase에 보낸 이전 승차 요청에 해당하는 데이터를 -1, 새로 신청한 승차 요청에 해당하는 데이터를 +1 하여, 중복 신청을 제한. | 100% 구현 | | **하나의 버스에 대한 서로 다른 정류장에서의 승차 알림 요청 기능** | 버스의 위치에 따라 다음에 도착할 정류장에 대한 승차 알림 만을 확인하여 알림 동작. 버스가 정류장에 도착하면 해당 정류장에 대한 승차 알림 요청 초기화. 버스가 정류장을 출발하면 다시 다음 정류장에 대한 승차 알림 확인. | 100% 구현 | |

**[첨부] 기술자료**