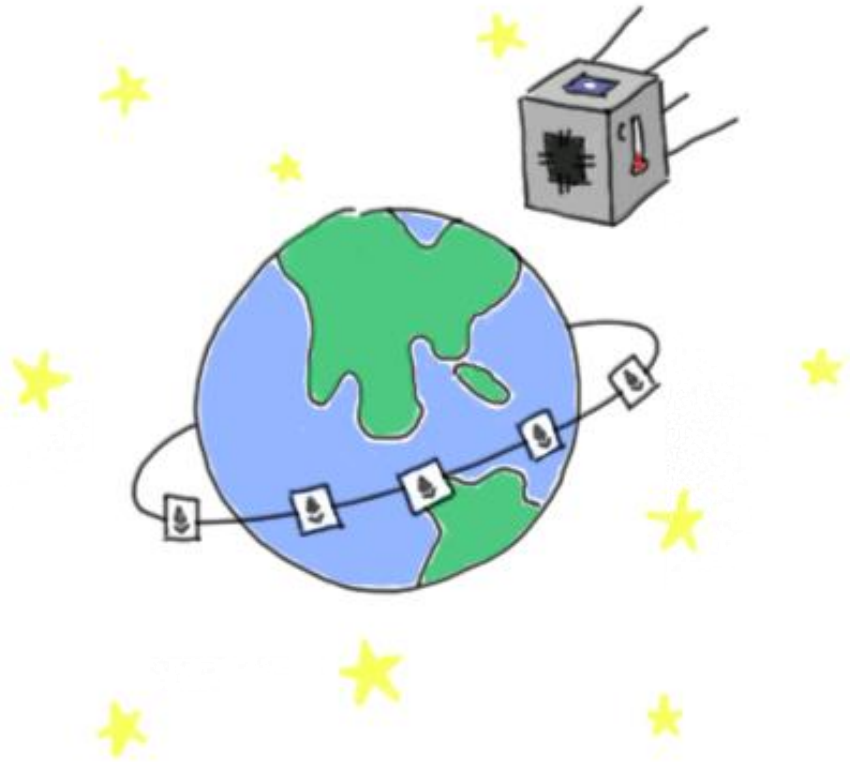


Monitoring von Sensordaten mithilfe der Blockchain und eines SmartContracts

Maturaarbeit von Artem Anchikov

Gymnasium Lerbermatt, 20e

Betreut von Herr Kai Rollé

14.10.19

Inhalt

Einleitung

Idee

Das Ziel dieser praktischen Maturaarbeit ist die Erstellung einer Datenbank zur Speicherung von Sensordaten mithilfe eines SmartContracts und des dazugehörigen Webinterfaces.

Die Datenbank speichert Sensor-Informationen, welche sie vom Mikrocontroller Arduino erhält. Die Datenbank ihrerseits wird auf der Ethereum-Blockchain gelagert. Dadurch wird sichergestellt, dass die Möglichkeit, die Information zu verändern (fälschen), eliminiert wird. Der Zugriff zur Datenbank sollte durch eine Webseite vermittelt werden. Um dies zu veranschaulichen, wird ein Modell einer Kühlbox, die für den Transport von Organen verwendet werden kann, gebaut. Die Kühlbox wird mit verschiedenen Sensoren, wie Temperatur- und Fingerabdrucksensor, ausgerüstet und die Sensordaten werden anschliessend herausgelesen und in der Blockchain gespeichert.

Motivation

Ich interessiere mich seit 2017 für Blockchain. Damals habe ich zum ersten Mal über SmartContracts gelesen und wollte selbst einen kleinen SmartContract schreiben. Im Februar 2019 habe ich an einem Hackathon (Cryptochicks Youth Hackathon) teilgenommen, welcher von dem Thema Blockchain und SmartContracts handelte. Hackathon ist ein Workshop oder Wettbewerb von Programmierern und Programmiererinnen (zum Lösen einer bestimmten Programmieraufgabe in einem vorgegebenen Zeitraum).[1] Die Organisatoren und Sponsoren dieser Veranstaltung haben verschiedene Workshops und Webinars durchgeführt, an welchen man lernen konnte, wie man einen SmartContract programmiert. Diese Maturaarbeit ermöglicht mir, meine Kenntnisse auf diesem Gebiet zu vertiefen und ein eigenes an einer Blockchain gebundenes Projekt anzugehen und umzusetzen.

Bisherige Situation und Kurzanalyse

Die untenstehende Grafik stellt dar, wie viele Leute 2018 in der Schweiz auf eine Organspende warteten, wie viele davon gestorben sind und wie viele Organe transplantiert wurden. Die Organtransplantation ist ein schwieriger und mehrstufiger Prozess. Ein sehr wichtiger Teil davon ist der Transport der zu transplantierenden Organe. Je nach Situation werden sie mit verschiedenen Verkehrsmitteln geliefert. Meistens sind es allerdings ein Helikopter oder die Ambulanz. Während des Transports befindet sich das Organ in einer Kühlbox oder in einer speziellen Lösung. Je niedriger die Temperatur ist, bei der das Organ transportiert wird, desto stärker wird der Stoffwechsel in den Zellen verlangsamt und desto länger behält das Organ seine Funktion ausserhalb des Körpers. Die Temperatur darf aber nicht tiefer als 4 Grad Celsius sein, weil es sonst zur Bildung von Eiskristallen kommen kann, die die Zellen schädigen würden.

Eine wichtige Rolle nehmen in dieser Branche die Gewebebanken ein. Diese haben zum Ziel, menschliche Gewebe nach der Entnahme aufzubereiten und zu lagern. Die Gewebe müssen auch entweder in einer Konservierungslösung kühl gelagert oder tiefgefroren werden. Auch hier ist es notwendig die Temperatur zu überwachen und in einem gewissen Bereich zu halten.

Ausserdem scheint das ganze System manipulationsanfällig zu sein und um Korruption zu verhindern, muss das System sicherer gemacht werden.

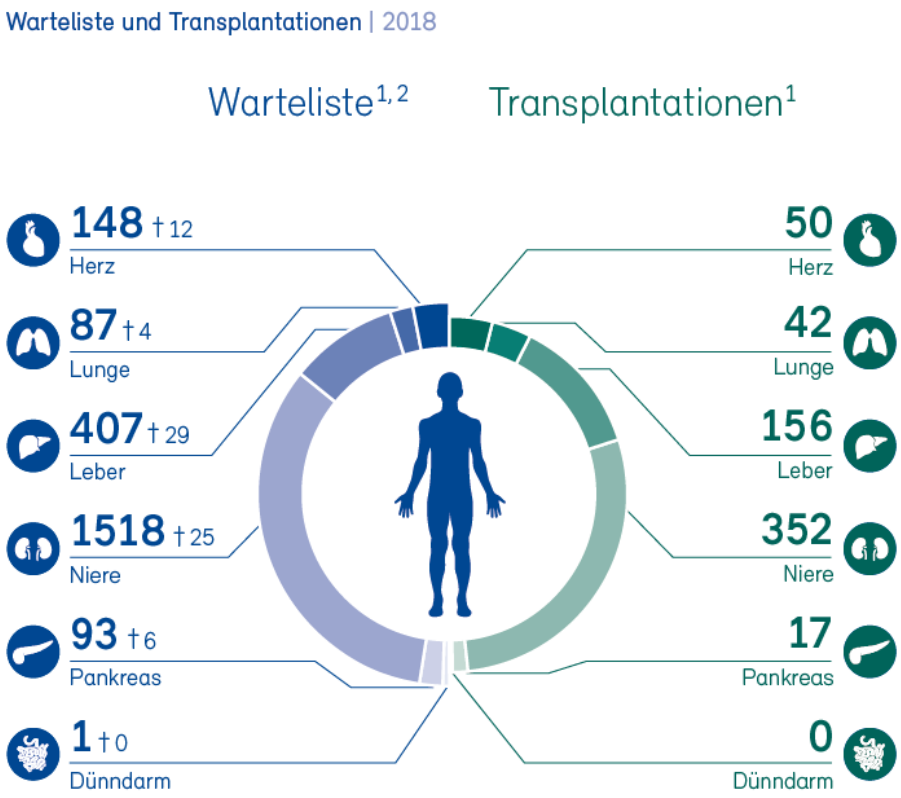


Abb. 1 Warteliste und Transplantationen 2018

Lösungsansatz

Einen möglichen Ansatz bietet die Blockchain an. Diese Idee ist während des Hackathons entworfen worden und wurde dann weiterentwickelt. Die Temperatur sowie die Daten von dem Fingerabdrucksensor werden in der Blockchain gespeichert und können jederzeit abgerufen werden.

Blockchain

Definition

Der Begriff Blockchain (engl. Blockkette) wird auf mehrere Arten verwendet. Blockchain kann als Name für eine Datenstruktur, für einen Algorithmus, für ein Technologiepaket oder auch als Oberbegriff für verteilte Peer-to-Peer-Systeme[1] verstanden werden. Vereinfacht ist die Blockchain eine Kette von Blöcken, wobei die Information in den Blöcken gespeichert wird. Da es eine Kette ist, müssen diese Blöcke miteinander verbunden sein. Jeder Block besteht meistens aus drei Bestandteilen. Diese sind der kryptographisch sichere Hash[2] des vorangehenden Blocks, der Timestamp (engl. Zeitstempel) und eine Information (z. B. bei Kryptowährungen[3] wie Bitcoin[4] setzt sich Information aus den Transaktionsdaten zusammen). Die wichtigste Komponente ist dabei der Hash des vorhergehenden Blocks, weil dadurch die Fälschung der Daten unmöglich wird. Wenn man die Information in einem alten Block verändern würde, würde diese Aktion die ganze nachfolgende Kette verändern.

Die Blockchain ist ein **verteiltes** (oder auch **dezentralisiertes**) Peer-to-Peer-System.Solche Systeme bestehen aus einem Netz aus untereinander verbundenen Komponenten. Dabei gibt es kein zentrales Element, das die Koordinierung oder die Kontrolle übernimmt. Die Einzelteile dieses Systems bezeichnet man als Knoten.

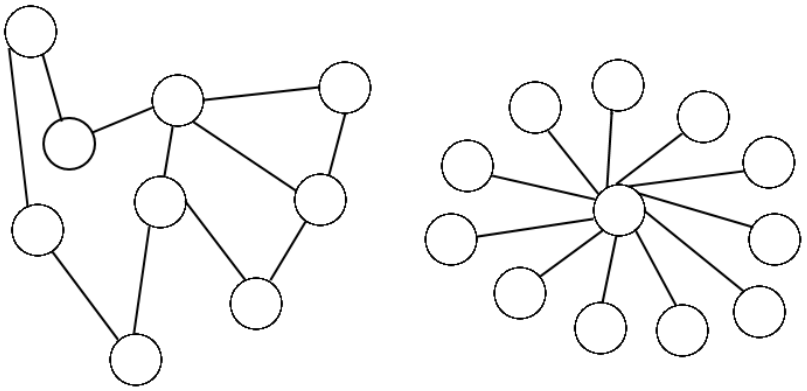


Abb. 2 Verteilte (links) und zentralisierte (rechts) Systemarchitektur

Vor- und Nachteile der Blockchain

### Höhere Rechenleistung Die Rechenleistung eines verteilten Systems entsteht als Summe der Rechenleistung aller verbundenen Elemente (Computer). Daher verfügen verteilte Systeme in der Regel über mehr Rechenleistung als ein einzelner Rechner.	### Höherer Aufwand für die Koordinierung und für die Kommunikation Verteilte Systeme verfügen über keine zentralen Mittel, die für die Koordinierung verantwortlich sind. Die Mitglieder müssen diese Aufgabe selbst übernehmen. Das ist eine Herausforderung, die Geld und Rechenleistung braucht. Ohne Koordinierung ist keine Kommunikation möglich.
### Kostensenkung Die Kosten für die Fertigung, die Wartung und den Betrieb eines Supercomputers liegen deutlich über denen, die für ein verteiltes System nötig sind. Der Austausch einzelner Computer in einem verteilten System ist ohne grosse Auswirkungen auf das Gesamtsystem möglich.	### Abhängigkeit von Netzwerken Ohne Netz gibt es nicht nur keine Kommunikation und keine Koordinierung zwischen den Knoten, sondern es kann überhaupt gar kein verteiltes System entstehen.
### Höhere Zuverlässigkeit In einem verteilten System gibt es keinen «Single Point of Failure» (engl. einzelner Fehlerpunkt). Ein Fehler an einer solchen Stelle kann zum Versagen des Gesamtsystems führen. Wenn in einem verteilten System ein Element ausfällt, übernehmen die verbleibenden Elementen dessen Aufgaben. Deshalb ist ein verteiltes System zuverlässiger als ein Supercomputer.	### Höhere Programmkomplexität Aufgrund der genannten Nachteile muss die Software in der Lage sein, diese zu bekämpfen. Es sind Berechnungsprobleme, Probleme bezüglich der Koordinierung, Kommunikation und Netznutzung zu lösen. Dies führt zu einer höheren Komplexität der Software.
### Fähigkeit zu natürlichem Wachstum Die Rechenleistung eines verteilten Systems ergibt sich durch die kombinierte Rechenleistung all seiner Bestandteile. Deswegen kann sie durch das Einbinden neuer Computer erhöht werden.	### Sicherheitsbelange Die Datenübertragung im Netzwerk muss sicher sein und der missbräuchliche Zugriff von unseriösen Entitäten und unbefugten Dritten muss verhindert werden. Je weniger Beschränkungen es für den Zugriff auf das Netz gibt, desto weniger sicher ist das System bezüglich des verteilten Systems.

Blockchain als Lösung

Die Blockchain wird als Tool genutzt, um Systemintegrität zu erreichen. Die Integrität bedeutet die Unversehrtheit von Daten und der korrekten Funktionsweise von Systemen.[2] Die Integrität wird benötigt, um Anwendern ein vertrauenswürdiges System gewährleisten zu können. Für die Sicherheit verwendet die Blockchain asymmetrische Kryptographie[5] und die kryptographische Hash-Funktion.

SmartContracts

SmartContracts sind elektronische Verträge, die hinterlegte Regeln automatisch überwachen und definierte Aktionen selbsttätig ausführen können. Die Blockchain ermöglicht solche SmartContracts sicherer zu gestalten, weil der Code der SmartContracts dezentral auf vielen Knoten im Netzwerk verteilt ist.[3] Somit können SmartContracts für verschiedene Zwecke eingesetzt werden. Oft braucht man SmartContracts in Bereichen, in denen es einen Middleman[6] gibt. Ein Middleman ist zum Beispiel ein Notar beim Abschliessen eines Vertrags. In einem solchen Fall übernimmt ein SmartContract die Notarfunktion, was die Kosten senkt. Es gibt viele verschiedene Einsatzbereiche für SmartContracts, zum Beispiel bei Wahlen: man stimmt mithilfe eines SmartContracts ab. Ausserdem kann man auch eigene Tokens[7] erstellen und verkaufen, diese können wie Aktien angeschaut werden. So können SmartContracts sowohl Information als auch Geld in sich speichern und damit vorgeschriebene Aktionen selbst ausführen.

Es gibt verschiedene Blockchains, die SmartContracts unterstützen, die bekannteste ist dabei die Ethereum[8] Blockchain.

Wie funktioniert ein SmartContract?

Wenn ein SmartContract initialisiert ist, kann man mit ihm interagieren. Diese Interaktionen können von Contract zu Contract unterschiedlich sein, zum Beispiel man kann in SmartContracts Information speichern, Geld überweisen, abstimmen usw. Jegliche Interaktion, die den Zustand von einem SmartContract verändert, wie zum Beispiel, wenn man eine weitere Information im SmartContract speichern möchte, muss bezahlt werden. Diese Kosten heissen Transaktionskosten (oder in Ethereum auch gas genannt) und werden benötigt, um die Transaktion in einen Block der Blockchain hinzuzufügen. Diese Kommission kann verschieden sein. Will man, dass seine Transaktion schnell in der Blockchain erscheint, kann man eine grössere Menge Geld ausgeben. Dieses Geld bekommen dann Miner, die diesen Block geminet haben.

Miner sind Leute, die Transaktionsdaten in Hashs umwandeln und diese in den Blöcken der Blockchain speichern. Gibt es viele Transaktionen, entsteht eine Warteschlange. Als erstes wird das geminet, was den Minern mehr Gewinn verspricht.

Warum Ethereum?

Ethereum ist die bekannteste Blockchain für SmartContracts, damit ist sie auch sicherer, weil es viele Ethereum-Knoten gibt. Dies macht es sehr schwer, sie zu hacken (dezentrales Netzwerk). Die Erfinder von Ethereum haben für die SmartContracts eine eigene Sprache entwickelt, diese heisst Solidity. Solidity hat eine nutzerfreundliche Syntax und kann sogar in Browser kompiliert werden. Dafür verwendet man Remix IDE (https://remix.ethereum.org). Ein weiterer Vorteil ist, dass auf Solidity geschriebene SmartContracts auch in einem lokalen Netz oder einem Testnetz laufen können. Dies ist sehr praktisch, weil ein solches Netz selbst Accounts freischaltet, welche benötigt werden, um Transaktionskosten zu decken.

Ein anderer Grund, warum ich die Ethereum Blockchain bevorzuge, ist, dass ich schon Erfahrungen mit Solidity habe.

Was auch dafür spricht: Ethereum und somit auch Solidity hat eine sehr freundliche Community mit Foren hat, in welchen man auf Fragen oft sehr schnell eine hilfreiche Antwort bekommt.

Aufbau des Projektes

Funktionsweise

Verschiedene Sensoren sind mit einem Arduino verbunden, welcher seinerseits mit dem Netzwerk verbunden ist. Über dieses Netzwerk werden Datenpakete an einen Server geschickt. Dieser Server läuft lokal im gleichen Netz, in welchem sich auch der Arduino befindet. Auf diesem Server werden die Daten verarbeitet und an einen SmartContract geschickt, welcher eine Verbindung zwischen einem Computer/Server und der Ethereum-Blockchain ermöglicht. Mit einem Webinterface können diese Informationen jederzeit vom SmartContract abgefragt werden.

Voraussichtlicher Aufbau des Modells

Ein Thermometer, ein Fingerabdrucksensor sowie ein Schrittmotor sind mit einem Arduino verbunden. Der Thermometer befindet sich in der Kühlbox, der Schrittmotor ist oben auf der Kühlbox angemacht und soll die Klappe öffnen, sobald der Fingerabdrucksensor einen gültigen Finger erkennt, nach einiger Zeit soll die Klappe wieder geschlossen werden. Der Arduino schickt alle X Sekunden Daten an den Server. Die Daten enthalten die Temperatur und die ID-Nummer der Person, die als letztes die Box geöffnet hat. Nachdem die Daten den Server erreicht haben, werden sie an den SmartContract geschickt.

Ethernet oder WLAN?

Um Sensordaten an einen Server zu schicken, benötigt der Arduino einen Anschluss ans Netz. Es gibt zwei Möglichkeiten, namentlich entweder eine Kabel- oder eine kabellose Verbindung. Für eine Kabelverbindung braucht es ein Ethernet-Kabel und einen Router oder einen Switch und für eine kabellose Verbindung ein WLAN.

Zuerst wollte ich eine Kabelverbindung zwischen dem Arduino und dem Server erstellen. Dafür benötigte ich einen Ethernet-Shield, den man einfach oben in den Arduino einstecken muss. Mithilfe eines Ethernet-Kabels sollte der Arduino mit dem Ethernet-Shield und der Router verbunden werden.

Der Server befindet sich im gleichen Netz und ist über WLAN mit dem Router verbunden. So können Datenpakete von Arduino an den Server geschickt werden.

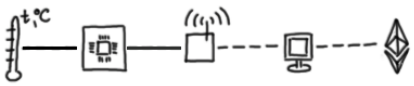


Abb. 3 Schema mit einem Ethernet-Kabel

Dieses Modell hat bei mir zuhause problemlos funktioniert, im Netzwerk des Schulhauses jedoch nicht. Anscheinend gelangt man in ein anderes Netz, wenn man sich über ein Ethernet-Kabel mit dem Schulnetz verbinden will, als wenn man sich mit dem WLAN verbindet.

Deswegen haben wir, mein Betreuer Herr Kai Rollé, Herr Oliver Nellen und ich entschieden, dass die einfachste Lösung wäre, wenn der Arduino über WLAN mit dem Schulnetz verbunden wäre.

Dafür brauche ich einen Arduino Uno Wifi. Jetzt wird der Arduino mit dem WLAN verbunden und damit werden die Sensordaten an den Server geschickt. Nachdem werden sie in einen SmartContract geladen.

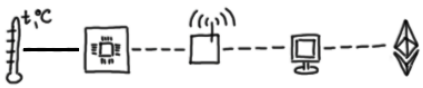


Abb. 4 Schema mit einer kabellosen Verbindung

Verwendete Komponenten

Thermometer

Ich verwende ein Thermometer, der BMP280 heisst. Dieser kann nicht nur die aktuelle Temperatur, sondern auch Drucke messen. Damit er funktioniert, muss man eine Bibliothek in die Arduino IDE importieren, diese heisst Adafruit^{x}_BMP280.h. Die Bibliothek sollte für meinen Sensor angepasst werden, da sie ursprünglich für eine andere Version des Sensors geeignet war. Der Sensor besitzt vier Anschlüsse: SCL, SDA, GND und 3V3. Diese müssen mit den entsprechenden Anschlüssen am Arduino verbunden werden.

Fingerabdrucksensor

Der benötigte Fingerabdrucksensor hat den Name FPM10A. Dieser Sensor kann bis zu 162 verschiedene Fingerabdrücke speichern. FPM10A braucht ebenfalls eine weitere Bibliothek, ihr Name ist Adafruit_Fingerprint.h. Damit der Sensor Finger erkennt, müssen sie zuerst im Sensor gespeichert werden. Dafür benötigt man das Programm «enroll», welches in die Bibliothek integriert ist. Der Fingerabdrucksensor FPM10A hat 6 Anschlüsse, davon werden aber nur 4 gebraucht: GND, RX, TX und 3V3. RX und TX werden mit den Ports 2 und 3 verbunden, diese müssen zuerst im Programmcode als «software serial Ports» definiert werden. Obwohl es einen 3V3 Port hat, was eigentlich für 3,3 Volt steht, funktioniert der Sensor besser, wenn man ihn mit 5V versorgt.

Schrittmotor

Der von mir gebrauchte Schrittmotor heisst 28BYJ-48 und wird für die präzise Drehwinkeleinstellung benötigt. Um diesen Motor mit einem Arduino zu verbinden, braucht man ein Treibermodul. Ich verwende ULN2003. Der Motor mit dem Treibermodul hat 6 Pins, zwei davon sind + und -, müssen also mit GND und 5V verbunden werden. Die anderen 4 Pins können mit dem Arduino beliebig verbunden werden. Ich brauche Ports 8, 9, 10, 11. Für den Schrittmotor brauche ich keine weitere Bibliothek, weil der Testlauf ergeben hat, dass keine der möglichen Variante das von mir gewünschte Ergebnis lieferte.

Arduino Uno

Der Arduino Uno ist ein Open-Source-Mikrocontroller-Board, der auf dem Microchip ATmega328P-Mikrocontroller basiert und von Arduino.cc entwickelt wurde. Das Board ist mit Sets von digitalen und analogen Ein-/Ausgangs-(I/O)-Pins ausgestattet, die mit verschiedenen Erweiterungskarten (Abschirmungen) und anderen Schaltungen verbunden werden können. Das Board verfügt über 14 digitale Pins, 6 analoge Pins und ist mit der Arduino IDE (Integrated Development Environment) über ein USB-Kabel vom Typ B programmierbar. Es kann über das USB-Kabel oder eine externe 9-Volt-Batterie mit Strom versorgt werden, wobei es Spannungen zwischen 7 und 20 Volt akzeptiert.[4]

Arduino Uno Wifi

Der Arduino Uno Wifi ist ein Arduino Uno Mikrocontroller, welcher mit einem Kryptochip-Beschleuniger ECC608 ausgerüstet ist. Dieser ermöglicht die Verbindung mit einem WLAN-Netzwerk.

Arduino IDE

Arduino IDE (Integrated Development Environment) ist die Programmierumgebung für Arduino. Die Programme werden auf einer C/C++ ähnlichen Sprache geschrieben. Gleich wie in C/C++ gibt es eine Möglichkeit, verschiedene Bibliotheken zu importieren. Arduino IDE ist Open-Source-Software und kann gratis von der Webseite von Arduino (https://www.arduino.cc/) oder auch aus dem Microsoft Store heruntergeladen werden.

Ubuntu und Nginx

Ubuntu ist eine Linux-Distribution und damit ein Open-Source Betriebssystem.

Nginx ist eine Webserver-Software für auf Unix-Kern (somit auch Linux) basierte Betriebssysteme.

TestRPC

Ethereum TestRPC ist ein schneller und anpassbarer Blockchain-Emulator. TestRPC stellt einen lokalen Ethereum-Knoten und Accounts zur Deckung der Transaktionskosten zur Verfügung.

Programmiersprachen

Für meine Maturaarbeit verwende ich mehrere Programmiersprachen.

Die Programmiersprache von Arduino wird benötigt, um den Mikrocontroller und die damit verbundenen Sensoren zu steuern.

PHP (Hypertext Preprocessor oder Personal Home Page) ist eine Skriptsprache, die zur Erstellung dynamischer Webseiten oder Webanwendungen verwendet wird. Wie die Definition schon sagt, brauche ich diese Sprache zur Erstellung der Serverseite meines Projekts.

JavaScript ist eine Skriptsprache, mit der die Interaktionen von Benutzern in Webbrowser ausgewertet werden können. JavaScript benötige ich, um Webinterface zu gestalten. JavaScript besitzt viele Bibliotheken, die das Programmieren erleichtern und den Einsatzbereich erweitern. Die Bibliothek Web3.js ermöglicht es, eine Verbindung mit einem SmartContract zu bilden. Die Bibliothek jQuery vereinfacht den Programmierungsprozess mit JavaScript und HTML.

HTML steht kurz für Hypertext Markup Langugage und ist eine Auszeichnungssprache. HTML ist verantwortlich für die inhaltliche und strukturelle Bedeutung der Elemente.

Mithilfe der **Cascading Style Sheets** oder kurz **CSS** werden die Grundformatierung und das Aussehen der HTML-Elemente mit Gestaltungsanweisungen festgelegt.

Solidity ist eine objektorientierte, anwendungsspezifische höhere Programmiersprache, die benötigt wird, um SmartContracts für Blockchain-Plattformen wie Ethereum oder Tron zu entwickeln.

Praktischer Teil

Basteln des Modells der Kühlbox

Als Material für mein Modell habe ich mich für Lego® entschieden. Lego® hat viele verschiedene Teile, die man brauchen kann, und Lego® ist fast in jedem Spielzeugladen erhältlich. Fürs Basteln mit Lego wird kein Leim benötigt und somit kann man verschiedene Konstruktionen ausprobieren, ohne Material zu verschwenden. Nachdem das Modell fertig war, wurde es angeschliffen und mit Farbe besprüht. Vor dem Besprühen wurden die Vorderseite und die linke Seite mit Malerband beklebt, damit es Figuren aufgesprüht werden können. Auf der Vorderseite befindet sich das Ethereum-Logo und auf der linken Seite ist ein Herz abgebildet.

Abb. 5 Das Ethereum-Logo

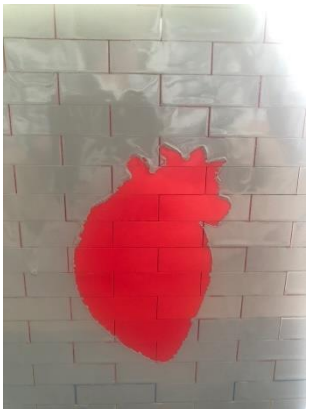


Abb. 6 Das Herz

Erstellung des SmartContracts

Für das Programmieren des SmartContracts benötigt man zwei Dinge, nämlich eine Programmierumgebung und ein Ethereum-Testnetz. Als Programmierumgebung habe ich die Remix IDE gewählt, weil man dafür nur einen Browser braucht und keine zusätzliche Software installieren muss. Das Testnetz ist TestRPC und es lässt sich sehr einfach durch die Kommandozeile installieren (die Installation benötigt Node.js[x]).

```
npm install -g ethereumjs-testrpc
```

Nachdem TestRPC installiert ist, kann man es mit diesem einfachen Befehl starten:

```
testrpc
```

In der Konsole erscheint eine solche Meldung:

```
Available Accounts
==================
(0) 0x8fe4a0b763b8ce469432339bf609cbf21fe851c7
(1) 0x84d376aaffcf1b425f83a63cf78ae01d22a8d969
(2) 0x65ddacd8ebb2af135462acf005c7944007b094bb
(3) 0x8409f44ca39b1b99bcd343e04e861afbc9035079
(4) 0x3f3aa53b695b5a37031b5e168a95e8dd075f7727
(5) 0x01d11a38ce916819f49c1ac651cdd3ecd6d575d6
(6) 0x273e57f31dea931a87e4a3181ca1522b3b321e48
(7) 0xeca6ef0453d95df33c33fcabe7992ecdf37b2094
(8) 0xca94d2e711fe48489af157fda4b6dd515b337e8a
(9) 0xdb8439ca212375a0109a12da720ef3f74d3fa44b

Private Keys
==================
(0) 5f8d2a1b4e36925cb311f1f08339c69cb1adb59d8e61f87a9aa73d1c0044c524
(1) d8e208c44b32bbb4b705cc0a8632930ece728deb6028fb30769e14db06dca8de
(2) 47951a775b6c4b7faeb144158737dc4eb36ea4547597e8db7ca8dbd9e36ad7f1
(3) 5e3913d4a3c6390660e1b1175a73e7351ec24a6035f8aed2971449c6d30f74f8
(4) ca7cbe8faaec315b5290c2a8fe93e00e33f6baefe402e8b5d07522a9501a5643
(5) 1ab6dd303b5f6bc76d20e672300b7568523801df6e7fd16d86492fbf80dd4171
(6) 7fde5252559d1fc7b688cac24962dc76ae756b97e0894f393fd63eaf565d7faa
(7) 74d02a3aaccd97688282a068f1cbc4edbe4b0f97a476e2570a504aca2eba96d1
(8) f301bd205336a2b2c505f9143183b3cecf64aae8d5839a63a37a3421bb0d6b1d
(9) ff69f1b6f431a48ae9ae7624200c1533e1234995279a5309b53ed0d445b07630

HD Wallet
==================
Mnemonic:      detect urban lobster siren crazy parade treat defense scheme master track eye
Base HD Path:  m/44'/60'/0'/0/{account_index}

Listening on localhost:8545
```

Abb. 7 Die von TestRPC zur Verfügung gestellten Accounts

Oben sieht man die 10 Accounts, die TestRPC zur Verfügung stellt. Diese werden benötigt, um die Transaktionskosten des SmartContracts zu decken. Unten erkennt man, dass das Testnetz am Port 8545 angeschlossen ist.

Wenn man TestRPC gestartet hat, muss man in der Remix IDE die Umgebung auf Web3 Provider wechseln und den Port des lokalen Testnetzes eingeben.

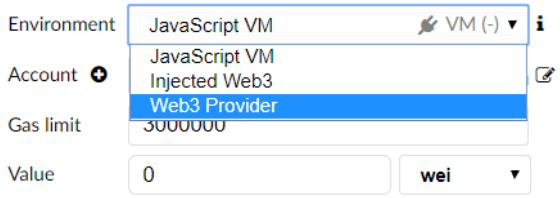


Abb. 8 Wechsel der Umgebung in der Remix IDE

Abb. 9 Eingabe des Ports des lokalen Testnetz in der Remix IDE

Erstellung des Servers und des Webinterfaces

Für die Erstellung des Servers brauche ich eine virtuelle Maschine[x] mit dem Ubuntu-Linux-Betriebssystem. Als nächstes wird Nginx installiert.

```
sudo apt-get update
sudo apt-get install nginx
```

Im Ordner /var/www/html werden die Serverdateien erstellt. Die Programmiersprache ist dabei PHP. Daher ist die Serverdatei eine name.php Datei. In diesen Dateien wird beschrieben, was der Server machen muss. Es wird auch ein Webinterface mithilfe von PHP, AJAX, html, CSS und JavaScript erstellt. Damit man auf das Webinterface zugreifen kann, müssen die Namen der Serverdateien in /etc/nginx/sites-available eingetragen werden.

Programmieren und Verbinden von Arduino und der dazugehörigen Sensoren

Der Arduino Uno Wifi unterstützt die Adafruit_Fingerprint.h. Bibliothek nicht und weswegen ich einen zweiten Arduino benötige. Dieser wird mit dem Fingerabdrucksensor, dem Schrittmotor sowie mit dem Arduino Wifi verbunden. Sobald der Fingerabdrucksensor einen Finger erkennt, schickt der Arduino ein Signal dem Arduino Wifi und dieses wird dann schliesslich an den Server geschickt. Der Schrittmotor sowie der Fingerabdrucksensor benötigen beide 5V (der Schrittmotor kann auch mehr brauchen), diese Aufgabe schafft der Arduino als Energiequelle nur knapp, daher verwende ich eine 9V Batterie, um den Schrittmotor mit Strom zu versorgen.

Der Arduino Uno Wifi ist mit dem Thermometer und auch mit dem WLAN verbunden.

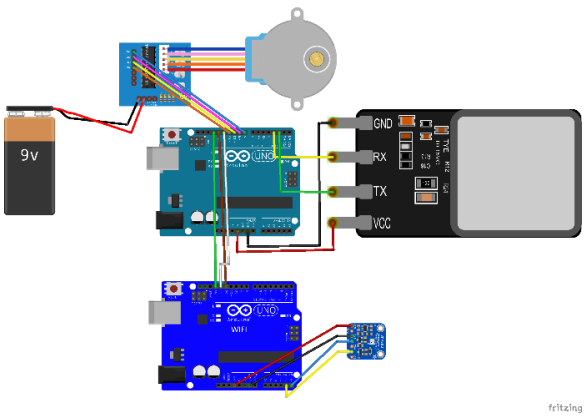


Abb. 10 Das Verbindungsschema von Arduinos und Sensoren

Verbindung Arduino – Server erstellen

Die Daten von Arduino werden über WLAN übertragen. Die Kommunikation erfolgt mithilfe eines GET Requests. Die Daten werden an die Datei add_data.php geschickt.

```
client.print( "GET /add_data.php?");
client.print("temperature=");
client.print( bmp.readTemperature() );
client.print("&");
client.print("&");
client.print("ID=");
client.print( finger.fingerID );
client.println( " HTTP/1.1");
client.print( "Host: " );
client.println(server);
client.println( "Connection: close" );
client.println();
client.println();
```

Abb. 11 Die Codesequenz des Arduinoprogramms, welche für die Kommunikation mit dem Server verantwortlich ist

Die Datei add_data.php seinerseits schreibt die erhaltene Information in Dateien in-1.txt und in-2.txt hinein.

add_data.php

```
<?php
        $S1 =  $_GET['temperature'];
        $myFile1 = "txt/in-1.txt";
        $fh1 = fopen($myFile1, 'w') or die("can't open file");
        fwrite($fh1, $S1);
        fclose($fh1);

        $S3 =  $_GET['ID'];
        $myFile3 = "txt/in-2.txt";
        $fh3 = fopen($myFile3, 'w') or die("can't open file");
        fwrite($fh3, $S3);
        fclose($fh3);


?>
```

Abb. 12 Die add_data.php Datei

Verbindung Webinterface – SmartContract erstellen

Dieses Webinterface muss eine Verbindung zum SmartContract erstellen können, dafür benötigt man die Web3.js Bibliothek. Jeder SmartContract hat seine unikale Adresse, diese findet man in der Remix IDE, nachdem der SmartContract deployed ist.

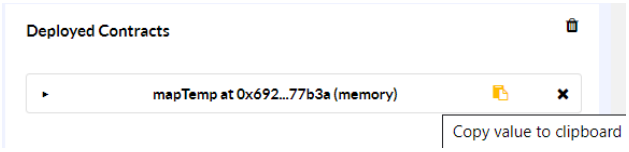


Abb. 13 Die Adresse des SmartContracts

Neben der Adresse besitzt jeder SmartContract eine ABI[9], diese ist in der Remix IDE unter Compile zu finden.

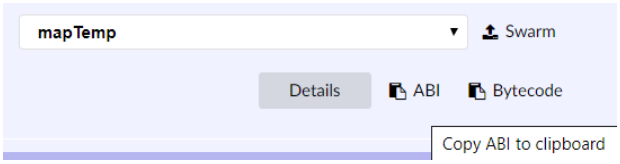


Abb. 14Die ABI des SmartContracts

Die beiden Werte setzt man in eine JavaScript-Codesequenz in der Webinterfacedatei ein. In den Zeilen 93-176 befindet sich die ABI (aufgrund der Länge ist diese hier nicht abgebildet), in Zeile 178 findet man die Adresse von dem SmartContract.

```
<script>
        if (typeof web3 !== 'undefined') {
          web3 = new Web3(web3.currentProvider);
        } else {

          web3 = new Web3(new Web3.providers.HttpProvider("http://localhost:8545"));
        }
        web3.eth.defaultAccount = web3.eth.accounts[3];

        var DataBaseContract = web3.eth.contract([
{…
},
{…
},
{…
},
{…
},
{…
}
]);
    var DataBase = DataBaseContract.at('0xcf6a25a9d40a27065e5a6b18b65c33dfc79dbf56');
```

Abb. 15 Die Adresse und die ABI des SmartContracts in der Webinterfacedatei

Die AJAX-Skripte holen die von dem Arduino erhaltene Daten mithilfe der Dateien temp-1.php und id-1.php.

```
        $.ajax({
            url: "transfer/temp-1.php",
            cache: false,
            async: false,
            success: function(html){
                $("#content").html(html);
        $temp=html;

            }
         });
        $.ajax({
            url: "transfer/id-1.php",
            cache: false,
            async: false,
            success: function(html){
                $("#content-1").html(html);
        $id=html;
            }
         });
```

Abb. 16 Die AJAX-Skripte

Die Dateien temp-1.php und id-1.php ihrerseits lesen die Daten aus bereits erwähnten Dateien in-1.txt und in-2.txt.

temp-1.php

```
<?php
$myFile = "../txt/in-1.txt";
$fh = fopen($myFile, 'r');
$theData1 = fread($fh, filesize($myFile));
fclose($fh);
echo $theData1;
?>
```

id-1.php

```
<?php
$myFile = "../txt/in-2.txt";
$fh = fopen($myFile, 'r');
$theData = fread($fh, filesize($myFile));
fclose($fh);
echo $theData;
?>
```

Abb. 17 Die temp-1.php und die id-1.php Dateien

Diese Information kann jetzt im Webinterface angezeigt und dem SmartContract zugeschickt werden. Das Schicken erfolgt mithilfe einer JavaScript-Codesequenz, das heisst, um neue Information zu schicken, muss man die Seite jeweils aktualisieren. Eine Information besteht dabei aus der gemessenen Temperatur, dem Zeitstempel und der ID-Nummer der letzten Person, die die Kühlbox geöffnet hat.

```
191        DataBase.setTemp($inp);
```

Abb. 8 Befehl, der Information an den SmartContract schickt

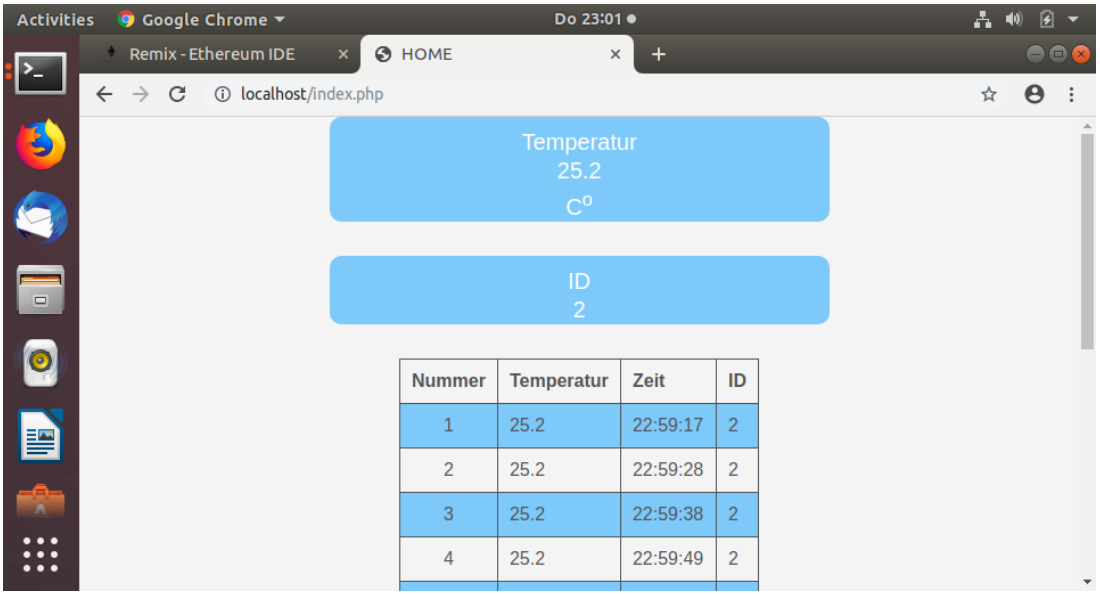
```
195        $l = DataBase.getCount();
```

Abb. 19 Befehl, der Information von dem SmartContract holt

```
225    <meta http-equiv="refresh" content="10">
```

Abb. 20Befehl, der die Seite jede 10 Sekunden aktualisiert

Nachdem die Information geschickt wurde, erscheint sie auf der Seite in einer Tabelle.



Nummer	Temperatur	Zeit	ID
1	25.2	22:59:17	2
2	25.2	22:59:28	2
3	25.2	22:59:38	2
4	25.2	22:59:49	2

Abb. 21 Das Webinterface funktioniert

Fazit

Ich habe gelernt ein grösseres Projekt über längere Zeit zu planen und dieses in einzelne Teile zu unterteilen. Alle Teile meiner Maturaarbeit mussten separat entwickelt werden und erst nachdem alles fertiggestellt und getestet wurde, konnten sie miteinander verbunden werden. Ich habe gelernt, wie man Probleme in kurzer Zeit lösen kann und wie man Alternativen finden kann. Ich wurde mit mehreren Problemen konfrontiert, wie zum Beispiel die Umstellung auf einen anderen Arduino, die Bearbeitung einer Bibliothek (beim Thermometer) oder die lange Suche nach einer Alternative (beim Schrittmotor). Ich habe den Aufwand unterschätzt, sich selbstständig eine neue Programmiersprache anzueignen. Obwohl ich Solidity, JavaScript, HTML und die Arduino-Sprache schon kannte, waren PHP, AJAX und CSS neu für mich. Das Thema Blockchain ist modern und anspruchsvoll. Man muss viel recherchieren und nachlesen, um zu verstehen, wie das ganze System funktioniert. Jedoch gefällt mir dieses Thema sehr und ich möchte meine Kenntnisse über die Blockchain weiter vertiefen.

In dieser Arbeit konnte ich aufzeigen, wie man einen Arduino mit einem SmartContract über einen Server verbinden kann. Das Modell funktioniert und alle Programme können ausgeführt werden. Nach Abschluss dieser Arbeit kann ich feststellen, dass ich meine Ziele erreicht habe.

Danksagung

Ich danke herzlich meinem Betreuer, Herrn Kai Rollé und meiner Familie für die Betreuung und Unterstützung bei meiner Arbeit. Ich möchte mich auch bei Oliver Bähler und Manuel Baumann bedanken, mit welchen ich beim Hackathon in einem Team zusammenarbeiten durfte. Ganz herzlich bedanke ich mich zudem bei meiner Kollegin Vera Vasilieva für Zeichungen, die sie für mich gemacht hat und auch bei meinem Kollegen Linus Crugnola für die Korrektur einiger Sprachfehler.

Glossar

[1]Peer-to-Peer-Systeme sind verteilte Softwaresysteme, die aus Knoten (Einzelcomputer) bestehen, welche sich ihre Berechnungsressourcen (Verarbeitungsleistung, Speicherkapazität oder Informationsverteilung) direkt gegenseitig zur Verfügung stellen. Alle Knoten haben dabei dieselben Rechte und Rollen.

[2]Hash; Hashfunktionen sind kleine Computerprogramme, die beliebige Daten ungeachtet des Umfangs der Eingabedaten auf eine Zahl mit fester Länge abbilden. Die dabei entstehende Zahl heisst Hash. Die Hashfunktionen sind Einwegfunktionen, das heisst, sie sind unumkehrbar. Den Hash kann man mit einem Fingerabdruck vergleichen: jede Zeichenkette hat einen einzigartigen Hash und es ist sehr einfach den Hash einer Zeichenkette zu ermitteln und fast unmöglich die ursprüngliche Zeichenkette aus dem Hash wiederherzustellen.

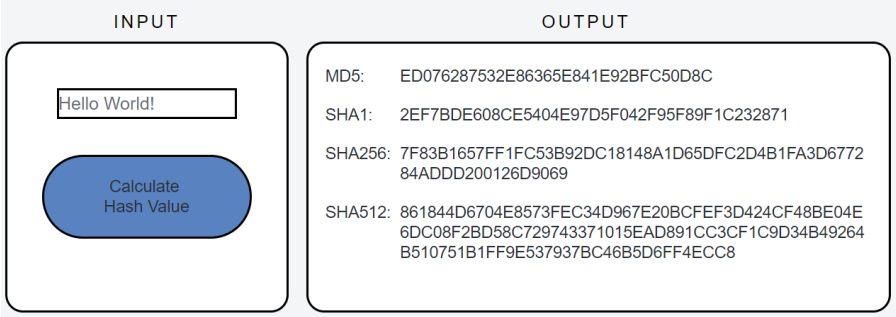


Abb. 22 Beispiel für einen Hash

[3] Eine Kryptowährung ist ein digitales Zahlungsmittel, das auf kryptographischen Werkzeugen wie der Blockchain und digitalen Signaturen basiert.

[4]Bitcoin ist die erste Kryptowährung (am 3. Januar 2009 wurde der erste Block in der Bitcoin-Blockchain generiert).

[5]Kryptographie ist die Wissenschaft, die sich mit der Verschlüsselung beschäftigt. Bei der asymmetrischen Kryptographie kommt ein Schlüsselpaar zum Einsatz. Ein mit einem der beiden Schlüssel erzeugter Geheimtext kann nur mit dem jeweils anderen Schlüssel entschlüsselt werden und umgekehrt. (einer davon ist der Privatschlüssel und der andere der öffentliche Schlüssel).

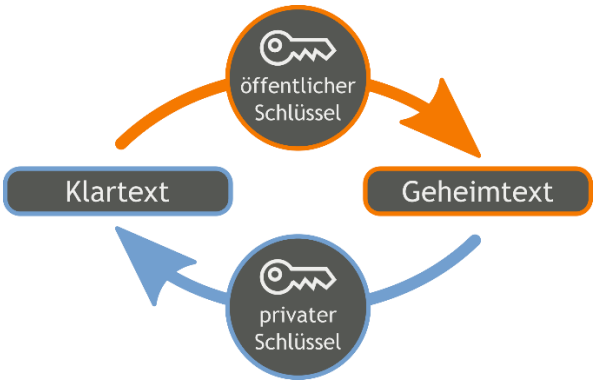


Abb. 23 Die Nachricht wird mit dem öffentlichen Schlüssel verschlüsselt und mit dem privaten entschlüsselt

[6]Middleman ist eine Zwischeninstanz, ein unabhängiger Vermittler zwischen zwei Parteien

[7]Tokens sind digitale Münzen, die für Bezahlungen geeinigt sind

[8] Ethereum ist ein quelloffenes verteiltes System, welches das Anlegen, Verwalten und Ausführen von dezentralen Programmen bzw. Kontrakten (SmartContracts) in einer eigenen Blockchain anbietet.

Ethereum verwendet die interne Kryptowährung Ether als Zahlungsmittel für Transaktionsverarbeitungen, welche durch teilnehmende Computer abgewickelt werden.

Ether ist Stand Januar 2019 nach Bitcoin die Kryptowährung mit der zweitgrössten Marktkapitalisierung[5]

[9] ABI (eng. application binary interface) ist eine Schnittstelle zwischen zwei Computerprogrammen auf Maschinenebene.

Literatur- und Quellenverzeichnis

Abbildungen

Das Bild auf dem Titelblatt: Vasilieva Vera, 09.10.2019

Abb. 1: https://www.swisstransplant.org/fileadmin/user_upload/Swisstransplant/Jahresbericht/Jahresbericht_und_Grafiken_2018/2018_Warteliste_und_Transplantationen.pdf, 17.02.2019

Abb. 2: Selbstgemacht

Abb. 3: Vasilieva Vera, 09.10.2019

Abb. 4: Vasilieva Vera, 09.10.2019

Abb. 5: Selbstgemacht

Abb. 6: Selbstgemacht

Abb. 7: Selbstgemacht

Abb. 8: Selbstgemacht

Abb. 9: Selbstgemacht

Abb. 10: Selbstgemacht mithilfe der Fritzing-App

Abb. 11: Selbstgemacht

Abb. 12: Selbstgemacht

Abb. 13: Selbstgemacht

Abb. 14: Selbstgemacht

Abb. 15: Selbstgemacht

Abb. 16: Selbstgemacht

Abb. 17: Selbstgemacht

Abb. 18: Selbstgemacht

Abb. 19: Selbstgemacht

Abb. 20: Selbstgemacht

Abb. 21: Selbstgemacht

Abb. 22: Selbstgemacht mithilfe http://www.blockchain-basics.com/HashFunctions.html, 23.06.2019

Abb. 23: https://www.linux-community.de/ausgaben/easylinux/2015/03/private-daten-schuetzen/, 11.10.2019

Abb. 24: Selbstgemacht

Abb. 25: Selbstgemacht

Abb. 26: Selbstgemacht

Abb. 27: Selbstgemacht

Abb. 28: Selbstgemacht

Gedruckte Literatur:

DRESCHER, D. 2017: Blockchain Grundlagen.

FERTIG, T. et al. 2019: Blockchain für Entwickler.

STIEGERT, H. 2008: CSS-Design. Die Tutorials für Einsteiger.

KAPPEL, B 2016: Arduino. Elektronik, Programmierung, Basteln.

Internetliteratur

ivizil (Ivan): Простое управление arduino через интернет (Einfache Steuerung von Arduino durch Internet), 01.05.2015, https://habr.com/ru/post/257115/, 24.04.2019

EquinoX, 04.03.2011,https://serverfault.com/questions/243109/cant-access-nginx-server-from-ip, 25.08.2019

Wikipedia,07.09.2019, https://de.wikipedia.org/wiki/Ethereum, 10.09.2019

Wikipedia, 25.09.2019, https://de.wikipedia.org/wiki/Bitcoin, 01.10.2019

https://www.coindesk.com/information/ethereum-smart-contracts-work, 5.10.2019

Internetvideos und Online Kurse

Ravaei, Niloo: CE101: Introduction To Cryptoeconomics, https://courses.blockgeeks.com/course/ce101-intro-to-cryptoeconomics/, 03.02.2019

Wu, Jack: Blockgeeks, https://courses.blockgeeks.com/course/workshop-seriesjavascript-primer/, 04.02.2019

Rabbani, Haseeb: ETH101: Intro to Ethereum, https://courses.blockgeeks.com/course/bg101-ethereum-course-101/, 05.02.2019

Lapotkov, Jan: YouTube, 22.10.2017 https://www.youtube.com/watch?v=00CJAVkBlJE&t=1s, 10.02.2019

Quellenverzeichnis

[1] Duden, https://www.duden.de/rechtschreibung/Hackathon, 10.10.2019

[2] Wikipedia, 24.08.2018, https://de.wikipedia.org/wiki/Integrität_(Informationssicherheit), 10.10.2019

[3] Gabler Wirtschaftslexikon, Prof. Dr. Mitschele, A, https://wirtschaftslexikon.gabler.de/definition/smart-contract-54213, 05.10.2019

[4] Wikipedia, 24.09.2019, https://en.wikipedia.org/wiki/Arduino, 01.10.2019

[5] Wikipedia,07.09.2019, https://de.wikipedia.org/wiki/Ethereum, 10.09.2019

Anhang

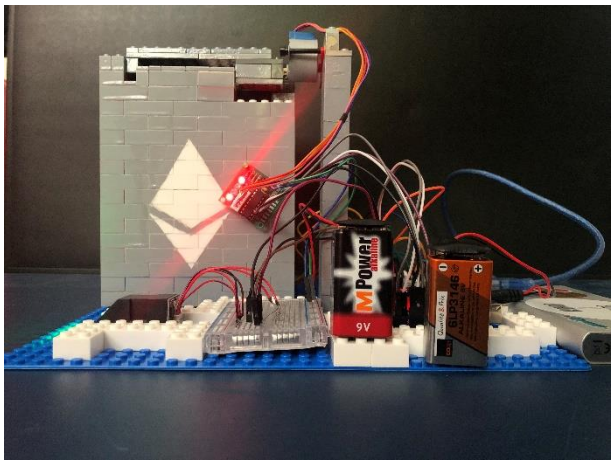


Abb. 24 Die Vorderseite

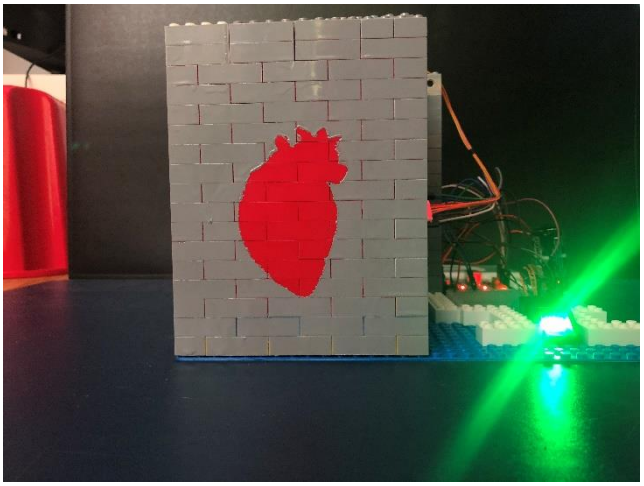


Abb. 25 Die linke Seite

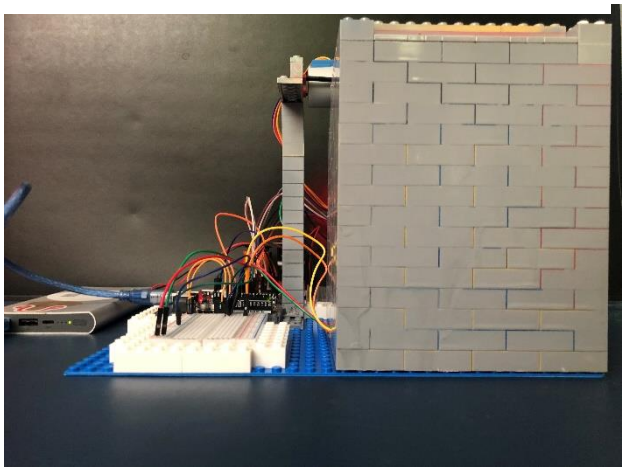


Abb. 26 Die Hinterseite

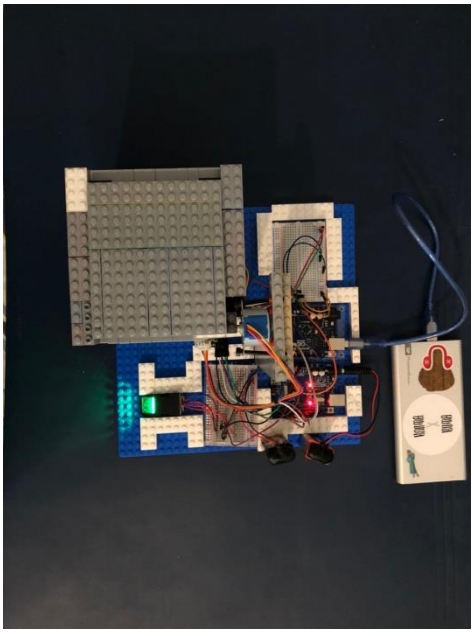


Abb. 27 Von oben

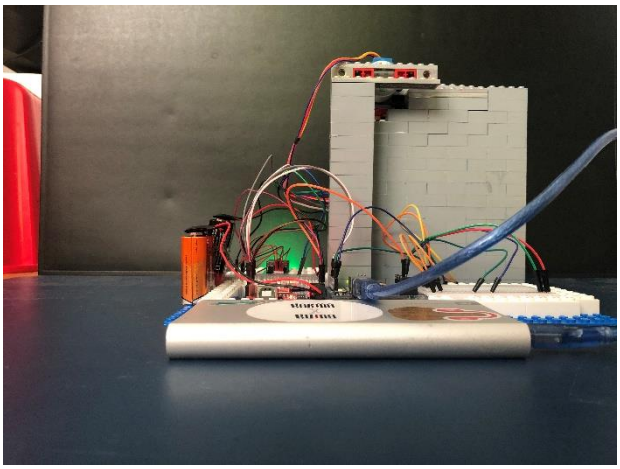


Abb. 28 Die rechte Seite