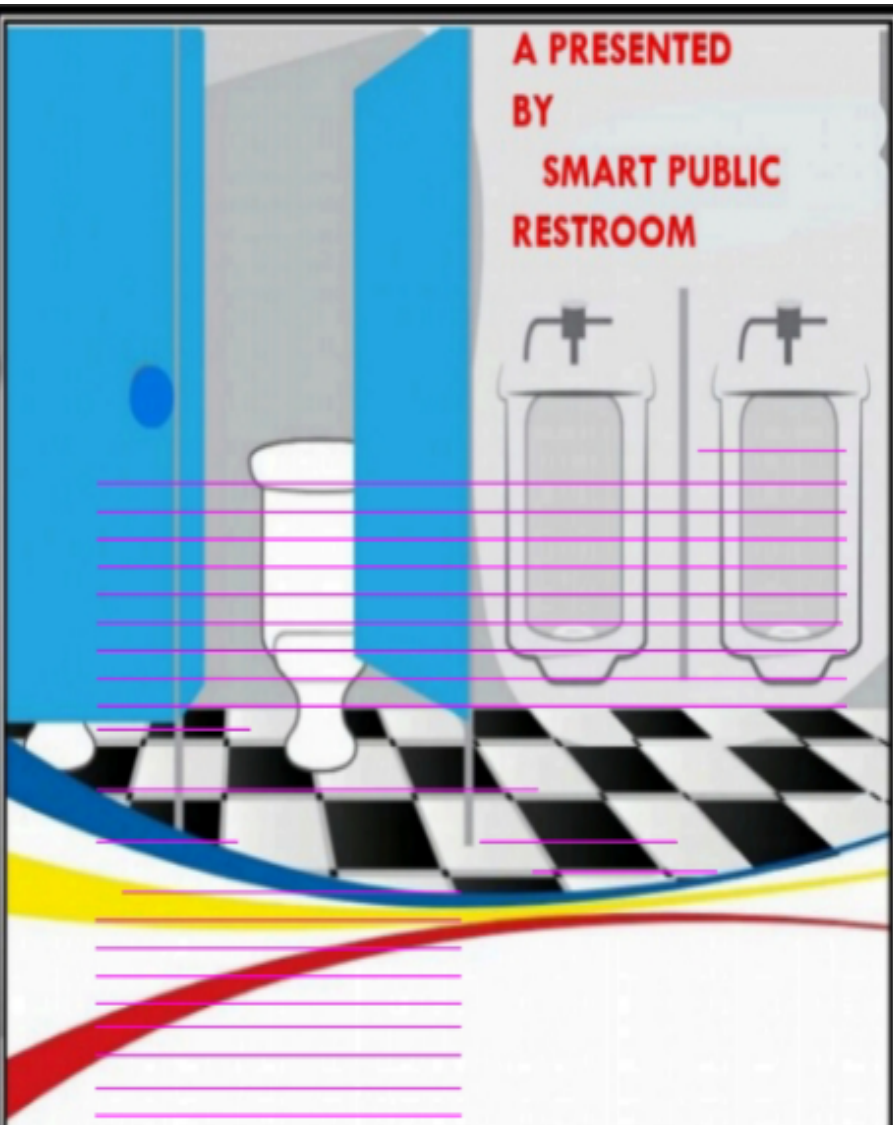
A PRESENTED
BY
SMART PUBLIC
RESTROOM



PRESENTED BY:

SATHISHKUMAR S
JOTHIKA M
VAIDHEGI S
VIJAY K
SUDHAKAR R

SMART PUBLIC RESTROOM

	Time/Weekdays	Monday	Tuesday	Wednesday	Thursday	Friday
	7:00 - 8:00	Janitor 1		Janitor 1		Janitor 1
F	8:00 - 9:00		Janitor 2		Janitor 2	
L	9:00 - 10:00	Janitor 1		Janitor 1		Janitor 1
O	10:00 - 11:00					
O	11:00 - 12:00	REST	REST	REST	REST	REST
R	12:00 - 13:00	Janitor 2	Janitor 1	Janitor 2	Janitor 1	Janitor 2
	13:00 - 14:00					
1	14:00 - 15:00					
	15:00 - 16:00					
	7:00 - 8:00		Janitor 2		Janitor 2	
F	8:00 - 9:00	Janitor 2	Janitor 1	Janitor 2	Janitor 1	Janitor 2
L	9:00 - 10:00					
O	10:00 - 11:00	Janitor 1		Janitor 1		Janitor 1
O	11:00 - 12:00	REST	REST	REST	REST	REST
R	12:00 - 13:00	Janitor 2		Janitor 2		Janitor 2
	13:00 - 14:00					
2	14:00 - 15:00					
	15:00 - 16:00					
	7:00 - 8:00	Janitor 2	Janitor 1	Janitor 2	Janitor 1	Janitor 2
F	8:00 - 9:00					
L	9:00 - 10:00					
O	10:00 - 11:00					
O	11:00 - 12:00	REST	REST	REST	REST	REST
R	12:00 - 13:00	Janitor 1		Janitor 1		Janitor 1
	13:00 - 14:00		Janitor 2		Janitor 2	
3	14:00 - 15:00					
	15:00 - 16:00		Janitor 3		Janitor 3	
	7:00 - 8:00					
F	8:00 - 9:00					
L	9:00 - 10:00		Janitor 1		Janitor 1	
O	10:00 - 11:00					
O	11:00 - 12:00	REST	REST	REST	REST	REST
R	12:00 - 13:00		Janitor 2		Janitor 2	
	13:00 - 14:00	Janitor 2		Janitor 2		Janitor 2
4	14:00 - 15:00	Janitor 1		Janitor 1		Janitor 1
	15:00 - 16:00		Janitor 1		Janitor 1	

FIGURE 20. Here is a sample schedule for janitors, complete with available shifts.

```
Toilet Schedule

  Time   User    Status

     0 Time Slot 00: Request Cleaning Toilet
     0 Time Slot 00: Cleaning in progress  0.000
     6 Time Slot 00: Cleaning Finished
    61 Time Slot 01: Request Cleaning Toilet
    61 Time Slot 01: Cleaning in progress  0.000
    76 Time Slot 02: Request Cleaning Toilet
    77 Time Slot 02: Delay after  1.087
    84 Time Slot 01: Cleaning Finished
   210 Time Slot 03: Request Cleaning Toilet
   210 Time Slot 03: Cleaning in progress  0.000
   215 Time Slot 03: Cleaning Finished
   243 Time Slot 04: Request Cleaning Toilet
   243 Time Slot 04: Cleaning in progress  0.000
   247 Time Slot 04: Cleaning Finished
   285 Time Slot 05: Request Cleaning Toilet
   285 Time Slot 05: Cleaning in progress  0.000
   290 Time Slot 05: Cleaning Finished
   348 Time Slot 06: Request Cleaning Toilet
   348 Time Slot 06: Cleaning in progress  0.000
   348 Time Slot 06: Cleaning Finished
   401 Time Slot 07: Request Cleaning Toilet
   401 Time Slot 07: Cleaning in progress  0.000
   410 Time Slot 07: Cleaning Finished
```

7 timeslot for 1 janitor

FIGURE 21. SimPy simulation result for janitor.

here. First, there is the total number of floors, which in this case is four, and the maximum number of persons per floor is thirty. The janitor's identification number follows, followed by four random names as references. Next is cleaning time, divided into eight rows starting with C1 and ending with C8. C1 is the start of the workday, usually around 7 am, and C8 is the finish, usually around 4 pm. The number of users is calculated using sensor data from the Internet of Things.

In Figure 24 to 28, the janitor schedule is compared to the recently suggested ARIGA model, an improved genetic

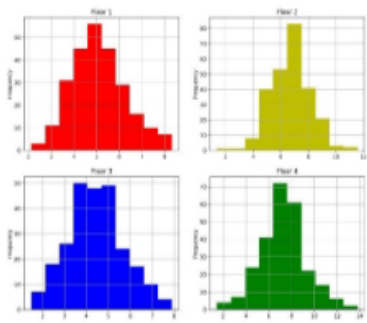


FIGURE 22. Histogram graph of user pattern with the selected hour.

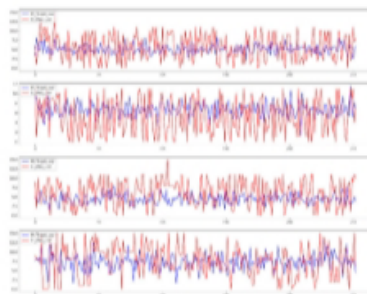


FIGURE 23. Line graph of real user with the targeted user.

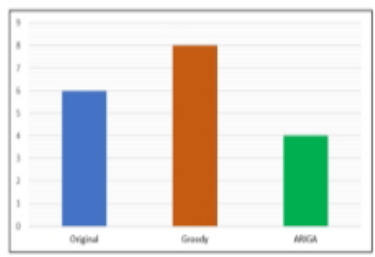


FIGURE 24. Cleaning demand = 10.

algorithm scheduling system. In the original system, all available time slots between the announcement date and the due date were chosen at random. The Greedy scheduler is also modelled for comparison. The Greedy scheduler gives each unit a cleaning time window.

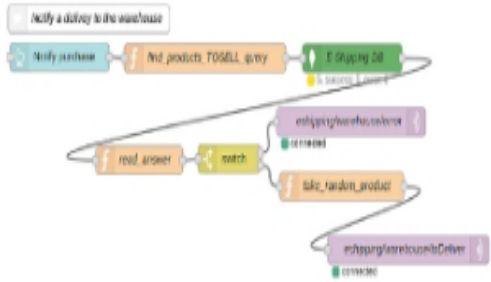


FIGURE 3 Node-RED flow—notification of a deliver to the warehouse

FIGURE 4 Node-RED flow—reception of a purchase's order notification

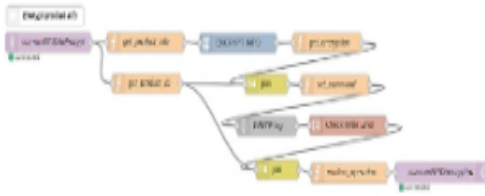


FIGURE 5 Node-RED flow—scanner's product encryption and write

supposed to be tracked by means of RFID tags; while smart vehicles are supposed to be equipped with a set of sensors, which automatically perform different tasks, depending on the information contained in the RFID tag associated to the delivery packages. The fundamental components of the system are the followings: (a) the smart vehicles; (b) the RFID tags; (c) the RFID scanners; and (d) the warehouse. Hence, the products can be associated with the following status: (a) *TO SELL*, if the corresponding product in the warehouse is waiting to be purchased; (b) *IN DELIVERY*, if the corresponding product in the warehouse is waiting to be delivered (ie, it has just been sold); and (c) *DELIVERED*, if the corresponding product has already been delivered. The smart vehicles are supposed to be equipped with the following sensors and devices: GPS, vibration (inspired by Arduino SW-420), humidity, temperature, dehumidifier, motor fridge. Different kinds of delivery can be carried out, depending on the features of the transported product: (a) *STANDARD*, in case of "standard" products a check is performed only on the geographical coordinates for setting up the route; (b) *FRIGO*, in case of fresh or frozen products, a check is done on the coordinates, the temperature, and the humidity of the smart vehicle; and (c) *FRAGILE*, in case of fragile products, a check is done on the coordinates and on the vibration level of the smart vehicle. Moreover, each vehicle must be in one of the following status: (a) *AVAILABLE*, if the corresponding vehicle is not engaged in any delivery; (b) *IN DELIVERING*, if the corresponding vehicle is engaged in a delivery. In the next section, the just detailed scenario will be properly designed and developed within Node-RED tool.

4 | NODE-RED DESIGN AND DEVELOPMENT

Node-RED[9] comes from an open project, developed by IBM, proposing a flow-based and event-driven programming tool. The application's behavior is thus represented as a network of black-boxes, which may communicate with each other and regulate the flow of the information within the designed system. A visual browser-based representation supports designers and developers in better understanding the happening interactions within the whole IoT network. In fact, many entities may be involved, both hardware (eg, sensors) and software (eg, services) ones. Node-RED also enables the real connection of hardware devices

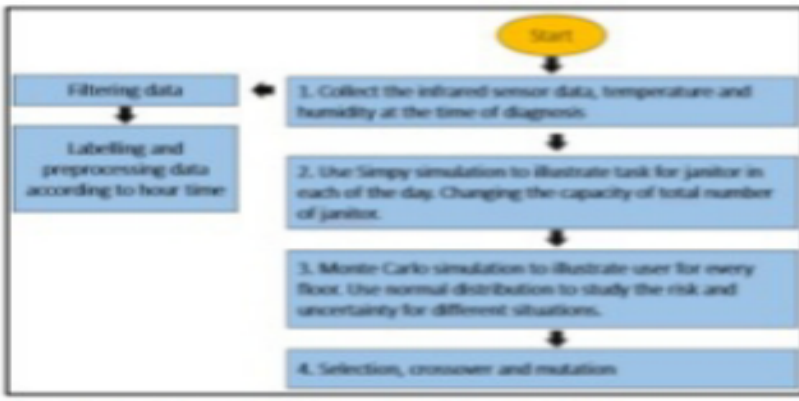


FIGURE 13. A diagram depicting a scheduling algorithm.

using random numbers as inputs into a mathematical model. Probability-based solutions to difficult mathematics problems. It studies risk and uncertainty in many contexts in academia. Risk and uncertainty assessments benefit greatly from the normal distribution. The simulation data follow a normal distribution;

- Standard deviation = 0.4
- Number of repetitions = 12
- Number of simulations = 253
- User target value = 0 to 25
- User probability = 0 to 1

Scheduling involves user input, data point relationships, system constraints, and genetic algorithm (GA) optimization. Despite not being the optimal choice for all topics, genetic algorithms (GAs) can sometimes provide a reasonable response. Despite being dubbed genetic algorithms. GA can solve even the hardest problems, unlike traditional algorithms, which solve issues step-by-step. GA optimizes problem-solving. If genetic operators execute ideally, GA can aid with functions specified across complex, discrete structures. GA requires this.

Inputs include the number of floors in each building, the number of janitors, the working days, and the starting time and timeslot. Systems can have harsh and soft limitations. However, strict boundaries are unchangeable. The scheduling engine always follows severe limitations. Soft limitations are negotiable. Soft constraints are less rigid. The scheduling engine will try to follow your soft limitations, but it may stray if necessary. The study limits janitors to one shift each day and one per level. Due to such constraints, the janitor cannot take a vacation during business hours and must evenly space out their shifts.

This study uses evolutionary computation to find a schedule within a suitable timeframe. A genetic algorithm uses natural selection and population genetics to search probabilistically. Figure 14 shows a quick genetic algorithm flow. Initial schedules are generated. The crossover and mutation operations merge two schedules into one, and then the schedule

TABLE 2. Sharp infrared sensor specifications.

Type	Specification
Operating voltage	4.5 to 5.5 V
Operating current	30 mA
Measuring range	100 to 550 cm
Output type	Analog
Dimension	58 x 17.6 x 22.5 mm
Mounting hole	4.2 mm

FIGURE 9. Programming Flowcharts for the Raspberry Pi.

re-entered into the database once the internet connection is stable again.

Different toilets have different bathroom detection distances. Males and females are measured by average human height. 50–200 cm. Most people defecate sitting or squatting. Defecating at 35 degrees is best. Each cubicle, squatting toilet, and bowl toilet has infrared sensors on top. Figure 10 shows that a sensor will point straight toward the cubicle's occupant. The sensor angle adjusts to x1 and x2.

The dataset was collected from the Faculty of Computing and Informatics (FCI) at Multimedia University (MMU) and the vaccination center at the Dewan Tun Canselor MMU. The team completed the data capture layer task for the proposed work. Sensors collected contextual data including

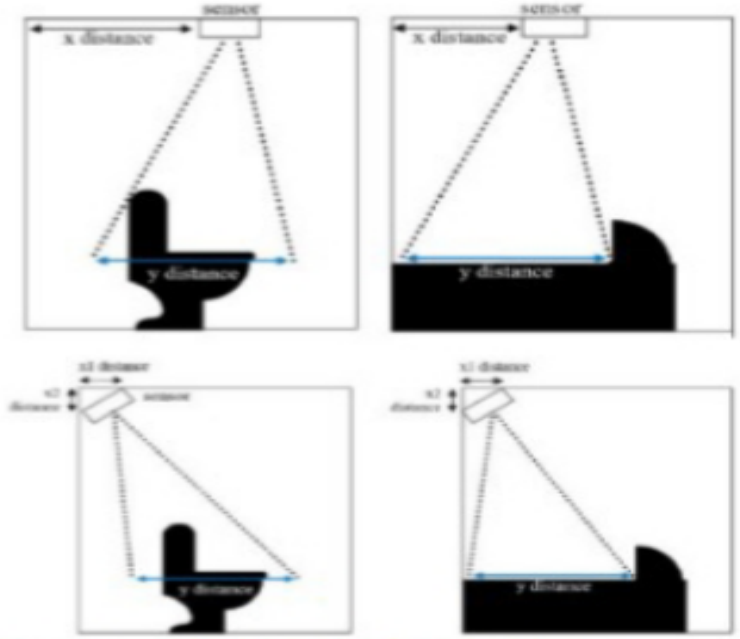


FIGURE 10. Inside the cubicle, where the sensor is located.

motion detection, distance, temperature, humidity, total user, ram, and so on. Passive Infrared (PIR) sensors were used to determine user occupancy in motion detection states 1–4. The experiment collected data from September 2019 through December 2021. Each floor of the FCI building included seven toilet facilities, three male and four female.

E. DATA ACQUISITION LAYER

Line graphs better showed user statistics. The x-axis shows the date, while the y-axis shows the number of individuals in each cubicle. Since university buildings are busy with classes, the total number of persons using them increases during the semester but decreases between semesters. From this vantage point, we can see which cubicles visitors use most for their toilet type (squatting or sitting). Sitting toilets have more users than squatting toilets, and most of them won't utilize the last cubicle. Figure 11 shows that cubicle 3 has fewer individuals than cubicles 1 and 2.

Pre-processing removed data outliers. Data noise was assumed since data recording is vulnerable to numerous external factors. Data recording prompted these hypotheses. Like outliers, this incident had several possible causes, such as sensor malfunction, measurement error, etc. Moving averages, loess, lowess, Rloess, RLowess, and Savitsky-Golay smoothing filters help eliminate outliers. Use Savitsky-Golay. In this study, the moving average filter was employed to make enormous volumes of data easier to deal with, as previous researchers have done [59].

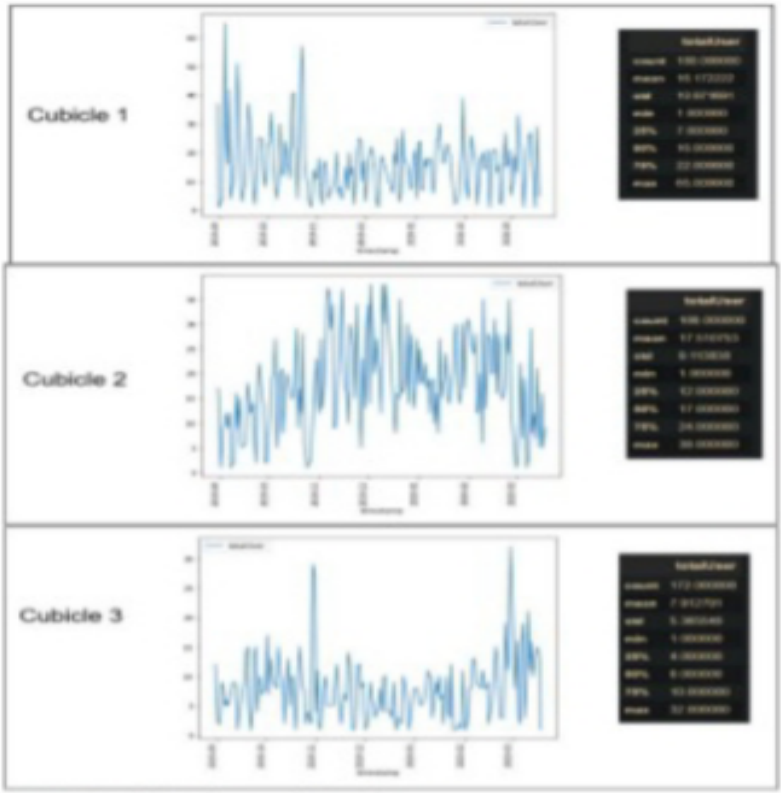


FIGURE 11. A line diagram for IoT data.

Inconsistent data and large samples require normalization. This reduced the information needed to make a prognosis and schedule. The machine learning models used normalized data with values in the same range. This ensures that weights and biases converge gradually. Normalizing whole sample data for machine learning algorithm models enhanced predictions and training.

Below are instructions for implementing this method. I'll explain the suggested method's operation after this. ARIGA combines ARIMA and Genetic Algorithm for predictive maintenance and scheduling. The algorithm will improve outcomes by combining both.

F. PREDICTION METHOD WORKFLOW

Predictive maintenance follows data collection. Pre-processing follows database export. Filtering data to predict only important facts. Step 1 of Figure 12 uses IoT monitoring sensor data to determine the degradation feature value at the diagnostic time. Next, feature value trends determine the predicted failure time distribution (step 2). The distribution can be calculated using the histogram of the remaining useful life based on many similar failure events. Finally, the distribution estimates maintenance costs for each projected future maintenance (step 3). Considering the distribution's uncertainty, the maintenance time is the projected time before an unexpected failure.

FIGURE 6. Architectural details of the process.

TABLE 1. Software description in tabular form.

Software	Specification
Operating System	Microsoft Windows 11 (64-bit)
Machine learning tool	Anaconda Navigator v1.18
Programming language	Python and C programming
Database platform	Open Innovation Platform

manages data, services, and IoT devices. Table 1 lists the system's software requirements.

C. HARDWARE ENVIRONMENT

Raspberry Pi and ESP32 boards helped build the sensor module. A strong CPU was needed to pre-process data for analysis in the proposed system's control kernel, which required Wi-Fi connectivity, the ability to integrate many sensors, and online data storage. After reviewing the many prototyping boards available, we selected the finest components for these needs. The Raspberry Pi and TTGO microcontrollers can meet the requirements as control kernels, according to studies. Thus, the project's control kernel was the Raspberry Pi and TTGO OS. Raspberry Pi, a single-board computer (SOC), runs Linux and may be upgraded by swapping the memory card. This expedites upgrades. Like a computer, it can multitask. Raspberry Pi applications can leverage networking, data transport, databases, and web servers. Secure Shell allows remote access. The Raspberry Pi cannot do ADC. The Raspberry Pi with TTGO combined control kernel for real-world applications is affordable for small and micro-businesses. Depending on the information needed, different sensors are used. It sends sensor data from wired or wireless networks to

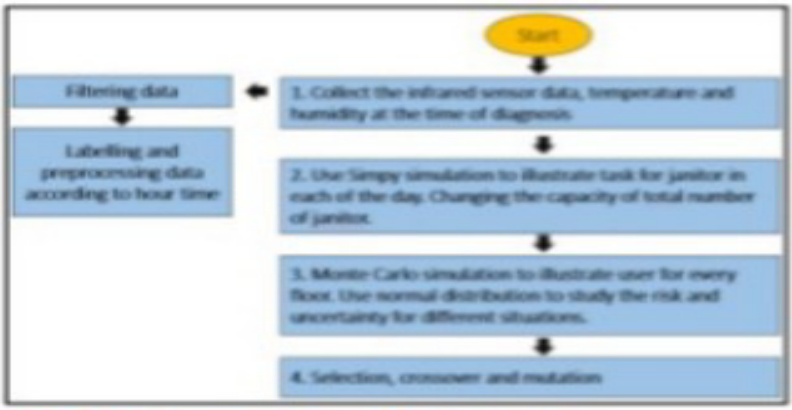


FIGURE 13. A diagram depicting a scheduling algorithm.

using random numbers as inputs into a mathematical model. Probability-based solutions to difficult mathematics problems. It studies risk and uncertainty in many contexts in academia. Risk and uncertainty assessments benefit greatly from the normal distribution. The simulation data follow a normal distribution;

- Standard deviation = 0.4
- Number of repetitions = 12
- Number of simulations = 253
- User target value = 0 to 25
- User probability = 0 to 1

Scheduling involves user input, data point relationships, system constraints, and genetic algorithm (GA) optimization. Despite not being the optimal choice for all topics, genetic algorithms (GAs) can sometimes provide a reasonable response. Despite being dubbed genetic algorithms. GA can solve even the hardest problems, unlike traditional algorithms, which solve issues step-by-step. GA optimizes problem-solving. If genetic operators execute ideally, GA can aid with functions specified across complex, discrete structures. GA requires this.

Inputs include the number of floors in each building, the number of janitors, the working days, and the starting time and timeslot. Systems can have harsh and soft limitations. However, strict boundaries are unchangeable. The scheduling engine always follows severe limitations. Soft limitations are negotiable. Soft constraints are less rigid. The scheduling engine will try to follow your soft limitations, but it may stray if necessary. The study limits janitors to one shift each day and one per level. Due to such constraints, the janitor cannot take a vacation during business hours and must evenly space out their shifts.

This study uses evolutionary computation to find a schedule within a suitable timeframe. A genetic algorithm uses natural selection and population genetics to search probabilistically. Figure 14 shows a quick genetic algorithm flow. Initial schedules are generated. The crossover and mutation operations merge two schedules into one, and then the schedule

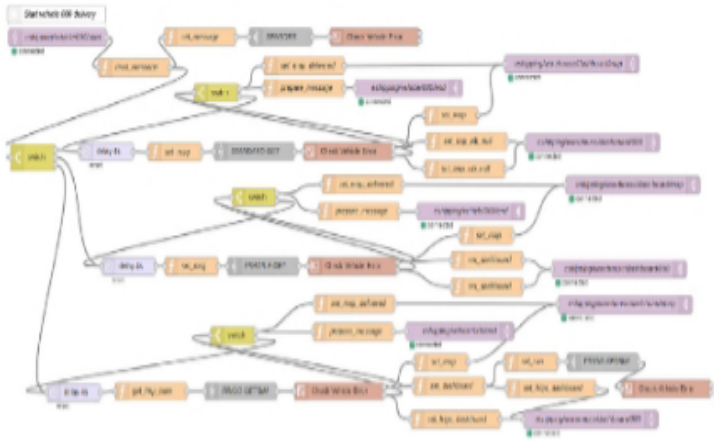


FIGURE 8 Node-RED flow—smart vehicle behavior

FIGURE 9 Dashboard showing information about the vehicles and statistics about products and alarms

Another dashboard provides real-time information about the status of the vehicles (coordinates, temperature, humidity, vibration), as sketched in Figure 9, which also shows statistics about the status of the products and the alarms. Note that also the creation and update of the dashboards are managed via proper Node-RED flows.

Summarizing, the obtained system provides: (a) interoperability among RFID devices, a central control system, which operates with MQTT notifications, HTTP requests/responses, and different programming languages; (b) real-time data processing; (c) dashboards for simulations and for users, reporting useful statistics. The system's components are also well-structured, so as modules can be independently modified, depending on the need of the SME which may decide to adopt the solution.

5 | CONCLUSION

The paper presented a smart transport and logistics scenario, designed, and developed by means of Node-RED tool. It aimed at demonstrating the feasibility and usability of Node-RED in representing and testing the behavior of IoT environments, due to the user-friendly interface and easy-to-use components.

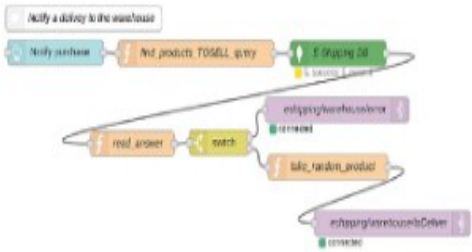


FIGURE 3 Node-RED flow—notification of a deliver to the warehouse

FIGURE 4 Node-RED flow—reception of a purchase's order notification

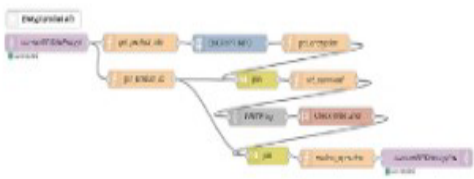


FIGURE 5 Node-RED flow—scanner's product encryption and write

supposed to be tracked by means of RFID tags; while smart vehicles are supposed to be equipped with a set of sensors, which automatically perform different tasks, depending on the information contained in the RFID tag associated to the delivery packages. The fundamental components of the system are the followings: (a) the smart vehicles; (b) the RFID tags; (c) the RFID scanners; and (d) the warehouse. Hence, the products can be associated with the following status: (a) *TO SELL*, if the corresponding product in the warehouse is waiting to be purchased; (b) *IN DELIVERY*, if the corresponding product in the warehouse is waiting to be delivered (ie, it has just been sold); and (c) *DELIVERED*, if the corresponding product has already been delivered. The smart vehicles are supposed to be equipped with the following sensors and devices: GPS, vibration (inspired by Arduino SW-420), humidity, temperature, dehumidifier, motor fridge. Different kinds of delivery can be carried out, depending on the features of the transported product: (a) *STANDARD*, in case of "standard" products a check is performed only on the geographical coordinates for setting up the route; (b) *FRIGO*, in case of fresh or frozen products, a check is done on the coordinates, the temperature, and the humidity of the smart vehicle; and (c) *FRAGILE*, in case of fragile products, a check is done on the coordinates and on the vibration level of the smart vehicle. Moreover, each vehicle must be in one of the following status: (a) *AVAILABLE*, if the corresponding vehicle is not engaged in any delivery; (b) *IN DELIVERING*, if the corresponding vehicle is engaged in a delivery. In the next section, the just detailed scenario will be properly designed and developed within Node-RED tool.

4 | NODE-RED DESIGN AND DEVELOPMENT

Node-RED[9] comes from an open project, developed by IBM, proposing a flow-based and event-driven programming tool. The application's behavior is thus represented as a network of black-boxes, which may communicate with each other and regulate the flow of the information within the designed system. A visual browser-based representation supports designers and developers in better understanding the happening interactions within the whole IoT network. In fact, many entities may be involved, both hardware (eg, sensors) and software (eg, services) ones. Node-RED also enables the real connection of hardware devices

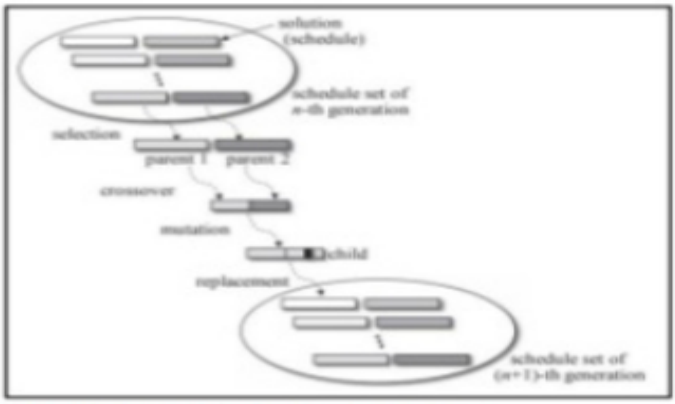


FIGURE 14. Scheduling step of a genetic algorithm.

set is evolved by replacing a schedule in the old set with the newly formed schedule. Repeat until the schedule set converges.

The approach presents a timeline as a matrix. The matrix rows represent the overall number of users, floors, and hours in the day, while the columns represent the real schedule. Each matrix cell might be 0 or 1. 0 means no cleaning, and 1 means yes. This investigation's encoding gave each floor covering kind its row. To clarify, the floor and total users are treated separately and given sufficient time to perform properly. This method simplifies encoding over defining numerous values for each matrix element. Two scheduling conditions should be considered when dividing jobs among available periods. Interruptible and non-interruptible scheduling. We use a finer time slot to shift an interruptible activity onto several non-interruptible jobs.

Selecting two existing schedules will create a new generation of schedules. This analysis uses the normal operating method, which yields the best schedule with four times the likelihood of the worst schedule. The pointer selects and creates the parenting schedule for the current time slot. Click the time slot. This is a probabilistic rule that favors better scheduling. We're using a probabilistic technique to avoid unfair classifications. Deterministic selection of the best schedules may quickly dominate the scheduling set, causing early convergence to a local optimum. We risk becoming dependent on their schedules.

After parental unit selection, the population undergoes crossover. Crossover recommends combining the productive parts of both parents to conceive a child, hoping to simplify routines. Crossovers create new generations. This includes randomly identifying a crossover location and swapping genes from both parents. Slicing a matrix column generates the crossing point.

Mutation operations, which change the developed child for variety, should be integrated into crossover operations, but this paper does not discuss mutation operations. The classic one-point crossover approach may not be practical since it

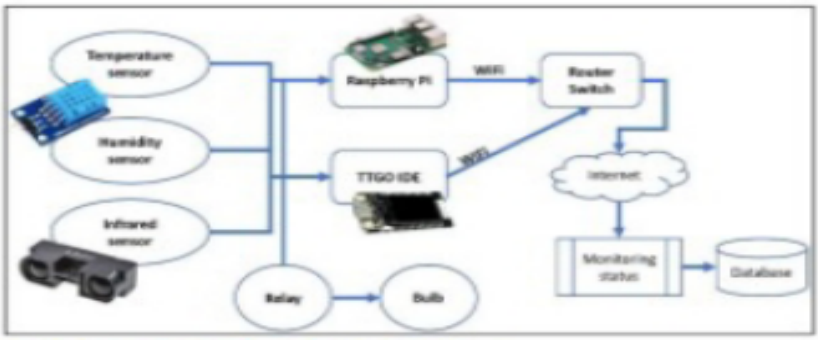


FIGURE 8. Floor plan for toilet implementation.

This study included temperature, humidity, and infrared radiation sensors. Sharp, the industry leader, makes most infrared detectors and rangers. Sharp Infrared Detectors and Rangers, available in many combinations, can measure distances precisely. Infrared distance sensors generate an analog signal proportional to the sensor's distance from the thing being measured. The datasheet states that the SHARP GP2Y0A710K0F output voltage ranges from 2.5 V at 100 cm to 1.4 V at 500 cm. Distance detection allows this range. Table 2 lists Sharp sensor specs. The ultrasonic sensor was superior to the infrared sensor [58]. The author details the monitoring mechanism. Because a human's body isn't flat on the area being scanned, ultrasonic sensors are better at identifying them. Increased detection distance reduces infrared sensor precision.

D. CODE IMPLEMENTATION AND EXECUTION

The Raspberry Pi's standard Python editor, IDLE (Integrated Development and Learning Environment), helped us program the gadget. An interpreter lets you test each instruction to validate the code. This site uses Python 3.5.3. Figure 9 depicts the method used to count restroom users. The system operates from 7 am until 10 pm. We chose this time because it matches the school's kids' and teachers' working hours. If the time limit is surpassed, the system won't run. A user can be "enter," "stay," "leave," or "nobody" depending on whether motion detection detects movement.

The program's initial iteration will check the pre-distance. If the distance is less than the limit, motion detection 1 will not be done. After this, motion detection 2 requires maintaining stationary for a certain time before moving on. To trigger motion detection level 3, the final and most essential stage, the distance must be much larger than the permissible margin of error. Moving on to the fourth and last motion detection, which will result in no movement. The word "nobody" indicates a distance reading error. After that, the gain will be calculated from scratch for the next iteration. The database will contain all available information. If there is a difficulty with the internet connection, it will try to reconnect without pausing or interrupting the program. The data will be

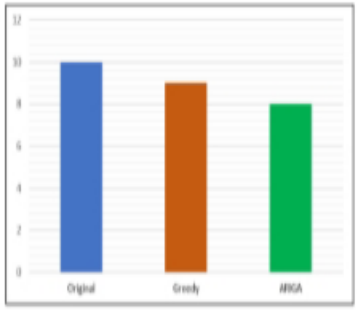


FIGURE 25. Cleaning demand = 20.

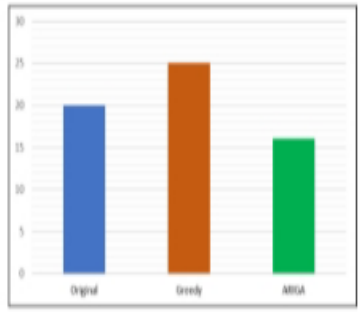


FIGURE 28. Cleaning demand = 50.

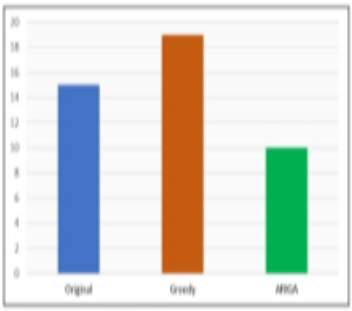


FIGURE 26. Cleaning demand = 30.

FIGURE 29. Janitor scheduling dashboard for facility view.

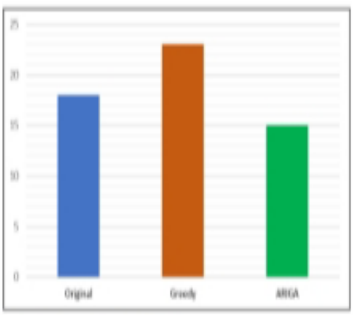


FIGURE 27. Cleaning demand = 40.

As shown in the figure, the proposed scheduling system performs better than the competition regardless of building size. The ARIGA model enhances performance by 24.7% on average and 15.0% overall compared to the baseline scheduling technique. In a high-rise structure with more than 50 cleaning demands, the proposed scheduling method performs better. The ARIGA model schedules most cleaning time demands for off-peak hours while trying not to exceed the threshold for a high number of cleaners. Since the ARIGA model was created, this is why. As cleaning time needs rise, the needed total number of janitors may surpass the threshold for increasing the number of janitors. A mechanism that lets users exchange their schedules has been proposed to fix this issue.

The Greedy scheduler performs marginally better than the baseline system in a scenario with 20 cleanliness needs. When the building's floors grow, the Greedy scheduler's performance worsens much more than the first schedulers. The Greedy scheduler can reduce performance by 29.4% compared to the first scheduling. Progressive stage systems can lead to unexpected results like this. The Greedy scheduler avoids the peak period, but it doesn't account for price progression, therefore it may produce excessive load in other periods.

Figure 29 shows how to see and manage janitorial schedules on the smart toilet dashboard's Scheduling System tab. By changing the janitor's name and ID, the facility management staff can receive help. The facilities management team can tell the cleaning firm how many janitors are needed to clean all the floors.

In all analyses, the proposed ARIGA approach performs well. Data collection and querying are simplified using ARIGA. During pre-processing, the ARIGA preserves space, and the findings reveal that the model's space requirements increase linearly with data size. ARIGA and LSTM, two archetypal approaches for forecasting time series data, are analyzed and compared in this paper. Using a range of cleaning demand scenarios, we found that the proposed algorithm

Source code:

```
[
    {
        "id": "febe5669d5db",
        "type": "mqtt in",
        "z": "df98344e.b5f8a",
        "name": "Occupancy
Status",
        "topic": "restroom/
occupancy",
        "qos": "2",
        "datatype": "auto",
        "broker": "2cfa9b4e.
efb416",
        "x": 150,
        "y": 260,
        "wires": [
            [
                "
54301f3650e8d8cf"
            ]
        ]
    },
    {
        "id": "54301f3650e8d8cf
",
        "type": "ui_text",
        "z": "df98344e.b5f8a",
        "group": "e4d6a356.
70a758",
        "order": 0,
        "width": "6",
        "height": "2",
        "name": "",
        "label": "Restroom Status"
,
        "format": "{{msg.payload}}
",
        "layout": "row-spread",
        "x": 410,
        "y": 260,
        "wires": []
    },
    {
        "id": "2cfa9b4e.efb416",
        "type": "mqtt-broker",
        "z": "",
        "name": "Local MQTT
Broker",
```

```
"z": "",
        "
name": "
Local
MQTT
Broker",
        "
broker": "
localhost",
        "
port": "
1883",
        "
clientid": "",
        "
usetls":
false,
        "
compatm ode":
true,
        "
keepalive": "60",
        "
cleansess ion": true,          "
birthTopi c": "",
        "
birthQos": "0",
        "
birthPayl oad": "",
        "
closeTopi c": "",
        "
closeQos": "0",
        "
closePayl oad": "",
        "
willTopic": "",
        "
willQos": "0",
        "
willPayloa d": ""
    },
    {
        "id": "
e4d6a35 6.70a758 ",
        "
type": "ui _group",
        "
name": "
```

```
        "id": "e4d6a356.70a758",
        "type": "ui_group",
        "name": "Restroom Status"
,
        "tab": "70c4cc0c.19325",
        "order": 1,
        "disp": true,
        "width": "6",
        "collapse": false
    },
    {
        "id": "70c4cc0c.19325",
        "type": "ui_tab",
        "name": "Restroom
Monitoring",
        "icon": "dashboard",
        "order": 4,
        "disabled": false,
        "hidden": false
    }
]
```

Output:
When you trigger the occupancy sensor, the Node-RED dashboard should update in real-time to display the "Occupied" or "Vacant" status of the restroom.

This is a simplified example to get you started. For a complete smart public restroom IoT project, you would need to integrate more sensors, actuators, and more sophisticated logic.

Figure 5.2 Traffic signal controller: Source C-DAC

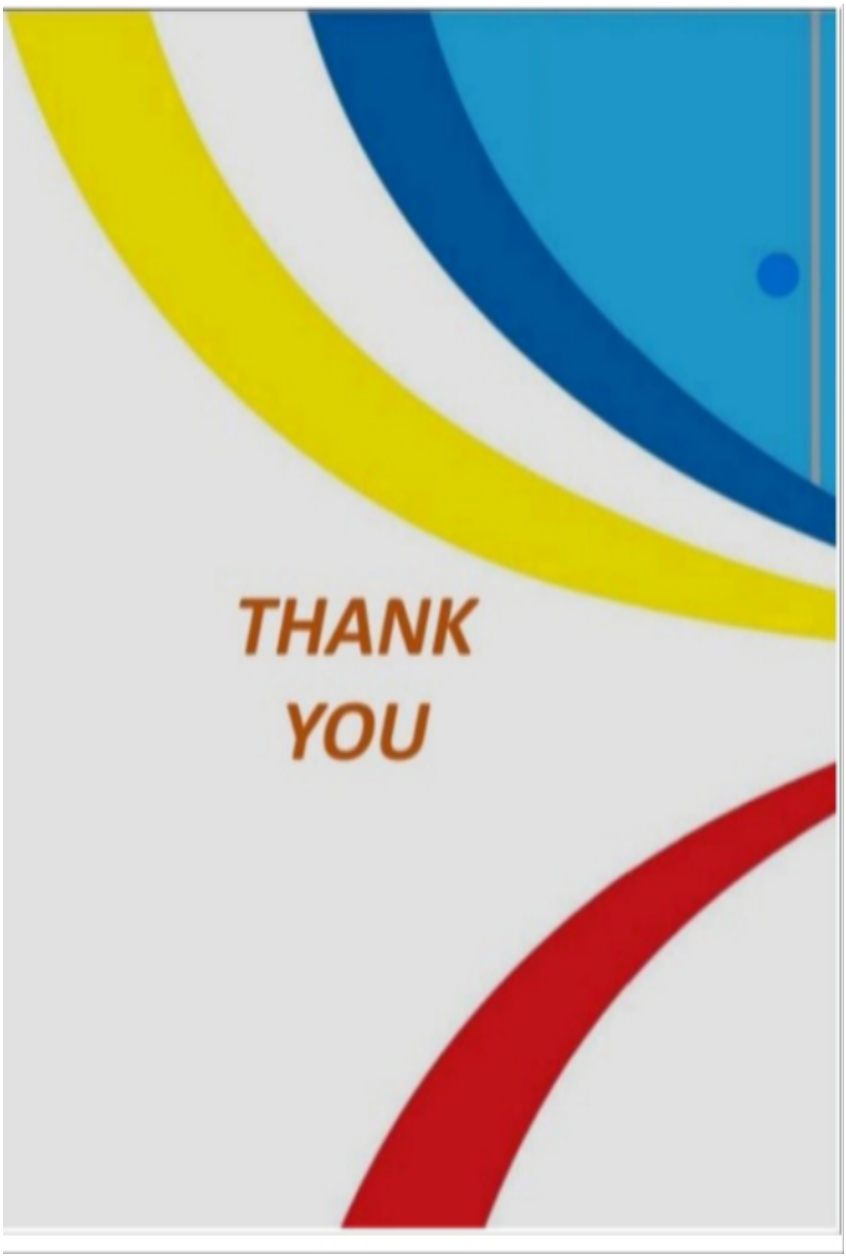
Some popular ATC models/ system are

- SCOOT – Split Cycle & Offset Optimization Techniques
- CoSiCoSt – Composite Signal Control Strategy
- SCATS – Sydney Coordinated Adaptive Traffic System
- OPAC - Optimization Policies for Adaptive Control
- RHODES – Real time Hierarchical Optimized Distributed and Effective System
- ACS -lite – Adaptive Control Software Lite
- SPOT/ UTOPIA – Urban Traffic Optimization by Integrated Automation
- MOTION – Method for the Optimization of Traffic Signals in On-line Controlled
- ITACA – Intelligent adaptive control area
- RTACL – Real time traffic adaptive control logic

CoSiCoSt is developed by C- DAC, Thiruvananthapuram, Govt. of India. As per comparison shown by C – DAC the difference in most popular system SCOOT, SCATS and CoSiCoSt are

Table 5.5: ATC Models / System

SCOOT	SCATS	CoSiCoSt
Centralized System	Hierarchical system	Distributed system
medium Scalability	Limited scalability	High scalability
Upstream detection	Stop line detection	Stop line detection
At every lane	For critical junction only	For every lane with filters
Good for homogeneous traffic and prediction limited for heterogeneous traffic	Good in ideal conditions	Good congestion management
Fallback - fixed	Fallback - fixed	Fallback - VA
Model based	Algorithmic	Model based
Time generated	Plan selection	Time generated


THANK
YOU