Blockchain Based Machine-to-Machine (M2M) Communication and Digital Twins

Mohammad Ghanem, Wolfgang Prinz
RWTH Aachen University, Templergraben 55, 52062 Aachen
Fraunhofer FIT, Schloss Birlinghoven, 53754 Sankt Augustin

Over the last two decades, the rapid advances in digitization methods put us on the fourth industrial era's cusp. It is an era of connectivity and interactivity between various industrial processes that need a new, trusted environment to exchange and share information and data without relying on third parties. Blockchain technologies can provide such a trusted environment. This paper focuses on utilizing the blockchain with its characteristics to build machine-to-machine (M2M) communication and digital twin solutions. We propose a conceptual design for a system that uses smart contracts to construct digital twins for machines and products and executes manufacturing processes inside the blockchain. Our solution also employs the decentralized identifiers standard (DIDs) to provide self-sovereign digital identities for machines and products. To validate the approach and demonstrate its applicability, the paper presents an actual implementation of the proposed design to a simulated case study done with the help of Fischertechnik factory model.

1. Introduction

Until today, the industry has seen three major revolutions, and the fourth is on its way [1] . The fourth industrial revolution (Industry 4.0) represents the next step in the evolution of traditional factories towards smart, automated factories. These factories are designed to reduce production costs, increase productivity, improve quality, and achieve efficient use of resources. Many technologies can be used to achieve the industry 4.0 goals like Robotics, Autonomous Systems, the Internet of Things, Cloud Computing, Intelligent Data Analytics, Artificial Intelligence, and many more [2]. However, all these technologies rely on centralized networks and need to trust intermediaries or third-party operators [3]–[7]. As a result, the industry faces many challenges related to the data like transparency, security, privacy, and trustworthiness. These challenges prevent Industry 4.0 from reaching its full potential. A decentralized and trusted platform is needed to facilitate the relationships among parties. Such a platform can be built with the help of blockchain technologies.

Given its key features such as immutability, traceability, and reliability, it represents a perfect candidate to be integrated into Industry 4.0 factories. This paper aims to shed light on the blockchain's capability to improve the manufacturing industry and understand how blockchain can work with other technologies to overcome the earlier challenges. In particular, the paper focuses on two aspects of manufacturing. The first one is the communication between machines to enable the concept of machine-to-machine (M2M) communication over the blockchain, where one machine can ask another machine to perform a particular task without human involvement. The second one is tracking and tracing the machines and products while executing the manufacturing processes and building a digital representation with the blockchain's help. Our work will also show how this technology can provide a blockchain-based digital identity for both the machines and the products, by applying emerging standards in this area: the Decentralized Identifiers (DIDs) and Verifiable Credentials (VCs) [8], [9].

The fine-grained objectives are the following:

- Build a digital twin of the machine using the blockchain. The digital twin should include information about the machine's functions and operations like tasks, sensor readings, alerts. It also should reflect the status of the machine.
- Build a digital twin of the product using the blockchain. The product's digital twin should include information about the operations performed on the product and related information.
- Provide a blockchain-based digital identity for the digital twins.
- Model the manufacturing process and its business logic using the blockchain. In other words, make all the communication between the machines go through the blockchain.
- The design should be generic and replicable to different use cases.

The remainder of the paper is structured as follows. Following the introduction section, we present the related work, which includes the literature review. Section 3 gives a conceptual design and modeling for the solution. Then section 4 proceeds by explaining the conceptual design's implementation details. Besides, it provides information about the case study used and its prototype. Section 5 presents the results with some screenshots of the final prototype. The last section summarizes the finding and discusses future work.

2. Related Work

There has been an increased interest in applying blockchain in the manufacturing industry in the past few years. We found two groups of work done in this area, and both are utilizing the blockchain in industrial applications. The first group focused on horizontal integration between manufacturing parties to enable manufacturing as a service between manufacturers themselves or between manufacturers and customers. For example, in [3], the goal is to build a trustless distributed network. Where different industrial organizations can collaborate and share information about manufacturing processes. The network was built using blockchain and smart contracts technologies. They stored information about manufacturers, machines, and their capabilities. A participant of this network can be a human, manufacturing machine, computing node, or an agent representing any organization.

Another work in the same direction is done in [5]. The authors integrated cloud manufacturing technologies with blockchain. The work proposed a distributed peer-to-peer network architecture to improve manufacturing cloud platforms' security and scalability. They used smart contracts to write the rules of the agreement between the end-users and the service providers. These rules contain the due date, quality measurement, and payment information. A similar approach can be found in [10]. The goal of this work is to improve communication between manufacturing service users and manufacturing services providers. A use case in the 3D printing manufacturing industry has been conducted, and the results showed that blockchain technologies could help solve some existing problems found in cloud manufacturing literature. In all these works that focused on horizontal integration, the authors neglected the actual manufacturing processes and focused more on the concept of trading using smart contracts. In other words, they did not consider what is happening inside the factories.

The second group focused on manufacturing processes by enabling M2M communication over the blockchain. One of the first research works that used the M2M concept with blockchain was done in [4]. The authors explored the applications of blockchain with Industry 4.0. They built a proof of concept where a blockchain is used to facilitate the interaction between machines. The goal is to enable the M2M electricity market, where industrial plants autonomously trade electricity over a blockchain. The agreement between the producer and the consumer is built using smart contracts. The information about the energy consumption (in kWh) published by the machines is stored as transactions in the blockchain. Each transaction has a fee (in USD) according to the agreement specified by the smart contract.

A similar approach has been followed in [7]. The authors focused on industrial M2M communication and how blockchain technologies can improve it. They introduced smart contract-based middleware for M2M communication to make it secure and decentralized. Through this middleware, IoT devices can communicate without the need for a trusted intermediary. The middleware controls and executes contracts to order tasks from field devices. Also, it monitors the field devices' states and executes actions based on a change in their state. It may also request a field device to perform service under a smart contract. All information about the actions and processes is recorded in the blockchain through smart contracts. This work emphasizes the real-time requirement for M2M communication in industrial processes. The result showed that smart contracts technology is still not mature enough to provide such an essential requirement.

Overall, in the second group of related work, the use cases were oversimplified and, in most cases, limited in size to only two machines. This does not reflect the actual communications between machines on the production line. They did not mention how the machines are being modeled in their systems, and this is an essential aspect of M2M-based systems because it will help build a fully autonomous manufacturing process. Also, none of the works discussed how the products are being modeled within the blockchain.

Another essential aspect is the identity management of the machines and the products. Each machine needs to know and identify other machines before establishing the communication. All the works we discussed used only the public/private key pairs as identities. This approach has many limitations and problems [11]. It makes the identity tightly coupled with the algorithm used to generate the key pair. None of the work spouted the issue of managing the digital identities of the machines or the products.

3. Conceptual Design

This section presents the conceptual design for an envisioned system that utilizes the blockchain to build M2M communication and digital twins solutions. The system is functioning alongside the existing infrastructure of the factory. It uses the blockchain to store and manage the data generated by the factory infrastructure. The data stored in the blockchain is used to build a digital representation of the machines and the products. Furthermore, all the communications between the machines go through the blockchain. Therefore, the system consists of the following three components:

- **Factory Infrastructure:** It represents all the existing hardware and software of the factory. It includes machines, sensors, and other devices. It also includes a client application that connects the factory infrastructure with the blockchain.
- **Blockchain:** It is the data storage and computational component of the system. It is the network of all the nodes processing the transactions and running smart contracts.

- **Web Application:** It is the application used by different actors to access the information stored in the blockchain. It consists of a front-end application and a blockchain client application.

The following figure shows a high-level diagram of the system components and the data flows between them.

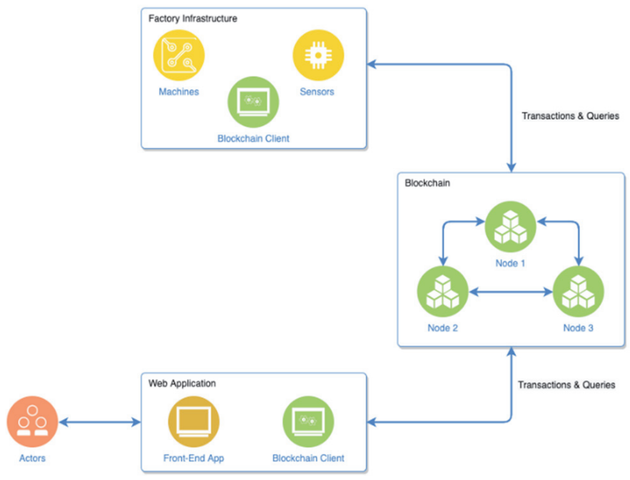


Fig. 1: High-Level System Overview

3.1. Machine Modeling

The machine is the main and the most crucial entity in the system. Our modeling is generic and can be applied to any machine. We use the term machine to refer to any factory component, including machines and robots of all sizes. We assume that the machine already has its digital representation provided by its manufacturer or third-party software. Each machine can perform several industrial tasks, and several processes can use the machine. There are no restrictions on the size and the complexity of the machine or its tasks.

3.1.1. Machine Digital Twin

We decided to model the machine as a smart contract to build the machine's digital twin in our system. Each machine will have a corresponding smart contract deployed into the blockchain. The smart contract with all the information stored within it represents the machine's digital twin created by the blockchain. Once the smart contract updates itself to include new information about the machine, it will be part of its digital twin, and it cannot be altered or changed. The ultimate goal when building a digital twin is to make a replica of the physical entity. However, we decided to limit the information included in the machine's digital twin to:

- **Identity:** The identity of the machine. More about the identity in the following section.
- **Basic Information:** Static information about the machine like the serial number, the model, manufacturing year, and similar info. Only the machine owner can provide such information, and once it is added to the digital twin, it cannot be changed.
- **Processes:** Information about the processes which use the machine. The authorized processes are allowed to assign tasks to the digital twin of the machine. The machine owner provides this information.
- **Tasks:** Information about the machine's tasks. It involves information about the starting time, finishing time, the parameters, the process, and the product.
- **Readings:** Any numeric information coming from the machine sensors like temperature or humidity. This information is sent by the physical machine and stored in the digital twin alongside the reading's timestamp.
- **Alerts:** Information about unexpected scenarios or failures. This information could be provided by the physical machine or by the digital twin itself. The digital twin can perform some logical checks as described later in this section and create alerts based on the check result.

All this information is managed and stored by the smart contract of the machine. Therefore, the machine's digital twin protects the information from being altered by unauthorized users or parties. However, all the information stored in the machine's smart contract will be publicly readable. So far, we have described the machine's digital twin as a registry of information. To go beyond this and make the digital twin of the machine an active component, we used the programmability feature of the blockchain. The smart contract of the machine can do complex programmable behaviors on the stored data. It could be programmed to perform conditions checks on the stored data and then do some actions once these conditions are verified. Of course, the physical machine can perform these checks by itself. However, having the digital twin to perform them will bring trust as the blockchain runs it. For example, it could be programmed to perform the following checks:

- **Product Quality Check:** This is a check performed to ensure that the product has some properties or meets a certain quality standard. The check might involve accessing the digital twin of the product.
- **Reading Values Check:** This is a check on the numerical values of the machine sensors. If a reading value, for example, the temperature, exceeded a certain threshold, the digital twin can do some actions like creating an alert.

3.1.2. Machine Identity

As we explained in the previous section, the machine's digital twin is an active actor. It needs a digital identity to facilitate communication and interaction with other entities of the system. The industrial and manufacturing systems have very long-life cycles, and therefore the identity of the machine should be the same during its lifetime. As our system is blockchain-based, identity management cannot be based on traditional approaches. Otherwise, the objectives of the system cannot be achieved. The identity must be owned and controlled by the machine rather than stored or managed by a third party. Therefore, we decided to use a blockchain-based identity management approach. One of the emerging standards that use the blockchain features is Decentralized Identifiers (DIDs) [8]. Our system is using DID as a standard to manage the identities of the machines. Each machine has a permanent identifier called DID. A simple text string e.g. did:example:123456789abcdefghi. Each DID resolvable to a DID document that contains information associated with the identity of the machine. The DID document is stored within the blockchain, making the DID and its document persistent and immutable.

3.1.3. Twin Interaction

According to our modeling, the information between the physical and digital twins is being exchanged in both directions. The machine sends information stored in the digital twin and vice versa; the digital twin sends information to control and change the physical machine's behavior. Interacting with blockchain is done by participating in the network and running the client software of the blockchain platform. The details of this software may be different depending on the implementation of the platform. Regardless of this, every network node needs this client to process the transactions and validate/create blocks in the chain. Such functionality requires a large amount of storage and processing power as the node needs to have a full copy of the whole chain. The manufacturing machines could not have such capabilities to be a node and run the blockchain client software. Therefore, we decided that the physical machine will not run the blockchain client by itself. Instead, the physical machine will communicate via some protocol with a gateway running the blockchain client and acting like one of the blockchain nodes. The gateway will use the machine's private and public key pair to interact with the blockchain on behalf of the machine. An assumption has to be made regarding the communication channel between the machine and the gateway. We assume the channel is secured, and no attackers can alter or modify the messages exchanged between the machine and the gateway. Through this gateway, the machine will send data to its digital twin inside the blockchain.

So far, we have explained how the physical machine can send data to its digital twin. The next type of interaction is when the digital twin wants to notify or send data to the physical machine. As the machine's digital twin is a smart contract deployed on the blockchain, it runs in a closed execution environment and cannot directly interact with external systems. The only way the digital twin of the machine can communicate with the outside world is through events. The smart contract of the machine the following events to interact with the physical machine:

- **Task Assigned:** An event emits when a process assigns a task for the machine. The emitted event has all the information about the assigned task.
- **Task Started:** An event emits when the machine starts executing a task.
- **Task Finished:** An event emits when the machine finishes executing a task.

Anyone can watch these events, including the gateway. Once the gateway receives new events, it forwards them to the machine. Figure 2 illustrates the interaction between the machine and its twin while executing a task.

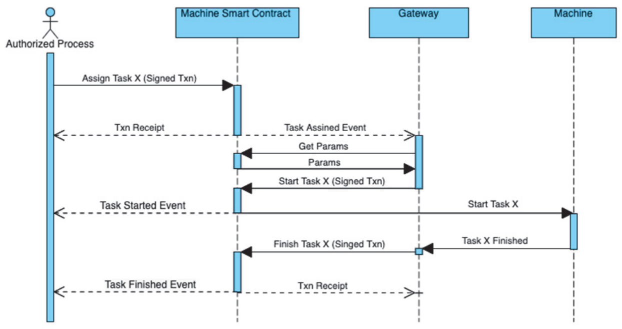


Fig. 2: Twins Interaction

The sequence starts when an authorized process assigns a task for the machine by calling a function on its smart contract. The contract emits a task-assigned event. The gateway is listening to the events generated by the smart contract of the machine. Once the gateway receives the task assigned event, it will first get its parameters and then send the task to the physical machine. At the same time, the gateway will call the start function on the machine's smart contract. Calling this function will store the starting time of the task and emit the 'Task Started' event. The physical machine is now performing the task. After it finishes the task, it will inform the gateway about the finished task. Then, the gateway calls the finish task function of the smart contract of the machine. This function call will store the finishing time of the task and emit the 'Task Finished' event. The machine might send information about the product operations being performed to the gateway during the task execution. The gateway takes this information and passes it to the machine's digital twin, which stores it in the product's digital twin.

3.2. Product Modeling

3.2.1. Product Digital Twin

The product is the second primary entity in our system. The product as an entity can be modeled in many ways. Lots of information can be stored throughout the product's life cycle. In our work, we focus only on what is happening during the manufacturing process inside the factory. Our modeling only takes into consideration simple non-compounded products, and the granularity is a single product. All the product information is stored in one smart contract called 'Product'. This information is provided autonomously by the machine's digital twins.

While the machine executes one of its tasks, the machine's digital twin will add this operation to the product's digital twin if the machine operates on the product. In this way, all the operations performed on the product by different machine is stored in one place, and they form the digital representation of the manufactured product. The contract stores the following information for each product:

- **Identity:** Information about the digital and physical identity of the product.
- **Operations:** This is general information about the operations performed on the product. Each operation is stored with a name, result, timestamp, and information about the machine that did this operation.
- **Processes:** Information about the authorized processes allowed to modify the digital twin of the product.

3.2.2 Product Identity

For the product identity, we are using the same identity management as the machine. Each product has a DID associated with it.

3.2.3 Product Credentials

Besides having the product information stored on-chain in the digital twin, the product has off-chain credentials. For every operation performed on the product, it receives an offline credential/claim from the machine. The credential can be verified by a third party to ensure its authenticity. To achieve this, we used the emerging standard from the W3C, which is called Verifiable Credentials (VCs) [9]. This standard fits well with the DID standard we used to build identity management. The machine is the issuer of the credential, and the product is the subject. It creates the credential containing information about the product, the operation, the machine's DID, and cryptographic proof. The product can claim it underwent a particular operation or satisfied a certain standard by presenting the corresponding credential to a verifier. The verifier can check the machine's DID document (the issuer) and cryptically verify the claim's authenticity. Having each operation as a separate credential allows the product to present the needed information to the verifier without revealing other information that might be sensitive.

3.3. Manufacturing Process Modeling

The actual manufacturing processes in real life tend to be complicated and consist of several stages or even sub-processes. Each one of them involves a lot of machines and devices. We are considering a simple manufacturing process that consists of several steps and no sub-processes. Each step is a high-level task performed by a specific machine, such as fetching an empty container. We also assume that the process is fixed. In other words, the steps, their order, and the corresponding task types are known in advance. However, the machine allocation is dynamic. So, the machine executing a specific task can be replaced by any other machine that can do the same type of task. Executing the process requires communication and interaction between the digital twins of the machines involved in the process.

3.3.1 Process Structure

We modeled the manufacturing processes as smart contracts. Each process is a smart contract written by a manufacturer and running on his behalf on the blockchain. The smart contract is programmed to assign tasks to the digital twins of the machines. The smart contract consists of several functions. Each function represents a step in the process. The function body assigns the task to the machine and executes other business logic if necessary. The contract is responsible for starting/finishing the execution of the process instances. For each execution, the contract creates a process instance and stores information like the starting time, the finishing time, and the execution status.

3.3.2 Process Execution

The process execution is done with the help of a client, which also runs in the gateway. The communication between the client and the smart contract of the process is done the same way explained in the twin interaction section. The client calls functions in the smart contract by signing transactions, and the smart contract emits the following events to communicate with the outside world:

- **Process Started:** An event emits when a process instance is started. The emitted event has all the information about the started instance.
- **Process Step Started:** An event emits when a process step is started.
- **Process Finished:** An event emits when a process instance is finished.

The client will be listening to these events and other events from machines' smart contracts. The execution is carried on by the client calling the process functions. Each function represents a step in the process, and the function call assigns a task to the corresponding machine. Before executing the process, the addresses of the digital twin of the involved machines must be supplied into the smart contract. It starts when the owner or an authorized actor triggers the process by calling the start function, which emits the 'Process Started' event. The client then calls the first step function, which assigns the first task of the process to the corresponding machine. Now the machine will work on the task as we explained in a previous section. Once the machine finishes its task, and the task finished event is emitted from its digital twin, the client can resume the execution by calling the second step function, assigning the second task to the responsible machine.

The execution continues in the same way until the process finishes all its steps, and then the 'Process Finished' event is emitted. Before assigning each task, the process's smart contract needs to authorize the machine if the task involves performing product operations. During the execution of the process, the smart contract of the process can access the digital twin of the product or the machines' digital twins and get data from them. In such a way, the smart contract will enforce the manufacturing process's rules and requirements. With this modeling, the machines act as separate entities and can be used by several processes.

To illustrate the execution, we present an example of a process that involves two machines. Each machine has its digital twin as a smart contract deployed into the blockchain. These two machines belong to different owners and might be in different locations. The process owner (the manufacturer) decided that his process needs two machines capable of doing task 1 and task 2. The manufacturer writes and deploys the smart contract of the process, including all the business logic that implements the argument between him and the machines' owners. The following figure shows the sequence diagram for executing this example process.

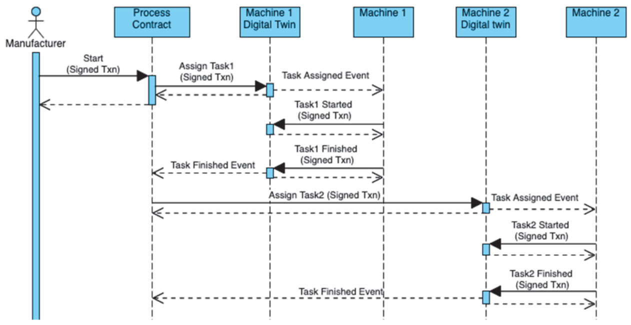


Fig. 3: Process Example

All the interactions between the machine and its digital twin go through the gateway. The gateway was omitted from the diagram for simplicity. However, the machines and the manufacturer might be using different gateways to access the blockchain in this process.

4. Implementation

To validate our conceptual design and modeling, we implemented it for a case study done with the help of the Fischertechnik Learning Factory [12]. The factory model is shown in figure 4. It has a built-in program that depicts the ordering, production, and delivery processes in digitized and networked steps. The factory model comes pre-configured and programmed to perform a set of built-in demo scenarios controlled and monitored through an online dashboard. Even though the Fischertechnik factory model is a fully functional simulation, we had to customize it to fit our needs. We split the factory into four machines Vacuum Gripper Robot (VGR), High-Bay Warehouse (HBW), Multi-Processing Station with Oven (MPO), and Sorting Line with Color Detection (SLD). Each machine has one or more task types. The machines collaborate to perform two processes, namely the supplying process and the production process.

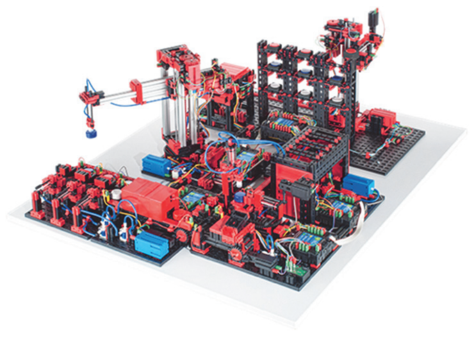


Fig. 4: Fischertechnik Factory Model

The implementation of this prototype was made using Ethereum blockchain with the help of open-source libraries and frameworks like Truffle, Ganache, Open Zeppelin, Web3.js, and others.

4.1. Smart Contracts Implementation

4.1.1. Machines

To make the implementation of the machine smart contract generic, we decided to use the template method pattern by making the machine contract an abstract contract. This abstract contract contains all the functionality mentioned in the machine modeling section, shared between all machines. In this way, all digital twins of our system will have the same interface, so other components can interact with any machine if its smart contract extent the abstract base contract. For a new digital twin of a machine to be created in the system, the machine's smart contract must extend the abstract machine contract. The new contract must implement few abstract functions to define the tasks and their types. The custom functionality can then be added to the child contract by using/overriding the parent contract's functions. The child contract can also implement and enforce custom rules or business logic by overriding the abstract contract's functions. For example, one machine can override the 'save reading' function and check the reading value. If the value is below/above a certain threshold, an alert will be created by calling the 'save alert' function.

4.1.2. Products

The implementation of the product digital twin is a single, smart contract called 'Product'. The contract stores information about all products of the system. The product smart contract allows creating a product by calling the create product function. The function takes an Ethereum address as the DID of the product and creates a record for this product. The caller of this function will be the product owner, and it cannot be changed. The product owner can add info to the product's digital twin using the web application by calling the corresponding functions and signing transactions with his keys. As we explained in the modeling section, the product smart contract authorizes processes that can authorize machines to modify the digital twin of a single product. The authorized machine can save the operations performed on a particular product in its digital twin. Operations info and their results can be accessed later by smart contracts to ensure that they meet specific requirements. Another essential info stored about the product is the physical identifier. The identifier could be an NFC UID or a barcode. It is used to access the digital twin of a product and get all the information about it. Another way of retrieving the info is using the product's DID, an Ethereum address.

4.1.3. Processes

The implementation of processes uses the same approach as machines. Therefore, we created an abstract smart contract called Process to include all processes' standard functionality. For a new process to be created in the system, the smart contract must extend the abstract process contract and implement the functions that define the number of machines, the number of steps of the processes, the order of execution, and the process name.

After deploying the process smart contract, the process owner must set the smart contract address for every machine involved in the process. The machine address can be changed at any time but only by the owner of the process. The process can then be started on a particular product by calling the start process function, which takes the product's Ethereum address (DID) as an argument. The start process function calls the authorize process function in the product smart contract to authorize itself. Only the product owner can authorize a process; therefore, the product owner can only call the start process function.

Once the process owner calls the start function and the corresponding transaction is confirmed, the contract emits the 'Process Started' event. The process client which runs in the gateway will be listening to this event type. After the process contract emits the starting event, the client will begin executing the process by calling the first step function. All steps functions take the process instance ID as an argument, and inside each of them, the corresponding task is assigned to the machine by accessing its digital twin and calling the assign task function. In addition to this, custom functionality can be part of the step function body. The process client is also listening to the task finished events emitted by the machine smart contracts. Whenever a machine finishes a task assigned to it, the process client will trigger the next step, which assigns the next task in the process. This execution continues until the process reaches its final step. Then the process client calls the finish process function to mark this instance of the process as finished.

4.2. Identity Implementation

We used the DID standard to assign digital identities for machines and products. The DID standard is just a specification, and the implementation details are left to the DID method. There are many DID methods available with a functional implementation. Each one of them has different functions, but all of them comply with DID specifications. We used the ethr DID method developed by uPort. The ethr method uses the registry specified by the ERC 1056 which is available as an open-source project [13]. The implementation of the verifiable credentials in our system is based on the library called did-jwt developed by Decentralized Identity Foundation [14]. It allows signing and verifying JSON Web Tokens (JWT), and all public keys are resolved using DIDs. The library support ethr DID method alongside many other methods. In our case study, the signer is the machine, and the subject receiving the credential is the product. Each machine client uses the library to sign a credential using its DID. The content of the credential could be anything if it is a valid JSON object. In our implementation, the credential content is information about a product operation. Anyone interested in verifying the credential can use the library or a similar library to check its validity. Under the hood, the library access the DID registry to check the credential signer's validity. We build a verifiable credential resolver to decode the credential and verify its validity.

5. Results

The implementation result is a fully functional prototype of the conceptual design applied to the Fischertechnik factory. The prototype includes a distributed web application (DApp) that allows different actors to interact with the machines, products, and processes through the blockchain. It provides dashboards to monitor and control manufacturing machines and processes that run autonomously by smart contracts. The following are some of the web application user interfaces. As the web user interface is big and cannot fit in one image, some images are not a full user interface but rather a snippet that shows a particular part of the interface.

Dashboard Interface

This interface shows three kinds of information: machines' current status, processes' current status, and events log. The current machine's status is coming directly from the machine's digital twin for each machine. The web application is listening to the smart contract events, and based on the emitted events, it changes the machine's status in the UI. The upper part of the interface shows the status of the Fischertechnik machines.

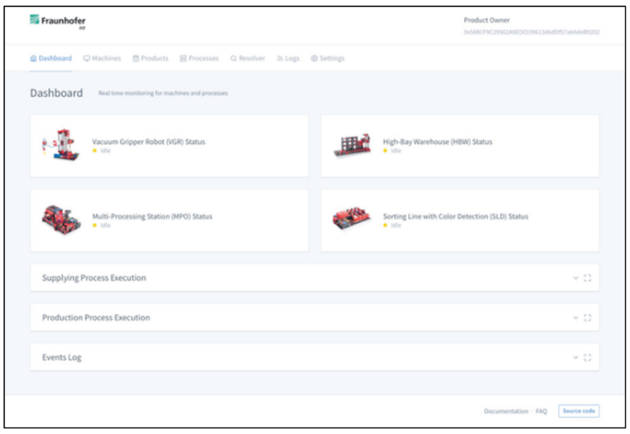
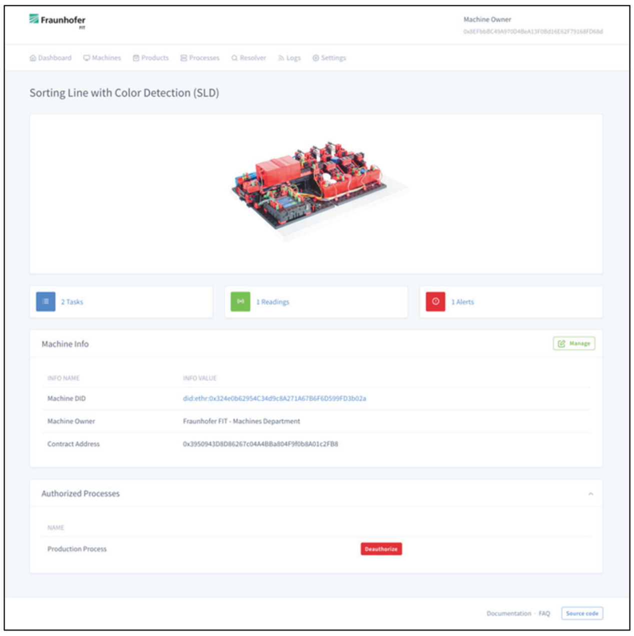

Fraunhofer
Product Owner
Dashboard
Machines
Products
Processes
Resolver
Logs
Settings
Dashboard
Real time monitoring for machines and processes
Vacuum Gripper Robot (VGR) Status
Idle
High-Bay Warehouse (HBW) Status
Idle
Multi-Processing Station (MPO) Status
Idle
Sorting Line with Color Detection (SLD) Status
Idle
Supplying Process Execution
Production Process Execution
Events Log
Documentation · FAQ
Source code

Fig. 5: Dashboard UI

Machine Interface

This interface displays the information stored in the digital twin of the machine. It includes information about tasks, readings, and alerts with links to the corresponding pages. Other information about the machine like the DID, the machine owner, the contact address is also presented. Moreover, the interface also lists the authorized processes with an option to unauthorize them. The following figure shows the interface for the SLD machine. There are three other similar interfaces for the rest of the Fischertechnik machines. This interface can display the information for any machine if its smart contract is inherited from the base Machine contract.



Machine Owner
0x8EFbbBC49A970D4BeA13F0Bd16E62F79168FD68d

Sorting Line with Color Detection (SLD)

2 Tasks | 1 Readings | 1 Alerts

Machine Info

INFO NAME	INFO VALUE
Machine DID	did:ethr:0x324e0b62954C34d9c8A271A67B6F6D599FD3b02a
Machine Owner	Fraunhofer FIT - Machines Department
Contract Address	0x3950943D8D86267c04A4BBa804F9f0b8A01c2FB8

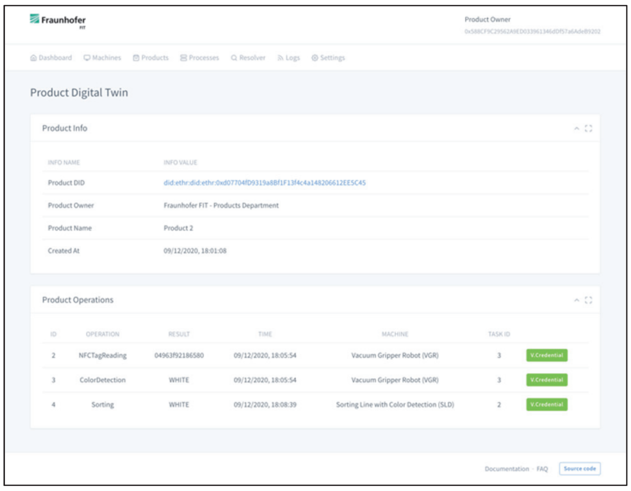
Authorized Processes

NAME
Production Process

Fig. 6: Machine UI

Product Interface

This interface shows information about the digital twin of the product. Information about the product like the DID, owner name, owner address, and the creation time is displayed. Furthermore, it shows all the operations performed on this product by different machines. Figure 7 shows the information about a product that went through three operations by two machines. For each operation, a link to the corresponding verifiable credential is provided. The verifiable credential resolver interface is used to display the credential's details by clicking on the link.



Product Owner
0x588CF9C29562A9ED033961346dDf57a6AdeB9202

Product Digital Twin

Product Info

INFO NAME	INFO VALUE
Product DID	did:ethr:did:ethr:0xd07704fD9319a8Bf1F13f4c4a148206612EE5C45
Product Owner	Fraunhofer FIT - Products Department
Product Name	Product 2
Created At	09/12/2020, 18:01:08

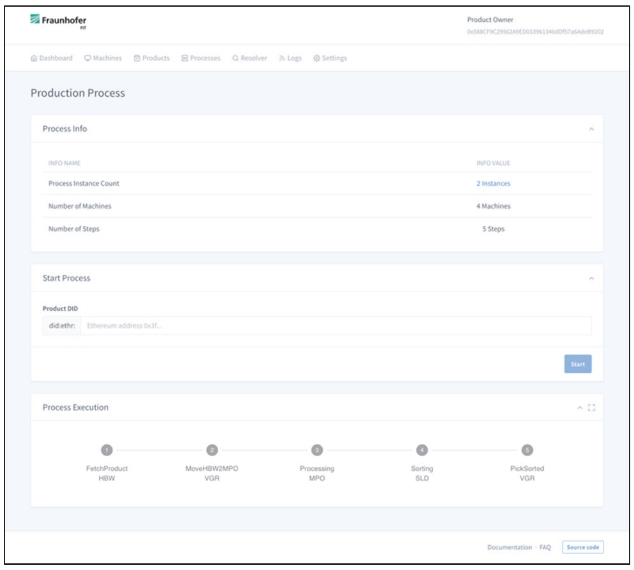
Product Operations

ID	OPERATION	RESULT	TIME	MACHINE	TASK ID	
2	NFCTagReading	04963f92186580	09/12/2020, 18:05:54	Vacuum Gripper Robot (VGR)	3	V.Credential
3	ColorDetection	WHITE	09/12/2020, 18:05:54	Vacuum Gripper Robot (VGR)	3	V.Credential
4	Sorting	WHITE	09/12/2020, 18:08:39	Sorting Line with Color Detection (SLD)	2	V.Credential

Fig. 7: Product UI

Process Interface

This interface displays the process information, including the number of instances, machines, and steps. Also, it allows starting the process on a particular product by providing the DID product. The execution can be tracked with the same UI compound used in the dashboard interface if the process is started. Figure 8 shows the interface for the Fischertechnik factory model's production process.



Product Owner
0x588CF9C29562A9ED033961346dDf57a6AdeB9202

Production Process

Process Info

INFO NAME	INFO VALUE
Process Instance Count	2 Instances
Number of Machines	4 Machines
Number of Steps	5 Steps

Start Process

Product DID
did:ethr: Ethereum address 0x3f...

Process Execution

Fig. 8: Process UI

5.1. Source Code

The source code of the implementation is divided into two parts. The first one is the source code for the Fischertechnik factory model. It is a fork of the source code provided by the manufacturing company [15]. In this fork, we made the necessary changes to implement the presented scenarios. The second part is the source code of the smart contracts, the gateway, and the web application [16]. Inside the source code repositories, the Readme file contains technical details on how to get started with the code and run it and other implementation aspects that were not discussed in this paper.

6. Conclusion

This paper investigated the applicability of blockchain in the manufacturing industry. Our main contribution was building a generic conceptual design for a system that utilizes blockchain and smart contract technologies to implement M2M communication and digital twins for machines and products. The conceptual design discussed how machines, products, and manufacturing processes are modeled as smart contracts. The modeling defined which information is stored in the digital twins of machines and products. We showed how the digital twin and the physical machine could interact and share information. We also spouted the blockchain-based identities for both the machines and the products. The design also discussed how M2M communication is implemented and executed through the blockchain.

Different aspects could be improved upon in our work, which forms a basis for future work. Our modeling for the digital twin of the machine only included the operational aspect of the machine. However, there is much information directly related to the machine's operational conditions, like maintenance operations. Extending the digital twin information by considering other aspects of the machine will allow building more services that fit the industry 4.0 needs. M2M communication represents another area of interest. It was modeled in our design to fit particular types of manufacturing processes. Additional work needs to be done in order to make it applicable to other types of processes. Another direction to improve the process modeling is to use model-driven engineering. It allows auto-generation of the smart contract source code instead of writing it manually.

Even using our conceptual design without modification can still provide a foundation for building other services and applications. The smart contracts of the machines and the processes can be extended to enforce any custom logic. It can be a business logic to implement the payment between the machine owner and the product owner. Alternatively, it can be used to implement the warranty agreement between the machine owner and the maintenance company. Lastly, some technical aspects of the solution can be improved. For example, the interaction between the machine and its digital twin can be optimized by eliminating the gateway role and letting the machine interacts directly with the blockchain.

References

[1] A. Rojko, "Industry 4 . 0 Concept : Background and Overview," vol. 11, no. 5, pp. 77–90, 2017.

[2] T. M. Fernández-caramés and S. Member, "A Review on the Application of Blockchain for the Next Generation of Cybersecure Industry 4 . 0 Smart Factories," 2018.

[3] A. Angrish, B. Craver, M. Hasan, and B. Starly, "A Case Study for Blockchain in Manufacturing: 'fabRec': A Prototype for Peer-to-Peer Network of Manufacturing Nodes," *Procedia Manufacturing*, vol. 26, pp. 1180–1192, 2018, doi: 10.1016/j.promfg.2018.07.154.

[4] J. J. Sikorski, J. Haughton, and M. Kraft, "Blockchain technology in the chemical industry: Machine-to-machine electricity market," *Applied Energy*, vol. 195, pp. 234–246, 2017, doi: 10.1016/j.apenergy.2017.03.039.

[5] Z. Li, A. V. Barenji, and G. Q. Huang, "Toward a blockchain cloud manufacturing system as a peer to peer distributed network platform," *Robotics and Computer-Integrated Manufacturing*, vol. 54, no. January, pp. 133–144, 2018, doi: 10.1016/j.rcim.2018.05.011.

[6] M. Y. Afanasev, Y. V. Fedosov, A. A. Krylova, and S. A. Shorokhov, "An application of Blockchain and Smart Contracts for Machine-to-Machine Communications in Cyber-Physical Production Systems," no. May, 2018, doi: 10.1109/ICPHYS.2018.8387630.

[7] C. Garrocho, C. Marcio Soares Ferreira, A. Junior, C. Frederico Cavalcanti, and R. R. Oliveira, "Industry 4.0: Smart Contract-based Industrial Internet of Things Process Management," pp. 137–142, 2019, doi: 10.5753/sbesc_estendido.2019.8649.

[8] DID-Core, https://www.w3.org/TR/did-core/ (accessed Jan. 15, 2021).

[9] VC-Data-Model, https://www.w3.org/TR/vc-data-model/ (accessed Jan. 15, 2021).

[10] A. V. Barenji, Z. Li, W. M. Wang, G. Q. Huang, and A. David, "Blockchain-based ubiquitous manufacturing: a secure and reliable cyber-physical system," *International Journal of Production Research*, vol. 0, no. 0, pp. 1–22, 2019, doi: 10.1080/00207543.2019.1680899.

[11] X. Zhu and Y. Badr, "Identity management systems for the internet of things: A survey towards blockchain solutions," *Sensors (Switzerland)*, vol. 18, no. 12, pp. 1–18, 2018, doi: 10.3390/sxx010005.

[12] Fischertechnik, https://www.fischertechnik.de/en/products/teaching/training-models/ (accessed Jan. 15, 2021).

[13] uPort, https://github.com/uport-project/ethr-did-registry (accessed Jan. 15, 2021).

[14] D. Identity, https://github.com/decentralized-identity/did-jwt/ (accessed Jan. 15, 2021).

[15] M. Ghanem, https://github.com/ghanemmhd/txt_training_factory/ (accessed Jan. 15, 2021).

[16] M. Ghanem, https://github.com/ghanemmhd/master-thesis-implementation (accessed Jan. 15, 2021).