BeTwin: Enhancing VR Experiences with BLE Beacon-based Digital Twins

Andrei George Rosu*, Anderson Augusto Simiscuka†, Mohammed Amine Togou‡ and Gabriel-Miro Muntean§
School of Electronic Engineering, Dublin City University, Dublin
E-mails: *andrei.rosu2@mail.dcu.ie, †anderson.simiscuka2@mail.dcu.ie, ‡mohammedamine.togou@dcu.ie, §gabriel.muntean@dcu.ie

***Abstract*—Bluetooth Low Energy (BLE) is one of the technologies that can be used for short-range communications. BLE beacons are small wireless devices that can store URLs and real-world data, including GPS locations and identifiers. This article proposes BeTwin, an efficient solution that integrates BLE beacons with a digital twin environment in order to enhance and customise a virtual 3D platform, bridging the real and virtual worlds. The beacons are used to bring information from real objects into the virtual platform. We extend the Eddystone messaging protocol and investigate the impact of beacon distance, energy levels, and number of beacons on the system performance. We also describe a testbed consisting of a virtual environment, a beacon reader along with beacons, and assess its performance in terms of communications latency, Received Signal Strength Indicator (RSSI), and number of beacons.**

***Index Terms*—Digital Twins, Beacons, BLE, Virtual Reality**

I. Introduction

Bluetooth Low Energy (BLE) beacons have been used in a wide range of contexts over the past years, from tracking devices within a building to sending push notifications containing promotions to customer phones in supermarkets. BLE beacons use the Eddystone protocol that allows the beacon to continuously transmit short messages. Nearby Bluetooth-enabled devices can retrieve these messages and use the info for different purposes [1]. BLE beacons are also able to store important data about real-world objects such as unique identifiers and location.

Digital Twinning (DT) [2], [3] refers to having an interactive representation of physical objects, systems or processes in a virtual world. The twins constantly share and synchronise information, including object properties and status data [4], [5]. Creating DT environments, however, can be a challenging task as it involves 3D immersive scenarios, which can be very complex, with multiple objects and systems.

Authors in [6] acknowledge that high-quality digitalisation of virtual objects requires a comprehensive mechanism for real-time rendering. The work also points that there is a need for novel approaches in relation to data exchanges between physical twin systems with the digital twin counterparts, requiring synchronisation of device states with low communication traffic and delays. Li et al. [7] discussed the lack of techniques for automatic resource representation during device registration, with physical details of digital objects being added to DT environments manually.

DT platforms can take advantage of BLE beacons for addressing these issues, thanks to their ability of storing and communicating data related to physical objects into digital systems. When users acquire a new device, system or object compatible with the platform, beacons can easily update the DT environment with the new virtual counterparts synchronised with the physical objects details. The beacons also provide security measures in relation to traceability of objects as they support unique identifiers and geolocation features.

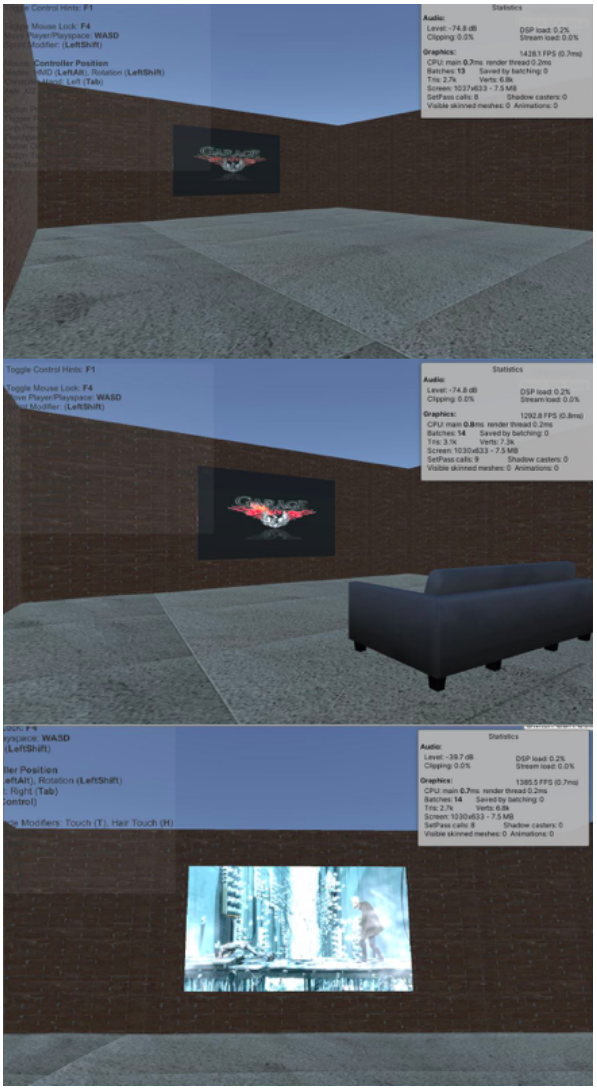

Fig. 1. BeTwin's digital twin environment with added objects.

This paper proposes *BeTwin*, an innovative mechanism which employs beacons to easily add and remove objects in a real-virtual world twinned context. The VR environment is shown in Fig. 1. The beacons can also be used to customise

existing virtual objects, such as modifying rich media content (e.g., images, videos, and audio) being displayed. The BeTwin mechanism is instantiated and tested using BLE beacons in a real-life DT platform. The testbed considered in this paper contains multiple beacons, a Raspberry Pi as the beacons' controller, and a VR server hosting the DT environment.

Thorough testing investigates the time and latency for generation of virtual objects with different number of beacons indicating that BeTwin supports seamless and near real-time experience.

The rest of the paper is organised as follows. Section II describes the technology-enablers of BeTwin, while Section III discusses its design. Section IV describes the testbed implementation. Results are presented in Section V. Conclusions and directions for future work end this paper (Section VI).

II. Background and Related Work

This section describes the protocols, approaches and tools employed in the design of the proposed BeTwin solution.

Bluetooth is a wireless communication protocol that uses the 2.4 GHz frequency to transmit data at a rate of up to 2 Mbps with its newer version. The effective range is between 10 and 100 metres, depending on the transmitted power [8]. Bluetooth employs the Attribute Protocol (ATT) and Generic Attribute Profile (GATT) to handle connections [9]. In addition, it uses the Generic Access profile (GAP) to advertise data without a connection and to make a device discoverable by other Bluetooth devices [10]. Bluetooth Low Energy (BLE) is a version of Bluetooth that was designed for low power consumption. As a result, its data rates are also low (i.e., maximum advertised data rate for BLE is 1 Mbps while maximum advertised data rate for classic Bluetooth is 2 Mbps). BLE is in a constant state of sleep, waking up only when a connection is established [11]. It is mainly used in communications where there is not a significant amount of data to be exchanged between devices. Another important feature of BLE devices is that they can survive for up to five years on the same battery, without being recharged. This makes BLE suitable for beacons and sensors [12].

BLE beacons are small devices that are used to continuously transmit a radio signal containing information such as beacon's UID and other service data. These devices have typically been used in positioning systems or proximity marketing (sending promotions to customers' phones in supermarkets). They can live on the same battery for a very long time, but when the battery dies, the beacon must be replaced as batteries cannot be changed or charged. Different BLE beacons have different expected battery life [13], [14]. They are naturally in a sleep state, and they wake up every time there is a need to broadcast information. The advertising interval can be changed by the user, affecting the beacon's battery life. The shorter the advertising interval, the faster other devices will retrieve the advertised data. The default TX power set on the beacon also affects battery life [15].

Eddystone is a Google protocol for beacons. It broadcasts service messages with a service ID and the corresponding service data [16]. There are four types of Eddystone frames defined: Eddystone-UID, Eddystone-URL, Eddystone-TLM and Eddystone-EID. The first is used to advertise a user-set 16-byte ID while the second is used to advertise a URL set by the user. Eddystone-TLM is used to advertise telemetry data, such as beacon battery voltage and beacon temperature whereas Eddystone-EID is used to advertise an encrypted ephemeral ID that changes at a rate set by the user [17]. All frames share the same service ID. They can all be used on one single beacon, but the service data for the next type is sent every other advertising cycle. Assume we want to transmit Eddystone-UID, Eddystone-URL, and Eddystone-TLM. In the first advertising cycle, Eddystone-UID will be sent. Then, the Eddystone-URL frame will be sent on the next advertising cycle. Finally, the Eddystone-TLM frame will be transmitted in the following advertising cycle. Note that even though Eddystone is not currently supported by Google, it is still suitable for the proposed prototype of the platform.

Unity is an IDE mainly used for creating 3D applications or games for different platforms. Unity has also been used for the creation of a number of digital twin platforms [18], [19]. A Unity application is composed of scenes, representing a level, a map or a menu. A scene contains several 2D and 3D objects, or empty objects that have scripts attached to them. These objects can be added, removed, or managed using the Unity Editor. Most of the interaction in the application is done using scripting, preferably in C. Scripts must be attached to an object to work and can be used to manipulate any other object from the scene [20].

Based on the features of BLE, Eddystone and Unity, it is possible to design and implement a digital twin platform and test the mechanism proposed in this paper. BeTwin aims to simplify the manual process of adding, removing and customising objects in the virtual environment with the use of beacons representing real devices.

III. BeTwin Solution Design

The architecture of BeTwin, illustrated in Fig. 2, consists of these main components: physical objects (e.g., smart devices, appliances, furniture, machinery) and their individual BLE beacons, a beacon reader, a 3D environment with the virtual objects and a visualisation device (e.g., VR headset, computer screen).

A. Beacon-based Physical Devices

As mentioned in Section II, Eddystone is a suitable messaging structure for BLE beacons. The Eddystone-URL protocol advertises a message set by the real-world object related to that beacon. Bluetooth devices can read this message and take action. BeTwin addresses challenges imposed by the beacon technology, making it feasible to use them in the platform: First, a messaging solution is proposed to work with Eddystone-URL 18 characters limit. Second, as VR headsets along with some computers are not equipped with Bluetooth adapters, or they are placed in areas that may be too distant from the beacons, a beacon reader solution is introduced.

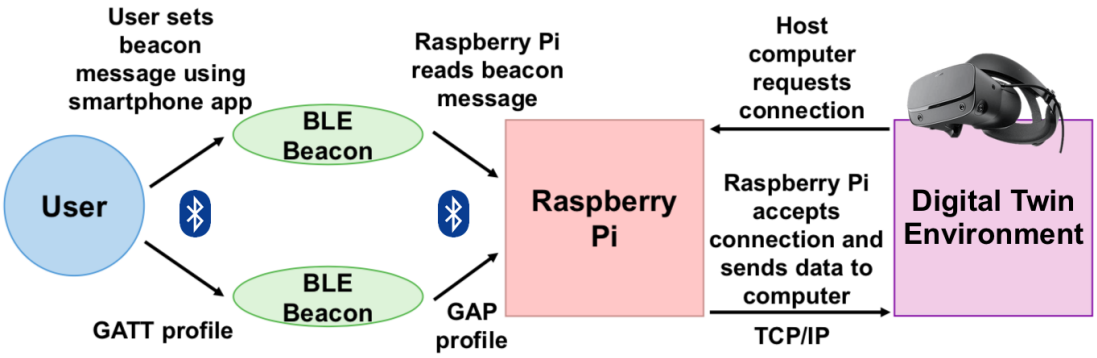

Fig. 2. BeTwin solution architecture.

Finally, communications between BLE beacons and the reader along with communications between the reader and the VR headset or VR server (i.e., the device that renders the digital twin environment) are performed in a timely manner.

B. Virtual Environment

In BeTwin, beacons are able to bring and remove physical objects into and from the virtual world and perform media-related functions. They can also give access to and install complex systems, create Internet of Things (IoT) virtual representations, and add or remove objects in a smart city simulation environment.

Once a user places a beacon close to the beacon reader, if the beacon contains the name of the object, the VR application adds that object and allows the user to use it. Rich media scenarios, such as virtual representations of smart homes are customised with songs or videos added by the beacons, played on a virtual screen in the VR world (as illustrated in Fig. 1). The virtual world is created in a 3D rendering engine, such as Unity and visualised with a VR headset (e.g., Oculus Rift) or a computer screen.

IV. Testbed Implementation

BeTwin has been implemented on a prototype in order to facilitate testing and evaluation of the solution. A Raspberry Pi is used to retrieve messages from beacons, manufactured by HID Beeks, with the TinyB library [21], as well as the Bluetooth GATT and GAP profiles. The Raspberry Pi is used as the beacon reader due to its portability, price, and Bluetooth signal strength. Smartphones can also be used as the beacon reader; however, the Raspberry Pi is more flexible in terms of development and Bluetooth profiling.

The TinyB library accesses the DBus BlueZ API from Linux and makes it possible to control Bluetooth devices and adapters using a Java program. Once the Raspberry Pi retrieves beacon messages, it forwards them to the server with the digital twin environment. This is done wirelessly via Java TCP sockets in a local network. The VR application will then use these messages to perform actions such as adding, removing or sending different parameters to virtual objects. The Raspberry Pi tracks when a new beacon is detected or removed.

Two new classes were implemented in BeTwin's prototype to retrieve and customise beacon messages, check if messages were updated and if a beacon was removed. The beacons' Eddystone URLs and messages are configured with an Android/iOS application called Beeks Toolkit.

A. Beacon Messaging

The class "BeaconMessaging" handles beacons communication. First, it imports the TinyB "BluetoothDevice" class and adds flags for when a message is updated and for when a beacon is removed. It also stores the beacon's decoded messages. In addition, this class implements the "BluetoothNotification" interface from TinyB, starting a new thread per beacon and allows the messages to be updated asynchronously. These threads are killed when beacons are removed (i.e., the RSSI is lower than a threshold, which is discussed in Section V.A).

Beacon messages are detected within a few seconds (approx. 3s) after a beacon is discovered. However, this depends on the beacon's advertising rate and the adapter's scan interval and scan window. A timeout argument is introduced to allow the search to be stopped if no messages can be found.

The BlueZ API returns the beacons' advertising data in the form of a map having the service ID and its bytes as the pair $\langle key, value \rangle$. The advertising data contains the Eddystone-ID, Eddystone-URL and Eddystone-TLM. Only Eddystone-URL is required for BeTwin's communications. The first byte of the map determines the frame type. If the first byte has a value of 16, it means that the frame type is Eddystone-URL. The service data is retrieved from the beacons using the TinyB library, which checks for the following conditions:

- The map is not empty.
- The map contains the Eddystone service ID as a key.
- The first byte of its value is 16.

When these conditions are met, the value, which is a byte array, is passed to the "resolveEddystoneURL" method that turns it into a string. The byte array structure can be described as follows. The first byte determines the frame type while the second indicates the TX power. The third byte determines the URL scheme (i.e., http or https). Bytes four to 20 represent the part of the message after the URL scheme. To make it possible to send longer URLs, Eddystone-URL encodes top-level domains in the last byte. If a top-level domain is not specified in the message, the last byte will simply be the last character of the message.

TABLE I
TESTBED SETTINGS

Parameter	Value
VR Server Model	Alienware Aurora
Processor	Intel Core i7-8700K CPU 3.70GHz
RAM	32GB
GPU	Intel Integrated HD Graphics
VR Engine	Unity
Real-life Beacons	HID Beeks
Simulated Beacons	Between 2 and 300 TinyB Beacons
Bluetooth Reader	Raspberry Pi 3B

Algorithm 1: Populating the HashMap of BeTwin

```
initialiseList(bluetoothManager, rssiThreshold, timeout)
 {
if bluetoothManager is not in discovery mode then
    start discovery;
    wait for 1 second;
end
foundDevices = get devices from bluetoothManager
 foreach device in deviceList do
    set remove flag to true;
end
foreach device in foundDevices do
    if allowedDevices contains device address then
        if device rssi is bigger than rssiThreshold and
         device rssi is smaller than 0 then
            if deviceList contains device then
                set remove flag to false;
            else
                add device to deviceList;
                set service data for device with timeout;
                enable service notification for device;
            end
        end
    end
end
}
```

Unlike the URL scheme, the top-level domain is not required to be added. Not adding the URL scheme will cause the message to be reset to a default URL. The "resolveEddystoneURL" method also checks the byte array size. If it is shorter than three bytes, it returns an empty string, as it means that there is no message set. Otherwise, it checks if the fourth byte is equal to an apostrophe character. This is a solution employed to allow beacons to transmit plain messages along with URLs, as the Eddystone-URL protocol allows only short URLs to be sent. Therefore the first character is used to indicate the type of message (i.e., string or URL) after "http://". When the Unity application retrieves messages, it treats messages as strings or URLs based on this character.

B. Beacons Management

The "BeaconsManagement" class creates a "BeaconMessaging" object for each beacon discovered by the Bluetooth manager. It has a Java HashMap of "BeaconMessaging" objects. The HashMap "deviceList" contains a key that represents the MAC address of a BLE beacon. The program assumes that all BLE beacons have different MAC addresses.

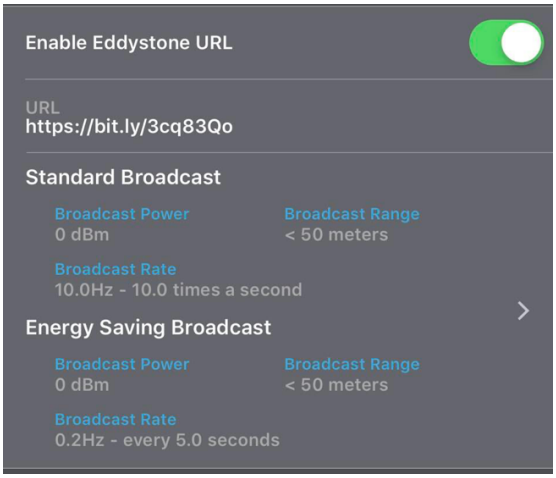

Fig. 3. Eddystone messaging.

Algorithm 1 is included in the "initialiseList" method, which is used to populate the HashMap of devices found by the Bluetooth manager. It only adds the devices that are within a certain range, determined by the "rssiThreshold". The method always checks if discovery mode is turned on at the start. In case it is off, the method turns it on and waits for one second to enable the Bluetooth adapter to start discovering devices. All discovered Bluetooth devices are put in a list and the "remove flag" of each beacon in the "devicesList" is set to true. The method then checks if the RSSI value of each of the discovered devices is higher than the pre-defined threshold and different from 0. BlueZ shows a value of zero decibels when it cannot retrieve the RSSI value of a Bluetooth device. If the RSSI of a Bluetooth device is lower than the threshold or 0 and the corresponding object is already in the "devicesList", the "remove flag" is set to false. If it is not in the "devicesList", a new object is created and stored in the "devicesList". The remove flag is set to true for every object in the "devicesList" at the start as there needs to be a way to know when a device is no longer in range. If the device is either not in range or cannot be discovered anymore, the remove flag will remain set to true. A "remove" message is then sent to the VR application, which removes the objects generated by that beacon or stops any media or customisation related to that beacon, unless they are permanent objects, specified by their unique MAC. The object is then removed from the "devicesList". The thread that updates the beacon messages is also stopped.

The "BeaconsManagement" class enables sending messages to the VR application via TCP sockets using the following format: *Beacon_MAC_Address : message*.

C. Unity Application

A VR Application was developed in Unity with the VRTK library in which the proposed BeTwin solution was deployed. An Oculus Rift VR headset is employed for visualisation. The virtual space is large enough to display several objects, including a screen in which videos and songs are played. Additional objects such as furniture, virtual appliances and

Fig. 4. Testbed - Beacons and Raspberry Pi.

industrial models are loaded once the BLE beacons are near the Raspberry Pi.

3D objects can be saved as prefabs. Prefabs can then be loaded at runtime and cloned using a C script. This script will control what objects are displayed or removed as well as multimedia playback. A separate thread is required to handle the server connection with the Raspberry Pi.

Another important method of the VR application, "ProcessQueueString", is used for processing received messages from Beacons. If the message is a URL, the application plays the multimedia content found at that URL. If the message contains the apostrophe character (as depicted in Fig. 3), indicating it is a simple string, it will generate the object with the same name as the string. If the message is "remove_obj" and an object was generated by the beacon ID that sent this message, the object will be removed. If the beacon originally sent a URL and then sent a "remove_obj" message, the video or song will stop playing.

V. Results and Discussion

The feasibility of using beacons to integrate real objects with their digital counterparts in BeTwin has been investigated in terms of RSSI and distance as well as the time needed for retrieving messages, storing beacons, and generating objects.

A. RSSI, Distance and Message Retrieval Time

In the first test, the distance between a beacon and the Raspberry Pi is increased, so the impact on the RSSI is analysed. Two beacons are placed on the same axis as the Raspberry Pi, with no obstructions between them. Fig. 4 illustrates the beacons placed very close to the Raspberry Pi. Ten measurements were taken and averaged per analysed distance. Fig. 5 indicates that as the distance between the Raspberry Pi and the beacon increases, the RSSI decreases. The RSSI decreases rapidly within the first metre from the Raspberry Pi.

The second test investigates the effect of RSSI on the time required by the Bluetooth adapter of the Raspberry Pi to retrieve the message from one beacon. To change the RSSI, the beacon is gradually moved further away from the Raspberry Pi. Fig. 6 clearly indicates that the lower the RSSI, the longer

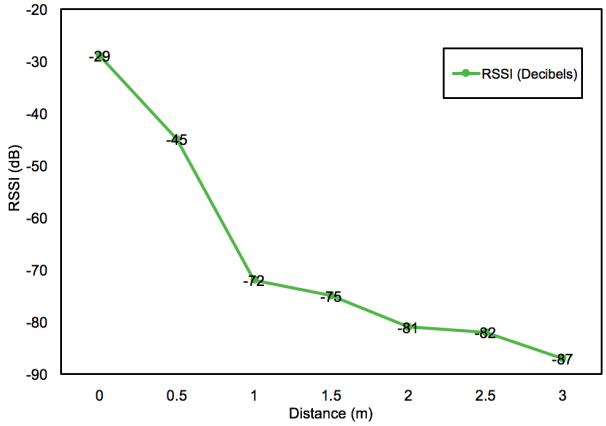

Fig. 5. Distance vs. RSSI.

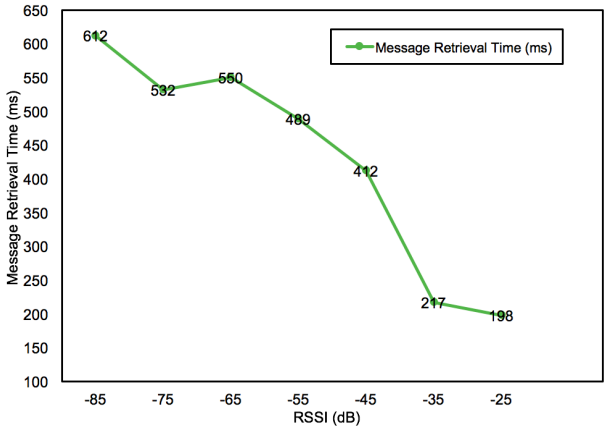

Fig. 6. Message Retrieval Time vs. RSSI.

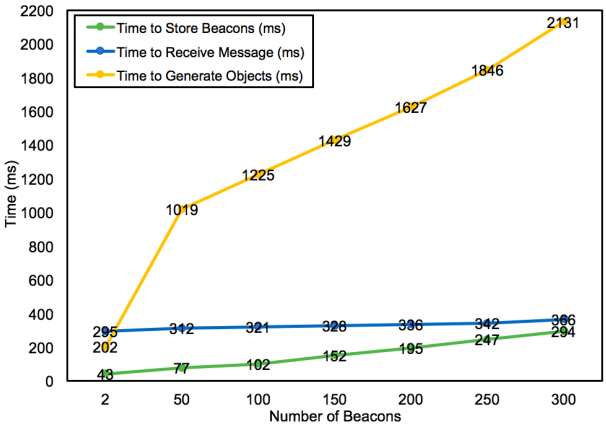

Fig. 7. Number of beacons vs. time to store beacons, receive messages and generate objects.

it takes for the Bluetooth adapter to retrieve data. Distance should not necessarily be a problem for the designed platform, as it is expected that beacons are placed close to the Raspberry Pi (so the RSSI is adequate). Based on the results presented in Fig. 6, we recommend a threshold of -85dB or higher for the HashMap algorithm and the overall detection of beacons, in order to minimise communications delays.

B. Effect of Number of Beacons

The third test verifies the effect of increasing the number of beacons on the time it takes the Raspberry Pi program to

fill the device HashMap and set messages for each device in the HashMap. The high number of beacons tested required a simulation where multiple "BeaconMessaging" objects from the same beacon were created and assigned new MAC addresses to simulate beacon devices. Simulation settings are available in Table I. This test checks the efficiency of the algorithm that decodes the message from the beacon. As shown in Fig. 7, the application itself only needs between 77 and 294 milliseconds to process between 50 and 300 simulated beacons, not including Bluetooth discovery times.

The fourth test investigates how the number of simulated beacons affects the time required to receive all the beacon messages through the Java socket. This test was performed in order to understand how long the VR application takes to get the message from the Raspberry Pi in case of a larger number of beacons. As portrayed in Fig. 7, there is very little difference between the time it needed by the server to send the message from two beacons and the time required by the server to send the message from 300 beacons. This can be explained by the fact that the messages sent are very short: approximately 40 to 50 bytes per beacon. That means approximately 15000 bytes for 300 beacons, which is not a massive considering the network available bandwidth.

The final test verifies how the number of beacons affects the time needed for objects to be rendered in the VR application. As seen in Fig.7, object generation requires substantially more time than the messaging process. The rendering, however, depends on how fast the application receives the data from the Raspberry Pi and how fast it converts messages into strings, splits them and insert them into the queue.

These results show that the slowest part of the system is the object spawning, which is influenced by the time it takes the VR application to receive the message from the Raspberry Pi and process it.

VI. Conclusions and Future Research Directions

This paper presents BeTwin, a solution for enhancing the VR experience by using BLE beacons in the process of adding new virtual objects and content into a digital twin environment. The approach was deployed and tested with the aid of a prototype that includes a 3D Unity application running on a server, a Raspberry Pi used as a beacon reader, and real and simulated beacons employing the Eddystone protocol. The solution includes a number of algorithms for the beacon messaging system, device discovery and rendering of the virtual objects.

The tests demonstrated that the application is able to process the beacon messages and generate the corresponding virtual content in a timely manner, as long as the beacons are placed near the beacon reader.

For future work, we aim to integrate a larger number of virtual devices to the platform and include a decentralised multi-user approach.

Acknowledgements

This work was supported by the European Union's Horizon 2020 Research and Innovation programme under grant 870610 (TRACTION Project) and the Science Foundation Ireland (SFI) via the Frontiers Projects grant 21/FFP-P/10244 (FRADIS) and Research Centres grant 12/RC/2289_P2 (INSIGHT).

References

[1] N. Kajikawa, Y. Minami, E. Kohno, and Y. Kakuda, "On Availability and Energy Consumption of the Fast Connection Establishment Method by Using Bluetooth Classic and Bluetooth Low Energy," in *Proc. of the Int. Symposium on Computing and Networking (CANDAR)*, 2016, pp. 286–290.

[2] M. Kuller, F. Kohlmorgen, N. Karaoğlan, M. Niemeyer, I. Kunold, and H. Wöhrle, "Conceptual Design of a Digital Twin Based on Semantic Web Technologies in the Smart Home Context," in *Proc. of the IEEE Int. Conf. and Workshop in Óbuda on Electrical and Power Engineering (CANDO-EPE)*, 2020, pp. 167–172.

[3] V. Gopinath, A. Srija, and C. N. Sravanthi, "Re-Design of Smart Homes with Digital Twins," in *Journal of Physics: Conf. Series*, vol. 1228, no. 1, 2019, pp. 1–9.

[4] A. A. Simiscuka, T. M. Markande, and G.-M. Muntean, "Real-Virtual World Device Synchronisation in a Cloud-enabled Social Virtual Reality IoT Network," *IEEE Access*, vol. 7, pp. 106 588–106 599, 2019.

[5] A. A. Simiscuka and G.-M. Muntean, "Synchronisation between Real and Virtual-World Devices in a VR-IoT Environment," in *Proc. of the IEEE Int. Symposium on Broadband Multimedia Systems*, 2018, pp. 1–6.

[6] X. Li, B. He, Y. Zhou, and G. Li, "Multisource model-driven digital twin system of robotic assembly," *IEEE Systems Journal*, vol. 15, no. 1, pp. 114–123, 2021.

[7] J. Li, Y. Zhang, and C. Qian, "The Enhanced Resource Modeling and Real-Time Transmission Technologies for Digital Twin Based on QoS Considerations," *Robotics and Computer-Integrated Manufacturing*, vol. 75, p. 102284, 2022.

[8] A. Aza, D. Melendi, R. Garcia, X. G. Paneda, L. Pozueco, and V. Corcoba, "Bluetooth 5 Performance Analysis for Inter-Vehicular Communications," *Wireless Networks*, vol. 28, no. 1, pp. 137–159, 2022.

[9] K. T'Jonck, B. Pang, H. Hallez, and J. Boydens, "Optimizing the Bluetooth Low Energy Service Discovery Process," *Sensors*, vol. 21, no. 11, p. 3812, 2021.

[10] M. Perri, F. Cuomo, and P. Locatelli, "BLENDER - Bluetooth Low Energy Discovery and Fingerprinting in IoT," in *Proc. of the Mediterranean Communication and Computer Networking Conf. (MedComNet)*, 2022, pp. 182–189.

[11] L. Reichert, S. Brack, and B. Scheuermann, "A Survey of Automatic Contact Tracing Approaches Using Bluetooth Low Energy," *ACM Trans. Comput. Healthcare*, vol. 2, no. 2, mar 2021.

[12] E. Park, M.-S. Lee, and S. Bahk, "AdaptaBLE: Data Rate and Transmission Power Adaptation for Bluetooth Low Energy," in *Proc. of the IEEE Global Communications Conf. (GLOBECOM)*, 2019, pp. 1–6.

[13] A. Mackey and P. Spachos, "Experimental Comparison of Energy Consumption and Proximity Accuracy of BLE Beacons," in *Proc. of the IEEE Global Communications Conf. (GLOBECOM)*, 2019, pp. 1–6.

[14] P. I. Andreev and B. R. Aprahamian, "Analytical Comparison of Bluetooth Low Energy Beacons," in *Proc. of the Int. Symposium on Electrical Apparatus and Technologies (SIELA)*, 2018, pp. 1–4.

[15] P. Bulić, G. Kojek, and A. Biasizzo, "Data Transmission Efficiency in Bluetooth Low Energy Versions," *Sensors*, vol. 19, no. 17, p. 3746, 2019.

[16] A. Bbosale, G. Benny, R. Jaison, A. Kbot, and S. Pati, "Indoor Navigation System using BLE Beacons," in *Proc. of the Int. Conf. on Nascent Technologies in Engineering (ICNTE)*, 2019, pp. 1–6.

[17] J. Kang, J. Seo, and Y. Won, "Ephemeral ID Beacon-Based Improved Indoor Positioning System," *Symmetry*, vol. 10, no. 11, p. 622, 2018.

[18] Z. Wang, K. Han, and P. Tiwari, "Digital Twin Simulation of Connected and Automated Vehicles with the Unity Game Engine," in *IEEE Int. Conf. on Digital Twins and Parallel Intelligence (DTPI)*, 2021, pp. 1–4.

[19] Y. Yang, W. Meng, and S. Zhu, "A Digital Twin Simulation Platform for Multi-Rotor UAV," in *Int. Conf. on Information, Cybernetics, and Computational Social Systems (ICCSS)*, 2020, pp. 591–596.

[20] J. Halpern, "Introduction to Unity," in *Developing 2D Games with Unity*. Springer, 2019, pp. 13–30.

[21] Intel, "TinyB," 2017. [Online]. Available: https://github.com/intel-iot-devkit/tinyb