Journal of Systemics, Cybernetics and Informatics (2023) 21(1), 61-72
https://doi.org/10.54808/JSCI.21.01.61

Multitask Scheduling on Distributed Cloudlet System Built Using SoCs

Manoj Subhash KAKADE
Department of Electrical & Electronics Engineering, BITS, Pilani, Pune Center, India
manoj.kakade@pilani.bits-pilani.ac.in

Anupama KARUPPIAH , Mayank MATHUR, Manasi BIBEKAR, Gaurav BASU , Aaditya RAGHAVAN, Ananth RAGHAV, Pranav LEKSHMINARAYANAN, Swarnab GARANG
Department of Electrical & Electronics Engineering, BITS, Pilani, KK Birla Goa Campus, Goa, India

ABSTRACT [1]

With the emergence of IoT, new computing paradigms have also emerged. Initial IoT systems had all the computing happening on the cloud. With the emergence of Industry 4.0 and IoT being the major building block, clouds are not the only solution for data storage and analytics. Cloudlet, Fog Computing, Edge Computing, and Dew Computing models are now available, providing similar capabilities as the cloud. The term cloudlet was introduced first in 2011, but research in this area has picked up only over the past five years. Unlike clouds, which are built with powerful server-class machines and GPUs, cloudlets are usually made using simpler devices such as SoCs. In this paper, we propose a complete novel distributed architecture for cloudlets, and we are also proposing algorithms for data storage and task allocation across various nodes in the cloudlet. This cloudlet system was built using Qualcomm Snapdragon 410c. We have analyzed the architecture and the algorithm for varying workloads, bandwidth and data storage. The primary aim of the algorithm and the architecture is to ensure uniform processing and data loads across the nodes of the system.

Keywords: IoT, system-on-chip, cloudlets, task allocation, scheduling

1. INTRODUCTION

There Internet of Things (IoT) has evolved rapidly over the last decade. There are billions of edge devices that are a part of multiple application domains such as smart homes, self-driven cars, wearables, smart-grid, smart cities, supply chain management and Industrial IoT, etc. [1], [2]. Billions of devices would naturally generate a large amount of data at high speeds and varying sampling rates. Multiple applications, specifically in the industrial domain, come with hard real-time constraints and, therefore, require predictable network latencies. Current IoT architectures connect the end devices directly or via coordinators to the cloud [3]. Since multiple applications would share the same cloud infrastructure, this leads to significant and unpredictable jitters in network latencies [4]. This is alleviated to a certain extent by using private clouds and Edge computing. While large industrial complexes may opt for a private cloud, medium or small-scale industries may prefer to avoid using a private cloud option. The use of edge computing necessitates the presence of powerful co-ordinators. Only some industrial applications may go in for coordinating nodes [5].
Industrial networks are inherently hierarchical and, hence, are apt for cloudlet-based solutions. The cloudlet layer is an additional layer introduced between the edge of the IoT system and the cloud. The cloudlet layer could handle processing tasks and data associated with short-term monitoring and control. Unlike clouds, which are built with powerful server-class machines and GPU, cloudlets are usually built using simpler devices such as SoCs [6], [7]. Our work focuses on building a cloudlet system that uses the Qualcomm Snapdragon 410c SoC platform. The research proposes a distributed cloudlet architecture along with an algorithm that can be used for uniformly storing data on the cloudlet nodes. We also offer a Task Scheduling algorithm across different cloudlet nodes so that the infrastructure is uniformly loaded. We also present an analysis of this architecture and the data storage and task scheduling algorithms. The algorithms are analysed for varying data storage and CPU utilization percentages under restricted network bandwidth.

The rest of this paper is organised as follows: Section 2 presents cloudlet computing, and Section 3 discusses the proposed cloudlet architecture. Section 4 details the proposed algorithm for task and load balancing in cloudlet systems. We have discussed the results and analysis in Section 5, Section 6 presents an application of digital twining done on cloudlets, and finally, section 7 presents our conclusion and future work.

[1] We would like to express our deeply felt gratefulness to Professor Joseph Rodrigues for his comprehensive and detailed peer-editing of this document, as well as for his constructive and insightful comments for writing this document.

2. CLOUDLETS

As the amount of data produced by industrial end devices is increasing daily, the requirement for the cloud device becomes higher. The resource requirement in terms of processing and storage on the cloud increases. This causes further delays, and cloud services sometimes appear inefficient for real-time industrial applications [8]. A cloudlet is defined as "A small-scale data center or a cluster of computing devices that are designed to provide cloud services to primarily mobile devices, such as smartphones, tablets and wearable devices that are in close proximity to it" [9], [10].
The Industrial Internet of Things system is made up of a large number of heterogeneous nodes that are connected using different wired and wireless networks [11]; generally, in an industrial system (even in the case of small-scale industry), there will be at least two levels of the network hierarchy, level one is filed level network made up of sensors and actuators, level two will be a plant level hierarchy made up of PLCs, and CNC systems. Industry IoT systems generally prefer a localised cloudlet architecture since this would guarantee data privacy, security and predictable jitters[12]. The cloudlet is a low-cost solution that small-scale or medium-scale industries can use to store their data and execute the more complex control algorithms. This solution eliminates the need to pay for high-cost cloud services that may only be available at some points in time, based on the geographical location of the industry[13]. This also solves the problem of large and unpredictable jitters by allowing tasks with short-term, hard real-time deadlines to be executed on the cloudlets. In contrast, the cloudlet can transfer long-running tasks with softer real-time deadlines to the cloud.

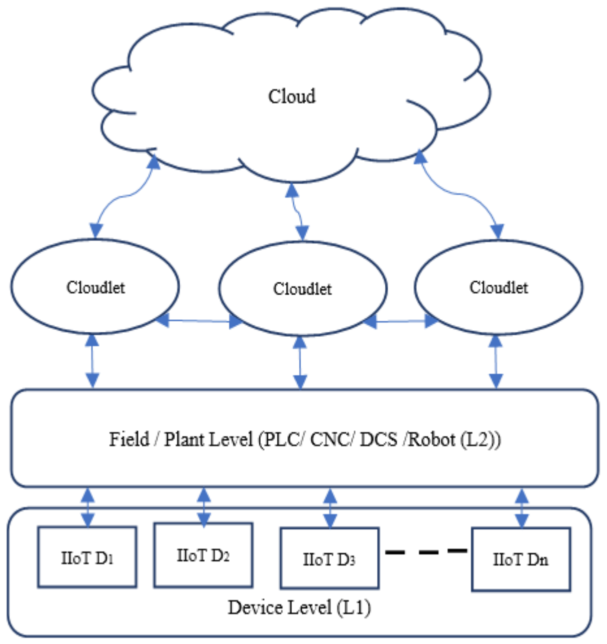


Figure-1 Cloudlet Architecture

Generally, Industrial networks have a minimum of 3 levels of hierarchy: the Device level, which is a network of various sensors and actuators. The primary function at the device level is acquiring data and controlling the devices, which can be termed different Industrial Internet of Things devices. The second layer is the Field level, which is made up of controllers and PLCs; this level controls manufacturing processes or industrial equipment, where it would be interesting to store information every few milliseconds or seconds; this level does not generally have large storage capacities as well as higher processing capabilities. The third and topmost level comprises one or two complex computing devices, usually forming the Edge in an IoT system. The entire level 3 can be replaced by a cloudlet system comprising distributed SoCs connected in an ad hoc network, as shown in Figure 1. The architecture should meet some of the requirements to achieve the needs of Industrial IoT use cases. It includes responsiveness, scalability, usability, and flexibility, Security.
The Cloudlet system in the Industrial IoT networks will sit above the plant-level hierarchy where we have a large number of distributed heterogeneous devices forming a network, where we collect industrial data in real time and have to transmit it to the cloud server for computing and control. If control is real-time, then predictable network latencies with zero jitter become a primary requirement since a cloud-based system caters to a large number and variety of end users; it may not be possible to guarantee latency with zero jitters. While edge computing has been offered as the solution in the case of some Industrial IoT systems, an alternative to this is cloudlet-based systems that have started emerging in the last five years. Cloudlets comprise low-power computing systems with limited processing capability and memory, connected together in heterogeneous networks [14].
The main advantage of using Cloudlet computing in Industrial IoT is as follows [15]

a. Improving system performance
b. Protect data security and privacy
c. Reduce operational cost
d. Reduce communication latency

3. PROPOSED ARCHITECTURE

The proposed system is entirely distributed; as shown in Figure 2, the cloudlet architecture is completely distributed. The architecture comprises multiple SoCs (In our case, Qualcomm Snapdragon 410c) connected to a wireless network using 802.11. End devices connect to various node SoCs, and the node SoC, in turn, connects to each other to form a cloudlet. Each node is aware of the state of all the nodes in the cloudlet; the SoCs proactively send their load status, storage availability and network bandwidth at regular intervals of time "τ". To further reduce the control overhead due to the status update, we suggest using a soft threshold (δ) and a hard threshold ($\gamma 1$, $\gamma 2$), where $\gamma 1$ is the high threshold, and $\gamma 2$ is the low threshold.

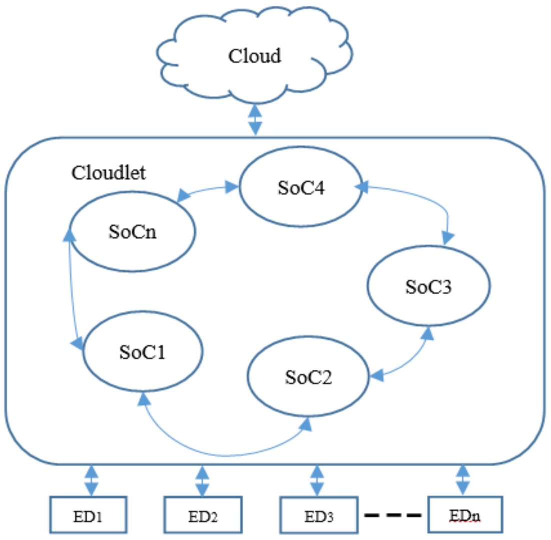


Figure -2 Proposed Cloudlet Architecture

If the current CPU load on a node is ≥ (previous load + δ), then the system status is transmitted to all the nodes. At any point in time, if the CPU load goes beyond γ1, information is immediately sent to all the nodes. If the CPU load on a node is ≤ (previous load - δ), then the system status is transmitted to all the nodes. If the CPU load drops below γ2, then this is broadcast to all the nodes. The same procedure is followed for storage and network bandwidth usage. The next section gives the complete details of the algorithm for task scheduling on the cloudlet for this architecture.

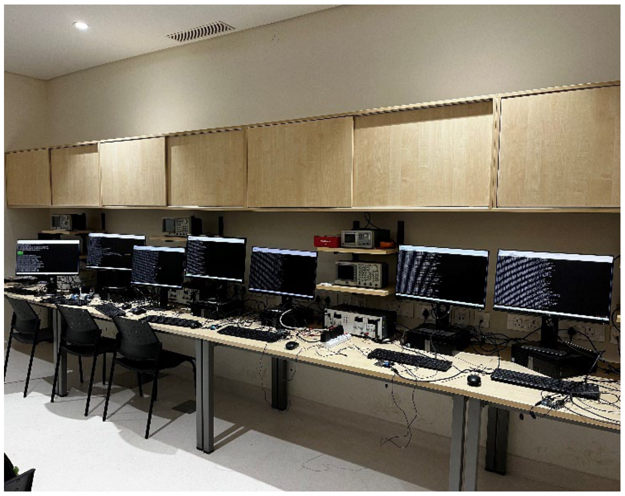


Figure 1. Actual cloudlet implementation using seven 410c

Figure 3 shows the actual cloudlet implementation using seven 410c's. The Dragonboard 410c development board is based on a Qualcomm APQ8016e processor. It comes with advanced processing power, Wi-Fi, Bluetooth connectivity, and GPS with supporting various OS. It also supports internal and external memory storage support, graphics and video support, and many other features, making this development board suitable for rapid software developments and a good candidate for cloudlet implementation.

4. PROPOSED ALGORITHM

Information is stored at each node in a cloudlet. Each node in the cloudlet will have an information manager with the following statistics: -

a. Percentage of CPU utilization
b. Current application software in use
c. Available application software for user
d. Available data space
e. Network bandwidth available for setting up a connection

A. Algorithm explanation

The algorithm for task allocation in the cloudlet is distributed as we follow the architecture described in the previous section. Any node can exist in three states: heavy, medium, and light.
These states are defined concerning the following:

a) CPU load
b) Data storage availability
c) Network bandwidth

a) CPU Load

Heavy state: A node in the cloudlet enters a heavy state when it is running application software or multiple instances of application software. If the CPU utilization percentage is greater than or equal to CPU_H, the node stops accepting requests for running the application. Tasks are migrated to other SoCs.
Medium state: If the CPU utilization percentage is greater than that of equal to CPU_M but lesser than CPU_H, the SoC is in the medium load stage. It can accept application tasks until its load is less than CPU_H.
Light state: If the CPU utilization percentage is less than CPU_M, it is in a light state. In such a state, the SoC can easily accept tasks from a heavily loaded SoC to this SoC, provided the lightly loaded SoC has an affinity towards the task if there is sufficient storage and the network bandwidth and is in the medium state or lesser.

b) Data Storage Availability

Intermediate data storage is done on the cloudlets before a complete backup needs to be done. The amount of storage space can fall into any of the three states: heavy, medium, and light.
Heavy: If the amount of data storage available on the node is lesser than $DATA_H$ MB, then it indicates high storage utilization; hence, any data arriving that is not related to the application currently under execution will be migrated to the other SoCs in the Cloudlet. Acceptance of new tasks also depends upon storage availability. So, while the SoC is in a data storage heavy state, even if the CPU load is medium, it will not accept new application tasks, a condition where the storage

utilization is high but CPU utilization is light, which will rarely occur.

Medium: If the amount of data storage space available is greater than or equal to $DATA_M$ MB but lesser than $DATA_H$ MB, then it is available for data storage. It can also accept new application tasks if the data space required will not cause the device to go into a heavy state.

Light: If the Data storage utilization is less than $DATA_M$ MB, it is in a light state. It can accept new applications as well as data. It can accept tasks migrated to it by the SoC, provided sufficient network bandwidth exists.

c) Network bandwidth

The knowledge of network bandwidth is necessary; for example, if a connection cannot be established due to heavy usage of network bandwidth, the task or data cannot be migrated to a Light state SoC. Network Bandwidth: network utilisation might be high, medium, or low based on the number of active connections and available network bandwidth.

B. Communication between the SoCs

The SoCs initiate a communication under the following conditions:

a. During initial setup – when each SoC starts.
b. When an SoC is lightly loaded regarding CPU, data, and network bandwidth. Under these conditions, it is actively soliciting for tasks or data.
c. When an SoC is heavily loaded regarding CPU, data and network bandwidth. Under these conditions, it seeks to migrate tasks to another SoC on the cloudlet.

C. Algorithm

1. The information manager on each SoC has the information of the node states (CPU, DATA and network).
2. When any new application instance arrives at a node
    a. It checks its state and data storage
    b. If the state and the CPU Utilization and data storage are at medium or light
        i. If the application instance can be run, it then accepts the task and the related data.
        ii. Else, a task migration request is broadcast.
        iii. Each SoC checks if it is in a medium or light state. Each node calculates its capacity to accept a task using several factors:
        Number of Tasks completed in the past window (TP).
        The number of tasks in the current execution window is (TC), and
        The number of tasks (TF) in a chosen queue length is ready for execution.

$$N = w1 * \frac{TP}{NP} + w2 * \frac{TC}{NC} + w3 * \frac{TF}{NF}$$

        Eq. (1)
        Where
        N is the affinity factor.
        NP is the Time window of the past tasks,
        NC is the Time window of the current tasks,
        NF is the Time window for future tasks and
        w1, w2, and w3 are the weights assigned.
        iv. If N < Th (Threshold set by system), then a task migration acceptance is sent after a delay determined by N. If N is high, the delay is more.
        v. If no node with the value N < Th is available, no node responds; hence, the task is sent to the cloud.
3. When any new data arrives at a node
    a. It checks its data storage availability
    b. If the data storage is at medium or light
        i. It then accepts the data.
        ii. Else, the data migration request is broadcasted. The data migration request has the quantity of data and its characteristics.
        iii. Each SoC looks at the storage utilization rate marked as light and medium. If the rate of utilization is < R_{TH} and if the number of tasks in the NF window of the queue is < N_{TH}, then the node sends data to accept after a delay δ. This δ depends upon the CPU load and available data space. As in the case of tasks, if the node is lightly loaded and has good storage space available δ will be smaller.
        iv. Data is sent to the cloud if no such node is available,

5. RESULTS AND DISCUSSION

A. Data Storage

In the previous section, we have described the algorithm for distributed uniform data storage on the cloudlets. We varied the lower and upper thresholds to analyze the algorithm's performance for differing data loads from the end devices. We tried to obtain the ratio of the data that would remain on the cloudlet that gets migrated to another cloudlet and transferred to the cloud. Figures 4 to 9 show the effect of data storage on the cloud.

We initially froze our lower threshold at 0.3 and varied the upper threshold from 0.5 (figure 4) to 0.8 (figure 7).

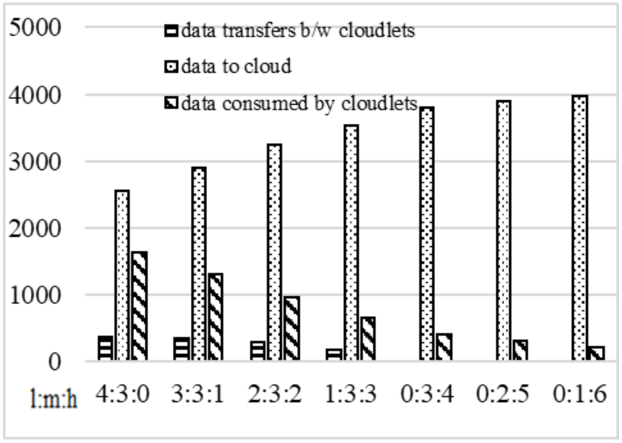


Figure 4. Data migration with LT at 0.3 and UT at 0.5

A threshold of 0.3 indicates that 30% of the storage space on the cloudlet node is already full. The upper threshold is then used to decide when the data should be moved from the cloudlet. The upper threshold of 0.5 means that when 50% of data space is full, the data is migrated either to another cloudlet or the cloud. For each threshold, we also vary the number of lightly loaded nodes, moderately loaded and heavily loaded. Figure 4 shows that as the number of lightly loaded nodes is reduced, the amount of data that gets moved between the cloudlets decreases and the amount of data transferred to the cloud increases. This is because we have kept a very low upper threshold of 50%. So when each node reaches 50% of its data size, it will offload data to another node in the cloudlet, which is more feasible if there are more lightly loaded nodes. But if the number of lightly loaded nodes is significantly less, the data will be automatically transferred to the cloud.

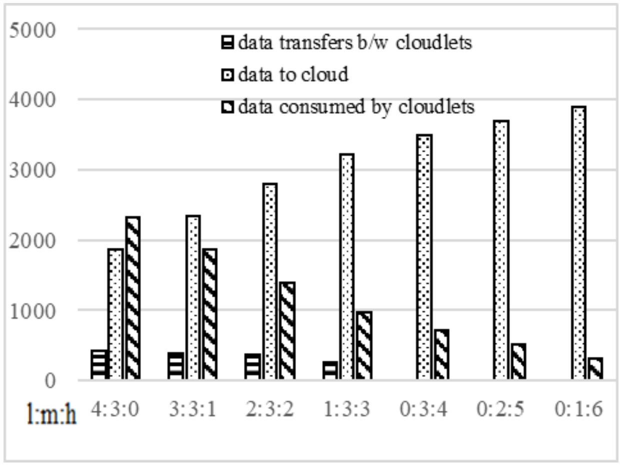


Figure 5. Data migration with LT at 0.3 and UT at 0.6

When we increase the upper threshold to 60% in Figure 5, comparatively more amount of data is transferred within the cloudlets. And the amount of data which is transferred to the cloud decreases. When there are no lightly loaded nodes, and the number of moderate nodes is also less except for the data that is consumed by the cloudlet node itself, the data gets migrated to the cloud rather than to another cloudlet node.

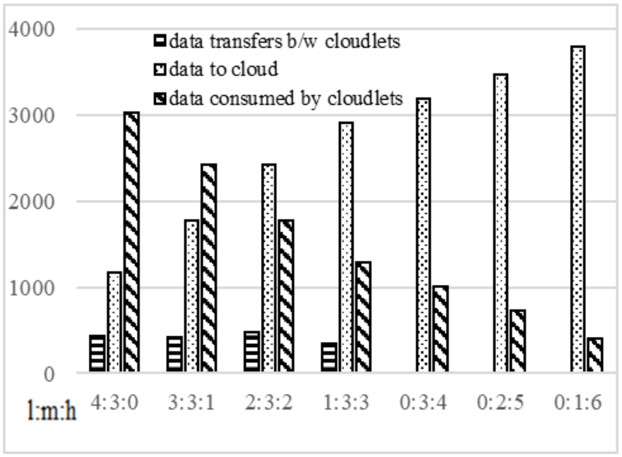


Figure 6. Data migration with LT at 0.3 and UT at 0.7

With the increase in the upper threshold, more data is stored on the individual cloudlet node, and less amount of data gets migrated. When we kept the low threshold constant and varied the upper threshold, an interesting observation that can be noted is that the data that is migrated between the cloudlets remains constant. The variation is only in the amount of data that is migrated to the cloud. This is because the algorithm is built so that it will try to store the data on individual cloudlet nodes connected to the end devices and only migrate the data when certain thresholds are reached. By this time, it’s quite possible that other nodes have reached the upper threshold. Hence, the data has to be migrated onto the cloud. And this can be observed in Figures 4-7.

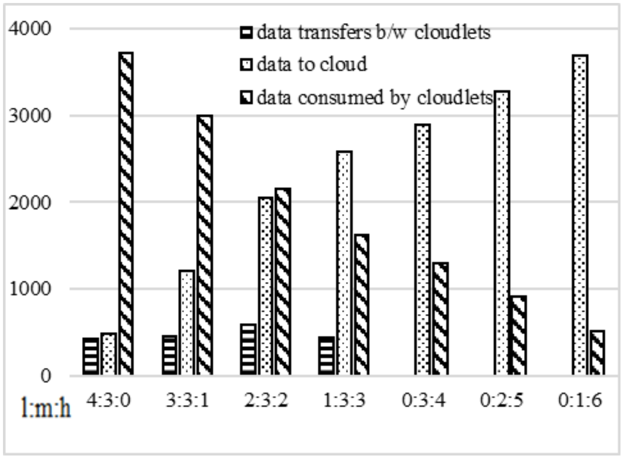


Figure 7. Data migration with LT at 0.3 and UT at 0.8

We then kept the upper threshold constant at 0.7 and varied the lower threshold from 0.2-0.4, meaning that the initial load on each node in the cloudlet was varied between 20%-40%.

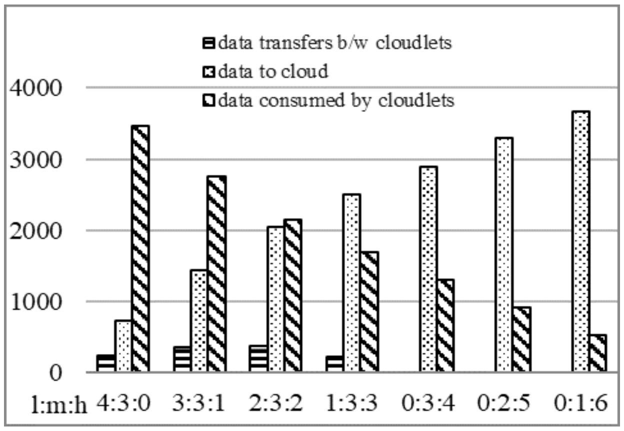


Figure 8. Data migration with LT at 0.2 and UT at 0.7

When the upper threshold is maintained constant and the lower threshold is increased from 0.2 to 0.4, more amount of data is migrated to the nodes within the cloudlets that are lightly loaded. This can be seen in Figure 6, Figure 8, and Figure 9.

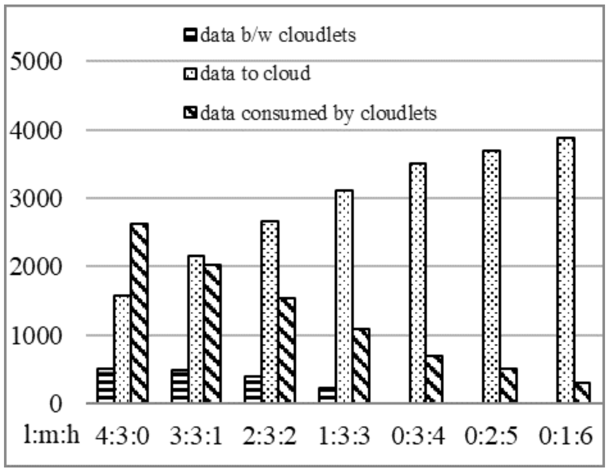


Figure 9. Data migration with LT at 0.4 and UT at 0.7

It's only when zero nodes are lightly loaded that the data gets migrated to the cloud.
In this case, there is an increase in the data migrated within the cloudlet as the lower threshold increases while the data that is migrated to the cloud remains constant.

B. Task Scheduling

In an IoT system, not only is the data stored in the cloud, but the tasks are also executed in the cloud. In our cloudlet system, similarly, it can act as a data storage platform as well as execute tasks. Regarding task scheduling, we kept our task thresholds constant at 0.3 and 0.7. The lower threshold of 0.3 indicates that 30% of Cloudlet's processing power is already in use, and the 0.7 upper threshold means that the CPU utilization can reach 70%. To analyze the effectiveness of task scheduling algorithms, we varied the weights of past, current and future windows (as described in section IV). When we assigned equal weightage to the past, current and future windows (33.33%), as seen in Figure 10.

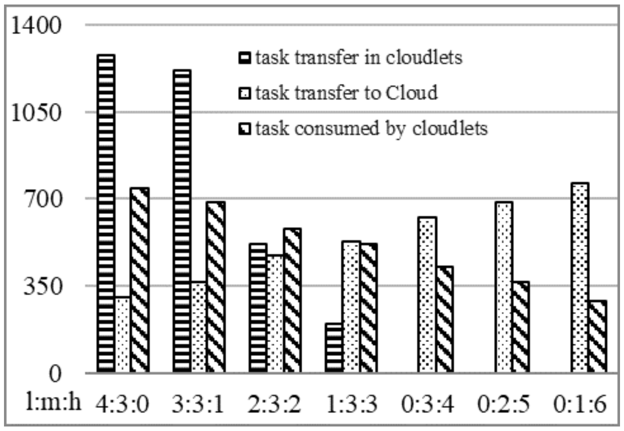


Figure 10. Task scheduling keeping past current and future windows weight 0.33:0.33:0.33

We found that when the number of nodes that were lightly loaded and medium loaded was more, a very less number of tasks were migrated onto the cloud. Only when there were no lightly loaded nodes, as per our algorithm, were the tasks migrated to the cloud.

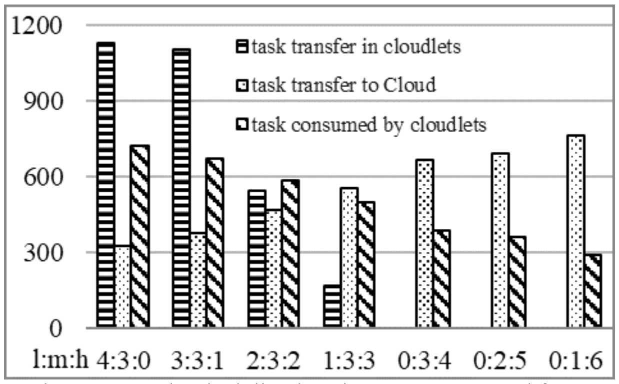


Figure 11. Task scheduling keeping past current and future windows weight 0.3:0.4:0.3

When we varied the weights by assigning the weight of 0.3 to past, 0.4 to current and 0.3 to future windows, we found that slightly more tasks were migrated to the cloud, as seen in Figure 11. The tasks consumed by the cloudlets remain relatively constant.

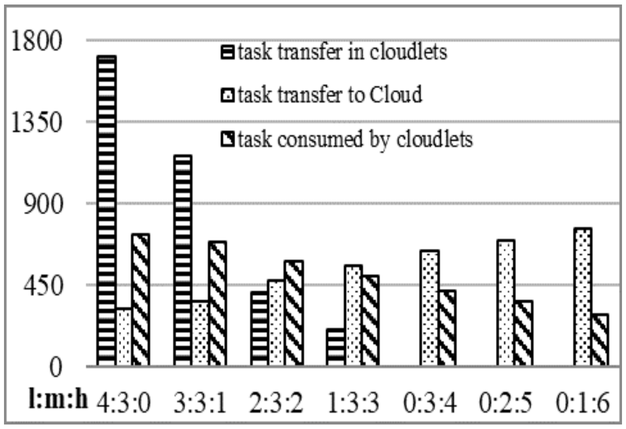


Figure 12. Task scheduling keeping past current and future windows weight 0.25:0.5:0.25

When the weightage of the past and the future windows was kept at 0.25, giving more weightage of 0.5 to the current window, more tasks were moved within the cloudlets, and the number of tasks migrated to the cloud remained relatively constant, as seen in Figure 12. This shows that at this weight value, the load is better balanced between node and cloudlets, and the CPU utilization remains fairly uniform.

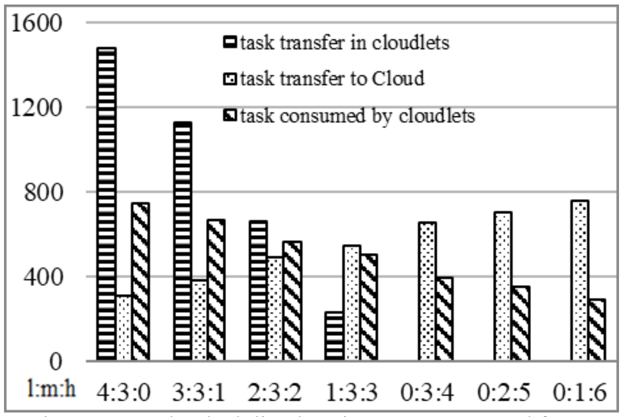


Figure 13. Task scheduling keeping past current and future windows weight 0.2:0.6:0.2

When we further increased the weightage of the current window to 0.6, 0.7 and 0.8, as shown in Figure 13, Figure 14 and Figure 15, respectively, the number of tasks that were transferred within the cloudlet increased up to 0.7 and when more weightage was assigned to the present window ignoring the past history and the future load the number of tasks that were migrated between the cloudlets decreased. Each cloudlet node tries to schedule the task by itself. The number of tasks migrated to the cloud was also much less unless there was no lightly or moderately loaded node.

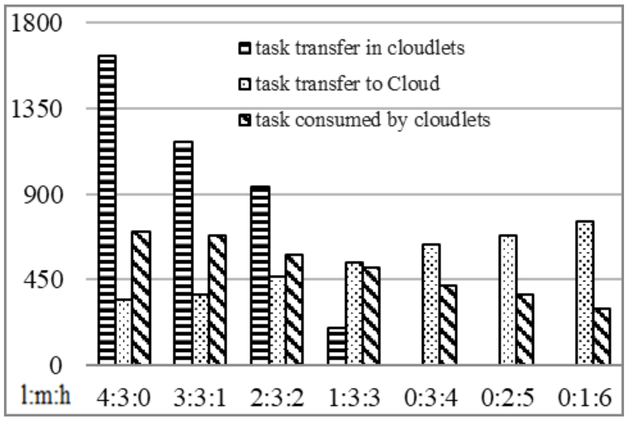


Figure 14. Task scheduling keeping past current and future windows weight 0.15:0.7:0.15

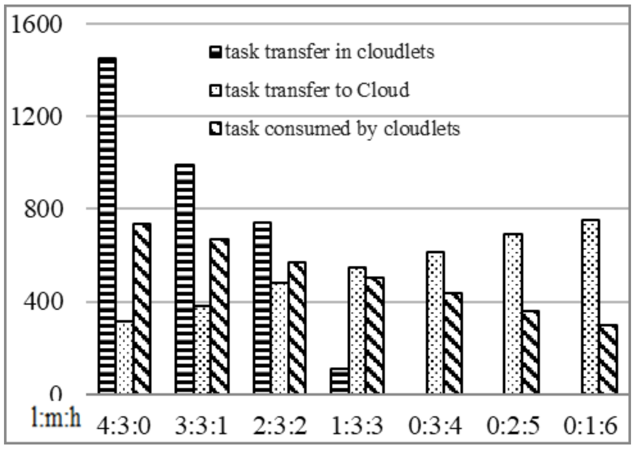


Figure 15. Task scheduling keeping past current and future windows weight 0.1:0.8:0.1

We can conclude from this that while we may assign a weightage between 0.5 to 0.6 to the current window, we should also be aware of the past scheduling history and be able to anticipate the future load on the cloudlet.

6. AN IMPLEMENTATION USING CLOUDLETS - DIGITAL TWINNING OF ROBOTIC ARM:

The term digital twin was coined in 2002. It is used to monitor and maintain remote equipment deployed in inaccessible terrains. Initially, Digital Twining was done to design, model and analysis; creating a 3-D digital virtual twin was used to figure out design glitches and simulate the complete system before manufacturing it.
Current digital twins are designed to build a virtual 3D model using large amounts of raw data from the actual system. Digital twins accurately simulate, analyze and even predict real-time events using real-time data from the physical world. To summarize, digital twins are virtual representations of the physical product. There can be a real-time connection between the physical world and the virtual model if data is relayed to it in real time, and the behavior of the digital twin varies with the data received from its digital counterparts.

[16] gives an overview of digital twin applications. Digital twins are used in different sectors. The sectors where digital twins are used are (a) manufacturing, (b) aerospace, (c) marine, (d) agriculture, (e) healthcare (f) mining. Digital Twins can be used for the following purposes: (a) simulation, (b) monitoring, and (c) control. Simulations are used to model the behavior of the system in virtual space. This allows optimization of the product or the production process. The monitoring includes current data so that the current state of the product or system can be interpreted. When used in control, Digital Twins can be used for predicting events and, based on the predictions, can control the behavior of the product or system.

Another way to classify the digital twin is based on the physical device that it is modelling. There are four categories of physical objects: (a) Manufacturing asset, (b) product, (c) infrastructure, and (d) Human. Manufacturing asset refers to an object from a tool like a CNC machine. Product – indicates a completed product. Infrastructure refers to bridges, cities, etc.; in the case of humans, it could refer to an employee in the industry who is being trained by a digital twin expert or an expert who is training the employees on hazardous systems.

The completeness of the digital twin has the number of features that are represented; for example, a digital twin of a room has parameters such as temperature, humidity, lighting, O_2 Levels, Occupancy, Power consumption and these parameters have to be continuously updated, so data is sent regularly, and the digital twin is updated real-time.
A digital twin can be created before or after the physical object is created. If created before to analyze behavior and then build the physical object, it is termed a prototype. If the twin is bound to the device later, it is called a Digital Twin instance. A digital twin instance of any machinery in the industry can be used for predicting events, even failures, as the real-time data is drawn from the machine into the digital twin.

The digital twin is one of the key technologies for Industry 4.0, together with AR/MR (Machine Reality) and IIoT, since they provide valuable tools for manufacturing, training, healthcare, and smart city environments. At the point of writing this paper, we have been unable to find any detailed developments that study and integrate these three technologies have been found in the literature.
Many of the use cases of Industry 4.0 combine AR/MR and IIoT. The following table summarizes the current work related to digital twinning.

Table 1: Summary of current research in Digital Twining

Ref No	**Application**	**Description**
[17]	Smart Shelf with QR codes	When scanned, the display is done on F4 smart glasses – of the strain on the shelf using a series of strain gauges. This is just a Proof of concept
[18]	Infrastructure	AR framework to visualize and detect any device using localization techniques
[19]	Microsoft HoloLens glasses	One M2M is tightly coupled with the Virtual Twin
[20]	Disaster Management	Visualize sensor data of the area affected using the HoloLens tool kit.
[21]	Training	Training and assistance of supervisors on various manufacturing processes
[22], [23]	Training	Real-time used to instruct shop floor operators.
[24],[25], [26]	Robot Navigation	It is used to give the robot better navigation information to assist it.
[27]	Documentation	Creating Manuals and 3D models of various documents
[28]	Real-Time Data Retrieval	It helps in decision-making for real-time use cases.
[29]	Maintenance	To detect the maintenance difficulties in the manufacturing industry
[30]	Simulation	Developing simulation base support applications by integrating contextual awareness using sensors
[31]	Device integration	Microsoft HoloLens glasses simplify AR and IoT device integration
[32]	Characterization of digital twin	Modelling logical items within a virtualized realm and mapping them to physical realities for the characterization of digital twin
[33]	3D manual creation	3D manual creation for training operators in manufacturing industries.
[34]	Rapid analysis and real-time decision	Digital twin used for rapid analysis and real-time decisions in the aerospace industry
[35]	Risk management for operators	Developed theoretical reference model using data collected from sensors, experts and historical data for risk management in various work environments to improve operator safety.
[36]	Industrial ice-cream machine	Uses real-time data collected from the sensors inside the machine for fault monitoring and performance assessment.
[37]	Robotic arm	Taking visualization inputs from AR devices and developing digital twins of robotic arms.

To create and maintain a digital twin in real time requires a large amount of memory and processing power.

Usually, cloud computing systems exist – but there are several issues that exist, such as:
(a) connectivity issues; data cannot be updated on the cloud in real-time because of latencies involved in geographically distributed clouds; hence, real-time events cannot be predicted and corrected
(b) Privacy, as most industrial data is proprietary; hence, security is a major issue. So, this is where cloudlets can offer a solution. Integrating cloudlets with digital twining in IIoT is a novel concept into which research is still to be done.
Real-Time Digital Twining will require Real-Time allocation of tasks and storage on the cloudlets in a distributed manner. Hence, our first step was to develop an algorithm for load balancing and task allotment.
Robotic arms are commonly used in manufacturing, from picking up parts and placing them on the conveyor belt to moving objects around.
So, we did a digital twin of the position of the robotic arm using a set of inertial measurement unit (IMU) sensors, a 3-axis accelerometer, a 3-axis gyroscope and a 3-axis magnetometer. We have only done one-half of the robotic arm to understand the processing capability of the cloudlet. The hand was put through various moments, and we look for its replication. The moment of the entire robotic hand manipulator will include a shoulder manipulator, elbow moment manipulator and wrist manipulator. Right now, we are only twining the manipulator of the elbow. The block diagram of the system twined using sensors connected to an end device is shown in Figure 16 below:

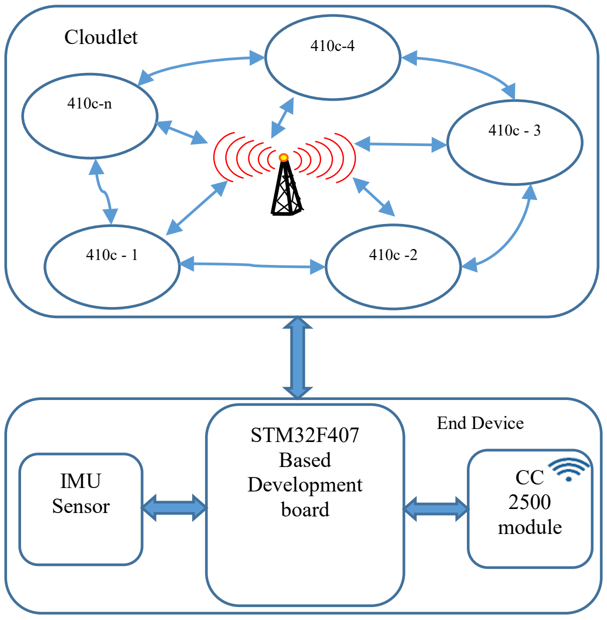


Figure – 16 Interface of the End-Device to the cloudlet system composed of multiple Qualcomm Snapdragon 410c.

We have used MPU6500 [38], Which has a three-axis accelerometer and a three-axis gyroscope, and GY271 [39], which is a magnetometer. Using these three sensors, it is possible to track the position of the arm. We were able to produce an exact 3D replica of the rotational position while, at the same time, displaying the values of the sensors.

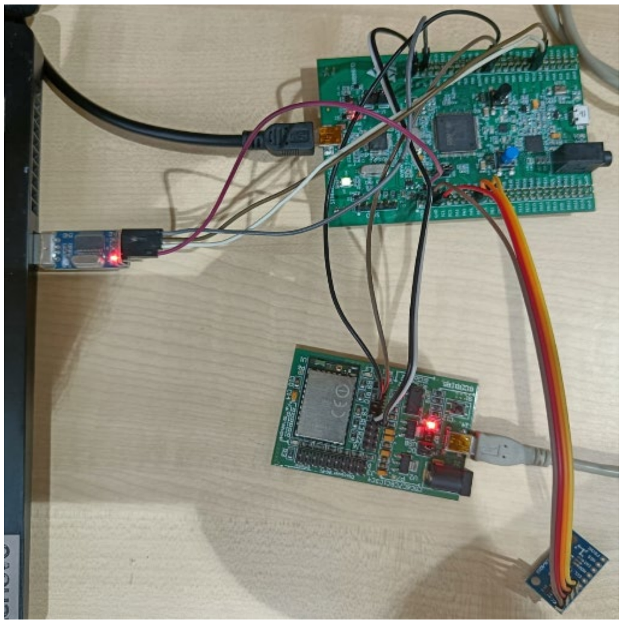
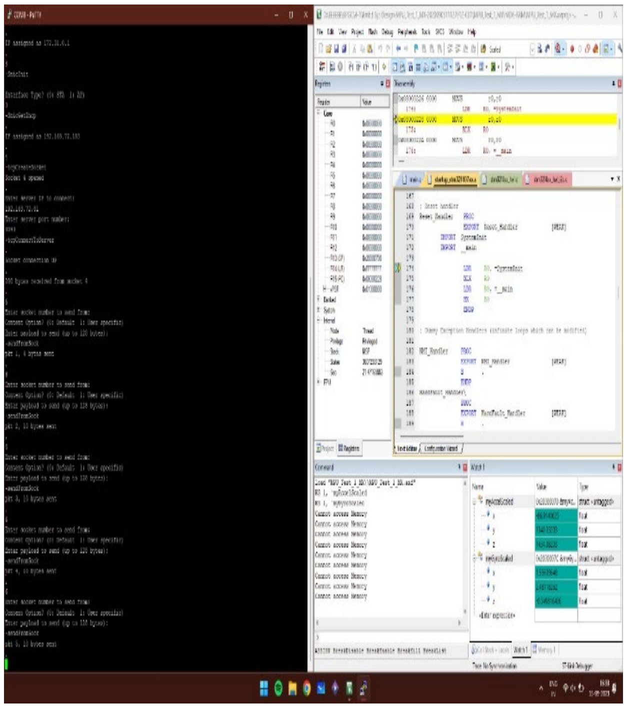


Figure 17 – Interfacing the End – device interfaced with the sensors and CC 2500.

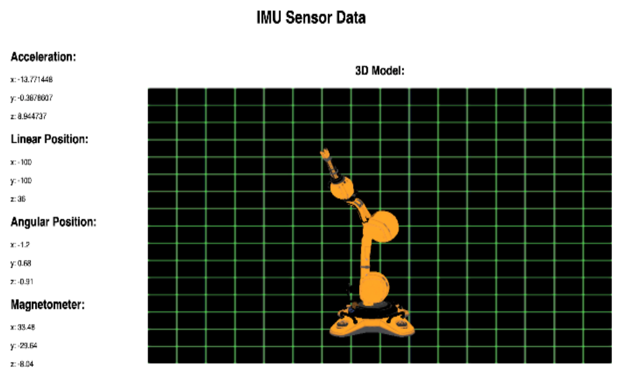


```
IP assigned as 172.31.0.1

-SnicInit
Interface Type? (0: STA  1: AP)

-SnicGetDhcp

IP assigned as 192.168.72.183

-tcpCreateSocket
Socket 4 opened

Enter server IP to connect:
192.168.72.81
Enter server port number:
8080
-tcpConnectToServer

Socket connection UP

100 bytes received from socket 4

Enter socket number to send from:
Content Option? (0: Default  1: User specific)
Enter payload to send (up to 128 bytes):
-sendFromSock
pkt 1, 4 bytes sent

Enter socket number to send from:
Content Option? (0: Default  1: User specific)
Enter payload to send (up to 128 bytes):
-sendFromSock
pkt 2, 10 bytes sent

Enter socket number to send from:
Content Option? (0: Default  1: User specific)
Enter payload to send (up to 128 bytes):
-sendFromSock
pkt 3, 10 bytes sent

Enter socket number to send from:
Content Option? (0: Default  1: User specific)
Enter payload to send (up to 128 bytes):
-sendFromSock
pkt 4, 10 bytes sent

Enter socket number to send from:
Content Option? (0: Default  1: User specific)
Enter payload to send (up to 128 bytes):
-sendFromSock
pkt 5, 10 bytes sent
```

Figure - 18 code and the data relayed to the SoC via CC2500.

The sensors were interfaced with the STM-32F407-based microcontroller development board [40], which was part of the end device. The actual end device is shown in Figure 17. The data was sent to the 410c SoC, to which the end device was interfaced via a CC 2500 module [41]. The data was sampled at a rate of 20 Hz and was continuously sent to the 410c SoC with which it was interfaced on the cloudlet. Our distributed data storage algorithm ensured that the data was uniformly distributed

among the various SoCs interfaced on the cloudlet. Once the data was received on the cloudlet, the tasks that were required to model the device and produce a 3D virtual image were invoked with every set of data.
Some of the sample figures of the arm position, along with the IMU Sensor data, are shown in figures 19a. to 19f.


IMU Sensor Data

Acceleration:
x: -13.771448
y: -0.3878607
z: 8.944737

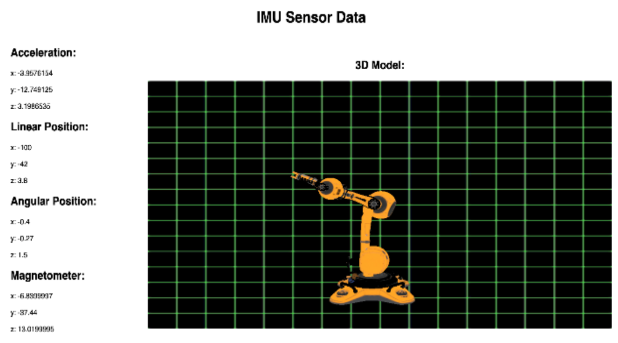
Linear Position:
x: -100
y: -100
z: 36

Angular Position:
x: -1.2
y: 0.68
z: -0.91

Magnetometer:
x: 33.48
y: -29.64
z: -8.04

3D Model:

Figure 19a – Arm rotated in an upward position


IMU Sensor Data

Acceleration:
x: -3.9576154
y: -12.749125
z: 3.1986535

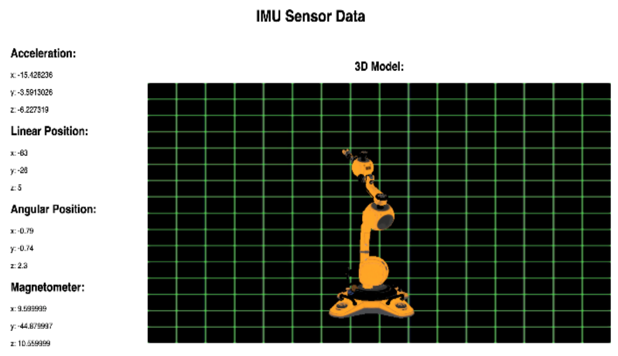
Linear Position:
x: -100
y: -42
z: 3.8

Angular Position:
x: -0.4
y: -0.27
z: 1.5

Magnetometer:
x: -6.8399997
y: -37.44
z: 13.0199995

3D Model:

Figure 19b – Arm in a straight position


IMU Sensor Data

Acceleration:
x: -15.426236
y: -3.5913026
z: -6.227319

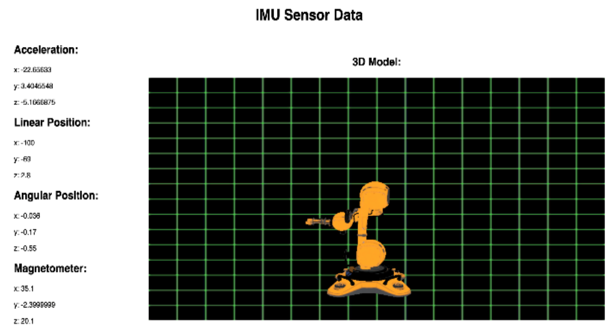
Linear Position:
x: -63
y: -26
z: 5

Angular Position:
x: -0.79
y: -0.74
z: 2.3

Magnetometer:
x: 9.599999
y: -44.879997
z: 10.559999

3D Model:

Figure 19c – Arm rotated in reverse upward position


IMU Sensor Data

Acceleration:
x: -22.65633
y: 3.4046548
z: -5.1066875

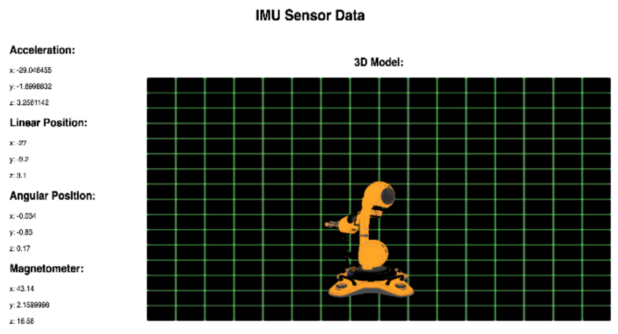
Linear Position:
x: -100
y: -69
z: 2.8

Angular Position:
x: -0.036
y: -0.17
z: -0.55

Magnetometer:
x: 35.1
y: -2.3999999
z: 20.1

3D Model:

Figure 19d – Arm rotated in a forward position


IMU Sensor Data

Acceleration:
x: -29.046455
y: -1.8998832
z: 3.2581142

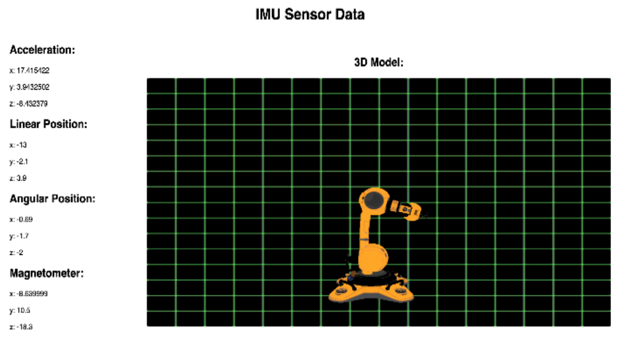
Linear Position:
x: -27
y: -9.2
z: 3.1

Angular Position:
x: -0.034
y: -0.83
z: 0.17

Magnetometer:
x: 43.14
y: 2.1599998
z: 16.56

3D Model:

Figure 19e – Arm rotated in a downward position


IMU Sensor Data

Acceleration:
x: 17.415422
y: 3.9432502
z: -6.432379

Linear Position:
x: -13
y: -2.1
z: 3.9

Angular Position:
x: -0.69
y: -1.7
z: -2

Magnetometer:
x: -8.639999
y: 10.5
z: -18.3

3D Model:

Figure 19f – Arm rotated in a backward position

7. CONCLUSION

This paper presents a novel cloudlet architecture and algorithms used for data and task management on cloudlets. Our cloudlet system was built using Qualcomm Snapdragon 410c. In the future, we plan to make this a hybrid approach involving the 820c and the 833 versions of Snapdragon. This would allow different types of tasks to be scheduled with varying architectures. This algorithm would form the basis for data storage and task scheduling. One of the anticipated applications of cloudlets in an industrial network is in the case of digital twinning. The tasks in digital twinning may vary from simple sensing to complex graphics processing. We have done digital twining of a part of the robotic arm in various positions using just four Qualcomm Snapdragon 410c in a distributed cloudlet architecture. A hybrid architecture would allow different types of tasks to be scheduled on different architectures. The scheduling algorithm can be very easily extended to include task affinity. The migration rules will not only be threshold-based but also depend on the nature of the task. Our task scheduling algorithm is generic enough to accommodate different varieties of tasks.

8. REFERENCES

[1] A. Yousefpour et al., "All one needs to know about fog computing and related edge computing paradigms: A complete survey," Journal of Systems Architecture, 2019. [Online]. Available:http://www.sciencedirect.com/science/article/pii/S1383762118306349

[2] Ericsson. (2016, January). Cellular networks for massive IoT. [Online]. Available:http://www.ericsson.com/assets/local/publications/white-papers/wp_iot.pdf.

[3] A. Botta, W. De Donato, V. Persico, and A. Pescape, "Integration of Cloud computing and Internet of Things: A Servey," Futur. Gener. Comput. Syst., vol. 56, 2016, pp. 684-700.

[4] P. H. Raj, P. R. Kumar, and P. Jelciana, ``Mobile cloud computing: A survey on challenges and issues," Int. J. Comput. Sci. Inf. Secur., vol. 14, no. 12, p. 165, 2016.

[5] E.Sisinni, A. Saifullah, S. Hong, M. Gidlund, "Industrial Internet of Things: Challenges, Opportunities and Directions", IEEE Transactions on Industrial Informatics, vol. X, no. X, 2018, pp. 1-23.

[6] Verma, M.; Bhardwaj, N.; Yadav, A.K. Real time efficient scheduling algorithm for load balancing in fog computing environment. Int. J. Inf. Technol. Comput. Sci 2016, 8, 1–10.

[7] Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are. White Paper. 2016. Available online: https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf.

[8] Ketel, m. Fog-cloud services for IoT. In Proceedings of the SouthEast Conference, Kennesaw, Ga, USA, 13-15 April 2017; pp. 262-264.

[9] T. Verbelen, P. Simoens, F. De Turck, and B. Dhoedt, "Cloudlets: Bringing the cloud to the mobile user," in Proceedings of the third ACM workshop on Mobile cloud computing and services. ACM, 2012, pp. 29–36.

[10] L. Tamilselvan, ''Client aware scalable cloudlet to augment edge computing with mobile cloud migration service,'' Int. J. Interact. Mobile Technol., vol. 14, no. 12, p. 165, Jul. 2020.

[11] Obst, M.; Holm, T.; Urbas, L.; Fay, A.; Kreft, S.; Hempen, U.; Albers, T. Semantic description of process modules. In Proceedings of the 2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA), Luxembourg, 8–11 September 2015; pp. 1–8.

[12] V. Dastjerdi, H. Gupta, R.N. Calheiros, S.K. Ghosh, R. Buyya, F og Computing: principles, architectures, and applications, in R. Buyya, A.V. Dastjerdi (Eds.), Internet of Things Principles and Paradigms, Elsevier, USA, 2016 ISBN: 978-0-12-805395-9, Chapter 4.

[13] M. Satyanarayanan, Z. Chen, K. Ha, W. Hu, W. Richter, and P. Pillai, "Cloudlets: at the leading edge of mobile-cloud convergence," in Mobile Computing, Applications and Services (MobiCASE), 2014 6th International Conference on. IEEE, 2014, pp. 1–9.

[14] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, ``The case for VM-based cloudlets in mobile computing," IEEE Pervas. Comput., vol. 8, no. 4, pp. 14_23, Oct. 2009.

[15] Mohammad Babar; Muhammad Sohail Khan; Farman Ali; Muhammad Imran; Muhammad Shoaib "Cloudlet Computing: Recent Advances, Taxonomy, and Challenges" IEEE Access Vol:9 pp: 29609 - 29622 Feb 2021.

[16] Enders, Martin & Hoßbach, Nadja. (2019). Dimensions of Digital Twin Applications - A Literature Review.

[17] Revetria, R., Tonelli, F., Damiani, L., Demartini, M., Bisio, F., Peruzzo, N.: A real-time mechanical structures monitoring system based on digital twin, Iot and augmented reality. In: Proceedings of the 2019 Spring Simulation Conference (SpringSim), Tucson, pp. 1–10 (2019)

[18] Jo, D., Kim, G.J.: AR IoT: scalable augmented reality framework for interacting with Internet of Things appliances everywhere. IEEE Trans. Consum. Electron. 62(3), 334–340 (2016)

[19] Lee, S., Lee, G., Choi, G., Roh, B., Kang, J.: Integration of OneM2M-based IoT service platform and mixed reality device. In: Proceedings of the 2019 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, pp. 1–4 (2019)

[20] Chusetthagarn, D., Visoottiviseth, V., Haga, J.: A prototype of collaborative augment reality environment for HoloLens. In: Proceedings of the 2018 22nd International Computer Science and Engineering Conference (ICSEC), Thailand, pp. 1–5 (2018)

[21] Alesky, M., Vartiainen, E., Domova, V., Naedele, M.: Augmented reality for improved service delivery. In: Proceedings of the IEEE 28th International Conference on Advanced Information Networking and Applications, Canada, pp. 382–389. IEEE (2014)

[22] Hoˇrejˇs´ı, P.: Augmented reality system for virtual training of parts assembly. In: Proceedings of the 25th DAAAM International Symposium on Intelligent Manufacturing and Automation (DAAAM), Vienna, pp. 699–706. Procedia Engineering (2015)

[23] Tang, A., Owen, C., Biocca, F., Mou, W.: Comparative effectiveness of augmented reality in object assembly. In: 2003 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI), Florida, pp. 73–80 (2003)

[24] Mosiello, G., Kiselev, A., Loutfi, A.: Using augmented reality to improve usability of the user interface for driving a telepresence robot. Paladyn J. Behav. Robot. 4(3), 174–181 (2013)

[25] Herrera, K.A., Rocha, J.A., Silva, F.M., Andaluz, V.H.: Training systems for control of mobile manipulator robots in augmented reality. In: Proceedings of the 2020 15th Iberian Conference on Information Systems and Technologies (CISTI), Sevilla, pp. 1–7. IEEE (2020)

[26] Lapointe, J.-F., Molyneaux, H., Allili, M.S.: A literature review of AR-based remote guidance tasks with user studies. In: Chen, J.Y.C., Fragomeni, G. (eds.) HCII 2020. LNCS, vol. 12191, pp. 111–120. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-49698-2 8

[27] Zollmann, S., Hoppe, C., Kluckner, S., Poglitsch, C., Bischof, H., Reitmayr, G.: Augmented reality for construction site monitoring and documentation. Proc. IEEE 102(2), 137–154 (2014)

[28] Moloney, J.: Augmented reality visualisation of the built environment to support design decision making. In: Proceedings of the 10th International Conference on Information Visualisation, IV 2006, London, pp. 687–692. IEEE (2006)

[29] Erkoyuncu, J., Khan, S.: Olfactory-based augmented reality support for industrial maintenance. IEEE Access 8, 30306–30321 (2020)

[30] Lampen, E., Lehwald, J., Pfeiffer, T.: A context-aware assistance framework for implicit interaction with an augmented human. In: Chen, J.Y.C., Fragomeni, G. (eds.) HCII 2020. LNCS, vol. 12191, pp. 91–110. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-49698-2 7

[31] Blanco-Novoa, ´ O., Fraga-Lamas, P., Vilar-Montesinos, M.A., Fernandez-Carames, T.M.: Creating the internet of augmented things: an open-source framework to make IoT devices and augmented and mixed reality systems talk to each other. Sensors 20(11), 3328 (2020)

[32] Minerva, R., Lee, G.M., Crespi, N.: Digital twin in the IoT context: a survey on technical features, scenarios, and architectural models. IEEE 108(10), 1785–1824 (2020)

[33] Hoˇrejˇs´ı, P., Novikov, K., ˇSimon, M.: A smart factory in a Smart City: virtual and augmented reality in a smart assembly line. IEEE Access 8, 94330–94340 (2020)

[34] Shafto, M., Conroy, M., Doyle, R., Glaessgen, E., Kemp, C., LeMoigne, J., et al.: DRAFT modeling, simulation, information technology & processing roadmap. Technology Area 11, NASA - National Aeronautics and Space Administration (2010)

[35] Bevilacqua, M., et al.: Digital twin reference model development to prevent operators' risk in process plants. Sustainability 12(3), 1088 (2020)

[36] Karadeniz, A.M., Arif, ˙I., Kanak, A., Erg¨un, S.: Digital twin of eGastronomic things: a case study for ice cream machines. In: 2019 IEEE International Symposium on Circuits and Systems (ISCAS), Sapporo, Japan, pp. 1–4. IEEE (2019)

[37] Aschenbrenner, D., et al.: Mirrorlabs - creating accessible Digital Twins of robotic production environment with Mixed Reality. In: IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR), pp. 43–48 (2020)

[38] MPU-6500 Product Specification Revision 1.3 2018, TDK InvenSense [Accessed 10-Sep-2023] https://invensense.tdk.com/download-pdf/mpu-6500-datasheet/

[39] GY271 Datasheet, Handson Technology, [Accessed 10-Sep-2022]https://handsontec.com/dataspecs/sensor/GY271%20HMC5883L.pdf

[40] STM32F4DISCOVERY Databrief , ST Microelectronics [Accessed 10-Sep-2023] https:// www.st.com /en /evaluationtools /stm32f4discovery.html

[41] CC2500 Low-Cost Low-Power2.4GHz RF Transceiver datasheet , Texas Instruments [Accessed 10-Sep-2023] https://www.ti.com /lit/ds/symlink/cc2500.pdf/