

## 컴퓨터정보공학 종합설계 과제 중간보고서

과 제 명	증강현실과 블루투스를 이용한 아두이노 도어락				
과제 참가자	학번	학기	성명	email	연락처
	12091554	4-1	송영균	<a href="mailto:zkaka2000@gmail.com">zkaka2000@gmail.com</a>	010-4935-3279
	12101486	4-2	서다영	<a href="mailto:g.latt2a@gmail.com">g.latt2a@gmail.com</a>	010-2886-5368
	12111680	4-2	최정석	<a href="mailto:jeongseokchoi.korea@gmail.com">jeongseokchoi.korea@gmail.com</a>	010-8624-2692
조	1조		대표학생	송영균	
프로젝트 개요	블루투스와 증강현실을 이용하여 스마트폰으로 비밀번호 입력이 가능한 아두이노 도어락 및 앱을 개발하여 보안이 강화된 도어락을 개발한다.				
주요 과제 내용	<ul style="list-style-type: none"><li>○ 아두이노 보드를 이용하여 블루투스 통신이 가능한 도어락을 제작한다.</li><li>○ Vuforia SDK와 Unity를 이용하여 증강현실을 구현한다.</li><li>○ 아두이노 블루투스와 Unity의 블루투스를 활용하여 도어락과 스마트폰의 통신을 가능하게 한다.</li><li>○ 증강현실로 스마트폰에 표현된 번호판을 임의로 위치시키도록 한다.</li></ul>				
최종 결과물	<ul style="list-style-type: none"><li>○ 아두이노 보드가 탑재된 도어락</li><li>○ 비밀번호 입력과 증강현실을 표현하는 안드로이드 앱</li></ul>				
소요 장비 및 SW	<ul style="list-style-type: none"><li>○ 안드로이드 기반 스마트폰</li><li>○ 아두이노 보드</li><li>○ Vuforia SDK</li><li>○ Unity</li><li>○ Arduinocc</li></ul>				

# 목 차

1. 과제 개요 .....	3
1.1 과제 제안 배경 .....	3
1.2 최근 국내외 도어락 현황 .....	3
1.3 과제의 목적 .....	3
2. 과제 수행 내용 .....	4
2.1 설계 내용 .....	4
2.1.1 전체 기능 및 동작흐름 .....	4
2.1.2 아두이노 도어락의 기능 및 동작흐름 .....	5
2.1.3 안드로이드 앱의 기능 및 동작흐름 .....	7
2.2 아두이노 도어락 제작 .....	12
2.2.1 사용 자원 .....	12
2.2.2 하드웨어 개발 내용 .....	13
2.2.3 소프트웨어 개발 내용 .....	14
2.2.4 개발 중 시행착오 .....	17
2.3 안드로이드 앱 개발 .....	18
2.3.1 사용 자원 .....	18
2.3.2 소프트웨어 개발 내용 .....	19
2.3.3 개발 중 시행착오 및 해결방안 .....	33
3. 산출물 .....	34
3.1 아두이노 도어락 .....	34
3.2 안드로이드 앱 .....	37
4. 평가 .....	39
4.1 성능 .....	39
4.2 품질 .....	39
5. 부록 .....	40
5.1 용어정리 .....	40
5.2 참고문헌 .....	42

## 1. 과제 개요

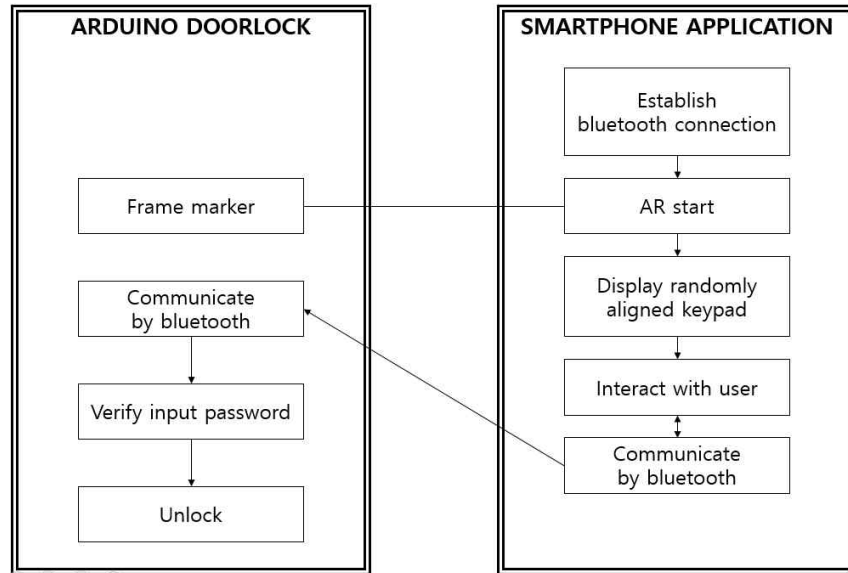
1.1 과제 제안 배경	
필요성	<ul style="list-style-type: none"><li>○ 기존의 도어락은 보안상 취약점이 너무 많다.</li><li>○ 카드나 스마트폰 분실로 인해 문을 열지 못하는 불편함이 있다.</li></ul>
목표	<ul style="list-style-type: none"><li>○ 기존의 도어락에서 보안성을 향상시킨다.</li><li>○ 카드나 스마트폰 분실로 인해 문을 열지 못하는 불편함을 없앤다.</li><li>○ 보다 간편하고 보안성이 강력한 도어락 및 앱을 제작한다.</li></ul>
1.2 최근 국내외 도어락 현황	
기존 도어락 문제점	<ul style="list-style-type: none"><li>○ 버튼 클릭 시 지문이 남음으로 인하여 발생하는 보안상 취약점이 있다.</li><li>○ 버튼을 누르는 손가락의 위치로 번호를 유추할 수 있음으로 인한 보안상 취약점이 있다.</li><li>○ 카드(NFC)나 스마트폰 분실로 인해 쉽게 문이 열리거나 들어갈 수 없는 취약점이 있다.</li></ul>
해결방안	<ul style="list-style-type: none"><li>○ 도어락은 아두이노와 마커만으로 이루어져 있어서 조작할 필요가 없으며 스마트폰에서 버튼을 누르게 함으로서 도어락에 지문이 남지 않도록 제작한다.</li><li>○ 버튼이 증강현실을 통해서 임의로 배치되게 하여 번호를 유추할 가능성을 제거한다.</li><li>○ 스마트폰을 분실하더라도 비밀번호를 모르면 아무나 들어갈 수가 없고 사용자는 다른 스마트폰을 통해서도 비밀번호만 알면 들어갈 수 있기 때문에 편리하고 강한 보안성을 지니게 된다.</li></ul>
1.3 과제 목적	
과제목적	<ul style="list-style-type: none"><li>○ 사용자에게 보다 편리하고 보안성이 강한 도어락을 제공한다.</li></ul>

## 2. 과제 수행 내용

### 2.1 설계 내용

전체기능 및 동작흐름	○ 전체적인 기능		
		주요기능	상세 설명
	어플리케이션	○ 도어락 비밀번호 초기화 및 수정 기능	○ 사용자가 처음 도어락을 사용하거나 사용 중에 비밀번호 변경을 필요로 하는 경우 사용
		○ 블루투스 통신 기능	○ 블루투스를 이용하여 도어락에 비밀번호 등의 정보를 전송
		○ 랜덤 번호 배치 기능	○ 증강현실을 통해 랜덤으로 번호 버튼을 출력
		○ 마커 인식 기능	○ 마커를 인식하여 스마트 폰 화면에 번호를 출력
	아두이노 도어락	○ 비밀번호 복호화 기능	○ 어플리케이션에서 암호화되어 전송된 비밀번호를 복호화
		○ 블루투스 통신 기능	○ 블루투스를 이용하여 어플리케이션에서 비밀번호를 전송 받음
		○ 도어락 오픈 기능	○ 저장된 비밀번호와 전송된 비밀번호를 비교하여 도어락을 오픈

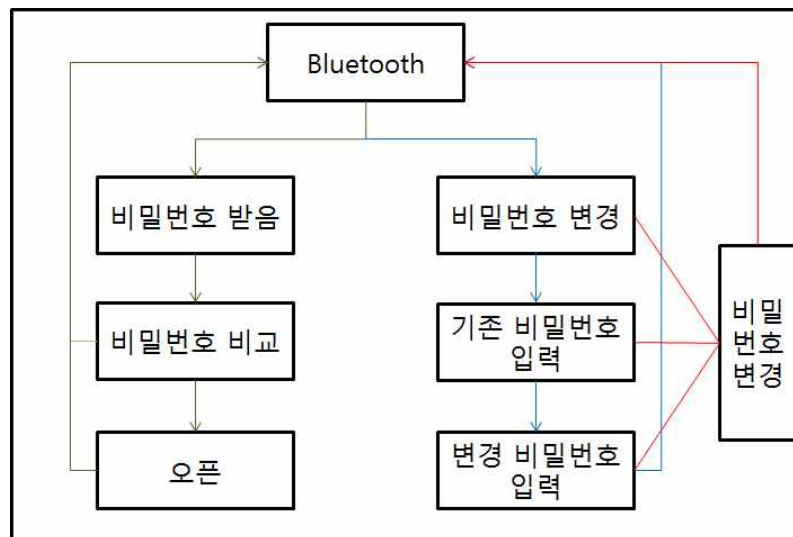
○ 동작흐름



1. 안드로이드 앱이 마커를 인식한다.
2. 증강현실로 번호판이 임의로 배치된다.
3. 블루투스를 실행하고 도어락과 연결한다.
4. 연결된 블루투스를 통해 통신한다.
5. 들어온 번호에 따른 작업을 한다.

아두이노  
도어락의  
기능 및  
동작흐름

○ 동작흐름

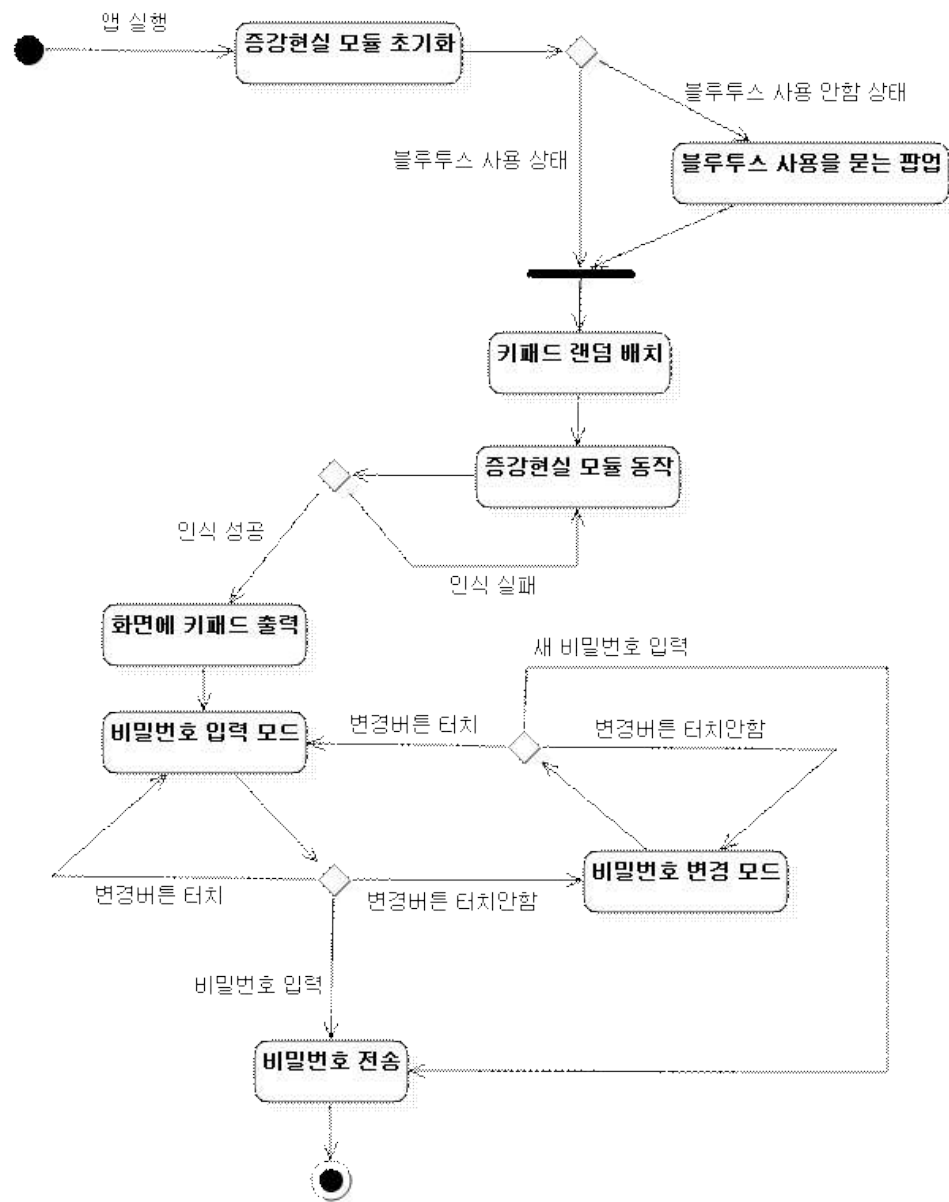


1. 블루투스 통신을 통해 비밀번호를 받거나 비밀번호 변경여부를 받는다.
2. 비밀번호를 받으면 비교하여 오픈할지 안할지 결정한다.
3. 변경신호를 받으면 기존 비밀번호를 입력하고 변경 비밀번호를 입력하여 변경한다.
4. 변경 도중 다시 비밀번호 변경 버튼을 누르면 초기화면으로 돌아가며 1로 돌아가서 지속적으로 번호를 받는다.

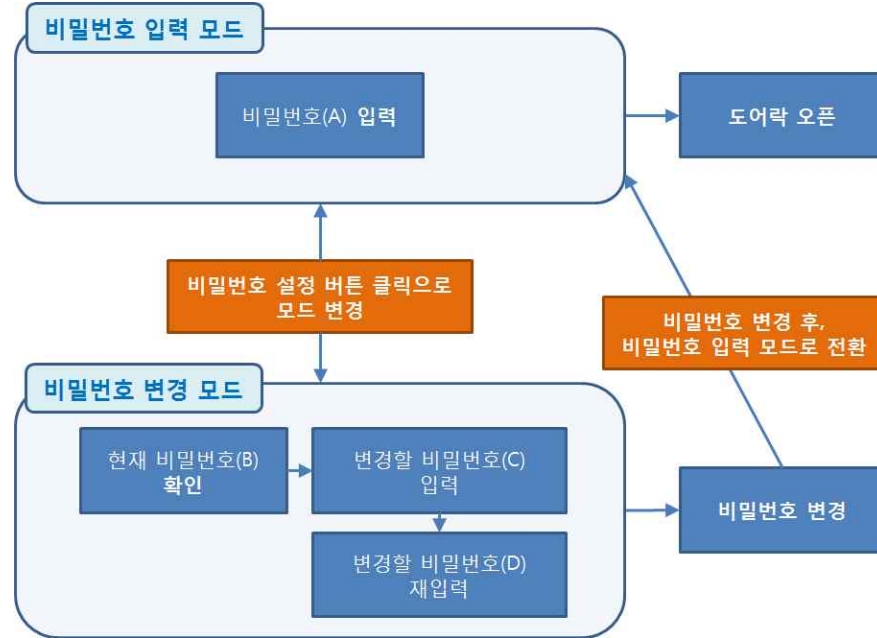
	<ul style="list-style-type: none"> <li>○ 도어락 <ul style="list-style-type: none"> <li>- 스테핑 모터를 이용하여 정확한 각도로 문이 열리도록 도어락을 제작하였다.</li> <li>- 어플리케이션과 블루투스 통신이 이루어진다.</li> <li>- 설정된 비밀번호와 복호화 된 비밀번호를 비교하여 일치하면 오픈한다.</li> <li>- 비밀번호 변경 신호가 들어오면 비밀번호 변경을 실행하며 이때 다시 비밀번호 변경 버튼을 누르면 초기 블루투스 통신화면으로 전환된다.</li> </ul> </li> <li>○ 블루투스 통신 <ul style="list-style-type: none"> <li>- HC-06 모듈을 통해서 블루투스 통신을 실행한다.</li> <li>- SoftwareSerial.h 헤더를 이용하여 자유롭게 블루투스 모듈이 사용할 포트를 정해줄 수 있도록 한다.</li> <li>- 값은 4자리 비밀번호만을 받으며 받은 후에 선언한 배열의 마지막 배열에 NULL값을 넣어준다.</li> </ul> </li> <li>○ 비밀번호 변경 기능 <ul style="list-style-type: none"> <li>- 어플리케이션에서 비밀번호 변경을 하겠다는 신호를 받으면 기존 비밀번호를 입력받고 일치하는지 확인한다.</li> <li>- 만약 기존 비밀번호가 일치하면 변경할 비밀번호를 입력받고 저장을 해둔다.</li> <li>- 완료되면 초기의 블루투스 통신 상태로 돌아간다.</li> </ul> </li> <li>○ LED 기능 <ul style="list-style-type: none"> <li>- 사용자가 좀 더 편리하게 기능을 활용할 수 있도록 LED 2개를 운용한다.</li> <li>- 파란 LED는 비밀번호 변경 상태일 때 ON 되며 완료되면 OFF된다.</li> <li>- 녹색 LED는 비밀번호 변경 상태에서 현재 비밀번호 입력이 맞은 경우 ON되며 완료되면 OFF된다. 또한 도어락 OPEN에 ON되며 CLOSE에 OFF된다.</li> </ul> </li> </ul>
--	---

안드로이드  
앱의 기능 및  
동작흐름

○ 동작흐름



○ 비밀번호 입력 변경 모드 동작흐름



1. 모드 변경

App은 비밀번호 입력 모드와 비밀번호 변경 모드 두 가지가 있다. 모드 변경은 언제든지 ‘비밀번호 변경’ 버튼 클릭으로 변경할 수 있다. 비밀번호 입력 모드에서 ‘비밀번호 변경’ 버튼을 누르면 비밀번호 변경 모드로 전환되고 아두이노 도어락에는 문자열 “abcd”를 전송한다. 비밀번호 변경 모드에서는 어느 과정에서든 ‘비밀번호 변경’ 버튼을 누르면 입력 중이던 Queue 스택이 초기화 되면서 비밀번호 입력 모드로 전환되고 아두이노 도어락에 문자열 ”bcde”를 전송하여 비밀번호 변경 모드가 끝났음을 알린다.

2. 비밀번호 입력 모드

비밀번호 입력 모드에서는 도어락의 비밀번호 4자리와 ‘입력’ 버튼을 클릭하면 입력한 비밀번호(A)를 아두이노 도어락으로 전송한다. 입력한 비밀번호(A)가 4자리가 아닐 경우 아무런 정보도 아두이노 도어락에 전송하지 않으며 도어락도 열리지 않고 어플리케이션에 다시 비밀번호를 입력해야 한다. 전송된 비밀번호(A)와 아두이노 도어락에 저장된 비밀번호가 일치할 경우 아두이노 도어락이 열리고, 틀릴 경우 열리지 않는다.

3. 비밀번호 변경 모드

비밀번호를 변경하기 위해서는 현재 비밀번호(B) 입력, 변경할 비밀번호



호(C) 입력, 변경할 비밀번호(D) 재입력 총 3단계로 구성되어 있다. 현재 비밀번호(B)를 입력 한 후, 변경할 비밀번호를 2번(C, D) 입력한다. 변경할 비밀번호(C)와 재입력한 비밀번호(D)가 일치 할 경우 현재 비밀번호(B)와 변경할 비밀번호(C)를 함께 아두이노 도어락에 전송한다. 변경할 비밀번호(C)와 재입력 비밀번호(D)가 다를 경우 아무런 정보도 전송하지 않는다. 비밀번호 변경 과정이 끝나면 "bcde"를 아두이노 도어락에 전송하여 비밀번호 변경 모드가 끝났음을 알리고 비밀번호 입력 모드로 전환한다.

일반적으로 비밀번호를 변경할 때 현재 비밀번호(B)를 입력하고 일치할 경우 비밀번호 변경을 한다. 하지만 아두이노 도어락에 비밀번호를 전송한 후 저장된 비밀번호와 일치하는지 판단한 후에 이를 다시 어플리케이션에 전송할 수 없다. 따라서 어플리케이션에서 현재 비밀번호(B)와 변경할 비밀번호(C)를 모두 한 번에 받아 아두이노 도어락에 전송하고 현재 비밀번호(B)와 아두이노 도어락에 저장된 비밀번호와 일치하면 전송받은 변경될 비밀번호(C)로 비밀번호를 변경하는 방법을 택하였다.

#### ○ 마커 인식

- 증강현실(Augmented Reality) 기술을 사용하여 카메라를 통해 프레임 마커(Frame Marker)를 인식하고 마커로부터 블루투스 연결과 관련한 정보를 읽어 들여 별도의 연결 형성 과정 없이 자동으로 블루투스 통신이 가능하도록 구현한다.
- 증강현실을 구현하기 위해 Qualcomm사의 VuforiaSDK를 사용하며, 개발 툴로 Unity3D Pro를 사용하므로 VuforiaSDK unity extension을 라이브러리로 사용한다.
- 개발 언어로는 C#을 사용한다.



※ Frame\_Marker

○ 화면에 숫자패드 표시

- 증강현실 모듈이 프레임 마커를 인식하면 화면에 비밀번호 입력을 위한 숫자패드를 표시한다.
- 숫자패드의 숫자 배치 순서는 매번 임의의 순서로 배치되도록 구현한다.



1. 0~9까지 10개의 개체(GameObject) 선언한다.
2. n번째 GameObject에 0~9 사이의 랜덤 번호를 입력한다.
3. 0~n-1번째 GameObject와 n번째 GameObject가 같은 것이 있다면 2번 과정을 반복한다. 같은 것이 하나도 없다면 n+1번째 GameObject로 넘어간다.
4. 10개의 개체에 번호를 입력하고 나면, 입력된 번호에 맞는 x, y 좌표를 지정한다.

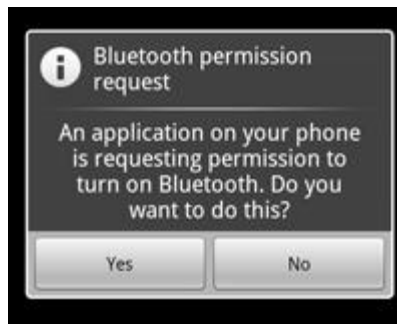


\* 결과 예시

○ 블루투스 통신

- Android.bluetooth API 사용
- 사용법

- ①.getDefaultAdapter()메소드를 이용하여 블루투스 통신 지원 여부 확인
- ②.isEnabled() 메소드로 블루투스 활성화 여부 확인
- ③블루투스가 비활성화인 경우 startActivityForResult()를 이용하여 블루투스 활성화 메시지 요청



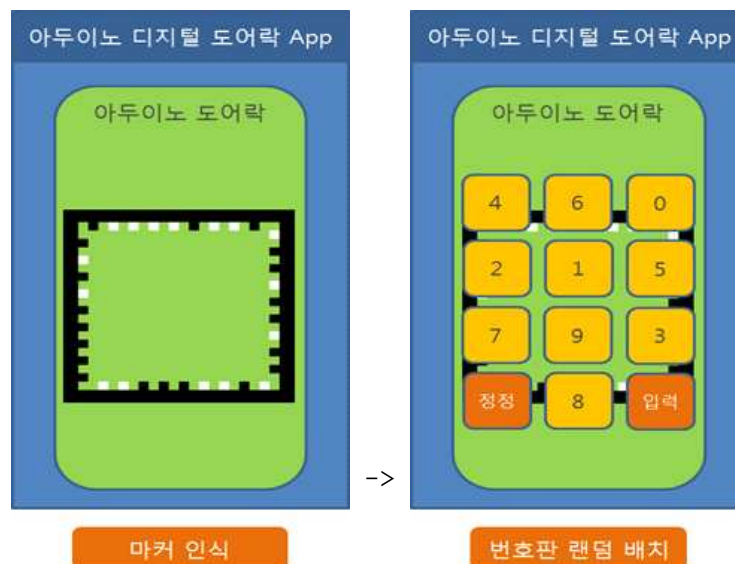
※ 블루투스 활성화 요청 메시지

- ④블루투스 연결 후 도어락 장치와 통신 비밀번호 초기화 및 수정

○ 비밀번호 초기화 및 수정

- 최초 비밀번호는 '1234'으로 설정한다.
- 비밀번호 입력을 통한 권한 인증 과정을 거친 후 새로운 비밀번호를 지정할 수 있도록 구현한다.

○ 유저 인터페이스



- 카메라 화면에 프레임 마커를 인식한 후 랜덤배치 된 번호판을 찍운다.

## 2.2 아두이노 도어락 제작

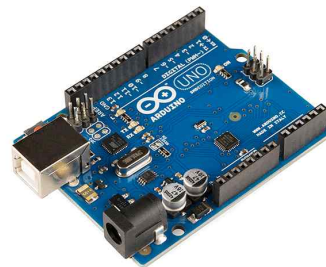
### ○ Arduino

- 오픈 소스를 지향하는 마이크로 컨트롤러(micro controller)를 내장한 기기 제어용 기판. 컴퓨터 메인보드의 단순 버전으로 이 기판에 다양한 센서나 부품 등의 장치를 연결할 수 있다. 컴퓨터와 연결해 소프트웨어를 로드하면 동작을 하게 되므로 제어용 전자 장치부터 로봇과 같은 것을 만들 수 있는 '오픈소스 하드웨어'라고 할 수 있다. 자유 소프트웨어 운동에서 출발한 오픈 소스라는 개념을 하드웨어 부문까지 확산시킨 것이다.

### ○ 스테핑 모터



### ○ 아두이노 보드



### 사용 자원

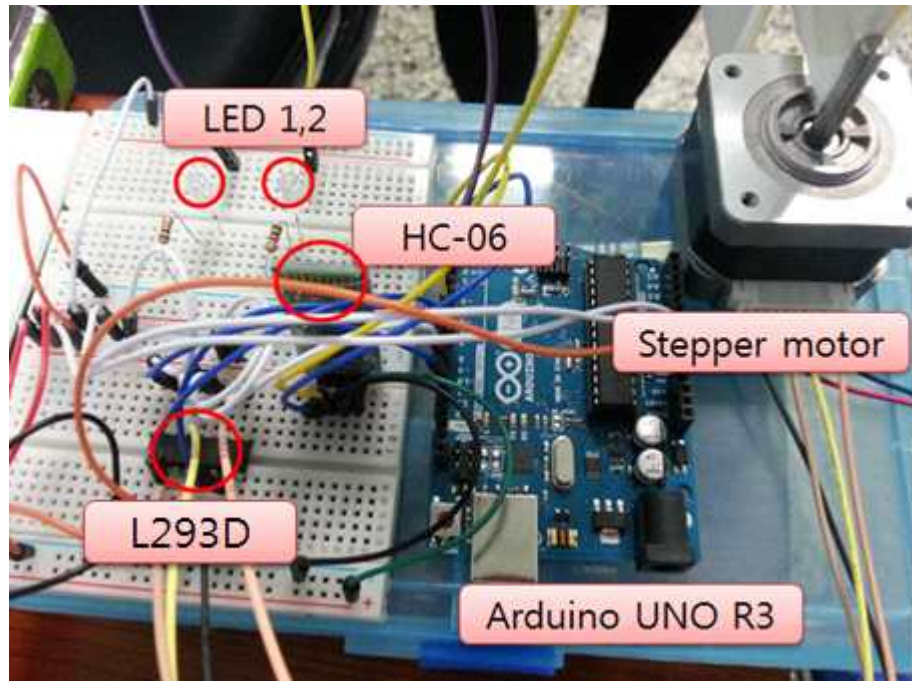
### ○ HC-06 모듈



### ○ 아두이노 IDE(통합개발환경)



○ 도어락 제작



하드웨어  
제작 내용

- Arduino UNO R3 보드를 통해 프로그램이 기본적으로 동작하게 된다.
- HC-06 모듈을 통해서 블루투스 통신이 이루어지게 된다.
- L293D를 통해 Stepper motor가 보드에서 받은 명령을 제대로 수행할 수 있게 한다.
- Stepper motor를 통해 도어락이 정확한 각도로 회전하면서 문을 OPEN 및 CLOSE를 할 수 있게 한다.
- LED1,2를 통해서 사용자가 좀 더 편하게 어플리케이션 및 도어락을 사용할 수 있도록 한다.
- 블루투스 통신을 위해서 RX선과 TX 선이 보드와 HC-06모듈이 서로 연결되어 있는 상태이다.
- LED1,2가 안전하게 운용되게 하기위해서 1k옴의 저항을 사용하여 가해지는 전력을 낮춰 주었다.
- Stepper motor에 우드락으로 제작한 문을 부착하여 시연에 용이하도록 제작하였다.

소프트웨어  
개발 내용

1. 헤더부분

```
#include <string.h>
#include <Stepper.h>
#include <SoftwareSerial.h>
```

- string.h는 입력받은 비밀번호와 저장된 비밀번호를 비교할 때 strcmp를 활용하기 위해서 선언하였다.
- Stepper.h는 스테퍼 모터의 운용을 위해 선언하였다.
- SoftwareSerial.h는 블루투스를 운용하기 위해 선언하였다.

2. 변수 선언

```
const int stepsPerRevolution =200;
Stepper myStepper(stepsPerRevolution, 8,9,10,11);
int revolution=100;
int r_revolution=-100;
```

- 스테퍼 모터의 핀 번호를 설정하고 회전 정도를 설정한다.

```
int led1 = 6;
int led2 = 7;
int led1_status = LOW;
int led2_status = LOW;
```

- LED의 핀번호를 지정하고 상태를 LOW로 지정하여 꺼진상태로 둔다.

```
int bluetoothTx = 0;
int bluetoothRx =1;
SoftwareSerial bluetooth(bluetoothTx, bluetoothRx);
```

- 블루투스의 핀 번호를 설정하고 bluetooth라는 함수로 운용하겠다고 선언한다.

```
char PW[5] = "1234";
char data[5] = {0};
char Change_PW[5] = "abcd";
char First_Page[5] = "bcde";
char flag = '1'; // base flag(distinguish input_pw, change_pw_pw, change_pw)
```

- PW는 저장된 PassWord이다.
- data는 블루투스로 입력되는 비밀번호를 저장하는 곳이다.
- Change\_PW는 비밀번호 변경 신호를 abcd로 하여 abcd가 들어오면 flag를 2로 하여 비밀번호 변경을 하겠다는 것으로 인식한다.
- First\_Page는 비밀번호 변경 모드에서 bcde라는 신호가 들어오면 변경 모드를 취소하고 flag를 1로 하여 초기화면 모드로 돌아간다.
- flag는 처음 비밀번호 입력모드라면 1, 변경모드라면 2, 변경모드에서 기존 비밀번호비교가 끝나고 비밀번호를 변경할 시에는 3으로 하여 운용이 된다.

### 3. setup() 함수 선언

```
void setup()
{
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  digitalWrite(led1, LOW);
  digitalWrite(led2, LOW);
  myStepper.setSpeed(100);
  Serial.begin(9600);
  delay(100);
  bluetooth.begin(9600);
}
```

- setup함수는 loop를 돌기 전에 초기화하는 곳이다.
- led1, led2를 사용할 것이기 때문에 핀을 설정하고 LOW로 하여 꺼진상태로 시작하도록 한다.
- myStepper.setSpeed()를 통해서 스텝퍼 모터가 도는 속도를 조절할 수 있다.
- Serial.begin, bluetooth.begin을 통해서 loop와 블루투스를 시작한다.

### 4. loop안에서의 동작

```
if(bluetooth.available())
{
  for(int i=0; i<5 ; i++)
  {
    data[i] = (char)bluetooth.read();
    delay(100);
  }
  data[4]='#0';
}
```

- bluetooth.available을 통해 블루투스로 전송이 된다면 if문 안으로 들어가게 됩니다. 들어오는 비밀번호의 개수는 4개이기 때문에 for문을 통해 받고 data[4]에 NULL을 넣어준다.

```
if((strcmp(PW,data)==0)&&(flag=='1'))
{
  Serial.println("Open");
  myStepper.step(revolution);
  digitalWrite(led2, HIGH);
  led2_status = HIGH;
  delay(2000);
  myStepper.step(-revolution);
  digitalWrite(led2, LOW);
  led2_status = LOW;
}
```

- strcmp를 통해 PW와 data를 비교하고 flag가 1이므로 비밀번호 입력 부분임을 알고 실행합니다. myStepper.step 함수를 통해서 스텝퍼 모터를 회전 시키고 digitalWrite로 led2를 HIGH로 조정해 줌으로써 led2가 ON되도록 합니다. 그리고 delay(2000)동안 OPEN하고 그 후에 다시 닫고 led2를 OFF 시킨다.

```
else if((strcmp(Change_PW,data)==0)&&flag=='1')
{
    Serial.println("Change_PW");
    digitalWrite(led1, HIGH);
    led1_status = HIGH;
    flag = '2';
}
```

- strcmp를 통해 abcd와 같은 데이터가 비밀번호 입력상태(flag=1)에서 들어오면 led1을 ON하고 flag를 2로 바꿈으로써 비밀번호 변경상태가 된다.

```
else if((strcmp(PW,data)==0)&&flag=='2')
{
    Serial.print("ReCheck_PW\n");
    digitalWrite(led2,HIGH);
    led2_status = HIGH;
    flag = '3';
}
```

- PW와 data가 같다면 기존 비밀번호를 알고 있다는 것이므로 led2를 ON으로 하고 flag를 3으로 변경할 비밀번호를 입력하라는 신호를 준다.

```
else if(flag=='3')
{
    for(int i=0; i<5 ; i++)
    {
        PW[i] = data[i];
    }
    PW[4]='\0';
    flag = '1';
    digitalWrite(led1,LOW);
    led1_status = LOW;
    digitalWrite(led2,LOW);
    led2_status = LOW;
    Serial.println("Changed PW");
    Serial.println(PW);
}
```



	<ul style="list-style-type: none"> <li>- flag가 3이라면 변경할 비밀번호를 입력받고 PW에 저장한다. 저장 후 flag를 1로 하여 비밀번호 입력상태로 돌아가고 led1,led2를 모두 off하여 비밀번호 변경이 완료되었음을 알린다.</li> </ul> <pre> else if(strcmp(First_Page,data)==0) {     Serial.println("First_Page");     digitalWrite(led1, LOW);     led1_status = LOW;     digitalWrite(led2,LOW);     led2_status = LOW;     flag = '1'; } </pre> <ul style="list-style-type: none"> <li>- 만약 비밀번호 변경 상태에서 First_Page로 돌아가고 싶다면 비밀번호 변경 버튼을 다시 눌러서 bcde를 받게 되고 그에 따라 flag를 1로 수정하면서 led1,led2가 모두 OFF되어 초기 상태로 돌아갔다는 것을 알려준다.</li> </ul>
<p>개발 중 시행착오</p>	<ul style="list-style-type: none"> <li>○ 암호화 문제 <ul style="list-style-type: none"> <li>- 어플리케이션에서의 AES암호화와 아두이노에서의 AES암호화가 서로 다른 결과를 나타내었고 이 문제를 해결하지 못해서 암호화는 삭제하였다.</li> </ul> </li> <li>○ 하드웨어 문제 <ul style="list-style-type: none"> <li>- 임베디드의 특성상 오류가 나면 하드웨어의 문제인지 코드의 문제인지 알기 어려운 점이 있다. 이 때문에 중간에 모터설드를 교체하는 등의 시행착오를 겪게 되었다.</li> <li>- 하드웨어 장비에 따라 보드에 사용되는 핀이 정해져 있어서 여러 개의 장비를 한번에 운용하는데 어려움이 발생하였다. LED를 현재 2개만 사용하였는데 원래 1개를 더 사용하여 붉은색으로 사용자에게 좀더 편의를 주고 싶었지만 핀의 부족으로 2개만 사용하게 되었다.</li> <li>- 꼭 써야하는 하드웨어 장비의 핀 번호가 겹치는 경우가 발생하였는데 핀 번호를 변경하는 함수를 찾아서 이를 해결하였다.</li> <li>- IDE에서 프로그램을 로드할 때 0번과 1번 핀을 통해 로드가 실행된다. 하지만 블루투스 RX,TX 핀 또한 0번과 1번을 활용하기 때문에 이 둘이 충돌하는 현상이 발생하여 로드가 안되는 일도 발생하였다. 이 일은 핀을 잠시 빼놓고 로드함으로써 해결하였다.</li> </ul> </li> </ul>

## 2.3 안드로이드 앱 개발

### 사용 자원

- 안드로이드 기반 스마트폰
  - 스마트폰(smartphone)은 PC와 같은 기능과 더불어서 최고급 기능을 제공하는 휴대 전화이며[1] 안드로이드(Android)는 휴대 전화를 비롯한 휴대용 장치를 위한 운영 체제와 미들웨어, 사용자 인터페이스 그리고 표준 응용 프로그램(웹 브라우저, 이메일 클라이언트, 단문 메시지 서비스(SMS), 멀티미디어 메시지 서비스(MMS)등)을 포함하고 있는 소프트웨어 스택이자 모바일 운영 체제이다. 개발 과정에서 안드로이드가 탑재된 스마트폰을 테스트 장비로 사용하였다.

- Unity
  - 유니티(영어: Unity)는 3D 비디오 게임이나 건축 시각화, 실시간 3D 애니메이션 같은 기타 인터랙티브 콘텐츠를 제작하기 위한 통합 저작 도구이다. 에디터는 윈도우와 맥 OS X 상에서 실행되어 윈도우나 맥, Wii, 아이패드, 아이폰 플랫폼으로 게임을 만들 수 있다. 유니티 웹 플레이어 플러그인을 이용하는 웹 브라우저 게임도 제작할 수 있다. 이는 플래시와 유사한 형태이며, 크로스 도메인 보안정책 및 스크립팅에서도 플래시 사용자가 쉽게 적응할 수 있도록 설계되었다.



- Vuforia SDK
  - Vuforia는 Qualcomm社에서 제작한 증강현실 SDK이다. Qualcomm社は 증강현실 API로써 안드로이드, iOS, Unity extension 총 3가지 버전을 제공한다. 우리는 이번 과제 수행에 있어 Unity를 개발 툴로서 사용하기로 결정했기 때문에 Unity extension을 사용했다.



- Android & Microcontrollers / Bluetooth assets
  - ‘Android & Microcontrollers / Bluetooth’는 블루투스 통신을 위해 안드로이드 앱과 HC-05, HC-06 모듈이 장착된 아두이노 보드 간의 블루투스 통신을 할 수 있도록 하는 유니티 assets(유니티에서 사용되는 플러그인을 이르는 말)이다. 이 플러그인은 BtConnection과 BtConnector, 총 2개의 클래스를 제공한다. BtConnection은 블루투스 통신을 위한 함수들을 정의해놓은 클래스이며 BtConnector는 BtConnection에 있는 함수들을 wrapping하여 보다 안전하게 사용할 수 있도록 정리해 놓은 클래스이다. 이 과제에서는 BtConnection을 직접 사용하지 않고 BtConnector를 사용해 블루투스 연결 및 통신을 수행한다.

소프트웨어  
개발 내용

#### 1. 증강현실 & 블루투스 연결

```
using UnityEngine;
using System.Timers;

/// <summary>
/// A custom handler that implements the ITrackableEventHandler interface.
/// </summary>
public class CustomTrackableEventHandler : MonoBehaviour, ITrackableEventHandler
{
    bool firstDetected = false;

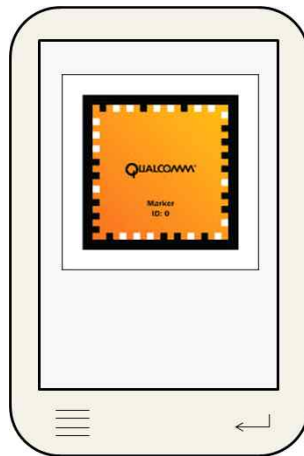
    #region PRIVATE_MEMBER_VARIABLES
    private TrackableBehaviour mTrackableBehaviour;
    #endregion // PRIVATE_MEMBER_VARIABLES

    #region UNITY_MONOBEHAVIOUR_METHODS
    void Start()
    {
        mTrackableBehaviour = GetComponent<TrackableBehaviour>();
        if (mTrackableBehaviour)
        {
            mTrackableBehaviour.RegisterTrackableEventHandler(this);
        }
    }
    #endregion // UNITY_MONOBEHAVIOUR_METHODS
}
```

[그림 3] 증강현실 초기화 과정

- Vuforia SDK는 기본적으로 event handler를 제공한다. [그림 3]의 노란색 상자는 Qualcomm社가 기본적으로 제공해주는 부분이며, 증강현실 기능의 초기화 단계이다. Start 함수는 Unity에서 사용되는 함수로서 시작 시에 단 한 번 호출되는 생성자와 비슷한 개념의 함수이다. 이 함수에서는 증강현실 마커 인식 시에 필요한 TrackableBehaviour 객체를 생성하고 event handler에 등록시키는 간단한 초기화 동작을 해준다.

빨간색 상자는 개발 과정에서 필요에 의해 추가로 삽입한 자체 개발 부분이다. firstDetected 변수는 증강현실 모듈이 마커를 한 번이라도 인식하게 된다면 true로 전환되며, 프로그램이 종료될 때까지 그 상태를 유지한다. 그 이유는 아래와 같다.



[그림 4] 마커 인식 전의 스마트폰 화면



[그림 5] 마커 인식 중의 스마트폰 화면

```

/// <summary>
/// Implementation of the ITrackableEventHandler function called when the
/// tracking state changes.
/// </summary>
public void OnTrackableStateChanged(
    TrackableBehaviour.Status previousStatus,
    TrackableBehaviour.Status newStatus)
{
    if (newStatus == TrackableBehaviour.Status.DETECTED ||
        newStatus == TrackableBehaviour.Status.TRACKED ||
        newStatus == TrackableBehaviour.Status.EXTENDED_TRACKED)
    {
        if (BtConnector.isConnected() == false)
            BtConnector.connect();

        OnTrackingFound();
    }
    else
    {
        if (firstDetected == false)
        {
            OnTrackingLost();
            firstDetected = true;
        }
    }
}
#endregion // PUBLIC_METHODS

```

[그림 6] 증강현실 기능의 메인 기능을 하는 함수

- [그림6]은 증강현실 모듈에서 가장 핵심이 되는 OnTrackableStateChanged 함수의 전문이다. 노란색 상자는 Qualcomm社에서 기본적으로 제공하는 부분이고, 빨간색 상자는 사용자 편의성을 위해 추가된 부분이며, 하얀색 상자는 블루투스 연결을 위해 추가된 부분이다.

노란색 상자의 조건을 살펴보면 DETECTED, TRACKED, EXTENDED\_TRACKED인 경우 OnTrackingFound()함수를 호출하여 대상이 인식되었을 때의 동작을 실행하는 부분이다.

EXTENDED\_TRACKED는 이 과제에서 사용되지 않으니 무시하자.

DETECTED는 대상을 전혀 인식하지 못하던 상태에서 대상을 새로이 인식했을 때 true가 되는 변수이고, TRACKED는 대상이 인식되어있는 중에 대상의 위치가 변경되었을 때 true가 되는 변수이다. 즉 노란색 상자의 조건을 만족하는 경우는 모두 프레임 마커가 성공적으로 인식이 되었을 경우이다.

빨간색 상자 부분에 들어가기 위해서는 노란색 상자의 조건을 만족하지 않아야 한다. 즉 증강현실의 대상이 아예 인식이 되지 않은 상태일 때 빨간색 상자 부분으로 실행 흐름이 넘어가게 된다. 앱이 실행될 때

에는 카메라가 마커를 아무리 빠르게 인식한다 해도 처음에 한 번은 빨간색 부분으로 실행이 넘어가게 되어있고, 그 때 OnTrackingLost() 함수를 호출한 후 [그림 3]에서 정의한 firstDetected 변수를 true로 전환시킨다. firstDetected 변수를 false로 다시 전환하는 코드는 없으므로 OnTrackingLost() 함수는 다시 호출 될 일이 없다.

OnTrackingLost() 함수는 증강현실 인식 성공 시에 출력되는 개체들을 화면에서 지워주는 역할을 한다. 따라서, 처음 앱을 실행시켰을 때 화면에 아무것도 출력되지 않게 하기 위해 firstDetected 변수를 false로 지정해 OnTrackingLost() 함수를 한 번 호출하는 것이다. 마커를 한 번이라도 인식한 후에 카메라가 마커를 비추지 않거나 마커에서 멀어지는 경우 키패드가 사라져 버린다면 사용자가 마커를 다시 인식시켜야하는 불편함 겪게 되므로 빨간색 상자 부분을 추가한 것이다.

하얀색 상자 부분은 블루투스 연결을 하는 부분이며, 코드의 내용은 매우 단순하다. 블루투스 연결이 성립되어있지 않을 때 블루투스 연결 요청을 보내는 것이다. 노란색 상자의 조건 아래에 있으므로 증강현실 대상이 인식 될 때마다 하얀색 상자 부분을 실행하게 된다. 블루투스 연결을 이와 같이 반복적으로 수행될 수 있게끔 구현한 이유는 안정성 때문이다. 블루투스 연결은 어쩔 수 없이 어느 정도의 지연이 발생하는 작업이고 성공률 또한 100%가 아니다. 필요에 따라 연결 요청을 여러 번 보낼 수도 있는 일인데, 버튼을 제공하여 사용자가 직접 연결을 하도록 하는 것은 사용자로 하여금 불편함을 느낄 수 있는 부분이다. 이를 해결하기 위해 마커를 인식하고 트래킹하는 동안 블루투스가 연결되어있지 않은 상태라면 여러 번 연결 요청을 보내도록 구현했고, 사용자가 인식하지 못하는 시간 안에 연결 문제를 해결할 수 있도록 한 것이다.

○ 비밀번호 입력 & 블루투스 통신

- 비밀번호 입력을 받기 위해서 버튼마다 스크립트 파일을 작성하고, 버튼이 눌릴 때마다 해당하는 숫자를 큐에 입력하여 '입력'버튼이 눌렸을 때 큐에 저장되어있는 숫자들을 하나의 비밀번호로 재구성하여 도어락으로 전송하는 방식을 사용했다.

```
public class Button1 : MonoBehaviour {  
  
    // Use this for initialization  
    void Start () {  
    }  
  
    // Update is called once per frame  
    void Update () {  
    }  
  
    void OnMouseDown()  
    {  
        InputQueue.queue.Enqueue (1);  
    }  
}
```

[그림 7] 1번 버튼에 삽입된 스크립트

```
public static class InputQueue {  
  
    public static Queue queue = new Queue();  
}
```

[그림 8] 비밀번호를 임시 저장하는 큐

```
if (Input.GetMouseButtonDown (0)) {  
    while (InputQueue.queue.Count > 0) {  
        password += ((int)(InputQueue.queue.Dequeue ())).ToString ();  
    }  
}  
  
BtConnector.sendString(password);
```

[그림 9] 입력 버튼에 삽입된 스크립트의 일부

- [그림 7]의 빨간색 상자 부분이 입력받은 비밀번호를 큐에 임시 저장하는 부분이다. OnMouseDown() 함수는 마우스 클릭 또는 터치 입력 이벤트가 발생할 때 동작하는 이벤트 처리 함수이다. 1번 버튼이 눌렸을 때 큐에 정수 1을 입력하고 있으며, 다른 번호에 해당하는 버튼들도 모두 [그림 7]과 같은 스크립트를 하나씩 가지고 있다. [그림 8]에서는 비밀번호를 임시저장하기 위해 사용되는 큐를 정적으로 정의해 놓은 것을 볼 수 있다. 큐는 앱 하나에 여러개 존재할 필요가 없고 여러 개의 버튼 객체에서 사용하게 되는 이유로 정적 바인딩을 했다.

	<p>- [그림 9]는 입력 버튼 스크립트의 일부이다. GetMouseButtonDown() 함수는 위에서 설명한 OnMouseDown()과 마찬가지로 마우스 클릭 및 터치 입력을 위해 사용되며 입력되었을 때 true가 되어 if문의 조건을 만족시키게 된다. 입력 버튼이 터치 되었을 때 앱 내부에서는 비밀번호 임시 저장용 큐에서 데이터를 하나씩 꺼내어 하나의 완성된 비밀번호 문자열을 만들고 블루투스 통신 모듈을 사용해 아두이노 도어락으로 전송되게 된다.</p>
--	---



## 2. 번호판 랜덤 배치 : Buttons.cs

### 2-1. 변수 선언

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class Buttons : MonoBehaviour {
5
6     int i=0, j, random_result;
7     int[] temp = new int[10];
8     bool bool_tmp = false;
9
10    public GameObject[] menu_object = new GameObject[10];
11
```

- int random\_result;

// 랜덤으로 생성된 숫자를 저장하는 변수

- int[] temp = new int[10];

// 바뀔 위치 변수

- bool bool\_tmp = false

// temp값이 중복되지 않도록 bool을 이용하여 루프를 돈다.

- public GameObject[] menu\_object = new GameObject[10];

실제 번호판 각각을 가리키는 오브젝트로 GameObject[1]은 숫자 '1' 버튼을 가리킨다.

### 2-2. Start()

Start() 함수는 프로젝트 실행 시 처음 한 번만 실행된다. Start()에서 temp값을 초기화 시킨다.

```
12 // Use this for initialization
13 void Start () {
14
15     // MENU NUMBER random setting
16     for( i=0; i<10; i++){
17
18         bool_tmp = false;
19
20         // bool_tmp가 true일 때까지 랜덤번호 생성
21         while( bool_tmp != true ){
22
23             random_result = Random.Range (0,10); // 랜덤번호 생성
24             bool_tmp = true; // 이전의 번호와 같지 않다면 true 유지
25
26             // 이전 temp값들과 같은 번호가 있는지 체크
27             for( j=0; j<i; j++){
28                 if( random_result == temp[j] ){ // 이전 temp값들과 같다면
29                     bool_tmp = false; // while 루프 계속 유지
30                     break; // temp값 체크 종료
31                 }
32             }
33         }
34
35         temp[i] = random_result; // 이전 temp값들과는 다른 번호를 temp[i]에 저장
36     }
37 }
```

### 2-3. Update()

Start()함수가 처음 한 번 실행된 후에 Update()함수가 실행된다. 번호판의 좌우가 x좌표, 위아래가 z좌표이므로 x, z좌표만 변경한다.

```
39 // Update is called once per frame
40 void Update () {
41
42     for (i=0; i<10; i++) {
43
44         // 원래 버튼의 x,y,z좌표 저장
45         float temp_X = menu_object[i].transform.position.x;
46         float temp_Y = menu_object[i].transform.position.y;
47         float temp_Z = menu_object[i].transform.position.z;
48
49         // x좌표 변경
50         // temp[i] 값이 1,4,7 이면 x좌표를 -180으로 변경
51         // temp[i] 값이 2,5,8,0 이면 x좌표를 0으로 변경
52         // temp[i] 값이 3,6,9 이면 x좌표를 180으로 변경
53         if( temp[i] == 1 || temp[i] == 4 || temp[i]== 7 )
54             menu_object[i].transform.position = new Vector3(-180.0f, temp_Y, temp_Z);
55         else if( temp[i] == 2 || temp[i] == 5 || temp[i] == 8 || temp[i] == 0 )
56             menu_object[i].transform.position = new Vector3(0.0f, temp_Y, temp_Z);
57         else if( temp[i] == 3 || temp[i] == 6 || temp[i] == 9 )
58             menu_object[i].transform.position = new Vector3(180f, temp_Y, temp_Z);
59
60         temp_X = menu_object[i].transform.position.x;
61
62         // z좌표 변경
63         // temp[i] 값이 1,2,3 이면 z좌표를 360으로 변경
64         // temp[i] 값이 4,5,6 이면 z좌표를 180으로 변경
65         // temp[i] 값이 7,8,9 이면 z좌표를 0으로 변경
66         // temp[i] 값이 0 이면 z좌표를 -180으로 변경
67         if( temp[i] == 1 || temp[i] == 2 || temp[i] == 3 )
68             menu_object[i].transform.position = new Vector3(temp_X, temp_Y, 360.0f);
69         else if( temp[i] == 4 || temp[i] == 5 || temp[i] == 6 )
70             menu_object[i].transform.position = new Vector3(temp_X, temp_Y, 180.0f);
71         else if( temp[i] == 7 || temp[i] == 8 || temp[i] == 9 )
72             menu_object[i].transform.position = new Vector3(temp_X, temp_Y, 0.0f);
73         else if( temp[i] == 0 )
74             menu_object[i].transform.position = new Vector3(temp_X, temp_Y, -180.0f);
75     }
76
77
78 }
79 }
```

### 3. 비밀번호 입력 및 변경

Filename	Button	Filename	Button
InputQueue.cs	-	Button7.cs	7
Buttons.cs	숫자 0~9	Button8.cs	8
Button1.cs	1	Button9.cs	9
Button2.cs	2	Button10.cs	정정
Button3.cs	3	Button11.cs	0
Button4.cs	4	Button12.cs	입력
Button5.cs	5	Button13.cs	비밀번호변경
Button6.cs	6	-	-

- InputQueue.cs에서 queue를 생성한다.

```
public static Queue queue = new Queue();
```

- 각 숫자버튼을 나타내는 파일에서 enqueue()함수를 이용하여 숫자버튼을 클릭할 때 queue에 해당 번호가 스택으로 쌓인다.

**Button0.cs :**

```
void OnMouseDown() {
    InputQueue.queue.Enqueue (1);
}
```

**Button9.cs :**

```
void OnMouseDown() {
    InputQueue.queue.Enqueue (9);
}
```

- 정정 파일(Button10.cs)에서 Clear()함수를 이용해 queue에 쌓인 번호를 모두 비운다.

```
void OnMouseDown() {
    InputQueue.queue.Clear ();
}
```

- 입력 파일(Button12.cs)에서 queue에 저장된 번호들을 password로 복사한다.

```
while (InputQueue.queue.Count > 0) {
```

```
password += ((int)(InputQueue.queue.Dequeue ())).ToString ();  
}
```

### 3-1. InputQueue.cs

```
1 using UnityEngine;  
2 using System.Collections;  
3  
4 public static class InputQueue {  
5  
6     public static Queue queue = new Queue ();  
7  
8 }
```

- queue를 생성한다. 0~9버튼을 클릭 시 queue에 저장한다.

### 3-2. Button1.cs ~ Button9.cs, Button11.cs (0~9버튼)

```
1 using UnityEngine;  
2 using System.Collections;  
3  
4 public class Button1 : MonoBehaviour {  
5  
6     // Use this for initialization  
7     void Start () {  
8     }  
9  
10    // Update is called once per frame  
11    void Update () {  
12    }  
13  
14    void OnMouseDown() {  
15        InputQueue.queue.Enqueue (1);  
16    }  
17  
18 }
```

- void OnMouseDown()은 해당 버튼이 클릭될 때 실행되는 함수이다. '1' 버튼이 클릭될 때 InputQueue.queue.Enqueue(1); 를 실행하여 Input.Queue에서 생성한 queue에 1을 저장한다. '2'버튼이 클릭되면 InputQueue.queue.Enqueue(2); 를 실행하여 queue에 2를 저장한다. 0~9 버튼을 클릭하면 해당 번호가 queue에 삽입된다.

### 3-3. Button10.cs

- '정정' 버튼으로 클릭 시 InputQueue.queue.Clear();를 실행하여 queue에 저장되어 있던 번호들을 지운다.

```

1 using UnityEngine;
2 using System.Collections;
3
4 public class Button10 : MonoBehaviour {
5
6     // Use this for initialization
7     void Start () {
8     }
9
10    // Update is called once per frame
11    void Update () {
12    }
13
14    void OnMouseDown() {
15        InputQueue.queue.Clear ();
16    }
17 }

```

#### 3-4. Button12.cs

- '입력' 버튼으로 클릭 시 0~9번을 클릭하여 queue에 쌓은 번호들을 처리한다.

##### 3-4-1. 변수선언

```

1 using UnityEngine;
2 using System.Collections;
3
4 public class Button12 : MonoBehaviour {
5
6     public string password;
7     public static string change_password;
8     public static string change_password_confirm;
9
10    public static bool b_pw = false;
11    public static bool b_checkPw = false;
12

```

- `public string password;`  
// 입력된 비밀번호(A, B)
- `public static string change_password;`  
// 변경할 비밀번호(C)
- `public static string change_password_confirm;`  
// 변경할 비밀번호 재입력(D)
- `public static bool b_pw = false;`  
비밀번호를 변경할 때 변경할 비밀번호(C), 변경할 비밀번호 재입력(C)을 확인하는 bool이다. 기본 false이고 변경할 비밀번호(C)를

입력 후에 true로 바뀐다. 변경할 비밀번호 재입력 후나 비밀번호 변경 모드가 끝나면 다시 false로 초기화한다.

- `public static bool b_checkPw = false;`

비밀번호를 변경 모드에서 현재 비밀번호(B)를 입력할 때 확인하는 bool이다. 기본 false이고 현재 비밀번호(B)를 입력한 후에 true로 바뀐다. 비밀번호 변경 모드가 끝나면 false로 초기화한다.

### 3-4-2. 변수 초기화

문자열 변수들을 공백으로 초기화한다.

```
13
14     // Use this for initialization
15     void Start () {
16
17         password = "";
18         change_password = "";
19         change_password_confirm = "";
20
21     }
22
23     // Update is called once per frame
24     void Update () {
25
26     }
27
```

### 3-4-3. OnMouseDown() 함수

```
void OnMouseDown() {

    // queue의 4자리가 입력되지 않았다면 queue를 초기화하고 끝낸다.
    if (InputQueue.queue.Count != 4) {
        InputQueue.queue.Clear ();
        return;
    }

    // password의 queue를 복사하기 전에 초기화
    password = "";

    // 비밀번호 입력 모드일 경우
    if ( Button13.b_changePw == false)
    {
        if (Input.GetMouseButtonDown(0)) // queue의 저장된 번호들을 password에 복사한다.
        {
            while (InputQueue.queue.Count > 0) {
                password += ((int)(InputQueue.queue.Dequeue ())).ToString ();
            }
        }

        BtConnector.sendString(password); // password를 블루투스 통신으로 아두이노 도어락에 전송
    }
}
```

```

// 비밀번호 변경 모드일 경우
else
{
    // 비밀번호 변경 모드에서 현재 비밀번호 입력
    if (b_checkPw == false)
    {
        if (Input.GetMouseButtonDown (0)) {
            while (InputQueue.queue.Count > 0) {
                password += ((int)(InputQueue.queue.Dequeue ())).ToString ();
            }
        }

        BtConnector.sendString(password); // 현재 비밀번호를 아두이노 도머락에 전송

        b_checkPw = true; // b_checkPw를 true로 바꿔 변경할 비밀번호를 입력할 수 있도록 한다.
    }

    // 비밀번호 변경 모드에서 변경할 비밀번호 입력
    else
    {
        // 변경할 비밀번호 입력하여 change_password에 복사
        if ( b_pw == false ){
            if (Input.GetMouseButtonDown (0)) {
                while (InputQueue.queue.Count > 0) {
                    change_password += ((int)(InputQueue.queue.Dequeue ())).ToString ();
                }
            }

            b_pw = true; // b_pw를 true로 바꿔 변경할 비밀번호를 재입력할 수 있도록 한다.
        }

        // 변경할 비밀번호 재입력하여 change_password_confirm에 복사
        else {
            if (Input.GetMouseButtonDown (0)) {
                while (InputQueue.queue.Count > 0) {
                    change_password_confirm += ((int)(InputQueue.queue.Dequeue ())).ToString ();
                }
            }
        }

        // 두번 입력한 비밀번호가 같으면 password 전송
        if (change_password == change_password_confirm)
        {
            password = change_password_confirm;
            BtConnector.sendString(password);
        }

        // 두번 입력한 비밀번호가 다르면 "bode" 전송
        else{
            BtConnector.sendString("bode");
        }

        // 비밀번호 변경 과정이 끝난 후 초기화
        InputQueue.queue.Clear ();
        b_pw = false;
        b_checkPw = true;
        Button13.b_changePw = false;
        change_password = "";
        change_password_confirm = "";
    }
}
}
}

```



### 3-5. Button13.cs

- '비밀번호변경' 버튼으로 비밀번호 입력 모드/비밀번호 변경 모드로 전환한다.

#### 3-5-1. 변수선언

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class Button13 : MonoBehaviour {
5
6     public static bool b_changePw = false;
7 }
```

- `public static bool b_changePw = false;`

비밀번호 입력/변경 모드를 확인하는 bool이다. 기본으로 비밀번호 입력 모드인 false이고 '비밀번호변경' 버튼을 클릭하면 비밀번호 변경 모드로 바뀌면서 true가 된다. 다시 비밀번호 입력모드로 돌아올 때는 false로 초기화한다.

#### 3-5-2. OnMouseDown() 함수

```
16 void OnMouseDown() {
17
18     if (Input.GetMouseButtonDown (0)) {
19
20         // 비밀번호가 입력모드 일 때 Button13(비밀번호변경버튼) 이 클릭되면
21         if (b_changePw == false)
22         {
23             // "abcd" 전송, b_changePw 변경
24             BtConnector.sendString ("abcd");
25             b_changePw = true;
26         }
27         // 비밀번호가 변경모드 일 때 Button13(비밀번호변경버튼) 이 클릭되면
28         else
29         {
30             // "bcde" 전송, b_changePw 변경
31             BtConnector.sendString ("bcde");
32             b_changePw = false;
33         }
34
35         // 비밀번호 변경 버튼 누를 때마다 Queue, bool값과 변수 초기화
36         InputQueue.queue.Clear();
37         Button12.b_pw = false;
38         Button12.b_checkPw = false;
39         Button12.change_password = "";
40         Button12.change_password_confirm = "";
41     }
42 }
```



개발 중  
시행착오

- 사용자 편의성 증대를 위한 작업
  - 사용자의 편의성을 위해 마커를 인식해 화면에 출력된 키패드를 고정된 위치에 출력하려는 시도를 했으나 증강현실과 무관하게 앱을 실행시키자마자 키패드가 출력되는 문제가 발생했다. 증강현실을 위해서는 마커에 키패드가 종속되어있어야 하는데, 마커랑 관계없는 독립된 위치에 키패드를 두려고 했기 때문에 발생한 문제였다. 이와 같은 문제점을 해결하면서 사용자 편의성 또한 놓치지 않도록 하기위해 한 번 마커를 인식해 키패드를 화면에 출력한 이후로는 마커가 카메라 상에서 사라지더라도 디스플레이 상에 남아있도록 하는 방법을 채택했다. 채택한 방법으로 개발을 하는 과정에서 [그림 7]의 OnTrackingLost() 함수를 없애 완전히 무시하는 식으로 구현했다. 이 때도 실행하자마자 마커의 유무에 관계없이 키패드가 화면에 출력되는 문제가 발생했다. 이는 2.3.2에서 설명한 바와 같이 처음 앱을 실행시켰을 때에는 마커가 없는 것으로 간주하여 OnTrackingLost()가 적어도 한 번 실행되는데, 이 함수를 완전히 무시했기 때문에 발생하는 문제였다. 따라서 [그림 4]와 [그림 7]의 노란색 상자와 같이 자체 개발 코드를 삽입하여 최종적으로 문제를 해결하였다.
- 보안성 증대를 위한 암호화 작업
  - 과제 수행 초기에는 256bit AES 암호화를 사용하여 비밀번호를 안전하게 전송하도록 구현하는 것을 목표로 잡고, 개발의 마지막 단계로 암호화/복호화 구현을 계획했다. C#의 Aes 클래스를 활용하여 암호화를 구현하는 것까지는 어렵지 않게 구현할 수 있었다. 문제는 아두이노와 공통된 암호화/복호화 과정이 아니라는 점이였다. 아두이노의 AES 암호화 라이브러리가 최적화를 위해 부분적으로 어셈블리어로 작성되어있었으며, C#의 Aes 클래스 내부 구조에 대해 파악할 시간적 여유도 부족해 정상적으로 잘 동작하는 두 암호화가 앱에서 암호화 한 데이터를 아두이노에서 복호화 했을 시에 전혀 다른 plaintext가 되는 문제를 해결할 수 없었다. 이러한 이유로 암호화를 제외하고 과제를 일단 완성하였다.

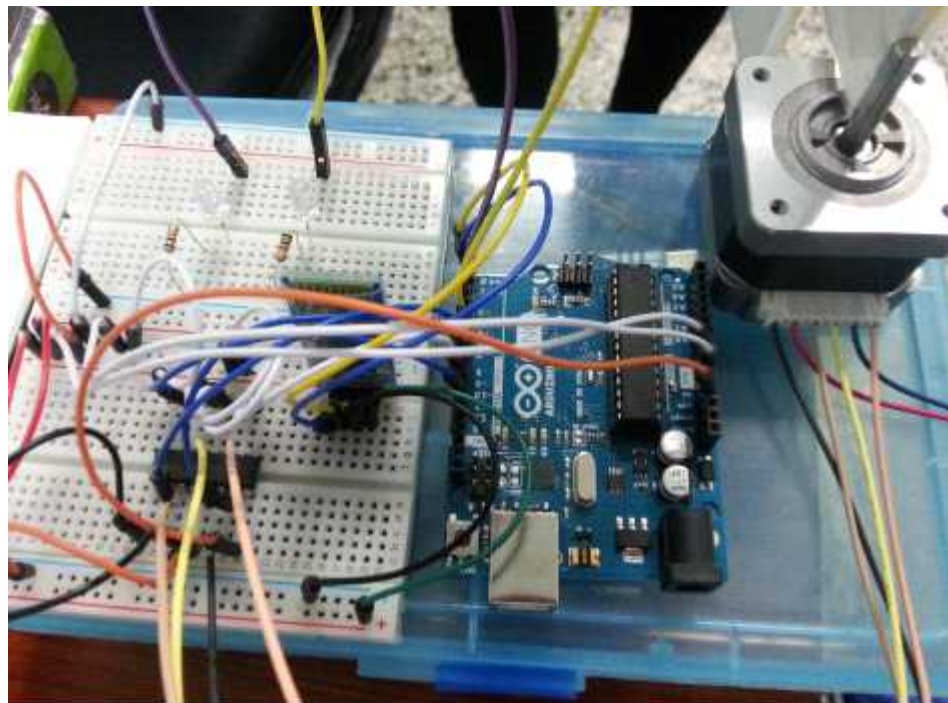
### 3. 산출물

#### 3.1 아두이노 도어락

- 아두이노 도어락

1. 비밀번호 입력 시 저장된 비밀번호와 비교하여 맞으면 오픈한다.
2. 비밀번호 변경 버튼 입력 시 파란 LED가 켜지면서 비밀번호 변경 모드로 돌입한다.
3. 비밀번호 변경모드에서 본래 비밀번호를 일치 시킨 경우 초록LED를 키고 변경할 비밀번호를 입력한다.
4. 비밀번호 변경 모드 중 언제나 비밀번호 변경 버튼을 다시 클릭하면 초기 모드로 돌아와서 비밀번호를 입력하여 문을 오픈할 수 있다.

아두이노  
도어락



- 제작된 아두이노 도어락



- 비밀번호를 입력하여 오픈이 되면서 초록불이 켜지는 모습



- 비밀번호 변경 버튼을 클릭하여 파란 LED가 켜진 모습



- 비밀번호 변경 시 현재 비밀번호 재입력에 성공하여 파란 LED와 녹색 LED가 동시에 켜져있는 모습



- 비밀번호 변경에 성공하여 LED가 꺼진 모습





◦ 변경된 비밀번호로 도어락이 오픈되는 모습

## 안드로이드 어플리케이션

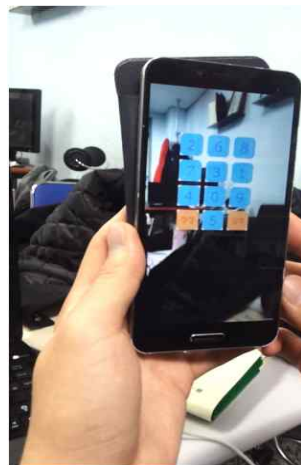
- 안드로이드 어플리케이션
  1. 어플리케이션을 실행한다.
  2. 증강현실 모듈을 초기화한다. 블루투스 사용 안함 상태일 경우 블루투스 사용 여부를 묻는 팝업을 띄운다.
  3. 키패드를 랜덤배치 시킨다.
  4. 증강현실 모듈이 동작한다.
  5. 인식이 성공하면 화면에 키패드를 출력한다.
  6. 기본으로 비밀번호 입력 모드가 된다. 비밀번호 4자리 입력 후 아두이노 도어락에 전송한다.
  7. 비밀번호 변경 버튼을 클릭하면 비밀번호 변경 모드로 바뀐다. 현재 비밀번호를 입력한 후 변경할 비밀번호를 입력하여 아두이노 도어락에 전송한다.



- 초기 어플리케이션 실행화면(카메라 화면 출력)



- 마커를 인식하여 버튼이 임의로 생성되는 모습



- 한번 인식 후에는 다른 곳을 비추어도 버튼이 유지되는 모습

## 4. 평가

### 4.1 성능

항목	목표	성능
안드로이드 앱의 증강현실 인식 및 동작 속도	사용자가 불편해 하지 않는 1s 이내	즉시 동작
블루투스 연결 형성 및 통신 속도	사용자가 불편해 하지 않는 1s 이내	즉시 동작
아두이노 도어락의 통신 및 동작 속도	사용자가 불편해 하지 않는 1s 이내	즉시 동작

### 5.1 품질

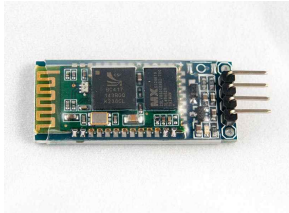

항목	평가	처리 방법
비밀번호가 4자리로만 입력이 되도록 한다.	Y	입력된 비밀번호가 4자리가 아닐 경우 queue를 비워 다시 입력 받는다.
블루투스 통신이 가능하도록 한다.	Y	블루투스 사용 설정 팝업을 띄워 프로그램 종료 없이 통신이 가능하도록 한다.



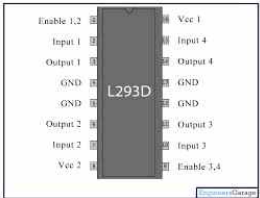
## 5. 기타사항

### 4.1 용어정리

<p>NFC</p>	<p>○ 13.56MHz의 주파수 대역을 사용하는 비접촉식 통신 기술이다. 통신거리가 짧기 때문에 상대적으로 보안이 우수하고 가격이 저렴해 주목받는 차세대 근거리 통신 기술이다. 데이터 읽기와 쓰기 기능을 모두 사용할 수 있기 때문에 기존에 RFID 사용을 위해 필요했던 동글(리더)이 필요하지 않다. 블루투스 등 기존의 근거리 통신 기술과 비슷하지만 블루투스처럼 기기 간 설정을 하지 않아도 된다.</p>
<p>아두이노</p> 	<p>○ 오픈 소스를 지향하는 마이크로 컨트롤러(micro controller)를 내장한 기기 제어용 기판. 컴퓨터 메인보드의 단순 버전으로 이 기판에 다양한 센서나 부품 등의 장치를 연결할 수 있다. 컴퓨터와 연결해 소프트웨어를 로드하면 동작을 하게 되므로 제어용 전자 장치부터 로봇과 같은 것을 만들 수 있는 '오픈소스 하드웨어'라고 할 수 있다. 자유 소프트웨어 운동에서 출발한 오픈 소스라는 개념을 하드웨어 부문까지 확산시킨 것이다.</p>
<p>블루투스</p> 	<p>○ 블루투스(Bluetooth)는 휴대폰, 노트북, 이어폰·헤드폰 등의 휴대기기를 서로 연결해 정보를 교환하는 근거리 무선 기술 표준을 뜻한다. 주로 10미터 안팎의 초단거리에서 저전력 무선 연결이 필요할 때 쓰인다. 예를 들어 블루투스 헤드셋을 사용하면 거추장스러운 케이블 없이도 주머니 속의 MP3플레이어의 음악을 들을 수 있다. 블루투스 통신기술은 1994년 휴대폰 공급업체인 에릭슨(Ericsson)이 시작한 무선 기술 연구를 바탕으로, 1998년 에릭슨, 노키아, IBM, 도시바, 인텔 등으로 구성된 '블루투스 SIG(Special Interest Group)'를 통해 본격적으로 개발됐다. 이후 블루투스 SIG 회원은 급속도로 늘어나 2010년 말 기준 전세계 회원사가 13,000여 개에 이른다.</p>
<p>증강현실</p> 	<p>○ 실세계에 3차원 가상 물체를 겹쳐 보여 주는 것으로 현실에 기반을 두고 실세계 환경과 그래픽 형태의 가상 현실을 실시간으로 합성하여 실세계에 대한 이해를 높여 주는 기술. 스포츠 중계 시 등장하는 선수가 소속된 국가의 국기나 선수의 정보를 보여 주거나 두부 장착형 디스플레이(HMD)로 사용자가 보는 실제 환경에 컴퓨터 그래픽스, 문자 등을 겹쳐 실시간으로 보여 주는 것 등이 증강 현실의 대표적인 예이다.</p>



<p>HC=06 모듈</p> 	<p>◦아두이노 보드에서 블루투스 기능을 사용하기 위해 추가적으로 필요한 하드웨어이다.</p>
<p>SoftwareSerial.h</p>	<p>◦아두이노 블루투스 포트를 사용자가 임의로 지정하여 사용할 수 있도록 한다.</p>
<p>AES</p>	<p>◦AES는 미국의 연방 표준 알고리즘으로서 20년이 넘게 사용되어 온 DES(Data Encryption Standard)를 대신할 차세대 표준 알고리즘이다. DES는 1972년에 미국 상무성 산하 NIST(National Institute of Standards and Technology)의 전신인 NBS(National Bureau of Standards)에서 컴퓨터 데이터를 보호할 목적으로 표준 알고리즘을 공모하여 IBM사가 개발한 암호 알고리즘이다. DES는 연방 표준으로 제정된 후에 5년마다 안정성을 인정받으면서 표준으로 존속되어 왔으나, 1997년 이후 안정성에 대한 논란이 대두되자 NIST가 DES를 대체할 차세대 표준 암호 알고리즘 제정을 위한 프로젝트로 추진한 것이 AES이다. AES 알고리즘이 채택되면 현재 사용되고 있는 DES를 대신하게 되고, 로열티 없이도 사용할 수 있게 된다.</p>
<p>프레임 마커</p> 	<p>◦Vuforia SDK가 사용하는 특별한 유형의 마크로 프레임 마커가 가지는 고유한 정보는 마커 이미지의 경계선을 따라 그려져 있는 이진 패턴에 담기게 된다. 프레임 마커는 어떤 이미지 또는 글씨 등을 마커의 내부에 그리거나 써넣을 수 있어 QR코드와 같은 기존의 다른 마커들보다 자연스러워 보이도록 만들 수 있다.</p>
<p>Vuforia SDK</p>	<p>◦Vuforia는 시각 기반(vision-based) 증강현실을 구현하기 위한 API를 제공하는 소프트웨어 개발 도구(Software Development Kit)이다.</p>

<p>Unity 3D pro</p> 	<p>○멀티플랫폼 3D 게임 엔진이다.. 3D 비디오 게임이나 건축 시각화, 실시간 3D 애니메이션 같은 기타 인터랙티브 콘텐츠를 제작하기 위한 통합 저작 도구이다. 초기의 개발 목적은 ‘웹에서 구동되는 3D 제작 툴’이었으나 Unity3D의 가능성에 관심을 가진 게임 개발자들이 Unity를 도입하여 게임을 개발하기 시작하면서 게임 엔진으로서의 기능들이 추가되었다.</p>
<p>스테핑 모터</p>  <p>56각</p>	<p>○전용 드라이버의 입력 Pulse 신호의 지령에 따라 모터의 상(相) 여자 전류가 제어됨으로써 결정된 기계적 각도로 회전하는 고정도, 고폭 특성 가진 하이브리드형 모터입니다. 디지털 신호(Pulse 신호)를 통한 제어가 용이하여 Computer 주변 단말기기, 반도체 제조 장비, 의료 기기, OA 기기 등 산업 현장에 폭 넓게 사용되고 있습니다.</p>
<p>L293D</p> 	<p>○ 아두이노에서 모터를 제어하는데 쓰이는 모듈</p>
<p>5.2 참고 문헌</p>	
<p>아두이노 완전정복</p>	<p>저자 : 김경연, 장정현, 박민상 / 복두출판사 / 2014</p>
<p>뷰포리아 공식 레퍼런스</p>	<p><a href="https://developer.vuforia.com/resources/api/unity/index">https://developer.vuforia.com/resources/api/unity/index</a></p>
<p>Unity 3D로 배우는 실전 게임 개발</p>	<p>저자 : 박승제 / 제이펍 / 2012</p>
<p>블루투스 API 가이드</p>	<p><a href="http://developer.android.com/guide/topics/connectivity/bluetooth.html">http://developer.android.com/guide/topics/connectivity/bluetooth.html</a></p>