Proposal of Digital Twin Platform Based on 3D Rendering and IIoT Principles Using Virtual / Augmented Reality

Roman Leskovský, Erik Kučera, Oto Haffner and Danica Rosinová
Faculty of Electrical Engineering and Information Technology
Slovak University of Technology in Bratislava
Bratislava, Slovakia
Email: roman.leskovsky@stuba.sk, erik.kucera@stuba.sk

***Abstract*—The paper describes proposal of a platform for digital twin monitoring, control and device state diagnostics. Content is based on common problems like data harvest, visualization and deliverance to an end user. At first, basic terms are explained, then the system is being revealed part by part, divided into units according to tasks the subsystems are used for. Not all the techniques are newly created as we consider also using the well-known software for solving partial problems and integrating that software in this platform solution. System itself, if implemented, should provide an alternative and a new approach to the problem of digital twin visualization providing the tools for data harvest from end devices and diagnostic tools to fetch those data to the platform user on the other end.**

***Index Terms*—Industry 4.0, digital twin, virtual reality, mixed reality, mechatronics, Internet of Things**

I. Introduction

Technology has massively changed in last two centuries which was also accelerated by inventions causing revolutions followed by a great step forward. Currently we are experiencing new revolution called Industry 4.0 which changed the way we look at data, data sources and its collection. Not only industrial areas, but also cities were flooded with many kinds of sensors connected to nodes that can communicate with the internet, which makes measured data accessible from anywhere. Even a basic household can posses multiple of such devices.

Collected data make room for different kinds of analysis, which was very hard to make or was not even possible before. Analysis can reveal important information about the measured conditions, based on which can be those devices or systems improved and their use can be more efficient.

By the tasks of communication, security and harvesting of data arising from Industry 4.0 revolution which provides several solutions, there is also a need of visualization of data. Data can be visualized in multiple ways. Dealing with this task the digital twin term was created, but due to the various visualization approaches and degree of interactivity, there were certain differences. From charts on web pages to the virtual and augmented reality with possibility of direct control.

Physical devices having its virtual counterpart can provide many benefits. The paper describes practical use for 3D visualization of devices with VR and AR and whole system infrastructure is proposed which is the first step in creating proposed digital twin platform.

II. Digital Twin

Digital twin can be described as a digital construction of information about a physical system. Standing alone, but both real and virtual copies are connected for data sharing. Digital twin should optimally include all the information related to physical device that can be obtained for the greatest resemblance and relevance of presented data. Therefore, twin is not only about the visual similarity as connection and data sharing is what makes digital twin a real twin with its full potential. By that, digital copies can be as stated in [1] divided by the form of integration to:

- Digital model
- Digital shadow
- Digital twin

Digital model is a digital representation of existing or planned physical object that is not using any form of automatic data exchange with physical device. Device state change does not affect model state. Digital shadow is a digital model but, in this case, physical device shares the data with its shadow and digital shadow reflects its state based on the received data. Its one-way communication.

Digital twin, unlike model and shadow, leads two-way communication with its counterpart. Therefore, not only device change affects the twin, but also changing the digital twin state may affect real device. Digital twin state changes can be used to control the device.

Digital twin evolution can be divided into four phases [2]:

- First phase – physical only
- Second phase – digital version created to complement physical data
- Third phase – interaction between physical and virtual
- Fourth phase – further interactions and convergence between physical and digital layer

Digital twin is using historical data from physical device to understand a context of condition of the device in real

978-1-7281-4381-1/20/$31.00

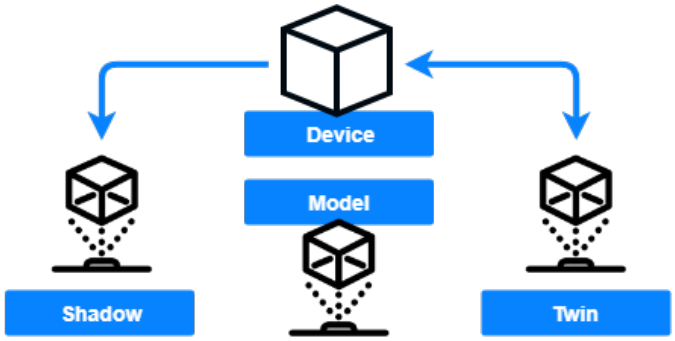

Fig. 1. Digital model, shadow and twin differences

time while also using real time data for presentation of current device state. In addition, device past can be used for predicting the future. GE Digital in [3] defines digital twin hierarchy by its layer to four main groups:

- Component – digital twin represents single part of a device
- Object - digital twin represents whole device of connected components
- System – composed of objects
- Process – consists of activities and operations, as a manufacturing process. Process twin can access data from multiple twins of objects or systems, but it is aimed at process analysis more than devices.

By knowing the actual condition of devices, future states can be predicted which helps to effectively monitor, simulate and control actuators or processes and optimize life cycles.

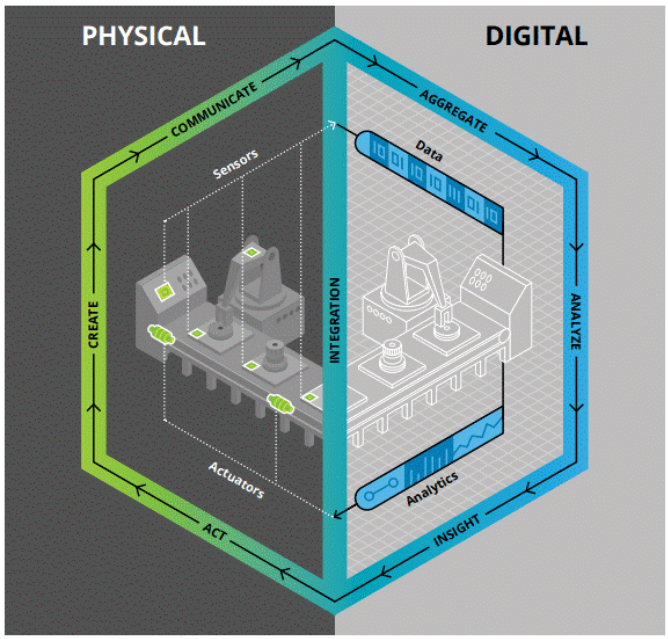

Fig. 2. Real world connected with virtual [4]

III. Digital Twin Platform Proposal

Considering digital twin features and possibilities we are designing a platform architecture which would handle tasks connected with digital twin creation and provide unified dynamic solution for data visualization, powerful but still simple and smart, as this is the feature that is still not available in existing solutions.

Whole proposed system should provide connection between devices, data storage, visualization, diagnostics and simulations which are identified as main tasks in solving the problem. Platform should allow creation of fourth phase digital twins - both-way communication with real devices with more complex communication and automation based on predictions, creating valuable output for users.

At this state, proposal is aimed mostly at visualization, storing data and diagnostics. Other parts of system, for now, count with integrations of other platforms which are perfect solution for those partial tasks.

In general, proposed platform is a server-oriented application connected with other platforms and machines that could provide solid computing power for some simulation calculations. It provides an access for its users, who can manage their devices through web interface, which is standardly protected. Except for this interface, there is a lightweight client application available. The application is used for loading data, visualization and editing of digital twins, it also allows device control and simulation.

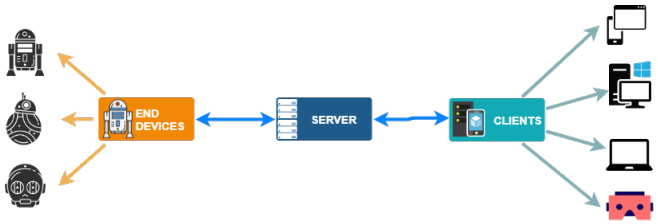

Fig. 3. Simplified system architecture

A. Visualization

As stated, visualization is done through client application, which is used as an end point in hands of a platform user. It is presenting data stored in system server in organized structures of relations described by relational database.

Even though data could be accessed directly from nodes with sensors, cross-server solution was selected, as too many client applications could cause not necessarily much traffic and overload nodes.

Besides that, application is used for communication back to server informing about the user actions so it could have effect on system or devices on the other end based on the action.

1) Displayed Data: The intension is to visualize several types of data and structures for different purpose and based on these factors, the data is dynamically created inside the application.

We can pre-divide visualized data into groups:

- graphical dependency data
- spatially oriented data
- data bound to individual components or to the entire system

Graphical dependency data represent classical sensor data of one or more values changing in time as a result of external factors. It is natural to display such data in two-dimensional charts as it is customary without experimenting.

However, there is still room for experiments in the other two data groups. Spatial-oriented data is described by values that, either absolute or relative, describe the position and rotation of components in space. Such data is more clearly displayed naturally, and thus also use the space and spatial representation in which we replicate the position and rotation.

2) Supported Data Types: Spatial rendering needed for spatial-oriented data is a complicated task consisting of many problems, the solution of which requires advanced knowledge of mathematics, informatics and physics, and the solution is a non-trivial long-term system creation.

Game industry is dealing with this problem for decades, which lead developers to create software, that helps to develop 3D applications. Software is called engine and it provides most common tools and reusable components prepared to solve different task in 3D rendering, which makes process of development quick and relatively easy.

Therefore, client application will be built in 3D engine. Engine Unity was chosen from multiple options as it directly or indirectly supports all the areas of planned development that are in focus. It is also easy to learn, provides decent pricing conditions, full documentation and is growing very fast, implementing new functionalities.

Using the engine, three-dimensional objects can be created in virtual space, moved and rotated as needed.

Engine tools are also needed for the last data category that binds data to individual components. Using engine, this data can be displayed by the exact component or affect component itself. For example, a fault part of the device may be displayed in a different color than the rest of the device, or graphical hints and tips may be displayed for individual components - information about them, or highlighted actions that can be taken - such information can be used for employee training or device repairs but also while just inspecting the device.

Camera is an important component in 3D creation that represents the point and direction the user is looking from. It affects the control and display of the application. The app will include multiple versions of camera view modes:

- PC desktop view
- AR views - Microsoft Hololens, mobile phones and others.
- VR view - headsets like Windows Mixed Reality

The advantage of the Unity engine is the ability to build a single application for multiple platforms, so the same application can be built for different platforms (PC, mobile phones, web). The engine also offers VR and AR tools that allow the camera to be automatically adjusted to the detected user machine and equipment including joystick or headsets. Based on this information the application determines how to set up controls and views to suit the user and his device the most.

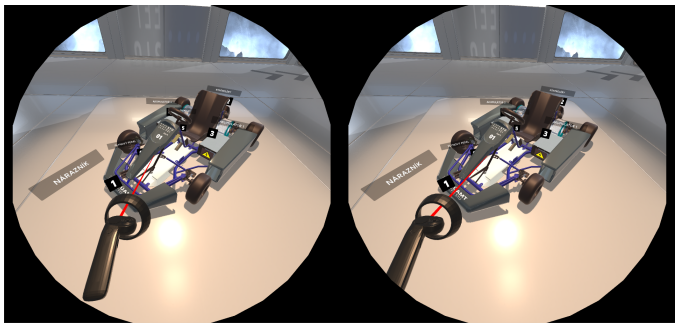

Fig. 4. Example of VR view rendered in Unity [5]

3) Application Options: The main goal is to make client application as simple and practical as possible in terms of architecture, considering possible future developments and applications in practice for many devices. From that point of view, many devices that the application would has to keep in it would eventually arise, increasing the size of the application to gigantic proportions. In addition to each new device, it would be necessary to add the device to the application and rebuild and distribute the application between clients. With the distribution we would have two options - either to build a separate application for each company, which would produce many versions for each enterprise, and for example, when editing the application core, each of these versions would have to be edited unnecessarily. The second option is to include all devices in one application, where security problems arise, as users could easily access other companies' devices

These are the reasons for implementing only camera, controls support, user input and basic menu to applications, while each additional content will be loaded into the application at runtime by user choice. The data will be stored on the server and available for download from the client application after logging in and validating the user based on role.

This method is safe because the user will only have access to those devices that belong to his company and have the authority to handle the device. Roles could be set in the web application server environment and add device permissions or device groups:

- Device View - user can view the device in the application and read the values displayed from the sensors.
- Device Control - user can interact with the twin, which affects the real copy of the device.
- Device Editing - user can edit the device.

Unity does not allow direct load of 3D models at runtime using any of the components, but it can be implemented via custom scripts. Another, more simple solutions are Unity bundles. These bundles can be exported from the application or engine environment, and they have the advantage that they can contain ready-made prefabricated objects - 3D models can have set up parameters, contain control scripts and other

necessary things to make the client application interact with it. Such a bundle can easily be imported into the application during runtime, and then the content loaded and displayed in the scene. Content - a device - that already has scripts that allow the user to control it. In this way, only downloading a package from external storage, that is managed by the server, is required. After downloading, device is displayed as defined 3D model.

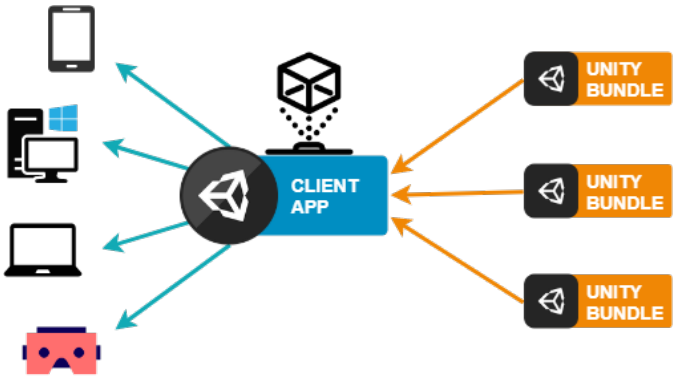

Fig. 5. Unity bundles loaded to client applications

The prefabricated object displayed already contains control and visualization components and is automatically linked to device data from the server as it is a twin of a specific device selected from a set of available devices.

At this point, the visualization elements automatically display data from the server and control components, after the action is invoked by users, send a message to the server along with the parameters of the action and server forwards this message to appropriate endpoint device. Real device responds to the action and these changes are visualized in real time on a virtual replica device that constantly monitors the actual state of the device and updates its settings according to the values obtained.

Depending on the device you are using to launch the application, you can select which devices to display. For a desktop PC, along with a VR version of the application, the user will have a list of all the devices to which he or she is authorized after login. One or more devices can be selected from the list and then displayed.

In addition to manually selecting a device from the list of provided, the mobile application and AR headsets application will also offer the possibility of automatically detecting devices in locations where they detect either the designated two-dimensional markings or the device itself. In this case, the replica occupies the position of its real world twin in the space and the displayed information and control components complement the surrounding space of the device, giving the user a complete overview of the device's state and sensor values measured without having to interact with physical device.

In addition to measured values, the user may also have access to materials that are not directly subject of measurement, but the devices are directly related to them and are added to the database. In this way, the user could display the technical specifications or documentation for the device, the instruction manual or other relevant data, not only in the form of two-dimensional static texts, but also in an interactive way using spatial representation.

4) Spatial Mapping: In AR visualization, twins are being rendered in real environments. There are several ways how to place these models.

It can be done similarly to a VR application, when model is placed in front of user or on a place specified by user. Little difference is in considering surroundings, so placed models are not rendered inside or behind walls. In Unity AR application, mesh is created from room scan provided by a device used to run the application. Mesh can be used for collision detection, therefore bounding points or specific points of interest selected by user can be identified. Holograms can be then placed on tables, walls or other items in room.

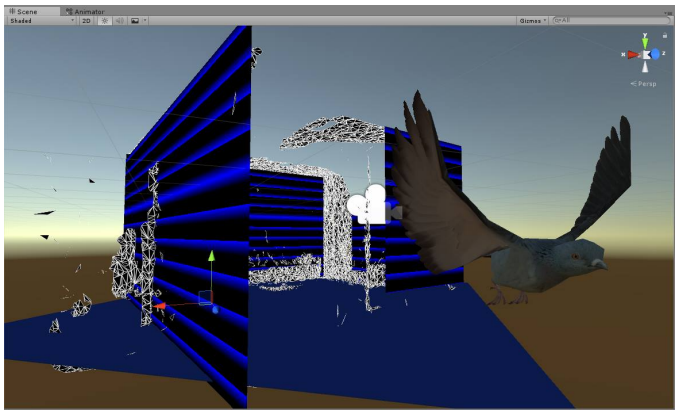

Fig. 6. Example of mesh created from scan of Microsoft Hololens

Another approach is to place twin models not based on users' position, but position of real device. This position can be determined based on:

- Marker
- Device
- Room

In each of these cases is application looking for some known element in camera scene. After finding it, the application synchronizes the position and recalculates it when ever user moves, based on the movement direction. The position relative to found element is calculated also when element is not present in view.

When element is device itself, model of device can be directly placed at this position. If element is a marker or a specific room, relative position of device to this element needs to be know. Then the device position can be extracted from markers position. Model is then placed to calculated device position.

For this task, Unity available plugins for computer vision are used. Either Vuforia or Wikitude are capable of mentioned operations and provide C# SDK for Unity, although Wikitude seems to have better results and performance in tests so far.

B. Industrial Internet of Things (IIoT) and Data harvest

Collecting data is a difficult task as often data sources are not able to communicate in same language. This is just the case of industrial devices, sensors and machines that differs in every possible way. Even though latter-day systems are mostly developed based on today's standards, it is not always the rule. Also, older devices are quite unique too.

IIoT platforms are platforms developed with purpose of connection with these devices by unified actions where background solves the differences that may exist in communication channels with the devices. Platforms also collect and store data about devices and measured values. Data can be accessed through platforms APIs, so it can be accessed from outside - instead of solving this problem by custom solution development, integration is possible through data exchange.

For integration, platforms like ThingWorx was considered. ThingWorx provides API and it supports communication through multiple channels like MQTT, HTTP, WebSocket. In case of unsupported channel, it provides Protocol Adapter Toolkit which is an SDK that allows to define own connector channels to connect also through technologies, that are not supported by platform out of the box [6] (Fig. 7).

Both send and receive data is possible.

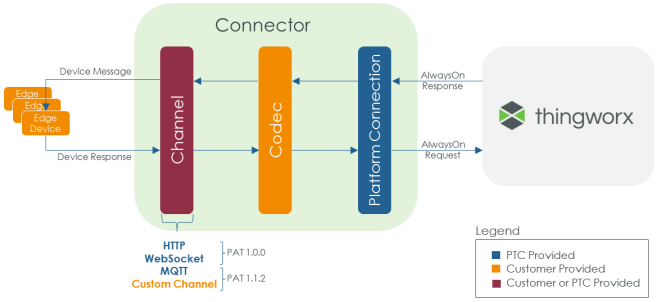

Fig. 7. ThingWorx Connector [6]

Also, platforms like Node-RED can be used. Based on JavaScript and Node.js, Node-RED provides visual scripting editor used for connecting existing components with different functionalities or creating own (Fig. 8). It can also serve as a REST API endpoint, where request starts specific actions or processes with defined response.

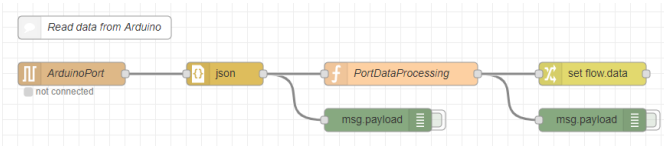

Fig. 8. Node-RED block editor

It provides default set of input and output components for communication which include MQTT, HTTP, WebSocket, TCP, UDP or serial USB connection (Fig. 9).

Custom components or functionality can be created using JavaScript or imported from online library or community.

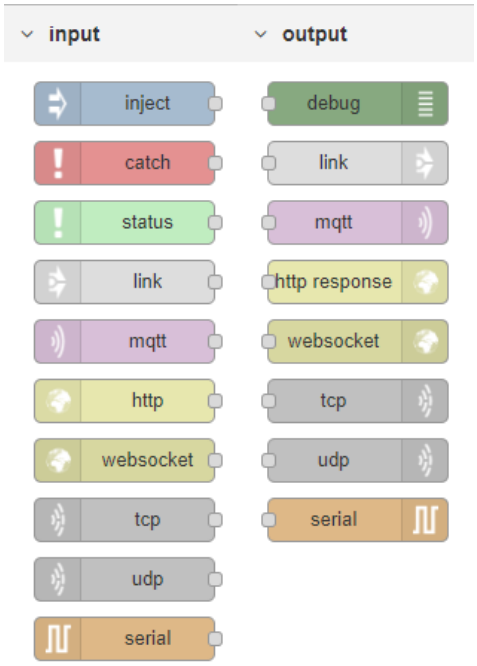

Fig. 9. Node-RED default I/O channels

Workspace sharing is fast and easy, as whole system is built in JavaScript - workspace can be exported or imported as a JSON file.

Another type of platform example that could be integrated is Home Assistant. Home Assistant is an open-source tool for home automation. It integrates most of the commonly used IoT devices technologies in form of components. As stated in [7] it also provides APIs for data access:

- REST API
- WebSocket API
- Server-sent events

Collected data is presented on fully customizable dashboards (Fig. 10).

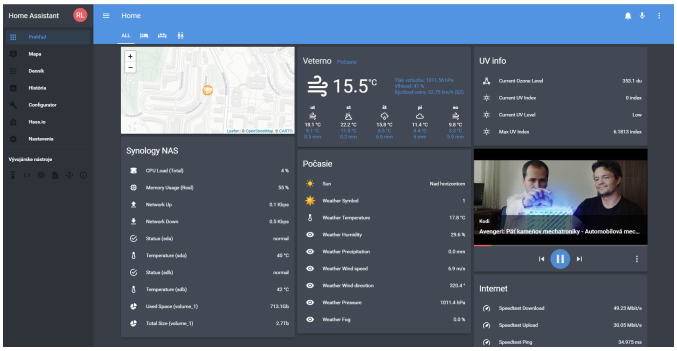

Fig. 10. Home Assistant dashboard

C. Data and storage

Proposed platform works with data coming out of sensors that are then visualized in client application. Visualization

presents mostly real-time or last sensor values, but also longer-term values must be stored for trend evolution visualization as well as for diagnostic purpose which is evaluated not only from single value but set of specific data. Too many sensors could aster some time generate big data, it will be stored just for a specific period.

Besides remembering sensor data, also information about system, properties, devices, sensors, addresses, users and their settings must be stored.

All values are related to others, so storage medium will be a relational database which could best describe connections and bindings between tables.

D. Diagnostics and Simulations

Diagnostics is in this case used for automated check of current state of a device. Simulations are used for predicting future states resulting from current conditions available from device sensors.

For automated diagnostics an artificial intelligence with classification algorithms are great solution. State of devices can be detected from its current and past values by using the data as an input for trained networks or classifiers.

Again, use of existing solutions is a small shortcut. Selected solutions with support for diagnostic needs are Matlab, Python with libraries and Microsoft Azure.

Matlab integration is possible in multiple ways. One way is communication through Production Server, which allows remoted function executions via REST API and JSON representation of Matlab data types.

Matlab code can be also deployed as a standalone application as described in [8], build with all its dependencies. Build does not need Matlab environment to run. It can run on Windows, Linux and Mac.

Another option is integrating Matlab to another environment. According to [9], Matlab version 2019 supports full (both-way) or partial integration with these languages:

- C
- C++
- Java
- Python
- .NET
- COM

Full integration exists with Python as shown in [10]. On the other hand, Python has with libraries like SciPy or TensorFlow all the required functions as libraries contains implemented algorithms for these tasks. Python can also easily create a service with REST API, that could serve for data exchange with local machine running the script.

Online solution could be Microsoft Azure. A platform based on principle of microservices, where single components of software are divided into separated services, with separated logic and functionality.

One of such services is service providing machine learning tools. It can be accessed from Machine Learning Studio, where service can be created and deployed in few clicks together with API and client application examples. Logic is easily designed via block schemes, where input and output values are defined together with whole process.

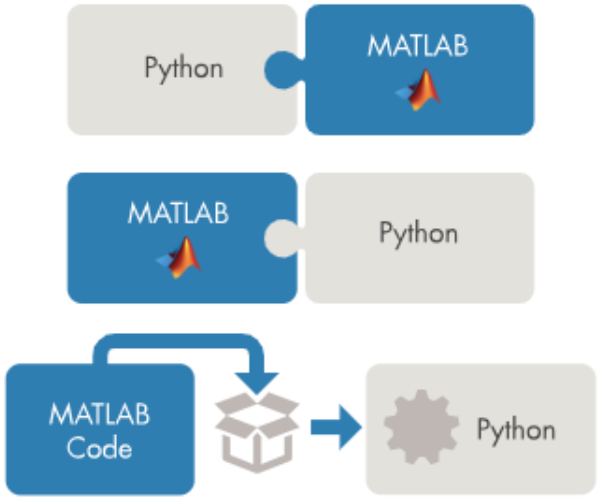

Fig. 11. Matlab integration with Python [10]

In Fig. 12 Multiclass Decision Jungle classifier was used to classify input data. It is a supervised learning algorithm that can classify more than two groups.

States of devices could be classified in a similar manner. Providing more state groups than running and broken could be beneficial. Groups like worn out or replacement needed could inform owner about how many devices need or will need to be repaired, based on what specific number of spare parts can be ordered from supplier.

This groups can be evaluated from sensor values like temperature or sound of an engine.

For simulations and future predictions, Ansys solution will be used. Ansys is multi-physical simulation software that is capable of simulations of complicated physical processes [11]. Despite that, proposed digital twin platform will use only simplified conditions for simulations as real-time or close to real-time results are required and processes too complicated takes too long to be calculated and generates too much data.

Data output from Ansys simulations can be exported to file and used in other applications. It is possible to visualize exported data also in 3D engines (Fig. 14).

Twin Builder is part of Ansys. It is used for creating a physical model of real device, based on which predictions are calculated. Model is designed so it simulates behavior of device. Model is stated to be an on simulation based digital twin in [13]. All Ansys physical features and components can be used to model the behavior (Fig. 15).

It is possible to access the created model and run simulation through Digital Twin Runtime Agent, that manage the process. Integration was tested on IIoT platforms like GE Predix or ThingWorx. Data from simulation predicts future values based on current state of the device. It can reveal information like lifespan.

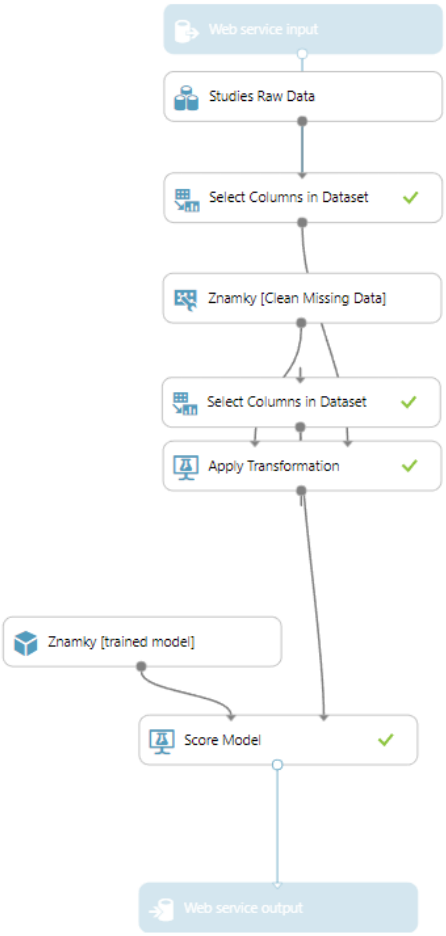

Fig. 12. Block schema in Azure Machine Learning Studio

Fig. 13. Example of device lifetime states

IV. Digital Twin Platform Infrastructure

Whole system architecture, as proposed, is presented by Fig. 16.

V. Conclusion

The paper described proposal of a digital twin platform by its critical components. Each component was introduced as a separated problem with possible solutions. After connecting these components, whole digital twin platform is created. It is capable of collecting data from devices and sensors, visualization of data presented also in VR and AR, device control through its virtual twin, diagnostics and simulations.

The platform, as designed, has wide application range including monitoring, employee training, malfunction detection, manipulation and documentation support, state predictions, security and others. Platform also offers saved time, space and financial means for its users.

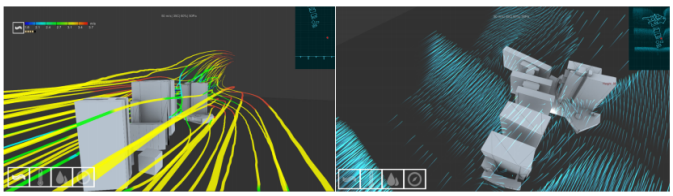

Fig. 14. Unity application visualizing Ansys output [12]

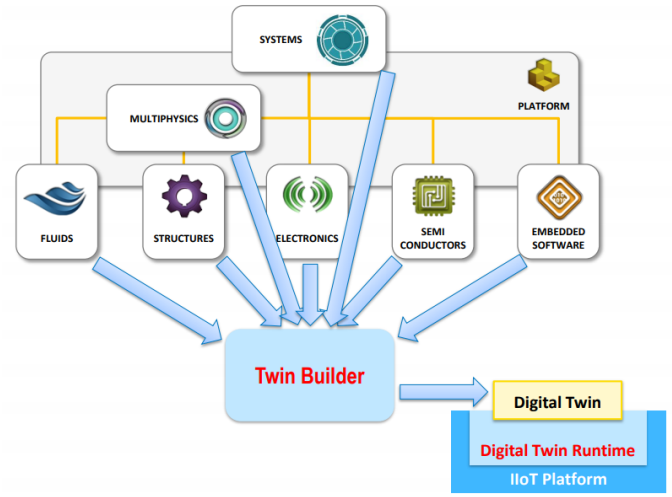

Fig. 15. Digital twin model creation with Ansys Digital Twin [13]

Acknowledgment

This work has been supported by the Cultural and Educational Grant Agency of the Ministry of Education, Science, Research and Sport of the Slovak Republic, KEGA 030STU-4/2017 and KEGA 038STU-4/2018, by the Slovak Research and Development Agency APVV-17-0190, and by the Tatra banka Foundation within the grant programme E-Talent, project No. 2019et006 (Development of Autonomous Vehicle using Virtual World).

References

[1] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, "Digital twin in manufacturing: A categorical literature review and classification," *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1016 – 1022, 2018. doi: https://doi.org/10.1016/j.ifacol.2018.08.474 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2405896318316021

[2] L. Kitain. (2018) Digital twin - the new age of manufacturing. [Online]. Available: https://medium.com/datadriveninvestor/digital-twin-the-new-age-of-manufacturing-d964eeba3313

[3] G. Digital. (2018) Digital twin | ge digital. [Online]. Available: https://www.ge.com/digital/applications/digital-twin

[4] A. Parrott and L. Warshaw. (2017, may) Industry 4.0 and the digital twin.

[5] E. Kucera, O. Haffner, and R. Leskovsky, "Interactive and virtual / mixed reality applications for mechatronics education developed in unity engine," 01 2018. doi: 10.1109/CYBERI.2018.8337533 pp. 1–5.

[6] smainente. (2018) Protocol adapter toolkit (pat) overview. [Online]. Available: https://community.ptc.com/t5/IoT-Tech-Tips/Protocol-Adapter-Toolkit-PAT-overview/td-p/534599

[7] H. A. developers. (2019) Rest api. [Online]. Available: https://developers.home-assistant.io/docs/en/external_api_rest.html

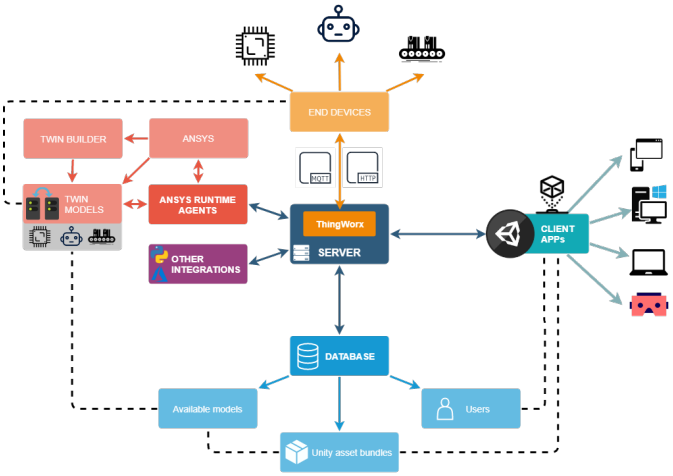

Fig. 16. Proposed platform architecture

[8] Matlab. (2019) Create standalone application from matlab. [Online]. Available: https://es.mathworks.com/help/compiler/create-and-install-a-standalone-application-from-matlab-code.html

[9] ——. (2019) Integrate matlab with external programming languages and systems. [Online]. Available: https://www.mathworks.com/help/matlab/matlab_external/integrate-matlab-with-external-programming-languages-and-systems.html

[10] ——. (2019) Using matlab with python. [Online]. Available: https://www.mathworks.com/products/matlab/matlab-and-python.html

[11] A. Inc. (2018) Ansys twin builder capabilities. [Online]. Available: https://www.ansys.com/products/systems/ansys-twin-builder/twin-builder-capabilities

[12] M. Berger and V. Cristie, "Cfd post-processing in unity3d," *Procedia Computer Science*, vol. 51, pp. 2913 – 2922, 2015. doi: https://doi.org/10.1016/j.procs.2015.05.476 International Conference On Computational Science, ICCS 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050915012843

[13] B. Dion. (2018, Apr.) Digital twin workflow and applications. [Online]. Available: https://uwm.edu/csi/wp-content/uploads/sites/450/2018/04/ANSYS-Digital-Twin-Workflow-and-Applications-FINAL2.pdf