A Virtual Space with Real IoT Data

Adam Faiers, Thomas Hoxey, Ian Stephenson

National Centre for Computer Animation, Bournemouth University

Figure 1: *The Laser Target Chamber*

Abstract

Large quantities of live data about an environment can be easily and cheaply collected using a network of small sensors (IoT). However these sensors typically do not display any information directly, and it can be difficult to understand the data collected. Conversely VR environments used for training, require scenarios to be created, populated with rich data. By linking the VR system directly to the IoT data broker we import the live (or recorded) status of real hardware from an industrial environment into the virtual world allowing a remote viewer to monitor the operation of the system.

CCS Concepts

•***Computing methodologies*** → ***Virtual reality; Mixed / augmented reality;***

1. Introduction

The University of Southampton operates a high power laser X-ray lab. In addition to the obvious dangers, entering the lab unnecessarily is discouraged, as it introduces contaminants to the sensitive environment. In order to help diagnose and monitor the status of the equipment, an ongoing project is adding a network of sensors to the hardware, recording temperatures, power usage, pressures, vibration and any other data that can be collected cost effectively.

These sensors push data to a central broker which makes this realtime data available to other network clients, and logs it for future analysis. Our VR environment uses this central data source, and links real world status to the virtual simulacrum.

2. Previous Work

VR has a long history of use by industrial clients, who create virtual copies of hazardous environments to enable staff to be trained without exposing them (or the equipment) to any real risk (eg [LYDS02]). Flight Simulators were one of the first such applications (including linking to physical hardware), but with the greater availability of VR, the potential is being explored in a wider range of industries.

Networking together multiple copies of a virtual environment is most commonly found in online games. It is quite common to have thousands of users existing in the same virtual space. While the technology that supports these games could be extended to link into real environments, it is typically proprietary, and customised for a specific application, relying on servers managed by the game developer.

DOI: 10.2312/egp.20181024

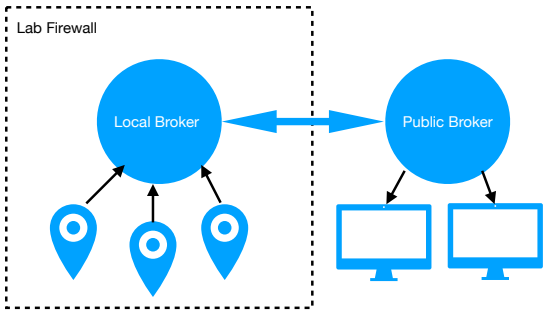

Figure 2: *MQTT Server Architecture*

MQTT is an open and standard protocol used within the industrial command and control field to link together sensors, actuators and control systems [Lig17]. It is highly portable and scalable, running on both cheap embedded development boards and high end servers capable of supporting thousands of clients. In addition it is highly configurable allowing sensitive data to be shared with a subset of users, while making read only access to other data fully public.

An MQQT system has a central "broker". All other participants act as clients to this broker. Once connected, a client “subscribes” to a number of data channels. Should any client “publish” data to that channel, the broker forwards it to all subscribed clients.

The VR system is implemented in Unity, allowing one of the standard C# MQTT client libraries to be used. Previous projects have demonstrated linking MQTT with Unity to provide simple networking services [ND], or to control simple devices from VR [Cab], but the full scale linking of a real world and a VR environment is still to be developed.

3. Implementation

Existing sensors within the lab publish their status to a local MQTT broker, as in figure 2. Each sensor has its own channel, which it can update whenever new data is available. This local broker is bridged to a public broker (outside the lab firewall) and pushes any shareable data out. By using two brokers, we have increased control of security, as there is no sensitive data on the public server. In addition this protects the local broker from the potential load of multiple external clients. The VR client connects to the public MQTT broker and subscribes to the channels for sensors of interest, allowing it to receive any updates.

Within the Unity VR environment a Broker object acts as a proxy for the real public broker so each VR instance makes only a single connection to the real broker. An object in the VR environment such as a cooling unit uses the local proxy and registers for a channel in a similar fashion to if it were connecting to a real MQTT broker. The proxy connects to the real broker and subscribes to the channel on the coolers behalf. It maintains a list of local objects and the channels they subscribe to, so when a message is received from the public broker it can forward the message to all subscribed local objects.

Upon receiving a message, the object in the VR scene can redraw itself to reflect the status of the real world object. For some objects this may be a direct change to the appearance to the object — for example a light turning on, or a case being opened. However much of the data collected is not directly visible. In such cases a bubble appears over the object when the user is looking directly at it, showing the objects status as seen in Figure 1.

4. Conclusions

The system currently tracks a small number of objects in the lab, but over time this is being extended. Some pieces of instrumentation within the lab display complex information, and will require extensive simulation to reproduce their appearance directly, but ultimately it should be possible to move round the VR system and view each piece of equipment. Work is also in progress to track users within the lab, so that their activity can be seen in the VR environment.

In the near future AR systems will be capable of providing the same data we are working with to users in the lab, making important, but currently invisible status information directly available. In addition because the data is logged by the MQTT data, it is possible to play back previous activity, allowing experiments to be verified and debugged.

References

[Cab] CABAGBAG E.:. Iot and augmented reality w/mqtt, unity3d and leap [online]. URL: https://www.youtube.com/watch?v=2zoZPAYiEfI. 2

[Lig17] LIGHT R. A.: Mosquitto: server and client implementation of the MQTT protocol. *The Journal of Open Source Software 2*, 13 (may 2017), 265. URL: https://doi.org/10.21105/joss.00265, doi:10.21105/joss.00265. 2

[LYDS02] LIN F., YE L., DUFFY V. G., SU C.-J.: Developing virtual environments for industrial training. *Information Sciences 140*, 1 (2002), 153 – 170. Interactive virtual environments and distance education. URL: http://www.sciencedirect.com/science/article/pii/S0020025501001852, doi:https://doi.org/10.1016/S0020-0255(01)00185-2. 1

[ND] NANDY A., DAS M.:. Iot-vr with unity,intel edison [online]. URL: https://www.codeproject.com/articles/1114744/iot-vr-with-unity-intel-edison. 2