산학연계SW프로젝트
중간보고서

프로젝트 제목 : 스마트 컨트랙트를 이용한
탈 중앙화된 블록체인 토큰 교환 시스템

프로젝트 수행기간 : 2018.10.01 ~ 2019.05.31.

프로젝트 팀명 :

지도 교수 :

참여업체 명 :

2018. 12. 07

광운대학교
KwangWoon University

산학연계SW프로젝트 중간보고서

<table>
<tr><td>과 제 명</td><td colspan="5">스마트 컨트랙트를 이용한 탈중앙화된 블록체인 토큰 교환 시스템</td></tr>
<tr><td>팀 명</td><td colspan="5"></td></tr>
<tr><td>수행기간</td><td colspan="5">2018년 10월 01일 ~ 2019년 05월 31일</td></tr>
<tr><td>과제비</td><td colspan="5"></td></tr>
<tr><td>지도교수</td><td>성 명</td><td colspan="2">이기훈</td><td>학 부</td><td>컴퓨터정보공학부</td></tr>
<tr><td rowspan="6">참여학생</td><td>성 명</td><td colspan="2">학 부</td><td colspan="2">학 번</td><td>email</td></tr>
<tr><td></td><td colspan="2">컴퓨터공학과</td><td colspan="2"></td><td></td></tr>
<tr><td></td><td colspan="2">컴퓨터공학과</td><td colspan="2"></td><td></td></tr>
<tr><td></td><td colspan="2">컴퓨터공학과</td><td colspan="2"></td><td></td></tr>
<tr><td></td><td colspan="2"></td><td colspan="2"></td><td></td></tr>
<tr><td></td><td colspan="2"></td><td colspan="2"></td><td></td></tr>
<tr><td rowspan="2">참여업체</td><td>회사명</td><td colspan="2"></td><td>담당자</td><td></td></tr>
<tr><td>연락처</td><td colspan="2"></td><td>email</td><td></td></tr>
<tr><td colspan="4">산학연계SW프로젝트의 결과물에 대한 동의서</td><td colspan="2">동의여부</td></tr>
<tr><td colspan="2">프로젝트에 대한 지원</td><td colspan="3">본 프로젝트는 과학기술정보통신부 및 정보통신기술진흥센터의 SW중심대학사업의 지원으로 수행된 것임
(과제번호: 2017-0-00096)</td><td>■</td></tr>
<tr><td colspan="2">프로젝트 결과물의 활용</td><td colspan="3">본 프로젝트의 결과물은 프로젝트에 참여하고 지원한 광운대학교(학생들과 지도교수), 참여업체 그리고 정보통신기술진흥센터의 소유물이며 향후 활용과 소유에 관해서는 참여자 간의 협의에 의해 결정할 수 있음</td><td>■</td></tr>
</table>

『산학연계SW프로젝트』지원계획에 따라 중간보고서를 제출합니다.

2018년 xx 월 xx 일

팀 장 (인)
팀 원 (인)
팀 원 (인)
팀 원 (인)
팀 원 (인)

지도교수 (인)

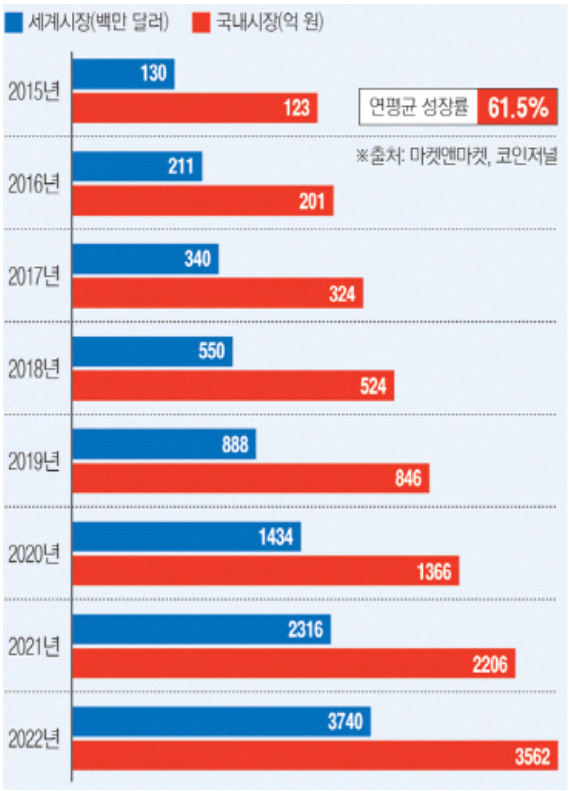
광운대학교 소프트웨어융합대학 귀하

목 차

1. 과제의 개요

가. 배경 및 필요성

4차 산업 혁명이 도래함에 따라 보안성이 뛰어난 블록체인 기술이 주목받고 있다. 이에 따라 국. 내외 많은 기업들이 블록체인 시장에 뛰어들고 있는 추세이다.



< 블록체인 국내·외 시장규모 및 전망 >

1. 탈중앙화된 거래(교환) 시스템 장점

1) 확장성
- 거래는 참여자들 간에서만 전송되고 원하는 수량만큼 한번만 거래하고 끝이기 때문에 공개 거래소와 같이 주문 수량이 다 채워질까 걱정하지 않아도 됨
- P2P 거래 방식에서는 이미 해당 주문의 수량만큼 관심을 표한 사람들을 매칭해주는 프로토콜을 사용하기 때문에 거래가 신속히 이루어짐

2) 공개성
- 두 참여자가 거래를 하기로 동의를 하게 되면 제 3자가 개입할 필요가 없게 됨
- 주문이 체결되고 나서야 주문에 대한 정보가 공개됨

3) 공정성
- 주문이 생성되어 거래 참여자들끼리만 전송하기 때문에 다른 사람이 이득을 취할 수 없음

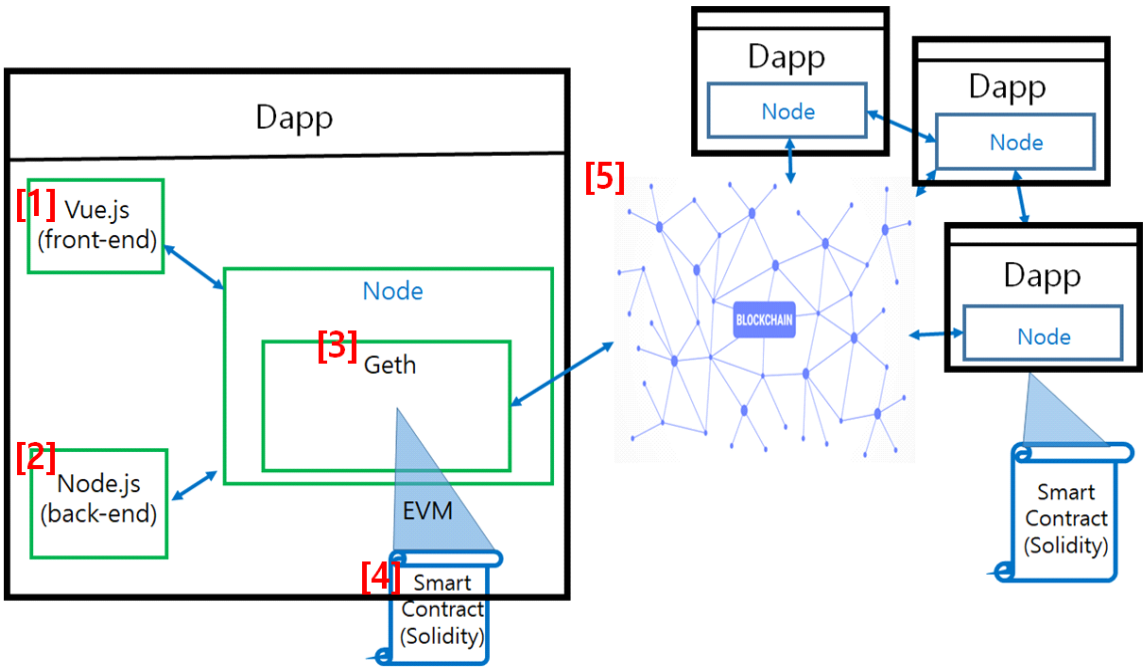
2. 토큰 교환 시스템의 필요성

각각의 플랫폼에 보유하고 있는 토큰은 그 플랫폼 안에서만 사용할 수 있도록 제한되어 있다. 이는 토큰의 이용 범위를 축소시키는 주요한 원인이 된다. 따라서 토큰의 활용도를 높이기 위해서는 토큰 간의 교환이 가능한 시스템이 필요하다.

나. 목적

현재 블록체인 토큰은 각 플랫폼 내의 사용으로만 제한되어있다. 예를 들어 A라는 서비스에서 사용되는 토큰 B가 필요한데 보유한 토큰이 C 밖에 없다면 A 서비스를 이용할 수 없게 된다. 만약 토큰 교환이 가능하게 된다면 토큰 종류에 구애받지 않고 서비스를 이용할 수 있을 것이다. 이 프로젝트를 통해 블록체인 기반의 토큰 교환 시스템을 구현함을 구현함으로써 토큰 교환의 보안성을 확보하고 토큰의 활용도를 높인다.

다. 설계 내용

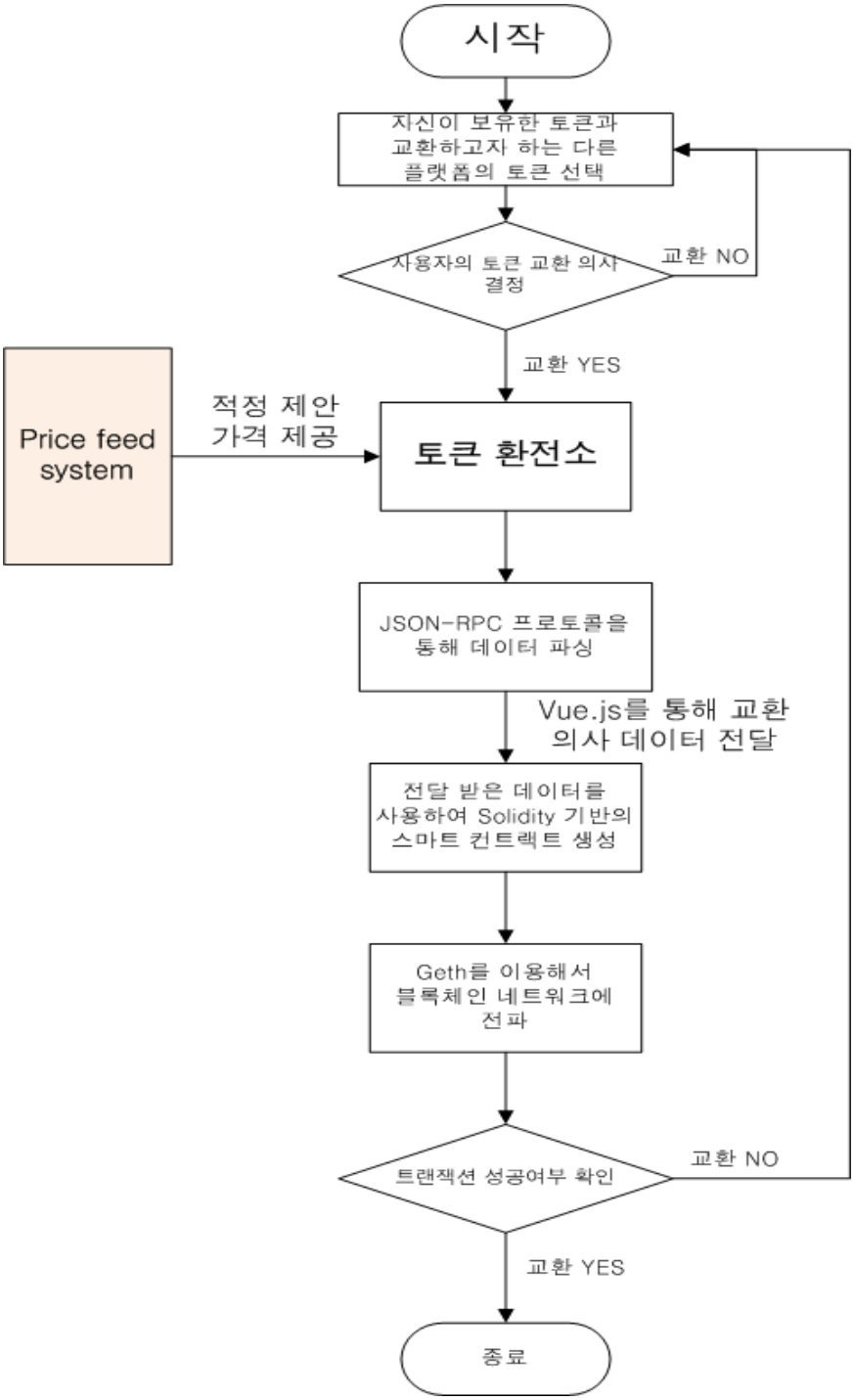


< 토큰 교환 시스템 Architecture >

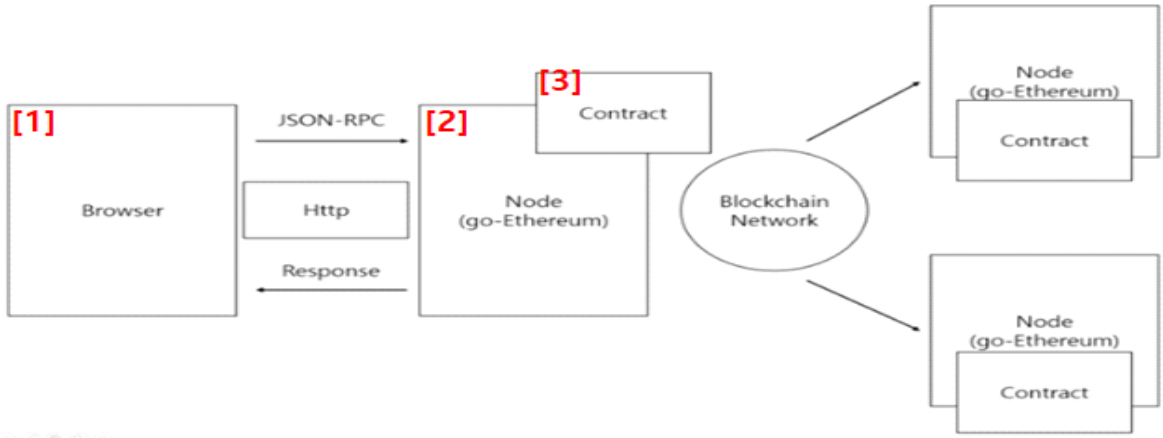
[1] Vue.js : 사용자 인터페이스를 구성
[2] Node.js : Vue.js로부터 입력 받은 정보를 geth로 전송
[3] Geth : Node.js로부터 전달 받은 정보를 이용해 스마트 컨트랙트를 생성 및 블록체인에 배포
[4] Smart Contract : Solidity 언어로 작성되어 있고 교환 의사 데이터가 내포된 코드
[5] BlockChain Network: Smart Contract가 배포되는 네트워크

1. 전체적인 시스템 구성

1) Flowchart



2) Contract 이동 과정



[1] Browser : 사용자가 교환하고자 하는 토큰 정보와 계좌 정보 입력 받음

[2] Node : 입력 받은 정보로 스마트 컨트랙트 생성 및 트랜잭션 유효성 검사

[3] Contract : 사용자가 입력한 토큰에 대한 환율 정보, 사용자의 계좌 정보를 가지고 있는 컨트랙트를 배포

2. 과제의 수행

가. 수행 방법 및 과정

역할 분담

성명	역할	공동
	- go-ethereum test net 학습 - 블록체인 기반 Smart contract 배포 구현 및 검증 담당	- 중간보고서 작성 - Dapp 개발 및 검증 - 최종보고서 작성
	- Solidity를 이용한 Smart contract 구현 담당 - go-ethereum를 이용한 transaction 학습	
	- Front-end 구현을 위한 vue.js 부분 담당 - crypto-zombie를 통한 solidity 학습 - 토큰 교환 시스템 구현을 위한 price feed system 구현 담당	

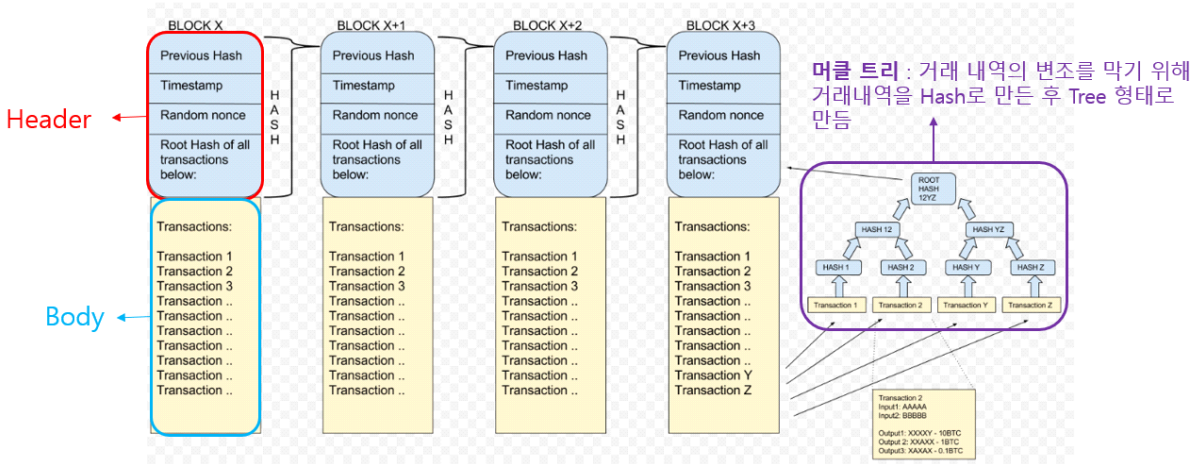
수행 일정

	9월	10월	11월	12월	1월	2월	3월	4월	5월
블록체인 개념 및 원리 이해	→								
스마트 컨트랙트 구현을 위한 solidity 학습		→							
토큰 교환 시스템 architecture 설계			→						
중간보고서 작성			→	→					
스마트 컨트랙트 구현 및 블록체인 환경 구성				→	→				
토큰 간 환율 정보를 제공하는 price feed system 설계 및 구현					→	→			
vue.js를 이용한 UI 개발						→	→		
토큰 교환 Dapp 시스템 검증 및 분석							→	→	→
최종보고서 작성									→

나. 중간 결과

1. Block chain 구조 학습

1) Block chain의 구조



2) 구성요소

① Hash 함수
- 어떤 데이터를 입력해도 같은 길이의 결과를 도출하는 함수
- 도출되는 결과가 중복될 가능성이 낮고, 결과 값으로 입력 값을 역으로 추정하기 어려움
- Hash 값 비교를 통해 데이터의 변경 유무 파악 가능

② Merkle Tree
- 각각 발생한 Transaction을 Hash화 하여 두 개씩 묶으면서 계속 Hash화 하면 마지막에 결국 하나의 Hash값, 즉 Merkle root를 갖게 되는 구조
- 밑의 정보가 하나라도 바뀌면 가장 꼭대기에 있는 Merkle root 값이 달라진다는 특징으로 데이터의 변조 파악 가능

③ Nonce
- 확정되어 있지 않고 새로 구해야 하는 값
- 최초 0에서 시작하여 조건을 만족하는 Hash 값을 찾아낼 때까지의 1씩 증가하는 계산 횟수

④ Difficulty
- 블록 생성을 할 수 있는 권한을 획득하기 위해 풀어야 하는 문제의 난이도
- Difficulty가 높을수록 조건을 만족하는 Hash 값을 찾는 것이 어려지므로 블록 생성이 힘듬

2. Crypto zombie를 통한 Solidity 언어 학습

이더리움 기반의 스마트 컨트랙트를 작성하기 위해 사용되는 객체지향형 프로그래밍 언어이다. Solidity 학습을 할 수 있는 무료 코딩 스쿨인 Crypto zombie를 통해 다음과 같은 특징들에 대해 중점으로 학습했다.

Contract

1. Solidity 코드는 contract 안에 싸여 있다.
2. Contract는 이더리움 애플리케이션의 기본 구성요소로, 모든 변수와 함수는 어느 한 contract에 속한다.
3. 모든 Solidity 코드는 'version pragma'로 시작해야 하는데 이는 해당 코드가 이용하는 solidity 버전을 선언하는 것이다.

이벤트

1. 컨트랙트가 블록체인 상에서 내 앱의 사용자 단에서 무언가 액션이 발생했을 때 의사소통 하는 방법을 말한다.
2. 컨트랙트는 이벤트가 발생하면 행동을 취한다.

Msg.sender

1. 모든 함수에서 이용 가능한 전역 변수이다.
2. 현재 함수를 호출한 사람의 주소를 가리킨다.
3. Solidity에서 함수 실행은 항상 외부 호출자가 시작한다.
   - 그렇기 때문에 항상 msg.sender가 있어야 한다.
4. 이더리움 블록체인의 보안성을 활용할 수 있는 키워드이다.

인터페이스

1. 블록체인 상에 있으면서 사용자가 소유하지 않은 컨트랙트와 사용자의 컨트랙트가 상호작용을 하기 위해 필요하다.
2. 정의된 인터페이스 안에는 상호작용하고자 하는 함수만 언급한다.
3. 언급한 함수의 몸체는 정의하지 않는다.

가스 줄이는 방법

1. 저장 공간을 줄이면 가스를 아낄 수 있다.

2. 그러나 변수 선언 시 uint형 보다 작은 uint8, uint16과 같은 하위 타입을 사용하는 것은 소용없다.
   => Solidity에서는 uint의 크기에 상관없이 256bit의 저장공간을 미리 잡기 때문.

3. 구조체 안에서는 예외적이다.
   - 구조체 안에서는 가능한 한 작은 크기의 정수 타입 사용.
   - 동일한 데이터 타입은 하나로 묶으면 저장공간을 최소화 함.

3. Geth

1) Geth란

Geth(Go ethereum) 란 대표적인 ethereum 이더리움 프로토콜을 구현 하고 있는 클라이언트이다. 구글에서 만든 Go 언어를 기반으로 개발 되었으며 http://github.com/ethereum/go-ethereum 에서 안정화된 stable release 중 최신 버전을 설치하여 진행한다. Geth Ethereum 사설 네트워크(Test-Net)을 구축 할 수 있기 때문에 공용 이더리움과 달리 ether를 사용자가 원하는 대로 설정 할 수 있다. 이는 Dapp 의 개발과 테스트가 용이하다는 장점이 있다.

2) Genesis Block

Test-Net 을 구축하기 위해서는 Blockchain의 최초 Block인 Genesis Block을 생성하고 설정해야 한다.

```
{
    "config": {
        "chainId": 12345687,
        "homesteadBlock": 0,
        "eip155Block": 0,
        "eip158Block": 0
},
    "difficulty" : "0x400",
    "gasLimit"   : "0x2fefd8",
    "alloc"      : {},
    "coinbase"   : "0x0000000000000000000000000000000000000000",
    "extraData"  : "",
    "nonce"      : "0x0000000000000042",
    "mixhash"    : "0x0000000000000000000000000000000000000000000000000000000000000000",
    "parentHash" : "0x0000000000000000000000000000000000000000000000000000000000000000",
    "timestamp"  : "0x00"
}
```

위는 Genesis.json 파일의 내용이다. 각 항목의 사용법은 다음과 같다.

① config : 새로운 버전의 Geth를 이용

② nonce : mixhash 옵션과 함께 현 블록의 작업 증명을 위해서 사용
③ mixhash : nonce 옵션과 함께 현 블록의 작업 증명위해 사용
④ timestamp : 해당 블록이 취득된 시점을 나타내는 옵션.
⑤ parentHash : nonce 와 mixhash 를 포함한 부모 블록의 헤더에 대한 해시 값
⑥ extraData : 추가 될지 모르는 옵션을 위한 32byte 임시 공간
⑦ gasLimit : 하나의 블록이 담을 수 있는 gas의 임계치
⑧ difficulty : 블록 생성을 위한 계산 난이도 조절.
⑨ coinbase : 해당 블록에 대해서 채굴 성공 시 얻게 되는 총 보상금(이더)을 160비트의 주소값으로 표현한 옵션.
⑩ alloc : 원하는 액수의 ether를 미리 송금을 예약

3) Test-net 구축

```
@ubuntu:~/block$ geth --datadir "./data/ init genesis42.json
```

Genesis 블록 파일을 사용하여 사설 네트워크의 최초 블록을 생성하고 데이터 디렉토리를 지정한다.

```
jeonje@ubuntu:~/block$ geth --networkid 1988 --identity "test" --datadir "./dat
a/" --nodiscover console
```

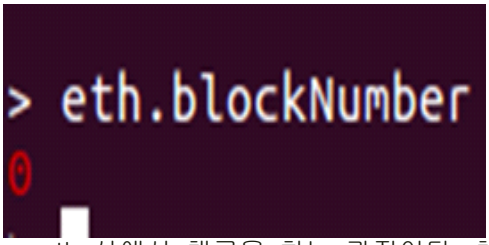
사설네트워크로 진행되기 때문에 블록체인 상에서 발견되지 않도록 nodiscover 로 설정하며 id 와 이름을 설정하고 대화형 자바 스크립트 콘솔 환경으로 geth을 실행한다. 콘솔 환경은 다양한 API를 이용하여 실시간으로 새로운 계좌를 생성 할 수 있고 Ehter를 다른 사용자에게 전송하고 기능을 활성화 또는 비활성화 할 수 있다.

```
> eth.accounts
["0x1562771898d08f99d6d944b69df5fa88c2bd4b2e", "0x5e1278a0fd14771b1e98c2296ece8e
15b9704af6"]
```

Test-net 상에서 Transaction을 위해 두 개의 계좌를 생성한다.

```
> eth.getBalance(eth.accounts[0])
0
```

새 계좌의 블락 수와 계좌잔고는 0이다.



geth 상에서 채굴을 하는 과정이다. 채굴을 통해 wei (10^-18 ether) 을 보상으로 얻고 eth.sendTransaction 명령어를 통해 receive 계좌의 주소로 이더를 전송한다.

receive 계좌의 잔액에 이더가 전송됨을 확인함으로써 Test-Net 상에서 Ether Transaction 을 확인하였다.

4. UI Scenario (별첨에 UI Image 첨부)

1) UI Scenario 의 스토리 보드

2) Dapp 의 첫 화면

3) 환전 금액 선택에 따른 환율 제공 화면

4) 토큰을 받을 계좌 선택

5) 입금을 대기하는 화면. 확인 또는 취소

6) 입금 실패 시 안내 메세지와 오류원인 출력. 처음으로 또는 다시 시도 선택

7) 입금 완료시 선택 화면. 환전 취소 또는 환전 진행

8) 환전소에서 송금 진행 화면

9) 환전소 측에서 송금 실패 시 화면. 실패 메세지와 오류 원인 출력

10) 환전 완료시 화면. 환전 내역을 확인하거나 처음으로

다. 향후 계획

Remix IDE를 사용해서 토큰 교환을 위한 스마트 컨트랙트를 구현한다. On-chain을 통한 price feed 제공은 신뢰성이 떨어지기 때문에 off chain에서 node.js 통해 price feed를 제공한다. 이를 위해 price feed system을 설계 및 구현을 진행할 예정이다. 블록체인상의 스마트 컨트랙트 교환 검증을 한 후 사용자의 데이터를 입력받을 웹 front-end를 vue.js를 통해 개발한다. 최종적으로 구현한 토큰 교환 Dapp system 검증한다.

3. 과제의 평가

가. 개선 방안

처음에는 환율 정보를 제공하는 price feed를 블록체인 상에서 제공하는 것으로 설계하였으나 Extales 와의 미팅을 통해 on-chain 에서는 price feed를 제공하지 못하고 off-chain에서 Node.js를 통해 토큰 환율 정보들을 종합해서 제공하는 방식으로 설계를 수정하였다.

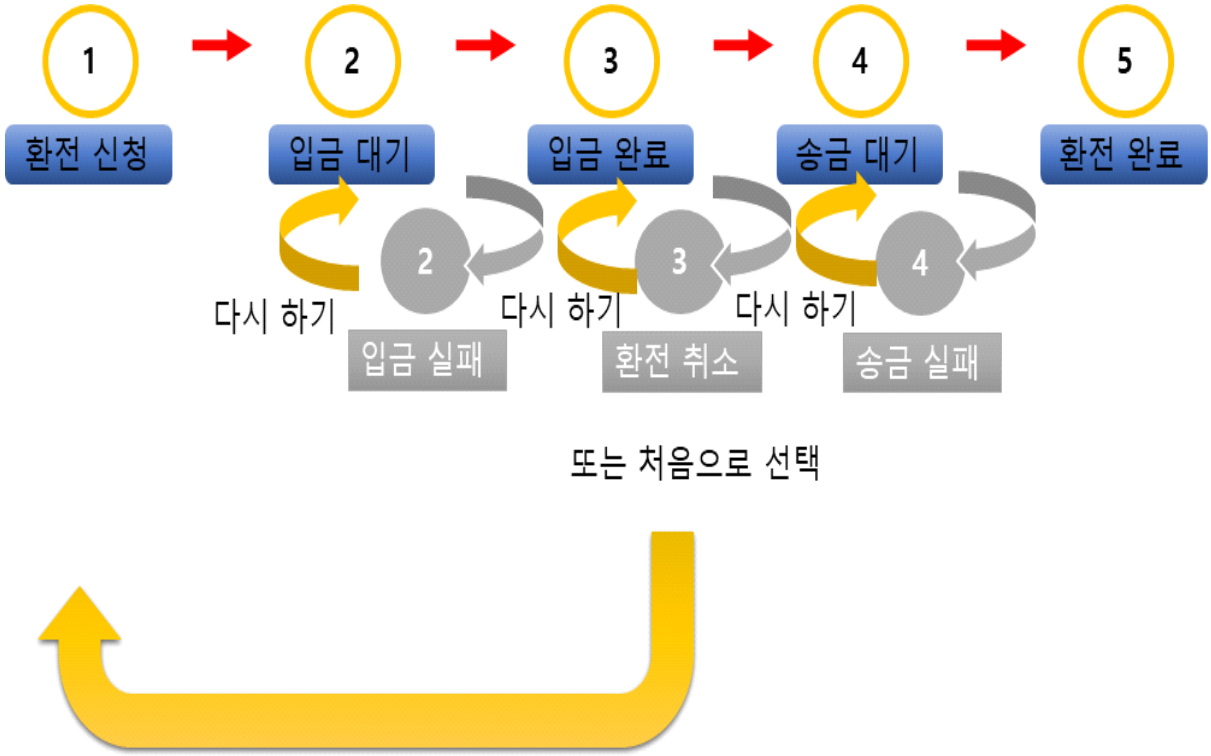
나. 기대 효과

현재 여러 블록체인 플랫폼 간의 토큰 교환은 어렵다. 따라서 다른 플랫폼의 서비스를 이용하기 위해서는 해당 플랫폼의 토큰을 새롭게 필요로 한다. 이는 사용자 입장에서 자신이 보유한 기존의 토큰을 이용하지 못하고 새로운 플랫폼의 토큰을 추가적으로 구매해야하기 때문에 비효율적이다. 하지만 토큰 간의 교환이 가능한 토큰 환전소가 있다면 토큰을 보다 효율적으로 활용할 수 있고 환전 시스템에 블록체인 기술을 적용시킨다면 보안성도 확보할 수 있다.

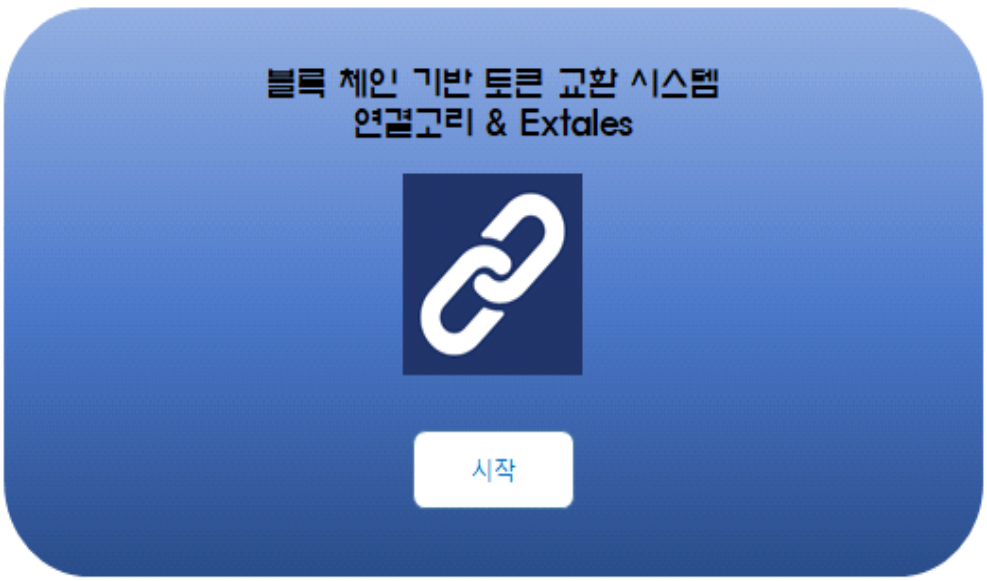
4. 별첨

“나. 중간결과”의 “4.UI Scenario”

1) UI Scenario 의 스토리 보드



2) Dapp 의 첫 화면

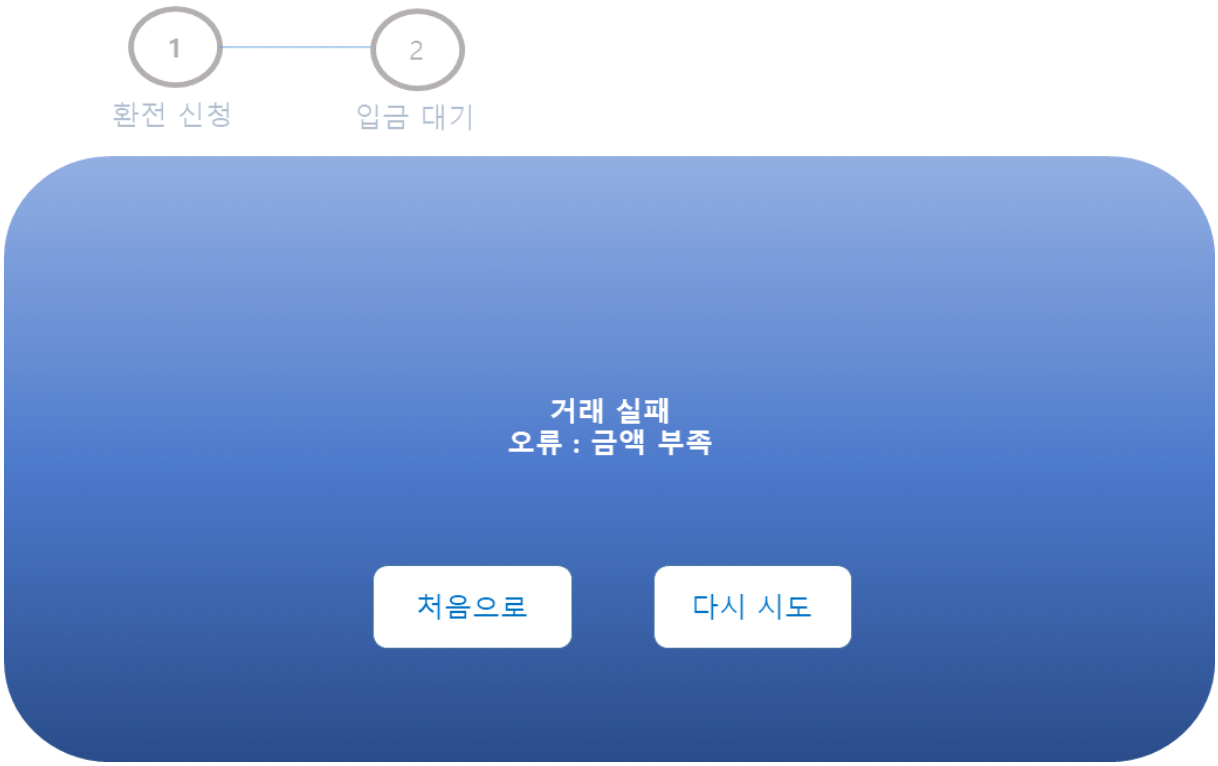


3) 환전 금액 선택에 따른 환율 제공 화면

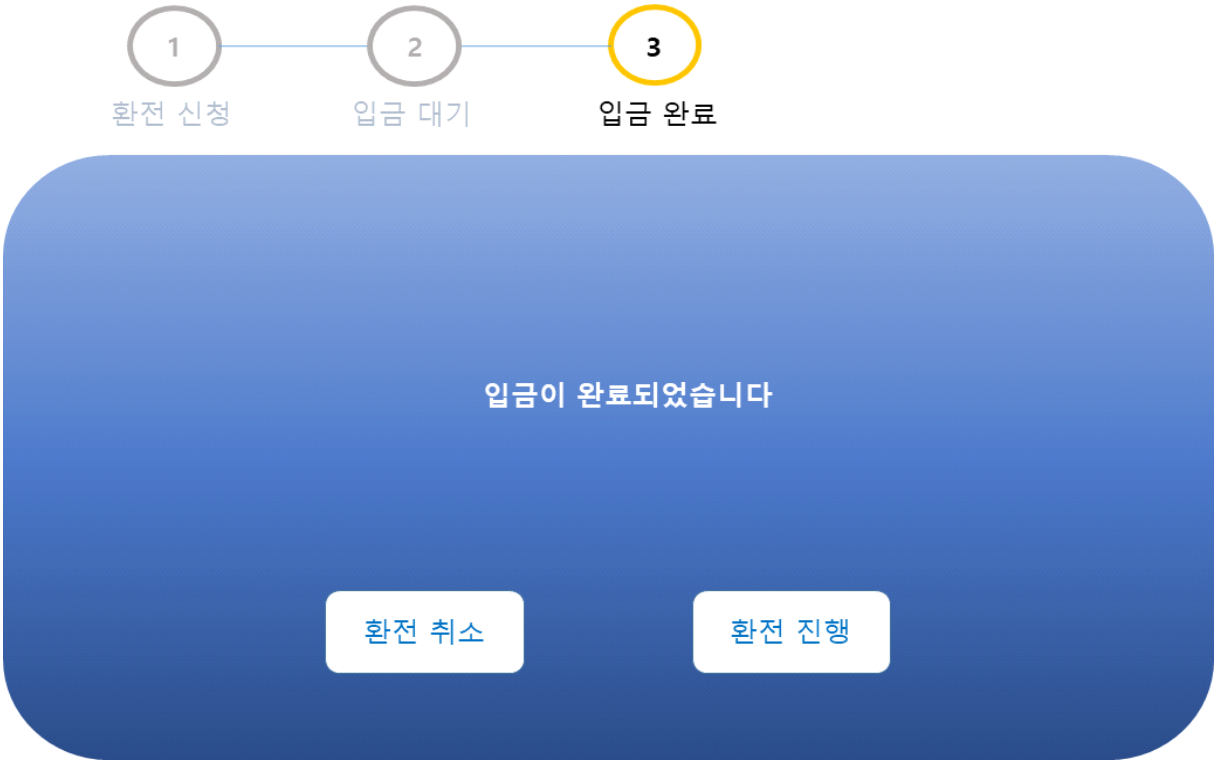
4) 토큰을 받을 계좌 선택

5) 입금을 대기하는 화면. 확인 또는 취소

6) 입금 실패 시 안내 메세지와 오류원인 출력. 처음으로 또는 다시 시도 선택

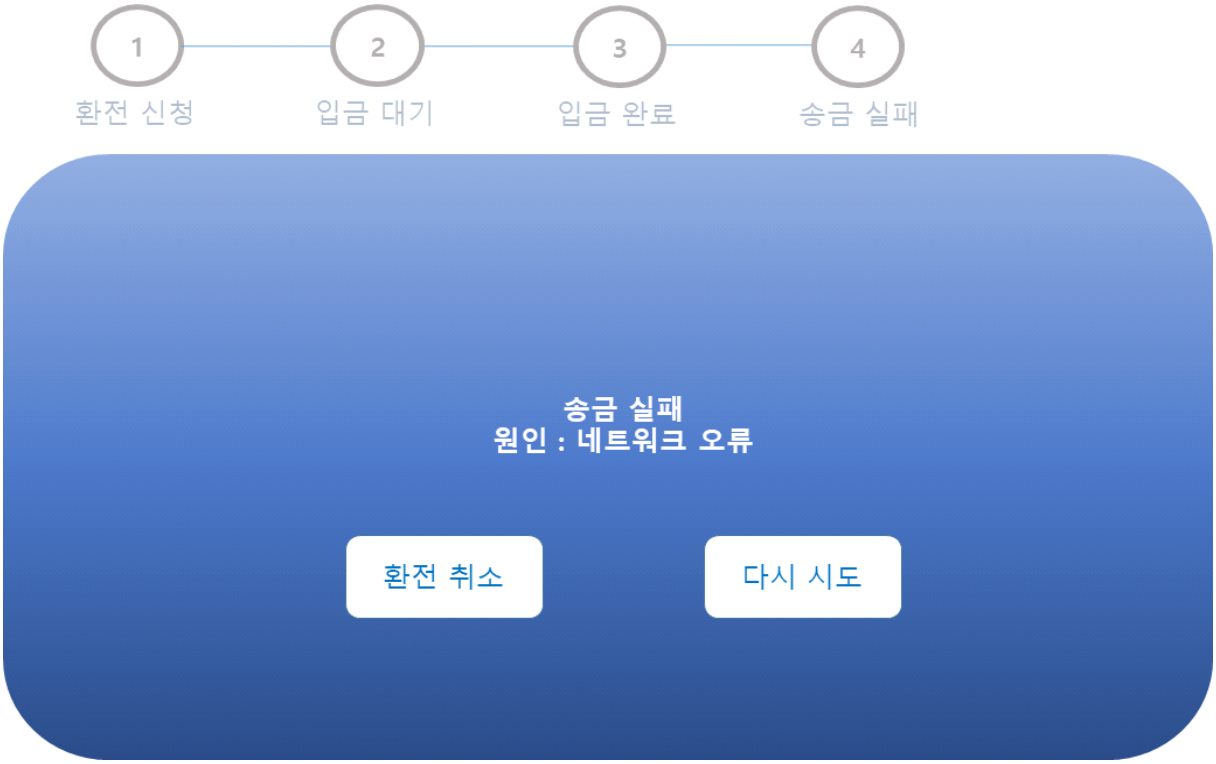


7) 입금 완료시 선택 화면. 환전 취소 또는 환전 진행



8) 환전소에서 송금 진행 화면

9) 환전소 측에서 송금 실패 시 화면. 실패 메시지와 오류 원인 출력



10) 환전 완료시 화면. 환전 내역을 확인하거나 처음으로