**Let’s park**



201535013 김진경

201535017 박예인

201535029 이주연

201535037 최은선

**목차**

1. **프로젝트 명**
2. **프로젝트 개발 목적&목표**
3. **개발 환경**
4. **개발 일정**
5. **시스템 구성도**
6. **Server & Data base**
7. **Aduino**
8. **회로도**
9. **주요기능**
10. **Android**
11. **UI**
12. **주요기능**
13. **실제 구현판**

**10.참고 문헌&사이트**

**부록; TELL ME PAY**

1. **개요**
2. **개발 동기**
3. **개발 목적**
4. **개발 환경**
5. **프로젝트 진행**
6. **cmd**
7. **web**
8. **Dapp 구축**
9. **Metamask**
10. **Mist**
11. **미니 월렛 예제 실행**
12. **개발 계획**
13. **프로젝트 미완성에 대한 보고와 분석**
14. **참고 문헌**

**1. 프로젝트명**

**LET’S PARK 라는 이름의 아두이노 적외선 센서 관련 안드로이드 APP입니다.**

**쉽게 주차를 하자는 의미에서 LET’S PARK라는 프로젝트 명을 작명하고, 기획하였습니다.**

**2. 프로젝트 개발 목적&목표**

**매일 어머니께서는 저녁 늦게 9~10시쯤에 퇴근하십니다. 어머니께서 퇴근하시고 주차를 하시려고 할 때면 저희 아파트 밑 지하 주차장은 언제나 포화하기 직전 상태여서 주차 하기가 어려울 때가 많다고 하십니다. 그렇기 때문에 차가 비어 있는 곳이면 즉시 차를 주차하시고 집으로 올라오십니다. 그래서 어머니께서는 자신이 주차하신 구역을 잊으실 때가 많습니다.**

**매일 동생과 엄마가 아침이면 “엄마! 차 어디에 세웠는지 아세요? “, ”엄마도 몰라. 나가봐야 알겠는데?” 하는 말을 매일 들어왔습니다. 주차장이 꽤 넓은 편이라 차의 소리를 듣고 찾아가는 데에도 한계가 있었습니다. 그래서 내 차가 어느 구역에 있는지를 직접 볼 수 있으면 편리 하겠다 라는 생각을 종종 하게 되었고 이번 프로젝트에서 기획하게 되었습니다.**

**그리고 또, 저희 부모님 뿐만 아니라 다른 운전자분들도 남아있는 주차 구역의 위치를 몰라서 주차장을 계속해서 배회하는 모습을 종종 보았습니다. 그 모습을 보고 차가 어디 남아 있는지를 알면 이렇게 주차장을 배회하지 않아도 되고, 시간을 절약할 수 있을 텐데 하는 마음에 실시간으로 주차장의 차 유무를 확인할 수 있는 기능을 추가하게 되었습니다.**

**또한, 자가에서 주차하는 경우가 아닌 백화점과 같은 바깥에서는 요금을 측정하는 곳이 많습니다. 이러한 곳에서 요금이 얼마 나왔는지 모르고 낭비할 때가 있습니다. 그렇기 때문에 주차 요금 계산 기능을 추가함으로써 돈과, 시간을 절약할 수 있다 생각이 들어 기능을 추가하였습니다.**

**마지막으로, 알람 기능을 통해 시간을 설정하여 사용자가 원하는 시간에 주차 시간 만료 알람을 받음으로써 시간과 돈을 절약할 수 있게 도와줍니다. 알람은 사용자 핸드폰의 상단바 알림과 소리 알림으로 나타납니다.**

**3.개발환경**



**아두이노는 wifi 모듈이 내장되어 있는 오렌지 보드와 적외선센서를 사용하였습니다.**

**안드로이드 api27 이상 버전을 사용하였고,**

**서버와 DB는 웹 호스팅을 통해 php와 mySQL을 연동하여 사용하였고,**

**웹은 WIX를 이용해 제작하였습니다.**

**4. 개발 일정**

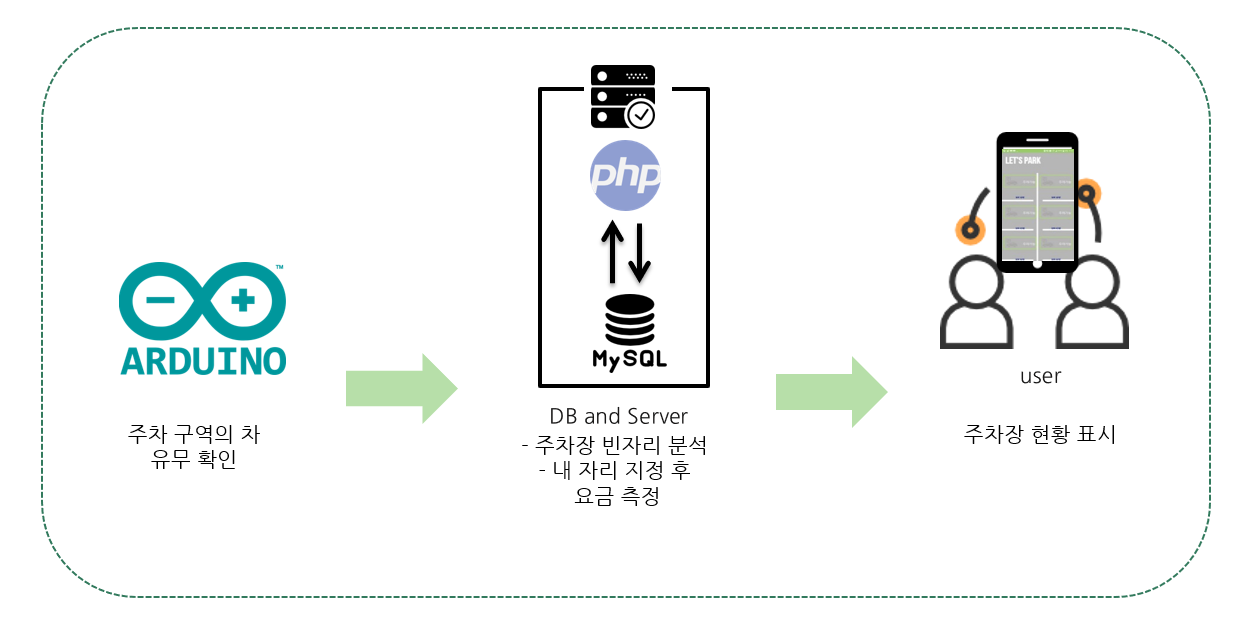
**앞에 두 달 동안은 Tell me Pay 라는 블록체인 Wallet 앱을 구현하는데, 실질적으로 기간내에 완료하지 못할 것 같아, 개발을 멈추고 5월 말부터 6월 중순까지 아두이노 관련 주차 앱을 계획하고 대략 3-4주 안으로 구현하였습니다.**

**5월 4주차에는 대략적인 앱 기획과, 스터디를 하였고, 그 다음주부터 아두이노와 안드로이드로 나눠 스터디 하였습니다.**

**6월 1주차에는 코딩 및 연결을 구현하였습니다.**

**6월 2주차에는 팀 조원 모두, 서버 구축과 데이터 베이스를 구축하였습니다.**

**6월 3주차에서는 서로 모르고, 오류 수정 난 부분을 같이 수정하였습니다.**

**5. 시스템 구성도** 

**아두이노에 연결된 적외선 초음파 센서가 data를 실시간으로 server에 전송해주고, db에는 어느 구역에 차가 주차되었는지 확인할 수 있습니다.**

**그리고 안드로이드는 서버와 연결해 그 data를 받아오는 구성을 가졌습니다.**

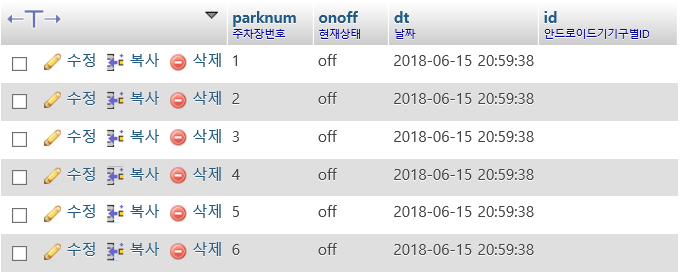
**6. Server & Data base**

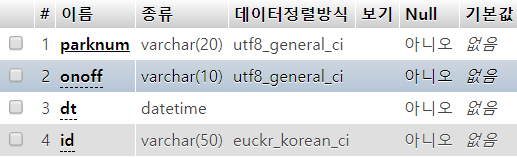
**프로젝트는 웹 호스팅으로부터 호스팅 서비스를 구매하고 PHP 파일을 생성하며, 호스팅 서비스에 MySQL을 이용하여 APP과 데이터 통신 및 파일을 저장하는 것을 목표로 하였습니다.**

**저희는 dothome에서 제공하는 무료 호스팅을 이용하여 서버를 개설하였습니다.**

**(주소;** [**www.letspark.dothome.co.kr**](http://www.letspark.dothome.co.kr)**)**

**APP에 사용될 DB를 생성하기 위해 phpMyAdmin에 접속해 DB 테이블을 생성했습니다.**



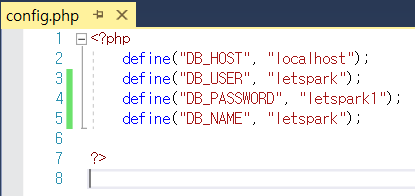


**변수는 parknum과 onoff, dt를 선언하였으며, 각각 주차장 구역 번호와 현재 주차장 구역의 차 유무를 확인하는 변수, 마지막으로 차가 감지된 시간을 나타냅니다.**

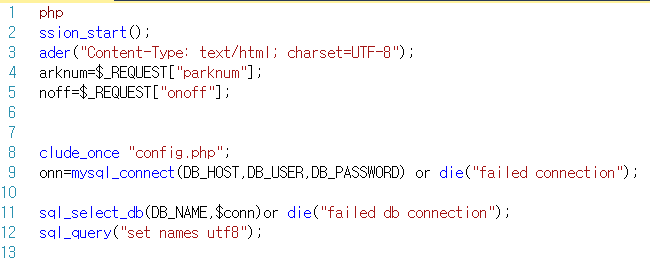
**Php 파일을 생성합니다.**

**생성한 모든 php 파일 상단에는 공통적으로 서버에 접속하는 부분이 들어가 있으므로 따로 서버 계정의 아이 아이디와 비밀번호와 이름 등과 같은 정보에 관한 변수들을 초기화한 config.php파일을 생성했습니다.**

**만약 정보와 일치하지 않을 경우 서버에 접속되지 않습니다.**



**모든 php파일의 상단에 포함되는 구문입니다. 이를 통해 서버에 접속할 수 있습니다.**



**insert.php파일입니다. 이를 통해 아두이노에 부착된 적외선 센서가 물체를 감지하면 DB에 저장된 onoff, dt변수들을 초기화합니다. parknum은 구역 번호를 의미하고 onoff는 현재 차가 있는지 없는지 판단합니다. 있으면 on 없으면 off으로 초기화됩니다.**



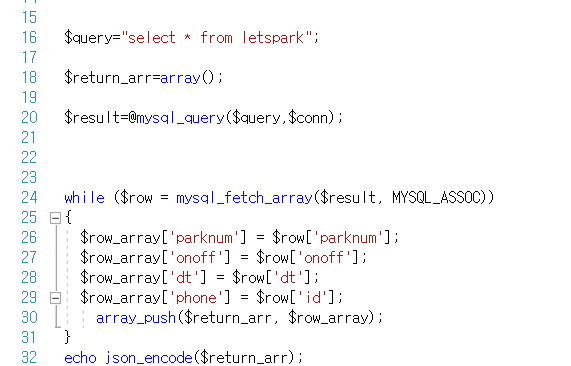
**Update.php파일입니다. 이를 통해 정보가 바뀐다면 즉시 정보를 수정합니다.**

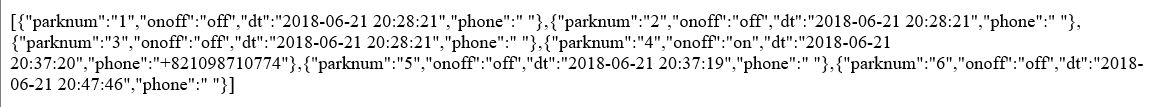


**Reset.php파일입니다. Reset url에 접속을 하면 변수들의 정보가 모두 초기화됩니다.**



**List.php파일입니다. 현재 변수들의 정보를 보여줍니다. 코드 아래 보이는 사진은 url에 접속했을 때 보이는 현재 변수들의 정보입니다.**





**마지막으로, 안드로이드와 서버를 연결하는 코드입니다(CurrentActivity.java 중). 이 코드를 통해 서버에서 만든 DB의 변수 parknum, onoff, dt를 안드로이드 어플 변수와 연동하여 사용할 수 있습니다.**



**7. Aduino**

**저희는 wifi로 핸드폰과 아두이노를 통신시킨 후 센서를 이용해 물체의 유무를 감지하도록 설계하였습니다. 우선 wifi 통신을 하기 위해 wifi 모듈이 내장에 장착되어 있는 오렌지보드를 선택하였고 물체의 유무 감지를 위한 센서는 초음파 센서와 적외선 센서를 고민하던 중 적외선 센서를 선택하였습니다.**

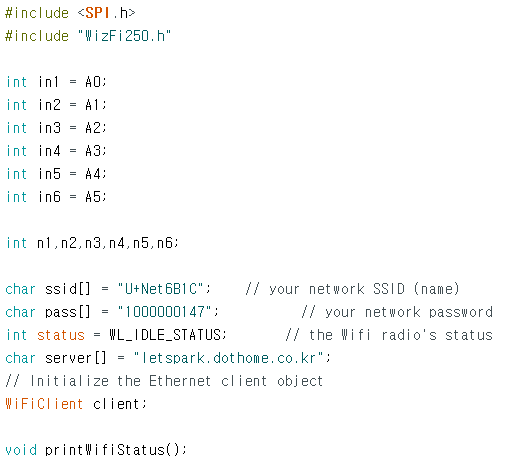
**초음파센서는 상대적으로 직진성이 좋지 않은 것 같지만 물체까지의 거리를 측정하기에 좋지만 적외선 센서는 반면에 초음파 센서보다 직진성이 좋은 것 같아 선택할 때 이러한 점에 유의를 두었습니다. 물론, 적외선 센서는 거리의 제약이 있기도 하고 주변 및 형광등 등에 노이즈를 받기도 하지만 적당한 필터를 주면 괜찮다고 생각하였습니다.**

**그래서 적외선 센서를 사용하는 방법을 선택하게 되었습니다. 비교기를 사용하여 수광부에 들어오는 빛에 따른 출력 전압이 기준전압보다 높은 것은 HIGH로 낮은 것은 LOW로 사용하는 방법을 택했습니다.**

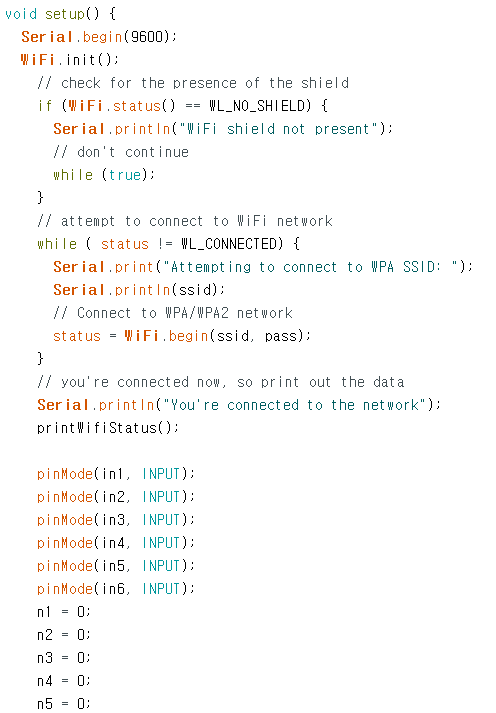


**<오렌지 보드와 초음파 센서>**

**코드와 회로도 중 우선, 코드에 대해 설명하겠습니다. 코드 사진을 넣은 뒤, 코드 사진 아래에 그 코드에 대한 부연 설명을 추가하겠습니다.**



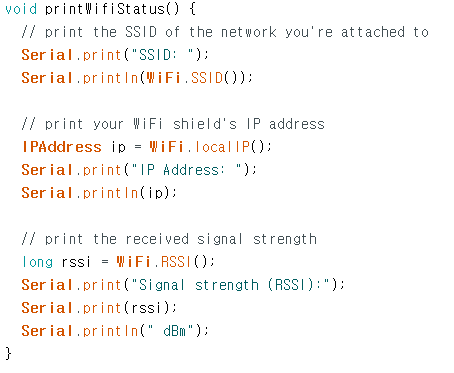
**아두이노 코드 중 상단 부분입니다. 우선 자체적으로 wifi 모듈과 spi통신을 하기 때문에 wifi 함수에서 사용을 하는 SPI를 선언하고 WizFi250를 선언하였습니다. 그 후 센서가 적용되는 핀의 자리를 선언하였습니다. 이는 총 6개의 주차 구역에서 들어왔다 나가는 물체의 유무를 판단해 주기 위함입니다. 또 그 뒤에 int형 변수 n1, n2 등을 선언한 이유는 중복을 방지하기 위함입니다. 자세한 설명은 다음 코드를 보며 덧붙이겠습니다. 그 다음 코드는 와이파이 모듈이 AP에 접속하기 이해 필요한 SSID와 패스워드를 저장하고 있는 코드입니다. 각 변수에는 자신이 접속할 AP의 SSID와 패스워드입니다.**



**다음 setup()함수입니다. 우선 wifi 모듈과 접속하기 위해 상단 부분에 해당하는 AP의 SSID와 패스워드가 코드에 나와 있는 SSID와 패스워드와 일치해야 합니다. WiFi 방어막이 없을 때 할당되는 접속 상태인 WL\_NO\_SHIELD일 때는 ‘WiFi shield not present’ 가 출력됩니다. WiFi와 접속이 되지 않을 경우는 접속 시도하고 있는 SSID를 출력합니다. 제대로 접속이 되었을 경우에는 ‘You’re connected to the network’가 출력되고 아래와 같은 printWifiStatus()함수를 통해 접속된 네트워크의 SSID를 출력합니다.**

**그 다음 적외선 센서에 해당하는 변수들을 pinmode를 통해 input 설정합니다. 저장한 핀을 INPUT으로 설정하면 해당 핀은 하이 임피던스 상태가 됩니다. 이러한 하이 임피던스 상태는 매우 작은 입력도 쉽게 감지할 수 있지만 해당 핀이 다른 회로에 연결되어 잇지 않으면 핀의 값은 무작위로 바뀌게 됩니다. 이를 보완하기 위해 잘 알려진 값으로 상태를 유지하도록 풀업/풀다운 저항을 연결한 회로를 구성할 수 있습니다.**

**그 후 중복 방지를 위한 변수 n1, n2 등은 모두 0으로 초기화합니다.**



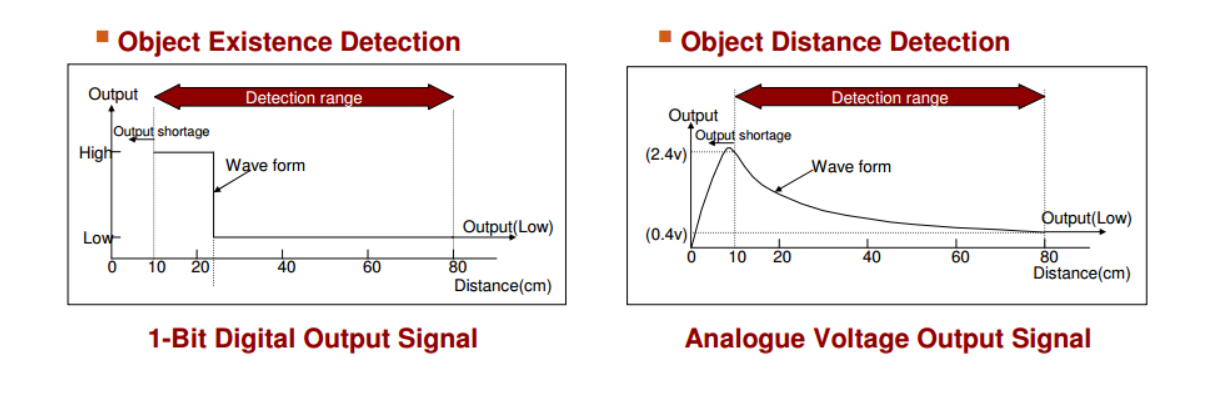
**<printWifiStatus() 함수>**





**Loop()함수입니다. 우선 중복방지 변수를 이용해 물체 유무를 판별합니다. 만약 현재 n1의 값이 0일 경우 센서에 물체가 감지되었는지를 확인하거나 감지가 되었으면 n1=1로 변경합니다. N1이 1일 경우에는 그 반대의 경우 n1은 0으로 변경됩니다. 즉, 차가 들어오고 나올 때의 순간을 감지합니다.**

**그 후 핀의 값을 읽어 현재 값이 HIGH인지 LOW인지 판별을 합니다. 적외선 센서는 적외선을 방출하는 Light emitter부분과 쏘아 올린 적외선이 다시 되돌아올 대 그 빛을 다시 감지하는 Light**

**Detector 부분으로 구성되어 있습니다. 아래 그림과 같이 아날로그 상태일 때 전압 값은 거리가 10cm이하로 떨어질 경우 오히려 전압 값이 감소하고 10cm 부근에서 최대의 전압 값을 갖다가 거리가 멀어질수록 다시 전압이 감소하는 형태를 띄고 있습니다.** 

**<**디지털 상태와 아날로그 상태일때의 거리에 따른 전압값의 변화**>**

**만약 제대로 물체를 감지해 센서가 LOW 값으로 변하게 된다면 해당 센서가 위치한 구역에 있는 onoff 변수가 서버를 통해 on으로 초기화 되고 해당 구역에 물체가 있다고 출력을 해줍니다. 다른 해당 변수도 이와 코드가 동일합니다.**

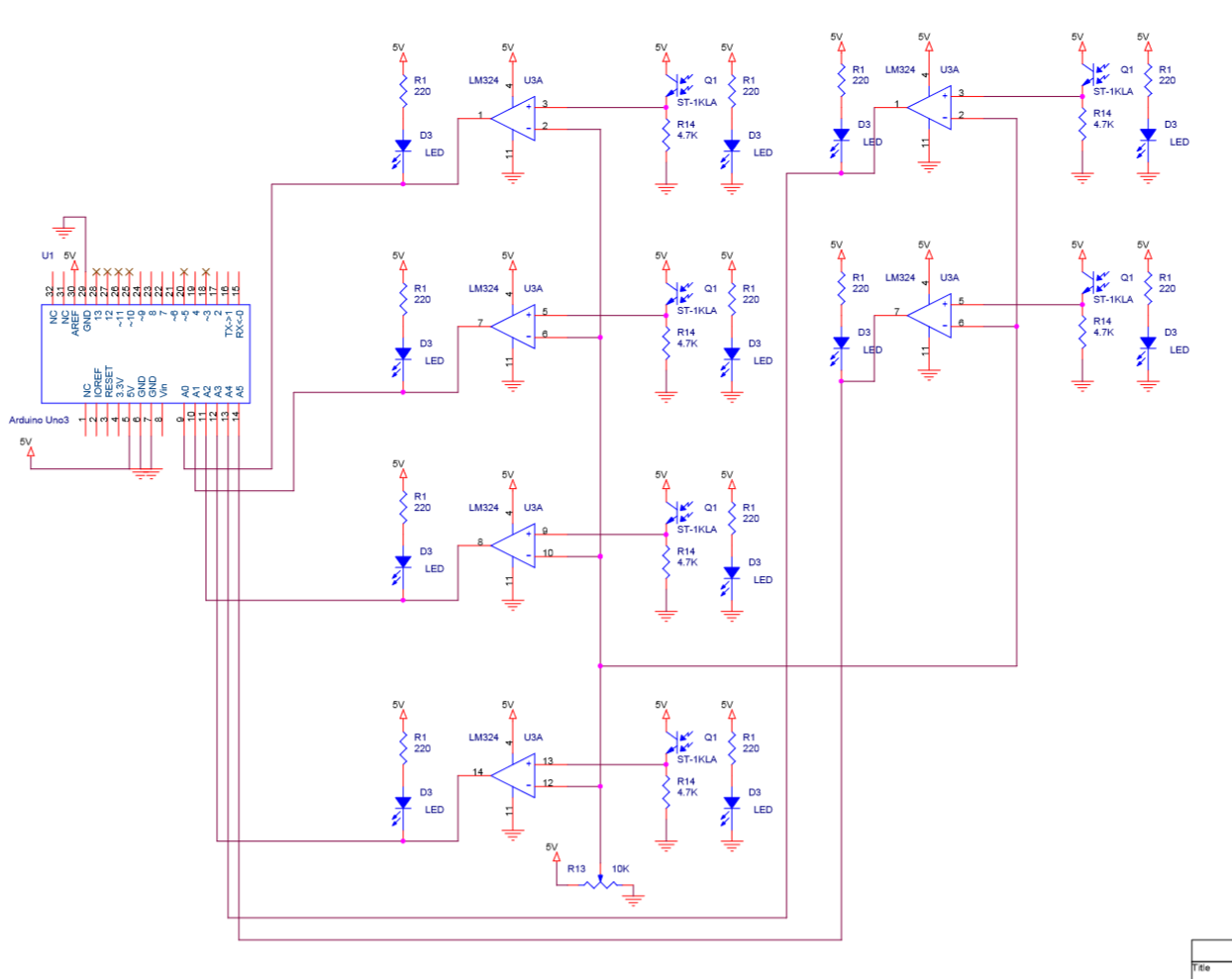
**또 loop()함수는 지연시간을 100ms으로 설정해 두었으며 이러한 일정시간후마다 반복하여 loop()함수가 다시 반복되어 실행됩니다. 그래서 서버에 그때그때 정보를 전달해줍니다.**

**마지막으로 delay()구문 아래에 있는 코드는 만약 통신이 끊겼을 때를 위해 확인 용도로 사용합니다.**

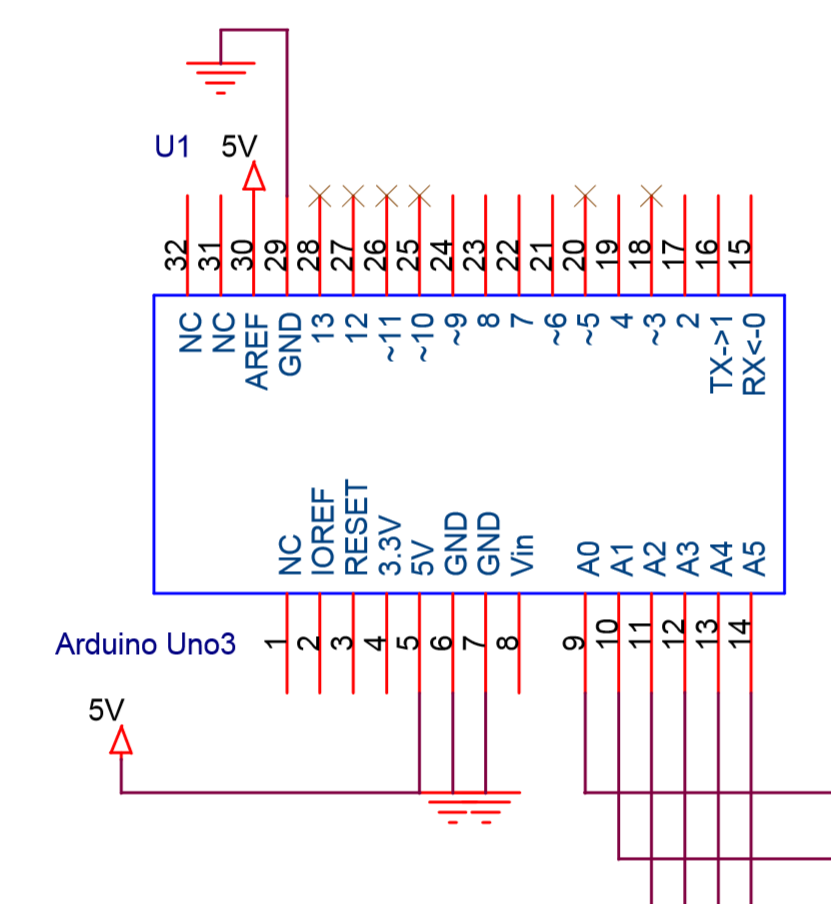
**마지막으로 아두이노 설계에 필요한 회로도의 전체 모습입니다.**

**보드에 부착한 적외선 센서 핀은 아두이노 코드와 동일합니다.**

**적외선 센서가 제대로 감지되었는지를 판별하기 위해 아두이노 자체에 적외선 led를 설치했습니다. 소자에서 HIGH에서 LOW로 떨어지면 그것을 아두이노에서 감지하고 그 신호로 인하여 LED가 켜지도록 하였습니다.**



**<전체 회로도>**



**<아두이노 핀>**

**8. Android**

(1) UI

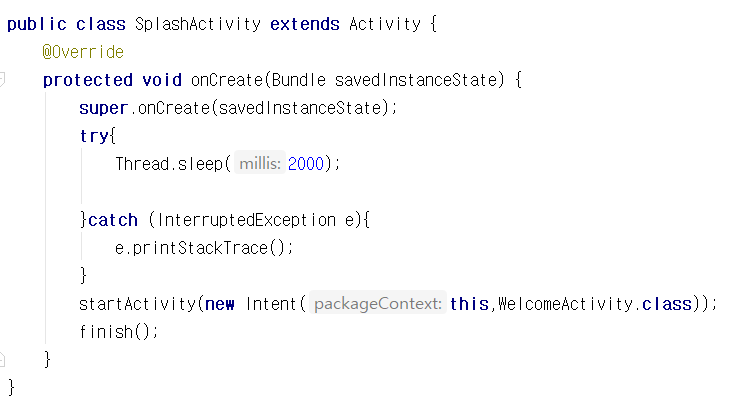
**<SplashActivity(로딩화면)>**



**SplashActivity란 로딩화면으로 MainActivity로 넘어가기 전에 보여지는 화면을 말합니다.**

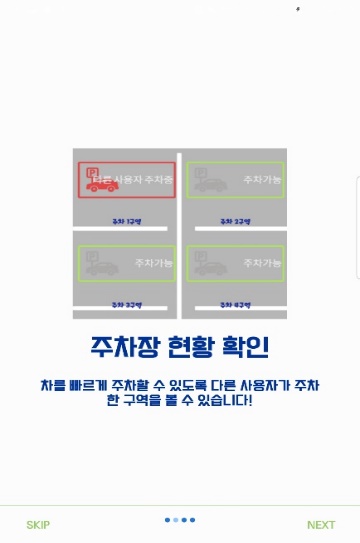


**기존 AndroidManifest.xml를 수정하여 MainActivity에 있던 intent-filter를 SplashActivity로 변경합니다. theme를 values 폴더 안에 있는 styles.xml에서 새로운 스타일을 하나 추가하여 넣어줍니다.**



**SplashActivity에 위 코딩을 해주어 2초동안 진행되게 하고 Intent로 다음 WelcomeActivity가 실행되도록 하였습니다. (사용설명Activity)**

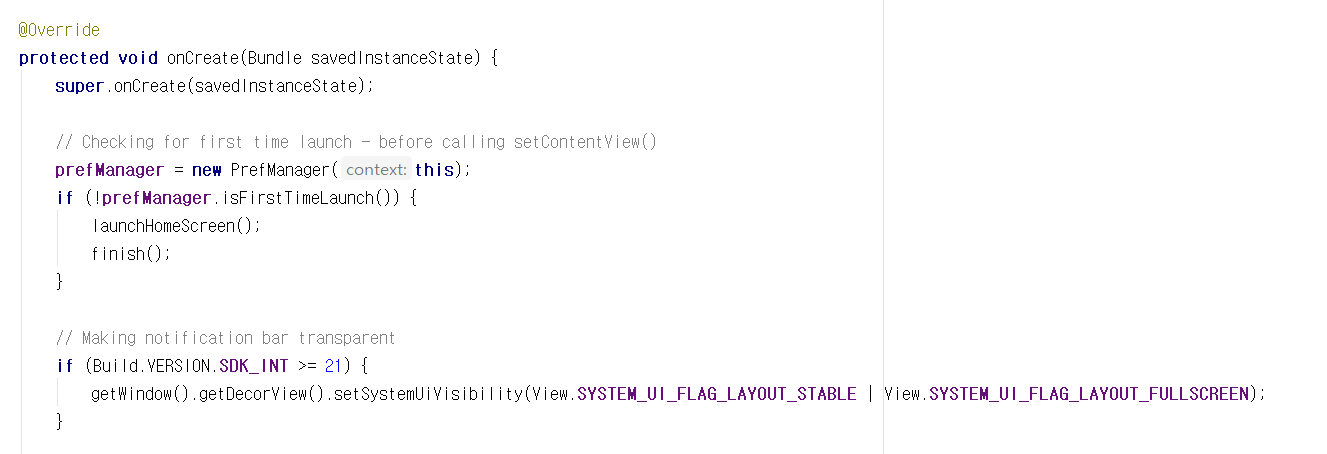
**<사용설명>**

**어플을 처음 실행했을 때에만 사용설명을 간략하게 보여주므로, 사용자들이 이용하기 편리하게 하였고 MainActivity에도 따로 사용설명버튼을 넣어 언제든지 사용자가 볼 수 있게 구현하였습니다. 또한 좌우 Swipe를 통해 페이지 전환이 가능하게 UI를 제작하여 쉽게 넘길 수 있게 하였습니다.**



**WelcomeActivity.xml 에서 ViewPager(Fragment)를 사용하여 좌우 Swipe를 통해 페이지 전환이 가능합니다. 총 네 개의 화면이 들어가므로 WelcomeActivity에 들어가는 layout를 만들어줍니다. (welcome\_slide1.xml , welcome\_slide2.xml, welcome\_slide3.xml, welcome\_slide4.xml)**



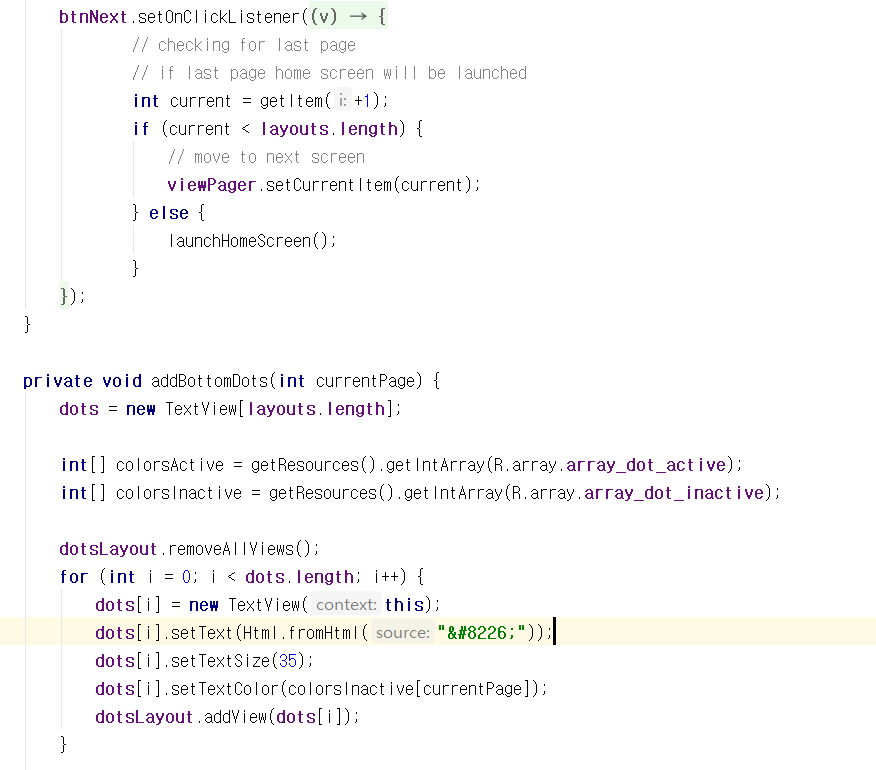
**<Welcome Actvity에서 IsFirstTimeLaunch를 통해 앱을 처음 실행하는지를 체크합니다.**

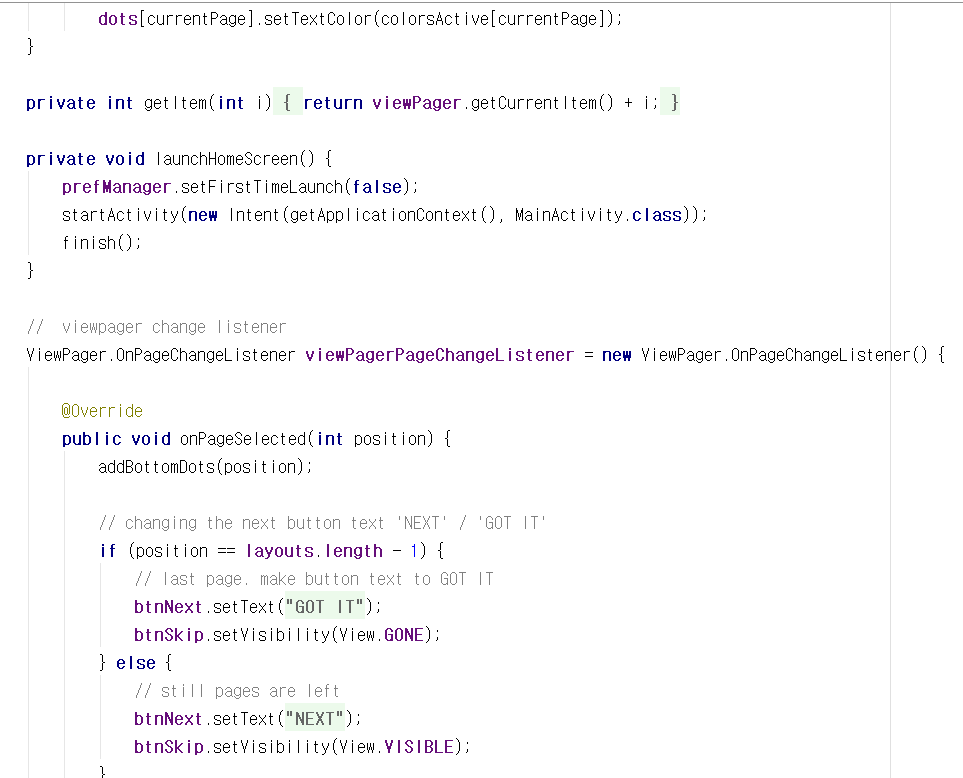
* **이로인해 어플이 처음 실행될때만 Welcome Activity가 실행됩니다.**

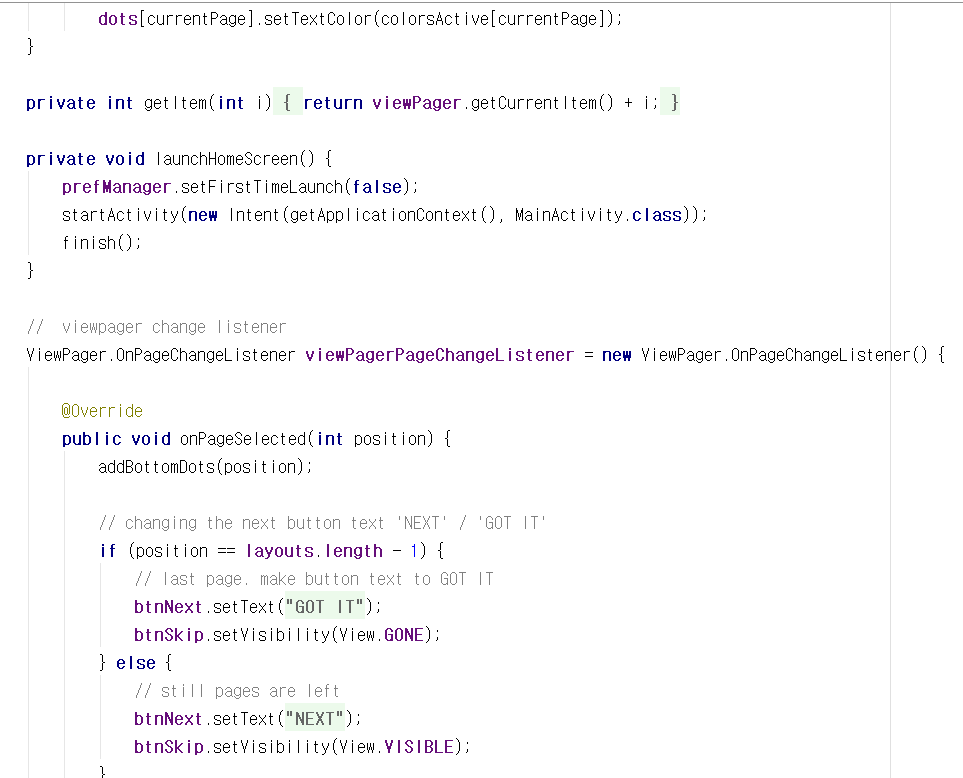


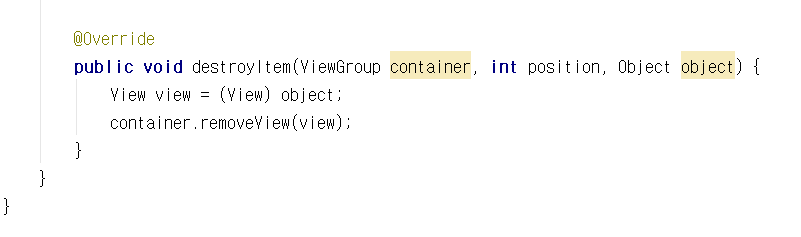
**Layouts로 WelcomeActivity에 넣을 layout들을 배열로 저장합니다.**

**viewPager.addOnPageChangeListener(viewPagerPageChageListener)를 사용하여 페이지가 변경될때 viewPagerPageChangeListener가 실행되게 합니다.**



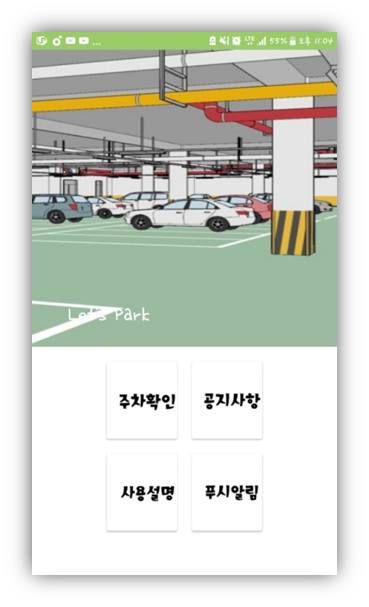






(2)주요기능

**<메인화면>**



LET’S PARK의 메인 화면입니다. 주차확인, 공지사항, 사용설명, 푸시 알람의 버튼이 있습니다.

**<주차확인>**



**주요기능인 주차확인 레이아웃입니다.**

**레이아웃 화면을 UI를 실제 구현 판과 동일한 모습으로 포토샵을 하여 적용하였습니다.**

**CurrentActivity를 두어 Permisson 정보를 가져와 허가되었을 시, 서버를 통해 주차장내에 접속하고 있는 data를 json 형태로 전송을 받아, 화면으로 뿌려줍니다.**

**그리고 setScreen이라는 함수를 두어 주차장 구역, 유무, data, 사용자의 핸드폰 번호 정보를 불러오고 저장해줍니다.**

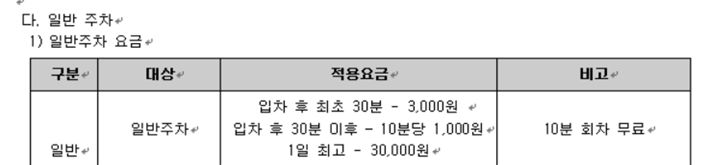


**왼쪽 빨간 아이콘은 다른 사용자가 주차 중일 때를 표시하는 아이콘입니다. 그리고 3번쨰 있는 아이콘은 내 차를 지정했을 때 나타나는 아이콘입니다.**

**자세히 보시면 등록된 날짜, 시간, 요금계산이 되어있는 것을 보실 수 있습니다.**

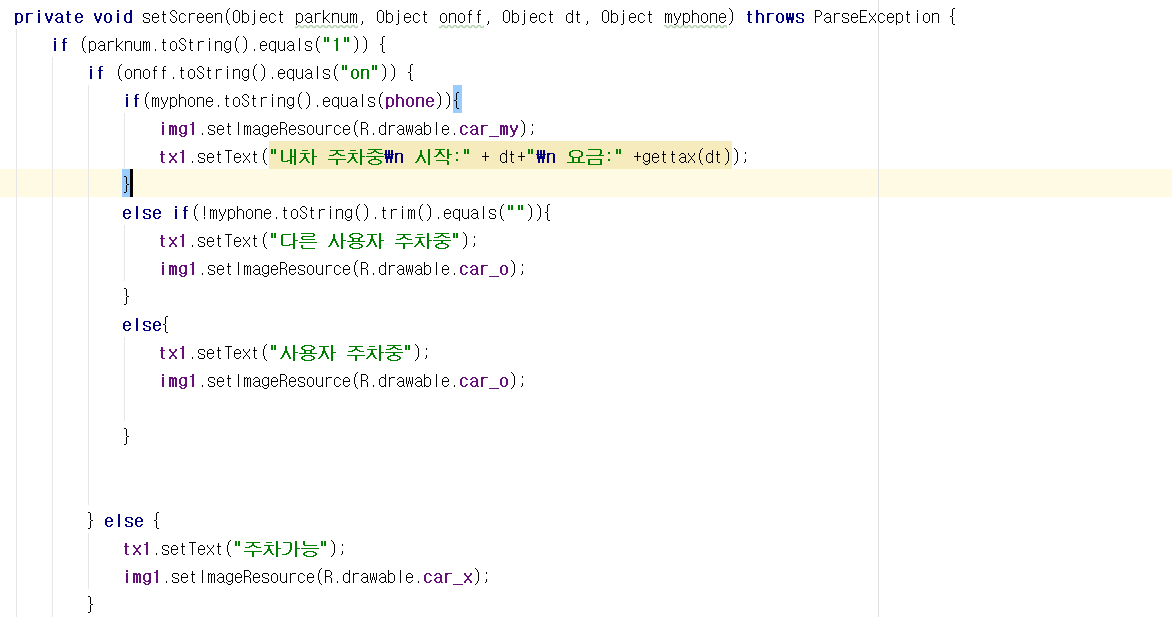
**그리고 날짜, 시간 형태를 표시 해주기 위해 SimpleDateFormat객체를 사용해주어 날짜, 시간 형태를 표현해 주었습니다.**

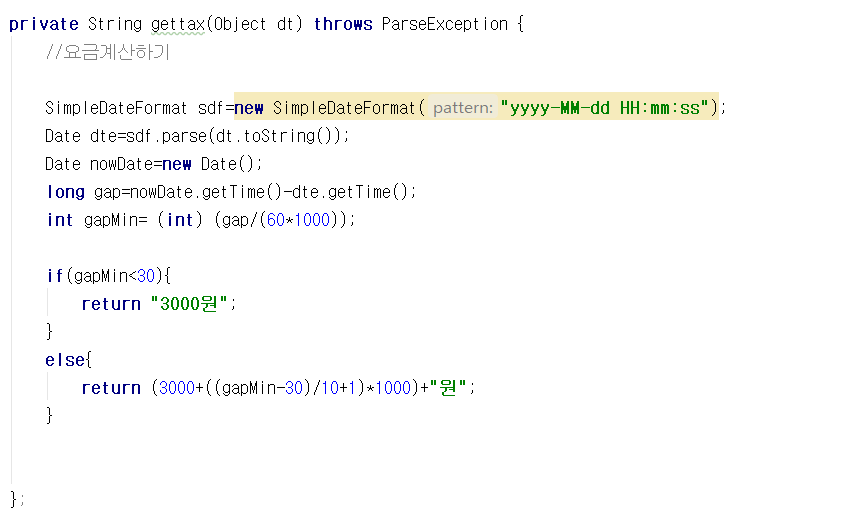
**밑엔 있는 사진들은 실제 성공회대 주차요금표와 그것을 계산한 코드입니다.**



**<실제 성공회대 주차요금표>**

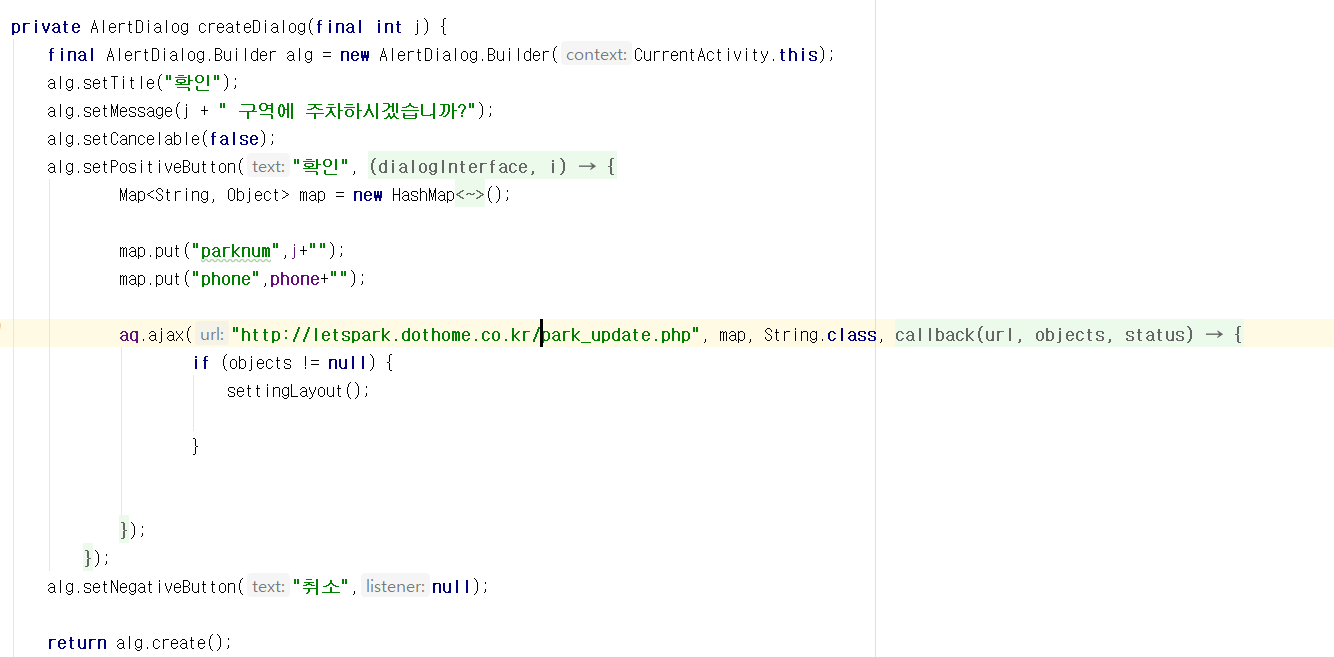
**안드로이드 스튜디오에서 서버의 정보를 받아 주차요금과 주차유무 받기**





**위 gettax()는 요금 계산하는 함수로 위 함수가 호출될 때에 해당 시간을 받은 뒤 현재시간에**

**해당시간을 뺀 뒤 요금을 계산하여 return합니다. (요금 계산은 성공회대학교 주차요금에 준함)**



**위 createDialog는 내 차 지정을 위해 필요한 함수로 내차가 주차된 구역에 터치를 하면 해당 구역에 주차하시겠습니까 라는 메시지박스가 나오고 확인버튼을 누르면 map에 해당 주차구역의 숫자와 휴대폰번호를 넣고 서버(Update)에 넣어 저장하는 것을 볼 수 있습니다.**

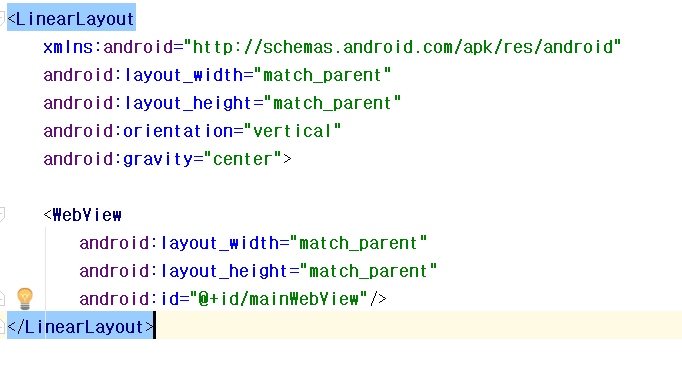
**<공지사항>**



**Wix로 제작한 웹페이지를 연결하여 공지사항을 볼 수 있게 하였습니다. 회원은 글을 쓸 수 있는 관리자와 글을 보거나 댓글을 달고 좋아요를 누를 수 있는 일반 회원으로 총 2종류로 나누었습니다. 관리자의 권한은 관리자가 되고 싶은 일반 회원에게도 부여할 수 있는데, 이 부여권은 웹을 만든 최종 관리자인 저희 팀에서 관리를 합니다.**



**실제 WIX로 제작된 Let’s Park 홈페이지입니다. (**[**https://pei5171.wixsite.com/letspark**](https://pei5171.wixsite.com/letspark)**)**



**activity\_notice.xml은 WebView를 width,height 모두 match\_parent로 하여 휴대폰에 Webview가 나타나도록 하였습니다.**

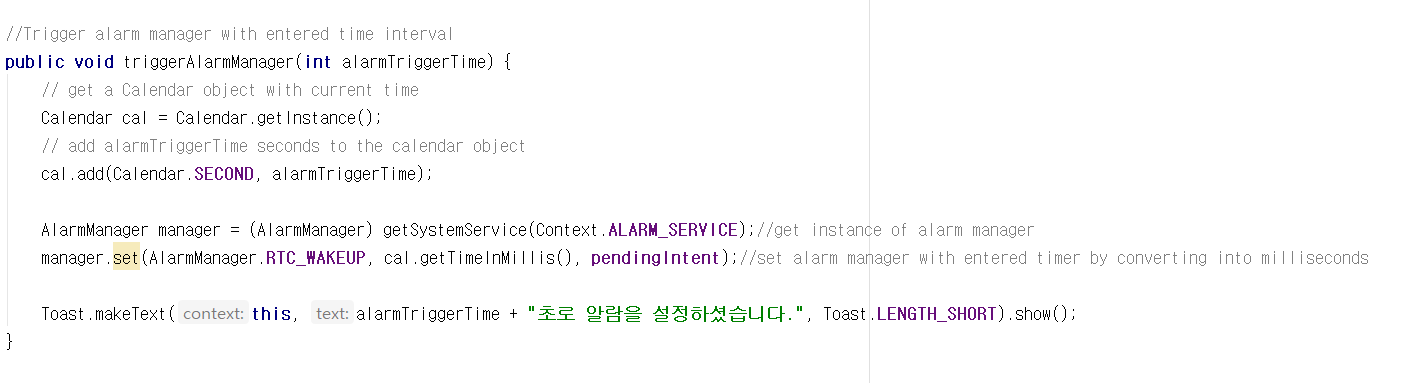


**WebView를 loadUrl로 let’s Park페이지 URL과 연결한 뒤 사이즈를 조정하고 WebViewClient에 상속을 받아 loadUrl을 호출면서 Android webview가 브라우저를 시작하게 합니다. (홈페이지로 들어가게 됩니다.)**

**<푸시 알람>**

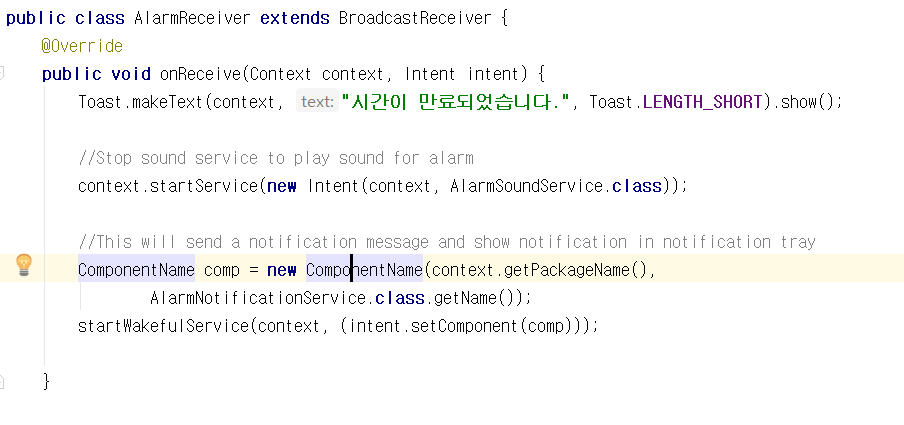


**푸시알람기능을 이용해 사용자는 시간을 설정하여 알람을 설정할 수 있습니다. 해당 시간이 되면 푸시알람과 함께 알림음이 울리며 사용자는 이 기능을 이용하여 주차 시간을 조절할 수 있습니다.**



**알람설정부분**

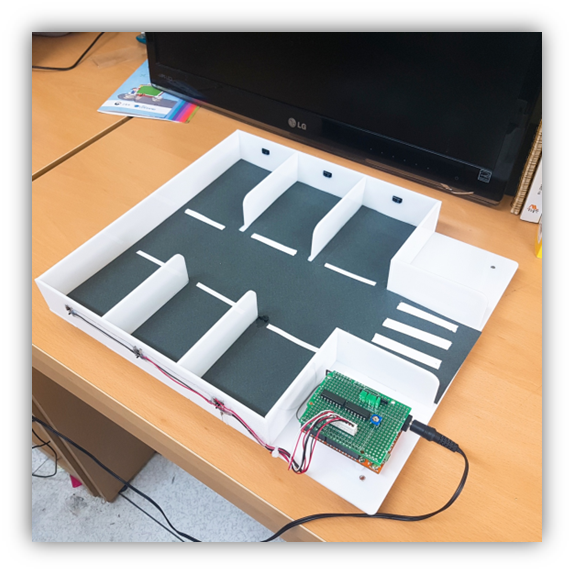
**알람 매니저의 instance를 얻은 뒤 milliseconds 단위로 변환하여 입력된 시간으로 알람 관리자를 설정합니다. 그리고 Toast메시지를 띄어 사용자에게 알람을 설정했다는 것을 확인시켜줍니다.**



**시간이 만료되었을 때 알람음 실행하는부분**

**AlarmReceiver는 BroadcastReceiver에게 상속을 받아 알람음을 울리게 됩니다. 방송이 수신되면 onReceive가 호출되어 알람음을 실행하게됩니다.**

**9. 완성된 구현 판**



**주차장에 차가 들어오면 아두이노 판에 연결된 구역의 전등이 들어오고 적외선 센서는 차를 감지해 서버에 data를 저장합니다.**

**그리고 안드로이드 내에서도 data로 받아 어느 구역에 주차되어 있는지, 다른 사람이 차량으로 등록하였는지, 요금이 얼마인지를 계산해줍니다.**

**10. 참고 문헌&사이트**

쉽게 따라하는 아두이노 배우기 -고세관 저 ,정보문화사

100% 실습! 리얼 아두이노 -정길도,류지형 외 2명 저, 성안당

아두이노와 안드로이드로 45개 프로젝트 만들기 -서민우 저,앤써북

안드로이드 프로그래밍 -천인국, 생능출판

https://lx5475.github.io/2017/07/15/android-splash/

http://coding-factory.tistory.com/50

http://itpangpang.xyz/284

<http://jeongchul.tistory.com/m/332> - 웹 호스팅을 통해 서버 제작

<http://makand.tistory.com/entry/PHP-Mysql-ConnectPHP-%EB%8D%B0%EC%9D%B4%ED%84%B0%EB%B2%A0%EC%9D%B4%EC%8A%A4-%EC%97%B0%EB%8F%99> – php 파일 생성

<https://m.cafe.naver.com/gameppt/135038?searchref=AxbZ4N1g7WMK41WfvNFcON%252FGGbskfyGvDn7WZmPulpY%253D> -ftp 프로그램 사용 방법

<https://kocoafab.cc/tutorial/view/71>

https://kocoafab.cc/tutorial/view/272 - 적외선 센서를 통한 거리 감지

<https://www.arduino.cc/en/Reference/WiFiStatus>

http://blog.naver.com/PostView.nhn?blogId=yuyyulee&logNo=220515735024 - wifi 모듈 사용

**부록; TELL ME PAY**

1. **개요**

**프로젝트 이름: Tell me pay**

**블록체인을 이용한 암호화폐를 다루는 월렛 어플리케이션**

1. 개발 동기

**높은 수준의 보안이 요구되고 지속적으로 블록체인에 대한 관심이 이어지면서 암호화폐 거래소를 노린 해킹도 끊이지 않고 있습니다. 대규모의 거래소들 또한 해커의 공격을 받아 엄청난 재산 피해를 입었고, 국내 뿐만 아니라 해외 거래소에서도 해킹으로 인해 수천 억 원 규모의 무단 인출이 발생했습니다.**

**해커는 다양한 방법으로 거래소와 이용자의 자산을 훔치려고 합니다. 이로 인해 암호화폐 이용자들은 거래하면서 해킹의 불안에 시달리고 거래소 폐장의 위기까지 겪게 됩니다. 해커들이 집중적으로 노리는 것은 핫 월렛(Hot Wallet)입니다. 이는 온라인 네트워크에 있는 저장소입니다. 또한 개인키(비밀키)를 온라인에 연결하여 화폐를 주거나 받고 살 수 있습니다. 대부분의 사람들이 핫 월렛을 이용하고 실시간으로 편리하게 사용할 수 있다는 장점이 있습니다.**

**이와 더불어 지인의 말을 빌려 시각 장애인들을 위한 인터넷 뱅킹이 없어 항상 누군가의 도움을 받거나 직접 은행에 가서 업무를 보아야만 한다는 정보를 접했습니다. 시각 장애인들은 핸드폰을 혼자 이용할 수 있지만 인터넷 뱅킹을 할 때 OTP 또는 보안카드들은 사용이 어렵다고 하였고 이 때문에 아예 은행 어플 자체를 설치하지 않는다고 하였습니다. 저희는 이런 불편을 블록체인과 함께 다루어 “음성인식을 지원하는 인터넷 뱅킹 관련 어플리케이션을 만들면 어떨까?”라는 생각을 하게 되었습니다.**

**따라서 저희 팀은 블록체인의 본질을 알고 어떻게 가상화폐가 거래되는지 이해하기 위해 핫 월렛인 ‘Tell me pay’를 개발하고자 하였습니다.**

1. 개발 목적

**“시각 장애인들을 위한 인터넷 뱅킹 어플을 개발해 보면 어떨까?” + “요즘 뜨거운 이슈인 블록체인을 이용한 가상화폐의 본질을 알고 어떻게 화폐의 송수신이 이루어질까?” 이 두가지를 함께 해결해 보고자 함이 목표였습니다. 음성인식을 통해 말을 함으로써 가상화폐의 송/수신이 보다 편리하게 이루어질 수 있게끔 하고자 하였습니다. 또한 음성인식만으로는 불편함을 느낄 수 있어 OpenCV를 이용한 사물인식을 통하여 QR코드로 된 개인키(private key)를 사용할 수 있게끔 하고 또는 지문인식을 통해 높은 보안 수준을 유지할 수 있게끔 하였습니다.**

1. **개발 환경**



**안드로이드 스튜디오를 사용하여 안드로이드 디바이스들을 위한 어플리케이션을 개발합니다.**



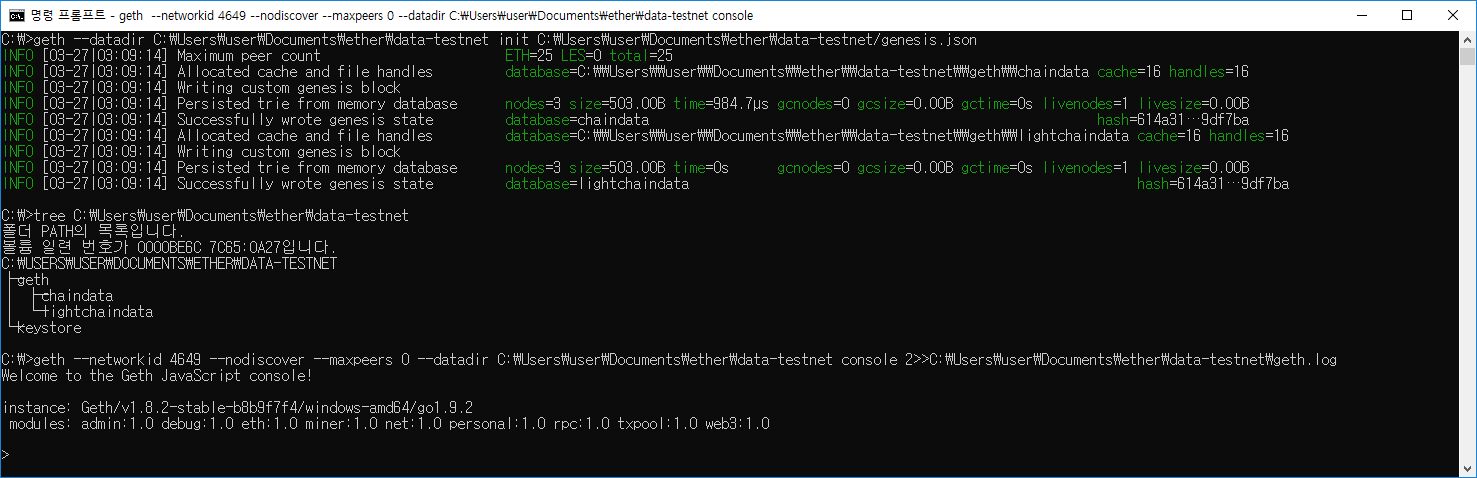
**이더리움 엔진은 Go 언어, C++, 파이썬 등으로 개발되었고 이 프로젝트는 Go-ethereum을 사용합니다. 이더리움은 블록체인을 활용한 모든 것을 프로그래밍할 수 있는 플랫폼입니다. 거래 기록뿐만 아니라 탈중앙화 앱(DAPP)들을 투명하게 운영할 수 있는 확장성을 제공합니다. 프로젝트 초반에 Windows에서 작동하는 Go-ethereum을 통하여 사설 네트워크를 구축하고 계정 생성 및 송금 등의 기능을 확인합니다.**

1. **프로젝트 진행**

**프로젝트는 web에서 먼저 개발을 한 이후에 구축된 web을 안드로이드와 연결하여 최종적으로 어플리케이션으로 개발하는 것이 최종 단계입니다.**

1. cmd

**사설 네트워크 구축을 위해서는 먼저 Geth(go-ethereum)의 설치가 필요한데 저희 팀은 Windows 환경에서 작업하기 때문에 Windows Geth 1.8.2 Version을 설치하였습니다.**

**Cmd 창에서 “ geth --datadir \\각자디렉터리\\ init \\제네시스파일경로\\ ”**

**을 입력하게 되면 geth를 초기화 하는 것이다. 실행 후에는 디렉터리에 새로운 파일들이 생성되고 정상적으로 init 되었음을 알 수 있다. (단, 처럼 초기화하기 위해서는 genesis.json 파일이 해당 경로에 위치하고 있어야 한다.)**

**이렇게 개인이 쓸 수 있는 사설 네트워크를 구축하고 나면 해당 네트워크 내에서 채굴을 하고 계정으로 송금할 수 있다.**

**>personal.newAccount(“pass0”) //pass0을 가진 새로운 계정을 만들고 개인키를 받는다.**

**>eth.getBalance(eth.accounts[0]) //accounts[0]의 보유 wei를 보여준다**

**>miner.start() //채굴을 시작한다 (정지는 start ()->stop() 으로 바꾸어주면 된다.)**

 **accounts[0]에서 채굴한 양**

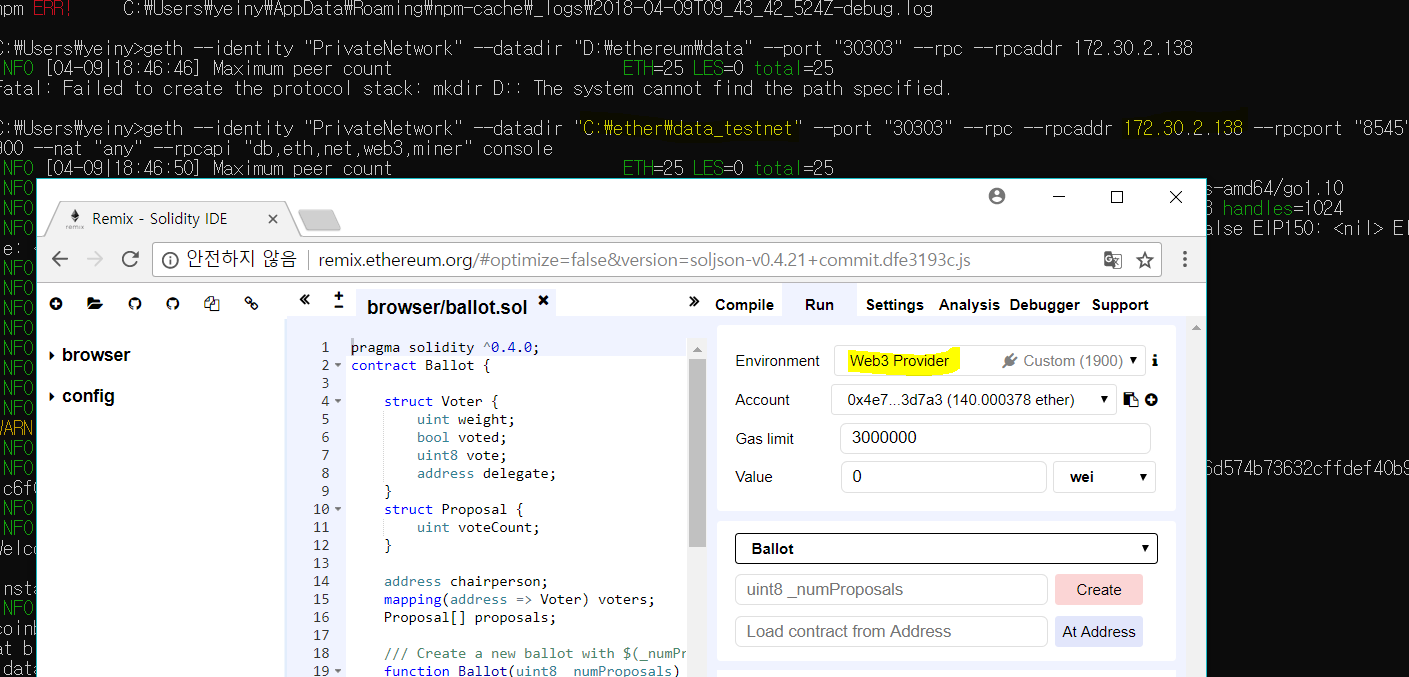
 **accounts[1]로 송금한 후 (계속 마이닝 중)**

 **accounts[1]의 보유 wei**

 **accounts[0]으로 부터 받은 후**

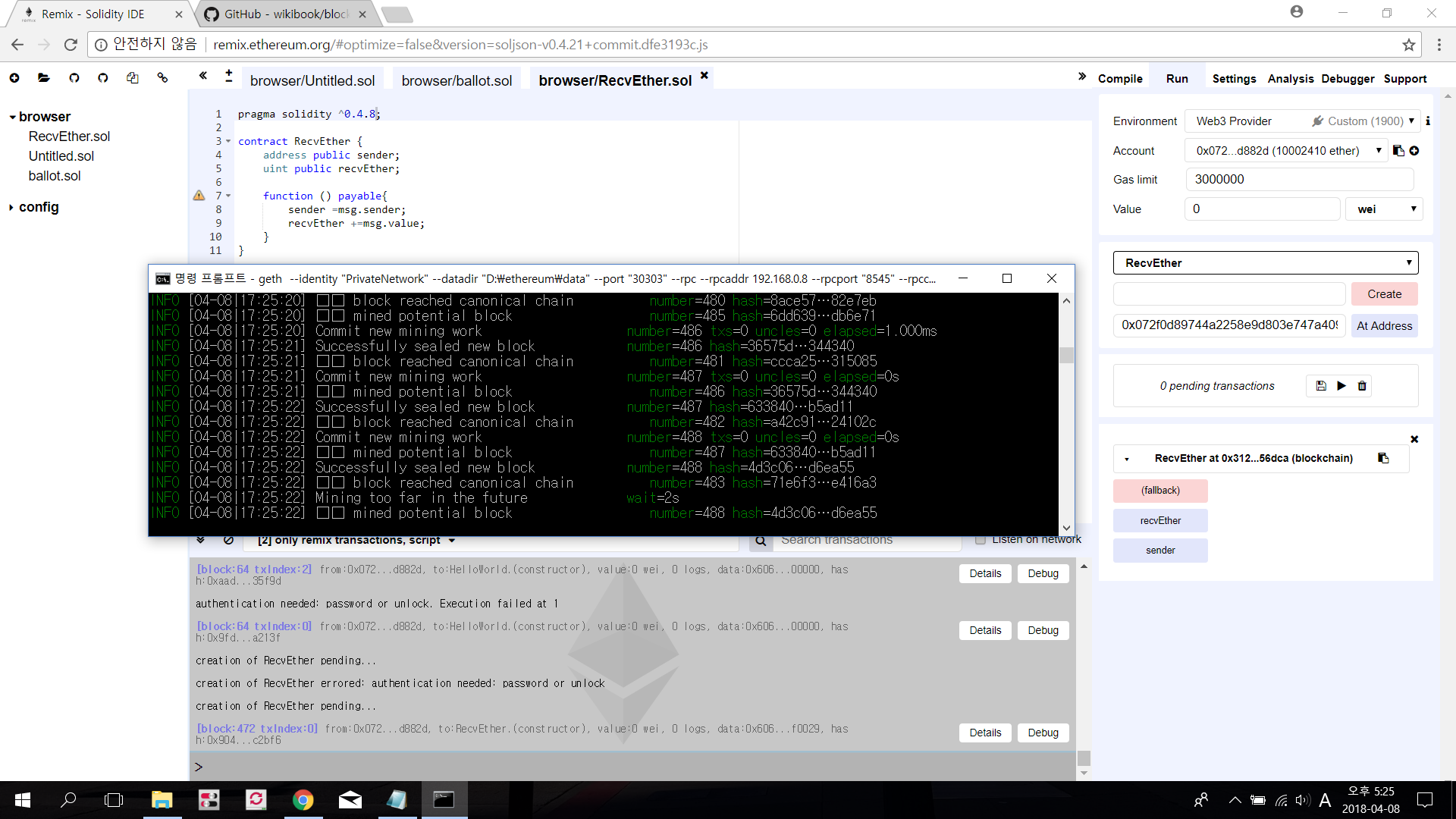
1. Web

**Remix IDE (**[**https://remix.ethereum.org**](https://remix.ethereum.org/)**)를 이용하여 Solidity 프로그래밍을 시작합니다.**



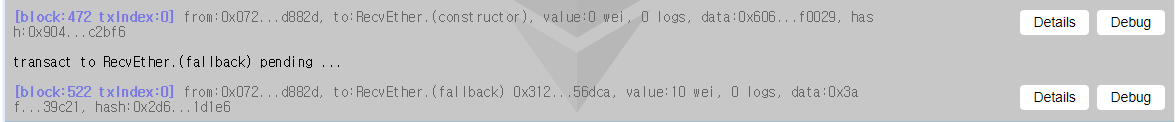
**Web provider3 환경에서 작업하였습니다. Compile 탭의 “Start Compile”버튼이 있는데 이를 누르면 작성한 코드를 컴파일하여 컨트랙트를 생성합니다.**

**컴파일이 완료된 이후에는 Run Tab에서 작성한 코드의 이름을 선택한 후에 create를 선택하면 호출 가능한 함수들이 뜨게 됩니다. (Create버튼을 누르면 unlock을 요구하는 메시지가 뜨게 됩니다. 잠금을 푼 이후에 해당 계좌의 계약 정보가 표시되면 성공입니다.) 이를 이용하여 cmd 창과 비교하면 같은 결과가 cmd 창과 Remix IDE에서 동일하다는 것을 알 수 있습니다.**

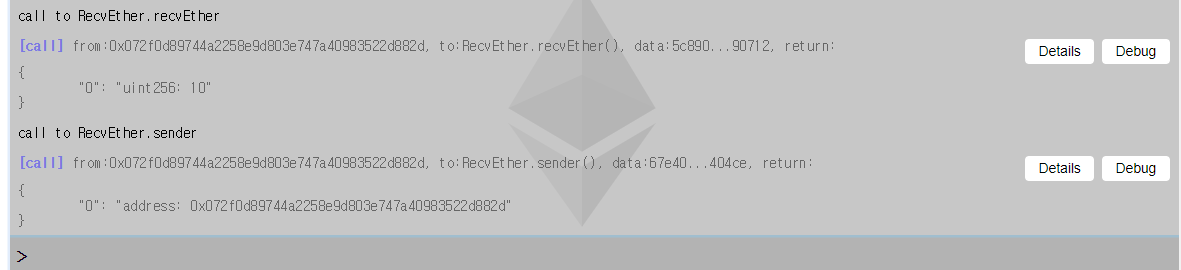


**RecvEther를 이용하여 계약으로 송금하는 모습입니다.**

**Address public sender은 보내는 주소 확인용 변수이고 unit public recvEhter은 받은 total ether, sender = msg.sender은 확인을 위해 상태 변수를 갱신한 것입니다.**



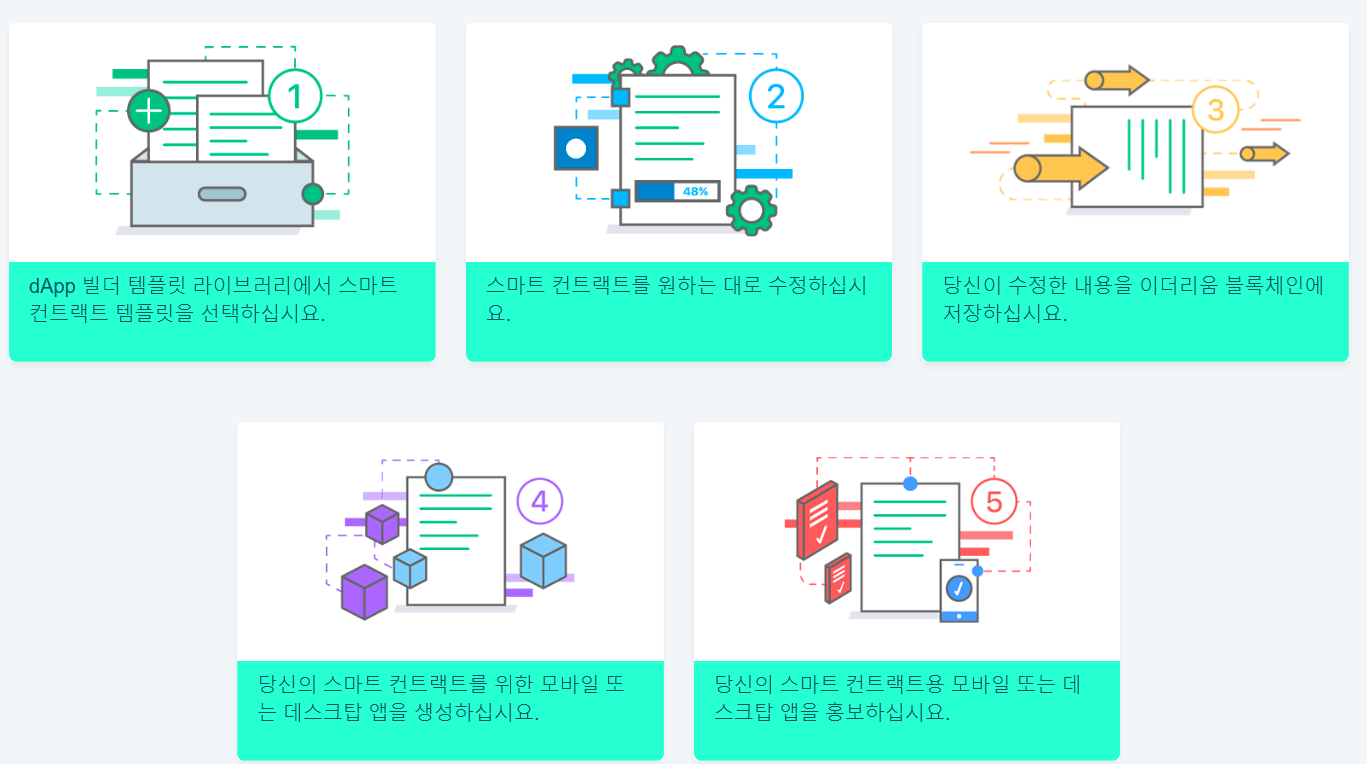
**송금 전과 후 확인**



**갱신**

**송금 전에는 0wei이었습니다. 송금을 하기 위해 Value값에 10wei를 입력하고 fallback 버튼을 누르니 value에 입력한 값이 자동으로 0이 되면서 조금 후에 10wei가 송금이 된 것을 확인할 수 있었습니다. 송금자 주소와 송금한 ether를 확인해보기 위해 sender버튼과 recvEther버튼을 클릭함으로써 갱신이 되었고 10wei가 송금된 것을 제대로 확인할 수 있었습니다.**

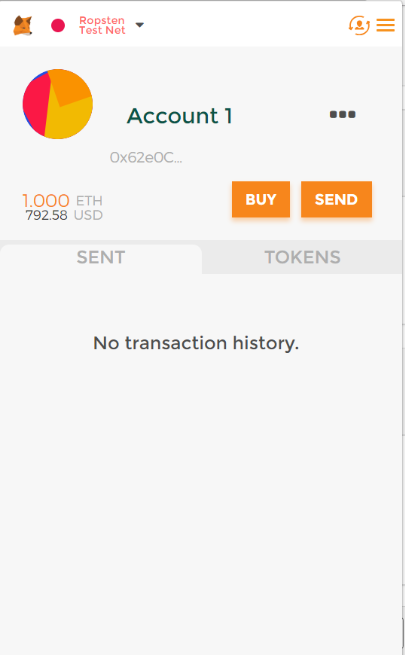
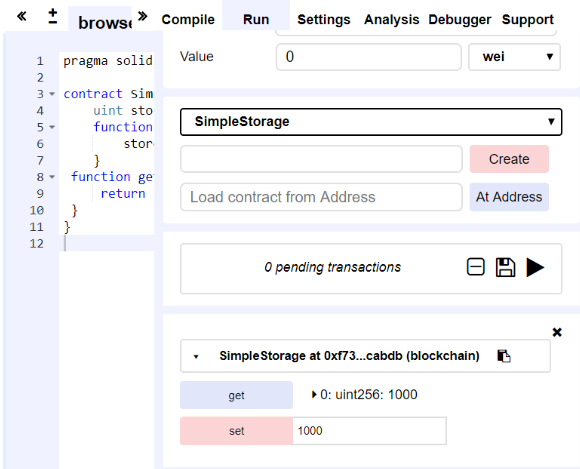
1. Dapp 구축



**(Dapp의 구동 방식에 대한 설명입니다.)**

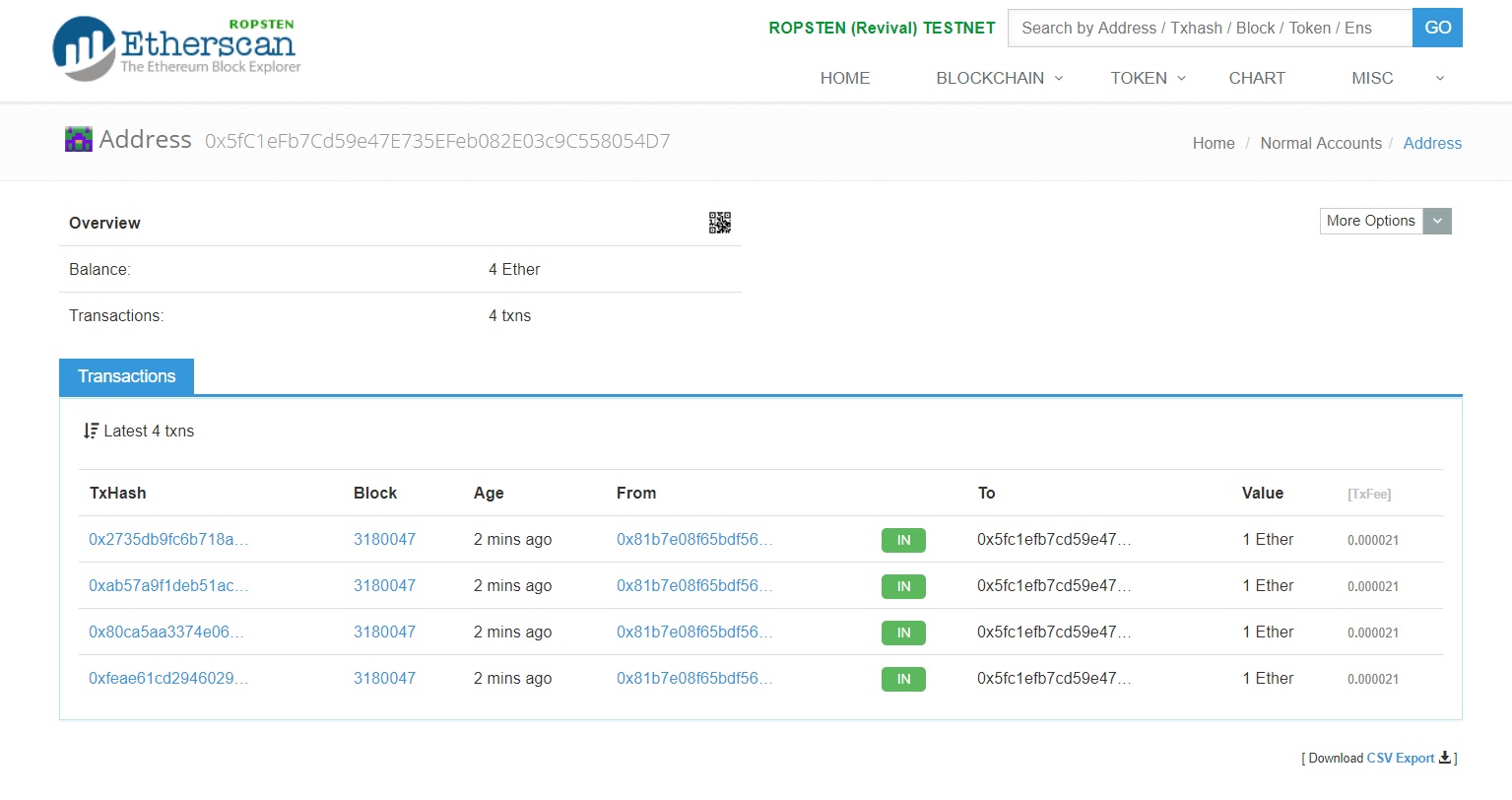
**저희 팀은 공통 서버 내에서 서로의 계좌에 송금하는 것을 시도하였습니다. 공통 서버를 사용하면서 Dapp을 만들 수 있는 방법은 많았지만 진행 중인 프로젝트에서는 metamask와 mist를 중심적으로 사용해보고 비교해보기로 하였습니다.**

1. Metamask

**(좌: metamask/우: Remix IDE)**

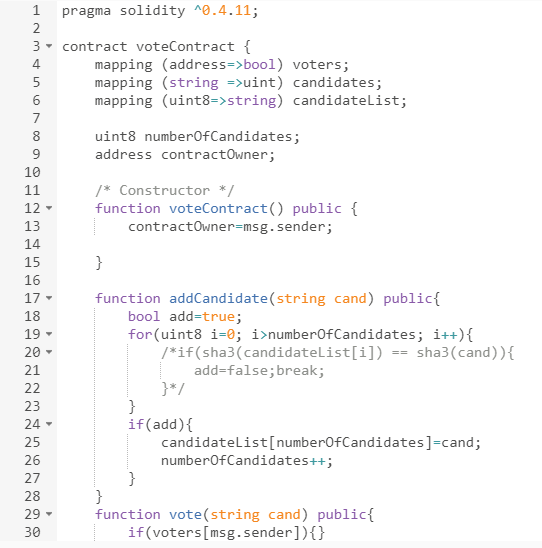
**metamask를 다루는 법을 배우기 위해 솔리디티와 연동시켜 다른 컴퓨터 사이의 이더 전송과 트랜잭션을 통해 정보를 보내는 예제입니다. 계좌와 계좌 사이에 트랜잭션을 보내 그 안에 값을 보내는 것입니다. 우측의 remix에서 SimpleStorage를 배포합니다. 이 컨트랙트에서 set 함수에 정수를 보내고 get을 통해 정수 값을 읽어 들이는 함수를 코딩했습니다. 우선 set값을 1000을 넣고 트랜잭션을 하면 metamask에서 트랜잭션 값에 16진수값 1000이 들어가 있는 것을 확인해볼 수 있었습니다.**



**위 사진은 송출금과 트랜잭션을 보여주는 페이지입니다.**

1. Mist

**미스트 프로그램 내에 기본적으로 다른 ip로의 송출금은 가능했습니다. 하지만 웹과 mist 프로그램을 연결시키기 위해서는 컨트랙트를 생성하고 그 주소가 필요했습니다. 그래서 컨트랙트를 알아보기 위해 투표 예제를 실행해보았습니다.**



**우선 mist 프로그램 내에서 컨트랙트를 생성하고 생성된 컨트랙트의 인터페이스와 주소를 복사합니다. 그 후 html파일 소스 내에 있는 문장을 수정하면 웹과 mist 프로그램이 연결되고 구동이 가능해집니다.**





**위 사진은 웹과 mist 프로그램이 연결된 상태입니다.**

1. **미니 월렛 예제 실행**



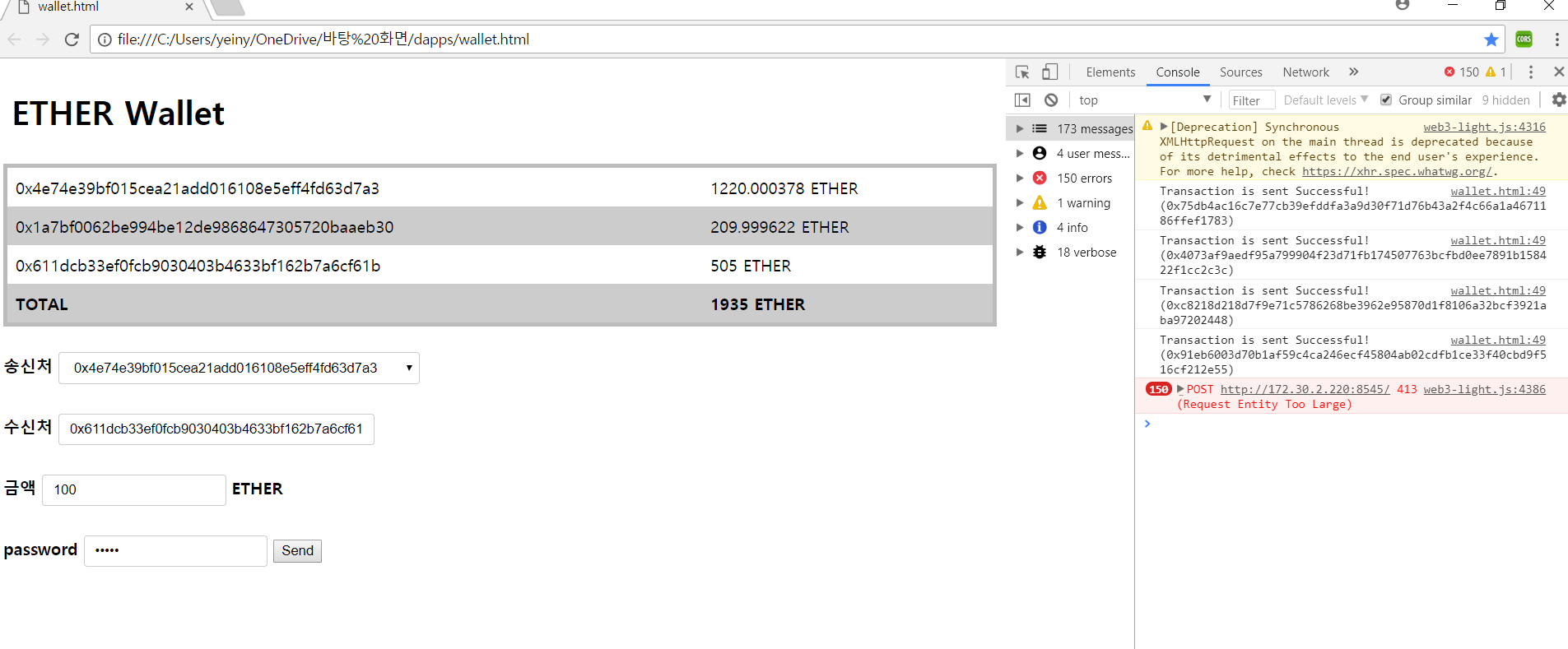
**Node.js는 chrome V8 JavaScript엔진으로 빌드 된 JavaScript 런타임입니다. Node.js는 이벤트 기반, Non 블로킹 I/O모델을 사용해 가볍고 효율적입니다. Node.js 의 패키지 생태계인 npm은 세계에서 가장 큰 오픈 소스 라이브러리 생태계이기도 합니다. Npm은 Node Package Modules라고도 합니다.**

**이번에 실행한 미니 월렛은 스마트 컨트랙트를 사용하지 않은 미니 이더리움 월렛입니다. 이더리움의 잔액 확인 및 이더를 송금하는 Mist의 기능을 html에서 작동되게 만들어 보는 게 목적입니다.**

**HTML파일에서 web3.js 라이브러리를 사용하려면 다운로드를 받아야 했습니다.**

**>npm install web3**

**이후에 미니 월렛을 만든 html 파일 위치에 lib 폴더를 생성하고 bignumber.min.js와 web3-light.min.js를 저장합니다.**

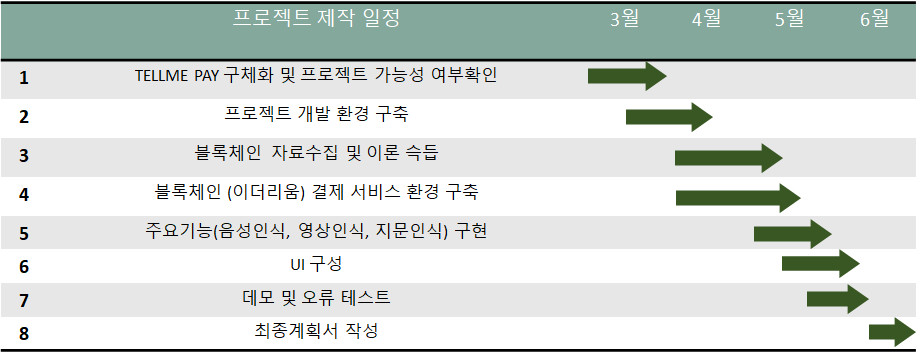


**Html페이지에서는 채굴함에 따라 계속적으로 증가하는 ether를 볼 수 있습니다. 해당 파일에서 가장 중요한 부분은**

**web3,setProvider(new web3.providers.HttpProvider(http://localhost:8545)**

**입니다. 포트번호 8545는 별도로 지정하기 전에는 동일합니다. chrome창에서 F12를 누르면 사진의 우측과 같은 개발자 페이지가 뜨는데 이를 통해 트랙잭션이 이루어졌는지 혹은 어떤 오류가 발생하였는지 등의 결과를 확인 할 수 있습니다. 채굴한 후 각각의 송신처, 수신처, 금액 등을 넣고 송신처의 비밀번호까지 입력하게 되면 트랙잭션이 성공했다는 결과를 얻게 됩니다.**

1. **개발 계획**



**처음 두 달은 프로젝트의 실효성 및 가능성에 대해 확인한 후에 개발환경을 구축하고 블록체인에 대한 스터디를 하는 것이 목표였습니다. 이것을 하고 난 후 이더리움을 통해 결제 서비스 환경을 구축 하고 주요 기능인 음성인식, 영상인식, 지문인식을 통해 프로젝트의 정체성을 만든 후에 UI를 구성하는 것이 다음 단계입니다. 최종 발표 및 프로젝트 완성을 위해 데모를 실행하고 오류 테스트를 통해 부족한 점들을 보완 후 최종 계획서를 작성하는 것이 마지막 목표였습니다.**

1. **프로젝트 미완성에 대한 보고와 분석**

**저희 팀은 위에서 보고한 바와 같이 약 5~6주만에 Let’s park 프로젝트를 성공하기에 앞서 Tell me pay를 개발했었습니다. 우선 블록체인은 저희에게도 듣기에는 매우 익숙한 이름이지만 그것을 개발한다는 것은 큰 도전이고 어려운 퀴즈를 푸는 것과 같았습니다. 팀원 모두 블록체인에 대해 관심이 높았고 4차 산업혁명이라고 불리는 지금 세계적으로 많은 이슈가 되는 만큼 공부하고자 하는 욕심이 커서 시작하게 되었습니다.**

**블록체인은 가상화폐인 금융에만 한정되어 있지 않고 범용으로 사용될 가능성도 높은 기술입니다. 중앙 관리 주체에 유일하게 보관 및 관리되던 거래의 기록을 블록체인 기술을 통해 모든 사용자들이 동일하게 관리할 수 있다는 것에 큰 매력을 느꼈습니다. 한 사용자의 조작으로 인해 모든 정보가 조작되지 않는 점에서 어떤 원리일까 궁금해하며 해당 프로젝트를 시작하기로 결심하였습니다.**

**블록체인의 기초 개념 또는 geth 설치에 대한 글들은 많은 사람들에 의해 사용되고 있었습니다. 그러나 windows 환경에서 개발하는 사람들은 비교적 적었습니다. 일반적으로 Mac, Ubuntu 또는 Linux 환경에서 개발하는 것이 대부분이었습니다. 따라서 설치 초기부터 약간의 문제를 겪게 되었습니다. 더 큰 문제는 많은 사람들이 기본만 배우고 이후에 깊은 이해나 개발에 대해서는 확실히 글들이 점점 줄어듦을 느낄 수 있었습니다. 즉, 개발할 때 넓고 얕은 정보들로 인해 개발이 점점 어려워졌습니다. 책만 읽으며 개발하기에는 턱없이 부족하였고 저희 팀은 구글링을 통해 정보를 습득하고자 하였습니다. 구글링을 할수록 얻을 수 있는 정보들은 한계가 있었고 그 한계점은 생각보다 낮았습니다.**

**블록체인을 돌릴 수 있는 web과 안드로이드 스튜디오를 연결하는 과정에서 저희는 한계를 느껴서 도중에 개발을 중단하게 되었습니다. 가장 큰 이유는 한정적인 정보들이었습니다. 또한 안드로이드 스튜디오를 다루는 데에 있어서나 블록체인을 다루는 데에 있어서 두 분야에 대해 모두 완벽하지 않았습니다. 그 큰 두 개념을 한꺼번에 공부하는 것은 확실히 한계점에 금방 도달하게 하는 또 다른 큰 요인이었습니다.**

1. **참고 문헌**

-이더리움 베이직 (스마트 컨트랙트 입문에서 DApp구현까지) –조수현, 이정빈, 박재용 외 2명 저

-블록체인 구조와 이론(예제로 배우는 핀테크 핵심 기술) – 아카바네 요시하루, 아이케이 마나부 저

-블록체인 애플리케이션 개발 실전 입문 – 와타나베 아츠시, 마츠모토 유타 외 2명 저