

## GUI Application for network administrators

<Release 1.0.2>

**Low-Power Low-Cost IoT solution to patron traffic management in food establishment**

LOOI KAI WEN [A0199855A]  
POON CHUAN AN [A0210750W]

**Spacey checks restaurant spaces for you!**

Spacey harnesses the power of a network of BLE minions, all of which engage in frivolous advertising and shameless polygamy to show you how crowded your nearest eatery is! You can use Telegram app to find our bot @spacynusbot for a trial run!

While WiFi seems to be the preferred option in the implementation of the Internet of Things, it consumes too much power and compels too much reliance on the network router to communicate data. This project seeks to implement a cheaper, and less power-hungry solution to the dreaded problem of many -- Making a trip to an eatery only to find that there aren't any seats for you from the start.

**Feature 1: Telegram Chatbot Frontend**

Tech Stack: Python, Telegram API

Ask our spaceybot about the crowdedness of the NUS food establishment of your choice! You may choose to talk with the bot via telegram commands or by interacting with the button popups. SpaceyBot will then pose your query to the database and return an image illustrating the restaurant's occupancy status via an image file!

**Feature 2: BLE Wireless Sensor Network with ESP32**

Tech Stack: C++, Arduino Library

Involves a network of ESP32 Li-ION powered ESP32 sensors that relay occupancy information from the terminal sensor nodes to the space coordinator/gateway via Bluetooth Low-Energy Communication. Such is achieved by implementing BLE concepts such as Generic Attribute Profile to model Client/Server framework in the network.

**Feature 3: Initialization Wizard for Network Administrators**

Tech Stack: Python

We provide a GUI interface for developers to easily set up the database to be configured to the on-site setup of the wireless sensor nodes! The GUI Wizard allows the database and the sensor network to synchronise into a more seamless workflow and allows the resultant image to be generated more easily.

Thank you for installing the Spacey Node Manager. Kindly follow the instructions below for a painless setup for the BLE network, so that your sensor nodes can start to talk with your database!

If you have any issues, kindly visit <https://github.com/kaiwen98/spacey/> and create an issue! You can also contact the author on GitHub to pose your feedback!

## Brief Summary



The Spacey GUI serves as the coordinating software between the sensor network and the database, especially the image generated. The GUI allows users to enter sensor-specific details for each sensor and register it to the database, such that a change in occupancy registered by the sensor node on the ground will result in an update of occupancy information in the database, on the sensor data structure which corresponds with the sensor node in terms of said details.

The GUI also doubles as a graphic editor, whereby the administrator can denote seats in their relative positions along the fixed grid lines with simple mouse controls and keyboard shortcuts. He or she can choose to upload an image of the restaurant floor plan as overlay as reference. After saving the configuration from the GUI, the coordinates of the seats will be saved in the database, whereby retrieval during user query will be used by a separate image processing library to draw a more refined representation before sending them out to Telegram.

Please note that you are to use this GUI software with an existing sensor network setup. Kindly refer to the next section to quickly fill in on the required operational summary, so that you can begin your setup proper!

## Technical precursor

---

Please make sure to understand the following before you proceed:

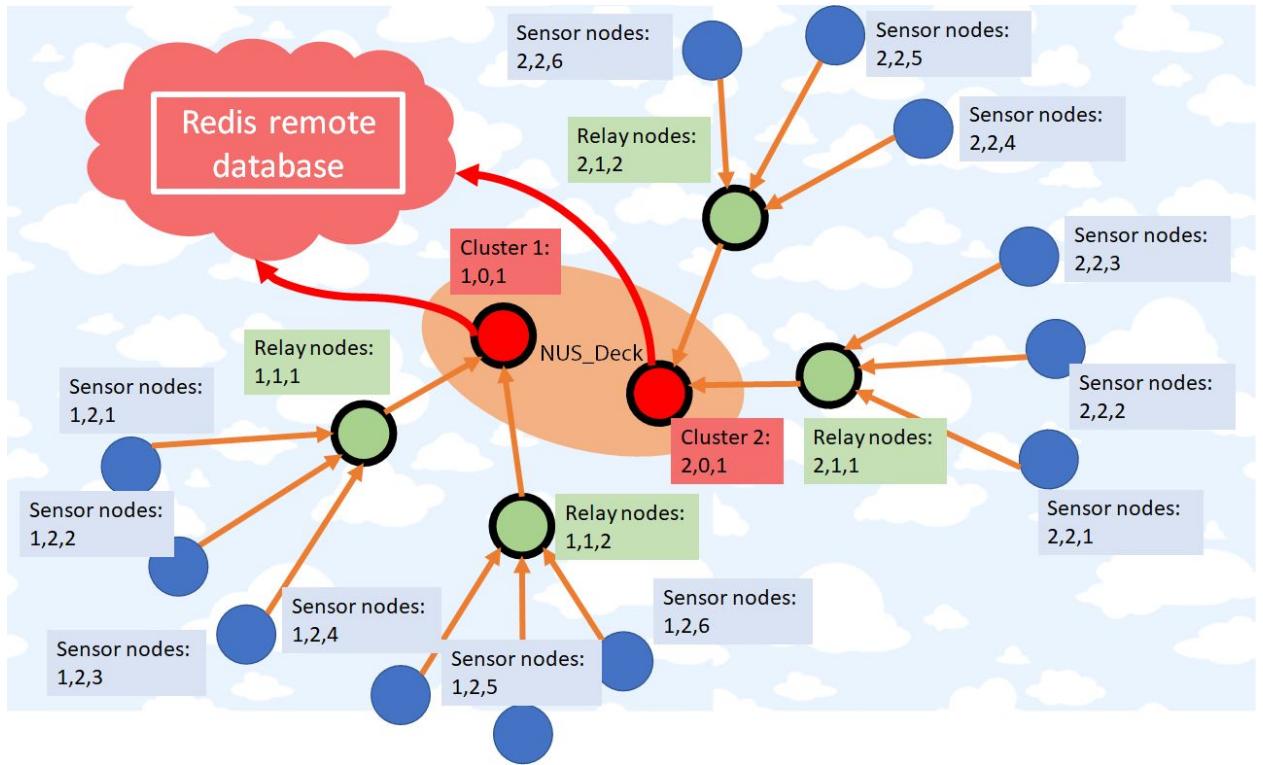
- **Hardware requirements:**

- a) Our solution requires the deployment of ESP32 with WIFI and BLE support. You can go to <https://github.com/kaiwen98/spacey> and download the dependency Arduino libraries required to program your devices. [kaiwen98/spacey/Sensor network/Spacey Nodes/]
- b) Our ESP32 sensor nodes use a capacitive touch sensor to detect seat occupancy. You will need to acquire them via DIY/ purchase from other suppliers.
  - i) The capacitive touch sensor is hooked up to GPIO 4 of the ESP32 microcontroller.
  - ii) While the threshold value is automatically calibrated for you, you will need to hardcode your values if you wish to change the sensitivity of the touch sensor.
- c) The ESP32 powers with minimally 2X18650 Lithium Ion batteries connected in series.
  - i) For an alternative power source, you are advised to acquire a power source that ranges between 5V to 12V as per ESP32 guidelines. ONLY POWER THE ESP32 THIS WAY VIA THE VIN PIN. THE ON-BOARD VOLTAGE REGULATOR STEPS DOWN THE VOLTAGE TO 3.3V AND WILL NOT BE ABLE TO SUPPORT HIGHER VOLTAGES.
  - ii) You may also power the board from a stable 3.3V power source, supplying it to the 3V3 pin on the board. YOU MUST ACCOMPANY THE SUPPLY WITH A FUNCTIONAL 3.3V VOLTAGE REGULATOR, SINCE THERE IS NO ELECTRICAL COMPONENTS OTHERWISE TO SHIELD THE ESP-WROOM MICROCONTROLLER FROM THE SPIKE IN VOLTAGE. WE STRONGLY RECOMMEND THE USE OF A BUCK-BOOST CONVERTOR ON ANY INDUSTRIAL-GRADE VOLTAGE REGULATOR, SUCH AS HT7333-A.

- **Software requirements:**

- a) You will need to install Arduino IDE to program your microcontroller the first time. Note that you only need to program it once for basic operations, resetting of the cluster number can be done while the sensor node is running.
- b) The GUI package from the release section on GitHub is already bundled with the required dependencies. The installer is only for Windows OS. Linux users will need to install the package as a Zip File from the GitHub page and run the program from the shortcut.
- c) For the resetting of the cluster number on the nodes, you will need to install nRF Connect. You can then communicate with the sensor directly to reset the cluster number. Refer to the appendix for explicit instructions.

- Setting up your BLE network:



#### Cluster Number conventions

- a) Each BLE node must have a unique cluster number to function properly. In addition, they must follow a certain convention as described below. This is because each node is configured to filter off nodes with incompatible cluster numbers to preserve the cluster independence and the directed flow of information. A relay node will only listen for BLE advertisements from nodes with the same cluster ID and a cluster level greater than its own cluster number by 1.

You will need to follow the following cluster number convention for the BLE network to work:

**XX,XX,XX**

*<cluster ID>, <cluster level>, <sensor ID>*

*[XX refers to an arbitrary integer between 0 to 255]]*

- Cluster ID refers to the sub-clusters that compose the entirety of your BLE network cluster and spans a subsection of your establishment. Each sub-cluster is represented by a single cluster head node, which consolidates all the information from the descendent nodes and relays them to the database. The bigger your restaurant, the more individual clusters you will need to span the entirety of the restaurant space.
  - Cluster Level refers to the distance of the node from the root node. All cluster head nodes have the cluster level of 0. If the node is the direct child of the root node, it will have the cluster level of 1.
  - Sensor ID is unique to each node that exists in the same cluster level in the same sub-cluster.
- b) Following the conventions above and ignoring connection-distance constraint, you can cover 256 seats with 256 + 51 + 10 + 2 nodes with:
- 256 sensor nodes with cluster level 3
  - 51 relay nodes with cluster level 2

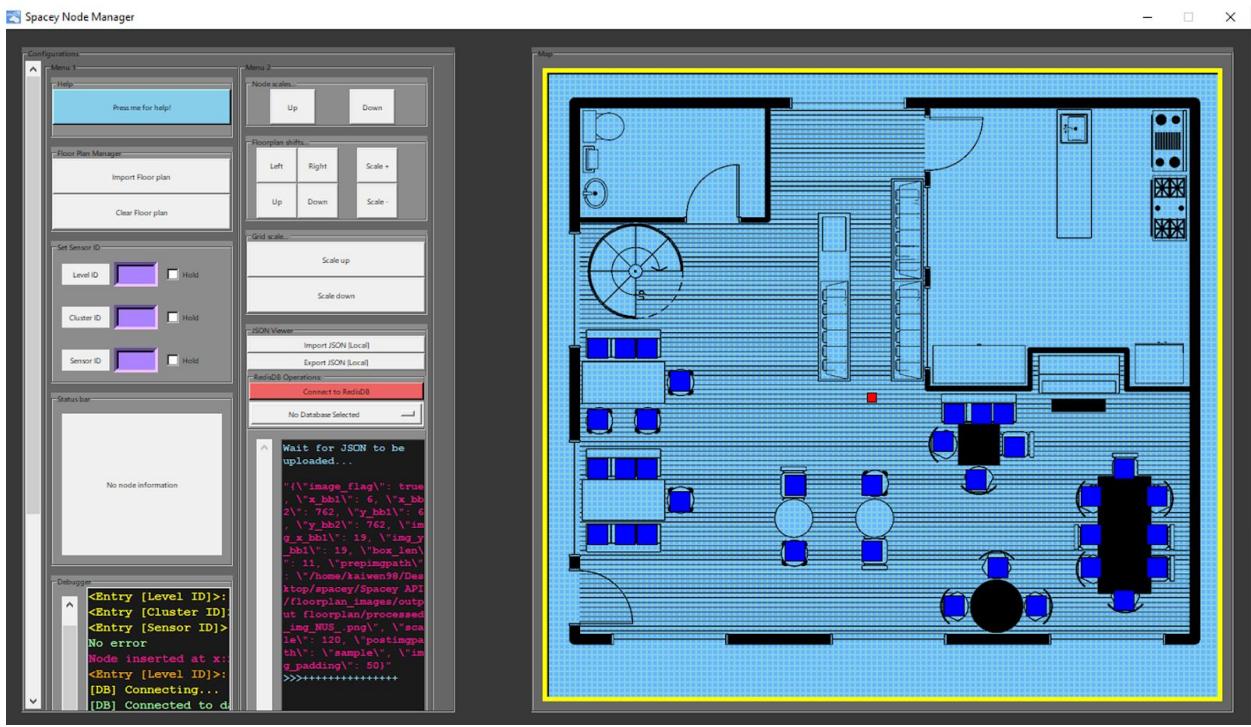
- iii) 10 relay nodes with cluster level 1
- iv) 2 cluster head nodes with cluster level 0

#### **Physical setup**

- c) Sensor nodes should be placed in a discrete location near the seats and reasonably undiscoverable by customers. The sensor nodes require a portable power source, but the relay nodes and the cluster head nodes can be connected to the mains depending on its location.
- d) Relay nodes one level below the sensor nodes should be distributed evenly such that each node covers around 5 or 6 sensor nodes ideally. Conversely, relay nodes should be distributed such that it covers around 5 to 6 direct children relay nodes. You must ensure that the parent and children nodes are within reasonable range of one another to make BLE transmission effective.
- e) Cluster head nodes MUST be placed in a location with strong WiFi signal.
- f) You are strongly advised to follow the following basic steps leading to the set-up of the BLE network:
  - i) Study your location and determine the possible locations of the various types of nodes.
  - ii) Have the nodes programmed with their respective sketches and assigned their cluster number.
  - iii) Place the nodes in location and ensure that the distance between each node is serviceable to maintain a reliable BLE connection.

You can then communicate your mapping of the various nodes with their assigned cluster numbers and their location to the GUI, which then converts the information to an output graphic which can be updated appropriately when the occupancy status of any of the nodes change.

## GUI Manual



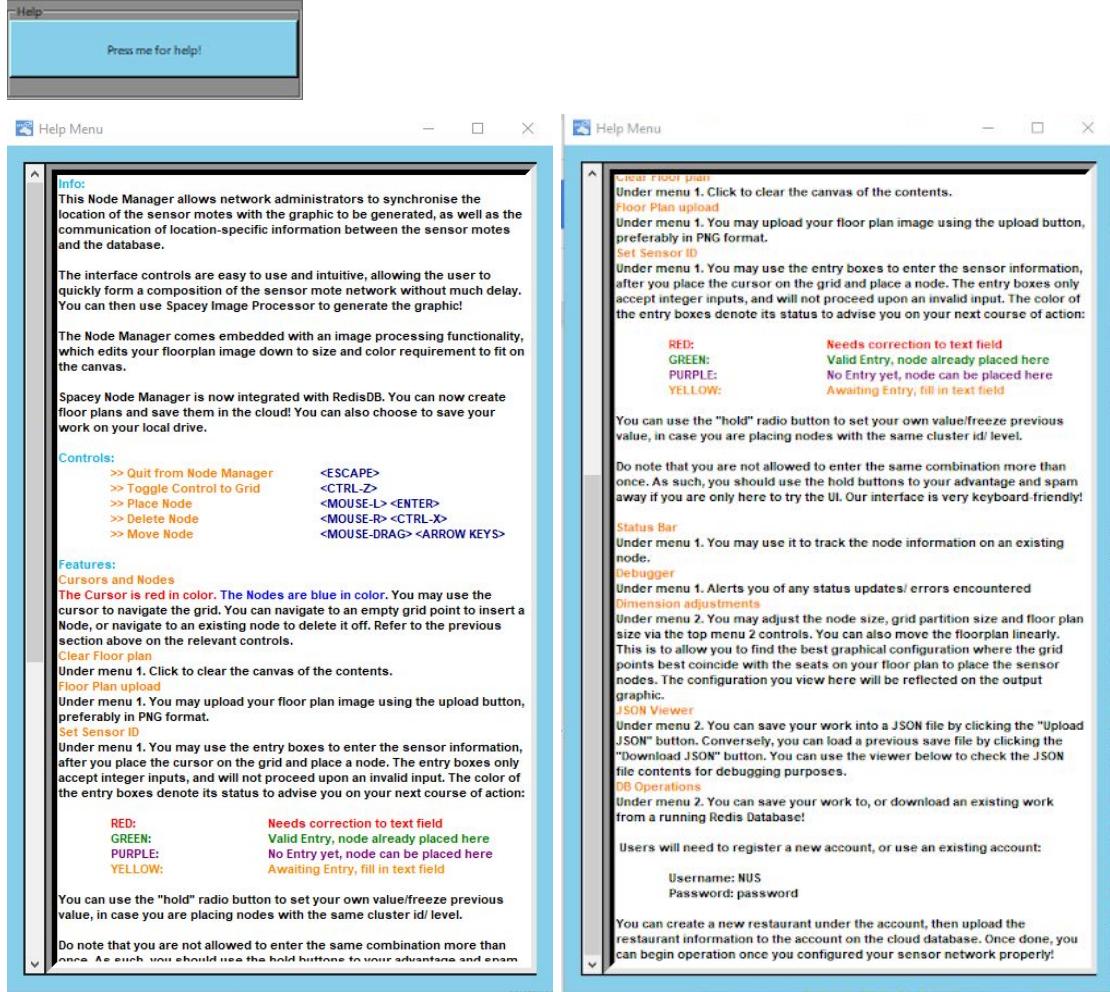
The GUI is rich with a plethora of features to help you integrate Spacey services to your establishment. The section below aims to document the various tools that are available to your free use.

- **Tips:**

- 1) What you see on the right is the Canvas, it depicts the output graphic in terms of dimension details, such as the size of the nodes, the spacing between them, and the relative border height and width etc. What you see on the canvas is approximately what will appear on your output graphic.
- 2) The left of the GUI are a list of controls that will be documented further below.
- 3) The GUI is designed to be extremely keyboard friendly. We highly encourage you to use the available keyboard controls to register the cluster numbers.
- 4) The GUI is also designed to be flexible in terms of size changes. Use the tools on the right to make sure your floor plan looks neat and tidy!
- 5) Refer to the help menu for further information.

- **Help Menu**

Features a pop-up window for users to browse through to quickly be acquainted with the controls of the GUI.

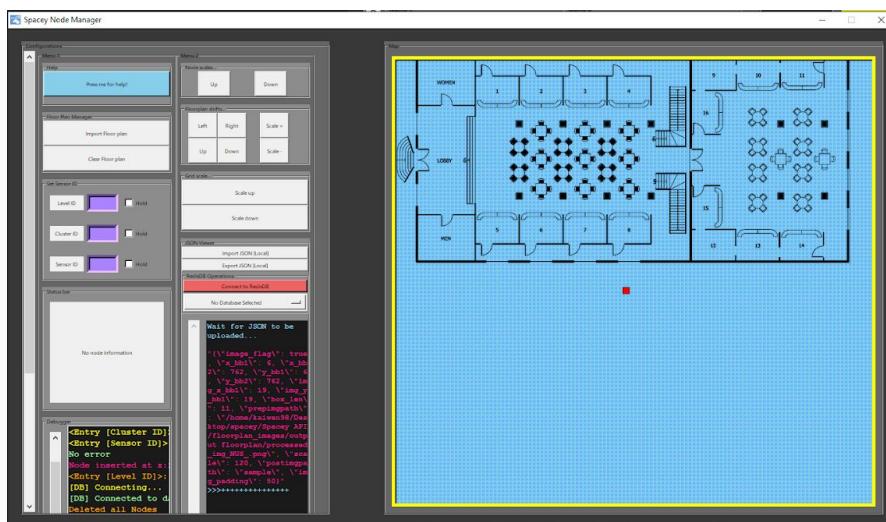
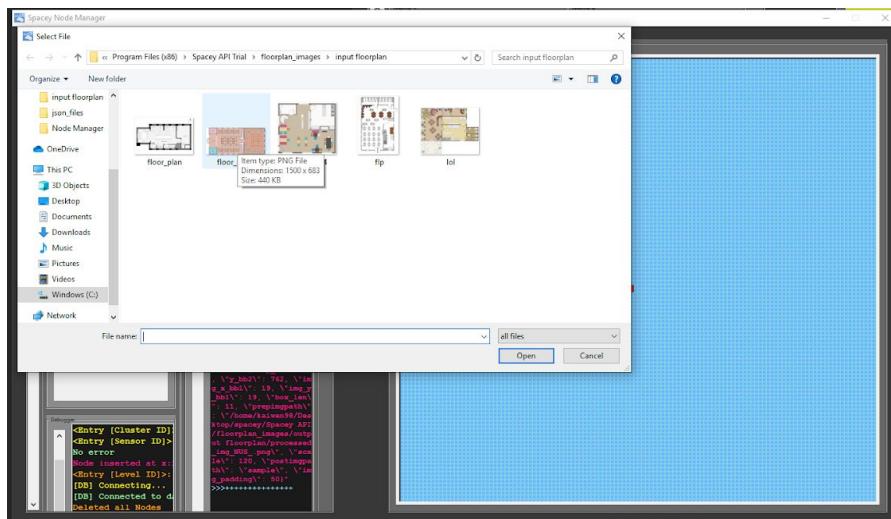
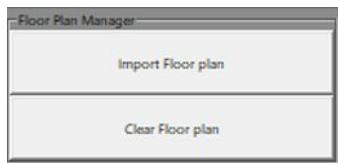


- **Floor Plan Control**

Floor Plans can be uploaded from the user's local drive to be processed. The GUI backend uses the Python Image Library to process the image to give a clean and standardised floor plan template to be used in the output graphic.

1. Convert the image to a sequence of RGBA pixel
2. If the pixel has greater values in the first 3 components (RGB), they are unlikely to form the outline of the floor plan. Hence, we set the pixel in the new image to alpha=0, meaning such regions will be transparent in the output.
3. If the pixel has smaller values in the first 3 components (RGB), they are likely to form the outline of the floor plan. Hence, we set the pixel in the new image to alpha=1 and of a standardised color, meaning such regions will appear in the output as outline.

After the process, the modified image will be saved in a PNG file and will be displayed on the GUI canvas.



- **Set Sensor ID**

Allows users to enter in the ID values of the sensor nodes. Although we have configured the 3 fields to be labelled as Level ID etc, the significance of each data field can be up to the administrator to decide, so long as the IDs set are all unique to one another.

To assist the administrators in the assignment of the sensor node IDs, we have implemented a hold feature for the user to freeze the values in place and edit the relevant field.

Default:

The screenshot shows the 'Set Sensor ID' interface with three input fields. Each field consists of a small white label box followed by a large purple rectangular input field. To the right of each input field is a small white checkbox labeled 'Hold'. The fields are labeled 'Level ID', 'Cluster ID', and 'Sensor ID' respectively.

The previous values are frozen:

This screenshot shows the same 'Set Sensor ID' interface as above, but with different values. The 'Level ID' and 'Cluster ID' fields now contain blue rectangular input fields with the number '2' displayed in them. The 'Sensor ID' field remains with its purple input field. Both the 'Level ID' and 'Cluster ID' 'Hold' checkboxes are checked.

Non-integer values not accepted:

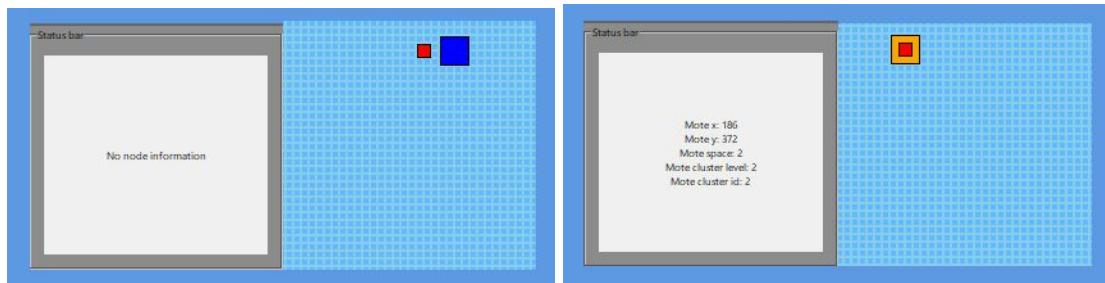
In this screenshot, the 'Level ID' field has been edited to contain the letter 'W' instead of a digit. This value is highlighted with a thick yellow border to indicate it is invalid or incorrect.

Valid combination of values accepted:

This screenshot shows the 'Set Sensor ID' interface again, but with valid integer values. The 'Level ID', 'Cluster ID', and 'Sensor ID' fields now have green rectangular input fields, indicating that the system has accepted the entered integer values.

- **Status Bar**

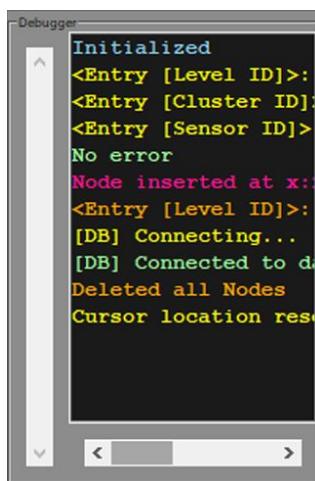
Displays the information of the nodes being placed, for reference purpose.



- **Debugger**

Displays status information and error messages, including:

- Error in input values of ID
- Not connected to network
- Reset of canvas
- Repeat in sensor values
- Invalid login information



- **Node scaling**

Allow the user to scale the node up and down in size for aesthetic purposes.



Before scaling:

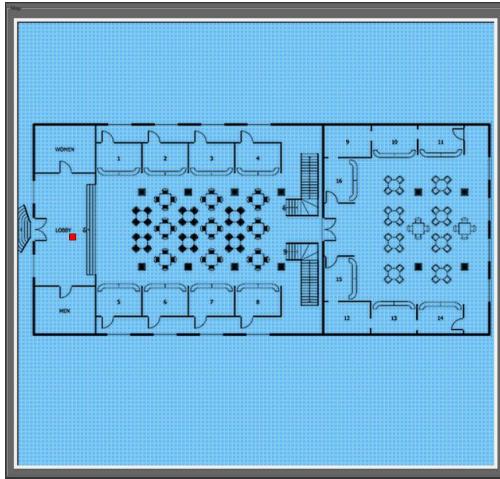
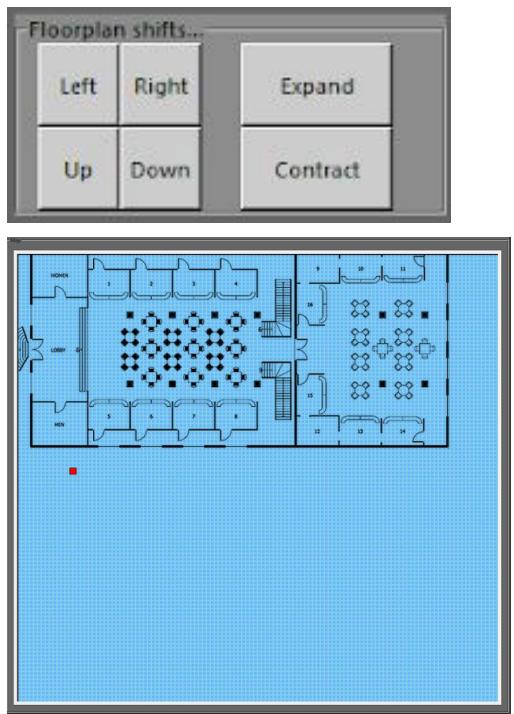


After scaling:



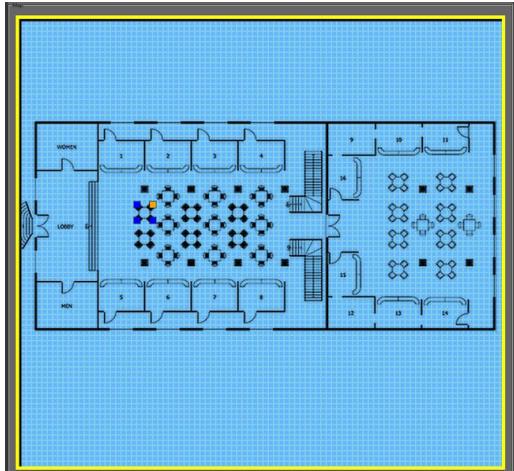
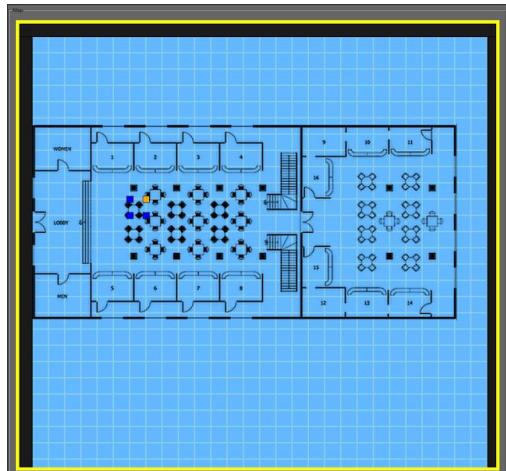
- **Floor Plan translation**

Enables the user to adjust the size and position offset of the image for aesthetic purposes.



- **Grid Scale**

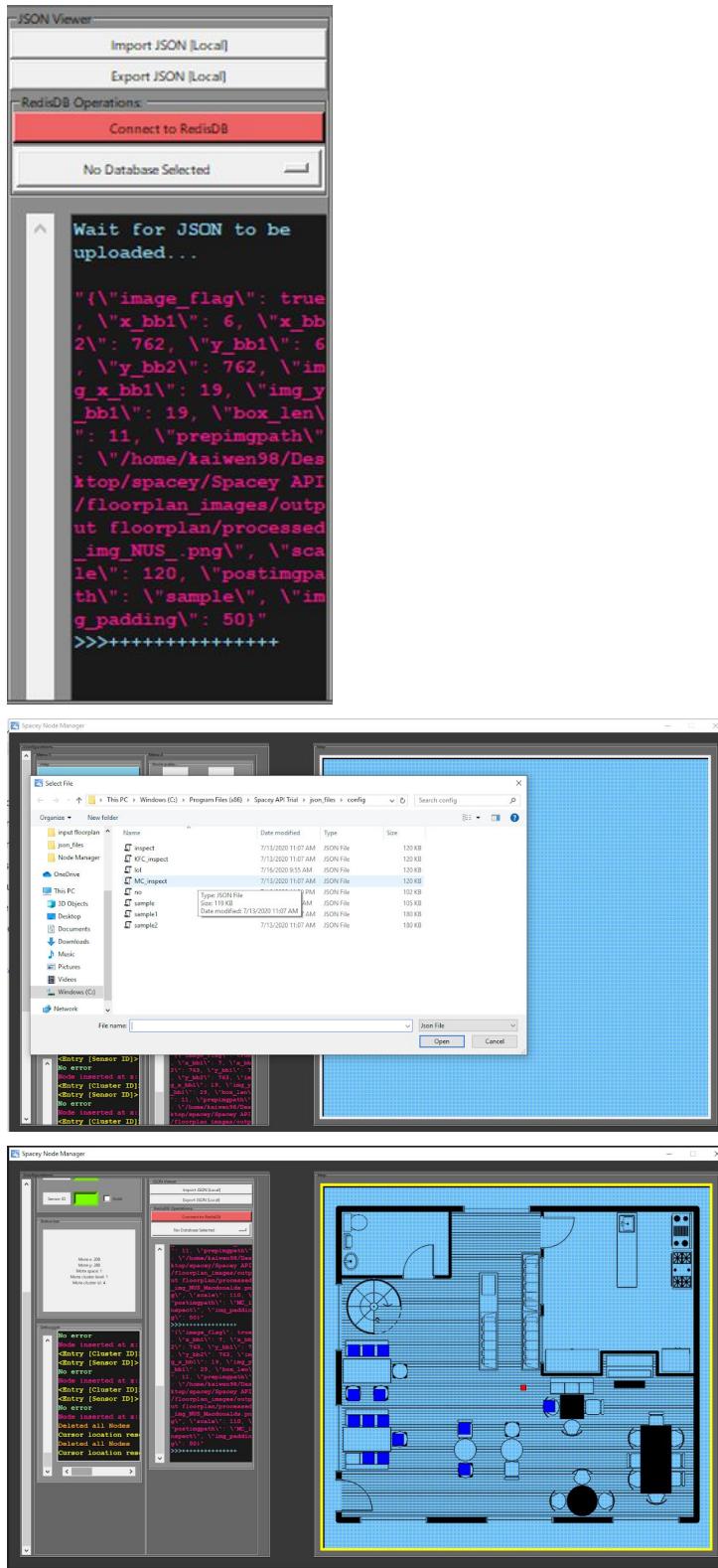
Recognizing the fact that floorplan may differ from one to the other by the relative positioning of the seats, we allow the users to adjust the scale of the grid so that they can find one that best coincides with the seats on the floor plan.



- **JSON viewer**

The GUI allows users to save their work into their local drive into a set of JSON files, in which they can load into the GUI in the future to continue their work.

The JSON viewer also allows the users to view the contents saved should they find the need to.



- DB capabilities

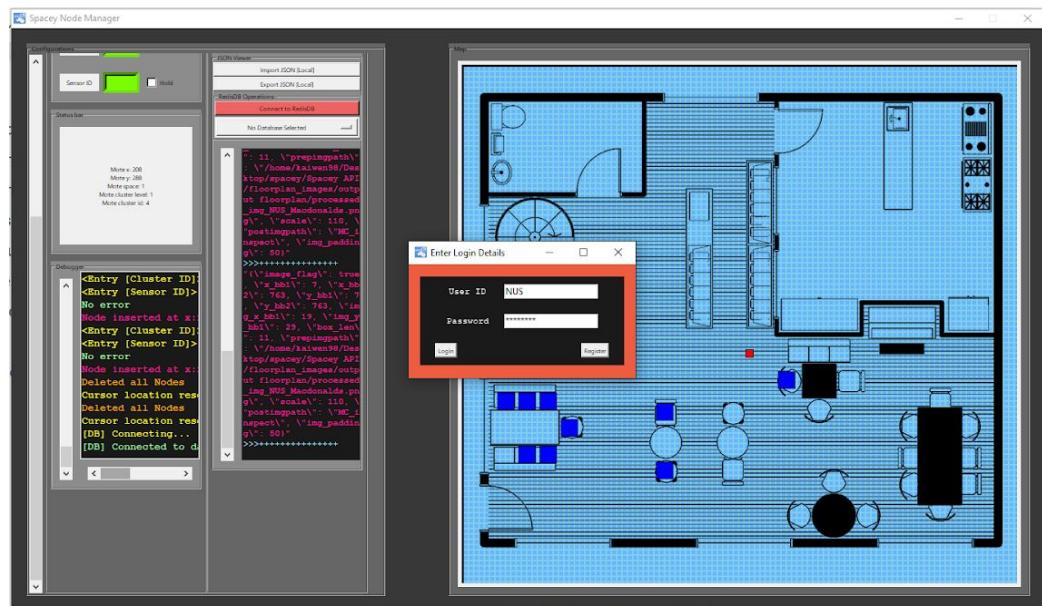
Administrators can connect to RedisDB to access uploaded floor plans/ upload floor plans to their account. They must register for a new account, or log in to an existing one to gain access to the database which they are authorised to view and edit.

The password of the users are not stored in the database as it is; their login details are processed through the "Spacey Secret Cipher" and SHA256 hash function before being stored in the database. In the event of accidental leak of data, the user's login details will not be easily compromised.

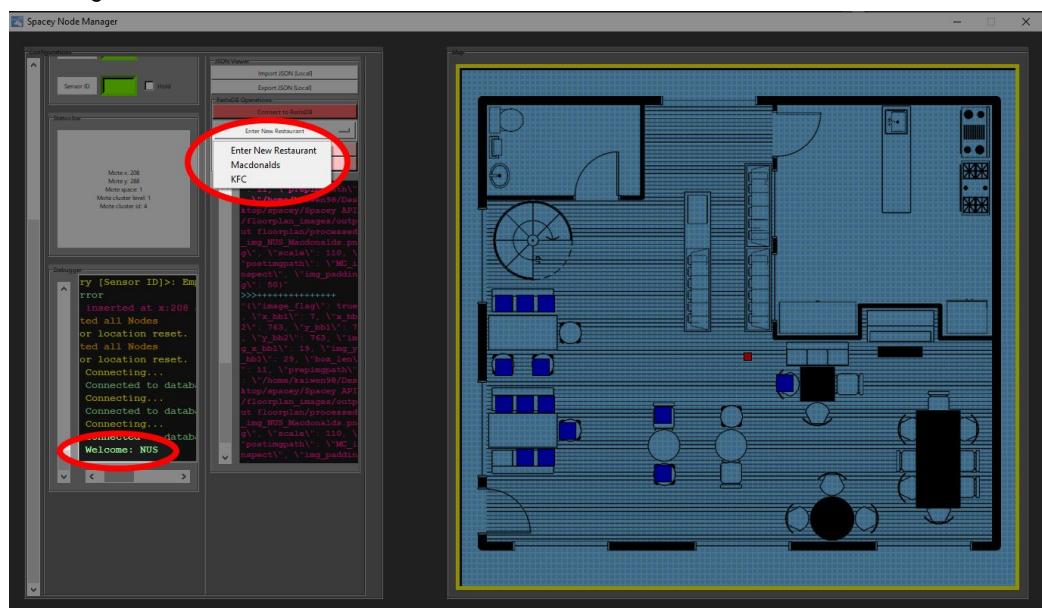
```
{'NUS': 'ec9193f8f25777fc0dbd511fdd617feee807ca9c4de6b51045b9cf98c535bcac'}
```

After a successful log-in, users can upload or download floor plans from existing restaurants for edit. They can even upload floor plans from their local drive to a new restaurant.

Before Login:



After Login:



Depending on the buttons you push, one of the following windows will show up:

Update Restaurant Det...

Update your restaurant information.

Restaurant Name

Enter location coordinates for Google Map support!

Latitude  Longitude

Restaurant Address

12 Science Drive 2, Singapore 117549

Operating hours:  
eg. <Mon-Sun, 7.00am - 9.30pm>  
Mon-Fri: 7.30am to 4.00pm\*  
Sat: 7.30am to 3.00pm  
\*Some stalls will be rostered to operate till 8.00pm during term time. Please check on NUS website

Enter Restaurant Details!

{Please enter the restaurant name, that field is compulsory!}

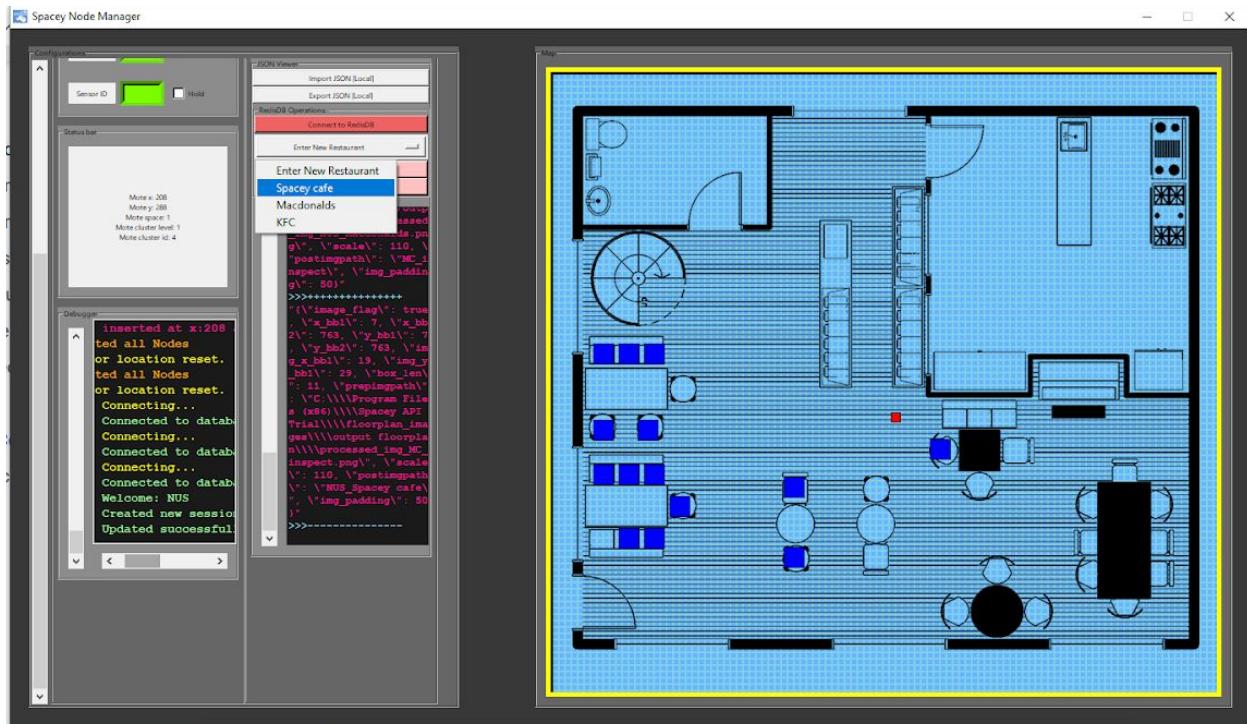
You can choose to input your restaurant details so that other customers can find you!}

Restaurant Name

Enter location coordinates for Google Map support!

Latitude  Longitude

Restaurant Address



## Appendix: Resetting a node's cluster number

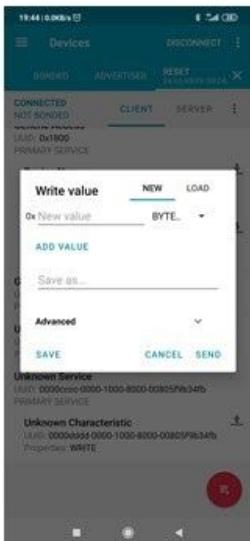


When the BOOT button on the node is pushed, it goes into a reset state, whereby it displays 3 slow blinks from the onboard LED.



The node will advertise itself as "Reset", and is discoverable from the nRFconnect app.

You can click on the first tab [Generic Access] to view the cluster number of the node. You can then click on the last tab [Unknown Service] to start writing to the node.



In the "New value" field, you may enter the new cluster number you want to assign to the node.



You may set a 6-digit pin within the Arduino sketch such that the correct pin needs to be entered before the cluster number entry for the node to accept it.



The node then reboots with a new cluster number.