The University of Sussex
SCHOOL OF ENGINEERING & INFORMATICS

Coursework declaration – to be included with coursework when not personally submitting

In making this submission I declare that my work contains no examples of misconduct, such as plagiarism, collusion or fabrication of results.

Title of Module: Computing for Business Project

Title of Assignment: Interim Report

Date: 12/11/2020

Interim Report

A cloud-based solution to IIOT server monitoring

Harry Hawkins
Supervisor: George Parisis

Table of Contents

Interim report

1 Introduction

The Industrial Internet of Things (IIOT) is the use of Internet of Things (IOT) devices and procedures in an industrial setting. As we become a more interconnected society, the use of IIOT devices is increasingly prevalent, with the global industrial IoT market expected to reach approximately USD 751.3 billion by 2023 [1]. IIOT devices are commonly found in wind farms, electric vehicle charging stations, power stations, factories, as well as a whole array of industrial environments. These devices often carry out important tasks such as reporting critical information gathered by precise sensors, as well as monitoring levels in the power grid. Because of this, security and stability are of paramount importance. To ensure the stability of these IIOT devices, they must be monitored. By monitoring an IIOT server, you can verify optimal management of CPU, disc, memory resources and network traffic, as well as ensuring a whole assortment of optional and customisable metrics are in check. IIOT devices are often spread across very large geographical areas, therefore, having a monitoring system that is both scalable and secure is of utmost importance. The best way to ensure this is by hosting it in the cloud. The cloud is becoming the new normal for server solutions, due to the low cost, ease of deployment, high flexibility and low maintenance. “The worldwide public cloud services market is projected to grow 17.5 percent in 2019 to total $214.3 billion, up from $182.4 billion in 2018, according to Gartner, Inc.” [2].

Many cloud-based products are based upon an architecture known as Microservices. [3] defines microservices architecture as “a cloud-native architecture that aims to realize software systems as a package of small services. Each service is independently deployable on a potentially different platform and technological stack”. When [3] refers to small services, these often take the form of containers. Containers are typically created using a technology such as Docker. “A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.” [4]. By using a microservices architecture, composed of various containers, we can create a quick to deploy, cloud ready solution. Using Docker, we can ensure that the necessary software for the monitoring system is stable on a variety of different IIOT device operating systems, which is desirable as a user may have multiple types of IIOT endpoints.

Most of the current solutions to this problem are provided by expensive enterprise applications and frameworks, which require licensing on a per node basis as well as yearly maintenance fees (see section 2.1). As well as the upfront fees and annual costs, these products often tie you into a large ecosystem of proprietary additional applications, modules, adaptors and extensions. This makes it difficult to freely innovate and adapt. By using a suite of free open-source, customisable and specially curated software, programming languages and tools, a novel solution can be built to give users a quick, customisable and easy-to-deploy system that provides the same, if not better, experience than that of an expensive enterprise product, combined with the ability to extend or reconfigure at lower cost.

By providing an intuitive and simple web application, users of the monitoring system will be able to quickly and easily add/remove nodes, configure dashboards and select desired metrics. This bypasses the usual manual configuration and installation process of the monitoring system, which requires deep technical knowledge of each component.

1.1 Project Aim and Objectives

1.1.1 Aims

The overall aim of this project is to use a selection of technologies and languages to design and develop a deployable cloud-based solution to IIOT server monitoring. By carefully selecting software, programming languages and frameworks, a system can be designed and developed allowing IIOT servers/devices to be monitored from a cloud based central server.

To facilitate the overall project aim, an intuitive administrator user interface must be implemented, to enable users to quickly and easily connect and configure the IIOT endpoints. This is important because one of the key advantages of this solution over competitors, is the ease of use and configuration. The administration page must be connected to a dashboard page, displaying the monitored metrics in a customisable format.

In order to develop the solution, research into IIOT and cloud technologies must be conducted, to find the resources that will enable the best solution possible.

To ensure the software is fully functional, research into available testing frameworks and technologies is required to find a solution for large scale testing and simulation of IIOT devices. This is a necessary part of the software development lifecycle to ensure that the solution meets intended requirements, as well as being secure and reliable.

1.1.2 Objectives

Primary

- The solution must collect server metrics from remote IIOT devices and display them in a customisable graphical interface. This will provide the user with important information about their endpoints, allowing them to monitor their status, security and stability.
- The solution must use a dashboarding tool to allow users to create customisable dashboards and alarms based on gathered metrics, this should be integrated to the administration page.
- The solution must allow the user to easily and quickly add IIOT device endpoints to the system through a simple user interface. Automating the complex and individual node configuration that would otherwise be required. The system will use automation technology to install, configure and scale the required resources.
- The solution should use a containerised architecture [4] to maintain scalability and stability. Containerisation results in a secure and ready to deploy program.
- The solution must collect time series data, stored in a well-designed database that can scale as required in relation to the number of endpoints.

Extension

- Research the generation of Machine Learning models to predict and understand patterns of failure in the IIOT devices.
- Use a container-orchestration system to orchestrate the deployment of the Docker containers.
- Use and test the final product on a large scale. Simulating 100+ endpoints.

1.2 User needs

The end user of this solution will likely be an IT professional, working for a business that relies on the use of IIOT devices.

Examples of businesses that use IOT:

- Power grid operators [5]
  - IIOT devices are used throughout the power grid to report sensor readings in substations and power generation technologies such as wind turbines.
- Electric Vehicle charging station providers [6]
  - Sensor readings to provide usage information from electric vehicle chargers
- Manufacturing companies [7]
  - Data from IIOT devices can be used to improve manufacturing efficiency.
- Healthcare providers (HealthIIoT)
  - IIOT to "collect personalized health information (e.g., heartbeat, temperature, and humidity data) in real time" [8]

These example businesses are likely to have budget constraints. This solution aims to reduce the time and knowledge needed to configure and deploy an IIOT monitoring system, improving the businesses profit margin.

The end user will have the appropriate knowledge to be able to operate the administration web interface, create and interpret dashboards and understand the necessary metrics. This solution aims to abstract the complex nature of the backend configuration and installation, allowing users to operate the system from the front end, without the technical knowledge of the back end.

Users will want to use the system because metrics gathered from monitoring IIOT devices can allow businesses to maintain and increase security and stability. From monitoring network performance and traffic, to ensuring CPU and RAM levels are satisfactory. This allows businesses to optimise their sensors and devices, as well as reduce downtime.

1.3 Problem area and motivation

Monitoring is becoming increasingly essential to all businesses that use sensors and servers. Industrial Internet of Things (IIOT) businesses benefit greatly from monitoring, as they have many different devices and sensors that are integral to business operations. The monitoring problem arises when factors such as scalability and diversity of devices are considered. A wind farm operator may want to multiply their number of turbines by 10, this means 10 times more sensors and IIOT devices. Adding these endpoints to a monitoring system could be a lengthy and costly process, as well as requiring specialist knowledge of network infrastructure and configuration of various pieces of software. Not only is the configuration of these new devices time consuming, it is also important to ensure that the server receiving the information can scale to accept this many new endpoints. A traditional monitoring server that was originally designed with the intention of managing x devices, may not be able to scale to 10x devices without significant performance issues, especially when handling time series data.

This solution aims to present the user with a way of configuring and administering a full monitoring system, without the complex configuration of the individual components. This means that users can spend more time analysing metrics, by saving time in the configuration process. By automating the installation of monitoring software on new endpoints, and letting users pick options from drop down menus and checkboxes, the usual complex configuration file process is circumvented.

Traditionally, to configure an IIOT monitoring system like this project, a business has two options:

- **Option A:** Purchase a license from a third-party company (see section 2.1), this could be upwards of $3,000p/a [9] and require maintenance and support, as well as being tied into an ecosystem.
- **Option B:** Design and configure a system themselves based on open-source or proprietary software. This requires in depth knowledge of each component, as well as time consuming configuration and installation for each node. This can be very costly for a business, as people with the required specific skills are often expensive.

The motivation for this project is to automate the time/resource consuming parts of Option B, giving a user a simple and quick way to deploy, configure and analyze their IIOT metrics, without the constraints of a costly 3rd party ecosystem. This will allow users to operate the system with a much shallower learning curve, compared to competing products.

1.4 High level architecture overview

Figure 1 is a diagram that shows the overall architecture of the proposed system, without detailing specific technologies. This is a high-level view of the system that helps to convey the overall solution goal.

The area highlighted in purple shows the components visible to the end user, these are the front-end interfaces.

The green area contains the back end, hosted in the cloud. The back end consists of a time series database, a web server, a monitoring service and the back-end code used to automate configuration and run the required services.

Information will be passed from the front-end administration web page to the back end, in order to add, remove and modify devices in the system. Due to the number of components in the system, it can be hard to understand how they fit together, *Figure 1* helps to illustrate the flow of data and information across the different components of the system.

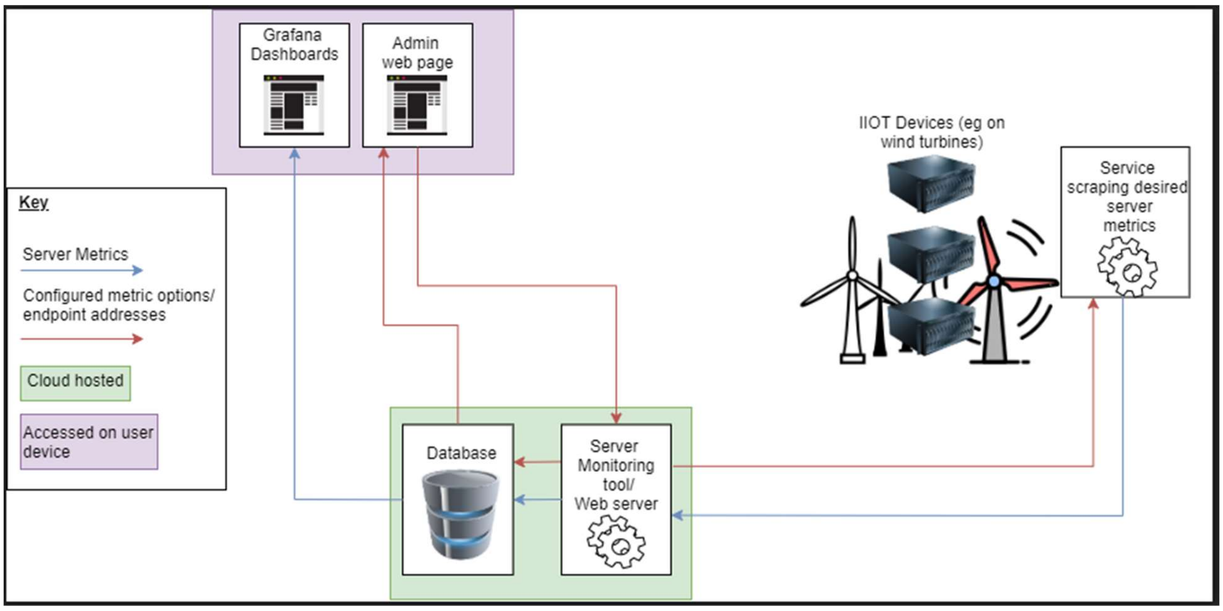

Grafana Dashboards
Admin web page
IIOT Devices (eg on wind turbines)
Service scraping desired server metrics
Key
Server Metrics
Configured metric options/ endpoint addresses
Cloud hosted
Accessed on user device
Database
Server Monitoring tool/ Web server

Figure 1 - High level architecture (Created using diagrams.net)

1.5 BCS Code of Conduct

There are 4 key principles of the BCS Code of Conduct [10]:

You make IT for everyone
This principle is regarding allowing wide access to IT. This project uses open-source languages and resources, which promotes equal access to IT for everyone.

Show what you know, learn what you don't
This principle encourages continuously learning and not taking on tasks that you don't have the skills to complete. I believe that this project is within my skill set and anything that I do not know, I will research and learn.

Respect the organisation or individual you work for
This principle ensures that you act in the best interest of your organisation or client. This includes taking responsibility for actions. This project aligns with the interests of the University of Sussex.

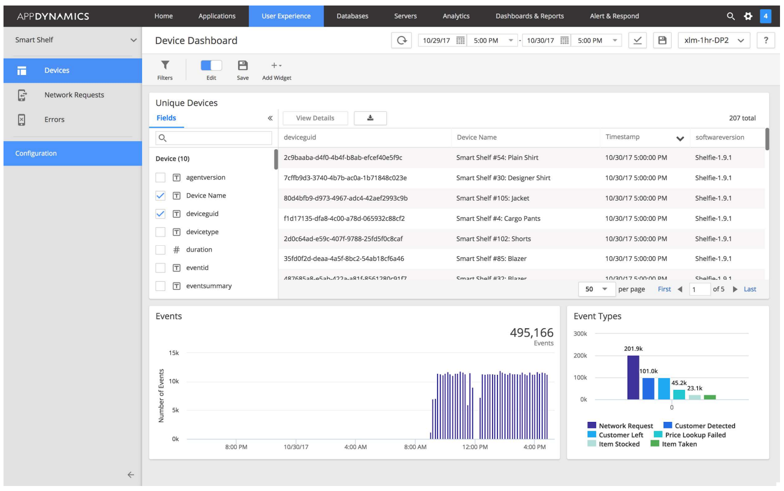
Keep IT Real. Keep IT professional. Pass IT on
This principle relates to acting as an ambassador for the IT industry. It encourages the positive promotion of the industry to the world. This project acts in BCS's best interest to uphold its reputation and support the IT industry.

2 Related work

2.1 Current related products

There are several solutions to server monitoring on the market, including products catered towards IOT. However, many of these products tie you into a technological ecosystem, forcing users to pay for licenses, as well as for maintenance and support. Not only are these solutions costly, some businesses could consider it a security risk to allow a third-party company to manage their server monitoring data, especially in the case of national infrastructure. Ideally, the final project will be an open-source novel solution to IIOT server monitoring, bringing together new and improved features compared to available alternatives. Below is a list of various products that achieve a similar goal to this project, along with their advantages and disadvantages.

2.1.1 AppDynamics



Device (10)
- agentversion
- ✓ Device Name
- ✓ deviceguid
- devicetype
- duration
- eventid
- eventsummary

207 total

deviceguid	Device Name	Timestamp	softwareversion
2c9baaba-d4f0-4b4f-b8ab-efcef40e5f9c	Smart Shelf #54: Plain Shirt	10/30/17 5:00:00 PM	Shelfie-1.9.1
7cffb9d3-3740-4b7b-ac0a-1b71848c023e	Smart Shelf #30: Designer Shirt	10/30/17 5:00:00 PM	Shelfie-1.9.1
80d4bfb9-d973-4967-adc4-42aef2993c9b	Smart Shelf #105: Jacket	10/30/17 5:00:00 PM	Shelfie-1.9.1
f1d17135-dfa8-4c00-a78d-065932c88cf2	Smart Shelf #4: Cargo Pants	10/30/17 5:00:00 PM	Shelfie-1.9.1
2d0c64ad-e59c-407f-9788-25fd5f0c8caf	Smart Shelf #102: Shorts	10/30/17 5:00:00 PM	Shelfie-1.9.1
35fd0f2d-deaa-4a5f-8bc2-54ab18cf6a46	Smart Shelf #85: Blazer	10/30/17 5:00:00 PM	Shelfie-1.9.1
487685a8-e5ab-422a-a81f-8561280c91f7	Smart Shelf #32: Blazer	10/30/17 5:00:00 PM	Shelfie-1.9.1

50 per page First 1 of 5 Last

AppDynamics dashboard [11]

Website: https://www.appdynamics.com/product/end-user-monitoring/internet-of-things

Key features:

- Monitor and manage C/C++ and Java apps by providing full visibility into every line of code.
- Uses AI to solve network issues, auto detect performance problems and identify root cause.
- Visualize revenue paths and correlate customer and application experience.

Advantages:

- Many advanced features.
- Part of cisco, trustworthy.
- Suitable for large enterprises.

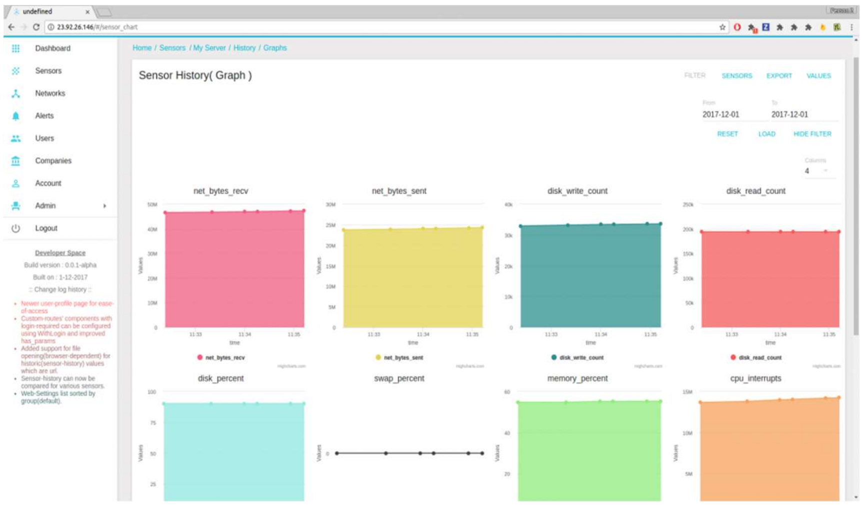
Disadvantages:

- Not open-source, it is a paid product with pricing available on request.
- Possibly too complex for users that only want to monitor device metrics.

Conclusion:

AppDynamics is an industry leading application intelligence platform which provides monitoring, management and analysis of systems. The product is paid for and contains a lot of features, this could be viewed as overly complicated for a customer that simply wants the monitoring aspect.

2.1.2 BASEAPP Swarmsense



Swarmsense dashboard [12]

Website: https://www.baseapp.com/swarmsense/server-monitoring-system-swarmsense-iot-platform/
Key features: "SwarmSense is a fully equipped and self-hosted IoT platform. With SwarmSense IoT platform, you can monitor any type of time-series data. Server metrics are also time-series data, so it is very easy to monitor your server stats with SwarmSense." [12].
Advantages:

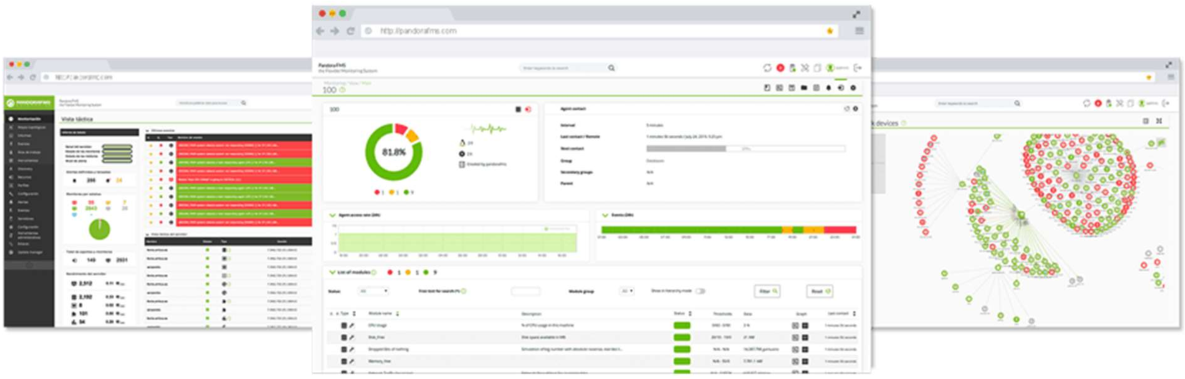
- Time series data allows real time monitoring.
- There is a wide range of optional metrics.

Disadvantages:

- Paid service.
- Requires python knowledge and advanced technical skills to configure and use.

Conclusion:
The functionality of Baseapp is similar to my proposed solution, however it is a paid service that requires technical knowledge to set up. The user must write Python scripts to configure endpoints. This could slow down the deployment and configuration process, as well as making it overly complex for the average IT administrator.

2.1.3 PandoraFMS


PandoraFMS dashboard [13]

Website: https://pandorafms.com/iot-monitoring/
Advantages:

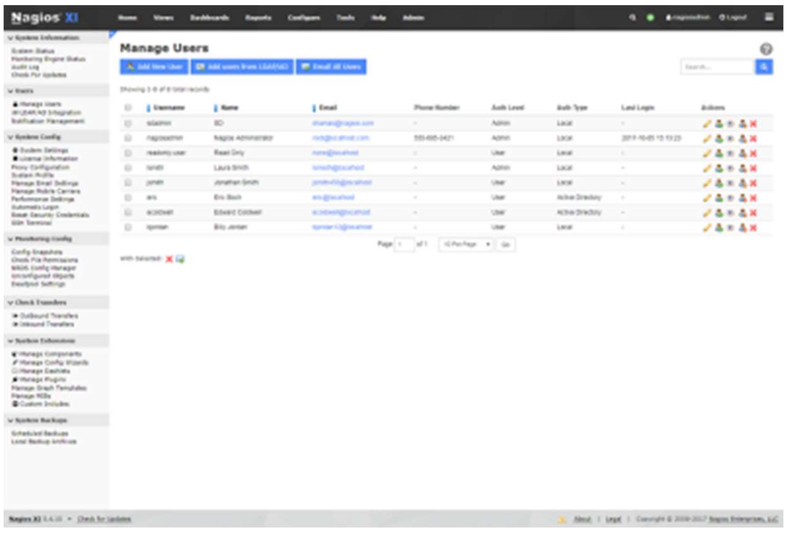
- This tool is similar to BASEAPP Swarmsense; however, it is easier to set up.

Disadvantages:

- Paid service.

Conclusion:
This product is good; however, the paid licensing structure makes this product undesirable compared to my project.

2.1.4 Nagios XI


Nagios XI dashboard [9]
Website: https://www.nagios.com/solutions/server-monitoring/
Advantages:

- This is a well-established, popular tool.

- Thousands of available plugins.

Disadvantages:

- Very expensive ($3,495 a year).
- Strict system requirements [9].

Conclusion:

This is a multifaceted, advanced product for large enterprises. Due to the high cost and strict system requirements, Nagios XI may be inaccessible to smaller businesses.

2.2 Related research papers – literature review

Due to the rising popularity of both IIOT and cloud-based technologies, managing, deploying and operating IIOT implementations has been the subject of extensive research in the last twenty years. There are many resources discussing the different applications of IIOT, as well as the ways in which monitoring can be implemented, in both monitoring server resources and industrial sensor readings.

As expressed in the theoretical paper [14], servers are the core of a network. [14] explores how to monitor servers in a lightweight methodology using the Simple Network Management Protocol (SNMP). This paper discusses the process of monitoring server resources and puts forward four key areas to be monitored: “static information (hardware description, software description, administrator, physical location, etc.), dynamic information (interface traffic, usage of CPU, memory and disk, etc.) network services (HTTP, FTP, DNS, SMTP, POP3, SQL Server database, etc.) and network performance” [14]. These are meaningful metrics that my solution aims to support.

Server monitoring is applicable to many different industries, [15] discusses the importance of flexibly and dynamically monitoring field servers. These field servers are like IIOT devices, as they are used to gather information from sensors in an industrial setting. This paper expands on the importance of monitoring by discussing the advantages in the context of agriculture and environmental sciences.

As shown by [16] , the Process Query System (PQS) can be used to monitor the state of processes, as well as the entire system. From this, [16] creates models to estimate the system state in order to identify issues with systems. One of my extension objectives is to further research the possibility of using Machine Learning to predict and identify failures by analysing gathered system metrics.

Monitoring is considered integral in many types of computer systems, [17] puts forward designs for a Virtual Machine performance monitoring system, which can be used to identify bottlenecks. As well as [18] which is a system for monitoring performance within a client-server architecture. Although these papers are not relating to the use case of IIOT devices, they contain invaluable information regarding monitoring of computer systems in general.

The value of monitoring IIOT devices in the healthcare industry (commonly referred to as HealthIIoT) is covered extensively in [19]. The paper proposes a system for monitoring sensor readings from HealthIIOT devices such as EGC. It highlights the benefit of hosting the monitoring system in the Cloud, which is a key feature of my solution.

IIOT theory is discussed in detail in [20], this paper helps to propose a robust definition of IIOT, as well as identify gaps in literature and understanding of IIOT, this provides use cases of IIOT as well as information regarding current infrastructure.

The discussed resources are invaluable sources of information that will aid with the completion of this project.

3 Requirements analysis

There are several features that make this project a novel solution. The system will be comprised of only open-source software, this is different to many competitors that operate on a paid enterprise licensing models and do not let users access the source code.

Furthermore, this solution aims to be easier to configure and operate compared to competitors. As seen in section 2.1, many current products require expert knowledge to configure, including having to write config scripts in programming languages and having advanced knowledge of computer networks. This project aims to be usable by anyone with sufficient knowledge of IT, in contrast to competitors which require computer scientists and engineers to operate. The web administration page aims to abstract the complex configuration process, making the solution more accessible.

Scalability and security are a huge influence on this project. By using a microservices architecture and being cloud ready, security and scalability can be achieved, making this solution stand out amongst competitors.

3.1 Functional requirements

Functional requirements		
Reference	Description	Mandatory/ Desirable
F1	The solution shall enable users to add a new IIOT device as an endpoint to the monitoring system	Mandatory
F2	The solution shall store information on the endpoints in a database, allowing endpoints to be modified/deleted	Mandatory
F3	The solution shall automate the installation and configuration of exporter software on IIOT endpoints	Mandatory
F4	The solution shall automate the configuration of new endpoints on the monitoring service	Mandatory
F5	The solution shall provide customisable dashboards with real time	Mandatory

	monitoring data	
F6	The solution shall be able to show historical data as well as current data	Mandatory
F7	The solution shall provide the ability to set warnings and alarms based on gathered metrics	Mandatory
F8	The solution should allow users to predict possible future issues, as well as complex analysis of the metrics	Desirable

3.2 Nonfunctional Requirements

Non-Functional		
Reference	Description	Mandatory/ Desirable
NF1	The solution shall be suitable for cloud deployment	Mandatory
NF2	The connections to the endpoint shall be as secure as possible	Mandatory
NF3	The deployment shall be able to scale automatically based on the number of endpoints	Mandatory
NF4	The administration web interface shall be created using Django - a Python Web framework	Mandatory
NF5	The database shall be able to scale to store time series data to provide quick and easy access to data	Mandatory
NF6	The solution should use a microservices architecture to improve scalability, security and ease of deployment.	Desirable
NF7	The solution should be self-healing. Meaning that if a component goes down, it fixes itself with minimal downtime.	Desirable
NF8	The solution should require minimal initial configuration, reducing the need for in depth technical knowledge	Desirable
NF9	The solution should support the use of HTTPS for communication between endpoint and server.	Desirable

3.3 Domain requirements

Domain		
Reference	Description	Action
D1	The system shall be suitable for enterprise IIOT implementations	Make sure the solution is well tested, usable and scalable
D2	The average IT employee should be able to operate the system	Simple UI, abstracted technical areas.
D3	There must be a variety of possible endpoint devices, making the solution suitable for various IIOT use cases	Allow the support of popular protocols and devices platforms

3.4 Software requirements

3.4.1 Dashboard tool

A tool is required to provide customisable dashboards, this will be used to show gathered metrics and monitoring information, as well as alarms and models. It will connect to the administration web page using hyperlinks and/or embedding. This tool can be seen in the purple area of *Figure 2*.

For the dashboard tool, I have decided on Grafana [21]. I have experience using Grafana for monitoring, which makes it an ideal choice, as it will enable me to focus on other areas of the solution, instead of learning a new dashboard tool. Grafana is industry leading and open-source. It has a Docker [4] distribution, making it easy to plug into the system.

3.4.2 Metric exporter

A metric exporter is a service which gathers metrics from a system and exposes the data via a chosen protocol. This service will be located on the IIOT device server itself – as seen in *Figure 1* in the far-right box captioned “Service scraping desired server metrics”.

For this service, there are two industry leading options, Prometheus NodeExporter [22] or collectd [23]. I have decided to use Prometheus NodeExporter. NodeExporter is produced by Prometheus, which is a well-established monitoring provider. NodeExporter and Prometheus are designed to work together. By using both in this project we can reduce compatibility issues between exporter and monitoring service.

3.4.3 Monitoring/Scraping service

A service is required to scrape metrics that have been exported from the IIOT device by NodeExporter, this will act as the entry point for server metrics to the overall monitoring system. The way this service fits into the system can be seen within the green section of *Figure 1*, in the box titled “Server monitoring tool”.
The most popular tools that provide this service are the following [24]:

- Prometheus
- Nagios
- Graphite
- InfluxDB
- OpenTSDB

Prometheus is seen as the industry standard for cloud-based monitoring, which can be deduced from [24] , which shows its comparison to other monitoring tools. As described in [24]:

- Nagios is based on exit codes, with no provided storage solution.
- Graphite is less detailed than Prometheus and harder to integrate, more suited for long term data storage.
- InfluxDB has same constraints as Graphite
- OpenTSDB has the same scope issues as Graphite, coupled with the high complexity of using Hadoop and hbase, this is overly complex for the project.

As a result of these comparisons, I have chosen to use Prometheus. I have experience using and configuring Prometheus and is built for cloud-based monitoring. As well as this, the compatibility with NodeExporter is advantageous.

3.4.4 Programming language

I have chosen to use Python [25]. Python is more suitable for the scripting and automation required within this system. Python has many packages and libraries that provide simple integration with databases and APIs, which will be needed for this project.

3.4.5 Web Development Framework

A web development framework or language must be used to build the Administration web page, as well as integrate the dashboard page. I have experience working with Flask [26], Django [27], React [28] and Bootstrap [29].

Django has been chosen as it is based on Python, which I will be using for the backend codebase. Django has more features than Flask (which is also Python based). By using a python-based web framework, overall system complexity is reduced. “Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design” [27].

The web framework will be used in the purple area of *Figure 1.*

3.4.6 Time series Database

Data will be gathered on a high frequency basis. To store this, a time series database is required. This can be seen in the green section of *Figure 1,* titled “Database”. Prometheus is often used in conjunction with TimescaleDB [30]. TimescaleDB is an open-source implementation of a PostgreSQL time series database. to enable secure and safe storage of time series data. Both Grafana and Prometheus have native integration with TimescaleDB. Therefore, I will be using TimescaleDB with Prometheus.

3.4.7 Containerisation tool

Docker [4] will be used to facilitate Containerisation. Docker is the industry standard for Containerisation technologies, it has the strongest support and plentiful documentation. Containers will be used across the whole system to manage deployment and orchestration of services.

3.4.8 Large-scale testing environment

Once the working solution has been verified on a small scale, by using traditional software development testing techniques such as unit testing, a cloud provider must be selected to test the solution on a larger number of endpoints. This provider has not yet been chosen; it will be selected once further research is conducted during the development phase. The end solution must be cloud deployable, this will be ensured by using a containerised microservices architecture [3].

3.4.9 Protocol

Protocols will be used across the system to allow separate components to communicate. Prometheus and Prometheus NodeExporter [22] communicate between each other over HTTP and HTTPS, this is a secure and simple way to communicate between the IIOT servers and the server monitoring tool on the cloud. HTTP runs on port 80, it is less secure than HTTPS, however HTTP could be useful for testing purposes. In practice, HTTPS is preferred, HTTPS runs on port 443, it is more secure and NodeExporter supports HTTPS with TLS 1.3 and TLS 1.2 [22].

3.5 Use case sequence diagram

Figure 2 is a sequence diagram, showing the use case of a user adding a new end point (IIOT device) to the system. (The terms end point, node and IIOT device/server are used interchangeably to refer to a device that is being monitored). This use case requirement will help in the development of the system, by acting as a guide to verify functionality.

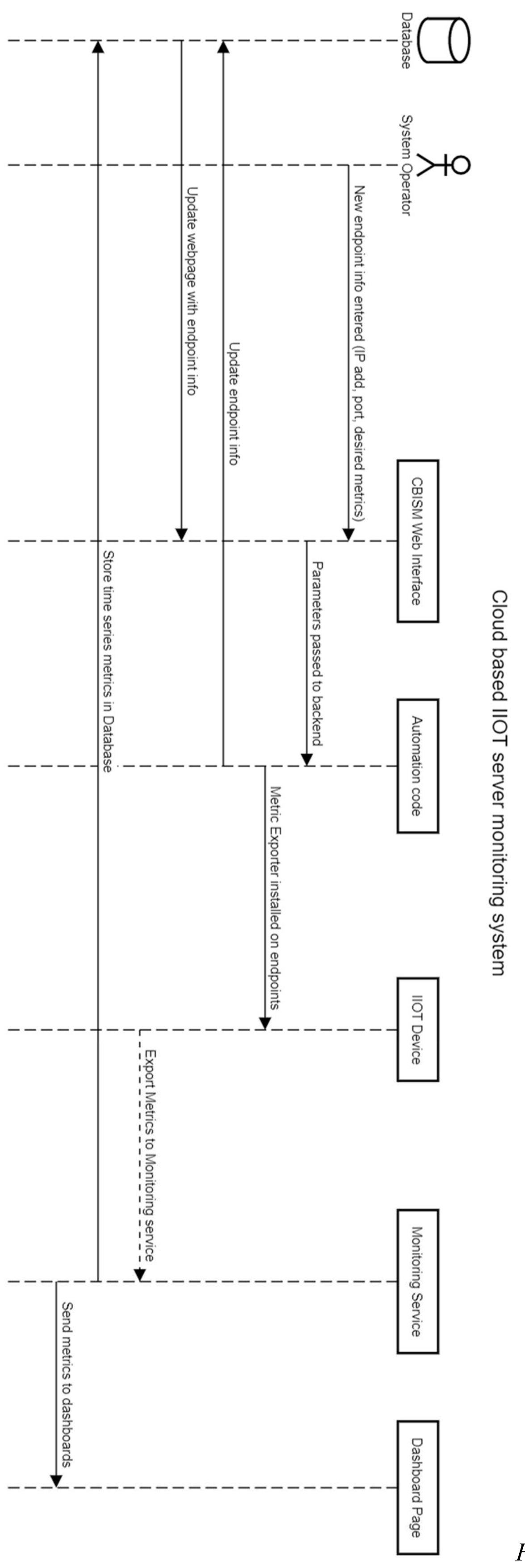

Cloud based IIOT server monitoring system
Database
System Operator
CBISM Web Interface
Automation code
IIOT Device
Monitoring Service
Dashboard Page
New endpoint info entered (IP add, port, desired metrics)
Update endpoint info
Update webpage with endpoint info
Parameters passed to backend
Metric Exporter installed on endpoints
Export Metrics to Monitoring service
Store time series metrics in Database
Send metrics to dashboards

Figure 2 – Sequence diagram

4 Project planning

4.1 Work completed to date

There has been extensive consolidation of requirements to ensure that development can commence. As well as requirements gathering, research and planning has taken place in order to gather the required sources as seen in section 2.2. Furthermore, architecture and design decisions have been considered to enable the selection of software and languages.

4.2 Tasks

The following tasks are a breakdown of the key activities that will take place in order to complete this project.

- Project Proposal, this is attached as appendix 6.2 and has been completed.
- Related work research, this can be seen in section 2.2. This involved the research of related papers that will be useful in aiding the development of this project.
- Requirements, these can be found in section 3, it is likely that more requirements will be discovered during the development process. However, the preliminary requirements have been selected.
- Interim report, the writing of this report.
- Preliminary architecture and designs, this can be seen throughout this report. *Figure 1* shows the overall high-level system architecture.
- Web application coding*, the programming and development of the web applications required in this project.
- Back end coding*, the programming and development of back end scripts and code to facilitate the overall functionality of the solution.
- Large-scale cloud testing, testing the solution in the cloud, simulating large numbers of end points.
- Improvements, changing or adding code to improve and fix the functionality of the system.
- Final report write up, this will be done continuously throughout the development of this project.

*These sections include continuous testing

4.3 Gantt chart

Figure 3 is a Gantt chart; this shows the tasks and milestones discussed in section 4.2, in a more visual format. This was created using the web tool Office Timeline [31]

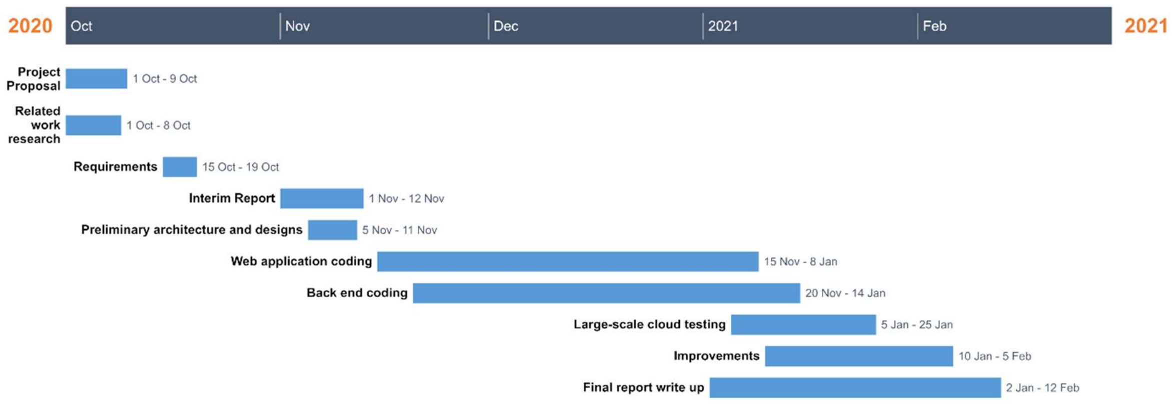


Figure 3 – Gantt Chart

5. Bibliography

[1] MarketWatch, "Industrial Internet of Things (IIoT) Market 2020 Size,Share Global Growth Analysis, Gross Margin Analysis, Industry Leading Players Update, Development History, Business Prospect and Industry Research Report 2023," 24 August 2020. [Online]. Available: https://www.marketwatch.com/press-release/industrial-internet-of-things-iiot-market-2020-sizeshare-global-growth-analysis-gross-margin-analysis-industry-leading-players-update-development-history-business-prospect-and-industry-research-report-2023-2020-08. [Accessed 5 November 2020].

[2] Gartner, "Gartner Forecasts Worldwide Public Cloud Revenue to Grow 17.5 Percent in 2019," Gartner, 2 April 2019. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2019-04-02-gartner-forecasts-worldwide-public-cloud-revenue-to-g. [Accessed 5 November 2020].

[3] A. Balalaie, A. Heydarnoori and P. Jamshidi, "Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture," *IEEE Software,* vol. 33, no. 3, pp. 42-52, 2016.

[4] Docker, "What is a container?," 2020. [Online]. Available: https://www.docker.com/resources/what-container. [Accessed 10 November 2020].

[5] F. A.-T. a. M. Abujubbeh, "IoT-enabled smart grid via SM: An overview," in *Future Generation Computer Systems, vol. 96*, Turkey, 2019, pp. 579-590.

[6] G. F. Savari, V. Krishnasamy, J. Sathik, Z. M. Ali and S. H. Abdel Aleem, "Internet of Things based real-time electric vehicle load forecasting and charging station recommendation," in *ISA Transactions Volume 97*, 2019, pp. 431-447.

[7] Y. Lu, P. Witherell and A. Jones, "Standard connections for IIoT empowered smart manufacturing," in *Manufacturing Letters Volume 26*, United States, National Institute of Standards and Technology, 2020, pp. 17-20.

[8] Y. Miao, Q. Tong, K.-K. R. Choo, X. Liu, R. H. Deng and H. Li, "Secure Online/Offline Data Sharing Framework for Cloud-Assisted Industrial Internet of Things," *IEEE Internet of Things Journal,* vol. 6, no. 5, pp. 8681 - 8691, 2019.

[9] Nagios, "Nagios XI - Easy Network, Server Monitoring and Alerting," Nagios, 2020. [Online]. Available: https://www.nagios.com/products/nagios-xi/#pricing. [Accessed 5 November 2020].

[10] BCS, "BCS Code of Conduct," 2020. [Online]. Available: https://www.bcs.org/membership/become-a-member/bcs-code-of-conduct/. [Accessed 5 November 2020].

[11] AppDynamics, "IoT Monitoring: Connected Devices and Internet of Things Applications," [Online]. Available: https://www.appdynamics.com/product/end-user-monitoring/internet-of-things.

[12] Baseapp, "Server Monitoring System with SwarmSense IoT Platform," [Online]. Available: https://www.baseapp.com/swarmsense/server-monitoring-system-swarmsense-iot-platform/. [Accessed 5 November 2020].

[13] PandoraFMS, [Online]. Available: https://pandorafms.com/iot-monitoring/. [Accessed November 2020].

[14] W. Zeng and Y. Wang, "Design and Implementation of Server Monitoring System Based on SNMP," *International Joint Conference on Artificial Intelligence,* vol. 2009, pp. 680-682, 2009.

[15] T. FUKATSU and M. HIRAFUJI, "Field Monitoring Using Sensor-Nodes with a Web Server," *Journal of Robotics and Mechatronics,* vol. 17, no. 2, pp. 164-172, 2005.

[16] C. Roblee, V. Berk and G. Cybenko, "Implementing Large-Scale Autonomic Server Monitoring Using Process Query Systems," in *Second International Conference on Autonomic Computing (ICAC'05)*, Seattle, WA, USA, 2005.

[17] T. Tarui, T. Tanaka, K. Mizuno and K. Naono, "Performance monitoring system, bottleneck detection method and management server for virtual machine system". US Patent US20100268816A1, 2009.

[18] A. Nasuto, G. Cassone and D. Gotta, "Method and system for monitoring performance of a client-server architecture". AT US CN WO EP Patent US7933988B2, 2004.

[19] M. S. Hossain and G. Muhammad, "Cloud-assisted Industrial Internet of Things (IIoT) – Enabled framework for health monitoring," *Computer Networks,* vol. 101, pp. 192-202, 2016.

[20] H. Boyes, B. Hallaq, J. Cunningham and T. Watson, "The industrial internet of things (IIoT): An analysis framework," *Computers in Industry,* vol. 101, pp. 1-12, 2018.

[21] Grafana, "Grafana," [Online]. Available: https://grafana.com/oss/grafana/. [Accessed November 2020].

[22] Prometheus, "Node Exporter," [Online]. Available: https://github.com/prometheus/node_exporter.

[23] collectd, "collectd," [Online]. Available: https://collectd.org/ . [Accessed November 2020].

[24] Prometheus, "Comparison," [Online]. Available: https://prometheus.io/docs/introduction/comparison/. [Accessed 8 November 2020].

[25] Python, "Python," [Online]. Available: https://www.python.org/. [Accessed November 2020].

[26] Pallets, "Flask," [Online]. Available: https://palletsprojects.com/p/flask/. [Accessed November 2020].

[27] Django, "The Web framework for perfectionists with deadlines | Django," 2020. [Online]. Available: https://www.djangoproject.com/. [Accessed 5 November 2020].

[28] Facebook, "React," [Online]. Available: https://reactjs.org/. [Accessed November 2020].

[29] Bootstrap, [Online]. Available: https://getbootstrap.com/. [Accessed November 2020].

[30] TimescaleDB, [Online]. Available: https://www.timescale.com/. [Accessed November 2020].

[31] "Office Timeline," [Online]. Available: https://www.officetimeline.com/. [Accessed 10 November 2020].

6 Appendix

6.1 Meeting Log

6.1.1 Meeting 1, 2nd October 2020

- Discussed Interim report
- Planned to research current projects and tools
- Set task to create a high-level Diagram of architecture
- Set work for proposal and timetable including 18-20 hours

6.1.2 Meeting 2, 16th October 2020

- Set work to look more at competitors and software
- Need to create more detailed diagrams
- Begin work on Interim report

6.1.3 Meeting 3, 30th October 2020

- Continue work on interim report
- Research Django and begin to look at developing web application
- Set up development environment

6.2 Original Proposal

Proposal - A cloud-based solution to IIOT server monitoring

Introduction

As we become a more interconnected society, the use of Industrial Internet of Things (IIOT) devices is increasingly prevalent. They are commonly found in wind farms, electric vehicle charging stations, power stations, factories, as well as a whole array of industries. These devices often carry out important tasks such as reporting critical information from precise sensors and monitoring power levels in the grid. Because of this, security and stability are paramount. To ensure the stability of these IIOT devices, they must be monitored. By monitoring an IIOT server, you can verify optimal management of CPU, disc, memory resources and network traffic, as well as ensuring a whole assortment of optional and customisable metrics are in check. IIOT devices are often spread across the whole world. Having a monitoring system that is both scalable and secure is of utmost importance. The best way to ensure this is by hosting it in the cloud. The cloud is becoming the new normal for server solutions, due to the low cost, ease of deployment, high flexibility and low maintenance.

Many of the solutions to this problem are provided by expensive enterprise applications and frameworks which require licensing on a per node basis as well as yearly maintenance fees. As well as the upfront fees and annual costs, these products often tie you into a large ecosystem of proprietary additional applications, modules, adaptors and extensions. This makes it difficult to freely innovate and adapt. By using a suite of free open-source, customisable and specially curated software, programming languages and tools, a novel solution can be built to give users a quick, customisable and easy-to-deploy system that provides the same, if not better, experience than that of an expensive enterprise product, combined with the ability to extend or reconfigure at lower cost.

Aims

- Use a selection of technologies and languages to design and create a deployable cloud-based solution to IIOT server monitoring
- Implement an intuitive administrator user interface, to enable a user to quickly and easily connect and configure the IIOT endpoints
- Research and explore IIOT and cloud technologies to find the best solution possible
- Explore opportunities for large scale IOT testing and simulation

Objectives

Primary

- The solution must collect server metrics from remote IIOT devices and display them in a customisable graphical interface. This will provide the user with important information about their endpoints, allowing them to ensure security and stability of their system.
- The solution must allow the user to easily and quickly add IIOT device endpoints to the system, through a simple user interface. Automating the complex and individual node configuration that would otherwise be required. The system will use automation technology to install, configure and scale the required resources.
- The solution should use a microservices architecture to ensure scalability and stability. Containerisation results in a secure and ready to deploy program.

- The solution must collect time series data, stored in a well-designed database that can scale as required.

Extension

- Research the creation of models to predict and understand patterns of failure in the IIOT devices
- Use Kubernetes to orchestrate the deployment of the entire system
- Use and test the final product on a large scale. Simulating 100+ endpoints

Skills/Tools/Languages

- Computer Networks
    - The design and implementation of this project will require research and knowledge of internet protocols, firewalls, computer security and network design.
- Web Applications
    - A framework is needed to create the user administration interface. This could be any of the following: HTML, CSS, ECMA, Django, Grafana, Flask
- DevOps
    - To provide scalability and stability, DevOps technologies such as Kubernetes, Docker, Podman, Ansible and Chef may be required
- Server administration
    - Knowledge of how to use and configure Linux servers is necessary for this project. Servers will run on Linux, as this is what IIOT servers typically use.
- Software development
    - As the project involves the development of a tool, software development skills such as creating UML diagrams and programming are needed. Languages likely to include Python, DockerFile and ECMA.

Resources and constraints

Resources

- For the testing of many IOT devices, I will require access to Linux based machines, either VM or cloud based.

Constraints

- If the large number of testing devices is not achievable, the product will be tested on a smaller scale, and research will be conducted to predict and extrapolate the results to a larger scale.

Research Sources

Many different technologies will be used, therefore there are a lot of resources required to develop this project.

Some of the most important resources are:

- https://grafana.com/docs/
- https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/
- https://www.docker.com/get-started
- https://aws.amazon.com/iot-core/?hp=tile&so-exp=below
- https://www.ansible.com/
- https://www.chef.io/

- https://iot.fedoraproject.org/
- https://cloud.google.com/solutions/devops/devops-tech-deployment-automation
- https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/

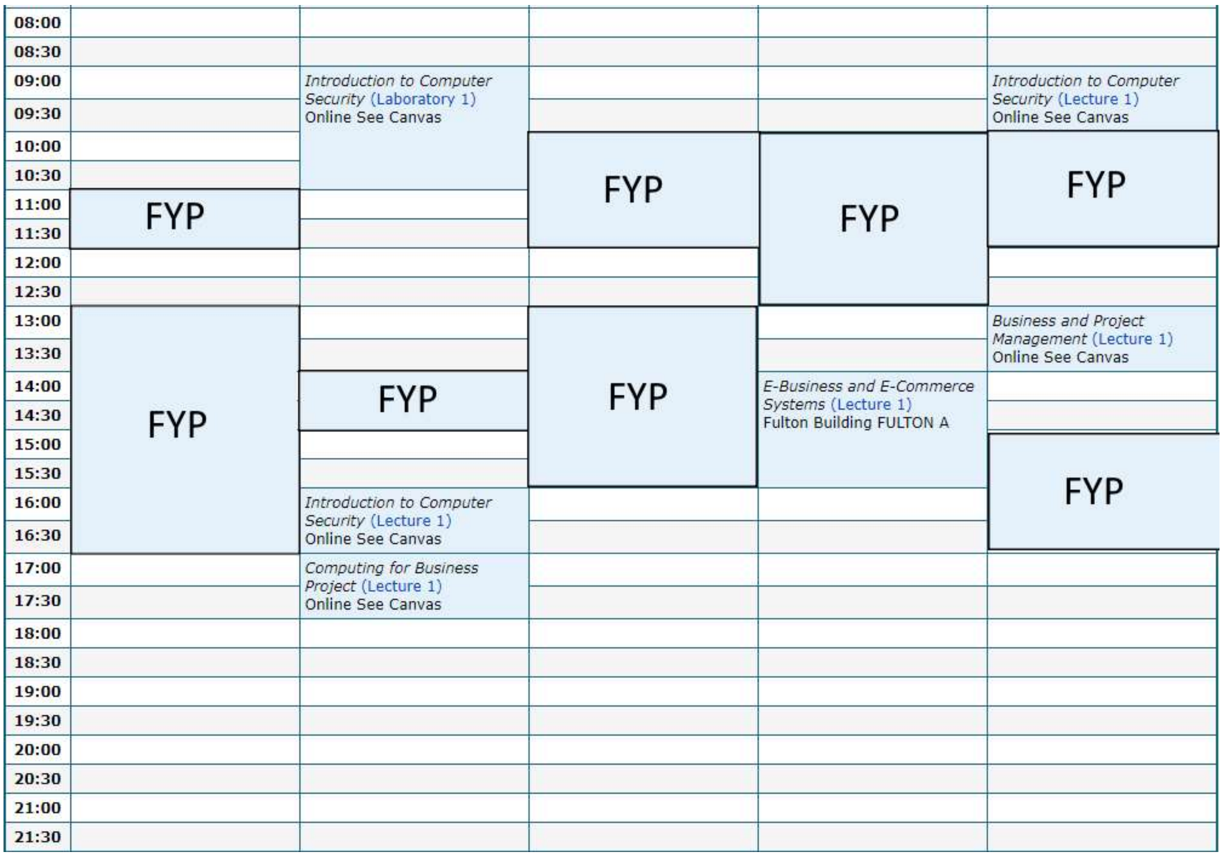
Current solutions

There are already some solutions to IOT server monitoring. However, many of these tie you into an ecosystem, forcing users to pay for use, as well as maintenance and support. It could also be a security risk to allow a third-party company to manage server monitoring, especially in the case of national infrastructure. Ideally, the final project will be a novel solution to IIOT monitoring, bringing together new features, as well as the best features of all available solutions, in a free and open source tool.

- AppDynamics
    - https://www.appdynamics.com/product/end-user-monitoring/internet-of-things
    - AppDynamics provide a system that allows users to:
        - Monitor and manage C/C++ and Java apps
        - Discover application and server topology and interdependencies
        - Unified view of device performance and API dependencies
    - It is not open source; it is a paid product with pricing available on request.
- BASEAPP Swarmsense
    - https://www.baseapp.com/swarmsense/server-monitoring-system-swarmsense-iot-platform/
    - "SwarmSense is a fully equipped and self-hosted IoT platform. With SwarmSense IoT platform, you can monitor any type of time-series data. Server metrics are also a time-series data, so it is very easy to monitor your server stats with SwarmSense."
    - This is like my project; however, it is a paid service that requires technical knowledge to configure and use. Python scripts are used to configure endpoints, which the user must write themselves, this could be difficult for the average user.
- PandoraFMS
    - https://pandorafms.com/iot-monitoring/
      This tool is like BaseApp; however, the lack of configurable dashboards and paid licensing structure makes this product undesirable compared to my project.
- Nagios XI
    - https://www.nagios.com/solutions/server-monitoring/
    - This is a great tool with thousands of available plugins, however it is very expensive ($3,495 a year), as well as requiring strict system requirements. https://www.nagios.com/products/nagios-xi/#pricing

Timetable

18 hours per week



08:00					
08:30					
09:00		*Introduction to Computer Security* (Laboratory 1) Online See Canvas			*Introduction to Computer Security* (Lecture 1) Online See Canvas
09:30					
10:00			FYP	FYP	FYP
10:30					
11:00	FYP				
11:30					
12:00					
12:30					
13:00	FYP		FYP		*Business and Project Management* (Lecture 1) Online See Canvas
13:30					
14:00		FYP		*E-Business and E-Commerce Systems* (Lecture 1) Fulton Building FULTON A	
14:30					
15:00					FYP
15:30					
16:00		*Introduction to Computer Security* (Lecture 1) Online See Canvas			
16:30					
17:00		*Computing for Business Project* (Lecture 1) Online See Canvas			
17:30					
18:00					
18:30					
19:00					
19:30					
20:00					
20:30					
21:00					
21:30					