Volume 27, Number 12 December 2022

Parking Today

www.parkingtoday.com

IPMI's
Cindy Campbell and
PT's Photographer
Have Fun at SWPTA

See Page 24

The Digital Twin

• By Abdelrhman Mahamadi Ph.D., and James Sartor BSME

The advancement in technology has led to improvement in manufacturing and automation fields, and one aspect of this improvement is the implementation of a *virtual product* i.e., the Digital Twin to enhance the design, development, commissioning, and operation of the physical product. We present an implementation of the Digital Twin concept for an AUTOParkit™ System, to improve the system life-cycle from design, to operation.

The AUTOParkit System is an autonomous parking system, composed of multiple subsystems that work together to efficiently store (park) and retrieve (return) vehicles. The composability of the AUTOParkit System structure introduces a wide range of decision-making processes to operate the subsystems concurrently to optimally and efficiently park and return cars. The goal of developing such a system is to write an algorithm that will make the best real-time decisions. The sequence of decision-making introduces complexity to the algorithm of this system; hence, any bad decision will lead to an exponential effect on the system performance.

Historically, developers used to debug their algorithms on a live automated system. However, such an approach requires (i) a physical system to be built, (ii) consumables readily available, and (iii) upstream and downstream systems to be operational or simulated, all of which escalated the cost of commissioning in both time and capital. We present in this research a virtual AUTOParkit System, which essentially replicates the exact system structure of a physical AUTOParkit System.

We use the Siemens Tecnomatix Plant Simulation software to build the digital AUTOParkit. We then connect the Digital AUTOParkit to an actual Programmable Logic Controller (PLC) that executes the main program that orchestrates the entire system. Finally, we digitally commission the main program by running real-life scenarios on the digital AUTOParkit, and we debug the algorithm accordingly and compare the cost results of the same AUTOParkit system that is commissioned digitally versus physically.

The use of digitalization technologies enabled virtual product and process planning, consequently resulting in large amounts of data being processed, analyzed, and evaluated by simulation and optimization tools to be able to make them available for planning in real-time (Boschert and Rosen 2016). One of these simulation-based planning and optimization concepts with great potential in many industrial fields is the *Digital Twin* (Tao, et al. 2017). which is the virtual and computerized counterpart of a physical system.

Building a Digital Twin is typically defined as consisting of a physical product, its virtual counterpart, and the data connections in between. The Digital Twin is increasingly being explored as a means of improving the performance of physical entities through leveraging computational techniques, themselves enabled through the virtual counterpart (Jones, et al. 2020). A manufacturing Digital Twin offers an opportunity to simulate and optimize the production system, including its logistical aspects, and enables detailed visualization of the manufacturing process from single components up to the whole assembly (Kritzinger, et al. 2018).

The remainder of this paper includes a summary of recent and relevant research on the digital twin applications in automation, followed by a description of the AUTOParkit System. Then we move on to the implementation, we present a description of the structure of a template AUTOParkit System, then we explain how we used the Siemens Tecnomatix Plant simulation software to build the virtual counterpart of the template system, followed by the description of the data-information interface that we developed to link the digital system to a physical controller. In the results section, we show the real-life scenarios that we developed to test the digital system, followed by a demonstration of *digital commissioning* versus *physical commissioning*. Lastly, we summarize our work in the conclusion section.

ABDELRHMAN MAHAMADI Ph.D., and **JAMES SARTOR** BSME are senior engineers with Dasher Lawless.

For the complete article, please log on to parkingtoday.com and click on the December issue of Parking Today. The article follows the PDF of the December issue.

310 Dana Street NE Warren, OH 44483

Reducing Commissioning Costs by Implementing an AUTOParkit System Digital Twin

Authors: Abdelrhman Mahamadi Ph.D., and James Sartor BSME

Dasher Lawless Automation, Warren OH, USA.

Abstract

The advancement in technology has led to improvement in manufacturing and automation fields, one aspect of this improvement is the implementation of a *virtual product* i.e., the Digital Twin to enhance the design, development, commissioning, and operation of the physical product. We present an implementation of the Digital Twin concept for an AUTOParkit™ System (API), to improve the system life-cycle from design, to operation. The AUTOParkit System is an autonomous parking system, composed of multiple subsystems that work together to efficiently store (park) and retrieve (return) vehicles. The composability of the AUTOParkit System structure introduces a wide range of decision-making processes to operate the subsystems concurrently to optimally and efficiently park and return cars. The goal of developing such a system is to write an algorithm that will make the best decision in real time. The sequence of decision-making introduces complexity to the algorithm of this system; hence any bad decision will lead to an exponential effect on the system performance. Historically, developers used to debug their algorithms on a live automated system. However, such an approach requires (i) a physical system to be built, (ii) consumables readily available, and (iii) upstream and downstream systems to be operational or simulated all of which escalated the cost of commissioning in both time and capital. We present in this research the digital AUTOParkit, which virtually replicates the exact system structure of a physical AUTOParkit System. We use the Tecnomatix Plant Simulation software to build the digital AUTOParkit. We then connected the Digital AUTOParkit to an actual Programmable Logic Controller (PLC), that executes the main program that orchestrates the entire system. Finally, we digitally commission the main program by running real-life scenarios on the digital AUTOParkit and we debug the algorithm accordingly and compare the cost results of the same AUTOParkit system that is commissioned digitally versus physically.

Introduction

This use of digitalization technologies enabled virtual product and process planning, consequently, resulting in large amounts of data being processed, analyzed, and evaluated by simulation and optimization tools to be able to make them available for planning in real-time (Boschert and Rosen 2016). One of these simulation-based planning and optimization concepts with great potential in many industrial fields is the Digital Twin (Tao, et al. 2017). which is the virtual and computerized counterpart of a physical system. Building a Digital Twin is typically defined as consisting of a physical product, its virtual counterpart, and the data connections in between, the Digital Twin is increasingly being explored as a means of improving the performance of physical entities through leveraging computational techniques, themselves enabled through the virtual counterpart (Jones, et al. 2020). A manufacturing Digital Twin offers an opportunity to simulate and optimize the production system, including its logistical aspects, and enables detailed visualization of the manufacturing process from single components up to the whole assembly (Kritzinger, et al. 2018).

The remainder of this paper includes a summary of recent and relevant research on the digital twin applications in automation, followed by a description of the AUTOParkit System. Then we move on to the implementation, we present a description of the structure of a template AUTOParkit system, then we explain how we used the Tecnomatix Plant simulation software to build the virtual counterpart of the template system, followed by the description of the data-information interface that we developed to link the digital system to a physical controller. In the results section, we show the real-life scenarios that we developed to test the digital system, followed by a demonstration of *digital commissioning* versus *physical commissioning*. Lastly, we summarize our work in the conclusion section.

Digital Twin

The origin of the Digital Twin is attributed to Michael Grieves and his work with John Vickers of NASA. Grieves presented the concept in a lecture on product life-cycle management in 2003 (Grieves 2014). Grieves continues this definition by describing the Digital Twin as consisting of three components, a physical product, a virtual representation of that product, and the bi-directional data connections that feed data from the physical to the virtual representation, and information and processes from the virtual representation to the physical. Grieves demonstrated this flow as a cycle between the physical and virtual states of data from the physical to the virtual, and of information and processes from the virtual to the physical, as depicted in Figure 1.

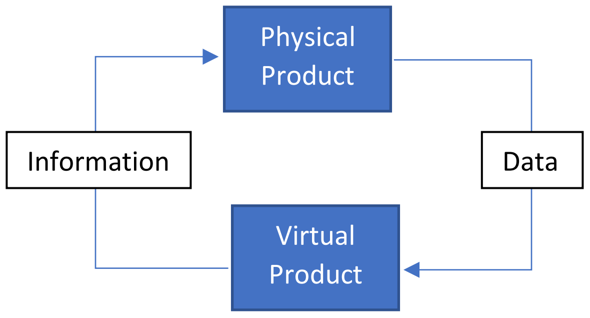

Figure 1 depicts the concept of twining and the data/information flow between physical and virtual counterparts

To make sure we covered all the aspects of implementing the digital twin to the AUTOParkit System, we reviewed most of the recent work in the domain and here we present a summary of the research that was recently done on the use of digital twin in manufacturing and automation.

A data-centric middleware digital twin platform was implemented by (Yun, Park and Kim 2017), that included a distributed digital twin cooperation framework. Their work proved dependable physical systems. Authors in (Lohtander, et al. 2018), used a 3D Model to build a digital twin for a Micro Manufacturing unit and they showed the correspondence between the physical and the digital unit which helped achieve accurate measurements to improve the physical unit.

Researchers have also shown that a Digital Twin can be used to dynamically reconfigure smart products based on new data. They presented continuous data feedback from physical products to virtual products in (Abramovici, Göbel and Savarino 2017). Also, in (Bitton, et al. 2018), authors have proved that a Digital Twin can be used to drive the cost of Industrial Controls Systems down. And in (Habib and Khoda 2017), authors used hierarchical scanning of data in the digital twin for additive manufacturing, they focused on the data structure as they implemented an API platform to show the data flow between digital and physical twins.

Further, the research in (Um, Weyer and Quint 2017), proposes how engineers set up the digital environment automatically by using a common data model inspired by the Digital Twin concept. Its instantiation for several assembly modules based on Automation Machine Learning (ML) consists of a set of engineering data and relevant models to support a proof-of-concept of Plug-and-Simulate within a modular, multi-vendor assembly line.

Authors in (Cai, et al. 2017), have used sensor data and information fusion to create digital twins for cyber-physical manufacturing. They utilized the sensor data to extract the machining characteristics profiles of a digital-twins machine tool, with which the tool can better reflect the actual status of its physical counterpart. Researchers in (Botkina, et al. 2018), have implemented a digital twin for a cutting tool and collected data throughout the product life cycle, and showed improvement in the physical tool production as a result.

The AUTOParkit System

AUTOParkit is a fully automated parking garage. It is oftentimes referred to as an Automated Parking System or Autonomous Parking Garage or Robotic Parking System or High-Density Parking. It requires no operator to park *(store)* and return (*retrieve*) vehicles. It operates upon user request. AUTOParkit is classified as a pallet-based rack and rail system and does not require intermediate floors. The rack structure is a steel bolt and frame grid system that can be used as the building structure or attached to the building's structure.

AUTOParkit uses a modular-scalable architecture. Each installation is built from a standard set of subsystems to deliver an optimized solution for a specific site. There are only 12 subsystems comprised in the AUTOParkit System that provide limitless design possibilities.

An AUTOParkit System is very simple to operate; users drive their vehicles through a traditional garage door into a Load Bay (user-accessible drop-off/pick-up area) and onto a pallet (an indivisible platform that holds a single vehicle), turn off their car, walk a few short steps to a Park Kiosk where they activate the AUTOParkit System or use the mobile APP. Then their car is safely stored away by an industrial, commercially adapted high-tech, distributed control system. Once parked, cars stay secure in the unoccupied rack structure. Retrieval is also simple; the user returns to a Retrieve Kiosk or mobile APP to request their vehicle. Moments later the car is delivered to a Load Bay with the vehicle nose out, ready to drive away.

The AUTOParkit System is a collection of subsystems (Load Bays, Lifts, Shuttles, Stackers, Stall Conveyors, etc.) that work together to store and retrieve vehicles efficiently. Figure (2) shows an example AUTOParkit System.

The AUTOParkit system is operated by a Main Programmable Logic Controller (PLC) which executes the main program (Algorithms) that orchestrate the movement of the various subsystems to achieve a store or a retrieve for a vehicle. Certain subsystems are equipped with a dedicated PLC that executes the subsystem's movements based on high-level instructions from the Main PLC. The fundamental premise is that all subsystems are independent of one another allowing for the concurrency of movement that contributes significantly to the overall system throughput. When a vehicle is passed from one subsystem to another, there is a Safe Sychnronoized Exchange (SSX) using a series of interlocks integrated into the machine safety circuit. The concurrency of movement contributes mightly to the complexity of the main algorithm to make real-time decisions on machine performance.

Implementation

The Digital Twin starts life as a Digital Twin Prototype (design phase). Then the Digital Twin Instances are created for the product during the realization phase, and the accumulation of the Instances forms the Digital Twin Aggregate (Kritzinger, et al. 2018). We adopted this method and we started by creating Digital AUTOParkit Prototype which included creating an entity to represent each subsystem, then we moved on to the next phase and we create Digital AUTOParkit Instances for the template example that is considered for this research. Finally, a Digital AUTOParkit Aggregate was demonstrated with the integration of the instances to the controller.

Description of an example AUTOParkit system.

The design of an AUTOParkit (API) system starts with the system architecture, in which the number of the subsystems is determined (System Selection) as well as their respective locations (System Configuration). The location is defined as a point on a 3-D axis system, (Level, Column, Row). The System Configuration will also set the range of movement, the direction of movement, accessible locations, and the neighboring subsystems for every subsystem. Using this modular System Configuration, a finite set of possible store (park) paths from each Load Bay to a Stall

Conveyor as well as all possible retrieve (return) paths starting from each Stall Conveyor to a Load Bay can be generated.

For this paper, we selected an example AUOParkit System, as shown in Figure 2, the system consists of 2 Load Bays, 2 Lifts, 6 Shuttles, 1 Stacker, 6 Queue Conveyors, and 30 Stall Conveyors for 105 parking stalls.

The system has three levels (B1, B2, B3) below grade for parking, each level has 7 columns and 7 rows, and 2 Shuttles. Columns 1 to 3 represent the left parking stalls, column 4 is the Shuttle Lane, and columns 5 to 7 are the right parking stalls. The Shuttle lane is 7 rows long. The Load Bays are on the ground level and the Lifts move Pallets vertically from the Load Bays down to the parking levels and vice versa. Each Lift has a Queue Conveyor on every level that connects the Lift to the Shuttle lane. The Stacker (a subsystem that is used to pack pallets in bundles when the system needs to create empty stalls, then unpack these pallets as they are needed) is located on the first parking level (B1) at row 1 and column 5.

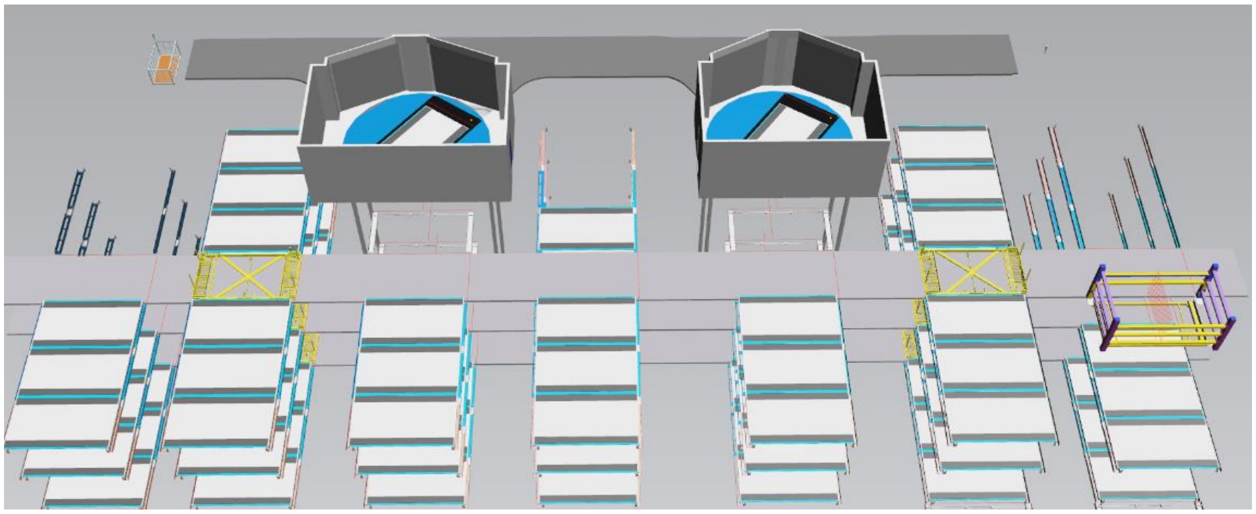

Figure 2 shows a sectional view of the example AUTOParkit system with 2 Load Bays, 2 Lifts, 6 Shuttles, and a Stacker. Each Shuttle lane has 12 cell zones for parking vehicles

Table (1) below shows the home locations for each subsystem and movement range.

Subsystem	Home location	Direction of motion	Motion range
Load Bay 1	(0,2,3)	No motion	No motion
Load Bay 2	(0,2,5)	No motion	No motion
Lift 1	(0,2,3)	Vertical (Axis 1)	From 0 to -3
Lift 2	(0,2,5)	Vertical (Axis 1)	From 0 to -3
Shuttle 1	(-1,4,1)	Horizontal (Axis 3)	From 1 to 6

Shuttle 2	(-1,4,7)	Horizontal (Axis 3)	From 7 to 2
Shuttle 3	(-2,4,1)	Horizontal (Axis 3)	From 1 to 6
Shuttle 4	(-2,4,7)	Horizontal (Axis 3)	From 7 to 2
Shuttle 5	(-3,4,1)	Horizontal (Axis 3)	From 1 to 6
Shuttle 6	(-3,4,7)	Horizontal (Axis 3)	From 7 to 2
Powered Conveyor 1	(-1,3,3)	Conveying only (Axis 2)	From 3 to 3
Powered Conveyor 1	(-1,3,5)	Conveying only (Axis 2)	From 3 to 3
Powered Conveyor 1	(-2,3,3)	Conveying only (Axis 2)	From 3 to 3
Powered Conveyor 1	(-2,3,5)	Conveying only (Axis 2)	From 3 to 3
Powered Conveyor 1	(-3,3,3)	Conveying only (Axis 2)	From 3 to 3
Powered Conveyor 1	(-3,3,5)	Conveying only (Axis 2)	From 3 to 3
Stacker 1	(-1,5,1)	Conveying and packing (Axis 2)	From 5 to 5

Notice that, each level on this system has a different height clearance. Also, the parking stalls in the system have different lengths. So, when a vehicle enters either Load Bay, the height, length, and width sensors will measure the vehicle's height, length, and width collectively referred to as the Vehicle Envelope. The readings of the Vehicle Envelope are sent to the Main PLC. The Store algorithm on the controller will assess the possible stalls that can accept the vehicle and the current activity of each subsystem and will select a store path to park the vehicle. Upon a retrieve request, the Retrieve algorithm will locate the vehicle, and based on the location, the algorithm will evaluate all possible paths to both Load Bays and select the fastest path to retrieve the vehicle. Due to the complexity of the store and the retrieve algorithm we proposed building a digital twin and connecting it to the controller so that the developer can debug and optimize the store and retrieve algorithms prior to the physical commissioning.

Digital twin for the example AUTOParkit system

In the design and building of the architecture for the AUTOParkit (API) Systems, a modular approach was taken for the physical equipment and its functions as well as for the programming hierarchy. While the virtual simulation software (Tecnomatix Plant Simulation) that was used to create the digital twin is a powerful tool that can simulate nearly every aspect of AUOParkit System the design plan for the first iteration was kept simple. The goal was to simulate and test the response of the Main PLC to user requests by observing how the Main PLC utilized the equipment available in the system in response to requests, where dictations for movement would be sent to the equipment in a standardized command packet.

Shuttle

The first subsystem equipment type to be created was a Shuttle type. Shuttles are equipped with a mechanical assembly for interacting with the Stall Conveyors that store vehicles while they are in the AUTOParkit System. The collection of Stall Conveyors serviced by a Shuttle is called a Cell Zones (CZ). They are akin to an Automated Guided Vehicle (AGV) of an automated system and are the last equipment to move a vehicle to its final parking stall and the first to handle movements of vehicles that are being retrieved. The functions of Shuttles are loading and unloading from either side and horizontal transportation. Shuttles traverse horizontally on rails which allows for fast (360 fpm), smooth, accurate transfer. The creation of the Shuttle in Plant Simulation was based on an object built into the software - an AVG called a Transporter, which runs on a track. It was a natural fit for the Shuttle, as well as for the Lift and Stacker carriage which are other subsystems to be discussed later. The default graphic could be easily substituted for graphics for each piece of equipment and an instance of the transporter that was generated from the parent type for the Shuttle looked like an authentic version, as demonstrated in Figure 3.

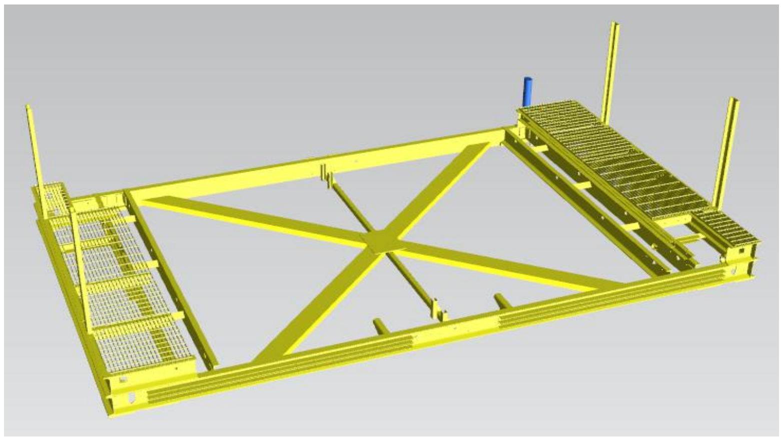
Figure 3 The Shuttle

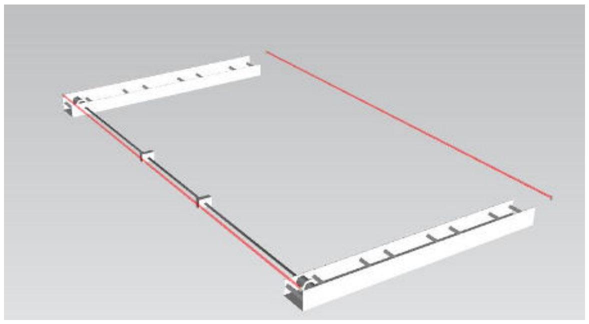

Figure 4 The Cell Zone conveyor

- **Cell Zone**

The CZs that the Shuttle interacts with also called parasitic conveyors, match well with another object built into the software also called a conveyor. Stationary, powered conveyors, as shown in Figure 4, were added with some modifications from the original parasitic conveyors.

- **Floor Zone**

The Shuttles, tracks (Shuttle Lanes), and conveyors are the basic building blocks for a Floor Zone (FZ) of the AUOParkit System. Most AUOParkit Systems have standard spacing for equipment, so building of a FZ with many conveyors and then multiple FZs was easily automated once the spacing parameters were designated. With the structure of the FZs built with all the conveyors and tracks in place, the Simulation could be started where the Shuttles, Pallets, and vehicles, could be generated.

- **Pallets**

The pallets were made from a default object called a container and the vehicles were made from a default object called a part. Work then began to get the digital Shuttles to respond to commands from the Main PLC as they would in the physical twin.

- **Lift**

The Lift was also created from a transporter object, the functions are loading and unloading from either side, vertical transportation, and rotation. The vertical functioning of the Lift took some creative problem-solving to work, the transporter object type is designed well off the shelf for horizontal movement, however, using manipulations that are available in the software the vertical movement was accomplished. The rotation function was a new creation not based on an object found in the program. The graphic for the Lift could be separated into two pieces, the Lift Carriage, and the Lift Turntable. The Lift Turntable graphic was able to be manipulated through animation commands in the program. It was later discovered that the graphics for anything on the Lift would need separate addition animation commands to match the movement of the Lift Turntable, but these were simple to add with the initial rotation animation in place. And these were all local to the Lift type that was created, so once this work was done all called instances of the Lift would have these animations, see Figure 5.

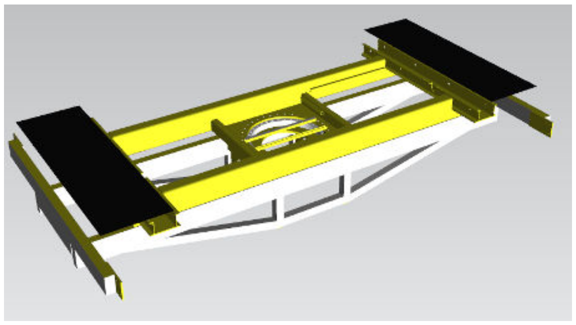

Figure 5 The Lift

- **Load Bay**

The next subsystem that was created was the Load Bay, this was built from a *station-type* object. A station-type did not offer any advantages of being like the Load Bay, apart from being stationary. The Load Bay, Figure 6, is the designated area where users drop off/pick up their vehicles. Animations were added for the Entrance Overhead, Exit Overhead, and Passage Door motion as well as displaying the user indicator light located outside above the Entrance Overhead Door.

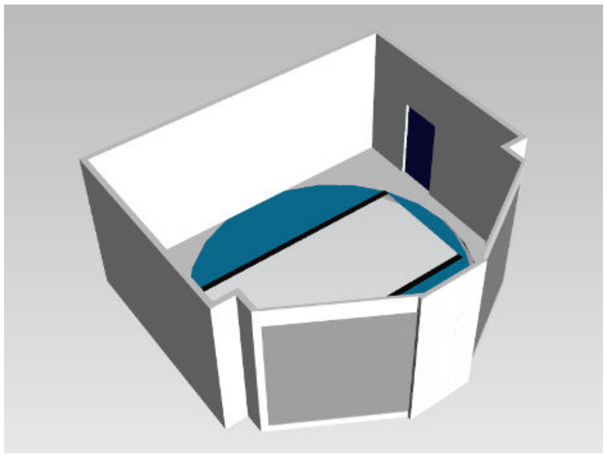

Figure 6 The Load Bay

- **Stacker**

The final subsystem to be created was the Stacker. The Stacker has the functions to (i) load a single pallet, (ii) unload a single pallet, (iii) load a bundle of (4) pallets, (iv) unload a bundle of (4) pallets, (v) vertical transport of all pallets in the Stacker Rack. The Stacker is used as an efficient place to grab and dump a pallet. It has the capacity to hold up to twelve pallets. It stores them in groups of four called bundles that can also leave the Stacker and be dumped and grabbed from in the system to conserve space. The Stacker was created using two objects, it is a transporter on a vertical track like the Lift, but it doesn't have the ability to store pallets, so a Store type object was used to pack and unpack pallets from. These two objects work together to complete the functions of the Stacker, see Figure 7.

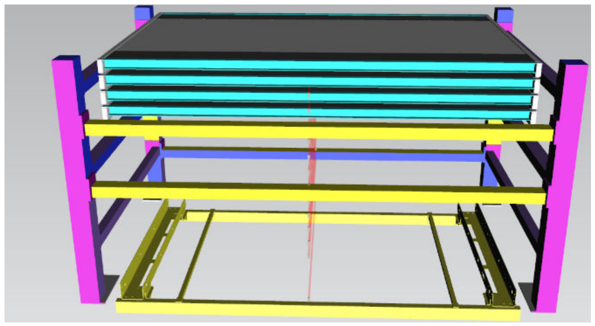

Figure 7 The Stacker

With all these Subsystem types created and the Entire system layout complete, full system testing was ready. Each subsystem was tested individually and responded to the commands coming from the Main PLC.

PLC connection

The MCP houses the Main PLC that orchestrates the entire AUTOParkit System by interfacing with every subsystem to coordinate all motion sequences for all transactions. The Main PLC also interfaces with the System Server PC housed in the Micro Data Center (MDC) for GarageFloor24™ (SCADA), remote internet access, and database storage. Finally, the Main PLC also interfaces to Digital AUTOParkit through the ProfiNet, an industrial LAN, to run the simulation scenario and collect information that will be used in commissioning the physical subsystem. The network connection topology is described in Figure 8.

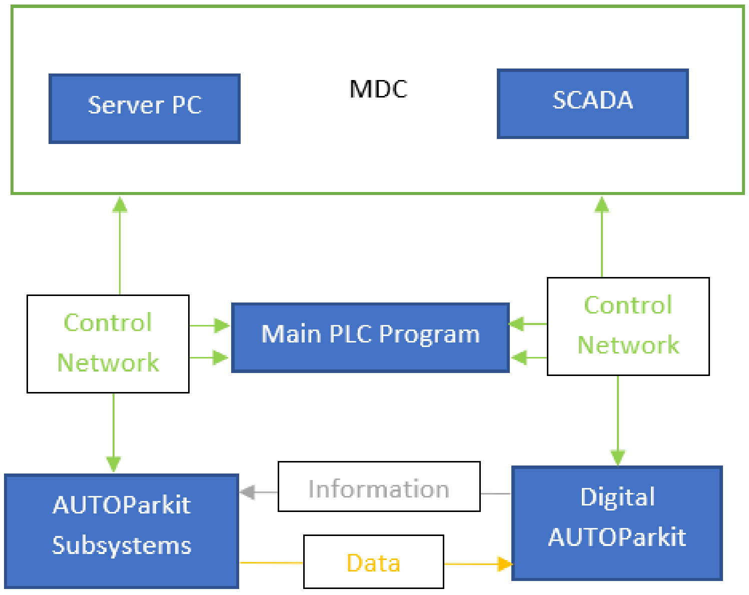

Figure 8 illustrates the network architecture for the AUTOParkit System with its Digital Twin

The Digital Twin here can substitute the various Subsystems; so, the MCP will be interfaced with the Digital AUTOParkit. This will enable digital AUTOParkit to execute the actual program that is written to operate the physical AUTOParkit. Later, the information collected from the Digital AUTOParkit System can be used to improve the Physical AUTOParkit System, and the real data coming from the Physical AUTOParkit System be sent to the digital twin for simulation and processing.

The Selected PLC controller for this project is SIMATIC S7-1500, CPU 1518-4 PN/DP, this controller has a central processing unit with 6 MB work memory for program and 60 MB for data, 1st interface: PROFINET IRT with 2-port switch, 2nd interface: PROFINET RT, 3rd interface: Ethernet, 4th interface: PROFIBUS, 1 ns bit-performances. We interfaced the controller to the PC that runs the digital system via the first interface ProfiNet IRT.

Results

The AUTOParkit System can be installed in a standalone garage or integrated into a commercial or residential building. Based on the building type we can develop a profile for the users' requests to the AUOParkit System.

We assume that the example AUTOParkit System of this paper is to be installed in a multi-family building; hence the system will be operational 24/7. We expect that most of the tenants will retrieve their vehicle early in the morning to go to work/school, which will make the morning time of the day is high retrieval activity time also known as the *morning peak hour demand*. We also expect that tenants who left early to work will come back home late in the afternoon/early evening, accordingly, we will mark this time of the day as high store activity time also known as the *afternoon peak hour demand*. Finally, there are other minor peak times. For example around noon time, some of the tenants who stay at home may go out for lunch or shopping. And during the evening time, we expect some of the tenants may go out to dinner or night activities, which will mark the late afternoon and late evening times as mixed activity times.

From the previous analysis, we conclude that to accurately commission this system we need to construct 3 scenarios of activity, (i) the *Store Scenario*, (ii) the *Retrieve Scenario*, and (iii) the *Mixed Scenario*.

Retrieve scenario

In the Retrieve Scenario, we assume that 60% of the tenants leave the building in the morning. Based on the assumption above we expect that the system will receive 60 retrieve requests every morning between 5 AM and 9 AM, we also use a normal distribution for these requests that span the 4 hours window with the peak at 7 AM, see Figure 9.

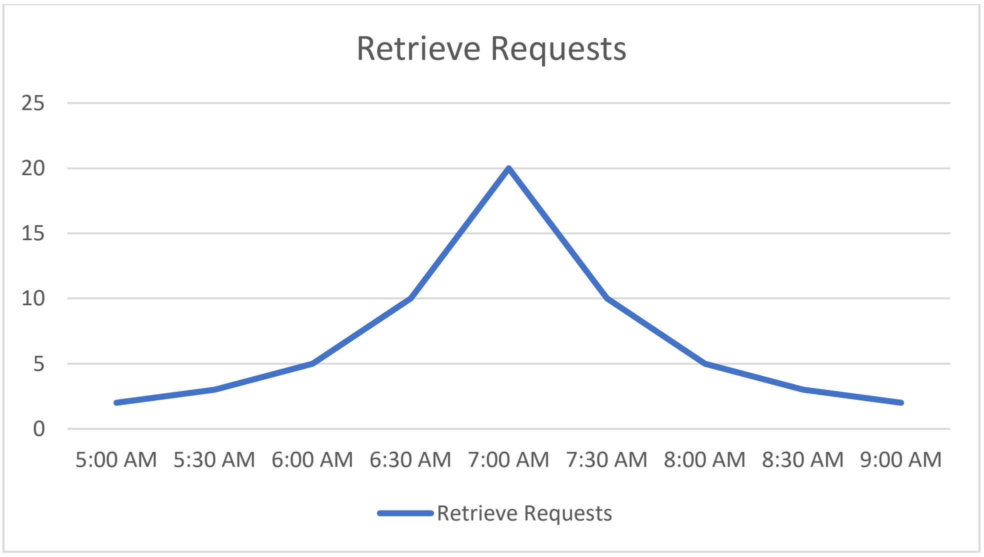

Figure 9 Shows the retrieve requests distribution during the high retrieve activity time

We started the simulation at 5 AM with a full system of 100 vehicles parked, see Figure 10. Next, we triggered the retrieve scenario, using a random number generator to select the vehicle to be retrieved. The system retrieved vehicles as the requests were incoming. At 7:20 AM the system came to a deadlock situation and required the developer to intervene. The simulation caught 39 logic bugs in the MCP program. The developer worked 2 weeks (80 hours) to fix these bugs.

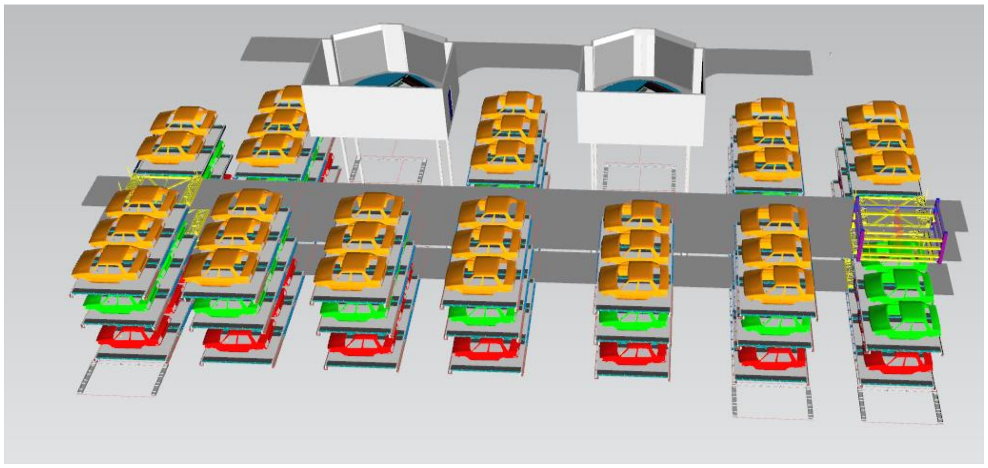

Figure 10 The start of the Retrieve Scenario, Full system

We re-ran the simulation again after fixing the 39 logic bugs exposed by the Retrieve Scenario simulation. The retrieve scenario was finished at 9:17 AM completing all 60 retrieves without intervention.

Store scenario

In the Store Scenario, we assume that 60% of the tenants come back to the building in the late afternoon until early evening times. Based on this assumption we expected that the system will receive 60 store requests every afternoon between 3:00 PM and 7:00 PM, we also use a normal distribution for these requests that span the 4 hours window with the peak at 5:00 PM, see Figure 11.

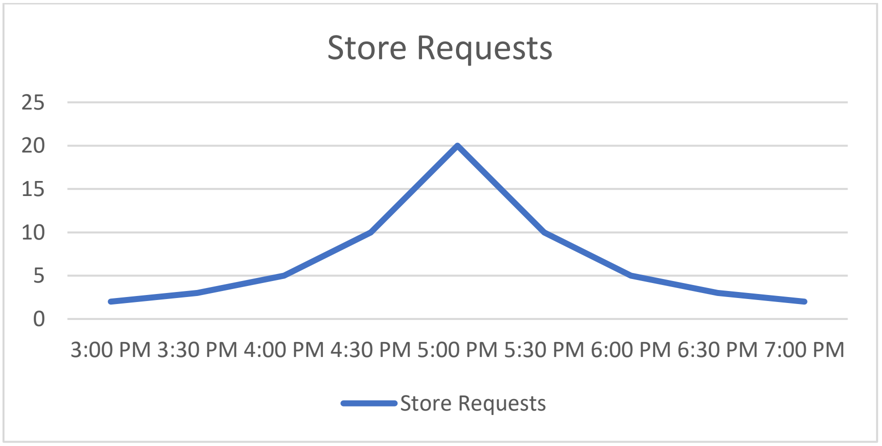

Figure 11 illustrates the store requests distribution during the high Store activity time

We started the Store Scenario simulation at 3:00 PM with an empty system, 0 vehicles parked, as it is shown in Figure 12. Ne text we triggered the store scenario, using a random number generator on the time between vehicle arrivals. The system did far better than the Retrieve Scenario because some of the bugs were already fixed in the Retrieve Scenario. However, the simulation still was not able to complete the scenario organically and caught 21 logic bugs in the MCP program. The developer worked for 1 week (40 hours) to fix these bugs.

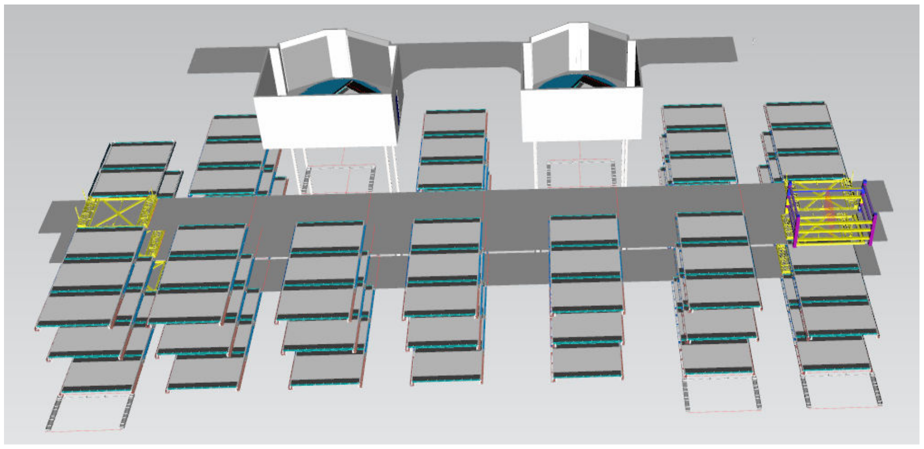

Figure 12 The start of Store Scenario, Empty System

After a complete debug and the Store Scenario, we re-ran the simulation again. The Store Scenario finished at 7:09 PM completing all 60 store requests without an intervention.

Mixed Scenario

In the mixed scenario, we assume 20% of the tenants go out during the late morning/ early afternoon hours for either lunch or shopping while the last 20% of the tenants leave the building later evening for dinner or night shift. Based on this assumption, we constructed Mixed Scenario-1 with 20 retrieve requests and 20 store requests between 10:00 AM and 2:00 PM and Mixed Scenario-2 with 20 retrieves and 20 stores from 8:00 PM to 4:00 AM. The number of bugs digitally found on Mixed Scenario-1 was 9 bugs while in Mixed Scenario-2 we found 3 bugs. All bugs were rectified in 3 working days (24 hours).

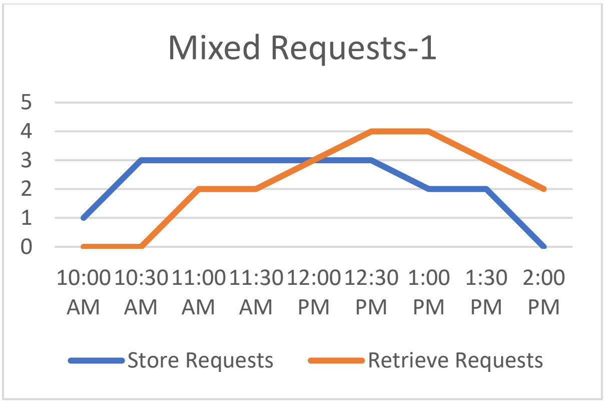

Figure 13 Mixed activity 1

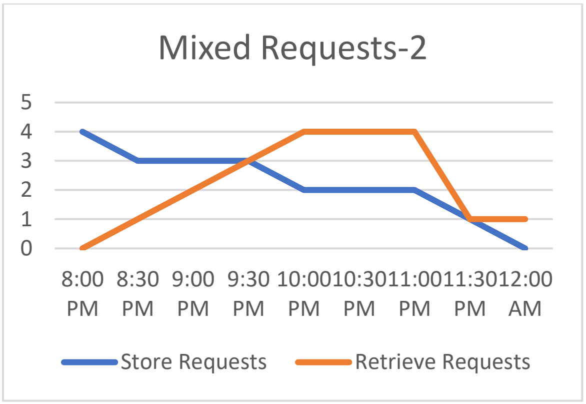

Figure 14 Mixed activity 2

Cost of Physical Commissioning

The physical commissioning is challenging for this project for a variety of reasons: the controls engineer needs to be on-site while debugging the main program, additional assistants are required to park and retrieve vehicles, and finally, we need vehicles available during the commissioning time. We based the results on a similar project that was commissioned in 2019 and it took 3 engineers spending approximately 4 months on-site to fully commission the project. 3 months were spent on debugging the Main Program, while 1 month was required for controls on-site acceptance testing. The Retrieve Scenario required 240 hours of debugging, while the Store Scenario required 160 hours with the Mixed Scenario-1 debugged in 50 hours and lastly, Mixed Scenario-2 required 30 hours of debugging. The cost of physical commissioning reached up to $160,000.

Cost of Digital Commissioning

The Main Program was fully commissioned digitally in less than 3-weeks. The total number of hours spent to fix all bugs was 144 hours. The cost of digital commissioning was calculated based on the average hourly salary for senior Controls Engineer ($50/ hour), with a total of $7200. After the system is digitally commissioned, we expect that engineers will need to be on-site for reduced time, since all program debugging was done digitally. However, engineering is required on-site to perform the site acceptance tests only. The estimated time for engineers on-site is one month with a cost of $40,000.

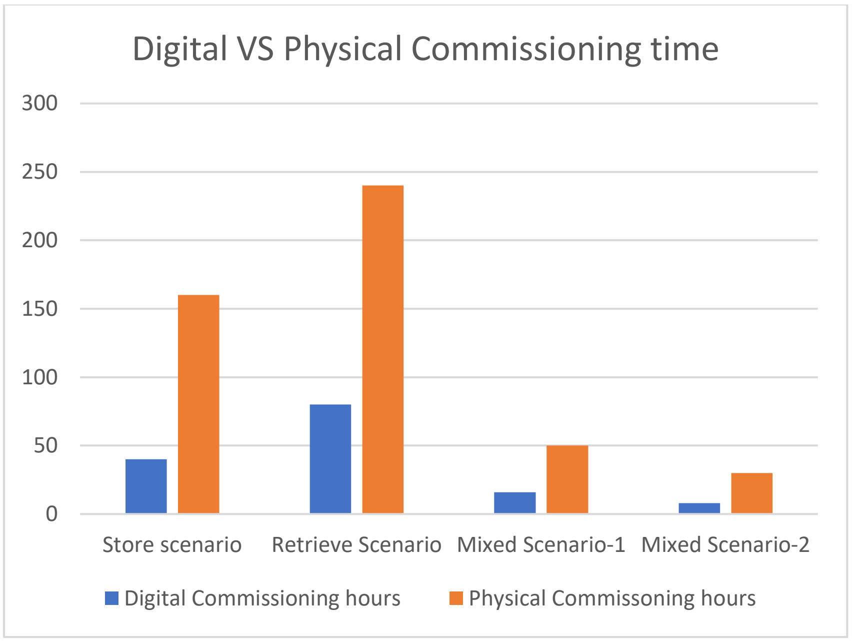

Figure 15 Digital commissioning hours VS Physical commissioning hours

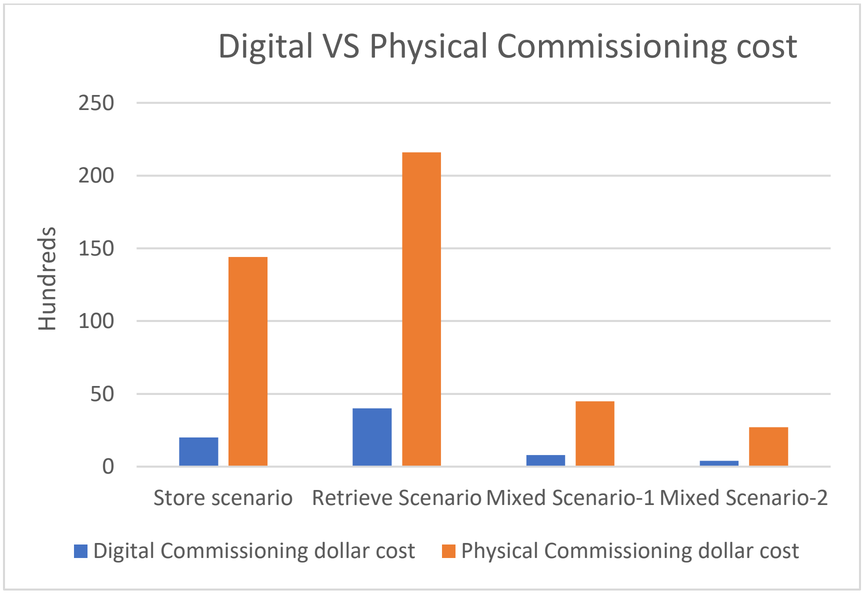

Figure 16 Cost of commissioning digitally VS physically for all scenarios

Conclusion

We proposed the use of the digital twin concept to improve the design, programming, commissioning, and operation of the AUTOParkit System. We reviewed previous work on building a digital twin in an automation domain then we focused our study to provide a baseline of the AUTOParkit System. A Physical AUTOParkit System detailed description was presented as an example as well as the implementation of the Digital AUTOParkit System counterpart using Tecnomatix plant simulation. The physical controller of the AUOParkit System was connected to the Digital API. Four real-life scenarios were created to test the Main Program of the Digital AUTOParkit System, and 72 bugs were exposed in the Main Control Program and fixed. We were able to show that the use of a digital twin resulted in a cost of 144 engineering hours to fully debug the main program with virtual commissioning versus 480 engineering hours for physical commissioning.

Our next focus is to study user behaviors on an actual operating AUTOParkit System and use the data on the digital AUTOParkit System to predict user behaviors and formulate an algorithm that anticipates user requests and re-organizes the vehicle position inside the system accordingly to further improve the overall system throughput.

References

Abramovici, M., J. Göbel, and P. Savarino. 2017. "Reconfiguration of smart products during their use phase based on virtual product twins." *CIRP Annals.* 165-168.

Bitton, R., T. Gluck, O. Stan, M. Inokuchi, Y. Ohta, Y. Yamada, T. Yagyu, Y. Elovici, and A. Shabtai. 2018. "Deriving a cost-effective digital twin of an ICS to facilitate security evaluation." *European Symposium on Research in Computer Security.* 533-554.

Boschert, S, and R Rosen. 2016. "Digital Twin—The Simulation Aspect." In *Mechatronic Futures*, by P. Hehenberger and D Bradley. Switzerland: Springer, Cham.

Botkina, D., M. Hedlind, B. Olsson, J. Henser, and T. Lundholm. 2018. "Digital twin of a cutting tool." *Procedia CIRP* 215-218.

Cai, Y., B. Starly, P. Cohen, and Y.S. Lee. 2017. "Sensor data and information fusion to construct digital-twins virtual machine tools for cyber-physical manufacturing." *Procedia Manuf* 1031-1042.

Grieves, M. 2014. "Digital twin: manufacturing excellence through virtual factory replication." *White Paper* 1-7.

Habib, M. Ahasan, and B. Khoda. 2017. "Hierarchical Scanning Data Structure for Additive Manufacturing." *Procedia Manufacturing* (NAMRC 45) 1043-1053.

Jones, D., C. Snider, Nassehi, A., J. Yon, and B Hicks. 2020. "Characterising the Digital Twin: A systematic literature review." *CIRP Journal of Manufacturing Science and Technology* 36-52.

Kritzinger, W, M Karner, G. Traar, J Henjes, and W Sihn. 2018. "Digital Twin in manufacturing: A categorical literature review and classification." *IFAC-PapersOnLine* 1016-1022.

Lohtander, M., N. Ahonen, M. Lanz, J. Ratava, and J. Kaakkunen. 2018. "Micro manufacturing unit and the corresponding 3D-model for the digital twin." *Procedia Manuf* 55-61.

Tao, Cheng, Qi, Zhang, and Sui. 2017. "Digital twin-driven product design, manufacturing and service with big data." *Int J Adv Manuf Technol* 3563-3576.

Um, J., S. Weyer, and F. Quint. 2017. "Plug-and-simulate within modular assembly line enabled by Digital Twins and the use of AutomationML." *IFAC-PapersOnLine* 15904-15909.

Yun, S., J. Park, and W. Kim. 2017. "Data-centric middleware based digital twin platform for dependable cyber-physical systems." *Ninth International Conference on Ubiquitous and Future Networks.* 922-926.

Terms

Term	Definition
Android	Open source Operating System created by Google for mobile devices.
AppSync	An application development service hosted in the Amazon Web Services (AWS) public cloud that synchronizes data for mobile and web apps in real-time.
APS	Automated Parking System
AUTOParkit APP	An application executing on a mobile device to allow a user to interface with an AUTOParkit System.
AWS	Amazon Web Services
Bluetooth	Bluetooth is a short-range wireless technology standard that is used for exchanging data between fixed and mobile devices over short distances using UHF radio waves in the ISM bands, from 2.402 GHz to 2.48 GHz and building personal area networks.
C#	C Sharp (C#) is a general-purpose, object-oriented programming language.
Cognito	Amazon Web Services (AWS) product that controls user authentication and access for mobile applications on internet-connected devices.
CRUD	Create, Read, Update, Delete, and basic operations to handle data.
Display Language	The human-readable language used by a mobile phone
FIFO	First In First Out
GarageFloor24.com	Official Website of GarageFloor24 LLC that provides access for user support, maintenance, and monitoring services for AUTOParkit Systems.
GraphQL	An API Query based language see: https://graphql.org/
HTTP	Also known as "Hypertext Transfer Protocol", accepted two-way communication between a web browser and server for websites
iOS	iPhone's Operating System
Key Fob	A small security hardware device that electronically holds a credential that is used to request vehicle retrieval, user identification, and door access. The credential is unique across all AUTOParkit Systems.
Lamda	AWS Lambda is a serverless compute service that runs your code in response to events and automatically manages the underlying compute resources for you.
MDC	Micro Data Center is a single IT cabinet that contains IT equipment used to service the AUTOParkit System. These devices included but are not limited to servers, network switches, UPS, network router/modem/firewall, Network Video Recorder (NVR), and KVM
Middleware	Middleware is computer software that provides services to software applications beyond those available from the operating system. It can be described as "software glue".
NFC	Near-field communication is a set of communication protocols for communication between two electronic devices over a distance of 4 cm (11⁄2 in) or less.[1] NFC offers a low-speed connection with a simple setup that can be used to bootstrap more capable wireless connections.
No-SQL	Database type that is unstructured, unlike a SQL database where the columns of an item strictly adhere to a table design. NoSQL databases are purpose-built for specific data models and have flexible schemas for building modern applications. NoSQL databases are widely recognized for their ease of development, functionality, and performance at scale. They use a variety of data models, including document, graph, key-value, in-memory, and search.
OPC	OLE for Process and Control. OPC is a software interface standard that allows Windows programs to communicate with industrial hardware devices.
OPC-UA	OPC Unified Architecture is a machine-to-machine communication protocol for industrial automation developed by the OPC Foundation.
PLC	Programmable Logic Controller
Profinet	Industrial LAN using Ethernet with TCP/IP and Profibus Application Layer.
RC	Retrieve Cancel

RR	Retrieve Request
RRP	Retrieve Re-Park
SCADA	Supervisory Control and Data Acquisition
WinCC	A supervisory control and data acquisition (SCADA) and human-machine interface (HMI) system from Siemens.