Chapter 1

System Overview

1.1 Concept

1.1.1 Monitoring

The solar panel generates electricity and feed it into the converter after being measured by the current and voltage sensor. The converter converts the electricity from a higher voltage to a desired votlage, which can be measured again by the sensor and used to charge the battery. The sensing data consist of input current and voltage, and output current and voltage, which are fed into the microcontroller. The microcontroller has a 4G cellular modem which enables communication with the surrounding 4G cellular base station, and the internet access is provided by the gateway at the base station. That means, the sensing data can be be sent from the microcontroller to the server through the 4G cellular network. Once the sensing data reaches the server, the data is processed, store into the database, and send to the client such as a web application. The concept for monitoring is shown in the figure 1.1.

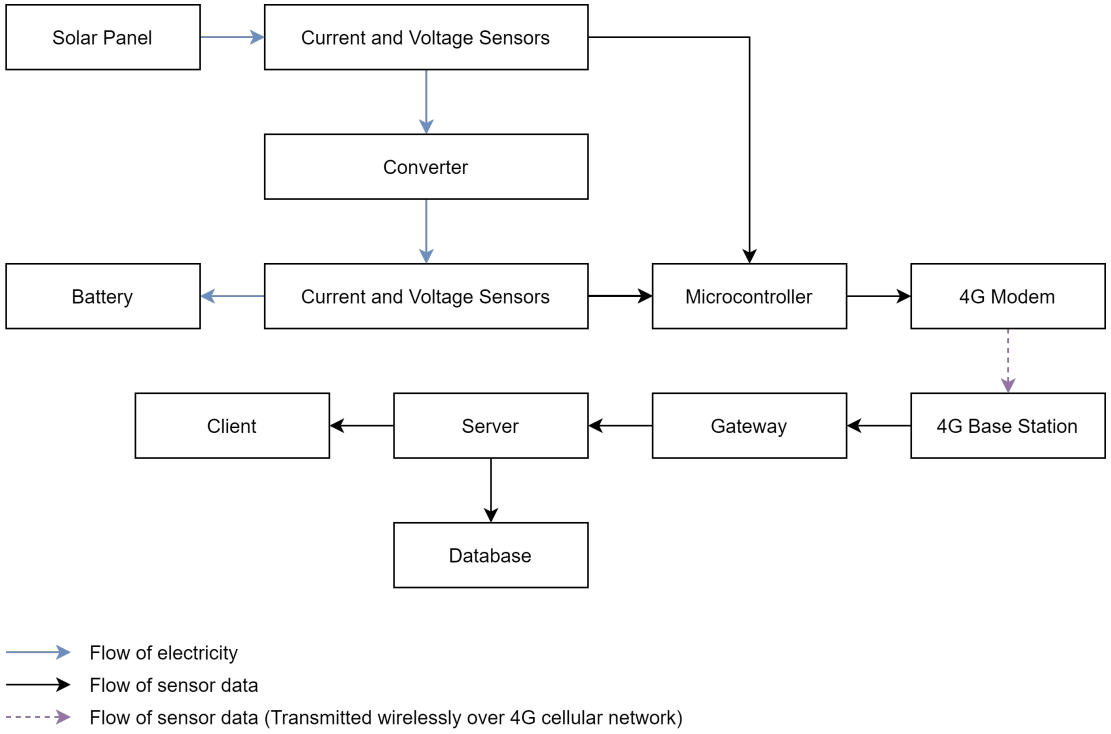

Figure 1.1: Concept of the proposed system for monitoring.

1.1.2 Controlling

The lifecycle of a command begins at the client initiated by the user. The command can be anything from setting the output voltage, to shutting down the converter. Once initiated, the client contact the server to let it know a command had been issued, and the server store it in the database.

On the other hand, the microcontroller performs periodic polling, that is, querying the server if there are new commands issued by the user every second. Similar to the monitoring, the query that is sent from the microcontroller to the server and the reply of the server that is consisting of a list of new commands are transmitted over the 4G cellular network. Finally, the commands are received by the microcontroller and the changes are applied to the converter. The concept for controlling is shown in the figure 1.2.

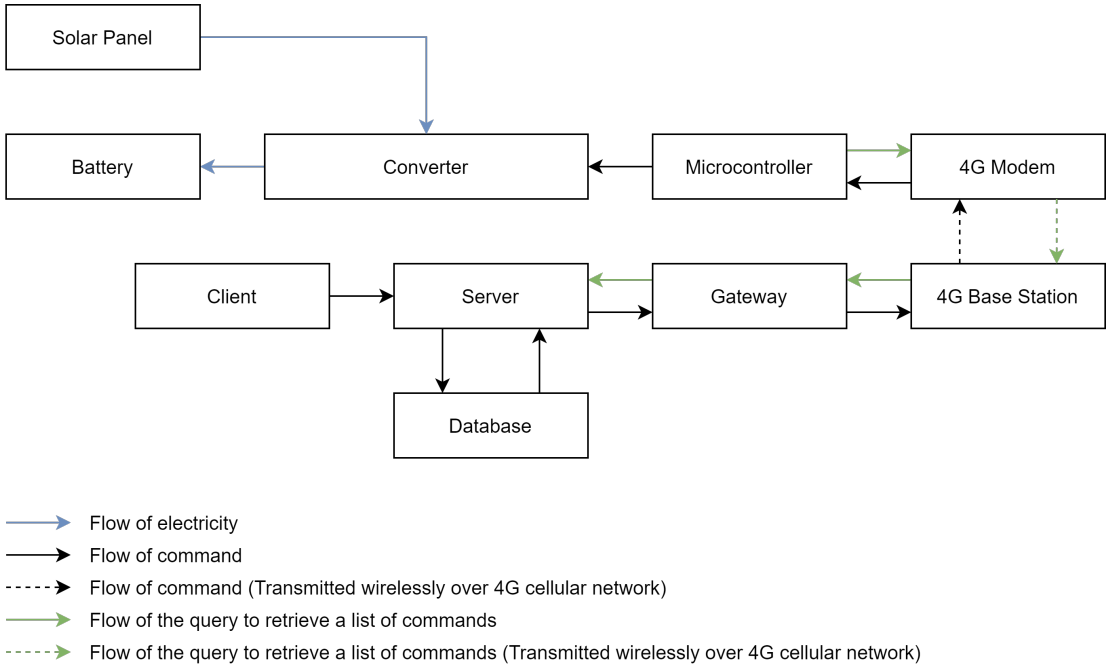

Figure 1.2: Concept of the proposed system for controlling.

1.2 Composition

The proposed system is composed of five layers separated by responsibilities. They are:

- Storage Layer
- Server Layer
- Relay Layer
- Simulation Layer
- Sensing Layer

The storage layer is responsible for storing the state of the server and sensing data received from the microcontrollers. The server layer is where the data processing logic, system monitoring, and data visualisation services are located. The relay layer contains the 4G infrastructures and providing internet access to 4G enabled devices. The sensing layer is responsible for measuring the data and control the generation of power. Lastly, the simulation layer is used to simulate the behaviours of the sensing layer for benchmarking purposes. Figure 1.3 shows the layers and the components within each layer.

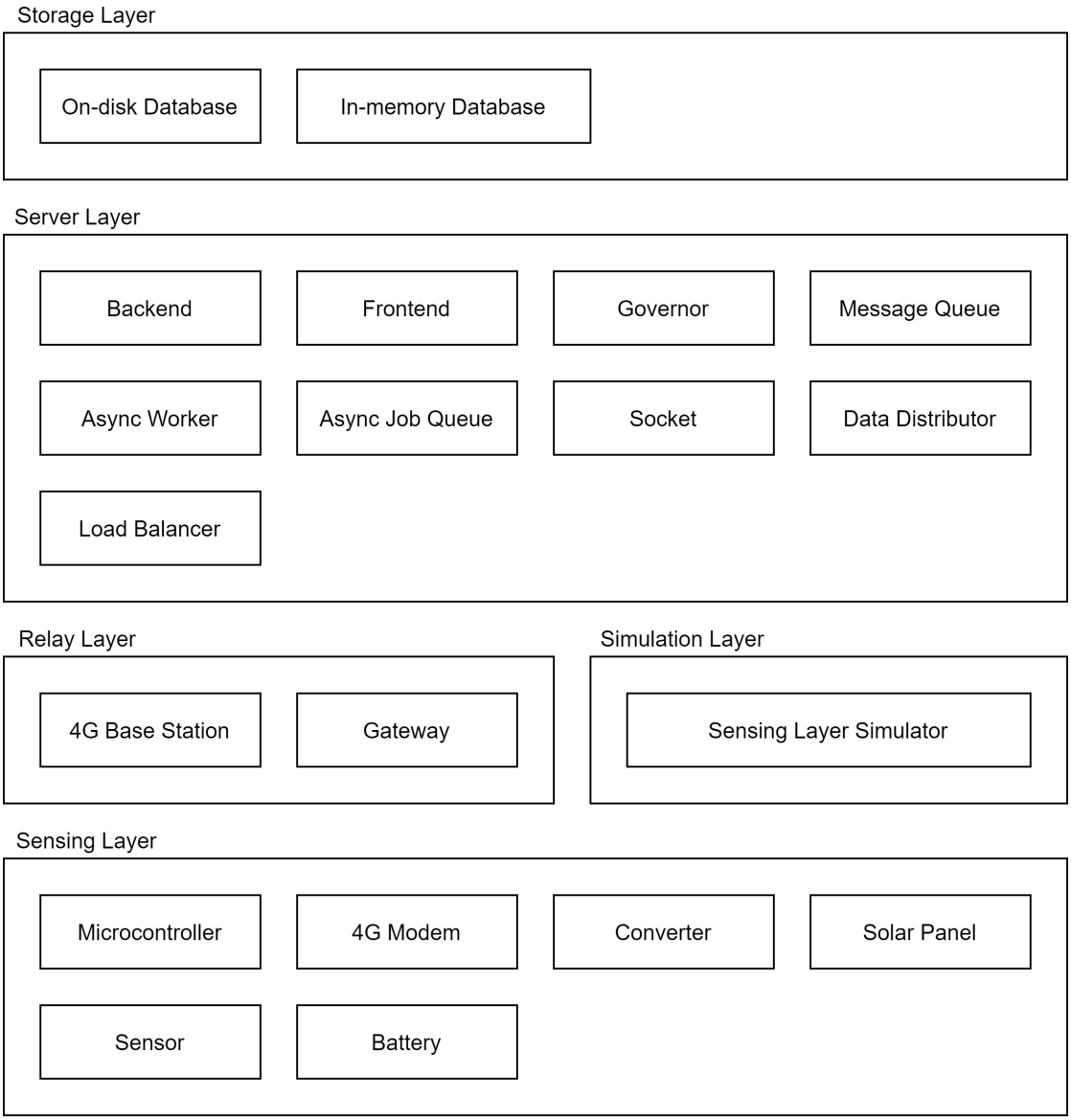

Figure 1.3: Layers and components of the proposed system.

1.2.1 Storage layer

On-disk database

The distinguishing characteristic of this type of database is the data that are stored in the database is persistant, which means the data remain intact even if the database is down due to failures. It is usually used to store a large amount of data reliably. The downside of using an on-disk database is the response time

of querying and inserting data into the database can be very slow. The primary usage in the proposed system is storing the information, status, and sensing data of a device such as a solar panel persistently.

In-memory database

The goal for this type of database is not storing the data persistently but aiming to access, modify, and insert data as quick as possible. This advantage is the result of using the memory instead of the disk of the server, which is significantly faster. However, it also comes with the problem of having significantly smaller capacity, and the data is volatile. That is, the data is gone when the database is down and the data may be purged if the memory of the server is running low. This is primarily used to synchronise the state and sharing data between backend instances that are running on the same server.

1.2.2 Server layer

Backend

Backend is a stateless HTTP server and its responsibilities are:

- Providing application programming interface (API) for microcontrollers to register itself, insert sensing data to the database, and retrieve issued commands.
- Providing API for frontend to retrieve data for visualisation and notifying a command had been issued.
- Providing context-aware load balancing for frontend's real-time data visualisation.

Frontend

Frontend is a web application and its responsibilities are:

- Providing a pleasant and usable user interface.
- Providing a real-time data visualisation.
- Providing a control panel for issueing commands.

Load balancer

It is a piece of software that distribute the incoming requests to the backend instance with the least amount of load. It also uses multiple threads to serve the frontend web pages to improve the performance under load.

Governor

The governors are daemons for house keeping. That is, they are processes that are running in the background of the server and performing some predefined tasks periodically. There are only two governors in the proposed system. The first governor is called data aggregation governor and its job is aggregating the real-time data every minute. The second governor is called liveness monitoring governor and it checks if a device is publishing sensing data normally or timed out unexpectely.

Message queue

This is an asynchronous communication system between the components of the server layer. The main contribution of the message queue is enabling processes such as backend, governor, and data distributor to communciate with each other despite they are completly separate programs.

The message queue is asynchronous because the originating process can send a message and continue to perform other tasks without waiting for the message to arrive or a reply is received. As oppose to synchronous communication, where the originating process would stop doing any work and waiting for a reply to receive before continuing.

Async job queue and worker

The asynchronous job queue is a queue of tasks to be executed in a first in first out fashion, and the worker is a thread that takes a job from the queue in order and actually execute them. The backend is using this mechanism to process lengthy but not time sensitive tasks such as inserting data into the database to improve performance. Similar to the message queue, the originating process adds a job to the queue and it continues to perform other work without waiting for the work to finish.

Socket

Socket is the key to efficient real-time communication between the backend and the frontend. This is the enabling tehcnology which allows the frontend to wait

for new data to be available as opposed to letting the frontend asking for new data every second, which is very inefficient.

Data distributor

The data distributor sits between the frontend and the backend during the real-time communication. Its purpose is reducing the load of the server by lazy loading the new sensing data at most one time, and publish the data to the frontend instances directly.

Without the data distributor, the frontend instances would be notified a new data is available, and all frontend instances would query the backend for the same piece of data, causing the backend to be accessed directly proportional to the number of frontend instances. By using the data distributor, it allows the total backend access for each update to be at most one time regardless of the number of frontend instances.

1.2.3 Relay layer

This layer contains two key infrastructures, the 4G base station, and the internet gateway. The 4G base station allows 4G enabled devices to communicate with it and forward the data to its internet gateway so that devices can access the internet wirelessly.

1.2.4 Sensing layer

This layer contains power generation devices such as solar panels, power converters, and batteries in our experimental setup. There are also monitoring devices such as sensors that are attached to the input and output of the converter and sends analog signals to the microntroller. Then, the microcontroller process those signals and send the data to the server over the 4G cellular network using the 4G modem.

1.2.5 Simulation layer

This layer is used for benchmarking as it can generate any number of virtual solar panels and push simulated data to the server. This enables the performance and characteristics of the system to be measured and this layer should behave identical to the sensing layer from the server layer's point of view.

1.3 Hardware

The table 1.5 outlines the hardware that had been used in the system. Details of each hardware can be found in the subsections below.

Table 1.1: List of hardware in the proposed system.

Type	**Hardware Model**
Benchmarking server	Custom Built Desktop (See subsection 1.3.1)
Deployment server	AWS EC2 T2-Medium
Microcontroller	Nuvoton M263A
4G Modem	Quectel EC25
4G Antenna	Antenova SRFL029
4G Base Station	Depending on the service provider
Gateway	Depending on the service provider
Converter	Morningstar PS-MPPT-40M
Sensor	ACS723 and ACS724
Battery	Sun Xtender PVX-890T

1.3.1 Benchmarking server

This is the server that is being used to benchmark the proposed communication system. The hardware specification is detailed in the table 1.2.

Table 1.2: Hardware specification of the benchmarking server.

Name	**Specification**
CPU	AMD Ryzen 3800X (8 Core 16 Threads at 4.3 Ghz)
GPU	Nvidia GTX 1080Ti
Memory	16GB 3200Mhz
Disk	4TB HDD + 512GB SSD

1.3.2 Deployment server

This is the server that is hosted on the AWS cloud computing platform and it is used to deploy the proposed system so that it can be accessed publically. The

hardware specification is detailed in the table 1.3. Note that it is not neccessary to deploy the proposed system to a powerful cloud server as there is only one solar panel used for real-world testing.

Table 1.3: Hardware specification of the deployment server.

Name	**Specification**
Instance Type	T2.Medium
CPU	2 Cores
Memory	4GB
Disk	32GB

1.3.3 Microcontroller

The selected microcontroller is Nuvoton's M263A. The critical features for this microcontroller that are relevant to the proposed system is shown in the table **??**.

Table 1.4: Critical features for Nuvoton M263A

Criteria	**Specification**
CPU	ARM Cortex M23 64MHz
APROM Memory	512KB
SRAM Memory	96KB
LDROM Memory	4KB
Has PWM Pins	True
Has Analog Pins	True
Has SIM card slot	True
Has 4G modem support	True

See figure 1.4 for more details.

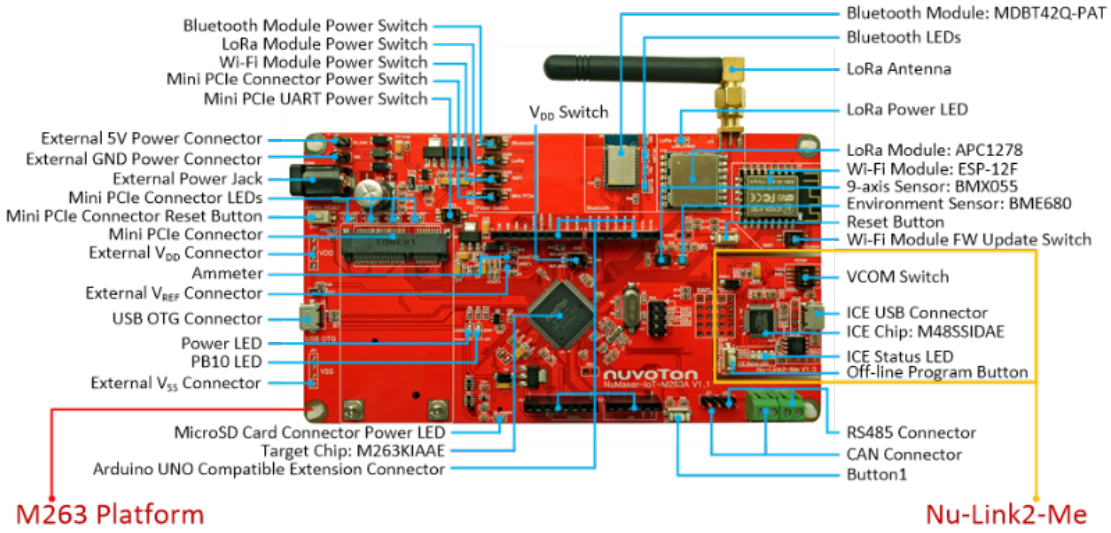

Figure 1.4: Nuvoton M263A [?]

Table 1.5: List of software in the proposed system.

Type	Hardware Model
Benchmarking operating system	Ubuntu Desktop 20.04 LTS
Deployment operating system	Ubuntu Server 20.04 LTS
Microcontroller platform	Mbed OS 15.5
Message queue	RedisMQ
Job queue and worker	RedisRQ
Frontend	ReactJS and nivo.rock
Backend	Python Flask
Socket	socket.io
Load Balancer	nginx
WSGI Server	Gunicorn
On-disk Database	MongoDB
In-memory Database	Redis
WSGI Server	Gunicorn
Programming language	Python 3.6

1.3.4 4G modem

1.3.5 4G base station and gateway

1.3.6 Converter

1.3.7 Voltage sensor

1.3.8 Current sensor

1.3.9 Battery

1.4 Software

1.4.1 Benchmarking OS

1.4.2 Deployment OS

1.4.3 Microcontroller platform

1.4.4 Message queue

1.4.5 Job queue and worker

1.4.6 Frontend

1.4.7 Backend

1.4.8 Socket

1.4.9 Load Balancer

1.5 Cost estimate

asdfsadfasdfasdfasdf