|  |
| --- |
| **보안을 강화한 스마트 홈 서비스**  컴퓨터공학과 2011104034 이근철  컴퓨터공학과 2011104027 안재성  **요 약**  현재 임베디드 기기를 현대 집에서 손쉽고 다양하게 사용하게 되었다. 이러한 상황에서 이러한 기기들이 보내는 정보들의 보안이 제대로 이루어지는 지에는 의문이 남는다. 그렇기 때문에, 스마트 홈 서비스에서 전달되는 정보들이 확실한 보안을 유지할 수 있으면 집 환경이 안전할 것이라 생각했다. 따라서 ECC칩을 통해가전기기의 정보 송/수신에 암호화/복호화 과정을 거치며, 기기 자체의 secure함도 확인할 것이다 |

**1. 서론**

**1.1. 연구배경**

현재는 pc를 넘어서 모든 분야에 컴퓨터가 사용되고 있다. 현재 일상 생활 속에 컴퓨터는 곳곳에 존재하며, 사물인터넷이란 것으로 IOT가 크게 대두되고 있다. IOT에서 우리가 핵심적으로 본 것은 정보의 교류가 시시각각 이루어지고 있다는 것이다. 최근 사소하게는 집안 난방부터, 각종 관공서, 접근 제한 처에서도 IOT기술들이 들어오기 시작했다. 따라서 이 정보의 교류에 보안은 핵심이며, 아직 크게 문제삼지 않아온 IOT 관련보안을 해결해야 한다.

또한 추가적으로 Arduino 관련 보안 Issue 는 크게 두드러지지 않는다. Arduino 를 가전제품이라고 생각하여 Arduino 상에서 발생할 수 있는 보안 Issue 들을 다루어 보고자 하였다.

**1.2. 연구목표**

Smart Home Service 가 현실로 다가 오고 있다. 하지만 항상 그렇듯, 점점 편리해 질수록 개인 정보는 점점 더 노출되기 쉬워지는 법이다. 이러한 정보들을 다루는데 있어서 좀더 보안 관련 Issue 가 적용된다면, 조금이나마 개인정보 노출을 막을 수 있지 않을까 하고 생각했다. Smart Home Service 가 대부분 Embeded 기기에서 진행하는 만큼, Arduino, Raspberry PI 등의 하드웨어를 이용하여 구현하고 암호화 알고리즘을 적용해 봄으로써 Open Source Hardware 상의 보안 Issue를 처리할 수 있도록 할 것이다.

**2. 관련연구**

**2.1. Smart Home 가전제품**

**2.1.1. Smart Home 가전제품 Test 제품 (Arduino)**

Smart Home System 에는 여러 개의 가전제품들이 존재한다. 각각의 가전제품들은 각자의 용도를 가지고 있으며, 동작 방식도 다르다. 이러한 요구사항을 만족시키기 위해 System을 표현할 수 있는 기기를 선정해야 했다. 여러 가지 기기를 살펴보던 도중, Open Source Hardware이며 여러 가지 Hardware( Led, Motor 등등)를 제어할 수 있는 Arduino 보드를 선택하게 되었다.

**2.1.2. Arduino 와 Hub 와의 통신 (Wi-Fi)**

Arduino 보드는 기본적으로LAN 을 제공하지 않는다. 하지만 Smart Home Service 를 재현해 내기 위해서는, Hub 장치와 통신은 필수적이었다. 이를 해결하기 위한 추가 장치로 Wi-Fi Shield 라는 제품을 Arduino에 장착시켜 Wi-Fi 를 통해 TCP/IP 통신을 가능하게 하였다.

**2.2. Smart Home Hub (Raspberry PI)**

Smart Home Service 를 제어하기 위한 Smart Hub 가 필요하다고 생각하였다. 여러 개의 Smart Home 가전제품들과 통신, 제어 등을 맡아야 하므로 소형 컴퓨터가 필요하다고 느꼈다. 그렇기에 Linux 기반 Hardware 인 Raspberry PI 를 이용하여 Smart Home Hub 를 구현해 보고자 하였다.

**2.3. Smart Home 제어 단말기 ( Android )**

Smart Home Service 를 실질적으로 제어하기 위해서는 단말기가 필요하다. 즉, 집 외부에서 가전제품의 상태, 가전제품 조절 등을 할 수 있어야 했다. 가장 중요한 것은 누구든지 가지고 있는 제품이어야 쉽게 Service를 사용할 수 있다는 점이었다. 그렇기에 Smart Phone 을 단말기로 지정하였고, 점유율이 더 높은 Android OS 기반 Smart Phone을 선정하였다.

**2.4. 암호화 알고리즘**

보안을 위해서 필요한 암호화 알고리즘이다. 암호화 알고리즘에 있어서 선정해야 될 방식은 크게 두 가지 이다. ( 1) 암호화 알고리즘 2) Key 교환 알고리즘). 암호화 하는데 있어서 보안성이 너무 낮으면 쉽게 해킹 당할 것이며, 반대로 너무 높다면 선정한 Hardware 들의 성능이 저하될 수 있다. 또한 Key 교환 알고리즘 역시 마찬가지 이다.

**2.5. 기존 연구의 문제점 및 해결 방안**

**2.5.1. 연구의 문제점**

이미 다양한 임베디드기기에서 나름대로의 암호화는 거치고 있으나, 고유 SSH와 HTTPS 사설 키들을 이용하고 있었다. 이들 상당수는 동일 벤더 혹은 다른 벤더와 키를 공유하고 있었다. SEC Consult의 연구원들이 70여 제조사의 4000여 임베디드 기기들의 펌웨어를 분석한 결과, 230개 이상의 키들이 400만개가 넘는 인터넷-연결 기기들에 활발히 사용되고 있었고, 150개 정도의 HTTPS 서버인증은 320만 대 기기에 의해 사용되고 있었고, 80개의 SSH 호스트 키는 90만대 기기에서 사용되고 있었다.나머지 키들 역시 인터넷에서 접속이 불가능한 다른 많은 기기들에서 사용될 수 있는데, 이들 기기의 경우로컬 네트워크를 통해 이뤄지는 중간자 공격에 취약할 수 있다. SSH 호스트 키들은 SSH 서버를 구동하는 기기들의 신원을 식별하는데 사용된다. 사용자가 암호화된 SSH 프로토콜을 통해 처음 그런 기기에 접속할때 공공-사설 키 쌍의 일부인 그 기기의 Public 키를 저장하게 된다. 다음 번 접속 시에 서버의 신원은 자동적으로 사용자의 SSH 클라이언트상에 저장된 공공 키와 기기상의 사설 키에 기Ω반해 인증된다. 만약 공격자가 기기의 SSH 호스트 사설 키를 훔치고 사용자의 접속 시도를 탈취할 수 있는 위치에 있다면, 그는 그 기기를 가장하고 사용자의 컴퓨터가 자신의 컴퓨터와 대신 소통하게 만들 수 있다. 공격자가 사용자와 자체 웹-기반 관리 인터페이스 사이의 커뮤니케이션을 암호화하는데 사용되는 기기의 HTTPS 사설 인증 접속권을 얻어도 비슷한 공격이 가능하다.

**2.5.2. 해결 방안**

**2.5.2.1 기기 인증 방식 (Authentication)**

가전제품과 허브 사이의 통신에 있어서, 정말로 적법한 가전제품이 나에게 통신을 요청하였나, data를 보냈는가를 알아내기 위해서는 Authentication 이 필요하다. 이를 이용하기 위해 Authentication 알고리즘 중 하나인 ECDSA 를 사용 하였고, 이를 하드웨어적으로 제어하기 위해서 ATECC508A-SSHCZ-TCT-ND Chip 을 이용하여 Arduino(가전제품) 기기를 인증하려고 한다.

**2.5.2.2 Arduino – Raspberry PI 통신간의 보안 ( micro-ECC library, ECDH )**

가전제품과 통신간 보안성을 증대 시키기 위해서는 암호화 알고리즘이 적용 되어야 한다. 암호화 알고리즘에서 가장 중요한 것이 바로 Secret Key 관리 인데, Secret Key 의 노출은 절대로 일어나서는 안 되는 보안 과제이다. 이를 해결하기 위해 우리는 Secret Key 를 통신이 연결되는 순간 생성하는 방법을 생각해냈고, 키 교환 방식으로는 Diffie–Hellman 키 교환방식을 사용하기로 하였다. Diffie–Hellman 키 교환방식 이란, 통신 양 측에서 Private, Public Key를 생성하고 Public Key 교환 후 Session Key를 생성하는 방식이다. 우리는 이러한 알고리즘을 사용하기 위해 ECDH(Elliptic Curve Diffie-Hellman) 알고리즘을 선택하였다.

Arduino 에서는 간소화된 micro-ECC library 를 사용하였고 Raspberry PI 에서는 OpenSSL 을 이용하여 암호화 복호화 하였다.

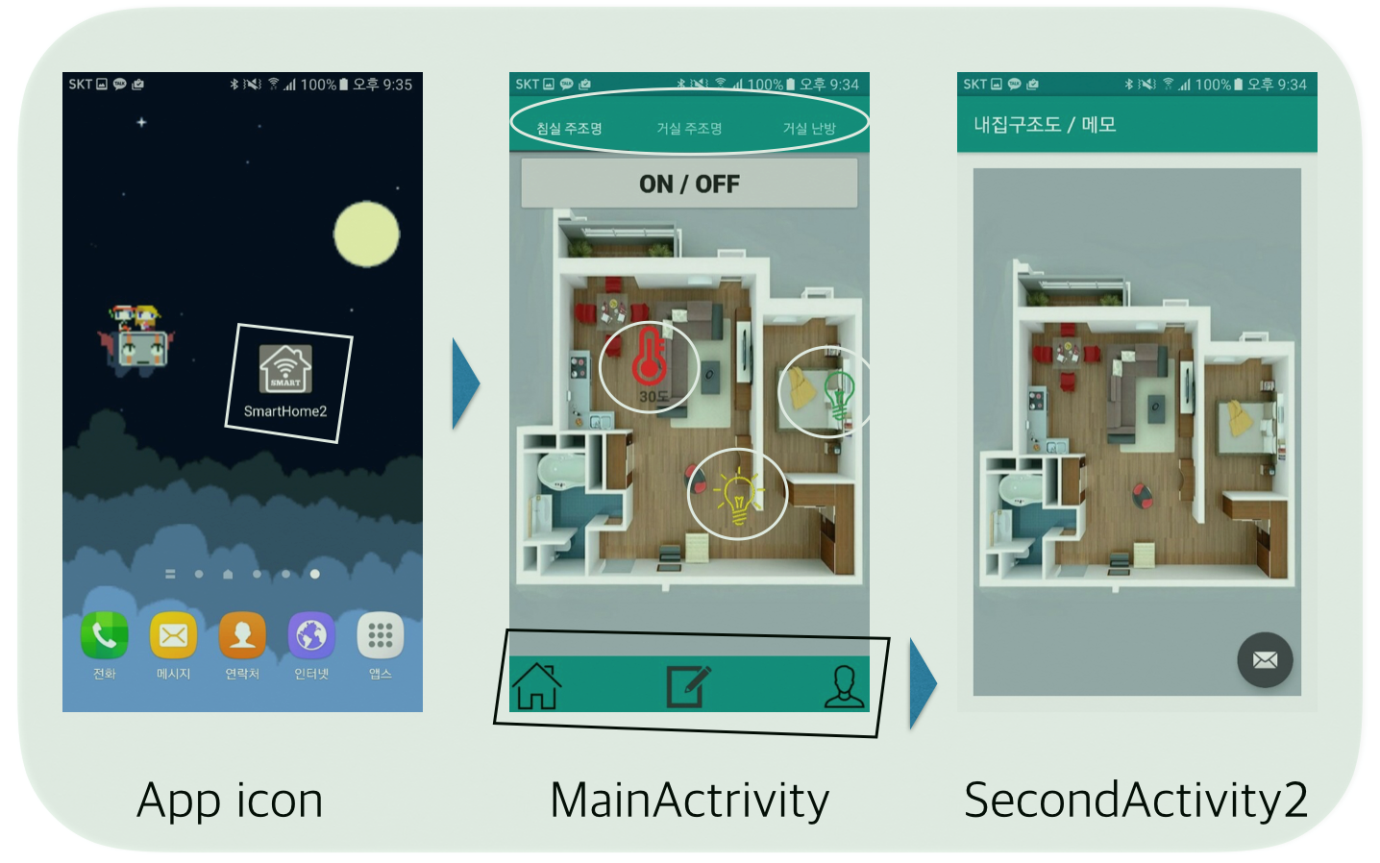
**2.5.2.3 암호화 알고리즘 AES**

실제 통신간 적용된 암호화 알고리즘이다. ECDH 로 교환한 Key를 이용하여 암호화, 복호화를 진행하는데 서로의 동일한 Key 값으로 암호화, 복호화를 진행하는 대칭형 알고리즘을 선택하였다. 그 중 현재 미국에서 표준 암호화 알고리즘으로 채택하고 있는 AES(Advanced Encryption Standard Algorithm) 를 적용하였다.

Arduino 에서는 간소화된 AES Library 를 사용하였고, Raspberry PI 에서는 OpenSSL 을 이용하여 암호화 복호화 하였다.

**3. 프로젝트 내용**

**3.1. 시나리오**

****

**[그림1, 2, 3] 어플리케이션 구상도**

[그림 1] 어플리케이션의 시작전 아이콘이다. 1번에 위치한 아이콘 클릭과 동시에 [그림 2]의 앱 기본화면이 실행되는데, 이 화면이 뜨기 위해선 이미 작동중인 중앙허브(raspberry pi)와 TCP접속을 해야한다. 접속이 실패하면 화면에 0이라는 연결불가알림 메시지가 잠시 뜬다.

[그림 2]에서 이제 우리는 3번에 위치한 아이콘들로 현재 가전기기의 상태를 알 수 있다. 3번의 상태값들은 가전기기(arduino)에서 측정한 값들을 중앙허브를 통해 가져온 것이다. 이 가전기기들을 조작하기 위해서 4번에 위치한 버튼들을 직접 눌러서 조작할 대상을 변경 할 수 있다. 또는, 5번에 위치한 화면에서 가볍게 손을 끄는 행위(Tap)를 하여 조작대상을 변경할 수 있다. 조작할 대상을 선택하였으면, 이제 6번에 위치한 버튼을 통해 조명을 키거나/끄고, 온도를 조절할 수 있다. 7번을 누르면 내집구조도/ 메모 페이지인 [그림 3] 의SecondActivity2를 킨다. 여기서 내 집의 전체 구조도를 확인 할 수있고, 9번을 통해 가볍게 메모를 남길 수도 있다.

**3.2. 요구사항**

**3.2.1. 가전기기(arduino)에 대한 요구사항**

**-** 메모리가 적은 임베디드 시스템에서 메모리 부족현상이 나지 않게 구현한다.

**-** 실제 가전기기 사용 예를 보이기위한 조명기능과 난방기능을 구현한다..

**-** 암/복호화가 빠르게 이루어져야 한다.

**-** 중앙허브와의 데이터 통신암호화를 위해 ECDH알고리즘으로 Session키를 생성해야한다.

**-** 데이터 암/복호화에 AES알고리즘을 적용한다.

**-** 데이터 Signatrue를 위해 ECC알고리즘을 적용한다

**-** 수신받은 데이터는 verify를 거쳐 확인한다.

**-** 중앙허브에 WiFi로 접근을 하며, 가전기기는 client역할을 수행한다. 연결이 빠르게 이루어져야한다.

**3.2.2. 중앙허브(raspberry PI)에 대한 요구사항**

**-** 가전기기와 앱간의 중앙다리 역할을 수행한다.

**-** 가전기기와의 데이터 통신암호화를 위해 ECDH알고리즘으로 Session키를 생성해야한다.

**-** 데이터 암/복호화에 AES알고리즘을 적용한다.

**-** 데이터 Signatrue를 위해 ECC알고리즘을 적용한다.

**-** 수신받은 데이터는 verify를 통해 확인한다.

**-** WiFi로 접근해오는 가전기기에게서 중앙허브는 server역할을 해야한다. 포트는 2345를 사용.연결이 빠르게 이루어져야한다.

**-** 앱에서 가전기기의 데이터를 요구하면, 암호화된 가전기기의 상태값을 가져와 복호화하고, 다시 재 암호화를 해서 앱으로 보낸다.

**-** 앱에서 가전기기의 상태수정을 요구하면, 해당 암호화된 요구를 중앙허브에서 복호화하고, 해당 기능수행을 위한 데이터를 가전기기에 보낸다.

**3.2.3. 앱(android app)에 대한 요구사항**

**-** 직관적으로 사용이 가능하게 구현한다.

**-** 알리지 않아도 되는 부분은 과감히 알리지 않는다.

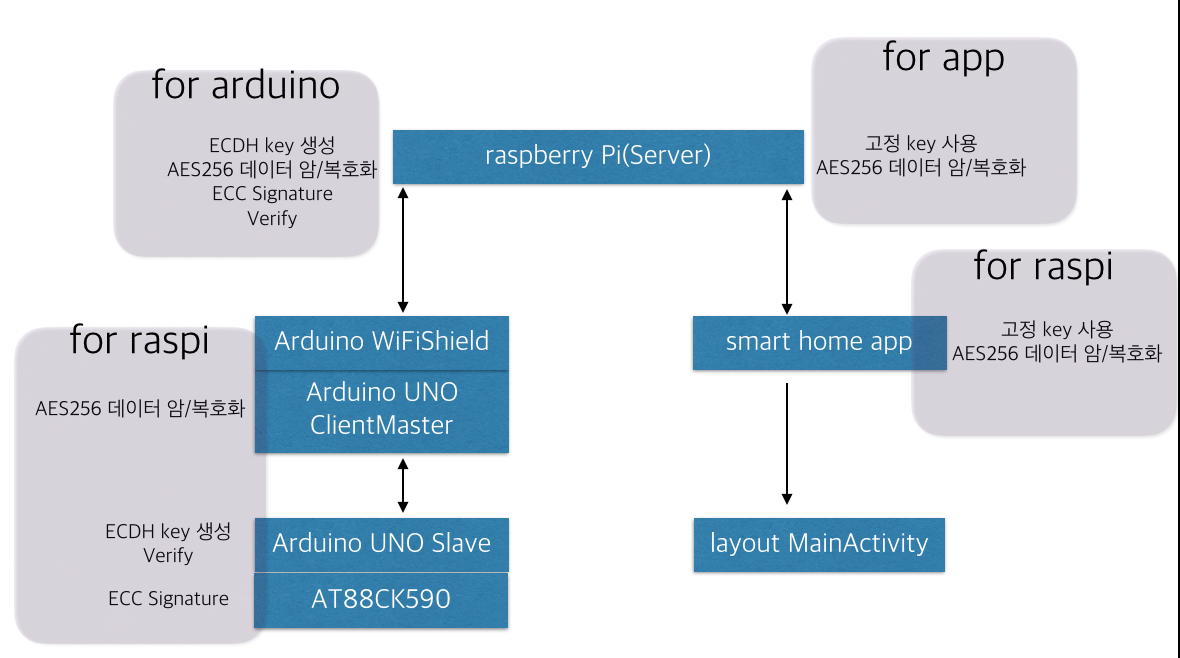
**-** 알아보기 쉬운 화면구성.

**-** 서버와의 연결을 최대한 빠르게 수행하며, 유지하게 구현한다.

**-** Activity구성 최소화.

**3.3. 시스템 설계**

**3.3.1. 시스템 구성도**

****

**[그림] 시스템 구성도**

이 System 은 총 4개의 Hardware 가 존재한다.

Smart Phone : 사용자가 Smart Home Service를 제어하기 위해 사용하는 단말기

Raspberry PI : Smart Home Service를 총괄하는 Hub

Arduino : 집에 존재하는 Smart 전자제품

( 분산처리 Arduino, 실제 1개의 기기와 같은 역할을 한다.)

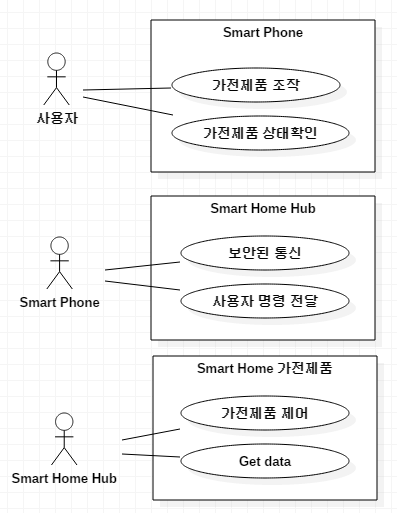
ECC 처리 Arduino : 암호화 관련을 처리하는 Arduino

Wi-Fi 처리 Arduino : Wi-Fi 통신을 처리하는 Arduino

.

**3.3.2. UML Diagram을 통한 시스템 모델링**

**3.3.2.1. Use Case Diagram**



**[그림] Use Case Diagram**

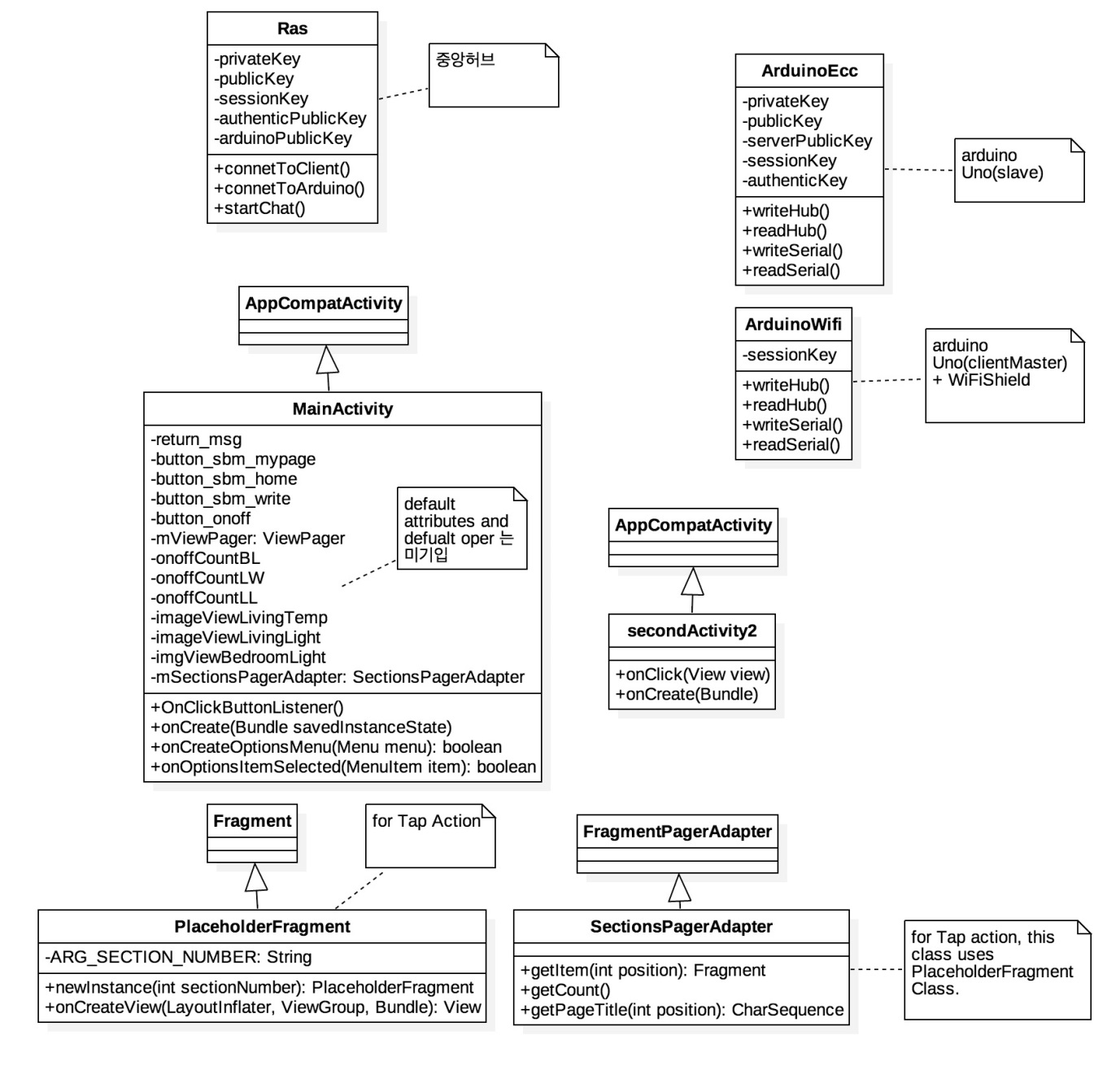
Use Case 는 3명의 사용자가 있다. 이는 통신 단계가 사용자, Smart Phone, Hub, 전자제품 순으로 3단계를 거쳐 이동하기 때문이다.

사용자는 가전제품을 조작할 수 있고, 상태를 확인할 수 있다. 이를 통해 현재 사용자의 집의 가전제품이 어떻게 동작하고 있는지를 알아볼 수 있다.

Smart Phone 은 Hub 를 제어한다. 보안된 통신을 이용하여 사용자의 명령을 Hub 로 전달한다.

Hub 는 전자제품(Arduino) 들을 제어한다. 전자제품을 원하는 상태로 변경할 수 있으며 현재 전자제품의 상태를 읽어올 수도 있다.

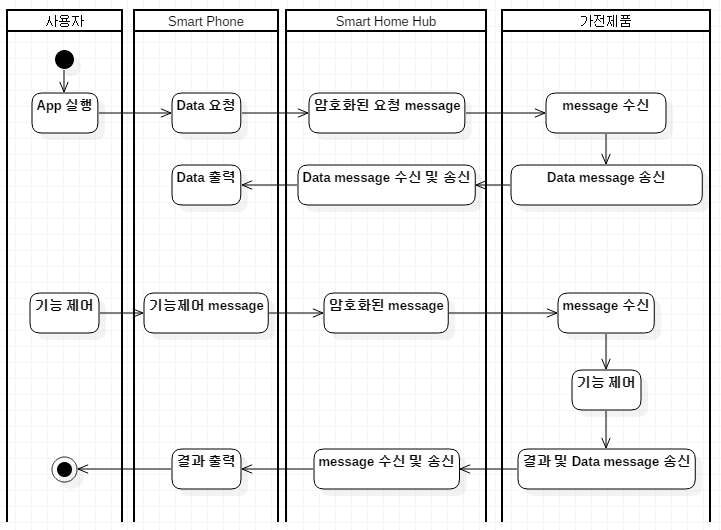
**3.3.2.2. Class Diagram**



**[그림] Class Diagram**

클래스는 크게 Bluetooth, Chords, Chords저장부분, Stroke, Chords Protocol부분으로 이루어져 있다. Chords를 잡는 부분은 사용자가 잡은 코드를 블루투스를 통해서 Stroke에 보내고 Stroke는 Chords부분으로부터 받은 코드로 Sound를 재생한다. ChordsSave부분은 연주되는 Chords부를 저장한다.

**3.3.2.3. Activity Diagram**



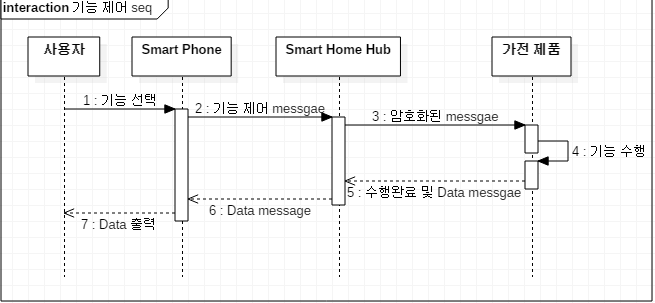
**[그림] Activity Diagram**

사용자가 APP을 실행한 후 크게 두 가지 요청을 할 수 있다. 1) Data 요청 2) 기능실행

Data 요청 시, Smart Phone 은 Hub 로 Data 요청 message를 보낸다. 이를 받은 Hub 는 암호화된 Data 요청 Message 를 가전제품에게 보내고, 요청을 받은 가전제품은 현재 Data 를 반환하여 Message 를 Hub로 보내준다. 이를 받은 Hub 는 Smart Phone 으로 다시 message를 보내게 되고 사용자는 직관적 Interface를 통해 확인할 수 있다.

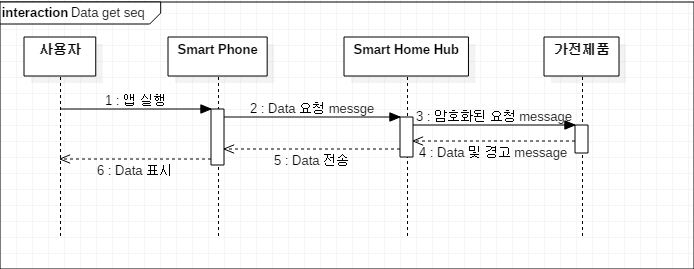
기능 제어 시, Smart Phone 은 Hub 로 기능제어 message 를 보낸다. 이를 받은 Hub 는 암호화된 기능제어 Message 를 가전제품에게 보내고, 요청을 받은 가전제품은 선택된 기능을 수행하고 성공 여부를 결과로서 Hub에게 보낸다. 이를 받은 Hub 는 Smart Phone 으로 다시 message 를 보내게 되고 사용자는 직관적 Interface를 통해 확인할 수 있다.

**3.3.2.4. Sequence Diagram**



**[그림] 기능 제어 Sequence Diagram**

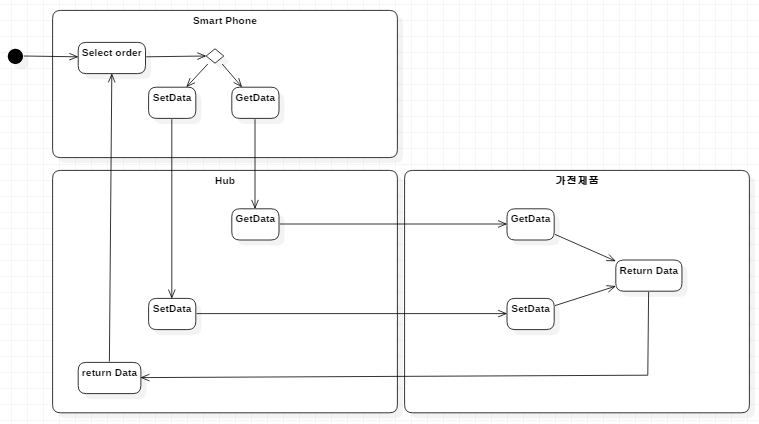
사용자가 기능을 선택하고 제어를 시작하면, Smart Phone 은 Hub 에게 기능 제어 message를 보낸다. 이를 받은 Hub 는 가전제품에게 암호화된 message를 보내어 가전제품이 기능을 수행하게 만들며, 기능수행을 완료한 가전제품은 Hub 에게 수행완료 Message를 보낸다. 이를 수신한 Hub는 다시 Smart Phone에게 message를 전송하고 사용자는 결과를 직관적인 Interface로 볼 수 있다.



**[그림]** **Data get Sequence Diagram**

사용자가 Data를 요청하거나 App을 실행하면, Smart Phone 은 Hub 에게 Data 요청 message를 보낸다. 이를 받은 Hub 는 가전제품에게 암호화된 message를 보내어 가전제품이 현재 상태를 전송하게 만들며, 가전제품은 Hub 에게 Data Message를 보낸다. 이를 수신한 Hub는 다시 Smart Phone에게 message를 전송하고 사용자는 결과를 직관적인 Interface로 볼 수 있다.

**3.3.2.5. State Diagram**



**[그림] System 의 State Diagram**

Smart Phone 에서 사용자의 명령을 판별한다. 이후 Set Data, Get Data 로 나뉘어 Hub 에 각 기능에 맞는 함수를 호출한다. Hub 는 연속적으로 가전제품에게 기능에 맞는 함수를 호출하고, 가전제품은 기능의 수행 결과를 Hub 에게 반환해 준다. 실행 결과를 받은 Hub 는 Smart Phone 으로 결과를 전송하고 사용자는 Interface를 통해 결과를 볼 수 있다.

**3.3.2.6. Event Table**

각 입력에 따라 발생할 수 있는 모든 이벤트를 표기하였으며, 각 기기( 가전제품(Arduino), Smart Home Hub(Raspberry PI), 단말기(Android) ) 따라 발생할 수 있는 상황을 모두 정의하였다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Smart Home Service Event Table** | | | | | |
| **Event** | **Trigger** | **Source** | **Use Case** | **Response** | **Dest** |
| **사용자가 제품 상태 변경** | **제품 Touch** | **사용자** | **On/Off message 전송** | **성공여부** | **Hub** |
| **사용자가 제품 현 상태 요청** | **제품 Touch** | **사용자** | **Data 요청 message 전송** | **현재 제품의 Data** | **Hub** |
| **Hub 가 기기에게 상태 변경** | **제품 Touch** | **Hub** | **On/Off message 를 선택 기기에 전송** | **성공여부** | **전자제품** |
| **Hub 가 기기에게 현 상태 요청** | **제품 Touch** | **Hub** | **Data 요청 message 를 선택 기기에 전송** | **현재 제품의 Data** | **전자제품** |
| **전자제품이 요청메세지를 받음** | **제품 Touch** | **제품** | **명령어 실행** | **결과값** | **Hub** |

**3.4. 구현**

소스코드 경로 및 폴더 구성은 아래와 같다.

Smart Phone Program : [15조]Source Code\Android Client

Hub : [15조]Source Code \Hub

ECC 처리 Arduino : [15조]Source Code \SmartHome\_ecc

Wi-Fi 처리 Arduino : [15조]Source Code \SmartHome\_wifi

**4. 프로젝트 결과**

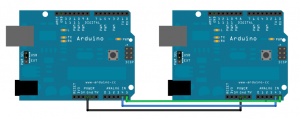
**4.1. 연구 결과**

이번 프로젝트에서 가장 중요한 점은 보안 알고리즘을 적용하여 통신이 잘 진행되는 가이다. Arduino 기기와 Raspberry PI 기기의 통신간에 있어서, 적정 수준의 보안성이 유지 되는가, 통신 속도는 양호한가, Data 의 암호화 복호화가 잘 진행되는가 의 평가 항목이 존재한다. 이를 통해 실제 Smart Home Service 에 적용될 수 있는 통신방식을 구현해 보는 것이다.

이 과정에 있어서 여러 가지 문제가 발생하였는데, 다음과 같은 방식으로 해결하였다.

**4.1.1. Arduino 제품의 메모리 부족 해결방식**

Arduino 보드는 사용가능한 memory 가 2Mb 뿐이다. Wi-Fi Shield 제어, 암호화 알고리즘 등의 각각의 기능들은 memory를 500Kb 이상 사용하였고, 이는 Arduino 전체적인 memory 부 족 현상을 일으켜 동작이 멈추는 상태가 발생하였다. 이를 해결하기 위해서 Arduino 보드 두 개를 Serial 통신으로 연결하여 한쪽의 Arduino 에서는 Wi-Fi Shield 제어 및 AES 암호화, 복호화를 진행하였고, 다른 한쪽의 Arduino 에서는 ECDH, ATECC508A-SSHCZ-TCT-ND Chip 제어를 진행하였다. 서로의 기능들을 진행 한 후, Wi-Fi Shield 쪽의 Arduino 로 data를 serial 통신 하여 보내고, 완성된 Data 를 Hub (Raspberry PI) 로 전송하였다.



**[그림] 두개의 Arduino 보드를 Serial 통신하기 위해 연결한 회로도**

**4.1.2. Data 교환을 위한 Arduino 와 Raspberry PI 사이의 통신 Protocol**

기본적으로 Arduino는 data를 읽고 쓸 때, char 형 변수로만 읽고 쓸 수 있다. Char 형 변수로 Arduino 를 제어하기 위하여 우리는 다음과 같은 간단한 Protocol을 만들어서 사용하였다.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** |  |  |  |  |  |  |  |  |  |  |  |  |  |

**Char arr[16]**

**A : Slave ID, 여러 개의 가전제품이 존재한다고 가정할 때, 각각의 가전제품에 접근하기 위한 ID**

**( 0~255)**

**B : 0 – Get Data Mode**

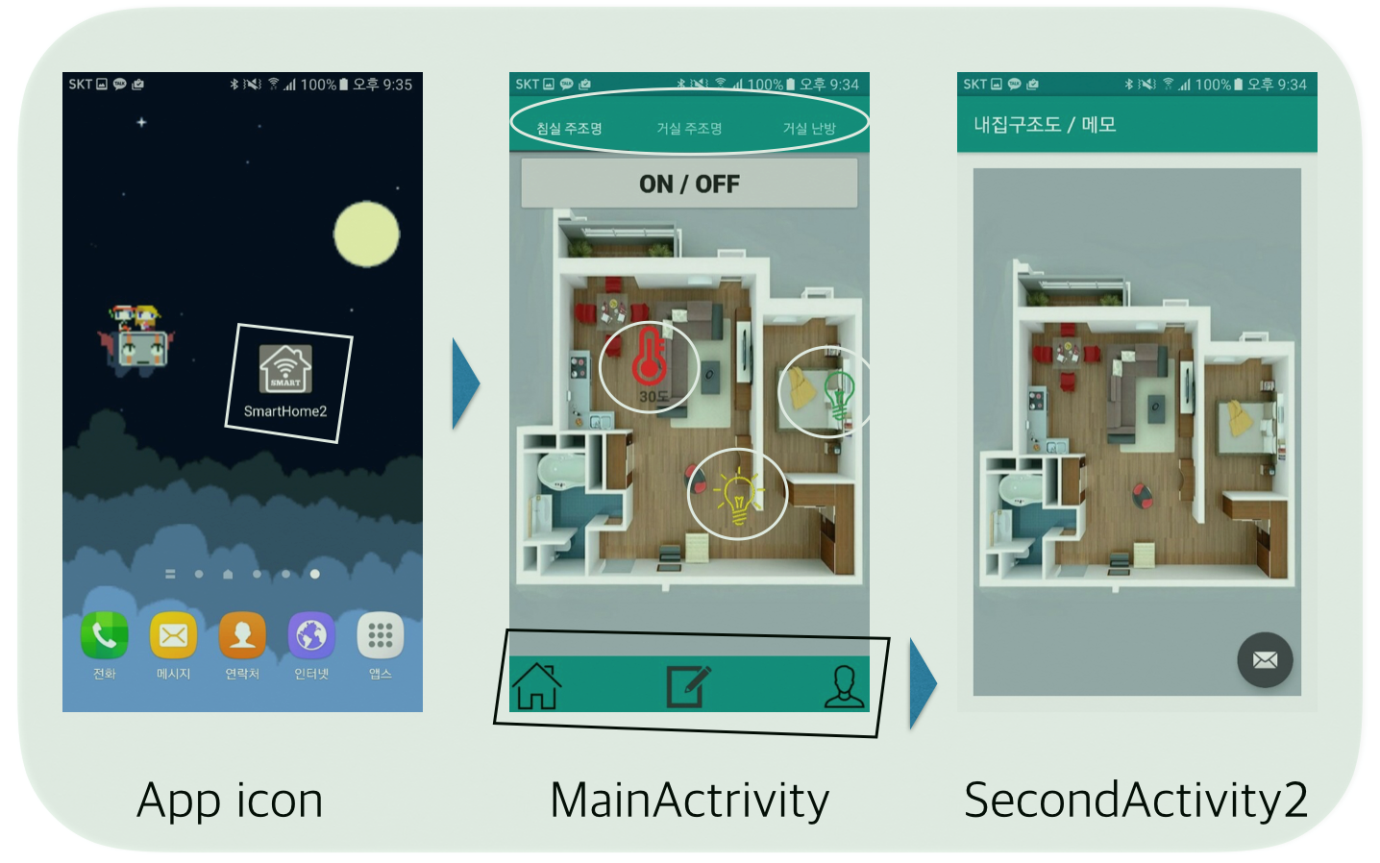
**1 – Set Data Mode**

**C : (Set Data Mode 일 시) Data 값**

현재 실질적으로 사용하는 byte 는 3개 이지만, 프로젝트가 발전함에 따라서 여러 개의 명령어가 추가 될 수 있다는 점을 고려하여 16 Byte로 Protocol 을 만들었다.

**4.1.3. 실제 집 모양과 비슷한 인터페이스 제공**

실제 집의 모습을 그대로 Smart Phone 에 표현하여 제공한다면, 제어하고자 하는 가전제품에 좀더 쉽게 접근할 수 있을 것이다.

****

**[그림1, 2, 3] 어플리케이션 구상도**

**4.2. 성능평가**

|  |  |  |
| --- | --- | --- |
| 항목 | 점수/배점 | 비고 |
| 통신속도 | **20/20** | **Arduino-** **Raspberry PI 통신속도 양호** |
| 암-복호화 속도 | **30/30** | **양 측 기기에서 암-복호화 속도 양호** |
| Signature 생성 속도 | **10/20** | **Signature 생성속도 느림.** |
| 안정성 | **15/30** | **두 개의 Arduino Serial 통신 간 약간의 오류가 발생하고는 함.** |
| 총평 | **75/100** | **Signature 생성 과정에 있어서 Chip과의 통신이 존재함. 이 과정에 있어서 속도가 느려지는 듯 함.**  **Arduino 의 자체적인 불안정성을 해결하지 못하여 Serial 통신간 오류 발생 가능성 존재** |

**5. 결론**

**5.1. 기대효과**

Smart Home Service 를 개발해 봄으로써 여러 개의 기기의 통신과정을 살펴보고 Data의 흐름에 익숙해짐.

Open Hardware 인 Arduino 에 보안 알고리즘을 적용해 봄으로써 Arduino Project 진행 시 보안적 요소를 적용할 수 있게끔 함.

Arduino 를 연결하여 Data를 분산처리 함으로써 Arduino 의 Hardware 적 한계 (memory 부족) 을 해결해 보고자 함

**5.2. 추후 연구 방향**

현재 Protocol 의 사용하지 않는 byte 를 활용할 방향을 생각해봄

Arduino 의 불안정성을 해결할 추가적인 Software, Hardware 적 기법을 탐색 및 적용

ATECC508A-SSHCZ-TCT-ND Chip 과의 통신 과정에 있어서 불필요한 부분을 제거하여 Signature 생성 및 전달 과정의 시간을 줄여보도록 함.

추가적인 가전제품을 모형화 하기 위한 Hardware 탐색 및 적용

**6. 참고문헌**

[1] <http://www.hardcopyworld.com/ngine/aduino/index.php/archives/740> ( Arduino Serial 통신 )

[2] <https://github.com/kmackay/micro-ecc> ( micro ECC )

[3] <http://forum.arduino.cc/index.php?topic=88890.0> ( AES Library )

[4] <http://www.atmel.com/devices/ATECC508A.aspx?tab=documents> ( ATECC508A Document )

[5] <https://www.openssl.org/docs/> ( Open SSL )