DIGITALISE REAL-TIME DATA IN A PHYSICAL INDUSTRIAL ENVIRONMENT USING MIXED REALITY

CO3201 Computer Science Project

Dhanil Capil Duvarcadas
School of Computing and Mathematical Sciences, University of Leicester

May 2022

Declaration of Plagiarism

Abstract

Mixed reality is a growing field, and when Microsoft introduced the HoloLens in 2016, it opened new ways we could interact with computers. This project aims to develop an application that can allow users to visualise an industrial component sensor data as a hologram. The project leverages Microsoft Azure Cloud to provide most of the infrastructure, including modelling an abstraction of a power plant in Azure Digital Twins to act as a data store for the most recent telemetry data from the sensors. The architecture is event-driven, utilising Azure Event Grid to trigger Azure Functions.

The user can see the component's data in the final solution, including sensors telemetry data using a gauge chart. The application uses the telemetry data to determine against a predetermined range if the data is anomalous, and an alert is raised. When an alert, the user is redirected by a directional arrow pointing towards the dashboard. The dashboard's location could saved for future sessions by utilising Azure Spatial Anchors. In theory, the Spatial Anchors could be shared with other users; however, the lack of an additional HoloLens to test this makes me unable to verify. The reason for using Spatial Anchors is because QR code stickers can wear off or become dirty, which is not ideal and NFC requires touch.

Thorough testing, including unit tests, integration tests, manual tests and usability test, were performed to ensure that any imperfections was caught and that the feedback received could be used to improve future versions.

The main conclusion is that the application is futuristic, especially with the hand menu however, more work needs to be done in terms of Trimble XR10 (a hard hat version of the HoloLens) ergonomics to make it suitable to use for a full day use before the application could even be considered to be used in production. The application could still be further improved by adding the ability to see past data and fix an outstanding hard to debug Azure Spatial Anchor bug.

Acknowledgements

I want to thank Peter Arrell, Ian Beeson, Ben Williams, Rene Koch, Soumya Chatterjee for their feedback and roles as Product Owners for this project and allow me to borrow the equipment. A big thanks to Charlie Surman, Joshua Malemba and Thomas Verliden for testing the software and giving me feedback.

I also want to thank my supervisor Professor Reiko Heckel for guiding me through the project.

a) Table of Contents

b) Table of Figures

c) Table of Abbreviations

Augmented Reality	AR	2
Azure Spatial Anchors	ASA	18
Bouvet Development Kit	BDK	18
Central Processing Unit	CPU	18
Comma Separated Value	CSV	13
Digital Model	DM	2
Digital twins	DT	2
Digital Twins Definition Language	DTDL	12
General-Purpose Input and Output	GPIO	16
Global Positioning System	GPS	4
Internet of Things	IoT	2
Mixed Reality	MR	2
Mixed Reality Toolkit	MRTK	18
Near-Field Communication	NFC	4
Product Owners	PO	31
Project Owners	POs	8
Random Access Memory	RAM	18
Structured Query Language	SQL	13
Virtual Reality	VR	2

1. Introduction

1.1. Aims

As part of my Year In Industry, I led training to engineers to use the Microsoft HoloLens so that virtual inspections could be conducted without travelling- especially crucial during the Covid-19 pandemic. This barely grafts the capabilities of said device, and I feel like there is untapped potential in the ways that we could use the HoloLens; hence I want to develop an application that will help the engineers.

One way to achieve helping engineers is to provide tools to help them diagnose faults and see the current status, which could be achieved by displaying live data from the sensors on the HoloLens. This will provide engineers with relevant information in their field of view, rather than holding a separate device, which will increase productivity. The information can be loaded

1.2. Project objectives

For this project to be a success, the following criteria have to be met:

- Perform a literature review and evaluate existing solutions
  - Research Mixed Reality
  - Research the HoloLens
  - Research how to model IoT data
  - Research anchoring dashboards to locations
- Plan, design and implement a event-driven architecture
- Evaluate the use of the HoloLens and the recommendations for the ecosystem
- Model a power station using a Digital Twin technology
- Develop an application for the Microsoft HoloLens using the Unity 3D engine to view real-time data
- Ability to anchor the dashboards to a location
- Use data from sensors using a Raspberry PI
- Test the implementation, including usability testing and receive user feedback
- Learn more about technologies:
  - Azure Cloud and how we can leverage it to reduce the time spent on infrastructure maintenance
  - Good programming practices and how to build secure applications
  - HoloLens and how it interacts with the real-world
  - Developing a mixed reality application, including necessary techniques to develop MR applications and languages used

2. Literature survey and existing solutions

This literature review is intended to help me understand how IoT data and IoT related technologies work and how it has been used in the past. Additionally, I will look into existing solutions and see how they are implemented to draw inspiration and innovate upon them.

2.1. Internet of Things

The Internet of Things (IoT) is a growing field and is "projected to grow from \$381 billion in 2021 to \$1,854.76 billion in 2028" [1]**.** IoT describes devices with sensors that connect to the Internet to share the data with services or other devices [2]. Devices such as video doorbells like the Ring Doorbell [3] allow users to monitor their houses when not at home. However, it does raise some concerns about privacy; in 2021, a judge in UK has ruled that the Ring Doorbell range of 12-20 metres "was excessive" [4] and invaded the privacy of the neighbours as they could be "unaware of the device, which breached the Data Protection Act 2018 and the UK General Data Protection Regulation" [5]. As the project will take in data from sensors, it is classified as IoT and hence, security and privacy need to be taken into account.

2.2. Extended Reality

IoT is not the only sector growing; reality-enhancing hardware is a growing field with a predicted worth of \$215 billion by 2021 [6]. The reality-enhancing field, frequently referred to as extended reality (XR), has three distinct categories: Augmented Reality (AR), Mixed Reality (MR) and Virtual Reality (VR); these three terms are often used interchangeably however, they represent different aspects of XR [7]:

- VR describes the use of an artificial environment with complete immersion, such as Oculus Rift or the HTC Vive, and is often used for gaming
- AR adds virtual objects to the real world and tends to be utilised often with smartphone cameras by using apps like Snapchat and Pokémon Go
- MR is a mix of AR and VR; it allows interaction of the virtual object to the real world and the virtual world, which is what Microsoft HoloLens utilises.

As we will be using the HoloLens, I will be focusing on MR in this literature review. MR has successfully been used in visceral surgery by Sauer et al. [8], describing MR systems with "a high potential" for surgery and allowing medical staff to learn in a more "intuitive manner". Sauer et al. project does have some limitations, such as the image of the scan not being overlayed over the patient but rather on a different axis, but it is great to see that this technology can still achieve good results, and the authors hope to correct this in a future version. The main takeaway from this is that MR has the potential to provide information to the users as it helps to visualise data.

2.3. Digital Twins

Digital twins (DT) is a digital representation of an object. It allows "real-time prediction, optimisation, monitoring, controlling, and improved decision making" from the data obtained [9]. Companies such as GE [10] and Microsoft [11] are investing heavily in Digital Twins technology, with GE saving their clients over \$1.5 billion (around £1.1 billion) due to the "lower operations and maintenance costs" [12]. DT should not be confused with a Digital Model (DM), as DT represents an object, therefore differentiating from a DM by being attached to an object and "living" through it. Octal et al. [13] describe DT as not being just one technology but a "technological cocktail", due to the involvement of machine learning, artificial intelligence, sensors and cloud computing to achieve this. This is due to the DT requiring real-time data to work, which can then be used for prediction. Aivaliotis et al. [14] successfully used DT to predict the remaining useful life using predictive maintenance. However, their work is mainly based on components of manufacturing that do not have a long remaining useful life, and therefore in future works, the authors aim to make it more of a "generic predictive maintenance." In this project, DT could be used to model a power station and act as a data store of the whole power station sensors. This would then allow in the future to use predictive maintenance or run AI models to predict values and the component's remaining lifespan.

2.4. Microsoft HoloLens

Sirilak and Muneesawang [15] found that "the HoloLens is highly beneficial" in the healthcare industry; however, "significant training must be provided ... (as) users are entirely unused to such technology". This seems to coincide with my own experience. During my Year in Industry, I have led training sessions so that engineers could use the HoloLens on the field, and I found needing a rigorous tutorial before the user can use it efficiently. Sirilak and Muneesawang [15] also note that if the user wears the HoloLens "for more than 30 minutes … the wearer can become fatigued". Although they were using the HoloLens 1, the HoloLens 2 has "a more comfortable and ergonomic design" according to Kognitiv Spark [16]. In my project, I will be using a Trimble XR10, a Microsoft approved modified version of the HoloLens, which sacrifices ergonomics for safety by being built into a hard hat. This is non-negotiable as a hard hat is vital for engineers on-site, so it could mean that the effectiveness of the HoloLens could be affected by usability and convenience. Scaravetti et al. [17] used the HoloLens for engineering training and described the "technology as a promising support tool", and does not "envisage a complete replacement of conventional tools". This suggests that the HoloLens technology is still in its infancy and more work needs to be done to make usable in day-to-day.

2.5. Existing solutions

I have encountered a picture [Figure 1] during my research similar to what I hope to achieve. The picture demonstrates data being summarised to engineers in what seems to be real-time information. The design shows a red header for an "Offline status" and green for "Online", which allows the end-user to see the status at a glance. The picture also suggests that the holograms are shared across all the users, further reinforced by the image shown on the Azure Spatial Anchors web page [18], a service provided by Microsoft that allows to save 3D content and share it with other users. This webpage is the only mention of the image, and hence I am unable to ascertain with certainty whether the solution exists or is just a render for promotional purposes, so the feature list can not be determined.

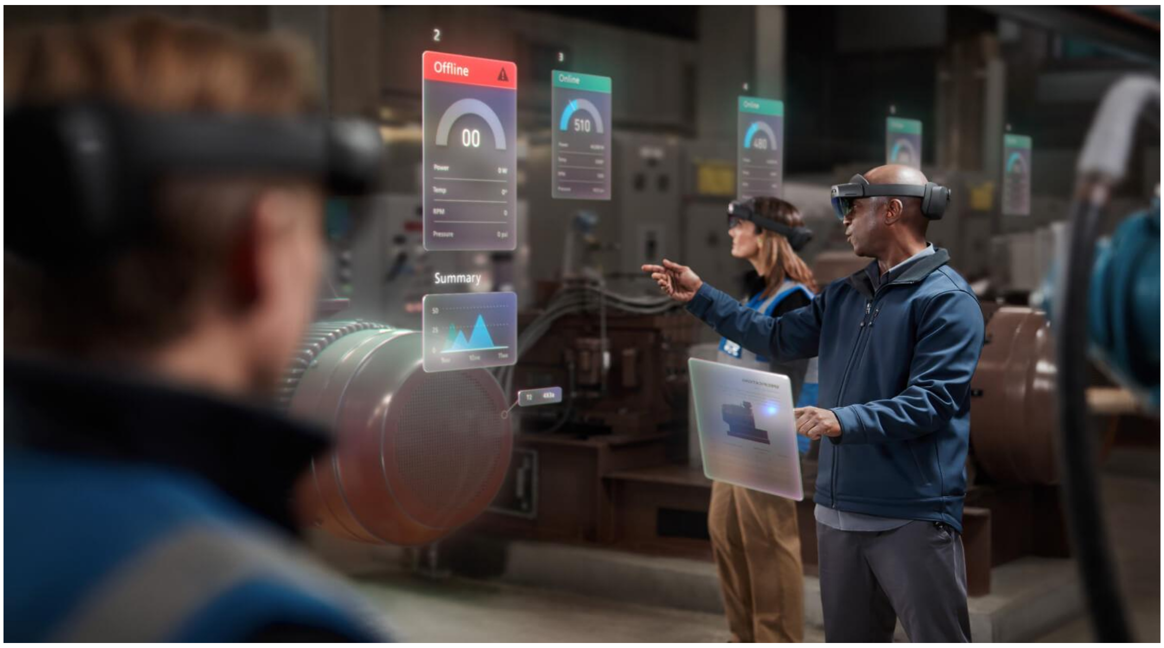
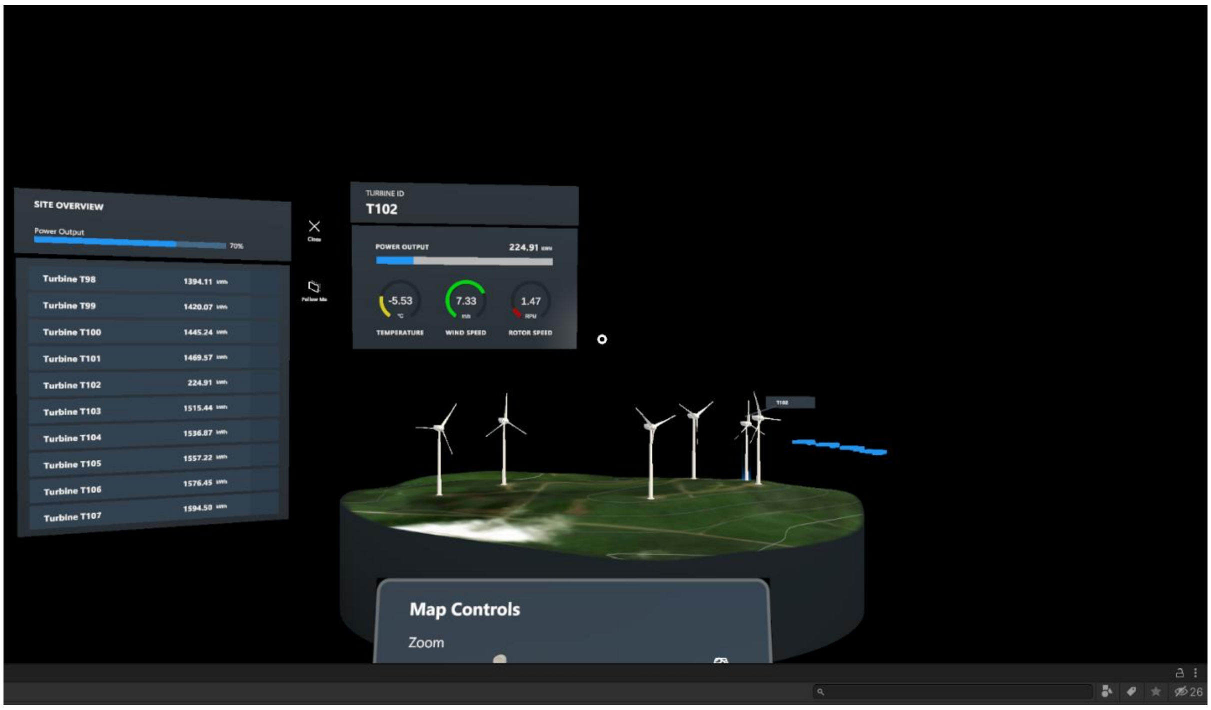


Figure 1 – Microsoft HoloLens Promotional picture [18]

Another resource provided by Microsoft is a tutorial for the HoloLens [19] and the final solution is shown in Figure 2. It consists of a 3D model with Wind Turbines, and their status is shown on the left dashboard. The solution is agnostic to the surroundings and can be placed anywhere. It does not provide access to past data, but it provides a clean interface. It utilises Microsoft Reality Toolkit, which is a library to aid MR development in the Unity engine for the HoloLens.



Turbine T98	1394.11 kWh
Turbine T99	1420.07 kWh
Turbine T100	1445.24 kWh
Turbine T101	1469.57 kWh
Turbine T102	224.91 kWh
Turbine T103	1515.44 kWh
Turbine T104	1536.87 kWh
Turbine T105	1557.22 kWh
Turbine T106	1576.45 kWh
Turbine T107	1594.50 kWh

Figure 2 - Microsoft HoloLens Monitoring Wind Turbines [18]

2.6. Anchoring dashboards

One of the ways that dashboards could be fixed into locations or be recognised upon sight is the use of Near-Field Communication (NFC). However, this has the drawback that it needs to be close to work [20]. This would be unsuitable as it would defeat the purpose of visualising the data whilst passing or being alerted to a faulty component's location. Global Positioning System (GPS) could be used, but the accuracy is around 7-13 metres [21], which would be unsuitable for indoor use in an industrial setting. Another alternative is QR codes, however, they could become dirty [22] and cover up the details, making them unsuitable. Microsoft provides a solution called Azure Spatial Anchor, which represents an important point in the world that the system should keep track of over time" [23]. Rosales et al. [24] found that spatial anchors provide "uniform distribution with a mean value of 46 mm" accuracy in his research, which would make spatial anchors a suitable choice to use for anchoring dashboards.

2.7. Summary

In conclusion, although the HoloLens has been available for a few years, there does not seem to be a commercially available solution that allows displaying data from sensors. There are guides by Microsoft that can help us achieve the barebones of the objective, but they seem to lack the ability to fix the dashboard in place, and this is where my solution will differ from it.

The engineer being able to see real-time data is a crucial point for productivity. DT can help us deal with the sensors' data and allow future versions of my solution to predict faults and maintenance. Libraries and frameworks such as MRTK can help us to reduce development time by providing scripts and demos for MR interactions. As for anchoring dashboards visualisation, this could be done using Azure Spatial Anchors, which have high accuracy.

The information displayed also needs to be succinct, that is, the less “noise”, the better, hence a clean user interface will be needed. The user interface will also need to be translucent so as not to block the view for safety reasons. Colour coding is also necessary to understand data at a glance rather than interpreting the data, and accessibility must also be considered in future versions. Safety to the user is crucial in a power plant, consequently, the use of the modified version of the HoloLens with a hard hat is required, so convenience and ergonomics are not a priority compared to it.

Furthermore, research on the HoloLens has previously indicated that the technology is not ready to replace the current tools, and hence I am going to try to draw my own findings

3. Requirements, Specification & Resources

3.1. Audience

The audience for the solution is the engineers that will use it during their daily duties. From remote inspection to power plant monitoring, they will utilise this application by augmenting their view with data from the sensors, which means they will not need to interact and look at another device to see the data; instead it will be overlayed in their vision.

3.2. Requirements

3.2.1. Functional Requirements

Based on my talks with the stakeholders, the following functional requirements were specified:

- Ability to see live data from the sensors
  - Having real-time data will mean that the engineer can react quickly
- Ability to see alerts
  - So that faults and outages can be noticed by the user
- Ability to anchor the dashboards to a location
  - This would mean that the dashboard for a component is fixed to a location and will be there if a user goes to another place
  - The dashboards can also (in theory) be shared with other users, however, due to the high cost of the HoloLens, I will not be able to test this
- Ability to redirect engineers to a location where an alert is
  - Engineers are quickly able to locate where a part is that is faulty
- Ability to see the past data from the sensors
  - It is helpful for the engineer to see the past data so they can use it for troubleshooting purposes

3.2.2. Non-Functional Requirements

3.2.2.1. Performance

As the data will be streamed live from a sensor, the data will need to have a maximum of 15 seconds of delay.

As there could be a possibility of having hundreds of sensors all around the power plant, the application's architecture has to be scalable. Utilising serverless services could help us achieve this.

3.2.2.2. Safety

As the solution will be utilised in a power station, I need to be careful that the application does not block the user's field of view as it could lead to safety incidents. The Microsoft HoloLens by Uniper is a modified version with a built-in hard hat, which is mandatory to visit a power plant. The regular version of the HoloLens would be unsuitable for live deployment.

3.2.2.3. Security

The solution must be secure because it will contain data about the infrastructure crucial to the nation’s power grid. This can be achieved by using HTTPS and SSL when calling the APIs and following best practices when developing an application. All the passwords, API keys need to be hashed, and the data needs to be encrypted when in rest and transport. Wherever possible, IP and port restrictions will be used to further strengthen security.

3.3. Resources and Equipment

The equipment that I will be utilising will be the following:

- Sensors such as temperature and pressure, i.e. BMP180
- Raspberry Pi 3 – provided by Uniper Technologies
- Trimble XR10 (a modified HoloLens with a hard hat) – provided by Uniper Technologies

I have chosen to use Microsoft Azure as my cloud provider due to the following reasons:

1. Uniper already utilises Microsoft Azure as their cloud provider, and hence it will be easier to fund the cloud resources
2. Microsoft Azure provides Azure Spatial Anchors, which integrates with the HoloLens, and rival cloud providers do not offer such services
3. I can take advantage of Platform as a Service (PaaS) and serverless services, and hence I can spend more time developing than worrying about the infrastructure

4. Design & Implementation

4.1. Methodology and the use of Git

This project utilised Scrum as its methodology. Scrum splits workloads into a sprint, where each sprint is two weeks long. At the start of the sprint, the Project Owners (POs) planned and scheduled the work items. The Scrum board managed the backlog of tasks to be completed.

Each user story received its branch that was merged when the entire feature was developed. This ensured that changes in priority in the stories could be reacted quickly by branching off main, which only contains tested and working code. This is following the best practices in the industry [25]

4.2. Architecture

As this is an IoT application, the data flow is an essential part of the solution. Based on my literature review, I have designed an architecture that is scalable and can be implemented within the budget constraints:

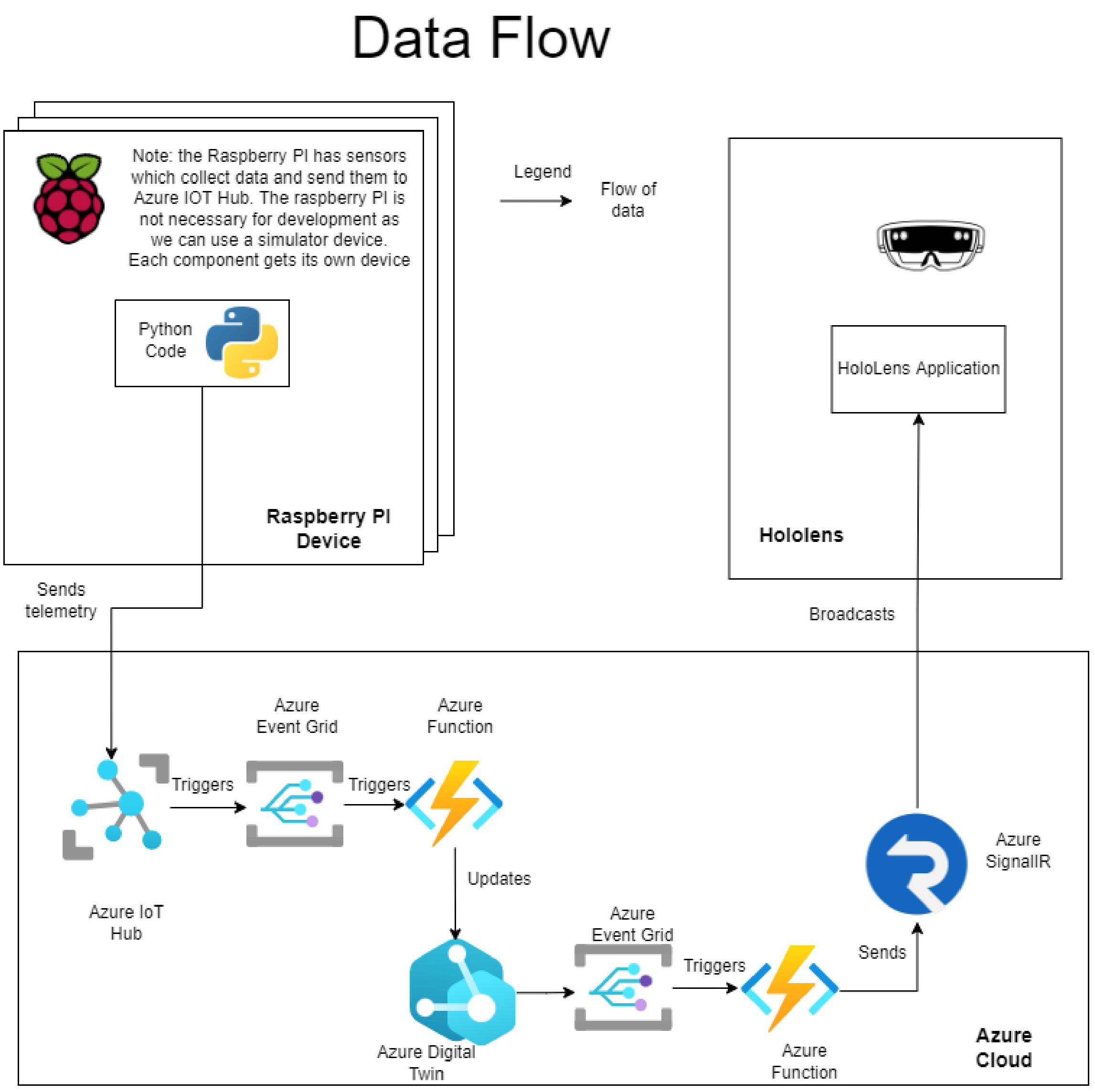


Figure 3 - Architecture of the final Solution

The Architecture of the data flow can be broken into three components:

- The Raspberry PI –uses sensors such as a thermometer to measure the temperature
- The Azure Cloud –is responsible for receiving, process, storing and broadcasting the data
- The HoloLens app- receives the data and uses it to display the data

4.2.1. Raspberry PI

This device will be used to retrieve the data from the sensors and send it to the cloud. A data simulator with a Python script that sends data to the IoT cloud will be used for development. It can also be replaced by any other service that can send data to the cloud.

4.2.2. Azure Cloud

I have tried to use as many serverless services as possible as it allows me to scale when multiple sensors are introduced, as it is not tied to the specification of a machine. This will mean that the architecture can achieve the requirement to scale to thousands of devices.

4.2.2.1. IoT Hub

This is where the data collected from the Raspberry PI will be sent to. It is essentially a message broker that is suited to IoT messages and devices. As it is a message broker, I can redirect the messages to the relevant services.

4.2.2.2. Azure Digital Twins

Based on my research, Azure Digital twins allow us to represent objects digitally. Although it would be simpler to bypass this and store the values in a database and retrieve the latest readings, modelling the power station opens up more options for future versions. We could run machine learning to predict faults before it happens and infer where the fault could lie, i.e. related joined components.

4.2.2.3. Azure Event Grid

One of the requirements is to be able to scale. Azure Event grid would allow us to fire off an action every time there is new data. Thus it makes it scalable as it is not dependent on the server specifications to run and can increase automatically when the load increase. This makes an event-driven architecture.

4.2.2.4. Azure SignalR

SignalR would allow broadcasting the data to all the clients connected. This is a better method, as the device does not have to poll for new data, so there will be a lower delay retrieving the live data and lower service usage. This is because the Signal IR broadcasts it continuously rather than a device making continuous API calls to refresh the data. An analogy to this method is that satellite TV is distributed, the satellite broadcasts it and the satellite box decode, compared to network streaming, where the device asks the server to send the data.

4.2.3. Microsoft HoloLens Application

For the application that will be developed, I decided to go for a Unity 3D application. This is mainly due to the documentation that Microsoft provides being catered for it. The Mixed Reality Toolkit that provides UI components for the HoloLens is also developed for Unity. This application will subscribe to the Azure SignalR broadcast allowing it to display live data to the user into dashboards. These dashboards will be fixed onto a location using Azure Spatial Anchors.

4.2.4. Differences & Challenges

My architecture plan was based on the research that I had performed at the start of the project (shown in Figure 25). However, some difficulties/flaws made me revise it:

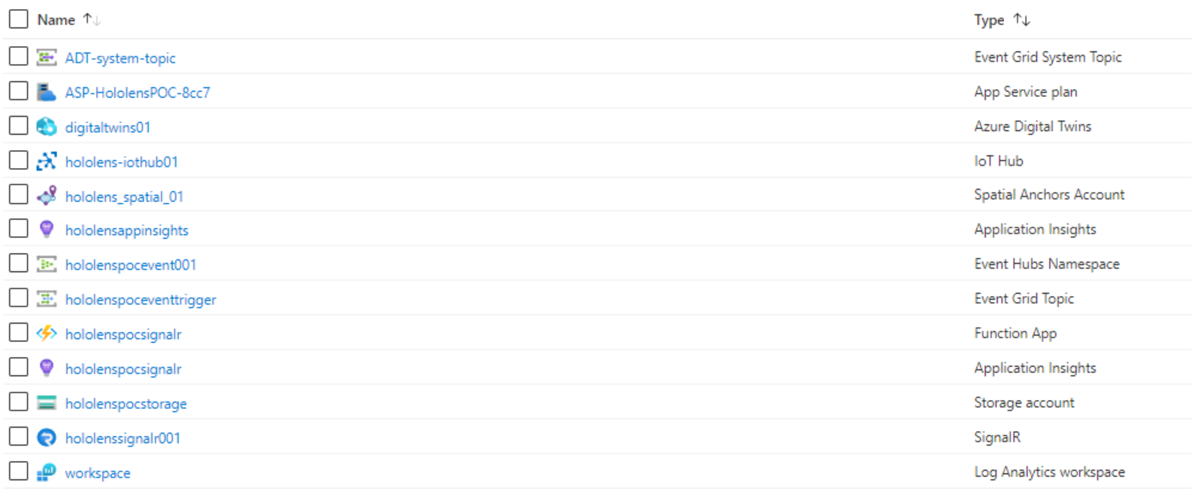
1. Contrary to my understanding, IoT Hub's Plug and Play, which utilises models, is not exactly Plug and Play with the Digital Twins, i.e. if a new property message is sent to IoT Hub, the changes will not be reflected in the Digital Twins. Instead, the Plug and Play means that they can share the same model files.
2. When Azure Spatial Anchors creates an anchor, it does not provide any storage for the anchor IDs, instead, the applications will need to store them. I chose to add a property to the Digital Twins model so that I would have a single source of data rather than an additional database

These flaws led to many changes in the data flow the, most notably the introduction of two services:

1. One to send IoT Hub's data to Azure Digital Twins
2. One to retrieve from and store the Azure Spatial Anchors in Digital Twins

These changes proved to be time-consuming as it meant that I now needed to rework how the data flows and spend more time than I expected working on this part.

4.3. Azure Cloud



Name	Type
ADT-system-topic	Event Grid System Topic
ASP-HololensPOC-8cc7	App Service plan
digitaltwins01	Azure Digital Twins
hololens-iothub01	IoT Hub
hololens_spatial_01	Spatial Anchors Account
hololensappinsights	Application Insights
hololenspocevent001	Event Hubs Namespace
hololenspoceventtrigger	Event Grid Topic
hololenspocsignalr	Function App
hololenspocsignalr	Application Insights
hololenspocstorage	Storage account
hololenssignalr001	SignalR
workspace	Log Analytics workspace

These are all of the resources deployed in the Azure Cloud. Some of them, such as Application Insights, App Service Plan, Storage Account and Log Analytics Workspace, are auto-deployed due to the resources described in 4.2.2. These resources store and analyse the logs and transactions, along with configuration changes to the resources.

4.4. Digital Twins

4.4.1. Overview

Based on my research, Azure Digital twins allow us to represent objects digitally. Although it would be simpler to bypass this and store the values in a database and retrieve the latest readings, modelling the power station opens up more options for future versions as we could run machine learning to predict faults before it happens and infer where the fault could lie such as related components.

4.4.2. Design

The diagram below (Figure 4) is a high-level abstraction of a power plant and will serve as the Azure Digital Twin model. This enables me to identify how each sensor relates to other sensors and the overall power station. From this plan, I can then create a device simulator for development (which later will be replaced by Raspberry PIs with sensors) to feed the model in the Digital Twin.

The high-level view shows how coal, water, steam, and energy move throughout a power station and in which direction. This will then be used to show the directionality of the components in the Digital Twin Model and how they interact with each other. The different types of sensors that components may contain are indicated at the bottom of the figure. For example, the boiler has two sensors- one for temperature and one for pressure. The sensors would then be replicated in the Digital Twin model as a property of the component.

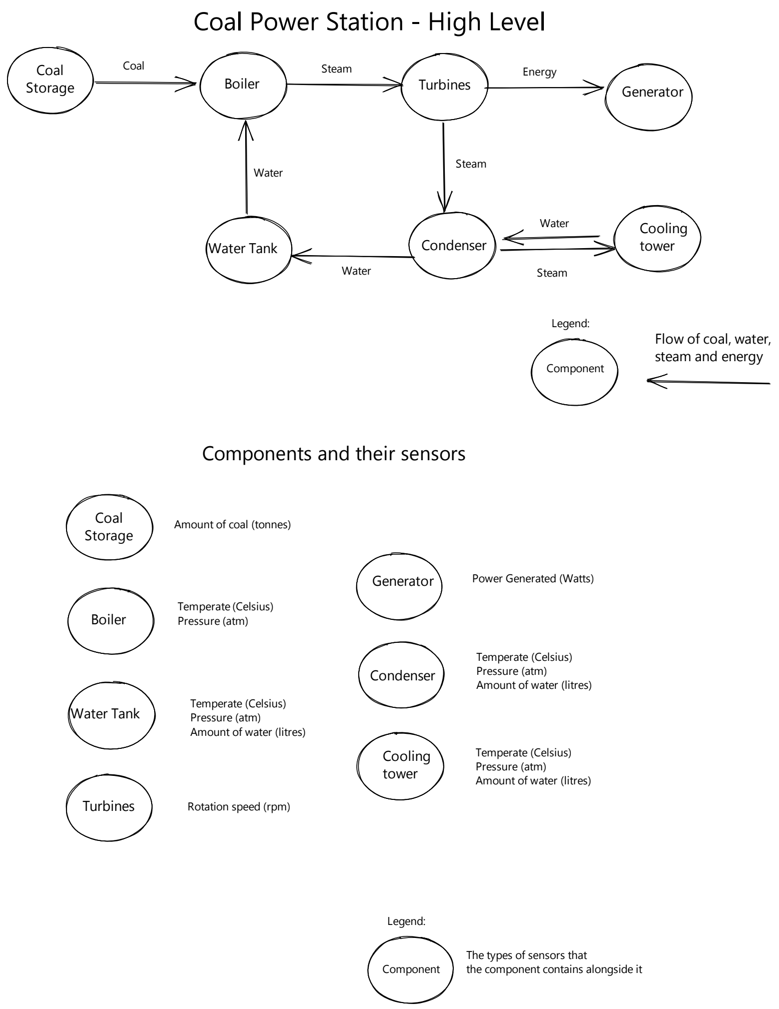


Figure 4 - Power Plant Azure Digital Twin Model

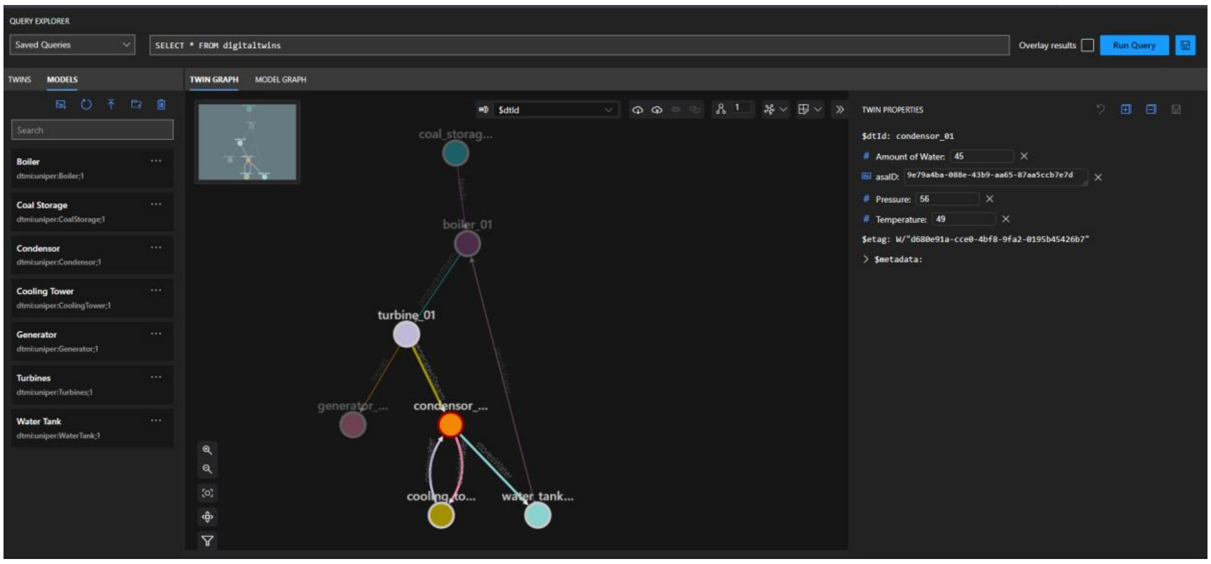
4.4.3. Implementation

Based on the models above, I used Digital Twins Definition Language (DTDL) to define it. Here is an extract of what the model files look like:

```
{
    "@id": "dtmi:uniper:Boiler;1",
    "@type": "Interface",
    "@context": "dtmi:dtdl:context;2",
    "displayName": "Boiler",
    "contents": [
      {
        "@type": "Property",
        "name": "Temperature",
        "schema": "double"
      },
      {
        "@type": "Property",
        "name": "Pressure",
        "schema": "double"
      },
      {
        "@type": "Property",
        "name": "asaID",
        "schema": "string"
      },
      {
        "@type": "Relationship",
        "name": "producessteam",
        "displayName": "Produces Steam",
        "target": "dtmi:uniper:Turbines;1"
      }
    ]
  }
```

As seen above, the DTDL is a subset of the JSON language. The first four lines define the metadata, including their ID and display name, whilst the content is an array for the properties. The final solution includes a property called 'asaID' that is not present in the design- see 4.2.4 for more details. It is also in the contents section where the relationships between models are defined, where the name of the associated model is identified. This is done by changing the type to a relationship and adding a target which is the model's id that is linked to it

Once the models' files are uploaded, we can then create the Digital Twins and their respective relationships by either uploading a file or through the UI:



```
SELECT * FROM digitaltwins
```

Boiler
dtmi:uniper:Boiler;1

Coal Storage
dtmi:uniper:CoalStorage;1

Condensor
dtmi:uniper:Condensor;1

Cooling Tower
dtmi:uniper:CoolingTower;1

Generator
dtmi:uniper:Generator;1

Turbines
dtmi:uniper:Turbines;1

Water Tank
dtmi:uniper:WaterTank;1

```
$dtId: condensor_01
# Amount of Water: 45
asaID: 9e79a4ba-088e-43b9-aa65-87aa5ccb7e7d
# Pressure: 56
# Temperature: 49
$etag: W/"d680e91a-cce0-4bf8-9fa2-0195b45426b7"
> $metadata:
```

Figure 5 - Digital Twins Viewer

The language used to query ADT is similar to Structured Query Language (SQL), which allowed for quick uptake and understanding of how to query the ADT.

4.4.4. Challenges

Creating DT models was not without its problems:

- I could not get the Digital Twins Explorer to run locally [26] and instead relied on the hosted version. This presented its own set of problems as it is still in preview; this included multiple issues when logging in, only to be redirected to log in again. This bug seems to now be fixed, but there was significant time lost trying to make it work
- Modelling relationships are handled by uploading a Comma Separated Value (CSV) file, however, I am unable to find any documentation that specified the format of the headers and instead, I relied on samples. When uploading the file, only a general exception is shown if it fails, with no useful information to help diagnose the issue. Due to this, I spent a large amount of time trying to get the relationships uploaded only for it to turn out to be a misspelt model.

4.5. SignalR

SignalR would allow broadcasting the data to all the clients connected. This is a better method, as the device does not have to poll for new data, so there will be a smaller delay retrieving the live data and lower service usage. This is because the SignalR broadcasts it continuously rather than a device making continuous API calls to refresh the data. An analogy to this method is that satellite TV is distributed, the satellite broadcasts it and the satellite box decode, compared to network streaming, where the device asks the server to send the data.

As SignalR act as the endpoint for the application, it provides a connection so that the clients are able to connect to it. The Azure Functions provide the broadcasting and negotiation events.

4.6. Azure functions

Azure Functions are serverless cloud services that run on demand, which allows scaling depending on the load. As it is serverless, there is no cost for not using it (i.e. at night when I am not developing), making development more accessible and cheaper.

There are 5 Azure functions in the final solution written in C#, a language that I did not have previous experience with.

4.6.1. IoT Hub to Digital Twins

Once a message arrives to IoT Hub, it needs to be redirected to Digital Twins. To do this, an Azure function alongside an event grid is used to trigger.

Nothing is returned as the function directly changes the Digital Twins data (shown below as no outputs."

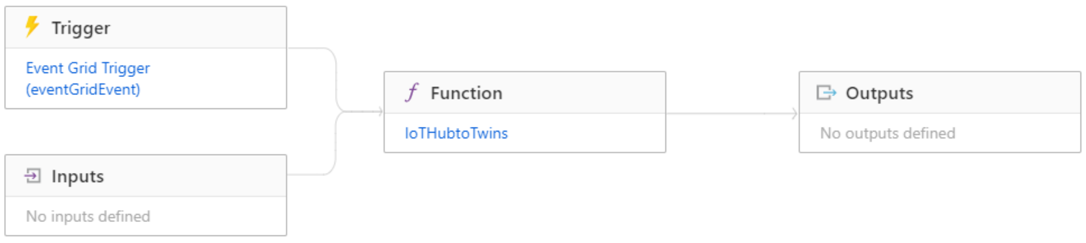


Figure 6 - Azure function for IoT Hub to Digital Twins (code not shown)

4.6.2. SignalR

SignalR allows broadcasting the newly updated data from the Digital Twins to the client applications. However, it still requires an Azure Function to format the data to be broadcasted. It utilises two Azure functions to achieve this.

4.6.2.1. Negotiate

This Azure Function build a connection when a POST request is made to it and returns the connection details to SignalR

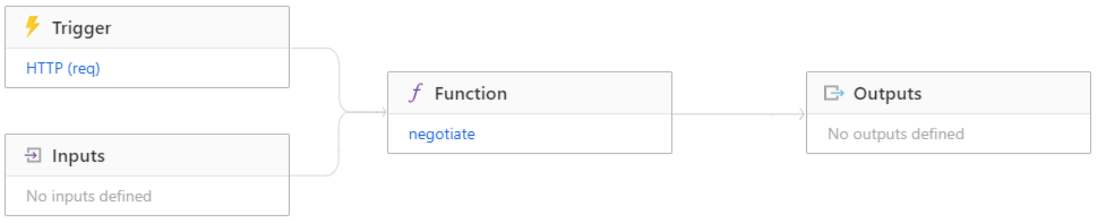


Figure 7 - Azure Function for SignalR (negotiate)

4.6.2.2. Broadcast

Once the connection is established, the client devices listen for broadcast events using this Azure function, which is triggered when an Azure Event Grid event for a change in Digital Twins is fired. This allows all the clients that are connected to retrieve the data.

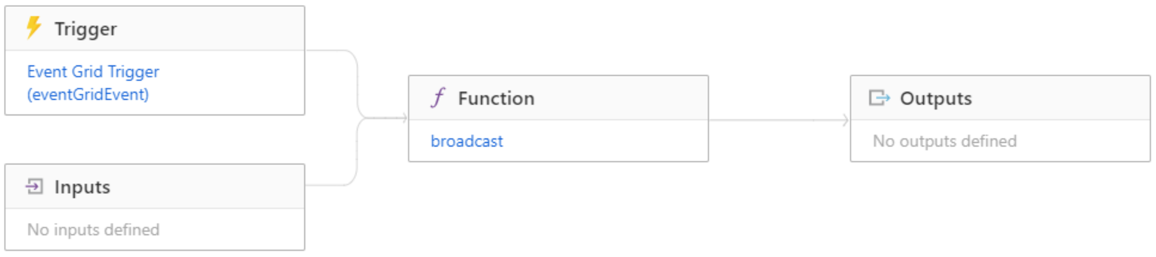


Figure 8 – Azure Function for SignalR (broadcast)

4.6.3. Azure Spatial Anchors

During development, I found that when creating an ASA, the ID is returned. This ID needs to be saved, and it is usually stored in a database. However, this would introduce another layer to the architecture. As this was metadata for each sensor, I decided to have the data in Azure Digital Twins instead, where it could be a single source of data for each component.

To do this, I needed to be able to retrieve and set the ASA id from the application and hence utilised two Azure Functions to achieve this. This would be similar to an API endpoint.

4.6.3.1. GetASA

This function is used by the application to retrieve the ASA from DT. It does this by sending a GET request to the endpoint 'anchors/{deviceID}', where the deviceID is the component's name. Once the Azure Functions finds the component in the DT, it will then retrieve the metadata and return the ASA id back to the application as a JSON output.

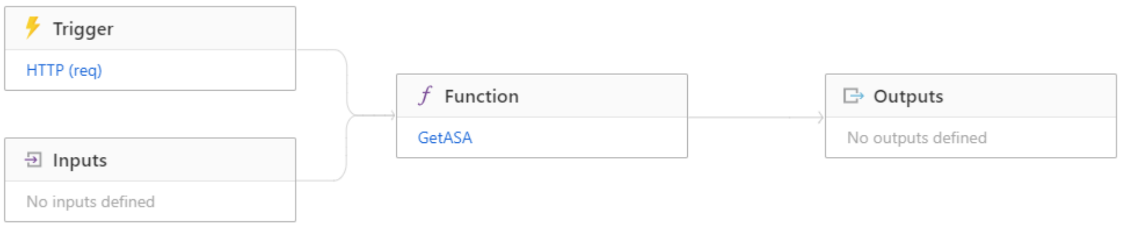


Figure 9 – Azure Function for Azure Spatial Anchors (GetASA)

4.6.3.2. UpdateASA

This is similar to the GetASA function but instead updates the component's ASA in DT. It utilises similar endpoints, but a POST request is used, along with a JSON body defining the new ASA id. This will then update the DT with the new id and return a success message.

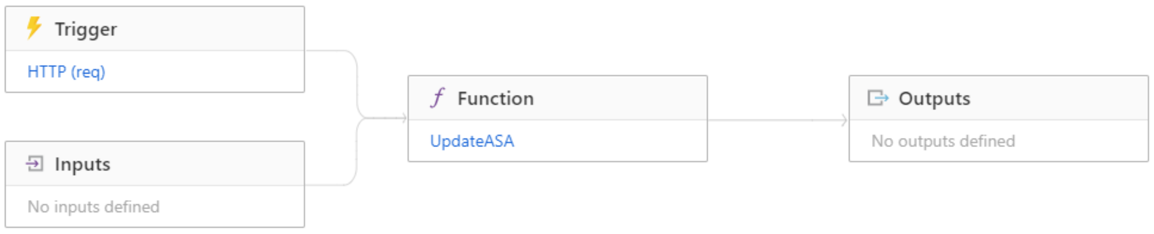


Figure 10 - Azure Function for Azure Spatial Anchors (UpdateASA)

4.6.4. Challenges & Difficulties

Originally, Python was the chosen language for the Azure functions, but due to the lack of support for the SignalR input and output, I had to change it to C#. This is a language that I was unfamiliar with before starting to work on this project, and hence it proved to be a challenge.

As seen in the screenshots above, the output is not defined although some of the functions return data. This made me think that I did something wrong, which, coupled with up to "5 minutes delays" when viewings logs, made it harder to diagnose bugs. Local development also proved to be a problem as it is hard to emulate Azure Event Grid triggers locally for testing purposes, making me deploy and wait 5 minutes for the logs. This slowed down my productivity and velocity of the sprints as it required longer than I expected.

4.7. Raspberry Pi & Device Simulator

4.7.1. Solution

As the final solution needs live data from the sensors, I choose a Raspberry Pi due to its low price. Due to large libraries that are compatible with the Raspberry Pi, especially the General-Purpose Input and Output (GPIO), which allows plug-in sensors, I chose Python as the language.

The software can either run as an emulation (using random numbers) or receive data from the connected sensors. The software is adaptable to interpret different sensors by creating a new sensor file and importing it, allowing flexibility for other sensors to be used instead.

I chose the BMP180 [27] as my sensor because it provides both Pressure and Temperature in a single component making it suitable for sensing the "Boiler" component. Furthermore, it does not require any resistors and can be plugged in with the corresponding pins, simplifying the connection and keeping costs down. Refer to Figure 25 to see how the components are connected.

The Raspberry Pi's software to capture and send the data starts as a service as soon the device boots up, making it 'Plug and Play' once configured.

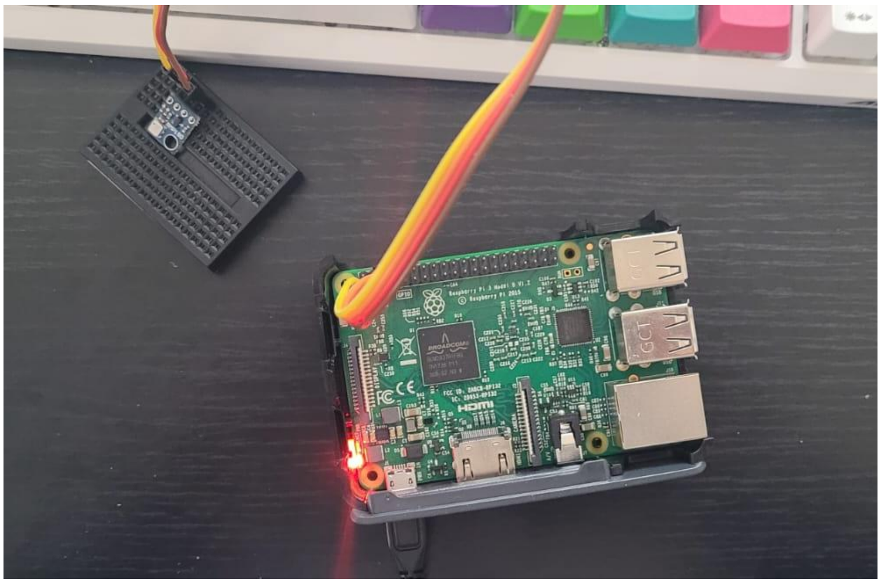


Figure 11 - Raspberry Pi 3B with the sensor attached

4.7.2. Challenges & Difficulties

One of the main challenges that I faced was reading the data from the sensor. Although examples exist to read the data from the BMP180, they were written for Python 2, which has been deprecated. I ended up finding a Python 3 version, which I adapted to fit my requirements.

Furthermore, I faced difficulties with the service starting automatically. I first tried to use a ‘init service’ [28], but I faced problems as it did not start automatically, even with the default services. Therefore I changed to a ‘systemd’ service [29], which works well.

4.8. Unity 3D application

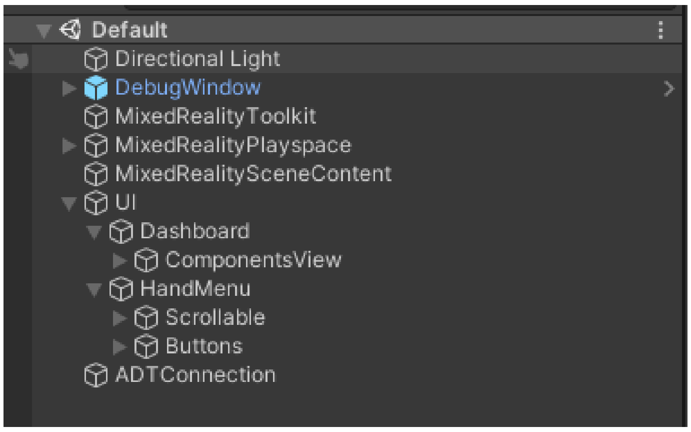
There are multiple ways to build an application for the HoloLens, such as Unity, Unreal Engine or JavaScript. Based on the features available in the table provided by Microsoft [30], Unity was the clear choice with the higher support for advanced features such as world locking tools compared to the alternatives.

The application utilised a library provided by Microsoft called Mixed Reality Toolkit (MRTK). MRTK "provides a set of components and features, used to accelerate cross-platform MR app development in Unity" [31], including UI components such as buttons and development prototype examples. Although alternatives exist, such as Bouvet Development Kit (BDK) [32], the support, documentation and examples are almost non-existent compared to MRTK.

The final solution consists of a single scene in Unity that is run when the application is started. In this scene, there are seven objects (Figure 12):

- Directional Light
- DebugWindow
- MixedRealityToolKit
- MixedRealityPlayspace
- MixedRealitySceneContent
- UI
- ADTConnection

Figure 12 - Objects in Main Scene



4.8.1. Directional Light

This object is included as default when a new scene is created. This is used in the development so that scene is illuminated. If this was missing, then the camera in development mode cannot capture anything and will be black.

4.8.2. MixedRealityToolKit,

This object is where the MRTK can be configured. For this project, I have modified the default profile to enable spatial awareness, changing the spatial awareness to occlusion and disabling the diagnostic system for the final solution.

Enabling spatial awareness allows the HoloLens to interpret its environment, making it possible to "see" the objects and their surface. Changing the spatial awareness to occlusion allowed the holograms to be placed behind objects and to hide the lines that spatial awareness generated (shown in Figure 27).

MRTK includes a diagnostic system that shows the current Frames-per-second (FPS) of the application, along with Central Processing Unit (CPU) and Random Access Memory (RAM) usage. For the final solution, I disabled this as it was no longer needed to monitor, which also helped with performance.

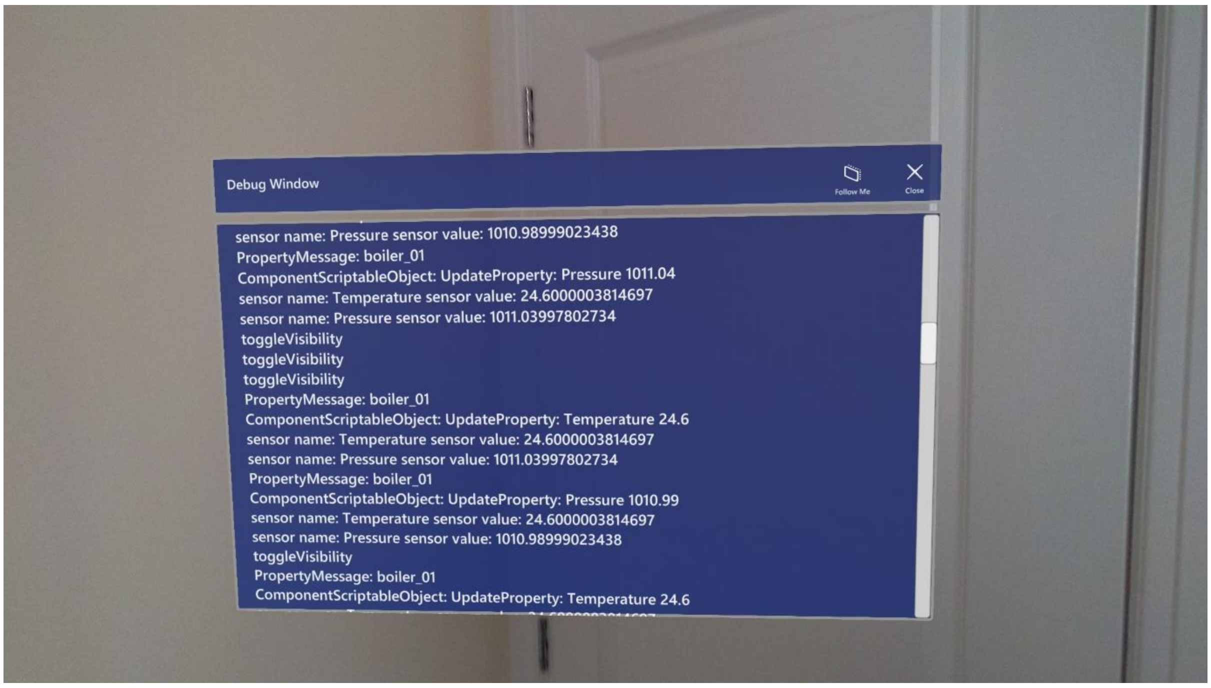
This object is also responsible for interfacing with the Azure Spatial Anchors (ASA). It establishes a session and manages the components dashboard's location.

4.8.3. MixedRealityPlayspace & MixedRealitySceneContent

The MixedRealityPlayspace is where the camera is placed and provides the input. The MixedRealitySceneContent is for initiating mixed reality content to ensure it is consistent across multiple devices [33]. Both are generated when the MRTK settings are applied.

4.8.4. DebugWindow

ASA only works when it is deployed in the HoloLens, and the Unity Play Editor is unable to emulate it. Due to taking it over twenty minutes to deploy any changes, it meant if there was something that was not working as it should and I wanted to debug it, I would need to add debug statements. Although Visual Studio shows the debug statements, it also shows additional information, making it harder to filter. Hence, I have added this so that the user can see the application's logs.



```
sensor name: Pressure sensor value: 1010.98999023438
PropertyMessage: boiler_01
ComponentScriptableObject: UpdateProperty: Pressure 1011.04
sensor name: Temperature sensor value: 24.6000003814697
sensor name: Pressure sensor value: 1011.03997802734
toggleVisibility
toggleVisibility
toggleVisibility
PropertyMessage: boiler_01
ComponentScriptableObject: UpdateProperty: Temperature 24.6
sensor name: Temperature sensor value: 24.6000003814697
sensor name: Pressure sensor value: 1011.03997802734
PropertyMessage: boiler_01
ComponentScriptableObject: UpdateProperty: Pressure 1010.99
sensor name: Temperature sensor value: 24.6000003814697
sensor name: Pressure sensor value: 1010.98999023438
toggleVisibility
PropertyMessage: boiler_01
ComponentScriptableObject: UpdateProperty: Temperature 24.6
```

Figure 13 - Debug Window

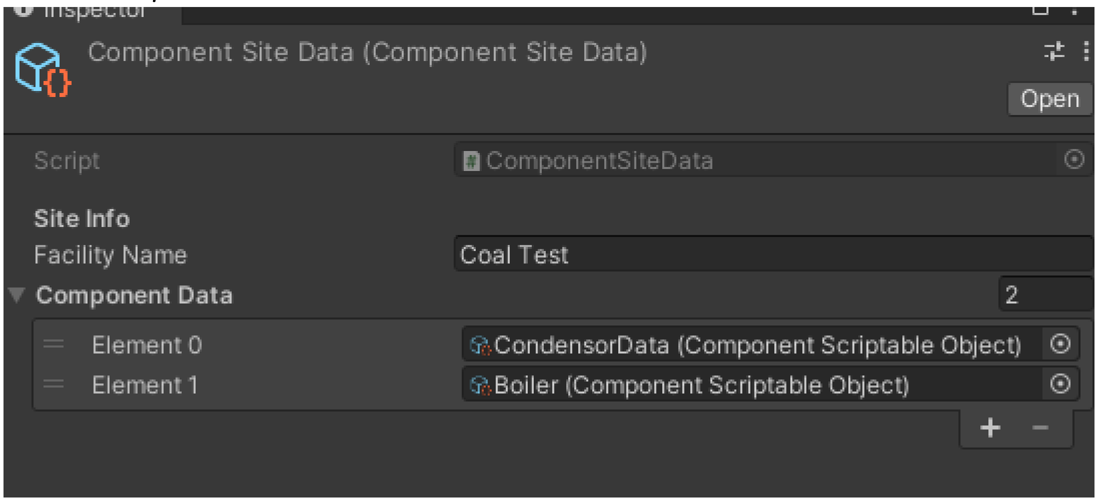
4.8.5. ADTConnection

The application needs to listen for events broadcasted from the SignalR, and this object handles this. When the application starts, it is responsible for connecting to the SignalR instance and updating the data of the models in the application. It utilises a Microsoft library for SignalR that allows it to interface with it, providing event handlers for when a new broadcast message is received so that they can be processed accordingly, such as updating the telemetry of a component.

4.8.6. UI

The UI object handles all of the dashboards. It is composed of two components: Dashboard and HandMenu. It utilises data from the ComponentSiteData object to know the properties of the different components, such as the current status, the sensor values and the if there is an alert active.

4.8.6.1. ComponentSiteData


Inspector

Component Site Data (Component Site Data)

Open

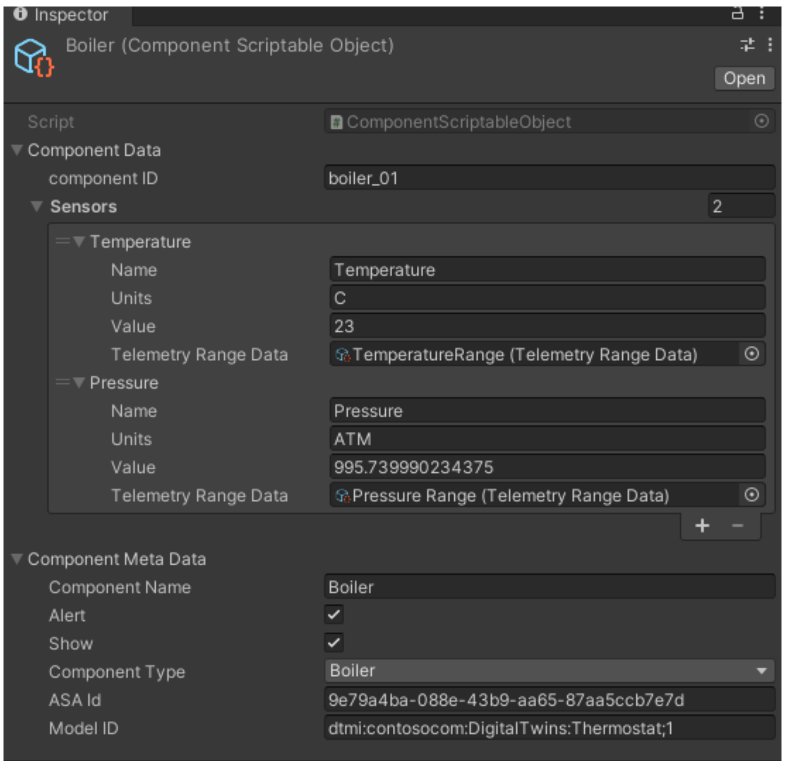
Script	ComponentSiteData
Site Info	
Facility Name	Coal Test
Component Data	2
Element 0	CondensorData (Component Scriptable Object)
Element 1	Boiler (Component Scriptable Object)

Figure 14 - ComponentSiteData

The ComponentSiteData is a scriptable object that contains data about a Site (group of components). It contains the name of the site and a list of all the components available, which are ‘ComponentScriptableObject’, another scriptable object. The advantage of using Scriptable object is that the data is independent of the classes, and most of it is static, i.e. name, sensors, component type, making it perfect for this use [34].

4.8.6.2. ComponentScriptableObject

Each component contains its own scriptable object and holds information pertaining to it. This includes whether the dashboard is currently visible, if there is an alert, the component’s name, the IoT hub’s model name and all of the sensors:


Inspector

Boiler (Component Scriptable Object)

Open

Script	ComponentScriptableObject
Component Data	
component ID	boiler_01
Sensors	2
Temperature	
Name	Temperature
Units	C
Value	23
Telemetry Range Data	TemperatureRange (Telemetry Range Data)
Pressure	
Name	Pressure
Units	ATM
Value	995.739990234375
Telemetry Range Data	Pressure Range (Telemetry Range Data)
Component Meta Data	
Component Name	Boiler
Alert	✓
Show	✓
Component Type	Boiler
ASA Id	9e79a4ba-088e-43b9-aa65-87aa5ccb7e7d
Model ID	dtmi:contosocom:DigitalTwins:Thermostat;1

Figure 15 – ComponentScriptableObject example

As seen in Figure 15, this ComponentScriptableObject belongs to the ‘Boiler’ component. The object details that its id is ‘boiler_01’ (the same name used in Digital Twins) and that it contains two sensors, one for temperature and one for pressure. It also has metadata that shows what type of component it is, the Azure Spatial Anchor ID, along with the state that it is in alert mode. The sensors are also their own scriptable object.

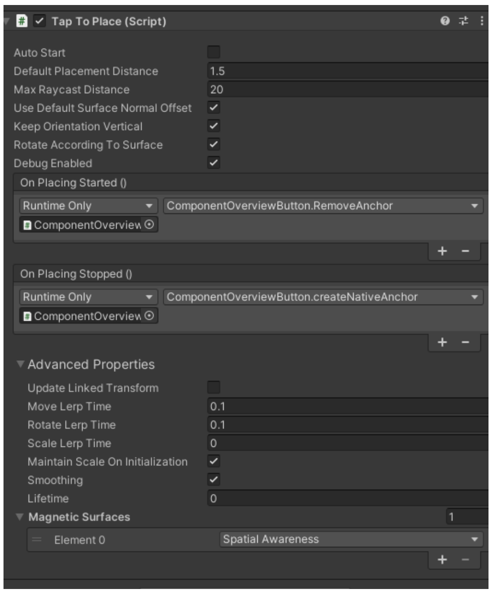
4.8.6.3. Dashboard

The Dashboard component is responsible for housing all of the dashboards. When the application is started, the component will look at all the components stored in ‘ComponentSiteData’ and instantiate the dashboard inside it. Depending on how many sensors the component has, it instantiates a prefab (an object template) called ‘ComponentOverviewButton’ with the correct number of sensor displays and adds as a child to a container object. This will then spawn the object in front of the user’s view.

4.8.6.3.1. ComponentOverviewButton

When the object spawns, the first thing it does is retrieve the latest Azure Spatial Anchor ID, so that the user can then load it to the correct place. The location from the ASA anchor is not loaded from the start due to that this application would likely be used in different places (i.e. during the Viva and testing phases), and hence limiting the location would mean that the application would not work.

The dashboard is coloured in a translucent grey, which is see-through. As one of my non-functional is safety, making the dashboards see-through is one of the ways to achieve this as it does not block the user’s vision.



```
Tap To Place (Script)
Auto Start                          ☐
Default Placement Distance          1.5
Max Raycast Distance                20
Use Default Surface Normal Offset   ✓
Keep Orientation Vertical           ✓
Rotate According To Surface         ✓
Debug Enabled                       ✓
On Placing Started ()
  Runtime Only    ComponentOverviewButton.RemoveAnchor
  ComponentOverview
On Placing Stopped ()
  Runtime Only    ComponentOverviewButton.createNativeAnchor
  ComponentOverview
Advanced Properties
  Update Linked Transform           ☐
  Move Lerp Time                    0.1
  Rotate Lerp Time                  0.1
  Scale Lerp Time                   0
  Maintain Scale On Initialization  ✓
  Smoothing                         ✓
  Lifetime                          0
Magnetic Surfaces                   1
  Element 0                         Spatial Awareness
```

Figure 16 - Tap To Place Script

The top of the dashboard contains the component’s name, which is followed by a gauge chart, a visual representation of the sensor’s data. The sensor value is in the middle, with the units of the value at the bottom. The sensor’s name is underneath the units. If a component has more than one sensor, the sensors are horizontally aligned next to each other.

Tapping the dashboard allows the user to move the dashboard. Here, the spatial mapping data from the HoloLens is used to allow it to interpret the surfaces (Figure 27). This utilises MRTK ‘Tap to place’ script, which implements the functionality. In Figure 16, you can see the script added to the ComponentOverviewButton. When the user taps, it first removes the anchor if it has one, ensuring the dashboard can move when an ASA is applied. When the dashboard is placed down, a native anchor is created to anchor the dashboard so that the user can then proceed to create an ASA anchor. If it did not do this, then the ‘Tap to place’ and anchoring would conflict with each other, making it unable to move.

Next to the main dashboard, there are five buttons:

1. The close button that will close the dashboard
2. Alerts Toggle will silence alerts
3. Close redirect- when the user chooses to be redirected to a component, the directional arrow will stay until the user taps this button
4. Create Anchor- this will create an ASA anchor where the dashboard currently is

5. Find ASA Anchor – this will use the ASA anchor's data and try to find the location that the dashboard should be in

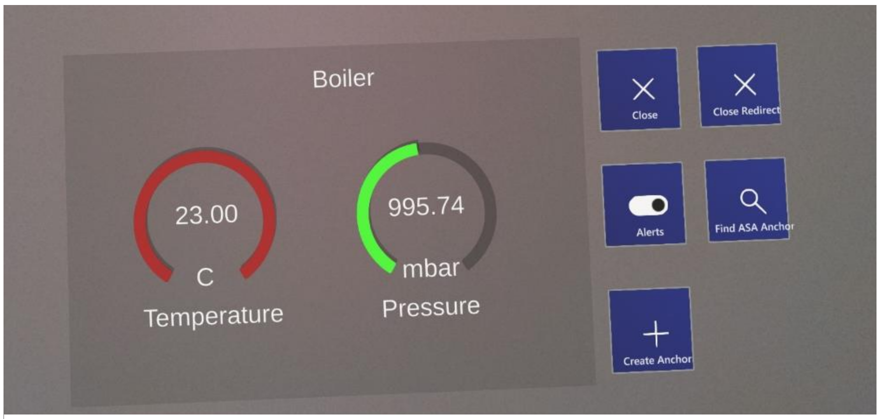


Figure 17 - Dashboard UI

4.8.6.3.1.1. Alerting system

One of the requirements is that the user is alerted whenever a sensor's data become anomalous. I have decided that anomalous would mean whenever the data is outside a predetermined range, which is specified in the ComponentScriptableObject sensor list called 'Telemetry Range Data'. Whenever an update to the telemetry is received from SignalR, the new value is compared to the range, and if it is outside, then an alert is raised.

Raising the alert would mean that a notification sound is emitted to warn the user. This is followed by a directional arrow pointing towards the dashboard and an alert symbol showing up on the dashboard. If the user wants to turn off the notification, they can toggle the switch on the left, which silence any notification from that dashboard.

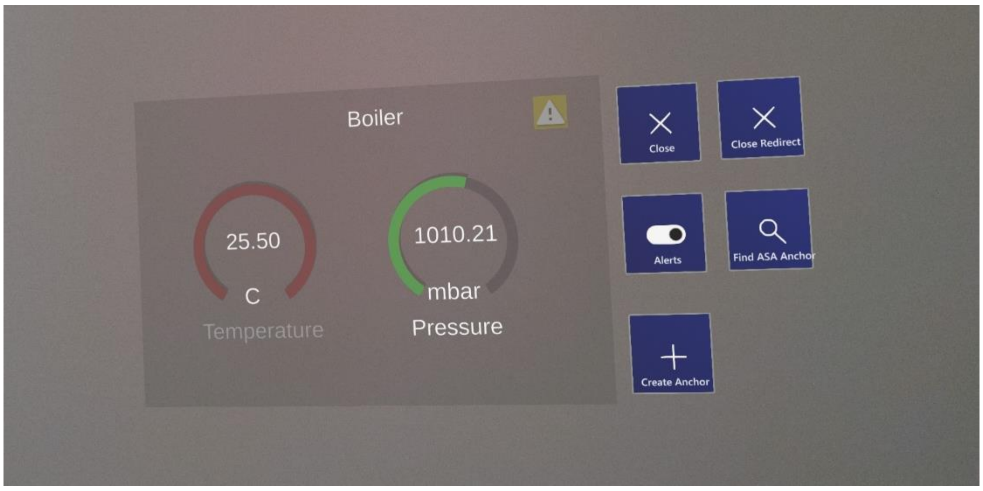


Figure 18 - Dashboard with the Alert Icon

The directional arrow utilises an MRTK script called DirectionalIndicator [35], and the model of the arrow was taken from a demo scene provided by Microsoft [36], whilst the warning icon is an image that is shown when enabled.

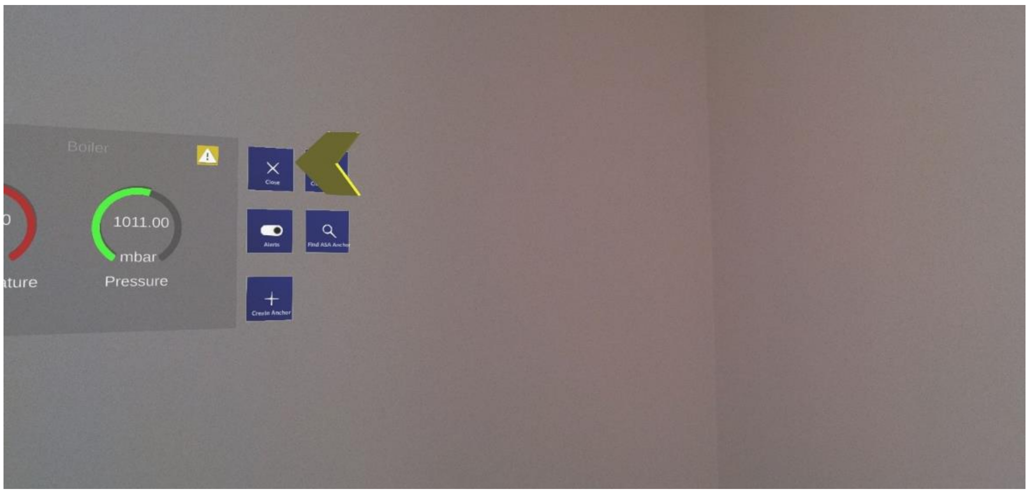
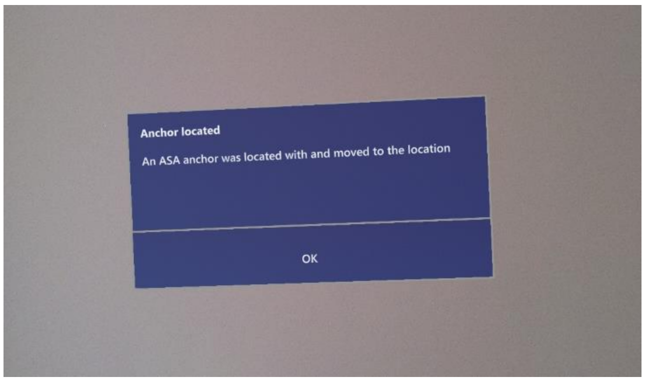


Figure 19 - Dashboard in alert state- the directional arrow only shows when not looking directly at the dashboard and hence why the dashboard is cut off

4.8.6.3.1.2. Azure Spatial Anchors

One of the requirements is to anchor the dashboards so that they are in fixed positions. This is done by utilising Azure Spatial Anchors, which allow to "perceive spaces, designate precise points of interest, and to recall those points of interest from supported devices" [37]. This makes it ideal for anchoring the dashboards to the location.

When the program starts up, the anchor location ids are loaded from Digital Twins by making a GET request (see [4.6.3.1 GetASA]). Then the user can click on the 'Find ASA Anchor' button, which will then create a watcher and try to match the data from ASA to the environment. Once it successfully matches, it will move the dashboard to that location. If the user moves the dashboard, they will need to create a new anchor by clicking the 'Create Anchor' button. This will then override the ASA anchor stored in DT.


Anchor located

An ASA anchor was located with and moved to the location

OK

Figure 20 - Dialogue showing that the dashboard was moved successfully

4.8.6.3.2. Difficulties & Challenges

One of the significant challenges that I faced was to modularise the UI to allow different sized components to display the data. The original plan was to make the background dynamic and grow with the number of sensors. However, this proved to be complicated due to the way that Unity handles the size, and it would make it harder to have the text lined up. In the end, I decided to make prefabs components that could then be instantiated. This would limit the number of sensors that I could have in a single component to three sensors, but as no component has more than three sensors, this is an acceptable alternative.

Making the 'Tap to place' script work required one sprint. Although the script is provided by MRTK, the documentation provided did not mention that it utilised colliders to make it work. I have only found out after spending hours trying to debug it using a demo scene. Furthermore, by default, the magnetic surfaces option (shown in Figure 16) selected everything, which caused me issues that the dashboard would jump around. Only by trial and error did I learn about changing it to spatial awareness surface only. This was further troubled by being an 'Advanced property' and hence hidden unless expanded, which made it harder to change.

Another challenge that I faced was implementing spatial anchors. Due to the poor documentation, I have ended up following three tutorials. One of the tutorials connected to a third party insecure website to save the anchor's ids (see section 6.3.6 for more details), whilst another implemented many features at once, making it harder to learn what was needed to implement it (see section 6.3.4 for more details).

4.8.6.4. HandMenu

4.8.6.4.1. Overview

The HandMenu is a menu that is only shown when the palm of the hand is facing the user. The purpose of this is to allow the user to see the current dashboards, find their location or open them if they are closed from "the palm of their hands" so to speak.

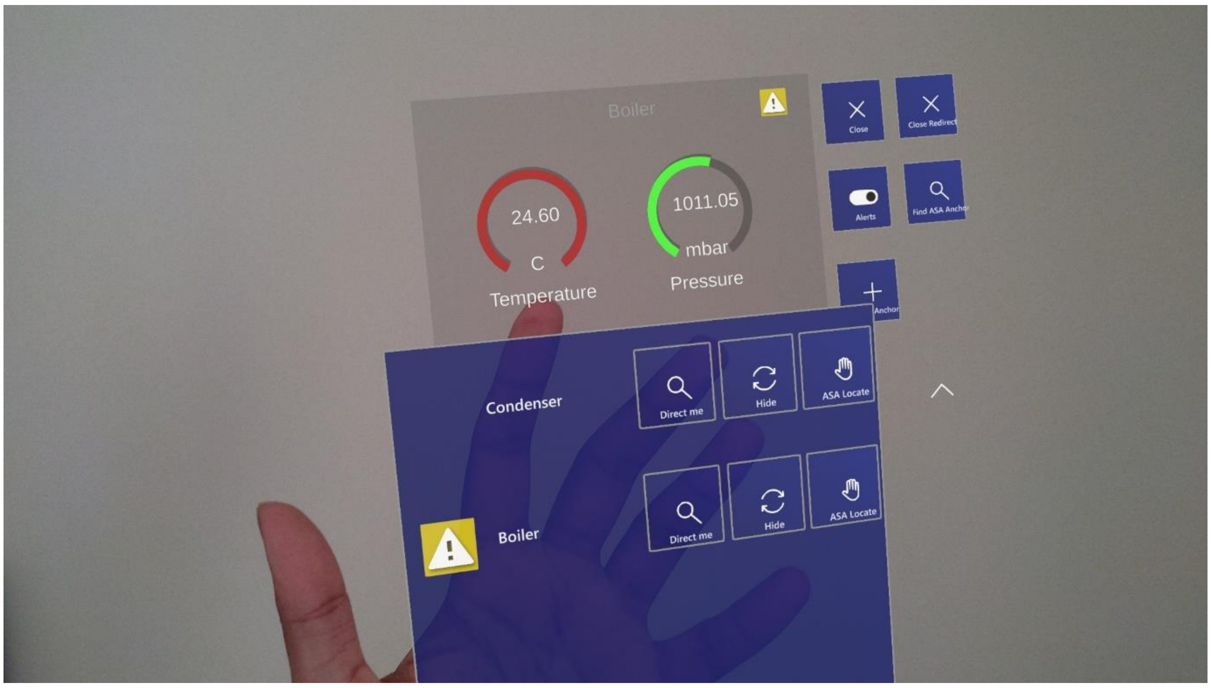


Figure 21 - HandMenu UI -showing the Boiler component in alert mode

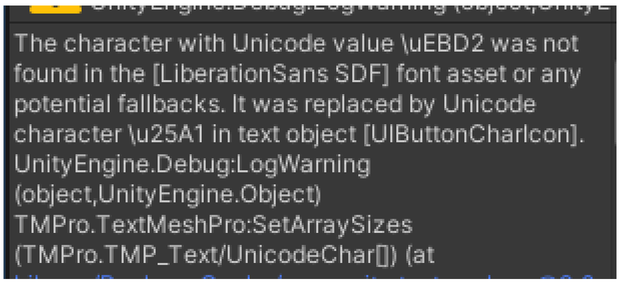
This is shown in Figure 21, where the list of the components is on the left. This list of components is inside a scrollable container, so if there are more components than the hand menu can fit, it will hide the overflow. By using the buttons on the right, the user can scroll up and down the list.

If the component has an alert, then the alert sign will come up on the left. The user can use this information to then redirect to the component by clicking on the 'Direct me' button, which will show the directional arrow towards the dashboard. This is next to the 'Hide' button, which will toggle the visibility of the dashboard, and the 'ASA locate' button will call the same function that 'Find ASA Anchor' does in the dashboard- that is, to load the dashboard into ASA position.

4.8.6.4.2. Difficulties & Challenges

Palm recognition and showing when it is facing the user is a feature provided by the MRTK library. However, I faced the following difficulties:

1. I was unable to enable it when the palm was facing the user. As it turns out, the palm recognition component comes with a solver, and the default value is tracking the eyes; instead, it needs to the hands.
2. I faced some issues regarding the encoding of the buttons, as it could not find some characters. However, the components that used those characters were not enabled, so this error message should not have appeared. To fix this, I had to manually edit the template for the buttons and remove those components.



```
The character with Unicode value \uEBD2 was not
found in the [LiberationSans SDF] font asset or any
potential fallbacks. It was replaced by Unicode
character \u25A1 in text object [UIButtonCharIcon].
UnityEngine.Debug:LogWarning
(object,UnityEngine.Object)
TMPro.TextMeshPro:SetArraySizes
(TMPro.TMP_Text/UnicodeChar[]) (at
```

Figure 22 - Error for Unicode Encoding

3. Toggling the visibility of the dashboard (closing or opening them) could be better represented as a toggle switch, however, because it is inside a scrolling container, the switch position does not change. This is an open bug from 2020 with no solution so far [38], and hence why I opted for a simple button that changes text and icon when it toggles the visibility.
4. As it turns out, changing the icon in the code also possesses its own set of issues; there is a method that allows changing by saying the icon name. As I am using the default icon set, I can just look up the icon names in the Unity editor, except there is a bug that prevents it from showing, requiring a workaround [39]. This should hopefully be fixed in the newer versions of MRTK [40].
5. At the start of the program, once I add the components button prefab as a child object of the scrollable container, the container must be updated. This is done by calling a function called 'UpdateCollection' [41]. However, whenever I called it, nothing happened; it took an obscure Stack Overflow answer to know that I needed a Coroutine to wrap the function to work [42]. This is not the only example that poor documentation slowed me down and added challenges to the development

5. Testing

As with any software developed, testing is an important part of the development cycle. Due to the use of the Scrum methodology, testing was done alongside development towards the end of the user story completion. At the final stage of the project, usability testing was performed. Due to this being a PoC application of an emerging technology, I am not expecting 100% reliability; after all, the aim of this application is to see the current state of the technology and recommendations for it. This does not mean that the application should have bugs; hence it will be tested thoroughly.

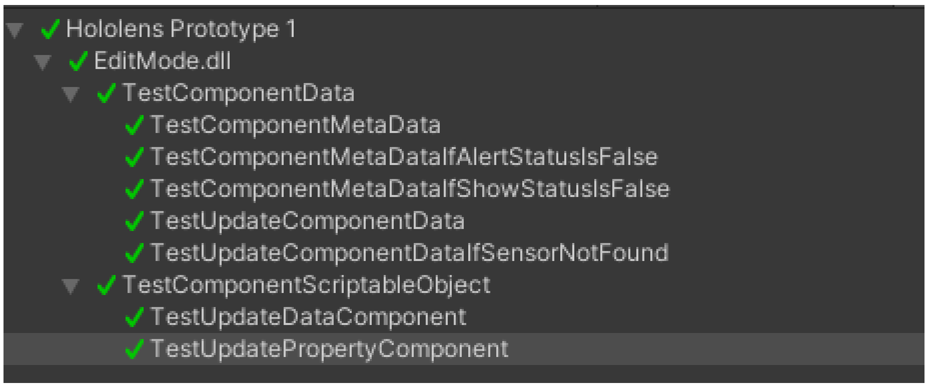
5.1. Unit Testing

Functions must be tested so that any changes in the functions can be checked if they work as they are supposed to. This will make sure there is no unintended behaviour.

The unit tests performed (but not limited to) include:

1. Testing that updating the sensor data
2. Testing that it does not update data if a sensor does not exist
3. Test to see if an alert is triggered when it exceeds the range
4. Test to see if the arrow displays if an alert occurs

For the Unity tests, I utilised the Unity built-in test runner. As seen in Figure 23, all of the tests pass, showing that it works as it is intended to.



```
▼ ✓ Hololens Prototype 1
  ▼ ✓ EditMode.dll
    ▼ ✓ TestComponentData
        ✓ TestComponentMetaData
        ✓ TestComponentMetaDataIfAlertStatusIsFalse
        ✓ TestComponentMetaDataIfShowStatusIsFalse
        ✓ TestUpdateComponentData
        ✓ TestUpdateComponentDataIfSensorNotFound
    ▼ ✓ TestComponentScriptableObject
        ✓ TestUpdateDataComponent
        ✓ TestUpdatePropertyComponent
```

Figure 23 - Passing Unit Tests

5.2. Integration Testing

While unit testing focuses on smaller units, integration testing makes sure that the overall system does not break when changes are made.

To achieve this, I have used Unity play mode testing to test. This differs from the testing done above because it loads a scene where the test is conducted. The result from it is that all the scenarios pass, including the data in the gauge chart changing. Furthermore, the test includes checking if a telemetry event from a sensor successfully broadcasts to SignalR.

All of the tests passed successfully.

5.3. Manual Testing

As there are some elements that can not be tested without deploying to the HoloLens (mainly the Azure Spatial Anchors), I need to resort to manual testing. For this, I prepared three scenarios for testing purposes.

5.3.1. Control

This is the baseline and the one used for development. Here I have created an ASA for a dashboard and restarted the application. Once the application is loaded, I would click on 'Find ASA Anchor' to move the dashboard to the place where the ASA anchor was created. This was then repeated a total of 10 times in different locations throughout the room, and the result was that there all the locations were loaded successfully, leading to a 100% accuracy.

5.3.2. Moving Furniture

This is similar to the one above, but some of the furniture, such as the bed and monitors, have been moved between restarting the application. This would test for variations in the environment, which is typical in an industrial environment. The results were 70% (7/10) accuracy when the environments changed. This is expected as the ASA uses the spatial mapping data to place the dashboards, so if the environment has changed, it will make it harder to match it.

5.3.3. Different rooms

This is a step up from the previous experiment. It involves creating an ASA anchor and then restarting the application in a different room. The results are what is expected, with no ASA anchors being able to find their anchors. This is due to the HoloLens using the spatial mapping from the current sessions to try to match the information from the ASA anchors and hence why in a different room is unable to match it. However, by going from the different room to the room where the ASA anchor was created, it is able to detect and place the dashboard 50% (5/10) of the time. This testing also resulted in an error message coming up saying 'ARAnchor not null'. Unfortunately, I could not solve this bug as I could not reproduce it every time I tried to debug it.

5.4. Usability & Black Box Testing

5.4.1. Methodology

In addition to the testing done on the code, usability testing is essential to determine what works for the intended users. To this end, I performed interviews with four participants:

1. Age 17 - this is a user with no previous experience using the Microsoft HoloLens currently studying A-Levels in Physics
2. Age 22 - a user with limited HoloLens experience (less than 1 hour) studying Computer Science
3. Age 30 - this is also a user with no previous experience using the Microsoft HoloLens and works as an engineer
4. Age 52 - this is a user with multiple hours of experience using the Microsoft HoloLens as an engineer, including remote sessions

As there are users with no previous HoloLens experience, it was necessary to spend time training them so that they are familiar with the interaction. To this end, I first taught them how to run the calibration tool, which also acted as a tutorial for the interactions. This also has the benefit of the HoloLens displays being adjusted for the user's eye so that the holograms would be clearer.

Unfortunately, due to security clearances, I could not test the project in a power station and instead was tested in a room.

5.4.2. Questions

Once they were more familiar with using the HoloLens, I showed them the application, asked a few questions and recorded their answers and behaviours when using the application. The questions asked to the participants included:

- How did you find moving the dashboards to other locations?
- What are your thoughts on visibility and reading the text from the dashboards?
- Do the alert notifications help?
- How reliable is palm recognition?
- What are your thoughts on the hand menu?
- Anything that you don't like about the final project?
- Any improvements that you can think would be helpful?

The questions are worded in a neutral tone so that they do not bias the users toward an answer.

5.4.3. Results

5.4.3.1. User 1 Interview

This user is a 17-year-old Physics and Electronics A-Level student aiming to become an engineer in the future. As this will be the next generation of engineers, their feedback must be considered, even though they do not have experience in the area. The user was unfamiliar with the Microsoft HoloLens but noted that it feels "very much like a big giant touchscreen tablet".

The user said that because they were familiar with the 'drag and drop' functionality in smartphones and tablets, they understood that he just needed to tap the dashboard to relocate it. The user mentioned that he likes that it is "magnetic to the wall" (it is magnetic to the surfaces, not just the walls) because that is where he thinks the dashboards will be. Regarding the visibility, making them grey makes them see-through, which the user mentioned helps increase visibility behind the dashboard.

The alerting notification was the feature that most impressed the user. They felt futuristic that they just are shown wherein the room the faulty dashboard was and hence the faulty component's location. Furthermore, the user stated that the hand menu is something that they had never experienced before and felt convenient to use.

The user mentioned that when anchoring the dashboards using Azure Spatial Anchors, the confirmation dialogues shown are blue, which does not match the dashboard's colour scheme, and he would like a more consistent theming. For the improvements, the user wants the ability to see past data from the sensors and the ability to call other engineers.

5.4.3.2. User 2 Interview

This user is a 22-year-old Computer Science undergraduate student who enjoys learning new technology and thinks Mixed Reality is the future. They had previously used the HoloLens when I demonstrated it to him during the early phases of the development. Nonetheless, I still did the tutorial and calibration, so every user that tested the application was comfortable enough to wear it and use it at the same level.

One of the things that the user liked was the hand menu; they noted that it was convenient to see the components and if they have an alert from "literally the palm of my hands". The user noted that the ability to close and show the dashboards contributed to the visibility and safety of the user as they could see through it. He found the dashboards clear, with the graphs helping him see visually the data and colour coding it accordingly. He mentioned that although the alert arrows were helpful, they felt that a notification dialogue to ask if they wanted to be redirected to the component would be better than assuming that the engineer would want to go there directly.

5.4.3.3. User 3 Interview

The third user is a 30-year-old engineer who has no previous experience using the HoloLens. This user will benefit the most from the technology being ready now as they can immediately start to utilise it.

The user had some difficulties moving the dashboards throughout the room, as he often kept misplacing them. When asked for visibility, the user mentioned that although the dashboards were see-through, sometimes he struggled to read from far as the font was too small, and hence for improvements, he wants to be able to adjust the size of the dashboards.

Regarding the hand menu, the user noted that it was beneficial to see the status of all the components. He described "feel(ing) like Iron-Man", emphasising how futuristic the technology is. The user found the notification sound annoying as it was the same sound as his phone, and hence he got confused. Therefore he recommends being able to change the notification sound as an improvement.

The user also requested the ability to hide some sensors, as some of them do not need to be shown at all times. Furthermore, he requested that the hand menu could contain brief text values of the sensors of the components as a card that can be expanded.

5.4.3.4. User 4 Interview

User 4 is a 52-year-old engineer familiar with the HoloLens, having previously used Microsoft Dynamic 365 Remote Assist (it allows Microsoft Teams to call from the HoloLens). The user is still required to go through the calibration and tutorial setup so that all of the users receive the same instructions.

One of the improvements the user mentioned is the ability to resize dashboards. In a power station, there are multiple components and hence, having the bigger components with bigger dashboards, whilst the smaller and less critical components having their dashboards smaller would help with visibility. Another suggestion was to show a 'dot' where the dashboard is and click on it to expand and show the whole dashboard. This would make the dashboard still visible whilst hidden so that the user is aware that they can view the component's data.

The user was impressed with the Spatial Anchors and found them good at detecting small changes in the environment, but big changes like moving tables and paintings would make them unable to get the dashboards to the location.

5.5. Discussion

The unit testing and integration testing show that the code works as it is supposed to. The manual testing revealed a bug with the Spatial Anchors that is hard to reproduce, and hence I was unable to fix it. However, the accuracy of the anchor locations is what is expected, with the ability to match if the environment does not change, but the accuracy starts to decrease once the room changes.

From the interviews, the alerting system, redirect and dashboards were successfully implemented. The users appreciated the hand menu, making remarks about its futuristic implementation. Furthermore, users are pleased with the visibility as they can see through the dashboards, making it safer when in the field.

There were a couple of improvements and new features that the users suggested:

- A consistent colour scheme as the dialogue boxes do not match the colours of the dashboards
- Ability to see sensor's past data

- Ability to call other engineers
- Make the redirect for the alert optional by asking if they want to go there rather than assume that they do
- Ability to hide sensors in the dashboard as currently, as opposed to hiding the entire dashboard rather than an individual sensor

Some of the features were initially planned, such as seeing the sensor's past data, but I could not implement them due to time constraints and feature prioritisation. The ability to have a consistent theme stems from a lack of something similar to CSS stylesheets support, which would mean that I would have to modify every MRTK component colour manually, a tedious and error-prone task. Being able to resize the dashboards and hide sensors are challenging features; given that I have faced many challenges even implementing dynamic multiple sensor components, this feature would not be easy for me to implement.

All of the users also mentioned that they felt uncomfortable after a few minutes of usage. This is normal and it was expected given that the hard-hat version is bulky and heavy, and that my research also encountered similar problems.

Overall I am happy with the results achieved from the testing. Although there were a couple of bugs, mainly with the Spatial Anchors, I believe that this project still is a viable PoC for the technology. It would be better to test it out in a power station, but security clearances prevented me.

6. Evaluation & Findings

6.1. Aims and Objectives

All of the aims and objectives set out at the start of this report have been met. I have created a PoC application using Unity, which uses data from a modelled power station in Azure Digital Twin. Furthermore, the architecture of the solution is event-driven, making it scalable. This was possible due to my literature review and research at the beginning of the project, which saved me time when developing. The final solution can utilise data from sensors connected to a Raspberry Pi, making it possible to receive the data in real-time, meeting two of the objectives that were laid out at the start.

The interviewees all mentioned the application is futuristic and helps to view real-time data. Furthermore, it follows the best practices and utilises Azure Cloud services, which helped me spend less time setting up the infrastructure. Overall, all the aims and objectives were met.

6.2. Functional and Non-Functional Requirements

6.2.1. Functional Requirements

I have met four out of five requirements that I set out to do. The user is able to see live data from the sensors, whether they are from sensors connected to a Raspberry PI or from a device simulator. Furthermore, when a component's telemetry exceeds the pre-determined range, it can be said that it is behaving erratically, and hence an alert is raised. The user of the application is then notified and will be redirected using an arrow that points towards the dashboard. Furthermore, the dashboards can be fixed in a location using spatial anchors.

Unfortunately, due to the many challenges and difficulties that I faced, I was unable to implement the ability to see past data from the sensors. This was supposed to be implemented in late January, however, the Product Owners (PO) wanted Azure Spatial Anchors to be implemented first as it was a higher priority. This, coupled with other QOL improvement features such as closing and showing the dashboards using the hand menu, left me with no time to implement it and hence it is an unfulfilled requirement. Furthermore, a bug relating to ASA is yet to be fixed.

Overall, even though I have only achieved four out of five initial requirements at the start of the project, I believe that the addition of other features more than makes up for it. Thanks to Scrum, I was able to change the initial plan when the PO's requirements priorities changed to implement the hand menu.

6.2.2. Non-Functional Requirements

One of the non-functional requirements is that performance is essential. There should be less than 15 seconds delay. During testing, the average delay between the sensor sending the data and the HoloLens receiving it was 3.4 seconds. This indicates that the requirement was successfully met.

Furthermore, as all the dashboards are see-trough or can be dismissed like the hand menu, it means that the user has full visibility, and hence their safety requirement is met. The user can also mute notifications, which can help with the visibility, therefore safety hence achieving this requirement.

All of the connections to Azure Cloud utilise HTTPS and API Keys unique to each device, therefore, if one device is compromised, it would limit the damage. Furthermore, Azure handles all of the data encryption when at rest. I have not restricted IP addresses due to the fact that I have a dynamic IP address, and for the purposes of the Viva presentation and demo to the Product Owners, the IP address will also be different; hence I have decided against implementing this for now. This could be achieved by going to the networking tab for the VLAN and restricting from public to specific whitelisted IP addresses.

6.3. Problems Faced & Difficulties

In addition to the challenges described in section 4, I have a few more general findings of the development experience that I would like to highlight based on this project.

6.3.1. Early preview means the customer is the tester

When I was building my device simulator, I used an application provided by Microsoft called Azure IoT Explorer, where I could monitor the event telemetry data generated by the simulator sent to the IoT hub. However, as the tool is in "preview" mode, bugs are expected, but this was the first time using it, and that is why I thought I was doing something wrong, whereas it turned out, the newest version regressed and introduced a bug so downgrading solved the issue. To help get the issue fixed, I have raised a GitHub issue [43]. Although the issue is now closed, the problem still persists, with the latest version not available to download anymore.

6.3.2. Different versions of Unity have different UIs

Many of the tutorials I used for the HoloLens often only worked on a specific version of Unity 3D. This means that there are breaking changes between different versions, so I am unable to follow it, and most of the time, they moved the menu to a new location. The hours I spent following the guides only for this to happen are far too great to count.

6.3.3. Complex Tutorials

One of the major issues that slowed the velocity of the sprints down the most was that the tutorial provided by Microsoft usually boiled down to "here is a premade thing we did- just drag it and run it" (see [44], [45]). Although the tutorial gave me a starting point, it made me reverse engineer to understand how it was doing it and what parts of the code were important. This proved to be very time consuming, as sometimes the feature was not able to work because of an obscure setting, such as the hand menu requiring the tracker to be set to the hand, which is not specified anywhere in the tutorial or the documentation.

6.3.4. Jack of all trades, master of none

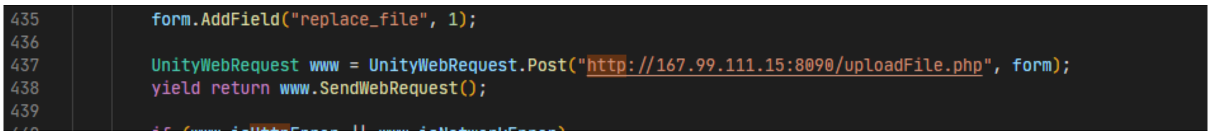
Another challenge that I faced in this project is that some of the tutorials and demos provided by the MRTK were feature-packed, which made reverse-engineering hard because the tutorial did not explain the necessary details. The extra complexity of the demo scene being configurable to multiple demos made it more difficult to understand what was happening. For example, the Spatial Anchor tutorial contains the ability to use machine learning, chatbot and spatial anchors, making the code more complicated to learn just Spatial Anchor alone.

6.3.5. Hardware Requirements

A limiting factor in the development speed was the computer I used for development. The Unity engine and Visual Studio alone required 50Gb of space, which is a significant proportion of my 256GB SSD. This is further coupled with each build taking around 10GB, which, when following tutorials and demo scenes, can add up substantially when not deleted, but doing so would mean I would have to redo the tutorial if I wanted to look at it again. Sometimes I have ended up running out of storage, so when I tried to build, it failed. The error message was not very helpful (Figure 28), and it took a few minutes until I figured out it was due to the lack of storage. Furthermore, Unity and MRTK allow to remote render to the HoloLens when developing, but my computer, more specifically the graphics card (GPU), did not meet the requirements for it as I had an Nvidia MX250, which is not supported, and hence my only option was to deploy to the HoloLens to try the changes. Furthermore, the Unity engine is resource-intensive on the RAM, and hence I suffered occasional crashes when I had too many open tabs on Google Chrome whilst working on the project.

6.3.6. Lack of security

When researching how to implement Azure Spatial Anchor in Unity, I came across this code sample from Microsoft that uploaded a file to an HTTP server with no domain name:



```
        form.AddField("replace_file", 1);

        UnityWebRequest www = UnityWebRequest.Post("http://167.99.111.15:8090/uploadFile.php", form);
        yield return www.SendWebRequest();

```

Figure 24 - Request made to a server using HTTP - a security vulnerability

Using HTTP means that the data is not encrypted and hence vulnerable [46], which poses a risk. Having no domain name means I am unable to verify who owns the domain, and hence I am unsure if it is safe to use. This should have never been in the tutorial, and it is unacceptable that Microsoft would provide this as an example to use it.

6.4. The current state of the HoloLens ecosystem and recommendations

Based on the project and my general finding, there are a few things that I would recommend to Microsoft so that the HoloLens and the developing applications for it can be improved:

1. Make the tutorials beginner friendly – I found that some of the tutorials have a steep learning curve, which increases the learning curve to develop for the HoloLens
2. Provider a wider range of tutorials – the main tutorials focus on building a rover application, which lacks scenarios, such as gaming focus tutorials or UI building. Third-party tutorials are almost non-existent, which makes it harder for the developer
3. Simplify demo scenes – some demo scenes like the Hand Menu have too many features being demonstrated at the same time and thus make reverse-engineering the code harder to understand how it works
4. Ergonomics- from my testing results, one of the key highlights was that using the HoloLens became uncomfortable after 20-30 minutes. Although the testing was done using the hard-hat version of the HoloLens made by a third party that is approved by Microsoft, improved ergonomics for industrial applications would increase the usability of the device and improve user experience.
5. Provide support- the ecosystem would stand to gain from Microsoft providing support to people developing applications. Due to this being a niche system, dedicated support would be beneficial
6. Battery life- the HoloLens has a battery life of 2-3 hours [47] on optimal settings. I have managed to achieve around 1-2 hours when using it, which makes it unsuitable for extended usage such as a full working day
7. Hardware power- when running the application and a video call together, whilst recording, the HoloLens suffered performance issues, skipping frames

Just from the ergonomics point alone, I cannot recommend the application for production use, as it would require the user to use it for more than 8 hours a day when I am unable to use it for more than 30 minutes at a time. I hope Trimble and Microsoft work together to improve it. Moreover, whilst MRTK makes it easier to develop for the HoloLens by providing premade components, developing responsive UI is still difficult, making me conclude that the technology is in a similar state that the 90s web development is; that is, frameworks are still in their infancy, but over time they will grow and become easier to develop using it.

7. Critical Appraisal

7.1. Critical Analysis

During the project, I developed a PoC application for the Microsoft HoloLens that utilised MR to display live data from the sensor in a visual way that could be used in industrial applications.

Looking back at the start of the project, I can confidently say that the research performed at the start helped me design the solution in a scalable way. As it utilises an event-driven architecture, it can accommodate a large number of data ingestion from the sensors, hence making it perfect as power stations and other industrial locations can have over ten thousand sensors. Furthermore, it is not dependent on a single entry point of data- as long the data can be pushed to Azure's IoT hub, the project can utilise the data from it, lending to the flexibility and configurability of the overall solution.

As I utilised Scrum methodology, testing was integrated into it. As seen by the results in the testing phase, only a hard to reproduce bug with ASA was found, and I am disappointed that I was unable to fix it. There was some accuracy drop with the ASA; this was more due to the environmental factors such as lighting and change of locations that confused it, which is expected as the technology is new.

I believe that I did a good job spreading the workload out. The Git commits were spread throughout the sprint and consistent. The only time that there was a gap was during the Christmas break so that I could use it to revise for the exams, which was already accounted for in the original timeline plan made at the start of the project.

One factor that should be highlighted is that I have underestimated the effort required to develop an application with MRTK and Unity. This has led to slowdowns and hiccups, including not being able to meet one of the functional requirements to see past data. This underestimation has led me to spend a full Scrum sprint on making a button to run a function, which shows how steep a learning curve is. In the future, I should spend more time researching the technologies and completing tutorials to familiarise myself with the basics so that when the time comes to start it, I would not be spending time with the basics and could get started with implementing the features.

Furthermore, not being able to implement one feature to see the past data is a point I am not satisfied with. I feel like I was unable to achieve what I set out to do by not having time to implement it. However, the final project does include features such as the hand menu that were not in the original scope but were added after the POs' request. This allowed to utilise unique elements of the HoloLens, as you can see past data in a normal web app, but you are not able to see the status of a component and redirect to it using a web app, making this addition far more valuable.

One of the major points that were raised during the interviews is the ergonomics of Trimble XR10. Users experienced discomfort after 30 minutes, making the device unwearable for eight working hours, and hence making the final solution hard to transition to production use from the PoC state. However, this is the second generation of the HoloLens and the first Trimble device, and hence, the newer generation will hopefully become smaller and more powerful, which would overcome this.

In general, I think this project went well for what it was trying to achieve. It is a challenging application that requires utilising new concepts and ideas, most of them cutting edge technology, such as Digital Twins and the HoloLens; however, I should have learnt more about using Unity before starting the project so as to be comfortable with it and hence have an easier time.

7.2. Context

7.2.1. Social

As this project is an industrial solution, the social impacts are not as well defined. However, being able to be in a remote call and seeing the dashboards whilst keeping the hands free, allows for better collaboration between users as you would not have to switch between devices and applications.

7.2.2. Sustainable

The final solution uses the Azure Cloud, which aims to be 100% renewable by 2025 [48]. This mean that the final solution is working towards a sustainable energy source for its hosting. The architecture of the final solution is event-driven, which makes it sustainable for growth and modifications.

7.2.3. Commercial

As this is an industrial problem that is being solved, it tends to have a large commercial impact. The solution developed is standalone, with no ties in with using infrastructure and being able to accommodate and adapt to the existing system by modifying the data entry point to IoT Hub. This makes the project potentially commercially viable if additional features like seeing past data are implemented. The architecture is event-driven, which lends to the ability to scale, further allowing it to be commercially viable.

7.2.4. Economic

An impact of this solution on the economic aspect is seeing the live data from sensors and being able to redirect the engineer to it; all of it overlayed in the user's vision. As it was designed with a power station in mind, being able to quickly see alerts can prevent power outages. Power outages can have a negative economic impact, and therefore by reducing the likelihood of it, this solution can make a positive economic impact. Furthermore, this project uses emerging technologies like Mixed Reality, which is estimated to be worth $215 billion by the end of 2021, and hence I hope the results from this project can be helpful in the field.

7.2.5. Academic

Although some of the technologies in this project have been theorised for a while, it was not until recently that DT became feasible thanks to Azure Digital Twins. In this project, I showed how it could be used in IoT and Mixed Reality contexts. This usage barely scrapped how it could be used in the future, with the possibility to run predictive models on the telemetry data so it could predict faults before it happens. My evaluation of the technology state and its views could also help the Mixed Reality field flourish by using my recommendation to improve current frameworks to make it easier for developers.

In my literature review, I found that Scaravetti et al. [17] research describes MR and the HoloLens as not being ready to replace conventional tools; this seems to corroborate my findings as I believe more research needs to be done to improve the current technology interaction and possibilities.

7.3. Personal Development

During this project, I ventured into a software field that I had no previous experience with: Mixed Reality. This has been a steep learning curve and frustrating due to poor documentation. The lack of useful tutorials further exacerbated the challenge of developing a PoC application, as can be seen by the challenges and difficulties I faced during the implementation phase. However, this has taught me perseverance and thinking outside the box to solve the challenges, as otherwise, I would not have been able to solve them.

Furthermore, I have become more familiar with cloud services, having utilised Microsoft Azure. Having learnt how to use cloud resources meant that I could focus on development rather than the infrastructure, hence saving me time and learning how useful cloud computing can be. Furthermore, cloud computing is a growing field [49], and hence it has improved my employability skills.

Not only did I learn new technologies, but I have also used my existing knowledge to create APIs for the Azure Spatial Anchors. Although I had created APIs before, they were written in Python and Java, so this was my first time doing it in C#, and I was also using Azure Function apps, which means they were non-standard APIs, but designed for cloud usage and scalability as they are serverless. This means that I have successfully expanded my previous skills.

The most significant and most complex thing that I have learned in this project is how to program using the Unity Engine. Unlike traditional web applications, which only have 2D placement, placing objects in 3 dimensions adds a layer of difficulty, which further exacerbates the extensibility of Unity. For example, at the start of the project, it took me a whole sprint to place a button that runs a function when clicked, however, towards the end, it will now take me less than 10 minutes to achieve the same thing. This shows how I have developed during this project as I have become more experienced in using Unity for development.

Furthermore, utilising Scrum gave me experience in using it and drilled into me the process of working in sprints. This means that during the sprint reviews and planning, I became more familiar with the methodology and increased my confidence during presentations. Moreover, it helped me understand the full development lifecycle.

Overall, this project helped me become more independent and persevere through the many challenges. This is the single hardest project that I have done, and I am proud of what I learned from it.

8. Conclusion

The final output of this project is a PoC Microsoft HoloLens Mixed Reality application that enables engineers to see live data from sensors. It displays data using gauge charts and shows an alert when the predetermined range is exceeded, and hence the engineer will be alerted and redirected to the component. Furthermore, by using the hand menu, the user can see if any of the components are in alert mode or not and can quickly be redirected to it if they wish.

The final solution utilised multiple technologies that I was unfamiliar with at the start, including the use of Unity, Digital Twins, Azure Event Grid, Azure SignalR, and C#. Learning these technologies allowed the final prototype to be scalable and support production size data with thousands of sensors.

Although the HoloLens still needs some work regarding ergonomics to be able to be used in production and work in improving the frameworks to help developers create better applications, this project shows that MR still has untapped potential in the ways that it can be used. I hope that Microsoft and Trimble work together to improve this so that it can successfully be used in industrial environments.

8.1. Limitations

One of my solution's limitations is that it cannot display a component that has more than three sensors. This is a design limitation as I could not create a dynamic UI and instead relied on prefabs.

Furthermore, the dashboard location anchor loading is restricted to the room that the HoloLens is. This is because the HoloLens utilises data from the current sensors, and hence if you try to load the location of the dashboard that is supposed to be in another room, then it will fail as it will not find the corresponding location until the HoloLens looks at it.

Another limitation is that the testing only interviewed four people. It would be better to have a wider range of users, including age differences and experience levels with the HoloLens, so that I could receive more feedback and other people’s perspective. Moreover, testing in a power station could have led to different results, but I was unable to do this due to the lack of security clearance.

8.2. Future work & improvements

For this project, I have used MRTK version 2.7 and updating to MRTK 3, which is due to release later half of 2022, with improvements such as being more flexible and better documentation according to Microsoft [50]. This would solve some of the issues that I faced and allow for theming the components, an improvement request by one of the users who tested the application. Furthermore, it could be further improved by implementing the ability to see past data of the component, a feature that I was unable to implement.

As described in the limitation section, the dashboard can only be loaded if the HoloLens can see the space, and hence it will not be able to redirect to the correct place. This could be further improved by saving the GPS coordinates alongside the ASA id so that the user could be redirected to the approximate location if it is a large site. This would pose problems for smaller sites, where the GPS accuracy would not be enough to differentiate the rooms, in which case, the spatial mapping of the building could be recorded and used as a reference point. Furthermore, an outstanding bug with ASA found during the testing phase still needs to be addressed.

9. Bibliography

[1] Fortune Business Insights, “Internet of Things (IoT) Market Size, Share & COVID-19 Impact Analysis, By Component (Platform, Solution & Services), By End-Use Industry (BFSI, Retail, Government, Healthcare, Manufacturing, Agriculture, Sustainable Energy, Transportation, IT & Telecom,,” 2021.

[2] Oracle, “What is IoT?,” [Online]. Available: https://www.oracle.com/uk/internet-of-things/what-is-iot/#technologies-iot. [Accessed 22 11 2021].

[3] Ring, “Ring Video Doorbell,” [Online]. Available: https://en-uk.ring.com/pages/doorbells. [Accessed 22 11 2021].

[4] D. Milmo, “Amazon asks Ring owners to respect privacy after court rules usage broke law,” *Guardian,* 14 10 2021.

[5] E. Moore, “We need to keep an eye on home surveillance,” *Financial Times,* 26 10 2021.

[6] T. Maineli, “Worldwide augmented and virtual reality hardware forecast update, 2017—2021: CY 4Q17,” IDC, 2017.

[7] N. Gupton, “WHAT’S THE DIFFERENCE BETWEEN AR, VR, AND MR?,” The Franklin Instute, 21 09 2017. [Online]. Available: https://www.fi.edu/difference-between-ar-vr-and-mr. [Accessed 22 11 2021].

[8] I. M. Sauer, M. Queisner, P. Tang, S. Moosburner, O. Hoepfner, R. Horner and R. a. P. Lohmann, “Mixed Reality in Visceral Surgery: Development of a Suitable Workflow and Evaluation of Intraoperative Use-cases,” *Annals of Surgery; Ann Surg,* vol. 266, no. 5, pp. 706-712, 2017.

[9] A. Rasheed, O. San and T. Kvamsdal, “Digital Twin: Values, Challenges and Enablers From a Modeling Perspective,” *IEEE access,* vol. 8, pp. 21980-22012, 2020.

[10] General Electric, “Digital Twin,” General Electric, N/A. [Online]. Available: https://www.ge.com/digital/applications/digital-twin. [Accessed 21 11 2021].

[11] Microsoft, “Azure Digital Twins,” N/A. [Online]. Available: https://azure.microsoft.com/en-gb/services/digital-twins/#overview. [Accessed 21 11 2021].

[12] General Eletric, “Remote Monitoring - Powered by Digital Twins,” N/A. [Online]. Available: https://www.ge.com/digital/industrial-managed-services-remote-monitoring-for-iiot/. [Accessed 21 11 2021].

[13] E. O. Popa, v. Hilten and E. a. B. J. Oosterkamp, “The use of digital twins in healthcare : socio-ethical benefits and socio-ethical risks,” *Life sciences, society and policy,* vol. 17, no. 1, p. 6, 2021.

[14] P. Aivaliotis and K. a. C. Georgoulias, “The use of Digital Twin for predictive maintenance in manufacturing,” *International Journal of Computer Integrated Manufacturing,* vol. 32, no. 11, pp. 1067-1080, 2019.

[15] S. Sirilak and P. Muneesawang, "A New Procedure for Advancing Telemedicine Using the HoloLens," *IEEE access,* vol. 6, p. 60224–60233, 2018.

[16] Kognitiv Spark, “The #1 Takeaway from Microsoft’s HoloLens 2 Announcement,” 19 2 2019. [Online]. Available: https://www.kognitivspark.com/blog/the-number-1-takeaway-from-microsofts-hololens-announcement/. [Accessed 23 10 2021].

[17] F. R. Scaravetti D, “Implementation of Augmented Reality in a Mechanical Engineering Training Context,” *Computers,* vol. 10, no. 12, p. 163, 2021.

[18] Microsoft, “Microsoft Azure Spatial Anchors,” 2021. [Online]. Available: https://azure.microsoft.com/en-gb/services/spatial-anchors/#overview. [Accessed 22 10 2021].

[19] Microsoft, “Connect IoT data to mixed reality with Azure Digital Twin and Unity,” N/A. [Online]. Available: https://docs.microsoft.com/en-us/learn/modules/connect-iot-hololens-azure-digital-twins-unity/. [Accessed 22 11 2021].

[20] GeekForGeeks, “Advantages and Disadvantages of Near Field Communication,” GeekForGeeks, 28 12 2020. [Online]. Available: https://www.geeksforgeeks.org/advantages-and-disadvantages-of-near-field-communication/. [Accessed 05 04 2022].

[21] K. Merry and P. B. F. Bettinger, “Smartphone GPS accuracy study in an urban environment,” *PloS one,* vol. 14, no. 7, pp. e0219890-e0219890, 2019.

[22] A. Grillo, A. Lentini, M. Querini and G. F. Italiano, “High Capacity Colored Two Dimensional Codes.,” in *Proceedings of the International Multiconference on Computer Science and Information Technology*, 2010.

[23] D. Davis, “A Look into Azure Spatial Anchors,” Dave's Two Cent, 19 03 2019. [Online]. Available: https://blog.davemdavis.net/2019/05/16/a-look-into-azure-spatial-anchors/#:~:text=According%20to%20Microsoft%20a%20spatial,holograms%20stay%20precisely%20in%20place.. [Accessed 05 04 2022].

[24] J. Rosales, S. Deshpande and S. Anand, “IIoT based Augmented Reality for Factory Data Collection and Visualization,” *Procedia Manufacturing,* vol. 53, pp. 618-627, 2021.

[25] Gitlab, “What are Git version control best practices?,” Gitlab, N/A. [Online]. Available: https://about.gitlab.com/topics/version-control/version-control-best-practices/. [Accessed 21 04 202].

[26] Microsoft, “Azure Digital Twins Explorer,” Microsoft, 20 10 2021. [Online]. Available: https://github.com/Azure-Samples/digital-twins-explorer. [Accessed 14 04 2022].

[27] MIKEGRUSIN, “BMP180 Barometric Pressure Sensor Hookup,” Sparkfun, N/A. [Online]. Available: https://learn.sparkfun.com/tutorials/bmp180-barometric-pressure-sensor-hookup-/all. [Accessed 24 04 2022].

[28] GeekForGeeks, “What is init.d in Linux Service Management?,” GeekForGeeks, 28 11 2021. [Online]. Available: https://www.geeksforgeeks.org/what-is-init-d-in-linux-service-management/. [Accessed 24 04 2022].

[29] FreeDesktop, “systemd.service — Service unit configuration,” FreeDesktop, N/A. [Online]. Available: https://www.freedesktop.org/software/systemd/man/systemd.service.html. [Accessed 24 04 2022].

[30] Microsoft, “Choosing your engine,” Microsoft, 07 03 2022. [Online]. Available: https://docs.microsoft.com/en-us/windows/mixed-reality/develop/choosing-an-engine?tabs=unity. [Accessed 11 04 2022].

[31] Microsoft, “What is the Mixed Reality Toolkit,” Microsoft, 04 05 2022. [Online]. Available: https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/?view=mrtkunity-2021-05. [Accessed 12 04 2022].

[32] Bouvet, “Bouvet Development Kit,” Bouvet, 12 04 2022. [Online]. Available: https://github.com/bouvet/BouvetDevelopmentKit. [Accessed 12 04 2022].

[33] Microsoft, “Mixed Reality scene content,” Microsoft, 23 11 2021. [Online]. Available: https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/features/experience-settings/scene-content?view=mrtkunity-2021-05. [Accessed 01 05 2022].

[34] Unity, “ScriptableObject,” Unity, 15 10 2018. [Online]. Available: https://docs.unity3d.com/Manual/class-ScriptableObject.html#:~:text=A%20ScriptableObject%20is%20a%20data,by%20avoiding%20copies%20of%20values.. [Accessed 27 04 2022].

[35] Microsoft, “DirectionalIndicator Class,” Microsoft, N/A. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/api/microsoft.mixedreality.toolkit.utilities.solvers.directionalindicator?view=mixed-reality-toolkit-unity-2020-dotnet-2.7.0. [Accessed 01 05 2022].

[36] Microsoft, “Solver overview,” Microsoft, 07 01 2021. [Online]. Available: https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/features/ux-building-blocks/solvers/solver?view=mrtkunity-2021-05. [Accessed 01 05 2022].

[37] Microsoft, “Azure Spatial Anchors overview,” Microsoft, 21 02 2022. [Online]. Available: https://docs.microsoft.com/en-us/azure/spatial-anchors/overview. [Accessed 01 05 2022].

[38] tmhab, “Switch Button doesn't toggle visually when inside a ScrollingObjectCollection with masking on #8738,” Github, 2 10 2020. [Online]. Available: https://github.com/microsoft/MixedRealityToolkit-Unity/issues/8738. [Accessed 24 04 2022].

[39] J. v. Schaik, “Quick and dirty fix for broken Button Icon Set Editor in MRTK 2.7.2,” DotNetByExample - The Next Generation, 23 10 2021. [Online]. Available: https://localjoost.github.io/Quick-and-dirty-fix-for-broken-Button-Icon-Set-Editor-in-MRTK-272/. [Accessed 25 04 2022].

[40] RogPodge, "Fixes for the button iconset inspectors #9999," Github, 21 06 2021. [Online]. Available: https://github.com/microsoft/MixedRealityToolkit-Unity/pull/9999. [Accessed 25 04 2022].

[41] Microsoft, "BaseObjectCollection.UpdateCollection Method," Microsoft, N/A. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/api/microsoft.mixedreality.toolkit.utilities.baseobjectcollection.updatecollection?view=mixed-reality-toolkit-unity-2020-dotnet-2.7.0#microsoft-mixedreality-toolkit-utilities-baseobjectcollection-updatecollection. [Accessed 04 30 2022].

[42] jules_io, "MRTK Grid Object Collection UpdateCollection() not working if called from script," Stack Overflow, 04 01 2021. [Online]. Available: https://stackoverflow.com/questions/65561113/mrtk-grid-object-collection-updatecollection-not-working-if-called-from-script. [Accessed 30 04 2022].

[43] D. C. Duvarcadas, "Azure IoT Explorer [BUG] Unable to stream Telemetry Events #477- GitHub Issue," 29 10 2021. [Online]. Available: https://github.com/Azure/azure-iot-explorer/issues/477. [Accessed 29 10 2021].

[44] Microsoft, "Exercise - Get started with Azure Spatial Anchors," Microsoft, N/A. [Online]. Available: https://docs.microsoft.com/en-us/learn/modules/azure-spatial-anchors-tutorials/3-exercise-get-started-with-azure-spatial-anchors?tabs=openxr. [Accessed 13 04 2022].

[45] Microsoft, "Integrate Azure Cloud Services to your Unity project on HoloLens 2," Microsoft, N/A. [Online]. Available: https://docs.microsoft.com/en-us/learn/modules/azure-cloud-services-tutorials/. [Accessed 13 04 2022].

[46] Google, "Secure your site with HTTPS," Google, 21 04 2022. [Online]. Available: https://developers.google.com/search/docs/advanced/security/https. [Accessed 30 04 2022].

[47] Microsoft, "About HoloLens 2," Microsoft, 02 11 2021. [Online]. Available: https://docs.microsoft.com/en-us/hololens/hololens2-hardware. [Accessed 05 04 2022].

[48] Microsoft, "Azure sustainability," Microsoft, N/A. [Online]. Available: https://azure.microsoft.com/en-gb/global-infrastructure/sustainability/#carbon-benefits. [Accessed 02 05 2022].

[49] ReportLinker, "The global cloud computing market size is expected to grow from USD 445.3 billion in 2021 to USD 947.3 billion by 2026, at a Compound Annual Growth Rate (CAGR) of 16.3%," Yahoo Finance, 05 11 2021. [Online]. Available: https://finance.yahoo.com/news/global-cloud-computing-market-size-081600295.html. [Accessed 30 04 2022].

[50] Microsoft, "MRTK Roadmap," Microsoft, 19 11 2021. [Online]. Available: https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/roadmap?view=mrtkunity-2021-05. [Accessed 27 04 2022].

10. Appendix

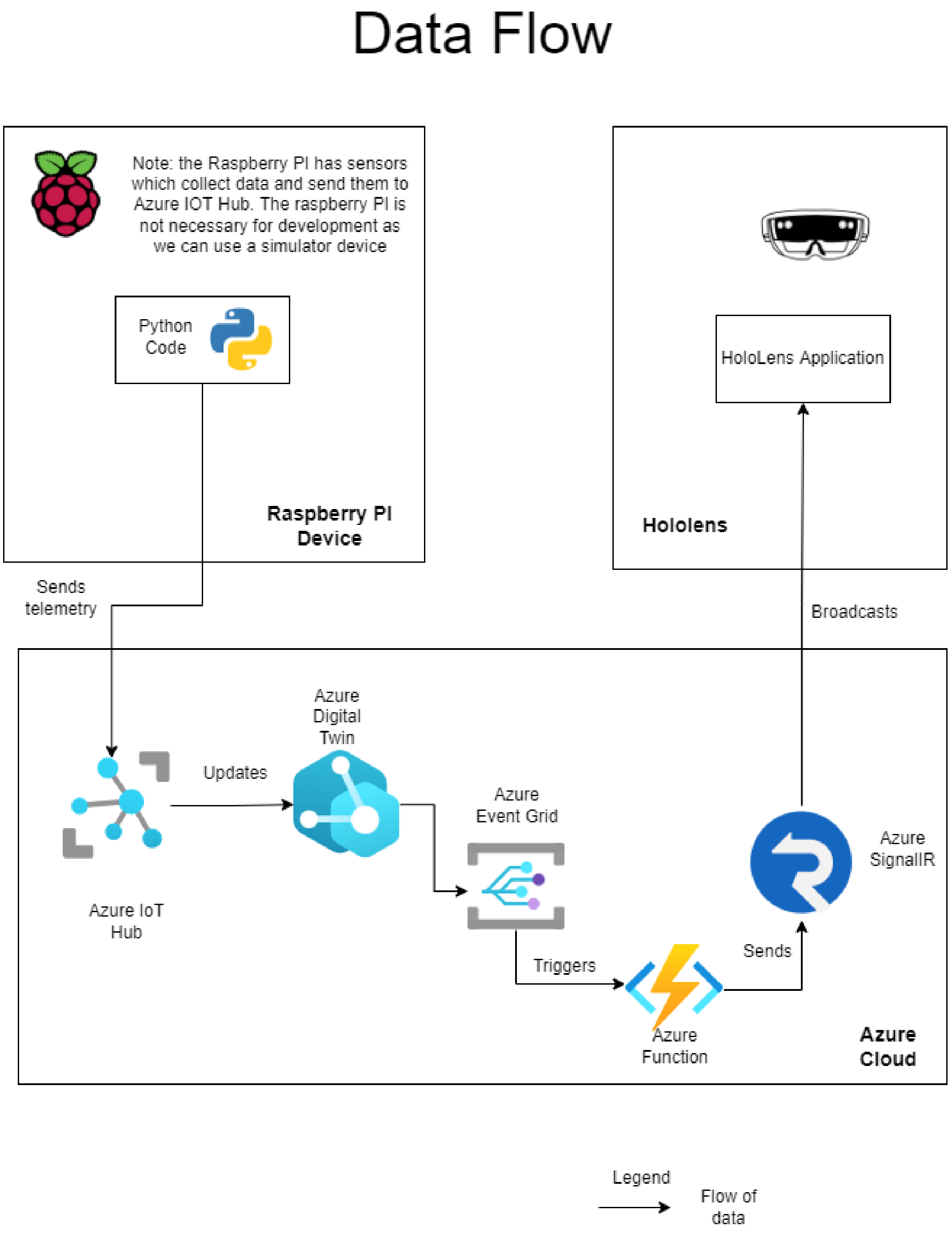


Figure 25 - Original Data flow diagram

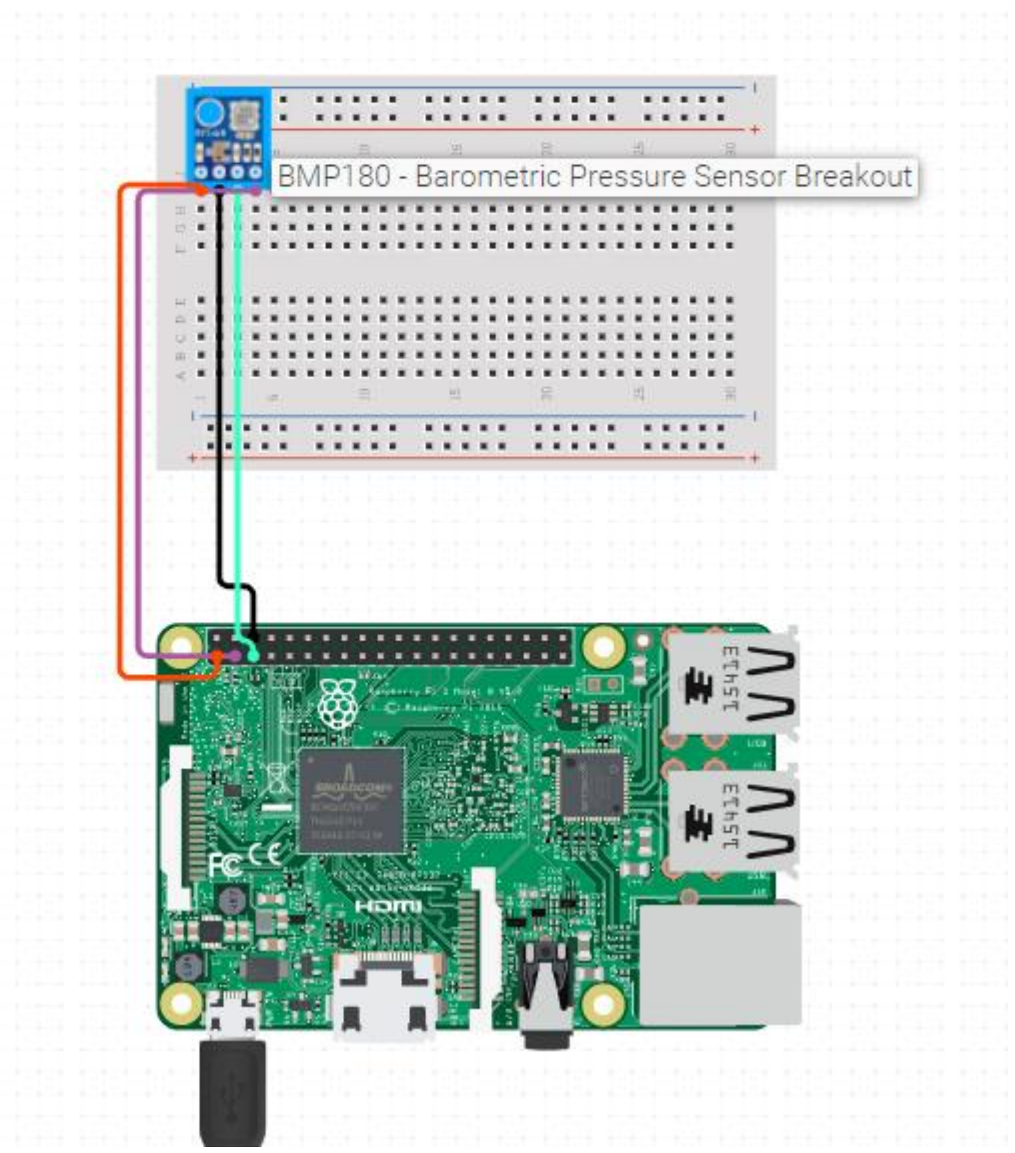


Figure 26 - Connection Layout for the BMP180 and Raspberry Pi 3B in a breadboard

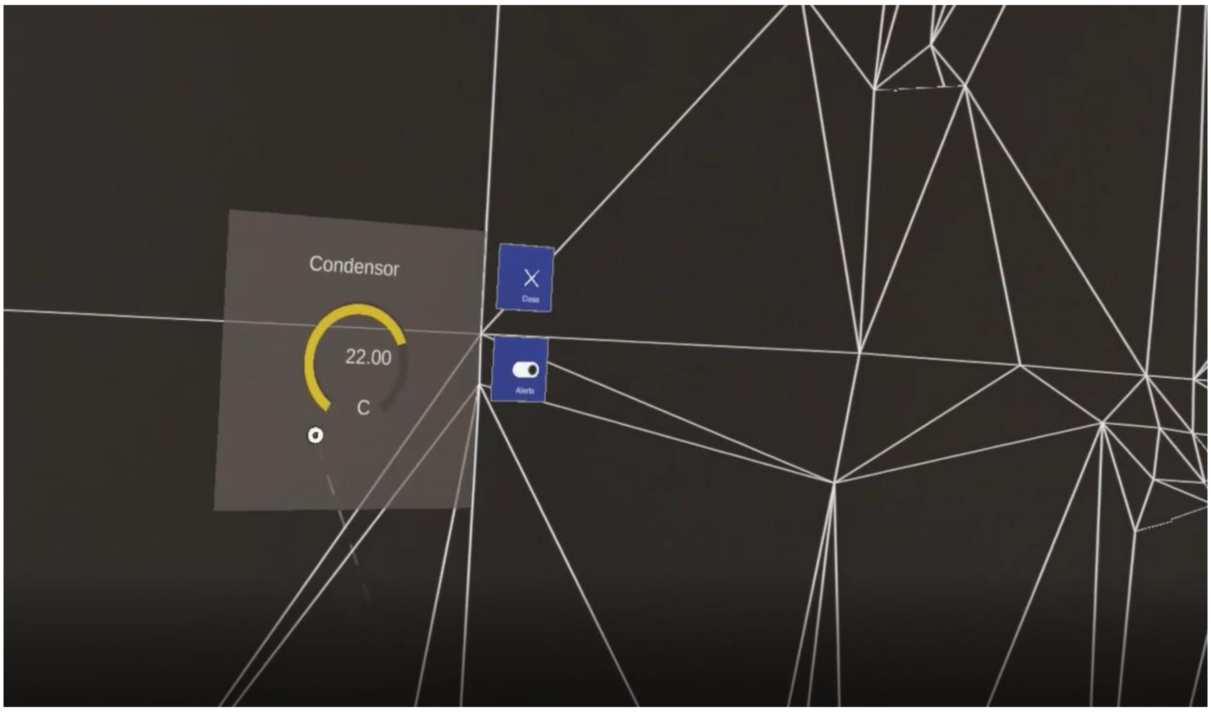
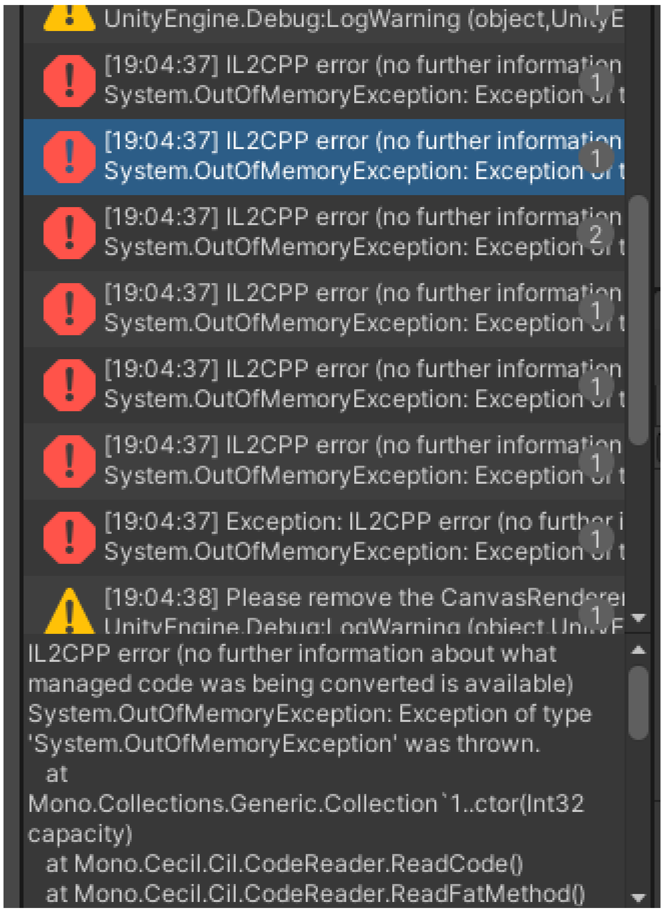


Figure 27 – Spatial mapping shown as white lines (note: picture taken from an earlier version of the project)



```
UnityEngine.Debug:LogWarning (object,UnityE
[19:04:37] IL2CPP error (no further information
System.OutOfMemoryException: Exception of t     1
[19:04:37] IL2CPP error (no further information
System.OutOfMemoryException: Exception of t     1
[19:04:37] IL2CPP error (no further information
System.OutOfMemoryException: Exception of t     2
[19:04:37] IL2CPP error (no further information
System.OutOfMemoryException: Exception of t     1
[19:04:37] IL2CPP error (no further information
System.OutOfMemoryException: Exception of t     1
[19:04:37] IL2CPP error (no further information
System.OutOfMemoryException: Exception of t     1
[19:04:37] Exception: IL2CPP error (no further i
System.OutOfMemoryException: Exception of t     1
[19:04:38] Please remove the CanvasRenderer
UnityEngine.Debug:LogWarning (object,UnityE     1
IL2CPP error (no further information about what
managed code was being converted is available)
System.OutOfMemoryException: Exception of type
'System.OutOfMemoryException' was thrown.
  at
Mono.Collections.Generic.Collection`1..ctor(Int32
capacity)
  at Mono.Cecil.Cil.CodeReader.ReadCode()
  at Mono.Cecil.Cil.CodeReader.ReadFatMethod()
```

Figure 28 - Error when trying to build