- 모두 :

**블록체인을 통한기부시스템운영 현황**

유니세프,알리바바(앤트 파이낸셜\_알리페이)에서 모금/기부금 사용내역 추적

<https://steemit.com/kr/@mastertri/7vvkwb>

**[해외]**

알리페이: 플랫폼의 기부섹션에 블록체인 기술로 자선단체 및 기부자가 기부금 이력과 사용현황 등을 추적

엔터테인먼트 시장특화 블록체인 플랫폼 STAYGE: 아티스트와 팬을 위한 블록체인 기부프로젝트. 팬들이 아티스트를 위해 기부금 모금. 굿즈,티케팅,스트리밍 등과도 관련. 팬활동을 하면 ACT로 보상하고 이 토큰을 스타의 이름으로 기부.

유니세프 Game Chaingers: 고성능 GPU를 갖춘 컴퓨터 게이머의 장비를 채굴장비로 채굴한 이더리움을 시리아 난민을 위해 유니세프에 기부. 암호화폐로 식수,의료서비스 등 지원.

**[국내]**

\*우리나라 세계경제순위 10위권, 기부현황은 75위

경기도, 기부금 운용내역 투명화로 기부문화 확대

E4Net의 기부플랫폼 ‘체리’ : 토큰을 사용한 기부로 사용성이 좋은 플랫폼을 만들고 토큰으로 게임 아이템 등 구매가능하도록

KT 기부포털 기브스퀘어: 봉사활동을 통해 포인트를 쌓고 보유한 포인트를 기부. 기부내역, 기부금 집행내역, 기부금 전달경로 등을 원장에 저장. P2P 기부까지 구현하려고 개발 중.

**- 태양광 등 잉여에너지 수집방법 및 블록체인과 접목방법**

1. 개인이 생산한 태양광 에너지를 한국전력에 SMP(전력판매가격)을 받고 판매, 한국수력원자력 등 대형 발전사에 REC(신재생에너지공급인증서)를 팔 수도 있음.

한국전력과 대형발전사가 고정거래처이지만 1000kW 이하의 소규모 신재생발전에는 의무적으로 전력을 받아주도록. 하지만 신재생에너지용 변전설비부족으로 송전은 어려움. 가격변동성도 문제.

개인태양광에너지 발전이 어려운 까닭 중 하나는 수익이 들어오지 않는 점이다. 보통 개인 태양광발전을 은행 이자보다 좀 더 좋은 투자수단으로 홍보를 하는데 한전에서는 여러문제로 돈을 지급해주지 않고 해외와 비교했을때는 애초에 전력요금이 비현실적으로 쌈.

가격변동성을 해결해주거나 보완해줄수 있는 제도가 필요.

- 각자 맡으신 부분 :

**이더리움을 웹과 연동하는 방법**

**태양열 등 잉여에너지를 기부코인화 하는 방법**

[원화-코인-원화로 환전하는 PG사의 코인활용 기부흐름]

(1) 도움이 필요한 조직/단체는 그 이유와 필요한 금액, 사용계획등을 정리해서 직접 공유게시판에 올립니다.  
  
(2) 잠재 기부자들의 질문과 답등을 통해 1차적으로 평가하고, 2차적으로 네트워크 투표로 임명된 큐레이터들이 직접 현장방문 검증도 실시할 수 있습니다.  
  
(3) 기부자는 이더, 다른 암호화폐, 크레딧카드, 원화 이체으로 원하는 대상에 기부할 수 있습니다. 원화기준으로 기부한 경우 이에 상응하는 기준 암호화폐로 자동전환되고, 모든 경우에 블록체인상에 기록되는 기부확인증을 받습니다.  
  
(4) 미리 설정된 조건에 충족될 경우, 기부를 받는 조직/단체의 이더 또는 기준 암호화폐 계좌로 모든 기부금이 전달되고, 이것은 다시 이 계좌에 연결된 원화계좌로 전환 입금됩니다. 블록체인상의 계좌와 원화은행계좌와의 연결확인은 큐레이터에 의해 검증됩니다.  
  
(5) 일정시간 경과이후, 다른 큐레이터에 의해 기부금액이 계획대로 적절하게 집행되었는지, 그리고 어떤 성과가 있었는지 등에 대해 다시 조사하고 이를 네트워크에 보고하게 합니다.  
  
(6) 기부네트워크에 의해 성공적으로 집행된 사례들이 집적될 수록 큐레이터들에 대한 평판 신뢰도는 더욱 올라갈 것입니다.

**solidity를 통하여 기부(거래)하는 코드 분석**

깃허브 링크: <https://gist.github.com/pau1m/baa7e8a8dfe1c2852f0c876b61ba6af2>

|  |
| --- |
| Pragma solidity ^0.4.4; |
|  |  |
|  | // TestRPC HD wallet |
|  | // warrior minimum breeze raven garden express solar flavor obvious twenty alpha actress |
|  |  |
|  | contract Donation { |
|  |  |
|  | // Instantiate a variable to hold the account address of the contract administrator |
|  | address public owner; |
|  |  |
|  | // Create a data structure to reperesent each of the participants. |
|  | struct Payee { //수취인 |
|  | // If the payee can administer their account. |
|  | bool status; |
|  | // The amount relative to aggregate weight that a payee will receive. |
|  | uint weight; |
|  | // A record of the amount held for the payee. |
|  | uint balance; |
|  | } |
|  |  |
|  | // Create an associative arrays with account address as key and payee data sturcture as value. |
|  | mapping(address => Payee) public payees; |
|  | // Create an array like mapping to behave as an index of addreesses |
|  | mapping (int8 => address) public payeesIndex; |
|  | // Keep note of total number of participants in the system so we can iterate over index. |
|  | int8 public payeesIndexSize; //초기화?? |
|  |  |
|  | // Declare events for actions we may want to watch |
|  | event NewDonation(address indexed donator, uint amt); //amt?? |
|  | event Transfer(address indexed from, address indexed to, uint amt);//거래이벤트 |
|  | event PayeeAction(address indexed payee, bytes32 action); // |
|  | event Withdrawal(address indexed payee, uint amt); //회수 |
|  | event OwnerChanged(address indexed owner, address indexed newOwner); |
|  | event ContractDestroyed(address indexed contractAddress);// |
|  |  |
|  | // Constructor |
|  | function Donation() { |
|  | // Set the address of the contract deployer to be owner. |
|  | owner = msg.sender; //msg?? |
|  | payees[owner].status = true; |
|  | payees[owner].weight = 10; |
|  | payeesIndex[0] = owner; |
|  | payeesIndexSize = 1; |
|  | } |
|  |  |
|  | // Check if current account calling methods is the owner. |
|  | modifier isOwner() { |
|  | if (msg.sender != owner) throw; |
|  | \_; |
|  | } |
|  |  |
|  | // Check if current account calling methods is a valid payee of the contract. |
|  | modifier isPayee() { |
|  | if (payees[msg.sender].status != true) throw; |
|  | \_; |
|  | } |
|  |  |
|  | // Aggregate all payee weights.//weight가 뭐지.. |
|  | function getTotalWeight() private returns (uint) { |
|  |  |
|  | int8 i; |
|  | uint totalWeight = 0; |
|  |  |
|  | for (i=0;i<payeesIndexSize;i++) { |
|  | if (payees[payeesIndex[i]].status == true) { |
|  | totalWeight += payees[payeesIndex[i]].weight; |
|  | } |
|  | } |
|  |  |
|  | return totalWeight; |
|  | } |
|  |  |
|  | // Function which will accept donations. |
|  | function deposit() payable { |
|  |  |
|  | if (msg.value == 0) throw; //throw=break??? |
|  | int8 i; |
|  | uint totalWeight = 0; |
|  |  |
|  | totalWeight = getTotalWeight(); |
|  | // Update account balances for all payees. |
|  | for (i=0;i<payeesIndexSize;i++) { |
|  | if (payees[payeesIndex[i]].status == true) { |
|  | uint divisor = (totalWeight / payees[payeesIndex[i]].weight); |
|  | payees[payeesIndex[i]].balance = msg.value / divisor; |
|  | } |
|  | } |
|  |  |
|  | NewDonation(msg.sender, msg.value); |
|  | } |
|  |  |
|  | // Add a new payee to the contract. |
|  | function addPayee(address \_payee, uint \_weight) isOwner returns (bool) { |
|  |  |
|  | payees[\_payee].weight = \_weight; |
|  | payees[\_payee].status = true; //payee 구조체를 건드릴 수 있게 해주는 부분?? |
|  | payeesIndex[payeesIndexSize] = \_payee; |
|  | payeesIndexSize++; |
|  |  |
|  | PayeeAction(\_payee, 'added'); |
|  | } |
|  |  |
|  | // Amend payee weight. |
|  | function updatePayeeWeight(address \_payee, uint \_weight) isOwner { |
|  | payees[\_payee].weight = \_weight; |
|  | } |
|  |  |
|  | // Disallow an account address from acting on contract. |
|  | function disablePayee(address \_payee) isOwner returns (bool) { |
|  | if (\_payee == owner) throw; // Don't lock out the main account |
|  | payees[\_payee].status = false; |
|  | PayeeAction(\_payee, 'disabled'); |
|  | } |
|  |  |
|  | // Allow an account address from acting on contract. |
|  | function enablePayee(address \_address) isOwner { |
|  | payees[\_address].status = true; |
|  | PayeeAction(\_address, 'enabled'); |
|  | } |
|  |  |
|  | // Allows payee to withdraw eth to their Ethereum account address. |
|  | function withdraw(uint amount) payable isPayee { |
|  | if (payees[msg.sender].status != true || amount > payees[msg.sender].balance) throw; |
|  | if (!msg.sender.send(amount)) throw; |
|  | Withdrawal(msg.sender, amount); |
|  | payees[msg.sender].balance -= amount; |
|  | } |
|  |  |
|  | // Transfer some Ether available to withdraw to another account. |
|  | function transferBalance(address \_from, address \_to, uint amount) isOwner { |
|  | if (payees[\_from].balance < amount) throw; |
|  | payees[\_from].balance -= amount; |
|  | payees[\_to].balance += amount; |
|  | Transfer(\_from, \_to, amount); |
|  | } |
|  |  |
|  |  |
|  | function getBalance(address \_address) isPayee returns (uint) { |
|  | return payees[\_address].balance; |
|  | } |
|  |  |
|  |  |
|  | function getWeight(address \_address) isPayee returns(uint) { |
|  | return payees[\_address].weight; |
|  | } |
|  |  |
|  |  |
|  | function getStatus(address \_address) returns(bool) { |
|  | return payees[\_address].status; |
|  | } |
|  |  |
|  |  |
|  | // Change ownership of the contract. |
|  | function transferOwner(address newOwner) isOwner returns (bool) { |
|  | if (!payees[newOwner].status == true) throw; |
|  | OwnerChanged(owner, newOwner); |
|  | owner = newOwner; |
|  | } |
|  |  |
|  | // Destroy the contract and pay out all enabled members. |
|  | // Any outstanding value will be transferred to owner. |
|  | function kill() payable isOwner {//enabled된 모든 payee를 삭제하는 코드?? |
|  | int8 i; |
|  | address payee; |
|  |  |
|  | for (i=0;i<payeesIndexSize;i++) { |
|  | payee = payeesIndex[i]; |
|  | if (payees[payee].balance > 0 ) { |
|  | if (payee.send(payees[payee].balance)) { |
|  | Withdrawal(payee, payees[payee].balance); |
|  | } |
|  | } |
|  | } |
|  |  |
|  | ContractDestroyed(this); |
|  | selfdestruct(owner); |
|  | } |
|  | } |