

Monitoring and Control of IITM campus water distribution network using GSM as a redundant system

A Project Report

submitted by

PARTHA PAUL

in partial fulfillment of the requirements
for the award of the degree of

MASTER OF TECHNOLOGY



DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, MADRAS.

May 2018

THESIS CERTIFICATE

This is to certify that the report entitled "**Monitoring and Control of IITM campus water distribution network using GSM as a redundant system**", submitted by Partha Paul, to the Indian Institute of Technology, Madras, for the award of the degree of Master of Technology, is a bonafide record of the research work carried out by him under my supervision. The contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Sridhara Kumar Narasimhan
Research Guide
Associate Professor
Dept. of Chemical Engineering
IIT-Madras, 600 036

Dr. Ramkrishna Pasumarthi
Research Co-Guide
Associate Professor
Dept. of Electrical Engineering
IIT-Madras, 600 036

Place: Chennai

Date:

ACKNOWLEDGEMENTS

Successful completion of this project work would not have been possible without the guidance, support and encouragement of many people. I take this opportunity to express my sincere gratitude and appreciation to them.

First and foremost I offer my earnest gratitude to my guide, Dr. Sridhara Kumar Narasimhan who has supported me throughout my project work with his patience and knowledge while allowing me freedom and flexibility to follow my own thought process.

A special thanks is also to my Co-Guide, Dr. Ramkrishna Pasumathy who patiently listened, evaluated, criticized and guided me periodically. The invaluable inputs and suggestions from him were instrumental in steering the project work in the right direction.

I would like to express my sincere gratitude and thanks to Mr. N Murali whose persistent efforts and pursuit of practical knowledge has been motivational.

My special thanks and deepest gratitude to Saravanan Chinnusamy, Prasanna Mohandoss, , Rohit R who has been very supportive. They have enriched the project experience with their active participation, deep understanding and knowledge of the subject matter and invaluable suggestions.

I would like to thank my parents for supporting and encouraging me throughout my studies at IIT and for tolerating my prolonged absence from home.

ABSTRACT

Water supply in most of the Indian cities is intermittent with poor levels of service. The poor service levels can be attributed to constraints on available water and resources, limited instrumentation, improper operation of the system and poor network maintenance. To monitor and control these geographically distributed networks, measuring instruments should be placed in appropriate locations and manual valves should be automated and appropriate communication between the devices enabled. The proposed method involves real time water level monitoring of storage reservoirs in Water Distribution Networks (WDN) and remote actuation of valves using Internet of Things (IoT) enabled devices. A low power wireless sensor and actuator network is developed for monitoring and control of water distribution networks and deployed in the IIT Madras water distribution network. The network consists of low cost water level measurement module (Remote Nodes), electrically actuated valves and, gateway nodes. LoRa/GSM is used to communicate between the nodes. The remote node is installed in one storage reservoirs and electrical actuators. A Raspberry Pi based gateway is developed which collects the level data from all Remote nodes and sends the data to local server and sends control signals to the actuator. A php based web application, an android application, and an open source SCADA are used for data visualization and also serves as a human machine interface for manipulating the valve remotely. LoRa is used as a primary communication between sensors and gateway. In case LoRa communication fails, GSM will act as a secondary communication to collect data from sensor and actuator.

This work involves real time data acquisition from level sensors and remote control of an actuator through GSM. This work also involves checking of availability of primary communication by sending random values from the gateways. As LoRa acts as primary communication channel, and GSM acts as secondary communication channel, the work involves clear understanding of LoRa communication protocol.

This project also involves design of web application and mobile application including configuration of an open source SCADA system. SCADA stands for supervisory control and data acquisition. The work includes simulation in order to decide correctness of implementation. This report presents various high level and low level design considerations and decisions to describe the design of the overall system. It also presents the interfaces among various modules. The project mostly involves software design aspects.

TABLE OF CONTENTS

THESIS CERTIFICATE
ACKNOWLEDGEMENTS

ABSTRACT

CHAPTER-1

INTRODUCTION

1.1	IoT in WDN monitoring	7
1.2	Related work	7
1.3	system architecture	10
1.4	HARDWARE DESIGN	10
1.4.1	Remote Node	10
1.4.1	Actuator Node	11
1.4.3	Gateway Node	12

CHAPTER-2

Device interface and data storage

2.1	Use of python program for interfacing devices and communication redundancy	13
2.2	Data storage	15

CHAPTER-3

Design of user interface

3.1	Design of web application	25
3.2	Design of mobile application	33
3.2.1	open/close command	34
3.2.2	Displaying data	34
3.3	Installation and configuration of SCADA	35
3.3.1	Supproted datatypes	36
3.3.2	Dara source	36
3.3.3	Datapoints	37
3.3.4	Sound alarm	38
3.3.5	Graphics	38
3.3.7	Adding Data Sources	39
3.3.6	Adding Data Points	40
3.3.7	Viewing the data through Watch List and Graphs	40
3.3.8	Alarm configuration	43
3.3.9	SQL query and daily report	43

CHAPTER-4

GSM operation and communication redundancy

4.1	Introduction	46
4.2	How 4G can be implemented?	47
4.3	Why are we using 2G?	50
4.4	2G operation mode	50
4.5	Algorithm	51
4.5.1	AT command	51
4.5.2	Checking by Arduino that modem is available for operation	52
4.5.3	Sending SMS	53
4.5.4	Receiving SMS	54
4.5.6	Function of the Arduino connected to the level sensor	55
4.5.7	Function of the Arduino installed with actuator	56
4.5.8	Reading SMS by the Arduino installed on the server	57
4.5.9	How the data is stored in database through GSM?	58
4.6	How Redundancy in communication is achieved?	59
4.6.1	updating random values by Gateway-1	59
4.6.2	updating random value by Gateway-2	60
4.6.3	Updating primary link availability in database	61
4.6.4	Sending SMS automatically by the server when primary link fails	62
4.6.5	Getting data through GSM manually	63

CHAPTER-5

Simulation

5.1	Simulation for the actuator:	77
5.1.1	Getting actuator status on mobile phone: after sending a SMS(@mobile#),	77
5.1.2	Sending an open command from user interface	79
5.2	Getting data from the level sensor installed at RR sump through GSM	80
5.2.1	Getting level data manually from the user interface:	82

CONCLUSION

88

REFERENCES

88

TABLE OF FLOWCHART

AT command	64
Checking by Arduino that modem is available for operation	65
Sending SMS	66
Receiving SMS	67
Function of the Arduino connected to the level sensor	68
Function of the Arduino installed with actuator	69
Reading SMS by the Arduino installed on the server	70
How the data is stored in database through GSM?	71
updating random values by Gateway-1	72
updating random value by Gateway-2	73
Updating primary link availability in database	74
Sending SMS automatically by the server when primary link fails(Arduino)	75
Sending SMS automatically by the server when primary link fails(Server)	76

CHAPTER-1

INTRODUCTION

1.1 IoT in WDN monitoring

Drinking water distribution systems carry potable water from water sources to consumers through complex pipe networks. Service reservoirs are used as intermediate storage devices to smooth and meet demands. Most households also own local storage tanks to further mitigate the variability of water supply. Though the WDN is initially designed to supply water continuously, rapid population growth and unplanned expansion of network results in intermittent supply. Poor service is further compounded by lack of instrumentation, improper operation and poor network maintenance. Continuous monitoring of WDN parameters such as, pressure, flow, water quality and water levels in storage reservoirs can improve the quality of service, reduce the Non-Revenue Water (NRW) by balancing supply and demand. The proposed method involves real time water level monitoring of reservoirs in Water Distribution Network (WDN) and remote actuation of valves using Internet of Things (IoT) enabled devices. IoT in WDN is used to collect crucial data about the system for monitoring, scheduling and equitable supply of water to consumers.

1.2 Related work

Recently smart WDN monitoring using Internet of things (IoT) has received more attention. The deployment of IoT devices in water distribution network monitoring is limited by the technical challenges such as long communication distance and cost. 3G/4G/LTE communication is still expensive for monitoring WDN and not energy efficient for remote monitoring devices which are resource and power constrained. Also, using 3G/4G need procurement of router/firewall and public static IP address which are expensive. For cost effectiveness, GSM has been used as secondary communication. Collecting of data from WDN and remote control of electrical actuator is the primary goal of this project. Data are collected from level sensors installed on underground water tank to know the water level. Level sensor and the actuator are connected to two controllers, one for GSM, one for LoRA. Sensors are installed at Lake-well, children park , staff-club, and RR sump. The actuator and the

sensors installed at Lake-well, children park , and staff-club are directly or indirectly connected to a gateway (gateway-1) installed at staff club. The level sensor installed at RR sump is directly connected to another gateway(gateway-2) installed at RR sump. Level data are stored in a database server placed in the process control lab. Data are stored in two ways.

1. Through loRA, gateway, and ethernet combination(Link-1)
2. Directly through GSM(Link-2)

Our primary communication channel for data storage is link-1. In case of link-1 failure, data is stored in the database through link-2. The data stored in the database is viewed by the user with the help of any one of the followings

1. An open source SCADA (SCADABR)
2. A PHP based web application
3. Android application

It is also possible to control (open/close) the actuator from the above-mentioned interfaces. Normally, Open/close command reaches the actuator primarily through link-1. In case of link-1 failure, it reaches through link-2. Command reaches the actuator through a database server in both the cases.

The project involves following works

1. Storing level data and actuator status in the database through GSM
2. Controlling of the actuator through GSM
3. Providing redundancy in communication
4. Creating a PHP based web application in an existing Apache web server.
5. Configuring an open source SCADA system to display the level database well as to control the actuator
6. Creating an android application to display the level data as well as to control the actuator

All these works have been done to mainly perform two tasks.

- Monitor level data/actuator status from user interfaces(operation-1)
- Sending open/close command from user interfaces(operation-2)

The user should not worry about physical layer of the communication, i.e. regardless of the availability of link-1, above two operations should be done smoothly. User interfaces can be SCADA, mobile app, or web app.

Operation-1: Operation-1 consists of the following process.

- i) Storing level data/actuator status in the database server through link-1/link2
- ii) Fetching the data stored in the database server by the user interfaces.

- Process-i through link-1 is done by the gateway (a python program is running in the gateway). Process-i through link-2 is done by the server (a python program is running in the server) installed in the process control lab.
- Process-ii is done by respective interfaces in their own ways, but they do the same job, i.e., reading the database.

Operation-2: Operation-2, which is more critical in nature, consists of the following process.

- Updating database by user interfaces
- Reading/toggling the data(which is updated by user interfaces) through link-1/link-2

If the data is read through link-1(a python program in the gateway), we call that command is given through primary link. If the data is read through link-2(a python program on the server), we call that command is given through secondary link.

For storing level data/ actuator status, we prefer link-1 over link-2. Link-2 is made active, only when link-1 fails. This is ensured by a separate python program running on the server.

1.3 system architecture

IIT Madras water network overview

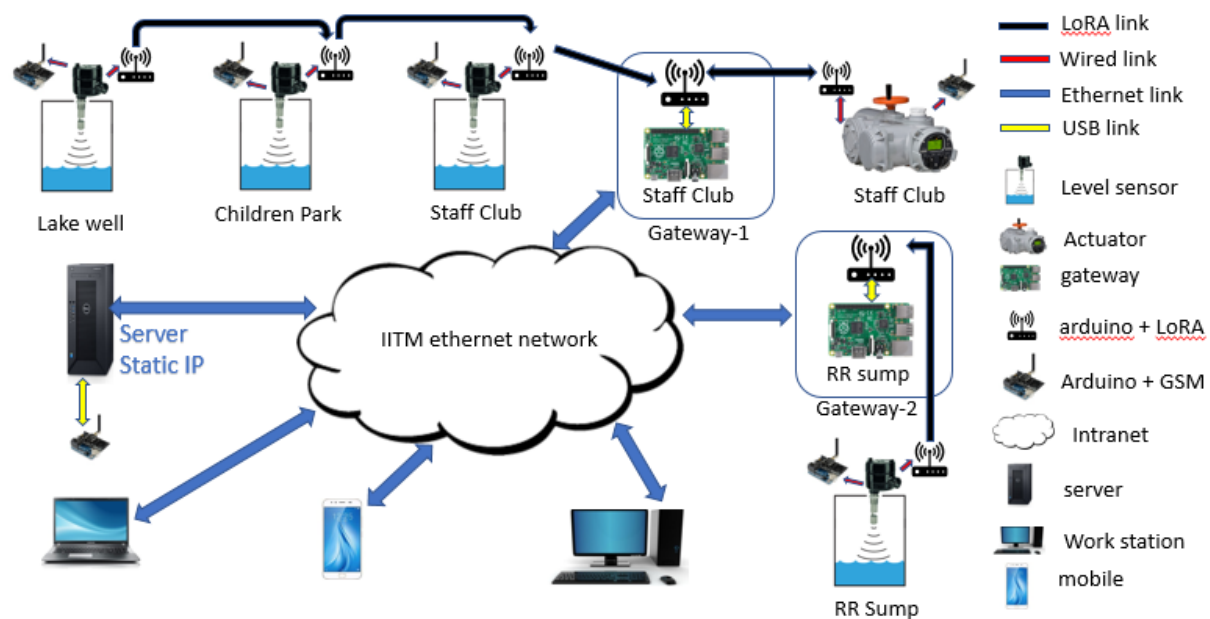


Figure-1.1 system Architecture

1.4 HARDWARE DESIGN

1.4.1 Remote Node

Water level in the reservoirs is usually measured by using float & board level gauge and dip sticks. The level gauge values are not accurate and require constant human effort. Improper readings can result in insufficient storage or reservoir overflow. Remote nodes are installed in the OHSR/UGSR to monitor water levels. It consists of Arduino MCU, GSM module (SIM-900), Ultrasonic sensor, LoRAModule. Arduino collects the water level in the reservoir every minute from

ultrasonic sensor and sends to gateway/server. Arduino Nano MCU is used, due to its small form factor and low power consumption.

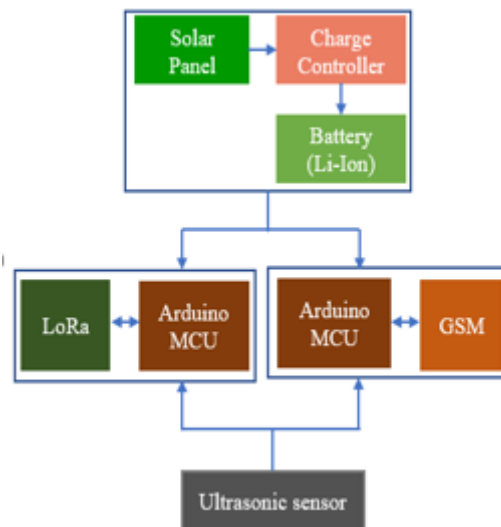


Figure 1.2. Block diagram of Remote node.

MaxBotix weather proof ultrasonic sensor (MB7060 XL-MaxSonar-WR) is used to measure water level in the reservoirs. The sensor and electronic devices are enclosed in IP65 box. GSM (Sim 900A module) is used as a redundant system. Figure 2 shows the block diagram of the developed module. Separate MCU is used for both LoRa and GSM. LoRa is connected to Arduino using serial peripheral interface (SPI). Power supply is designed to supply 5V to Arduino, ultrasonic sensor and GSM. Power for LoRa is taken from Arduino 3.3V output. Remote nodes take reading for every minute and send to gateway.

1.4.2 Actuator Node

The block diagram of the actuator node is shown in Figure 4. The actuator has local status indication (open/close) and button to operate the valve locally. It also has remote control option, i.e., it can be opened/closed externally using relay or PLC. As shown in Figure 5, the electric actuator is fixed on top of the valve using supports and a long stem is used to transfer the torque from actuator to valve. Potential free contacts are provided to read actuator status. Remote node reads the actuator status for every five seconds and sends to gateway using LoRa/to the server using GSM. Actuator status consists of Local/Remote mode, Fully open, Fully close. Based on the command received, Arduino can energize or de-energize the relay to open/close the actuator. These nodes are powered from actuator power supply, so batteries are not used. Power supply is designed to supply 5V to Arduino and GSM. Incase primary communication LoRa fails, GSM would be used

as a redundant system. To avoid single point failure, Separate MCUs are used for both LoRa and GSM. Actuator status and controls are connected to both MCUs. Both LoRa and GSM are initialized during starting. GSM is always in receive mode to save power.

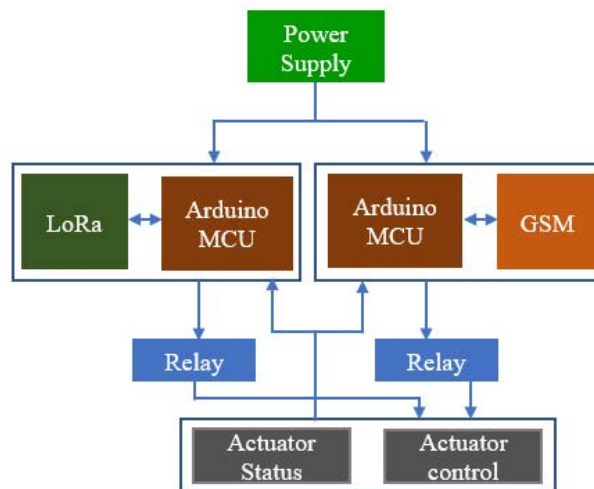


Figure 1.3. Block diagram of Actuator node. Figure 1.4. Electrical Actuator

1.4.3 Gateway Node

The gateway node collects the data from remote nodes and actuator nodes, processes the data and sends to centralized server. Raspberry Pi is used for data processing and internet connectivity. The cost of raspberry Pi based gateway is low (~75 USD). LoRa Modem is connected to Raspberry Pi using serial Peripheral interface (SPI). Raspberry Pi is connected to centralized server using local area network (LAN). The gateway pushes the data to central server whenever it receives data from any of the remote/actuator node. It receives command from the centralized server and transmits to the actuator node to open/close the actuator. Gateway LoRa is always in receive mode. Python is used for programming Raspberry Pi.

CHAPTER-2

Device interface and data storage

2.1 Use of python program for interfacing devices and communication redundancy

Python has been used as the scripting language to put the data into a database field, after collecting it from Arduino. Sometimes, it is also used to send the data to Arduino. It is running in the gateways as an interface to the field devices, and in the server to ensure redundant communication. Python has the following advantage.

- Python has extensive support libraries.
- It is open source
- It is easy to learn, and a lot of support is available
- It has a user-friendly data structure
- It has good speed.

Total four python programs have been used, two in the gateways, and two in the server. For the sake of simplicity, we define the python programs as Python-S1, Python-S2, Python-G1, Python-G2.

Python-S1: It is running on the server. It communicates with an Arduino which is connected to a GSM modem through a serial port. This program is doing the following job.

It establishes serial communication with the Arduino

- It gives a command to the Arduino through serial port-2 to send SMS to sensors and actuator, when link-1 fails.
- It updates level data and actuator position/mode(local/remote) in the database field. Incoming SMS received by GSM modem (from level sensors and actuator) are sent to the Arduino by the modem through serial port-1. The Arduino places the SMS into the serial port-2(Physically, it is aUSB port,

Arduino has inbuilt USB to serial converter). Python reads the SMS from serial port-2, extracts the data(level or actuator status) from the SMS, and writes to the database, i.e, updates the data in the database fields.

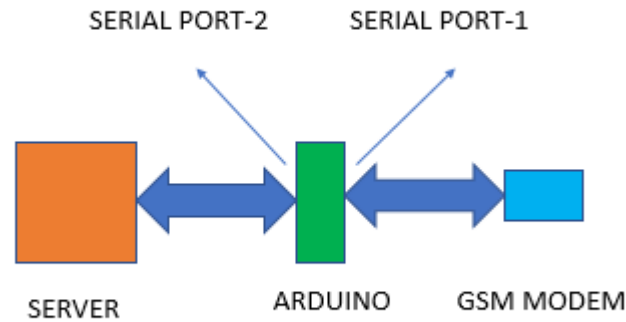


Figure-2.1: Interfacing of GSM modem with server

Python-S2: It is also running on the server. It determines the communication availability of sensors and actuator with respective gateways and updates in the database field.

Python-G1: This program is running in the gateway installed in the staff club. This program is doing the following job.

- It collects a data string which consists of level data and LoRA address (installed at lake-well, children-park, and staff club) from the Arduino (Arduino gets these data through LoRA) through serial, and updates the levels in the database fields.

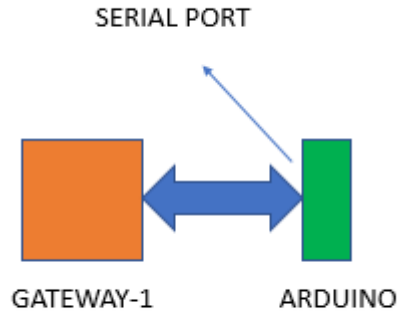


Figure-2.2 Interfacing of Arduino with gateway-1

- The data string collected by this program consists of loRA address (installed at lake-well, children-park, and staff club). Therefore, this program can understand which LoRAs are reporting to the Arduino. If the loRA connected to a particular sensor reports to the Arduino, this program generates a random value for that sensor and updates the random values in the database.
- It collects a data string which consists of actuator position and mode(local/remote) from the Arduino (Arduino gets these data through loRA) through the same serial port, and updates actuator status/mode in the database fields.
- The data string collected by this program consists of actuator address. Therefore, this program can understand if the actuator is reporting to the Arduino through loRA. If the actuator reports to the Arduino, the program generates a random value for the actuator and updates the random value in the database field.
- It also sends open/close a command to the Arduino through Serial port. (Arduino sends a further command to the actuator through loRA)

Python-G2: This program is running in the gateway installed in the main sump. This program is mainly for the following job.

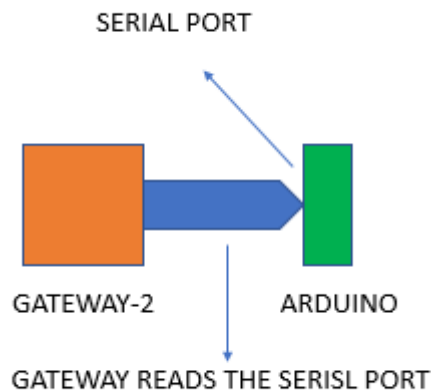


Figure-2.3 Interfacing of Arduino with gateway-1

- It collects a data string which consists of level data and loRA address (installed at main sump) from the Arduino (Arduino gets these data through loRA) through serial port , and updates the levels in the database fields.
- The data string collected by this program consists of loRA address of the main sump. Therefore, this program can understand if the loRA is reporting to the Arduino. If the loRA reports to the Arduino, the program generates a random value for the main sump sensor and updates the random value in the database field.

2.2 Data storage

For storing level data, actuator status, and for sending an open/ close command to the actuator a database has been used instead of using any file, as the database has better security, flexibility, quick access and better connectivity with php, python, and java. Data sent by the level sensors through link-1 are stored in the same table where the data sent by the level sensors through GSM are stored. As MySQL is a powerful, free open-source database management system that has been around for years and it is very stable and has a big community that

helps maintain, debug and upgrade it, MySQL has been used here. The advantage of using MySQL is

- MySQL is very fast, reliable, and easy to use
- MySQL uses standard SQL
- MySQL compiles on a number of platforms
- MySQL supports php, python and the open source SCADA software (SCADABR) and java. We need to connect to MySQL with java for android application and SCADABR, with python for sending open/close command to the actuator
- It is very easy to access any data with powerful SQL query.

The data in a MySQL database are stored in tables. A table is a collection of related data, and it consists of columns and rows. A separate database called “scada” has been created which is used by php, python, and java. Mobile app(java), web app and open source “scada” software(SCADABR) use the same database.

The database “scada” has 6 tables (AI, DI, DO, Login, command, and gsm_test). Each table has 3 columns (id, name, and value). Id is the primary key which is unique and is used while querying. Three different types of operations (Read, update and insert) have been done with this database “scada”. Before doing any operation by any program, the database is connected to the program, the operation is performed, and then the database is closed. Remote access to MySQL has been enabled so that the gateway can communicate to MySQL. For connecting to the database, the followings are used.

Username: root

Password: gowsalya

Database name: scada

Hostname: 10.21.160.201

Details of each table are given below.

#	id	name	value
1	1	lake_well	9.77690288714
2	2	children_park	8.39107611549
3	3	stuff_club	12.8031496063
4	4	RR_sump	8.03
5	5	lake_well_random	477
6	6	children_park_random	444
7	7	stuff_club_random	378
8	8	RR_sump_random	190
9	9	Actuator_random_value	173
*	HULL	HULL	HULL

Figure2.4 Table-1-AI

The table AI has 9 id fields, 9 name fields, and 9 value fields. id and name fields remain constant. For simplicity, we define value field corresponding to id $i = \text{value } i$.

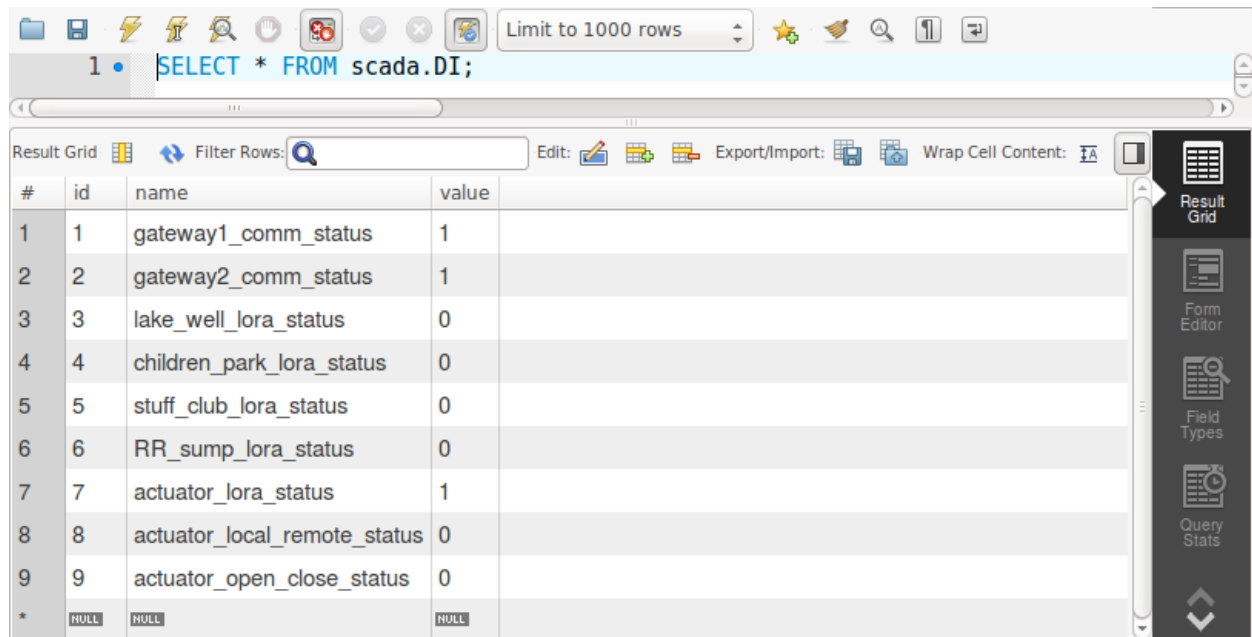
Value-1 to value3: These fields are for storing level data for lake-well, children park, and staff club. Value- i is updated by Python-G1 program when loRA- i is available (Link-1 is available) and is updated by Python-S1 program when loRA- i is not available, i.e. when GSM is activated. (i belongs to $\{1,2,3\}$). These fields are also read by php program, android app, and scadaBR to display on user interfaces.

Value-4: This field is for storing level data for the mainsump(RR sump). Value-4 is updated by Python-G2 program when link-1 is available and is updated by Python-S1 program when link-1 fails, i.e. when GSM is activated. value-4 is also read by php program, android app, and scadaBR to display on user interfaces.

Value-5 to value7: These fields are for storing random values corresponding to sensors installed at lake-well, children park, and staff club. These fields are updated by Python-G-1 and are read by Python-S-2.

Value-8: This field is for storing a random value corresponding to sensor installed at the main sump. This field is updated by Python-G-2 and is read by the Python-S-2 program.

Value-9: This field is for storing a random value corresponding the actuator. This field is updated by Python-G-1 and is read by the Python-S-2 program.



The screenshot shows a database query result in a web interface. The query is `SELECT * FROM scada.DI;`. The result is displayed as a table with 4 columns: #, id, name, and value. The table contains 9 rows of data, with the last row being a NULL row. The right sidebar shows various tool icons like Result Grid, Form Editor, Field Types, and Query Stats.

#	id	name	value
1	1	gateway1_comm_status	1
2	2	gateway2_comm_status	1
3	3	lake_well_lora_status	0
4	4	children_park_lora_status	0
5	5	stuff_club_lora_status	0
6	6	RR_sump_lora_status	0
7	7	actuator_lora_status	1
8	8	actuator_local_remote_status	0
9	9	actuator_open_close_status	0
*	NULL	NULL	NULL

Figure 2.5: Table-2-DI

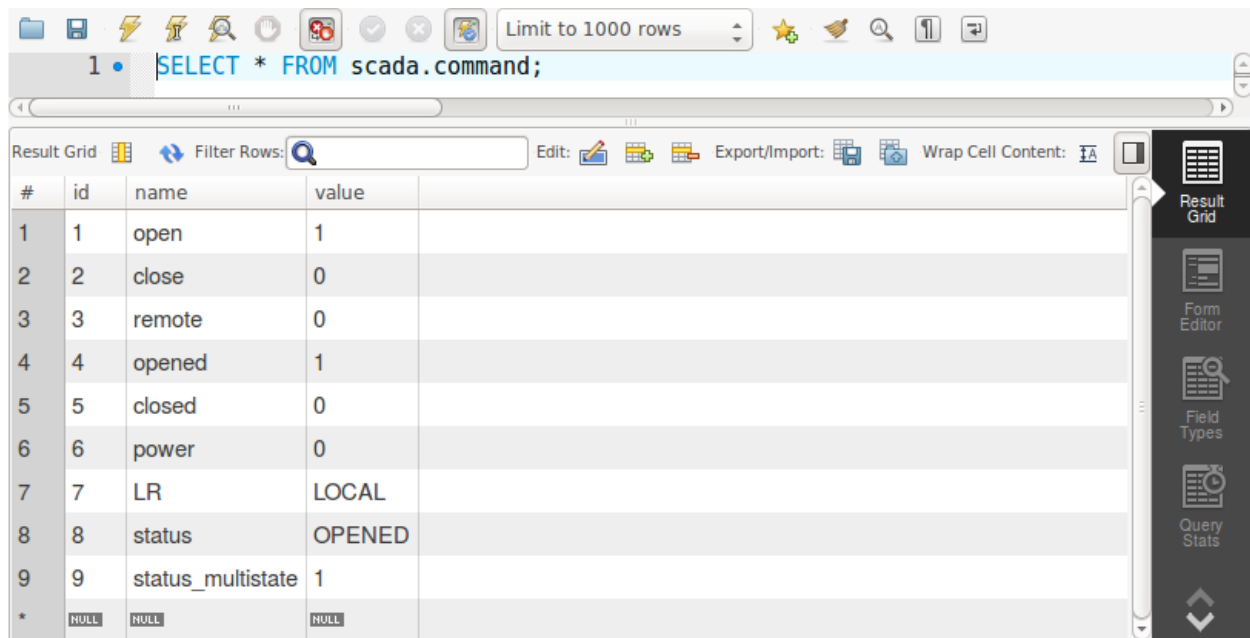
The table DI has 9 id fields, 9 name fields, and 9 value fields. Id and name fields remain constant. Value fields corresponding to id-1 and 2 have not been used.

Value-3 to value-7: These fields are used for storing link-1 availability status. These fields are updated by python-S2.Value- (3+i) of this table is set to 1 by Python-S2if value- (5+i) of table AI gets updated.(i belongs to {0,1,2,3,4}).

Example: value-6 of table AI updates =>value-4 of table DI=1

So, Python S-2 ensures that if the random value corresponding to a sensor is coming from the gateway, then link-1 status for that sensor is ok.

This table is not used by php program, android app, and scadaBR.



The screenshot shows a database query interface. At the top, a SQL query is entered: `SELECT * FROM scada.command;`. Below the query, a table grid displays the results. The table has four columns: #, id, name, and value. The results are as follows:

#	id	name	value
1	1	open	1
2	2	close	0
3	3	remote	0
4	4	opened	1
5	5	closed	0
6	6	power	0
7	7	LR	LOCAL
8	8	status	OPENED
9	9	status_multistate	1
*	NULL	NULL	NULL

Figure 2.6: Table-3-command

The table command has 9 id fields, 9 name fields, and 9 value fields. Id and name fields remain constant. value-1 and value-2 have not been used.

Value-3 to value-6 are for storing actuator status. These fields are updated by Python-G1 when the IoT module (connected to the actuator) communicates to the gateway. These fields are also updated by Python S-1 when link-1 fails, i.e. GSM is activated. These fields are also read by PHP, or Android app, or ScadaBR to display on the user interfaces.

Value-7, 8 and 9 are the special fields used by the SCADA.

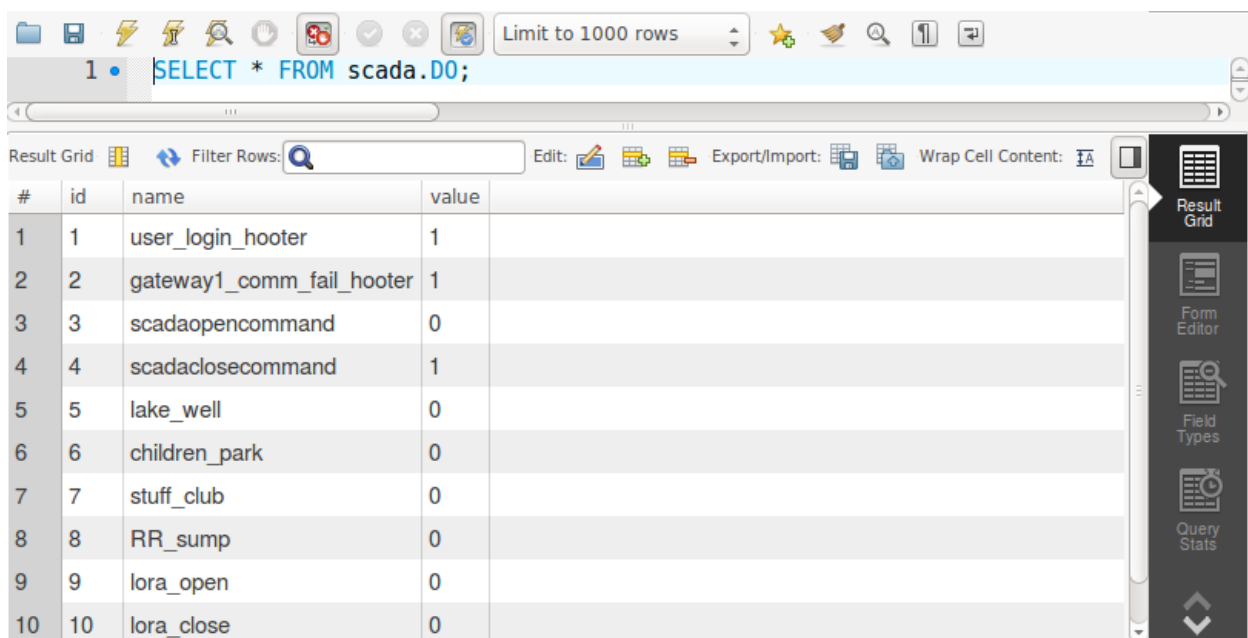
Value-3: If the actuator is in remote mode, Python-G1 updates this field to 1. If the actuator is in local mode, Python-G1 updates this field to 0. This field is also updated through GSM, if link-1 fails. This field is read by php, or android app, or ScadaBR to display status on the interfaces.

Value-4: If the actuator is fully opened, Python-G1 updates this field to 1. If the actuator is in an intermediate position or in fully closed condition, Python-G1 updates this field to 0. This field is also updated through GSM, if link-1 fails.

This field is read by PHP, or Androidapp, or ScadaBR to display status on the interfaces.

Value-5: If the actuator is fully closed, Python-G1 updates this field to 1. If the actuator is in an intermediate position or in fully opened condition, Python-G1 updates this field to 0. This field is also updated through GSM, if link-1 fails. This field is read by PHP, or Androidapp, or ScadaBR to display status on the interfaces.

Value-6: If the power supply to the Arduino is available, Python-G1 updates this field to 1. It is 0 otherwise



#	id	name	value
1	1	user_login_hooter	1
2	2	gateway1_comm_fail_hooter	1
3	3	scadaopencommand	0
4	4	scadaclosecommand	1
5	5	lake_well	0
6	6	children_park	0
7	7	stuff_club	0
8	8	RR_sump	0
9	9	lora_open	0
10	10	lora_close	0

Figure 2.7: Table-4-DO

The table “DO” has 9 id fields, 9 name fields, and 9 value fields. Id and name fields remain constant.

Value-1 and value2: Not used.

Value-3: This field is updated to 1 by PHP when an open command is given by the user from mobile app/web application. If the command is given from SCADA, the field is updated by java which is installed in the virtual guest machine (ubuntu-14.04 LTS). This field is read by Python-S1. If it is 1, Python-S1 updates it to 0, and does any one of the followings depending on link-1 availability.

1. Updates value-9 to 1 (when link-1 is available)
2. Sends open command to the actuator through GSM (when link-1 fails)

This field remains 0 until a further open command is given.

Value-4: updated to 1 by PHP when a close command is given by the user from mobileapp or web application. If the command is given from SCADA, the field is updated by java. This field is read by Python-S1. If it is 1, Python-S1 updates it to 0, and does any one of the followings depending on link-1 availability.

1. Updates value-1 to 1 (when link-1 is available)
2. Sends close command to the actuator through GSM (when link-1 fails)

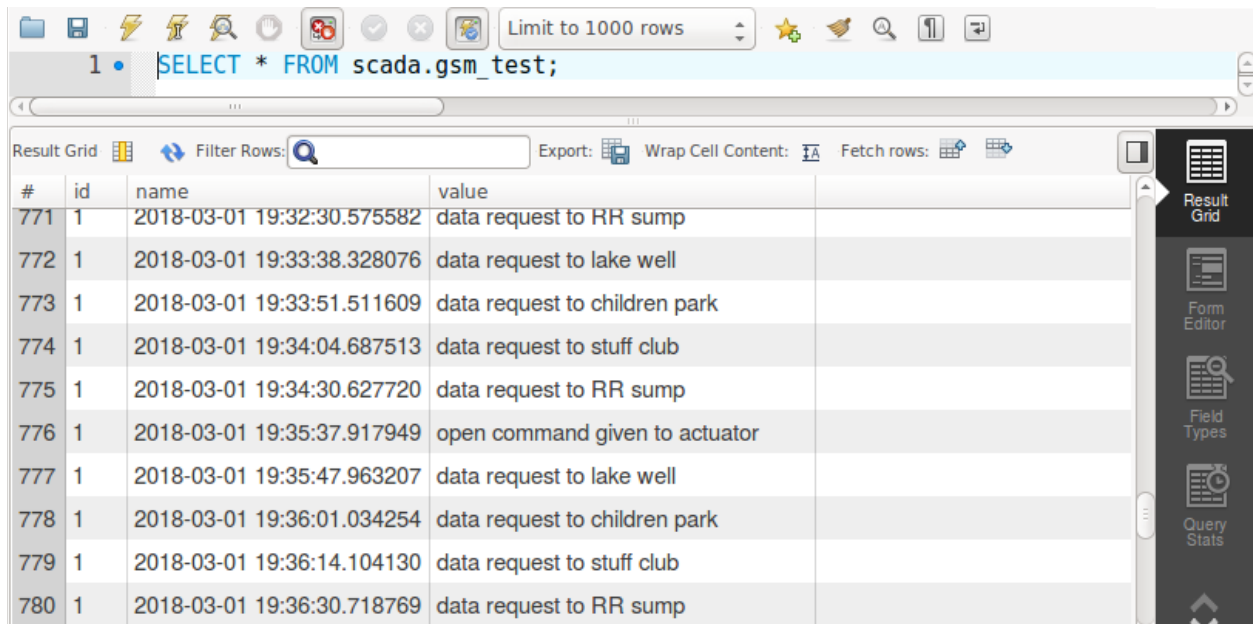
This field remains 0 until a further close command is given.

Value-8: We call this field as manual data request field for RR sump. Regardless of availability of primary link, it is possible to get level data manually through GSM from level sensors. This feature is made available only in the SCADA. A manual command must be given from the SCADA to get the data manually through GSM. Value-8 becomes-1, when such command is given. This field is also read by Python-S1. If it is 1, Python-S1 updates it to 0(initial state)

Value-5 to value-7: These fields have been used to get level data manually through GSM from lake-well, children Park, and staff club respectively.

Value-9: If value-3 is 1, and the actuator is communicating through primary link(link-1), Python-S2 updates value-9 to 1 and value-3 to 0. If value-9 is 1, Python-G1 updates it to 0 after sending an open command to the actuator through loRA.

Value-10: If value-4 is 1, and the actuator is communicating through primary link, Python-S2 updates value-10 to 1 and value-3 to 0. If value-10 is 1, Python-G1 updates it to 0 after sending a close command to the actuator through loRA.



1 • `SELECT * FROM scada.gsm_test;`

Limit to 1000 rows

Result Grid

#	id	name	value
771	1	2018-03-01 19:32:30.575582	data request to RR sump
772	1	2018-03-01 19:33:38.328076	data request to lake well
773	1	2018-03-01 19:33:51.511609	data request to children park
774	1	2018-03-01 19:34:04.687513	data request to stuff club
775	1	2018-03-01 19:34:30.627720	data request to RR sump
776	1	2018-03-01 19:35:37.917949	open command given to actuator
777	1	2018-03-01 19:35:47.963207	data request to lake well
778	1	2018-03-01 19:36:01.034254	data request to children park
779	1	2018-03-01 19:36:14.104130	data request to stuff club
780	1	2018-03-01 19:36:30.718769	data request to RR sump

Export: Wrap Cell Content: Fetch rows:

Result Grid
Form Editor
Field Types
Query Stats

Figure 2.8: Table-4-gsm_test

The table “gsm_test” is used to insert a new row whenever an event occurs. The following events along with the timestamp are stored.

- When the open/close command is sent through loRA
- When the open/close command is sent through GSM
- When manual data requests to the sensors are sent

Therefore, this table stores historical events.

1 • `SELECT * FROM scada.Login;`

Limit to 1000 rows

Result Grid

#	user_id	username	password
1	1	admin	123456
2	2	root	123456
*	NULL	NULL	NULL

Filter Rows:

Edit: Export/Import: Wrap Cell Content:

Result Grid
Form Editor
Field Types
Query Stats

Figure 2.9: Table-4-Login

The table “Login” has 2 id fields, 2 username fields, and 2 passwords fields. All fields remain constant. This table is used to store valid username and password. These username and password fields are read by php. If these usernames and passwords are entered by the user, he can login to the web application. If the user enters the username and password as “admin” and “123456” respectively, he can control the actuator, as well as can monitor the level parameters. If the user enters the username and password as “root” and “123456” respectively, he can only monitor the level parameters.

Chapter-3

Design of user interface

3.1 Design of web application

Web application has been designed by HTML, CSS, PHP, and ajax.

HTML (the Hypertext Markup Language) and CSS (Cascading Style Sheets) are two of the core technologies for building Web pages. HTML provides the structure of the page, CSS the (visual and aural) layout. Along with php scripting, HTML and CSS are the basis of building Web pages and Web Applications.

HTML is the language for describing the structure of Web pages. HTML gives headings, text, tables, lists, photos, etc of a webpage. It also retrieves online information via hypertext links, at the click of a button. HTML is used to describe the structure of pages using markup. The elements of the language label pieces of content such as "paragraph," "list," "table," and so on.

CSS is the language for describing the presentation of Web pages, including colors, layout, and fonts. It allows one to adapt the presentation to different types of devices, such as large screens, small screens, or printers. CSS is independent of HTML and can be used with any XML-based markup language. The separation of HTML from CSS makes it easier to maintain sites, share style sheets across pages, and tailor pages to different environments. This is referred to as the separation of content from presentation.

As usual, in this web application, HTML provides contents of the webpage, and CSS gives the representation including content location. The static portion has been designed using HTML and CSS.

PHP which is an acronym for "PHP: Hypertext Preprocessor" is a widely-used, open source scripting language. PHP scripts are executed on the server having static IP 10.21.160.201. It is free to download and use. Biggest blogging system on the web, WordPress and largest social network, Facebook have been designed with PHP. PHP files can contain text, HTML, CSS, JavaScript, and PHP code. PHP code are executed on the server, and the result is returned to the browser as plain HTML (a webpage). PHP files have extension ".php".

PHP can do the followings.

- PHP can generate dynamic page content
- PHP can create, open, read, write, delete, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in the database
- PHP can be used to control user access
- PHP can encrypt data
- PHP file contains php logic (<?php “logic here”?>), HTML, CSS, Javascript.

Advantages of using php are:

- PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP supports a wide range of databases
- PHP is fully open source.
- PHP is easy to learn and runs efficiently on the server side

Apache is the web server which accepts http request from the client and gives a webpage as a response. As, PHP files are kept on Server, it is a server-side component. PHP Interpreter interprets PHP language and executes instructions as per code. It does not need compilation.

LAMP stack has been used to develop this web application. LAMP is an open source Web development platform that uses Linux as the operating system, Apache as the Web server, MySQL as the relational database management system and PHP as the object-oriented scripting language. Because the platform has four layers, LAMP is sometimes referred to as a LAMP stack.

An Architecture Diagram showing how each of the components in a system is connected to each other and how the data flowing between is given below.

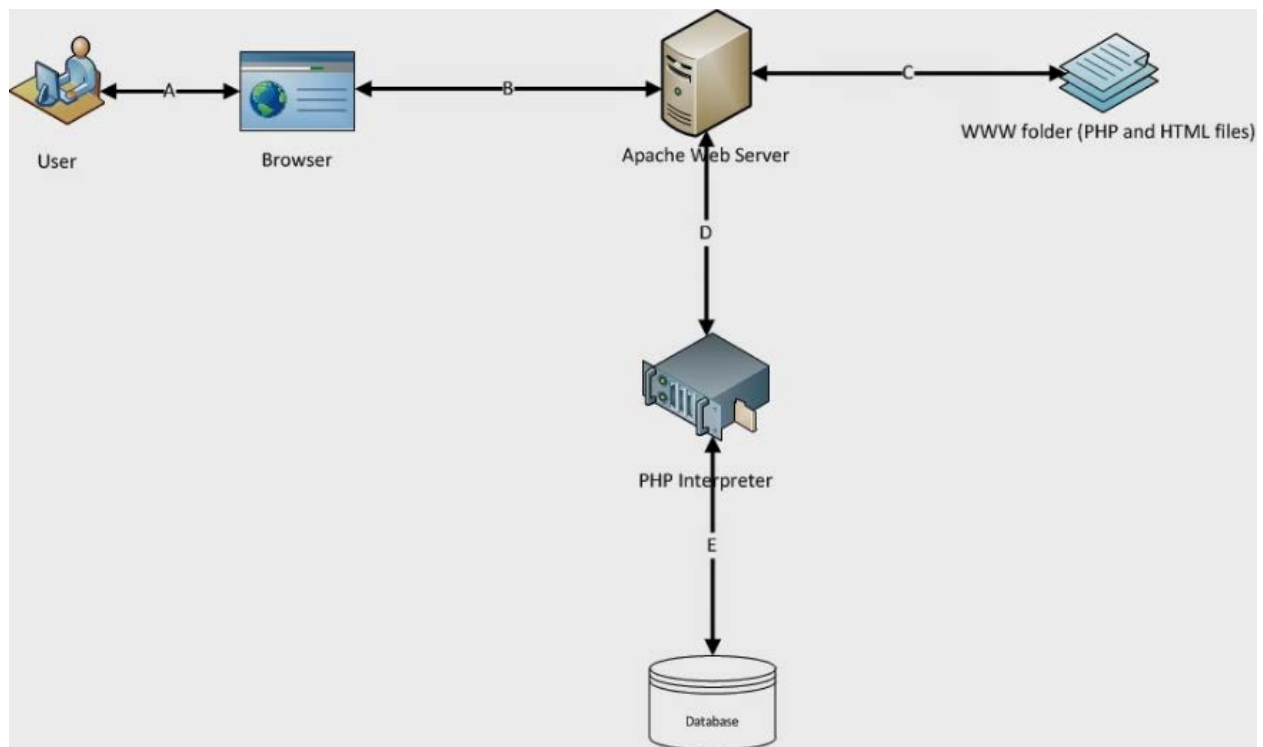


Figure-3.1 Architecture of web application

At first, any user accesses the website through a browser, i.e. user enters the URL of the website (like: 10.21.160.201/test.html) in the browser. The page request on browser first reaches to the Web Server (Apache). Web server collects that requested page (HTML or PHP or Image file etc) from its document root (www folder). Now, if it is a static element like HTML, CSS, image file or JavaScript file, then Apache sends it directly to the browser, and Browser will render it to the user on the screen.

If the user enters link containing PHP file (like 10.21.160.201/Login2.php) on the address bar of the browser, Apache sends the content of the file to PHP Interpreter. PHP interpreter interprets the PHP code and executes it. If DB operation is written in the PHP file, the interpreter performs the same.

PHP Interpreter generates output (if the PHP code is to generate any output) and sends to Apache. Apache sends that content to the browser, and the browser renders it to users' screen.

All static components like HTML, CSS files, ImageFiles, Java Scripts etc don't need an interpreter. Our web browsers are built to render them and display on screen.

properly. Only if the requested page is a PHP page Apache will send it to PHP interpreter to get it translated and executed.

In this web application, PHP programs have been used to perform 4 tasks mainly. Some of the PHP files return a webpage, some of them update a particular field in the database, some of them call another PHP program, and some of them return a particular portion of a webpage, instead of returning the whole webpage (using ajax technology). The link to access the web application is <http://10.21.160.201/Login2.php>.

Security of the application

If "<http://10.21.160.201/Login2.php>" is entered in the address bar of any browser, the page request first reaches to the Apache server (IP: 10.21.160.201). Apache sends the content of Login2.php file to PHP Interpreter. PHP interpreter interprets the PHP code and executes it. PHP Interpreter generates output and sends to Apache. Apache sends that content to the browser, and the browser renders it to users' screen. As a result, user gets the following webpage.

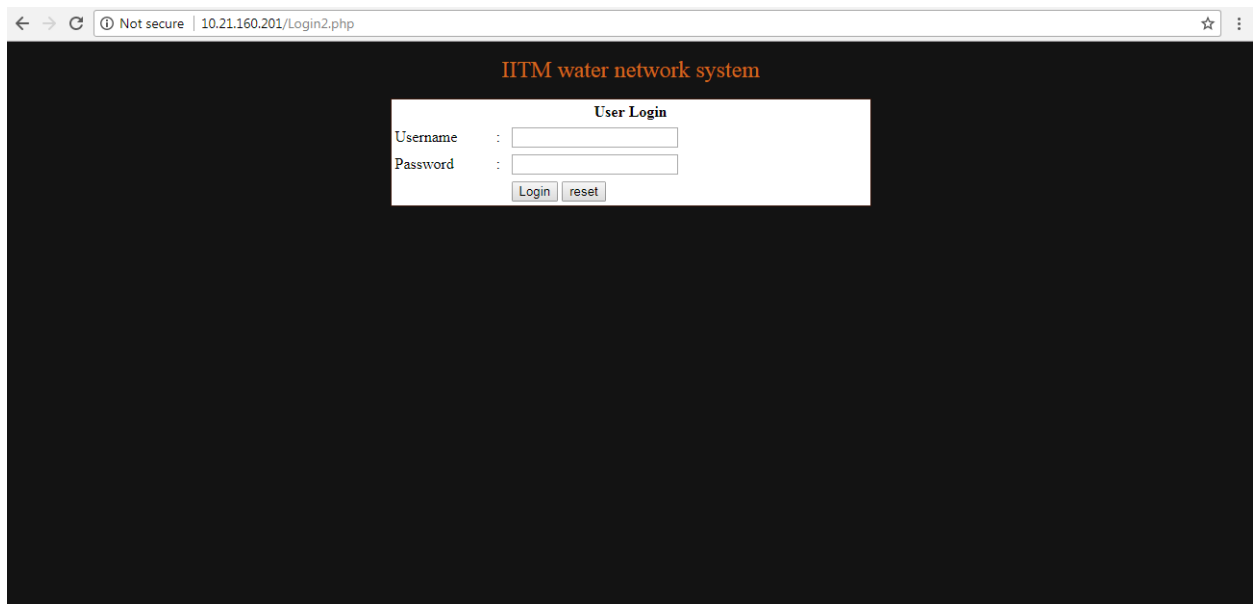


Figure-3.2: webpage returned by apache on execution of Login2.php

The webpage returned by apache has a username and password textbox field, Login and reset button.

A html form for user login has been created using http POST method. There are two methods to create a html form-GET and POST. Both GET and POST create an array with key-value pair, where keys are the names of the form controls and values are the input data from the user.

Both GET and POST are treated as `$_GET` and `$_POST`. These are global, which means that they are always accessible, regardless of scope. It can be accessed from any function or file.

`$_GET` is an array of variables passed to the current script via the URL parameters.

`$_POST` is an array of variables passed to the current script via the HTTP POST method.

Information sent from a form with the GET method is visible to everyone (all variable names and values are displayed in the URL). GET may be used for sending non-sensitive data.

Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send. As client needs to enter a password which is sensitive, http POST method has been used here.

“username” and “password” are the id of username and password textbox field, i.e whatever the username is entered by the user, it gets automatically assigned to the variable `$_POST['username']` (This is a property of POST method). Similarly, the password entered by the user gets assigned to `$_POST['password']`.

Therefore,

`$_POST['username']=username entered by user`

`$_POST['password'].=password entered by user`

Similarly, “submit” and “reset” are the name of the Login and reset button respectively. Unlike id is used for text-field, name is used for button. The variable `$_POST['submit']` remains “NULL” until Login button is pressed by the user. After pressing Login button, `$_POST['submit']` gets a value automatically.

The `isset()` function is used to check whether a variable is set or not. If a variable is already unset with `unset()` function, it will no longer be set. The `isset()` function return false if testing variable contains a NULL value. `Unset()` function has not been used here.

`isset($_POST['submit'])=TRUE` if login button is pressed.

`isset($_POST['submit'])=FALSE` if login button is not pressed.

When login button is pressed by the user, one PHP file gets called to connects to the database “scada” of MySQL server if no connection error is there.

We have used the concept of session in our web application.

A session is a way to store information (in variables) to be used across multiple pages. Unlike a cookie, the information is not stored on the user's computer. A session here is working with this application, logging in it, doing something, and then logging out of it. Php knows the user logged in to do something. It knows when user logs in the application and when ends.

Session variables stores user information to be used across multiple pages. Here, session variables last until the user logs out of the application.

So, Session variables hold information about one single user and are available to all pages in our application.

A session is started with the `session_start()` function.

Session variables are set with the PHP global variable: `$_SESSION`. `$_SESSION['variable1']`, `$_SESSION['variable2']` etc are used as global variable and can be accessed by any PHP file inside the folder.

If user presses the login after entering the username and password in the textbox fields, PHP connects the database, and start a session. Then, whatever the

username and password are entered by the user get assigned the variable \$username and \$password respectively.

Then PHP search the Login table of the database “scada” for \$username and \$password.

If the username and the password entered by the user matches with the username and password stored in the database, a global variable \$SESSION[login_user] is created and is assigned by the username entered by the user. Then, a new PHP program “watert.php”(<http://10.21.160.201/watert.php>) gets called. So, the PHP program “watert.php” gets called if and only if \$SESSION[login_user] is not NULL.

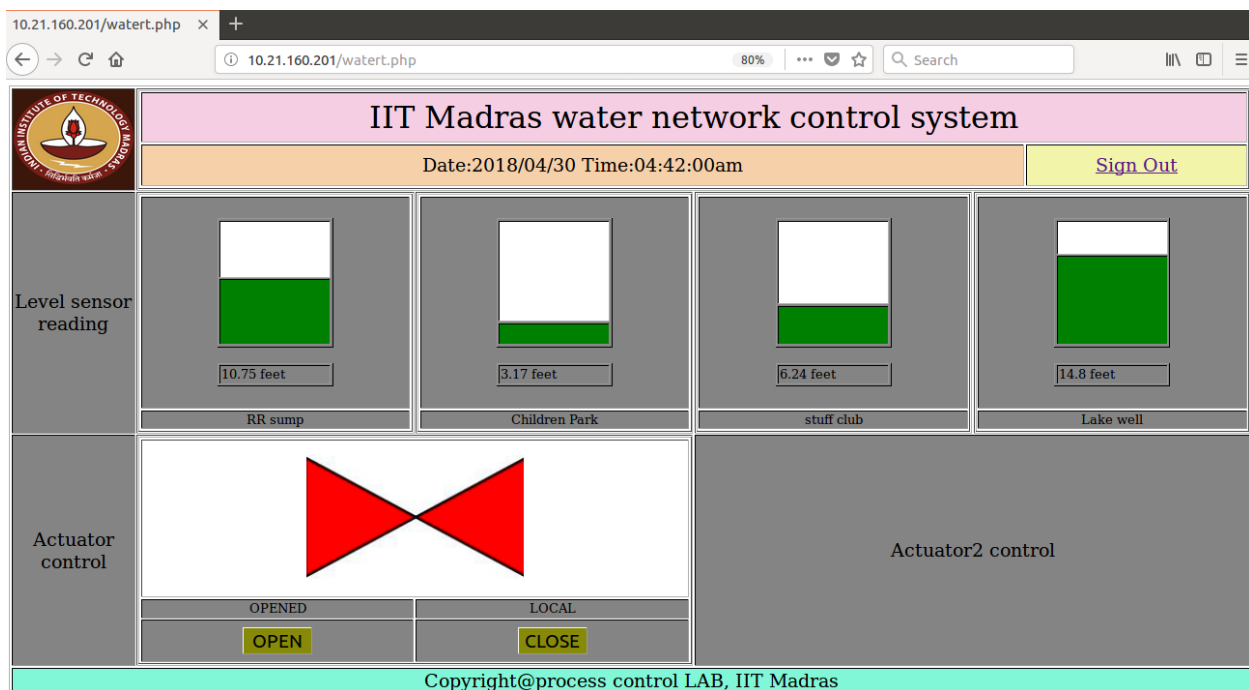


Figure-3.3: webpage returned by apache on execution of Login2.php

If any user enters the link <http://10.21.160.201/watert.php> without logging in, he is redirected to <http://10.21.160.201/Login2.php> automatically ensuring that it is

not possible to monitor/control without logging in. Thus using POST method, session variable, and database connectivity, the application is made secured.

Ajax technology has been used to create this web application to ensure the followings.

- Manual page refresh is not required by the user to get updated data
- Processing is same for all browser types because JavaScript is used.
- Using ajax it is possible to develop faster and more interactive web applications.
- Ajax based application use less server bandwidth, because no need to reload complete page..

Updating data:

Level data is read from databasewatert.phpfile, and is displayed on thisweb page.

An empty HTML table (green color) with variable height indicates the water level of the tank on the basis of dynamic level data obtained from the database.

Actuator mode(local/remote), and position(opened/closed) are also read from the database and displayed on this page. A red colored image (present in www directory) for the valve is called by the phpfile, when actuator status is open in the database. The green colored image is used for displaying closed condition. Date and time also get updated on the top of the page. Updating of level data, HTML table, and time are done with ajax.

Open/close command: When open/close button is pressed, open/close command is reached to the actuator either through the gateway (primary link), or through GSM(when primary link fails).

3.2 Design of mobile application

Like the web application, a mobile application has also been developed for monitoring/controlling purpose. The mobile application reads the same data which are read by the web application. As android application cant read data from MySQL database directly, a web service is created, which acts as an interface between android application, and MySQL database.

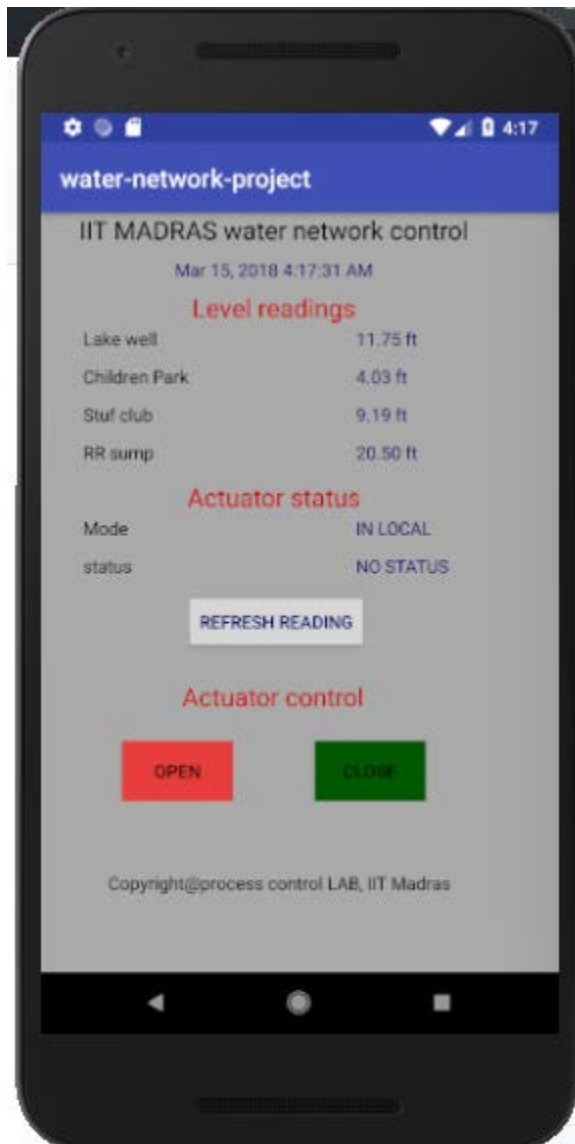


Figure-3.4: android app

I have integrated PHP and MYSQL with android application. This can be done, as I created a webserver which is present on the intranet, and I wanted to access its data on an android application.

MYSQL is used as a database on the webserver and PHP is used as a web service which will read data from database/update database.

3.2.1 open/close command: When open/close command is given, Android communicates with the PHP page, which acts as web services, with a variable which is shared by both PHP and java. The variable is first created by java, and is shared with php. When open command is given from android application, java assigns this variable to 1. As this variable is shared by php, php updates the database field to 1.

When this field is 1, command reaches to the actuator either through primary link or GSM.

3.2.2 Displaying data: When the button "refresh reading" is pressed from the mobile app, java requests the web services, i.e. PHP contacts MYSQL database and fetches all data and returns the results to us in JSON format. Java parses the JSON, and shows the reading on the application

Following steps have been followed to create the application:

- Existing database has been used as data sources
- Web services have been created with PHP
- Android Studio IDE has been used to create the application
- Modified src/MainActivity.java file to add Activity code. (button press)
- Modified layout XML file res/layout/activity_main.xml add any GUI component if required.
- Modified AndroidManifest.xml to add INTRANET permissions.

3.3 Installation and configuration of SCADA

SCADA is an acronym for Supervisory Control and Data Acquisition. SCADA systems serve as an interface between the operator and processes of the most varied types, such as industrial machines, automatic controllers and sensors of the most varied types. With SCADA systems, they are built from simple sensing and automation applications to the famous "Control Panels" in various industries.

A typical SCADA should offer communication drivers with equipment, a system for continuous data logging ("datalogger") and a graphical user interface known as "HMI" or Human Machine Interface. At the HMI, graphic elements such as buttons, icons and displays are represented, representing the actual process being monitored or controlled.

Among some of the most commonly used functions in SCADA systems are:

- Generation of charts and reports with the history of the process;
- Alarm detection and event logging in automated systems;
- Process control including remote sending of parameters and set-points, activation
- command of equipment;
- Use of scripting languages to develop automation logic

ScadaBR software is an open-source model with a free license.

The source code of the system is available, including being allowed to modify and re-distribute the software if necessary. ScadaBR is a cross-platform Java-based application, ie PCs running Windows, Linux and other operating systems and can run the software from an application server (Apache Tomcat being the default choice). When running the application, it can be accessed from an Internet browser, preferably Firefox or Chrome. The main interface of the ScadaBR is easy to use and already offers visualization of the variables, graphs, statistics, the configuration of the protocols, alarms, construction of screens type HMI and a series of configuration options.

After configuring communication protocols with the equipment and defining the variables(inputs and outputs, or "tags") of an automated application, it is possible to set up web operator interfaces using the browser itself.

SCADABR can be installed on ubuntu or windows. It requires javajdk, apache tomcat, and mysql database. Development of a software like SCADABR requires 2-3 years. I have only configured it.

3.3.1 Supproted datatypes

Five types of data are supported.

Binary (or Boolean) values can have only two states, referenced in the system with zero (0) and one (1) values. We can use converters to display binary values on any necessary labels, such as on / off.

Multiple States values have multiple distinct states. By this approach, the binary type is a particular case of a multiple state. Values are primitively represented as integers (eg 0, 1, 2, 7, ...), but as in binary values it is possible to assign labels to each value such as "opened/ closed / intermedddiate"

Numeric (or analog) values are decimal values represented in the system with afloating-point variable. Examples of numerical values can be cited: temperature, humidity, price and altitude. Text renderers can be used to determine the display of features such as number of decimal places, thousands separation (with periods or commas), display of suffixes (° F, kW / h, moles, etc.).

Alphanumeric values are simply strings, such as the "COMM FAIL/COMM OK".

Values in Images are binary representations of image data. They are stored in files in the file system of the host server (not in the database).

We have used binary(LOCAL/REMOTE status of actuator), neumatic(level data), multistate(actuator position-opened, closed and intermediate), and alphaneumeric (COMM OK/COMM FAIL) as our data types.

3.3.2 Dara source

Data sources are a fundamental part of the operation of this application. A data source is a "place" from which data is received. Virtually anything can be a data source, as long as the communication protocol is supported by the application.

Some examples: If we have a Modbus network accessible via RS232, RS485, TCP / IP or UDP / IP, It is possible to create a Modbus data source that will "poll" the network at a defined interval. If we have equipment or applications that can send data over HTTP, It is possible to start a data source HTTP receiver that will listen for incoming connections and send the data to the appropriate points. Values can be "polled" at defined intervals. Data can be read and updated in a SQL database external to the system. Data can be generated randomly or predictably using a Virtual data source. Data values received or collected by a data source are stored in data points. We have used MySQL database as our data-source.

3.3.3 Datapoints

The data point is a collection of associated historical values. For example, the particular point may be the temperature reading of a quarter, while another point could be the humidity reading of the same room. Points can also be control values, such as an indicator to turn an equipment on or off.

There are many attributes that are used to control the behavior of points. The logical separation of data source and data point depends on the communication protocol in question.

Data point attributes can also determine many other aspects of the point, such as its name, how it should be recorded (all data, only value changes, or none), how long to maintain the data, how to format the data for display, and how chart with the values. It is possible to set up data points with value detectors, which are used to detect conditions of interest in point values, such as whether the value has been too high for too long, too low, frequently changes, if not changes, etc.

Points can be arranged in a hierarchy, or tree, to simplify their management and display using the Hierarchy functionality.

We have used our datapoints as level data, actuator mode(local/remote), actuator position(opened/closed/intermediate) , communication availability(loRA OK/FAIL), and open/close command.

Events:

An event is the occurrence of a condition defined in the system. There are both system- defined and user-defined events. System-defined events include data

source operation errors, user logins, and system startup and shutdown. User-defined events include value detectors, scheduled events, and compound events that detect conditions on multiple points using logical arguments. There are also "automated events" that occur when users make changes (additions, modifications, and removals) that affect run-time objects, including data sources, data points, value detectors, scheduled events, compound events, and event handlers. Once an event has been detected, it is handled by handlers. An event handler is a user-defined behavior that must be performed when a particular event occurs, such as sending e-mail or "setting" the value to a set point.

3.3.4 Sound alarm

ScadaBR can play sounds when alarms are active. By default, sounds for alarms are executed for urgent, critical alarms, not for informational alarms, but the alarm sounds can be set individually. To enable private alarm sounds, valid files in mp3 format must be placed in the defined directory.

3.3.5 Graphics

ScadaBR contains a small graphics library that can be found in the graphics folder. Each sub-folder contains all the images in that image definition and an optional properties file called info.txt. This property file contains name / value pairs for the following attributes (all optional):

name: The name that will be used to describe the image in the user interface.

width: The width of the image. By default, all images are of the same size.

height: The height of the image. By default, all images are of the same size.

text.x: The text position relative to the left border of the image, in pixels. The default value is 5

text.y: The text position relative to the upper limit of the image, in pixels. The default value is 5.

Since an image definition is used in the views, the folder should not be renamed. The folder name is used internally as the image definition identifier. Image files are arranged alphabetically by name and are case sensitive. Name / value pairs are separated by '='. Lines beginning with '#' are considered comments.

'Thumbs.db' files are ignored. Compressed files (zip, gz, tar, etc) can not be used because image files must be accessed by the web server. The image settings are loaded at system startup, so any changes require restart.

3.3.7 Adding Data Sources

We have used data source as our existing database with jdbc driver, as java is used by SCADABR,

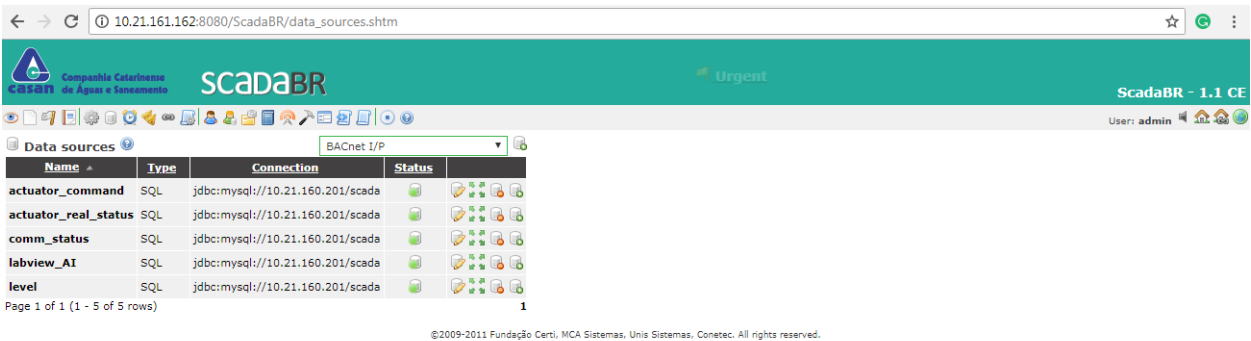


Figure-3.5 used data source

To use data-source, as MySQL database, SQL query is used. In the following example, data has been take from table AI(level data).

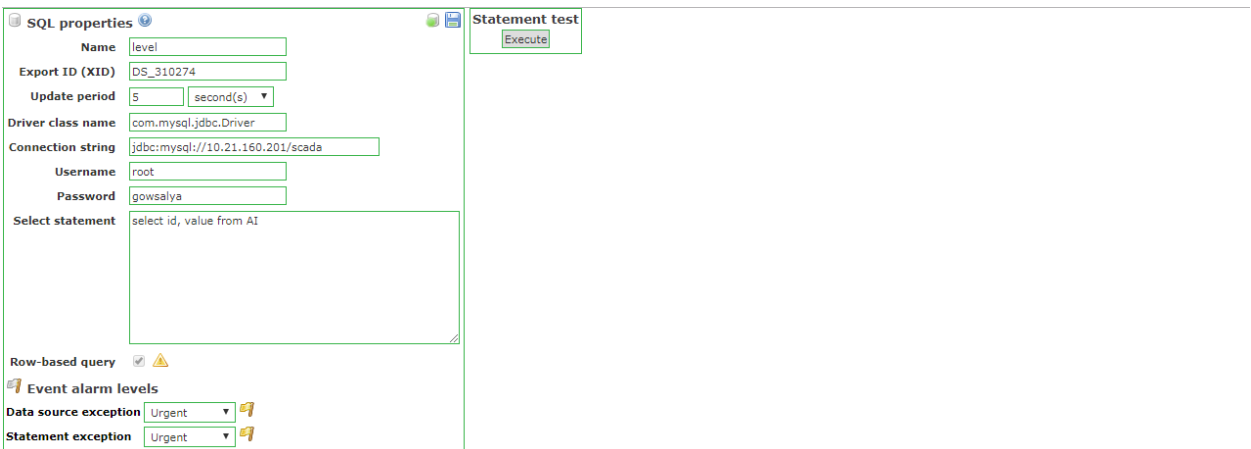
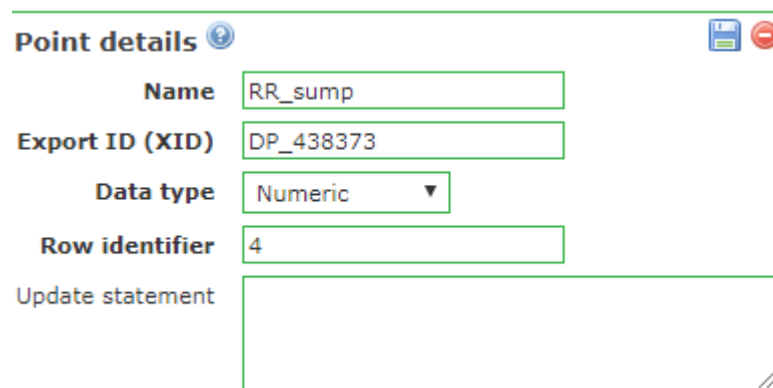


Figure-3.6used SQL query (Table-AI)

3.3.6 Adding Data Points

After selecting the data source(table here), a particular data is selected. This is called datapoint. We can think it as atag. In the following example, Level data corresponding to row-4 of AI table of database “scada” has been used as a datapoint.



Point details

Name RR_sump

Export ID (XID) DP_438373

Data type Numeric

Row identifier 4

Update statement

Figure-3.6 used SQL query (Table-AI)

3.3.7 Viewing the data through Watch List and Graphs

Once, datapoint is added, it is automatically added into the a list, called watch list

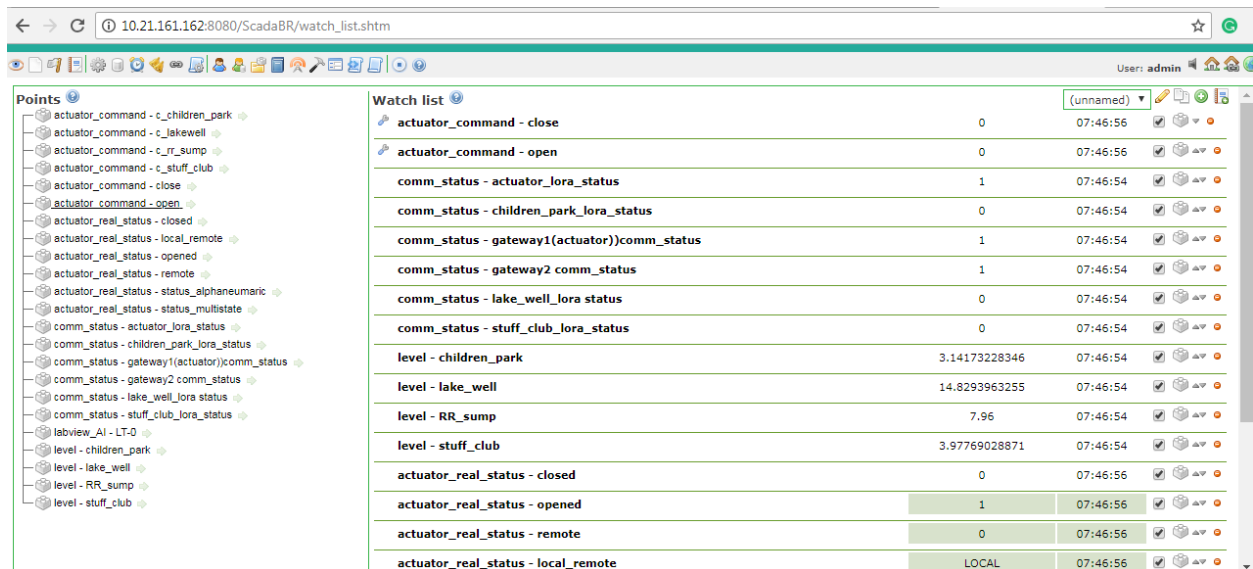


Figure-3.7 Watch list

It is possible to view graph(selecting time-from, and time upto) for a particular data point, after clicking on the point on the watch-list.

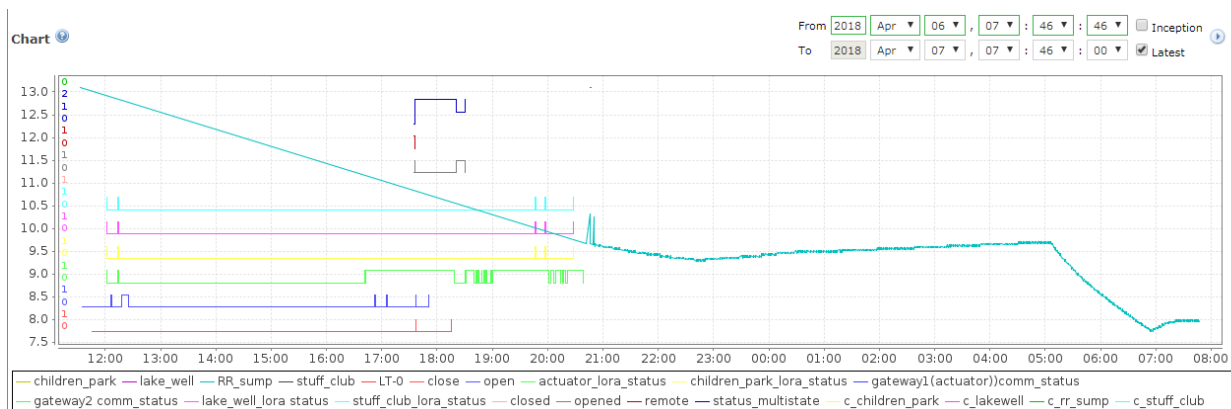


Figure-3.8 Graph of datapoints from watch-list

To create more elaborate visualizations of the data, we can construct "Graphical Representations". In the main menu, After choosing the Graphical representation option, we have to click on new Representation then to choose a name for our first picture. Then, we have to select an image and to upload that. Then we have to assign a datapoint to that image.

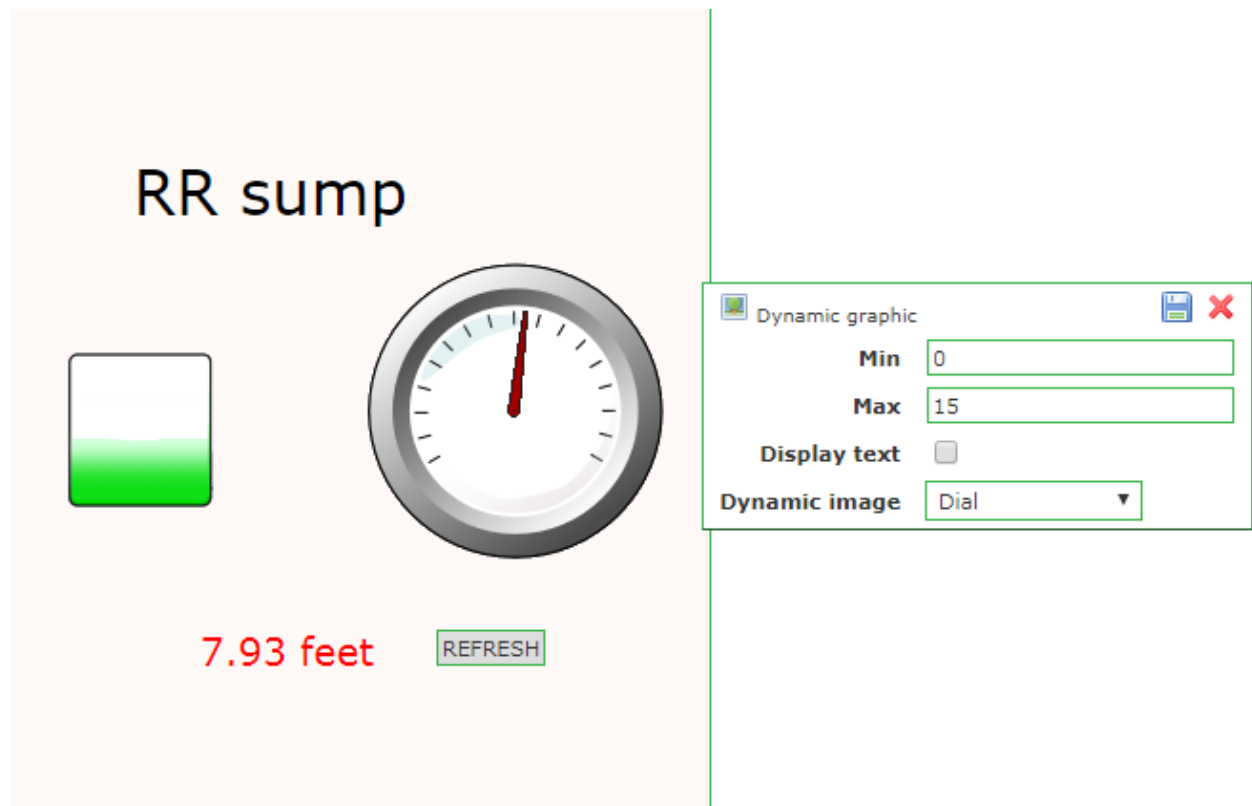


Figure-3.9 Graphical representation-Uploadingimage

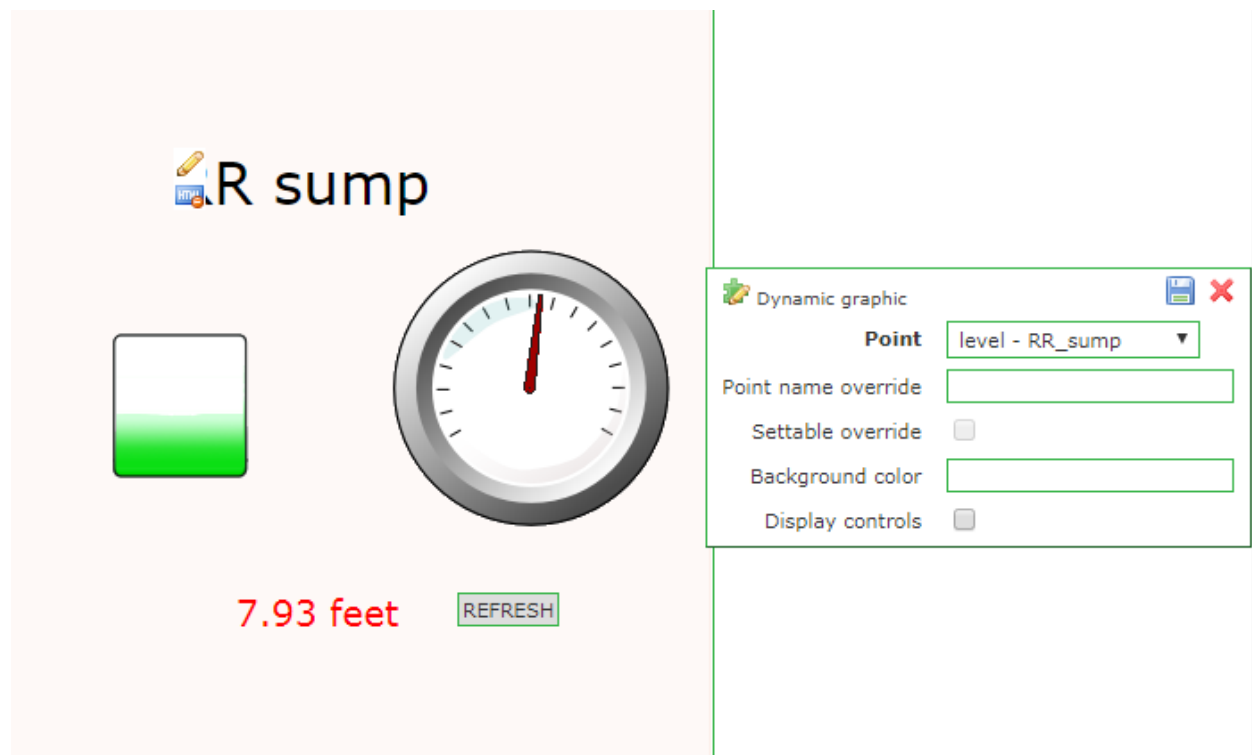
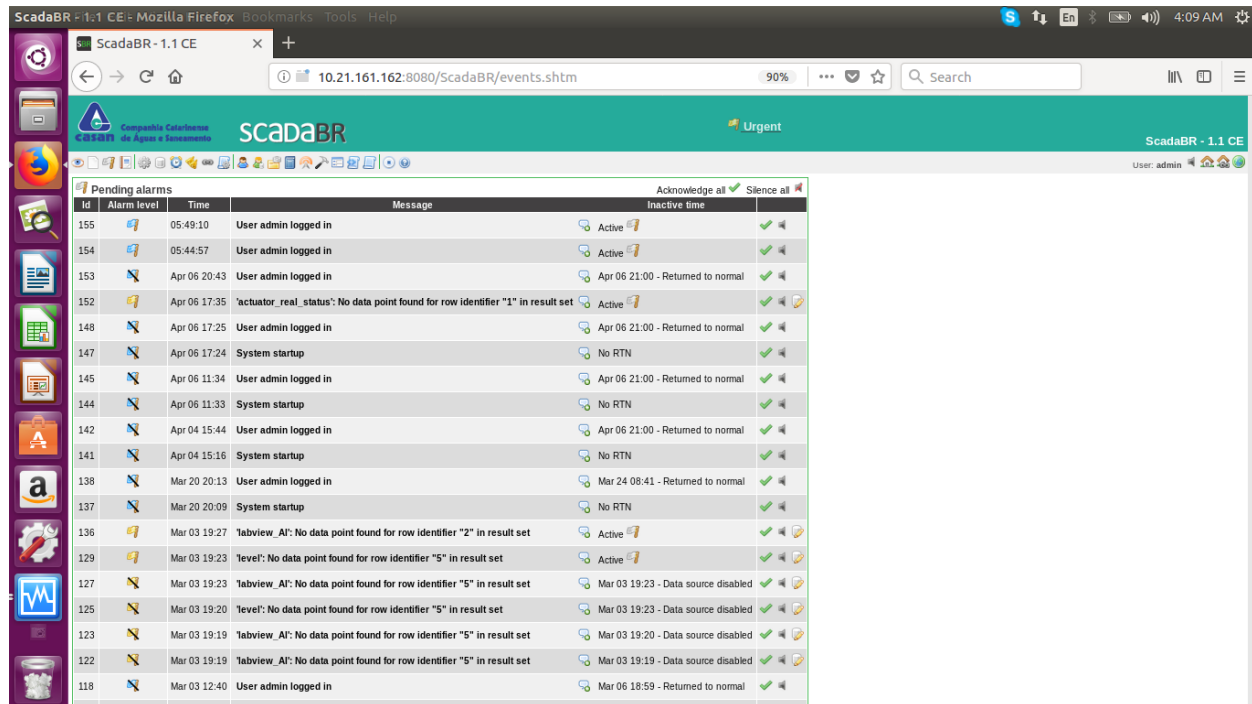


Figure-3.10 Graphical representation-Assigning datapoint to image

3.3.8 Alarm configuration

Alarm can be configured, if any critical events occurred. We have configured alarm, critical events as “deletion of data points” ,”creation of new datapoints” , “user login” “server shutdown” etc.



Id	Alarm level	Time	Message	Acknowledge all	Inactive time	Silence all
155		05:49:10	User admin logged in	Active		
154		05:44:57	User admin logged in	Active		
153		Apr 06 20:43	User admin logged in	Apr 06 21:00 - Returned to normal		
152		Apr 06 17:35	'actuator_real_status': No data point found for row identifier "1" in result set	Active		
148		Apr 06 17:25	User admin logged in	Apr 06 21:00 - Returned to normal		
147		Apr 06 17:24	System startup	No RTN		
145		Apr 06 11:34	User admin logged in	Apr 06 21:00 - Returned to normal		
144		Apr 06 11:33	System startup	No RTN		
142		Apr 04 15:44	User admin logged in	Apr 06 21:00 - Returned to normal		
141		Apr 04 15:16	System startup	No RTN		
138		Mar 20 20:13	User admin logged in	Mar 24 08:41 - Returned to normal		
137		Mar 20 20:09	System startup	No RTN		
136		Mar 03 19:27	'tabview_AI': No data point found for row identifier "2" in result set	Active		
129		Mar 03 19:23	'level': No data point found for row identifier "5" in result set	Active		
127		Mar 03 19:23	'tabview_AI': No data point found for row identifier "5" in result set	Mar 03 19:23 - Data source disabled		
125		Mar 03 19:20	'level': No data point found for row identifier "5" in result set	Mar 03 19:23 - Data source disabled		
123		Mar 03 19:19	'tabview_AI': No data point found for row identifier "5" in result set	Mar 03 19:20 - Data source disabled		
122		Mar 03 19:19	'tabview_AI': No data point found for row identifier "5" in result set	Mar 03 19:19 - Data source disabled		
118		Mar 03 12:40	User admin logged in	Mar 06 18:59 - Returned to normal		

Figure-3.10 Alarm window

3.3.9 SQL query and daily report

SQL query can be performed. ScadaBR has its own report generator. We generates daily report.

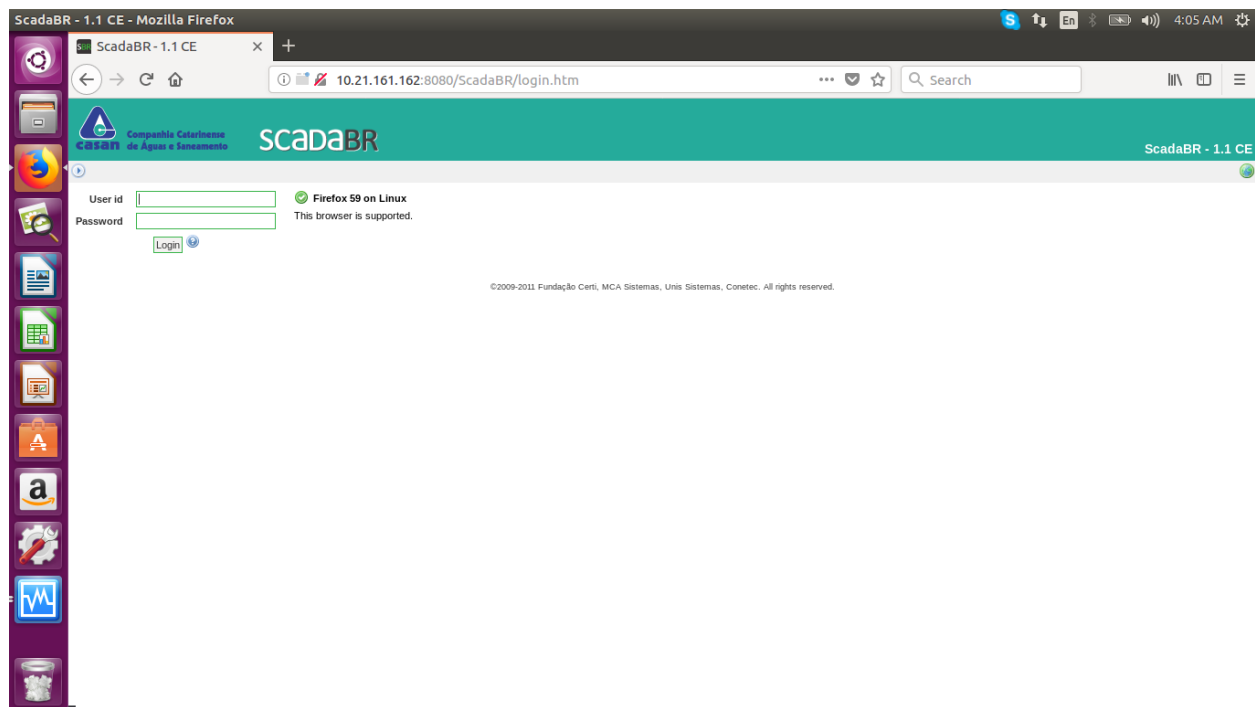


Figure-3.11SCADABR login page

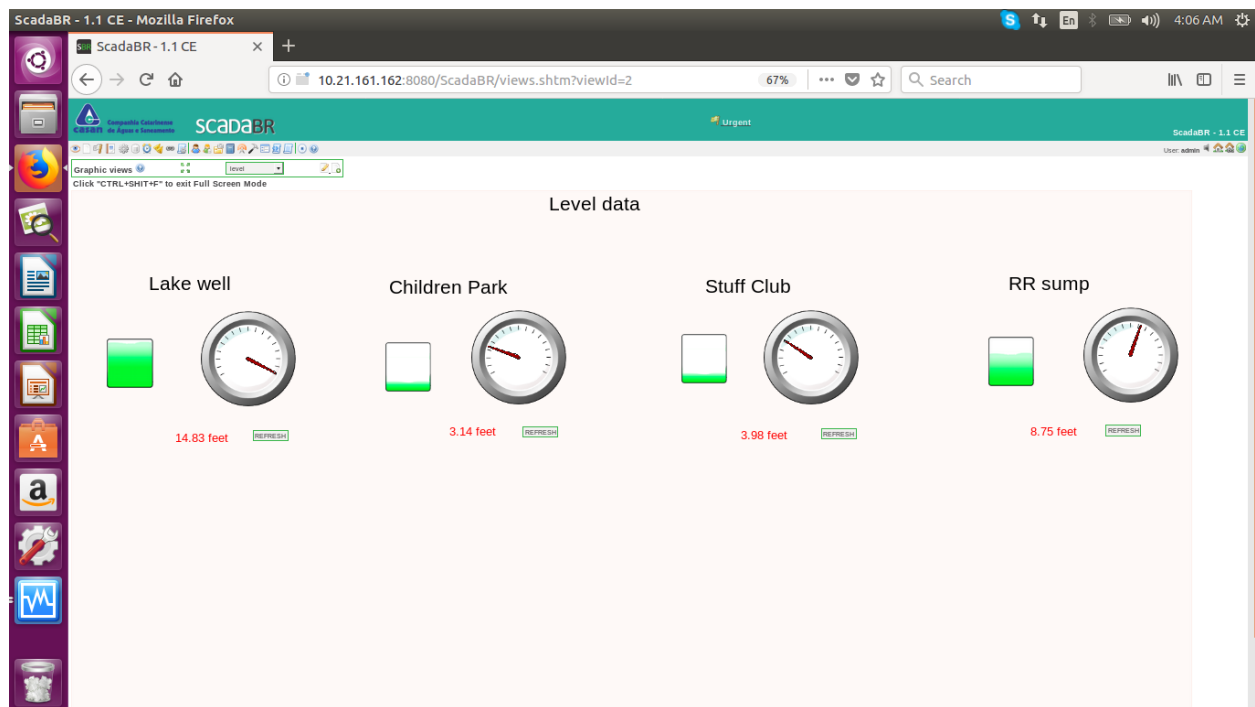


Figure-3.12 Level data

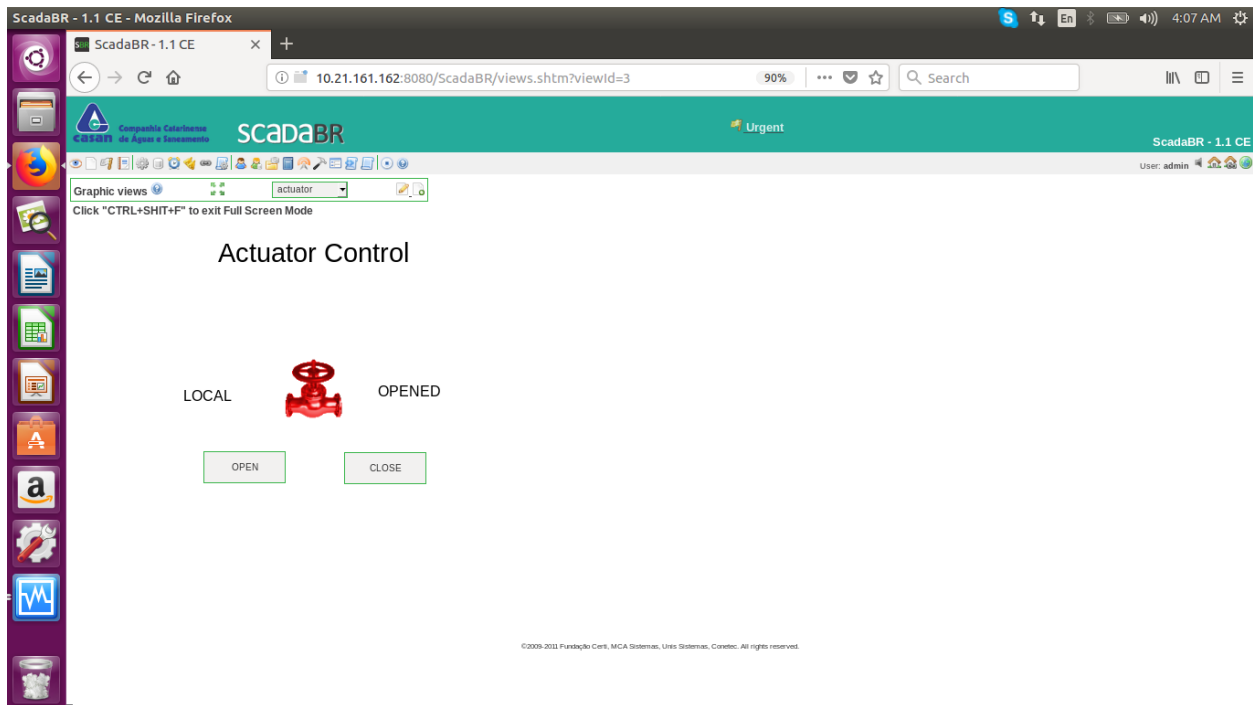


Figure-3.13 Actuator control

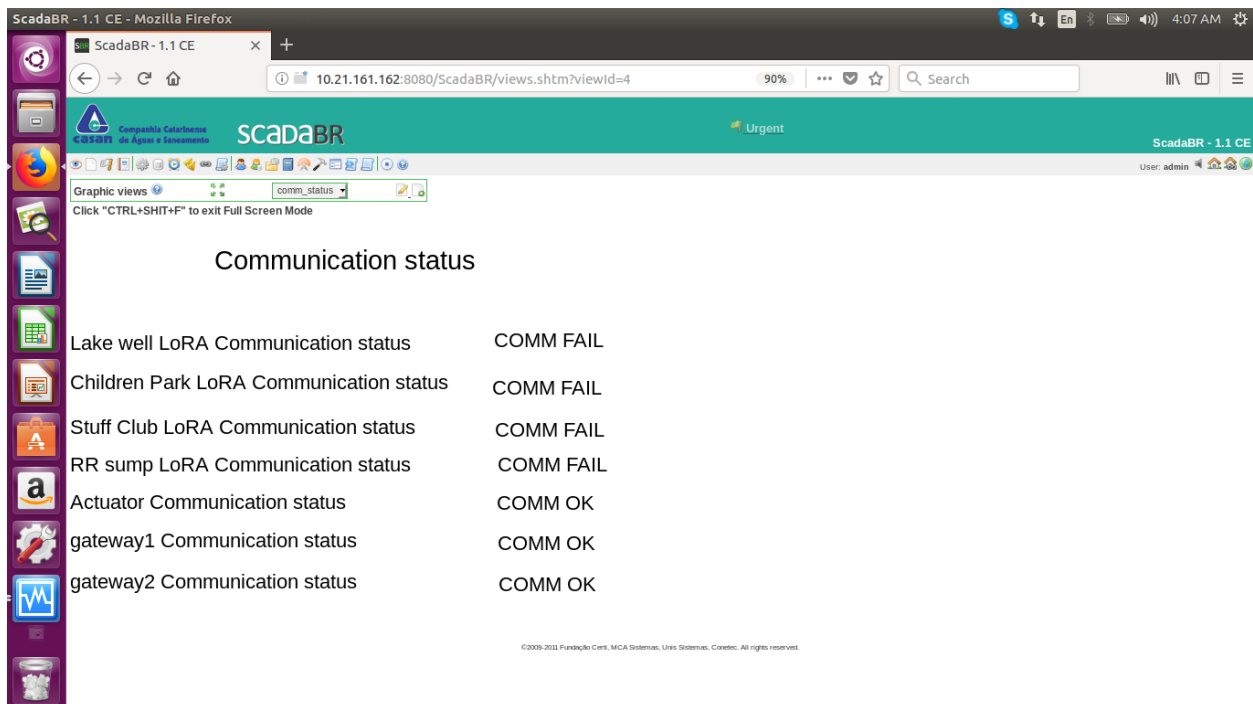


Figure-3.14 Communication availability

CHAPER-4

GSM operation and communication redundancy

4.1 Introduction

GSM stands for Global System for Mobile Communication. It is a digital cellular technology used for transmitting mobile voice and data services.

- GSM is the most widely accepted standard in telecommunications and it is implemented globally.
- GSM is a circuit-switched system that divides each 200 kHz channel into eight 25 kHz time-slots. GSM operates on the mobile communication bands 900 MHz and 1800 MHz in most parts of the world.
- GSM makes use of narrowband Time Division Multiple Access (TDMA) technique for transmitting signals.
- GSM was developed using digital technology. It has an ability to carry 64 kbps to 120 Mbps of data rates.

The uplink frequency range specified for GSM is 933 - 960 MHz (basic 900 MHz band only). The downlink frequency band 890 - 915 MHz (basic 900 MHz band only).

Further improvements were made with the introduction of GPRS and EDGE that extended the capabilities of GSM networks. Multimedia messaging was added to its list of features allowing subscribers to send pictures, audio clips, and even short video clips to each other. EDGE also increased the speed of mobile internet browsing to Dial-up speeds.

2G – The second generation of cell phone transmission. A few more features were added to the menu such as simple text messaging.

3G/4G/LTE is a whole new technology that was introduced as a replacement to the aging GSM technology. It offers substantial improvements over its predecessor in almost all aspects imaginable. For starters, mobile internet speeds for 3G/4G/LTE networks varies from Mbps to Gbps.

Data can be updated with high rate into database using OFC link, MPLS-VPN connection, and 3G/4G/LTE. Laying of OFC cable or using MPLS_VPL service from ISP is associated with very high cost. So, a better option is to use 4G or 2G.

4.2 How 4G can be implemented?

Using GSM Data connectivity-option1

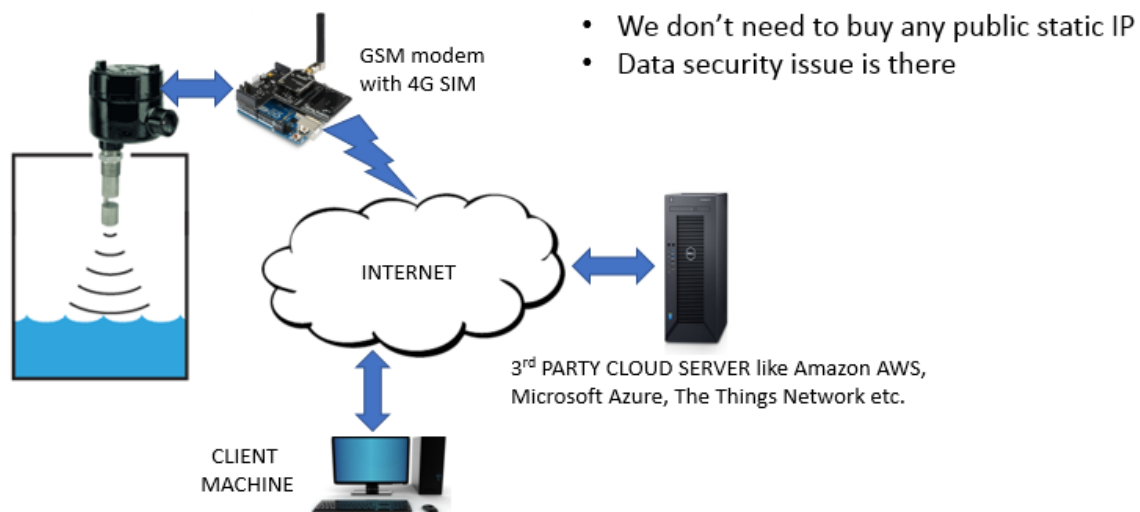


Figure-4.1 Data upload to cloud server using 4G

Using GSM Data connectivity-option-2a

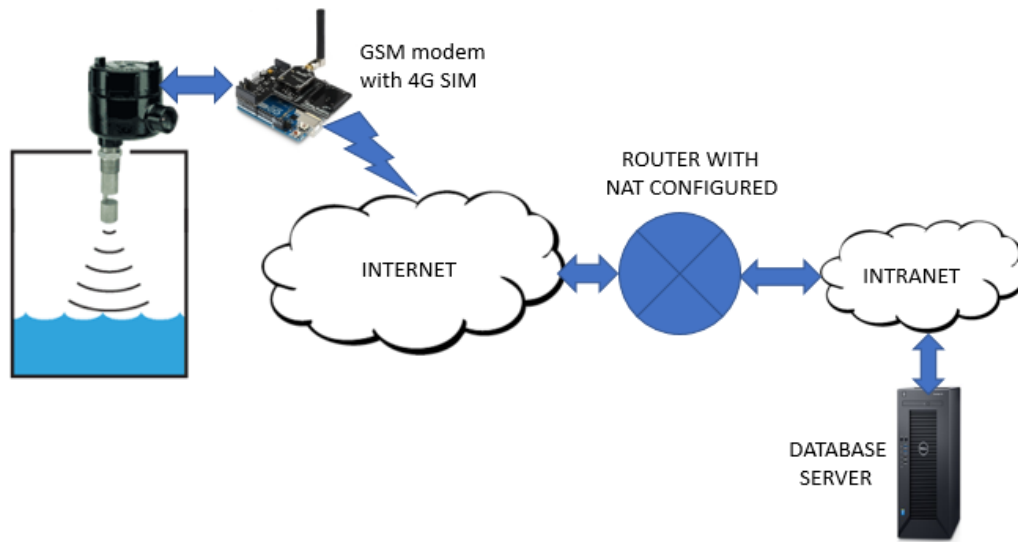


Figure-4.2 Data upload to local server using router

Using GSM Data connectivity-option-2b

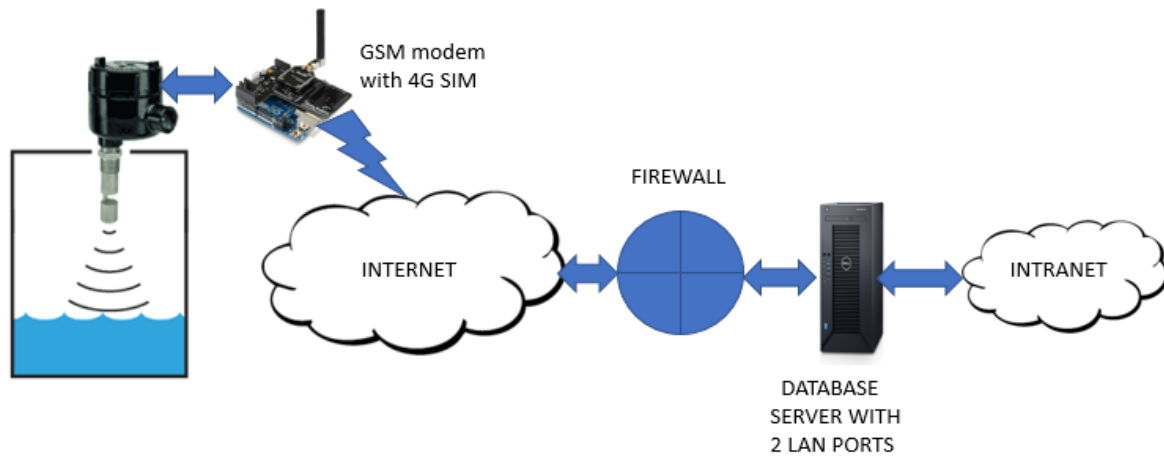


Figure-4.3 Data upload to local server using dual LAN port technique

Why option-2a is best?

- The advantage of using option-2a is that, we don't need to buy any additional public static IP address from ISP. But, in this case, NAT must be configured in existing router.
- In option-2b, we have to procure a public static IP address and a firewall device. The server must have two LAN cards.
- Data can be imported into local database in both the configuration using modern IoT protocol like MQTT or COAP

4.3 Why are we using 2G?

- We are using 2G as redundant communication system

- LORA is highly reliable
- We don't need to buy any firewall or router
- We don't need public static IP
- 3G/4G/LTE modem is costly
- Though data rate for 2G is very less(100 SMS per day for SMS pack, it is suitable in our application.

4.4 2G operation mode

The server is connected to a controller (Arduino) along with GSM modem. To send/receive sms, server(Python-S1) communicates to the .ino program running on the Arduino. Each sensor/actuator is also connected to a controller(arduino) along with a GSM modem.

Two different types algorithm are used

- Each level sensor sends a response SMS to the server, each time it receives a request SMS from the server (Time period: 15 min)
- When the actuator receives a open/close command from the server, it starts sending SMS to the server with a time period of 1 min. Total 4 sms are sent. SMS consists of status of the actuator along with the actuator address.

We can receive SMS on our mobile phone from the nodes.

4.5 Algorithm

4.5.1 AT command

GSM modem is connected to an Arduino through a serial port. To send/delete a SMS, or to monitor network parameter, Arduino sends an AT command to the modem. If Arduino gets a valid response, within 2 sec, the AT command is valid. For example, to send a SMS, Arduino, after giving corresponding AT command to the modem, should get a response "OK" from the modem within 2 sec. A flowchart (Flow chart-1) shows how AT command works. Before sending AT command to the modem, Arduino ensures, that no character is available on the serial port.

4.5.2 Checking by Arduino that modem is available for operation

First Arduino pings the modem by sending corresponding AT command to the modem. AT command is sent until ping response is not obtained. After getting a ping response, The followings are checked.

- SIM is plugged in
- Signal strength is available
- SIM is registered to the network

If these are satisfied, then it is assumed by the Arduino that Modem is ready for operation. A flowchart (Flow chart-2) shows how this checking is done.

4.5.3 Sending SMS

Before sending any SMS, the modem must be set in text mode. Arduino sends AT command to set the modem in text mode. If the response is "OK", then text mode is set. Then Arduino creates a message, assigns it to sms1, assign any number to number1, sends AT command to the modem for sending sms1 to number1. If the response is "OK", SMS is sent. A flowchart (Flow chart-3) shows how the SMS is sent.

4.5.4 Receiving SMS

Arduino waits for any string available on the serial port. The string is not AT command responses, it is an incoming SMS. Any SMS which is received by the GSM modem is forwarded to the serial port by the modem. If the incoming SMS has a start marker(@ here), and end marker(# here), then Arduino reads the SMS. A flowchart (Flow chart-4) shows how SMS is received.

4.5.6 Function of the Arduino connected to the level sensor

- The level sensor is connected to a Arduino which is further connected to a GSM modem through Serial port. A firmware is written on the Arduino to perform the following task.
- The Arduino first ensure that the modem is ready for operation.
- Then it waits for any incoming SMS.
- If the incoming SMS consists of the word “start” and it is from the server, the Arduino reads the distance measured by the sensor. This is a count value (The sensor output is (0-5)V which is read by the Arduino. The ADC of the Arduino converts this analog voltage into a corresponding count value (0 to 1024). Maximum distance that can be measured by the sensor is 10 m=1000 cm. As 1000 cm corresponds to 1024 which is nearly equals to 1000, count value x corresponds to x cm approximately) which is related to tank level. Then, it makes a SMS(let sms1) consisting of the count value(converted into string first), and the sensor address in the format @deviceDp256q# (where, deviceD is the address of the sensor installed at RR sump, “256” is the count value, i.e. distance measured by the sensor = 256 cm approximately.)
- Finally, sms1 is sent to the server. If the incoming SMS is from a mobile phone, then sms1 is sent to the mobile phone.
- After sending sms-1 either to mobile or to the server, incoming sms is deleted.
- A flowchart (Flow chart-5) shows the sequence.

4.5.7 Function of the Arduino installed with actuator

If incoming sms is @mobile#, it sends a reply SMS (let sms-1) in the format @actuator-p676q#. SMS-1 is sent to the SIM card from where incoming SMS is received. Here “actuator” is the address of the actuator. “676” is the status of the actuator, which is understood by the server.

If incoming SMS is @open# from the server, actuator starts opening and sends SMS-1 to the server. Sms is sent four times after each 1 min.

If incoming SMS is @close# from the server, actuator starts closing and sends SMS-1 to the server. Sms is sent four times after each 1 min.

A flowchart (Flow chart-6) shows the sequence.

4.5.8 Reading SMS by the Arduino installed on the server

The Arduino (having 2 serial ports) is connected to the modem through serial port-1, and to the server through serial port-2. First, the Arduino ensures that the modem is ready for operation, then it waits for any for incoming SMS. If incoming SMS contains the character('@' and '#'), Arduino read this SMS, and places on serial port-2. Now, the data (ex: @deviceDp256q#) sent by nodes are available on serial port-2. This is read by the server (Python-S1) to store in Database. A flowchart (Flow chart-7) shows the sequence.

4.5.9 How the data is stored in database through GSM?

As mentioned above, It is done by the server (Python-S1) program. Python S-1 has two subroutines, Subroutine-1 is for storing data to database, and subroutine-2 is for sending SMS to nodes. . A flowchart (Flow chart-8) shows the sequence. Incoming data from Arduino is the data available on Serial Port-2.

4.6 How Redundancy in communication is achieved?

Data is updated into database through primary link(Link-1). In case of primary link failure, a SMS is sent to the node, response SMS is received, required data extracted from response SMS, and is stored in the database. Therefore we need to know if primary communication is available. Gateway-1 and Gateway-2 send some random value in the database for various nodes installed at the various locations with some time period as mentioned below.

Lake well-90 sec, Children Park-60 sec, Staff Club-30 sec, RR sump-30 sec

Random value for the actuator is updated with a period of 5 sec.

This is done by the gateways (Python-G1, and Python-G2)

4.6.1 updating random values by Gateway-1

The data received through LoRA by gateway-1 is a string, consisting of LoRA address, and level data of the nodes. LoRA addresses of lake well, children Park, and staff club are 2, 3, and 4 respectively. 1st element of the string is the address. If 1st element is 2, then data is coming from lake-well through children Park, and staff club. If 1st element is 3, then data is coming from children park through staff club,

but data is not coming from lake well. If 1st element is 4, the data is coming from the staff club only. So depending on the 1st element of the string, random values for the nodes installed at the following locations are updated on the database.

First element of the string	Random value update for
2	lake-well, children Park, Staff club
3	children Park, Staff club
4	Staff club
5	actuator

Table-4.1 Updating random value according to the string received by gateway

4.6.2 updating random value by Gateway-2

Updating of Random values for the node installed at RR sump is done, as done by Gateway-1(Python-G2). It is simpler, as only one node is there.

Flow charts show how the random values are updated. Flowchart-9 is for gateway-1, Flowchart-10 is for gateway-2

4.6.3 Updating primary link availability in database

Random values updated by the gateways are read by a Python Program(Python-S2) to identify communication availability. communication availability is also updated in the database. Random values are read by the program with the following time duration.

For RR sump: 40 sec

For Lake well, and children Park: 2 min

For staff club: 40 sec

Flowchart-11 shows for RR sump, and actuator only.

4.6.4 Sending SMS automatically by the server when primary link fails

Primary link availability of various nodes through primary channel is known now. If this communication is not available for a particular node, a SMS is sent to that node, by the server automatically. If actuator is not available through primary link, and open command is given from any interface, then SMS(@open#) is sent to the actuator by the server.

For sending open command to the actuator, server (Subroutine-2, Python-S1) writes 'x' on Serial Port-2. Arduino sends the open command to the actuator after reading this 'x' from the serial port.

All other SMS sending procedures are given in the flowcharts. AT command is given through Serial port-1, as modem is connected to that port. Once sms is sent successfully, status is sent to Serial port-2 for reading it by the server (Subroutine-2, Python-S1).

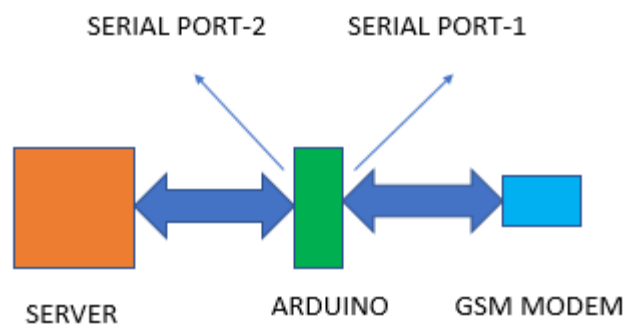


Figure-4.4 GSM modem connectivity with server

Flowchart-12 shows the job sequence of the Arduino, and the Flowchart-13 shows the job sequence of the server.

4.6.5 Getting data through GSM manually

Manual data request fields are the fields in the database 'scada' corresponding to id-5 to id -8(value-5 to value-8). Subroutine-2 of Python-S1 updates value-8 to 10 and sends a message to the nodes installed at RR sump. This is the way how manually data is obtained through GSM. This event happens automatically after each 15 min, if the node installed at the RR sump is not available through primary link. Same concept is applied for sending open/close command to the actuator. If refresh button(under RR sump graphics) is pressed, value-8 is updated to 1.

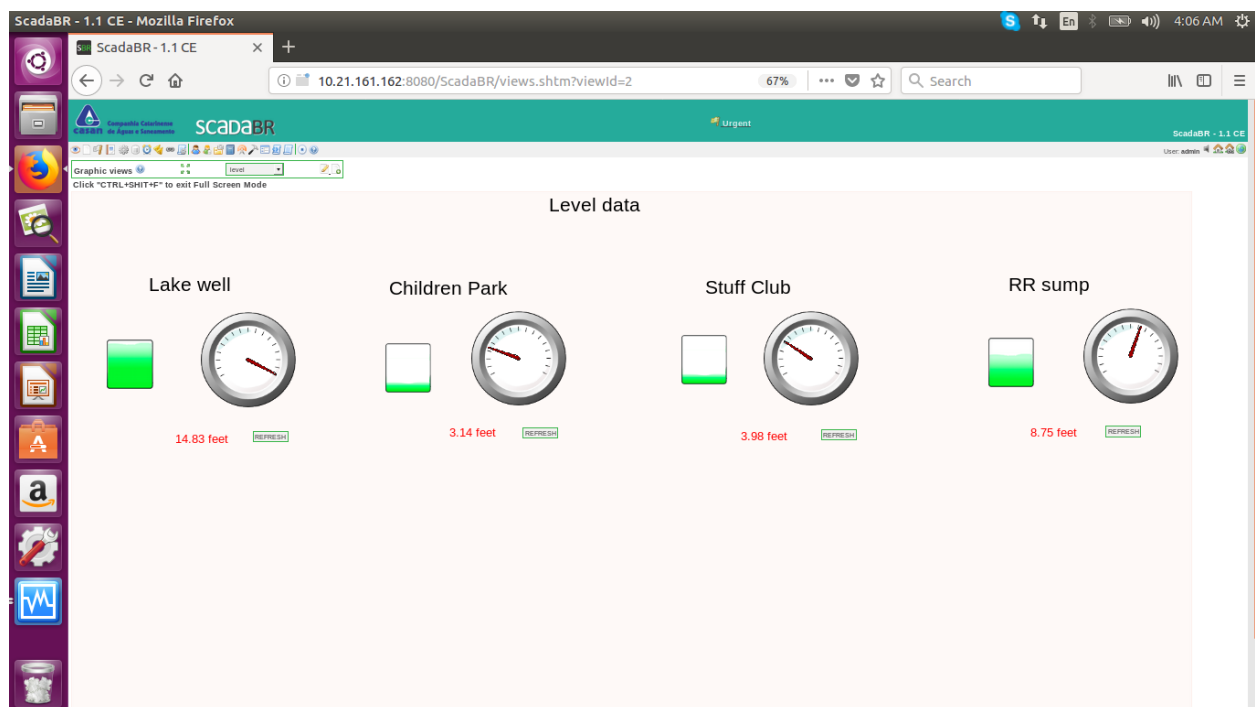
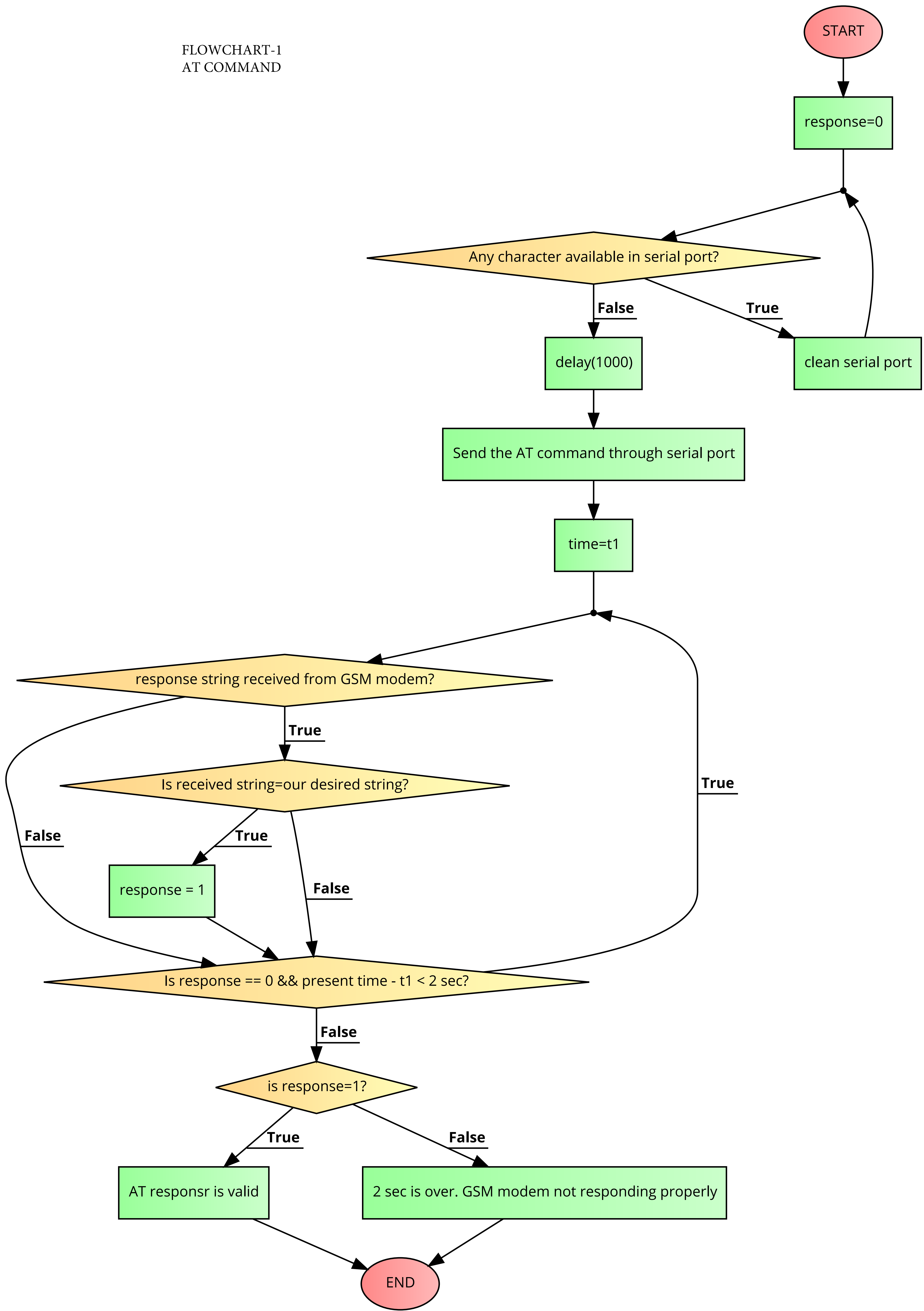
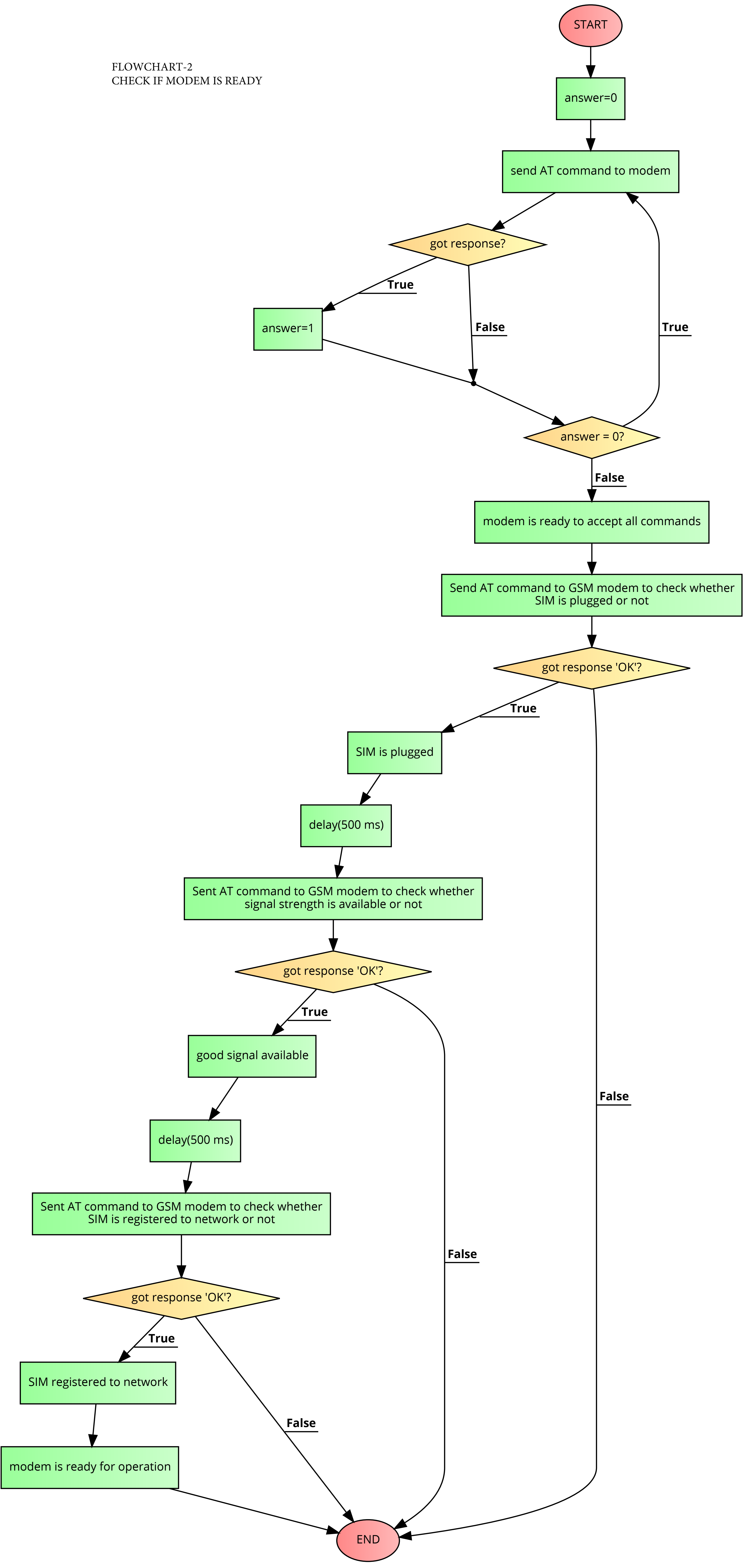


Figure-4.5 Refresh button on SCADA

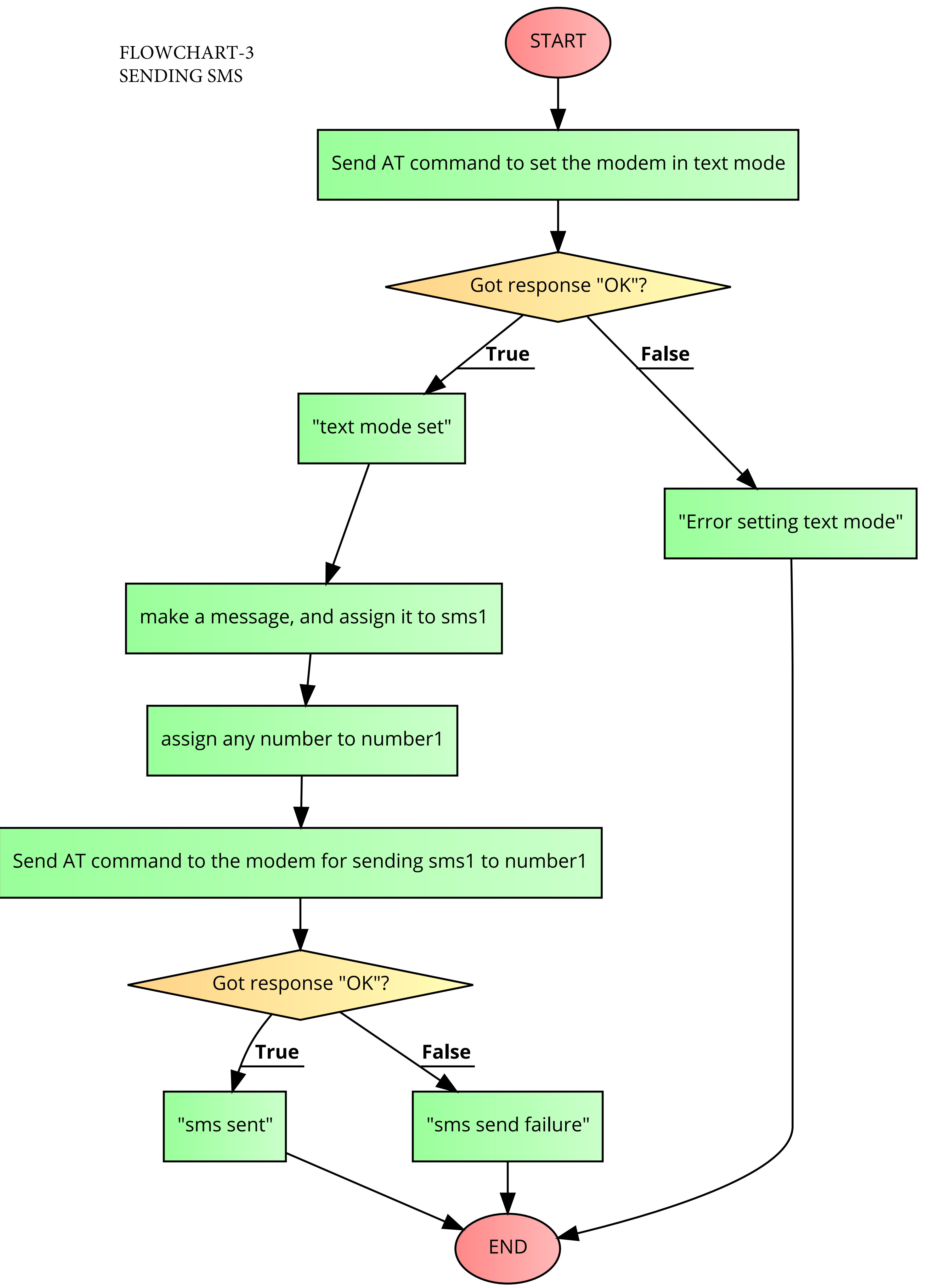
FLOWCHART-1
AT COMMAND



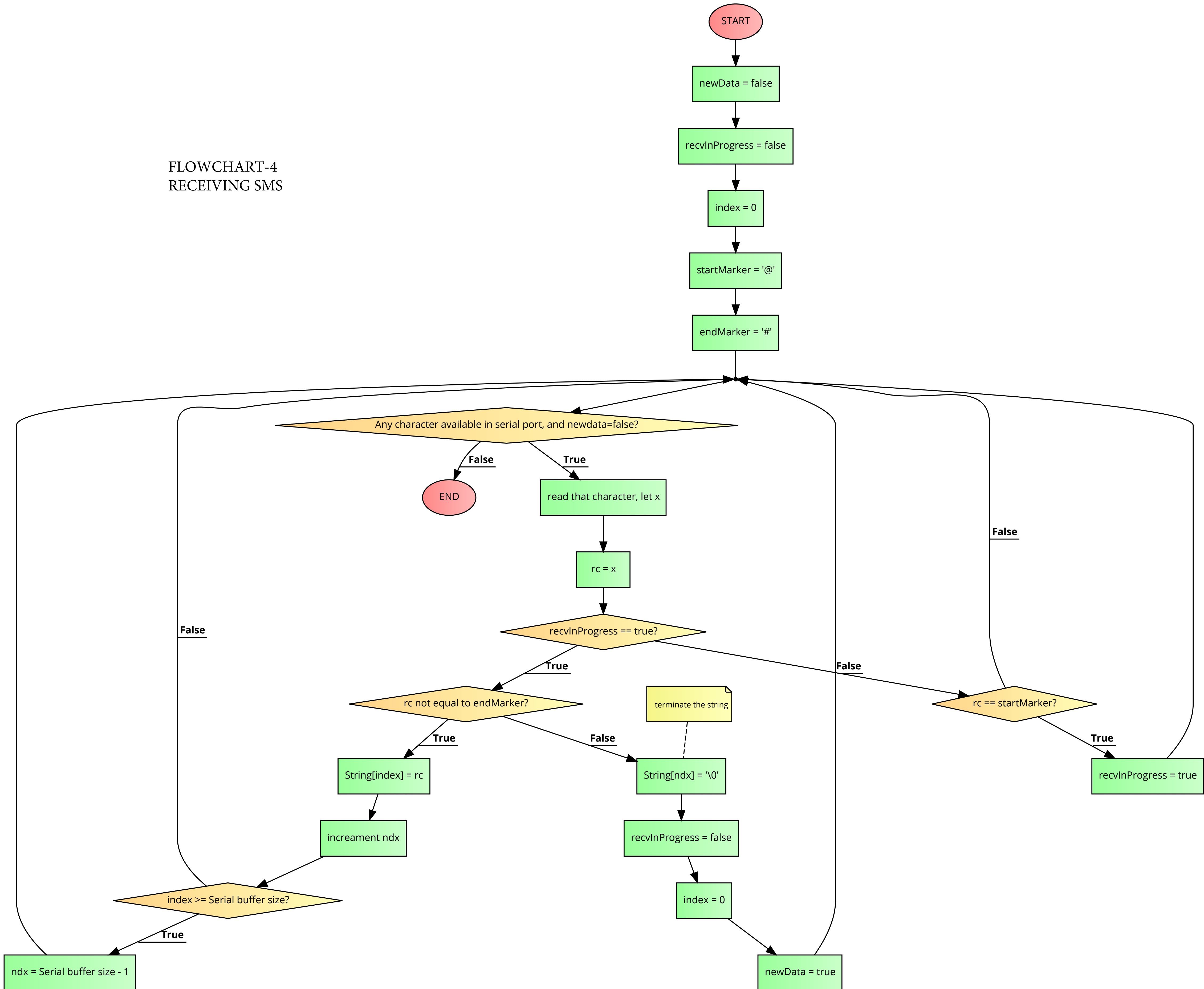
FLOWCHART-2
CHECK IF MODEM IS READY



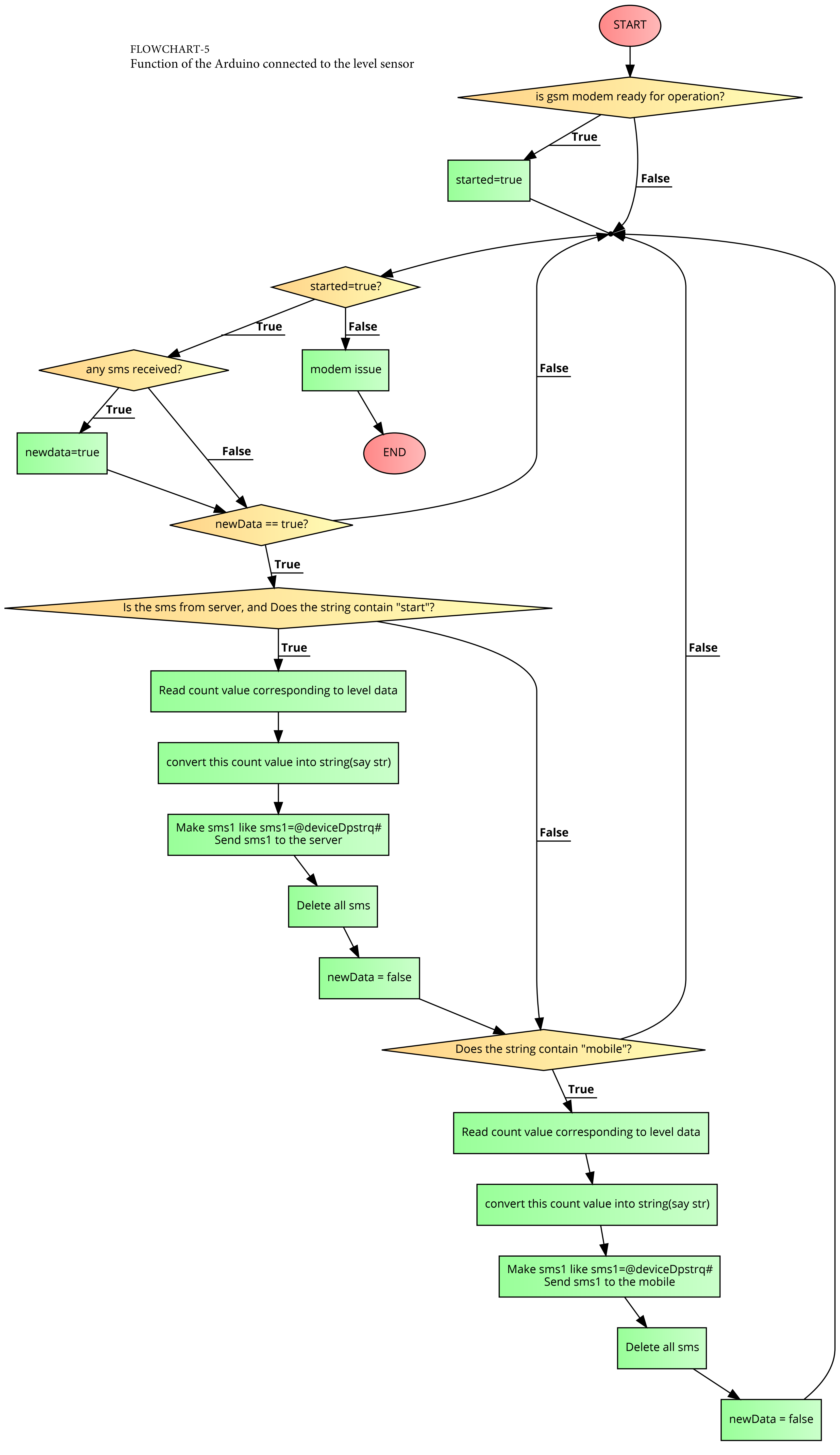
FLOWCHART-3
SENDING SMS



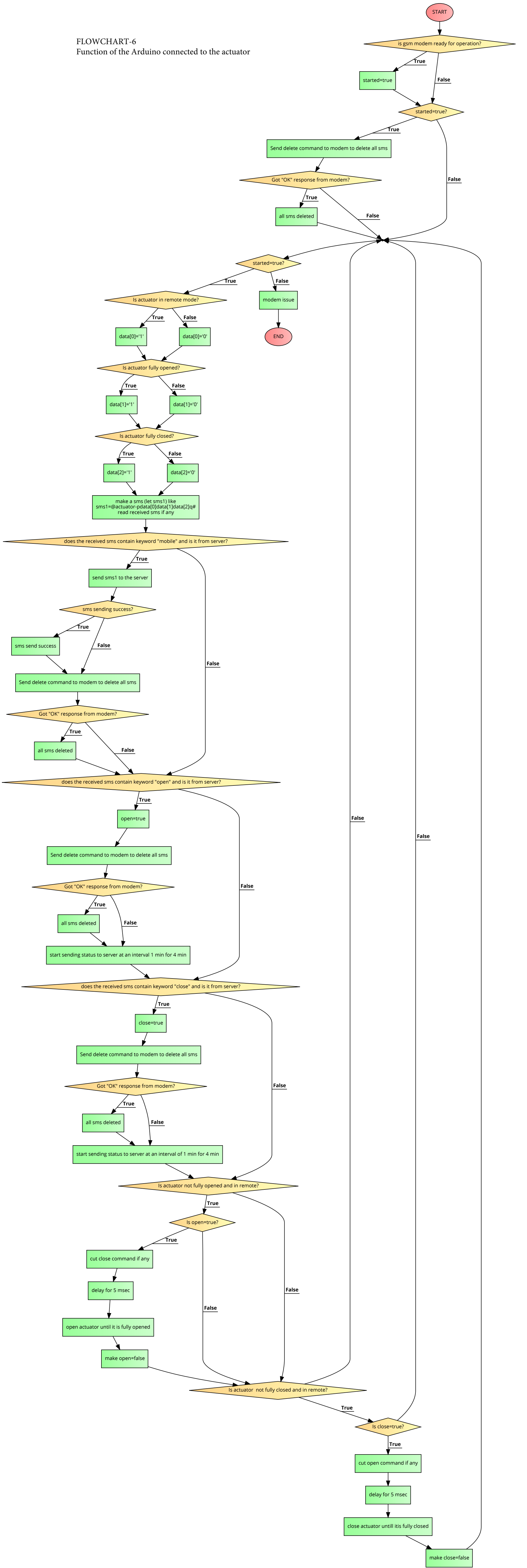
FLOWCHART-4
RECEIVING SMS



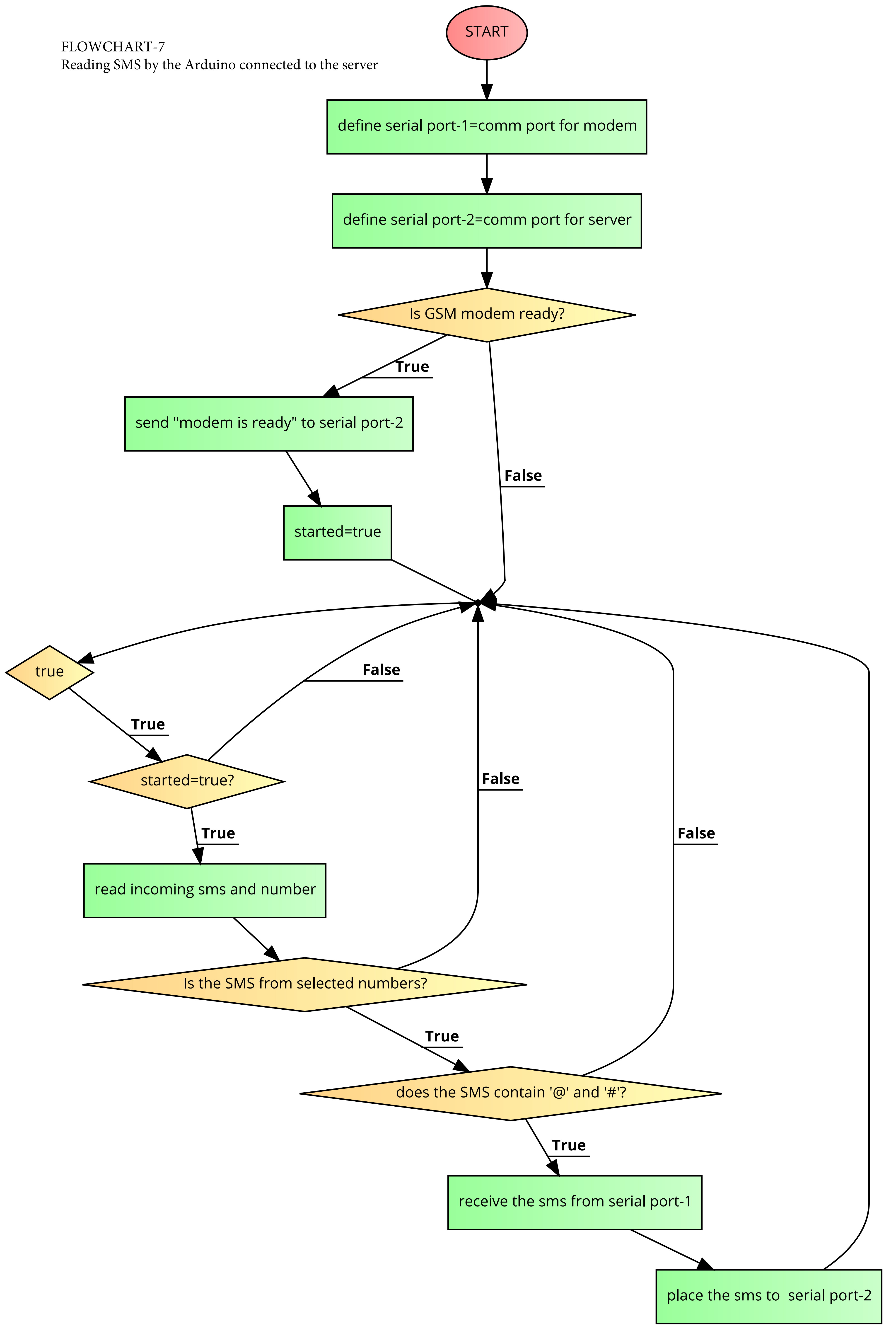
FLOWCHART-5
Function of the Arduino connected to the level sensor



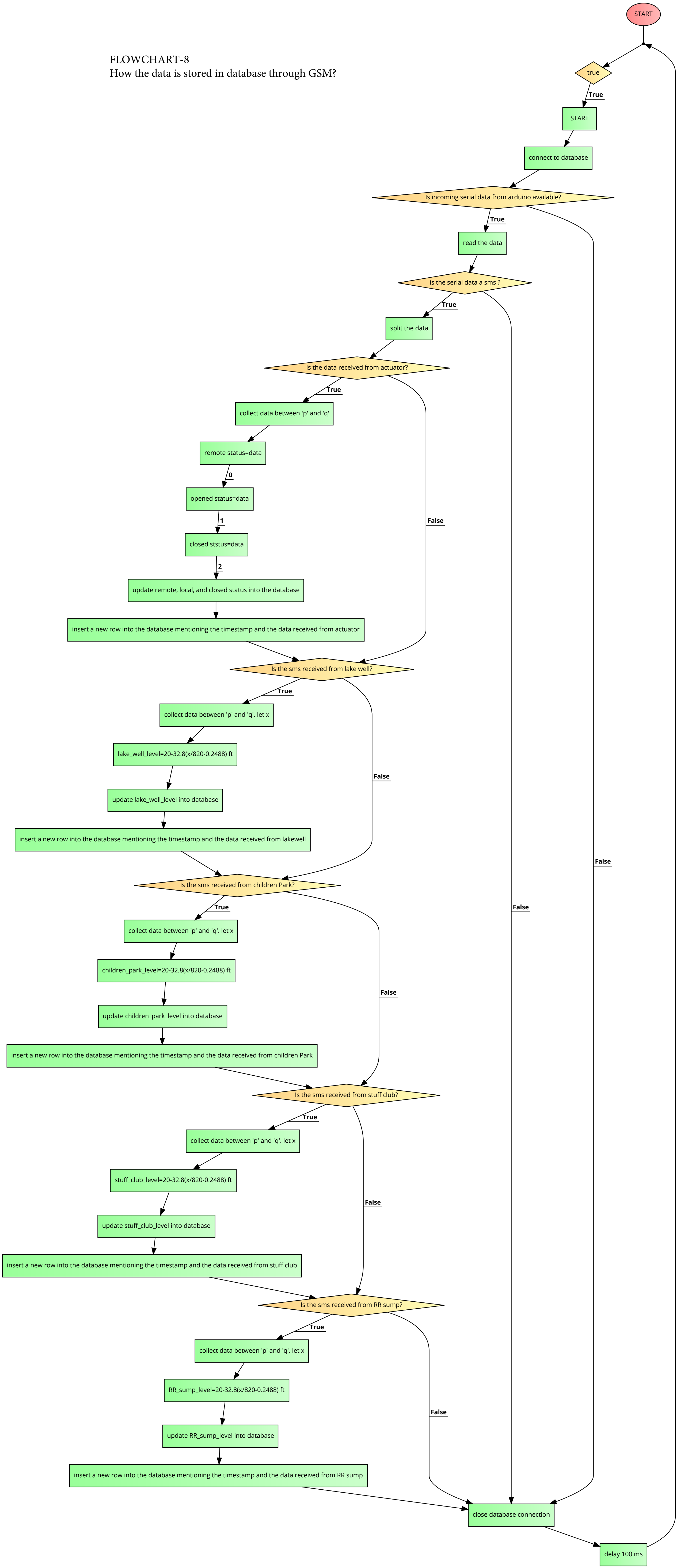
FLOWCHART-6
Function of the Arduino connected to the actuator



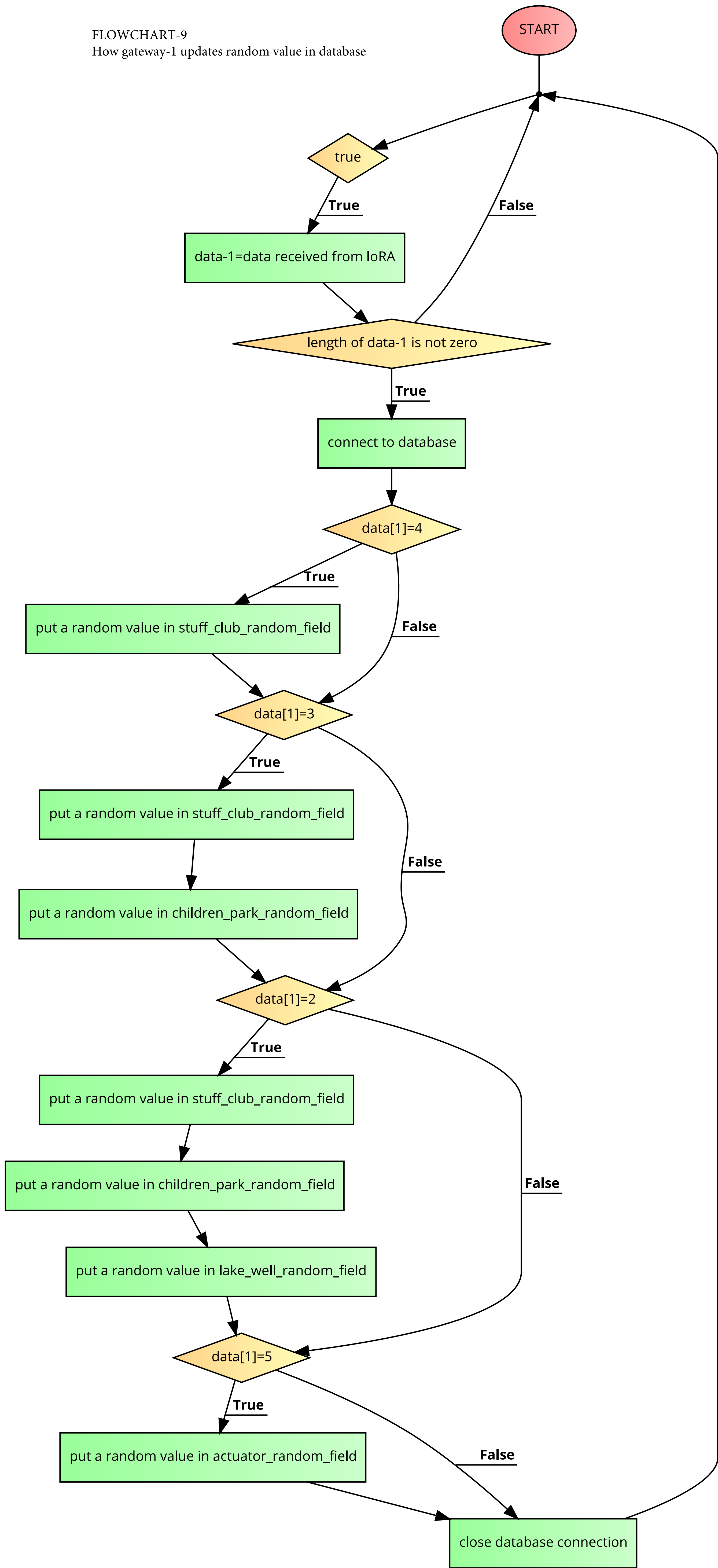
FLOWCHART-7
Reading SMS by the Arduino connected to the server



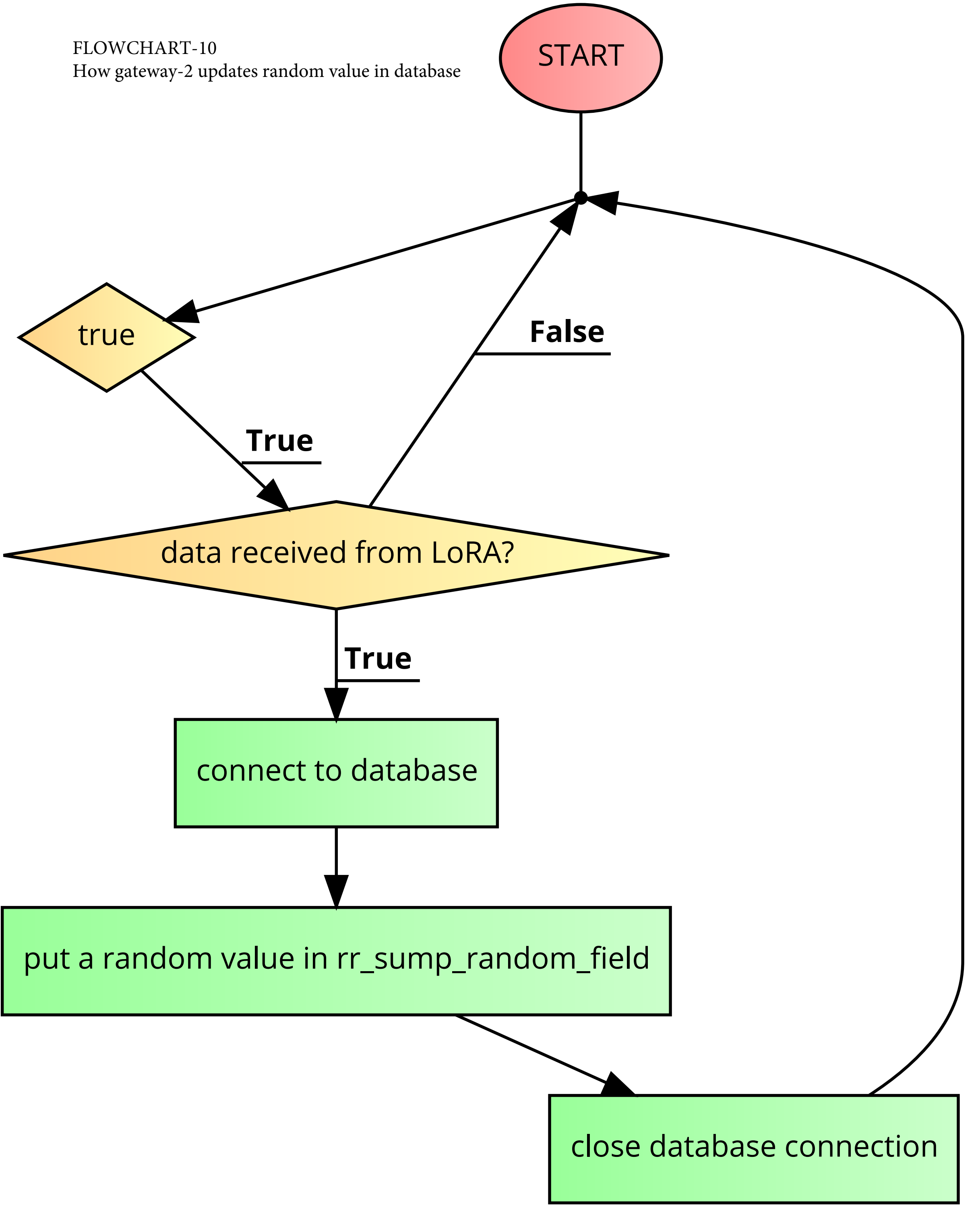
FLOWCHART-8
How the data is stored in database through GSM?



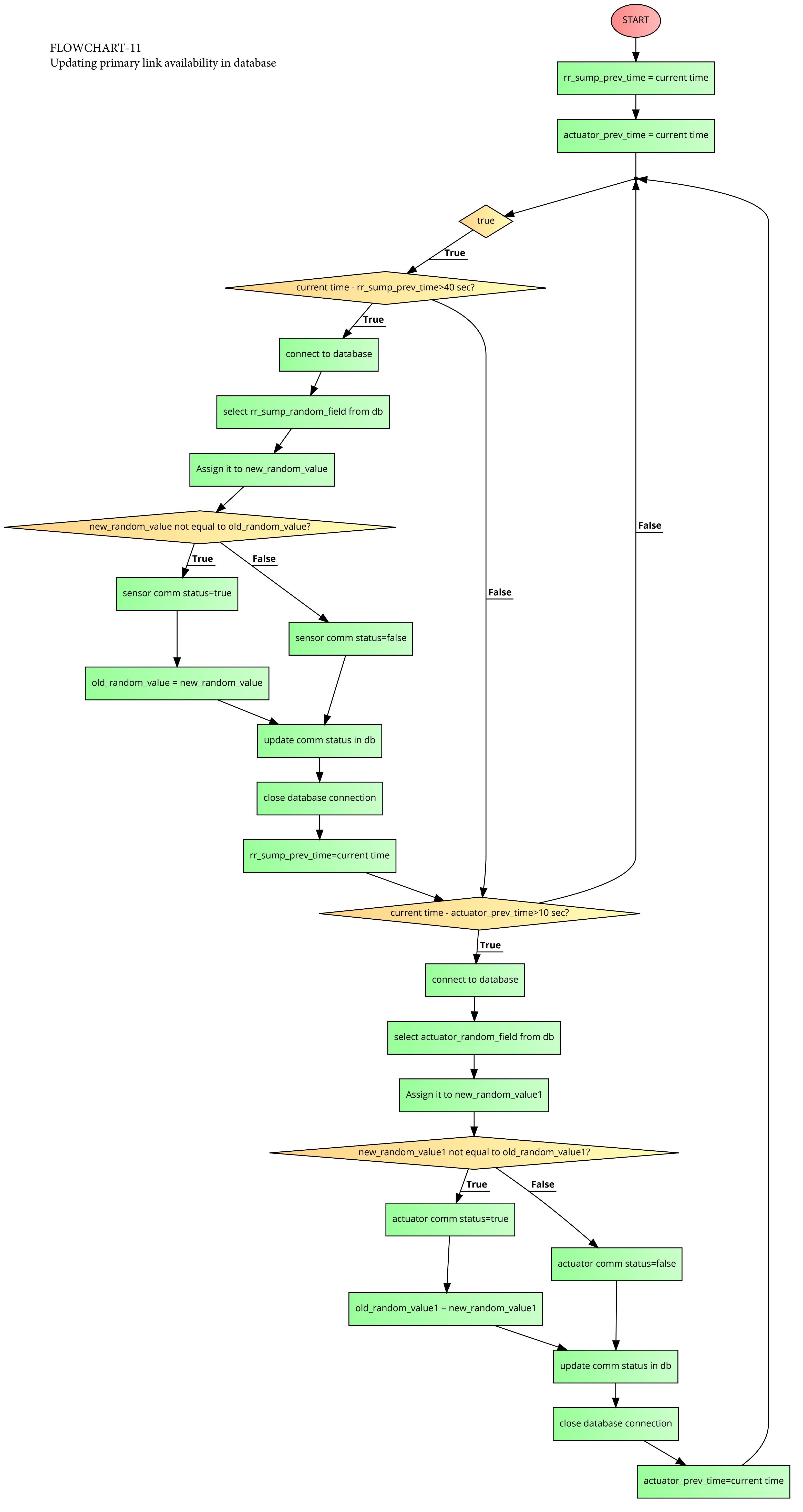
How gateway-1 updates random value in database



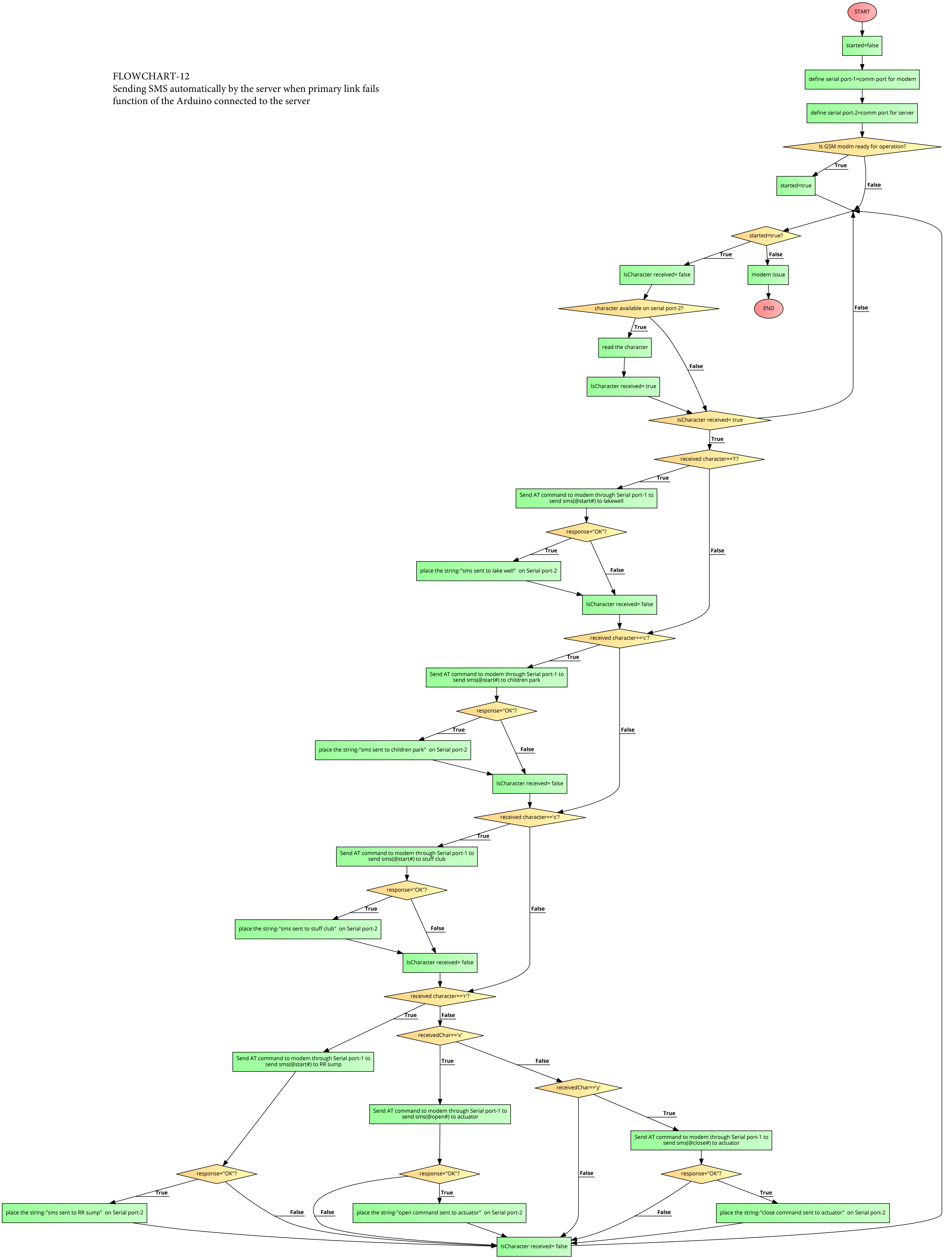
FLOWCHART-10
How gateway-2 updates random value in database



FLOWCHART-11
Updating primary link availability in database

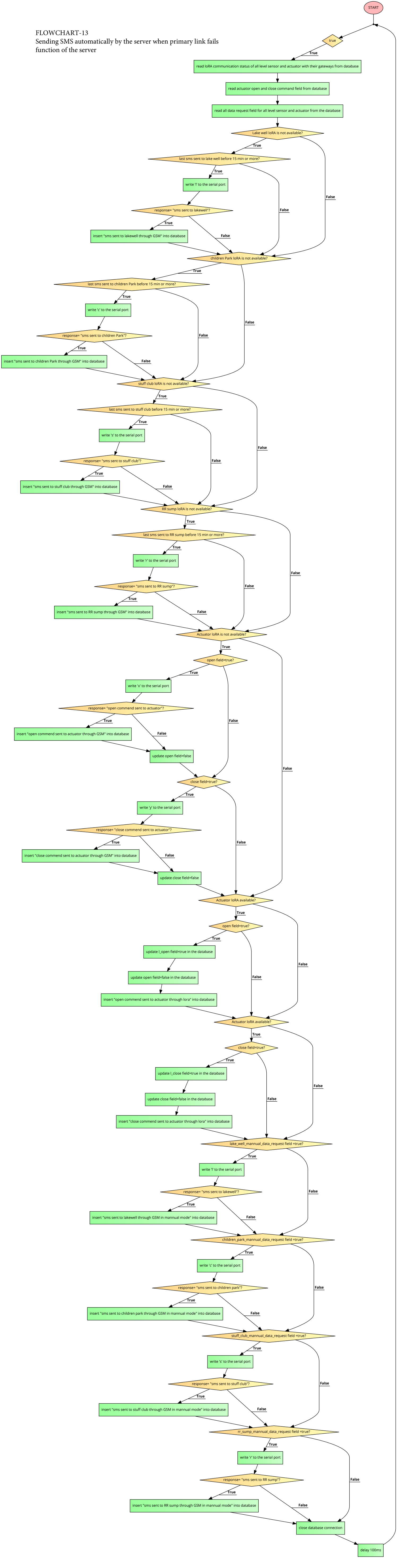


FLOWCHART-12
Sending SMS automatically by the server when primary link fails
function of the Arduino connected to the server



FLOWCHART-13

Sending SMS automatically by the server when primary link fails
function of the server



CHAPTER-5

Simulation

5.1 Simulation for the actuator:

5.1.1 Getting actuator status on mobile phone: after sending a SMS(@mobile#), from the mobile(which is allowed), a response SMS(@actuatotp676q) is received by the mobile.676 is the status of the actuator, which indicated that the actuator is in local mode, and is in opened condition.

Me GSM Proje...
+919952987526

CALL MORE

M

@actuatorp676q#

16:51

18:06

@mobile#

M

@actuatorp676q#

18:06

M

Modem is ready

18:22

18:23

@mobile#

M

@actuatorp676q#

18:23

18:23

@open#

M

Modem is ready

18:27

18:27

@open#

18:33

@mobile#

M

@actuatorp676q#

18:34



Enter message



Figure-5.1 SMS received from actuator node

5.1.2. Sending an open command from user interface: The program running on the gateway is stopped to ensure, that primary link is down. After clicking on the open button from the web application, status of the program running on the arduino(connected to the actuator) is monitored. It is observed that the SMS(@open#) is received by the arduino. The arduino, then, starts sending the status to the server.

```
data= 676
message= @actuatorp676q#
Waiting to read
ATT: +CMGL
RIC:
+CMGL: 1,"REC UNREAD", "+919080404532", "", "18/05/04,18:27:15+22"
@open#

OK

ATT: OK
RIC:
OK
```

Figure-5.2 Open command received by the arduino through GSM


```
DEBUG:>  
ATT: +CMGS  
RIC:  
+CMGS: 153  
  
OK  
  
SMS sent OK  
ATT: OK  
RIC:  
OK  
  
All sms deleted  
data= 676
```

Figure-5.3 Arduino starts sending SMS to the server after getting an open command

5.2 Getting data from the level sensor installed at RR sump through GSM

5.2.1 Getting count value corresponding to distance measured by the sensor on mobile phone: After sending a SMS(@mobile#), from the mobile(which is allowed), a response SMS(@deviceDp230q#) is received by the mobile. 230 is the count value corresponding to the distance measured by the sensor

← Me GSM Proje...
+917358699052

CALL MORE

Thursday, 19 April 2018

02:50 @mobile#

17:49 @mobile#

Friday, 27 April 2018

13:13 @mobile#

Friday, 4 May 2018

17:55 @mobile#

M @deviceDa2bp230q# 17:55

17:58 @mobile#

M @deviceDa1bp232q# 17:58

18:05 @mobile#

M @deviceDa2bp239q# 18:06

Enter message



Figure-5.4 SMS received from sensor node

5.2.1 Getting level data manually from the user interface:

Before clicking on the refresh button on the scada, RR sump reading was 10.26 ft.

After clicking on the refresh button, a SMS is sent to the node.

```
gowsalya@Gowsalya-PC:~$ python server_side_command.py
File "server_side_command.py", line 21, in <module>
    mycursor=conn.cursor()
File "/usr/lib/python2.7/dist-packages/mysql/connector/connection.py", line 1383, in cursor
    raise errors.OperationalError("MySQL Connection not available.")
mysql.connector.errors.OperationalError: MySQL Connection not available.
gowsalya@Gowsalya-PC:~$ python server_side_command.py
Starting...
modem ready
ok
good signal quality
SIM is plugged
Registered to network
Received message node set
Waiting to send sms...
2018-05-04 19:02:06.041244
manual data request to RR sump
Setting SMS mode...SMS mode set
Sending SMS

SMS sent to children Park
1
All message deleted
2018-05-04 19:03:49.049136
manual data request to RR sump
Setting SMS mode...SMS mode set
Sending SMS

SMS sent to children Park
2
All message deleted
2018-05-04 19:05:47.428430
manual data request to RR sump
Setting SMS mode...SMS mode set
Sending SMS

SMS sent to children Park
3
All message deleted
```

Figure 5.5 Sending SMS to node at RR sump by the server

Then a SMS is received by the server

[illegible]

Figure-5.6 SMS received by the server from RR sump

Then, same reading(9.64 ft) gets displayed on scada, mobile app, and on the web application.

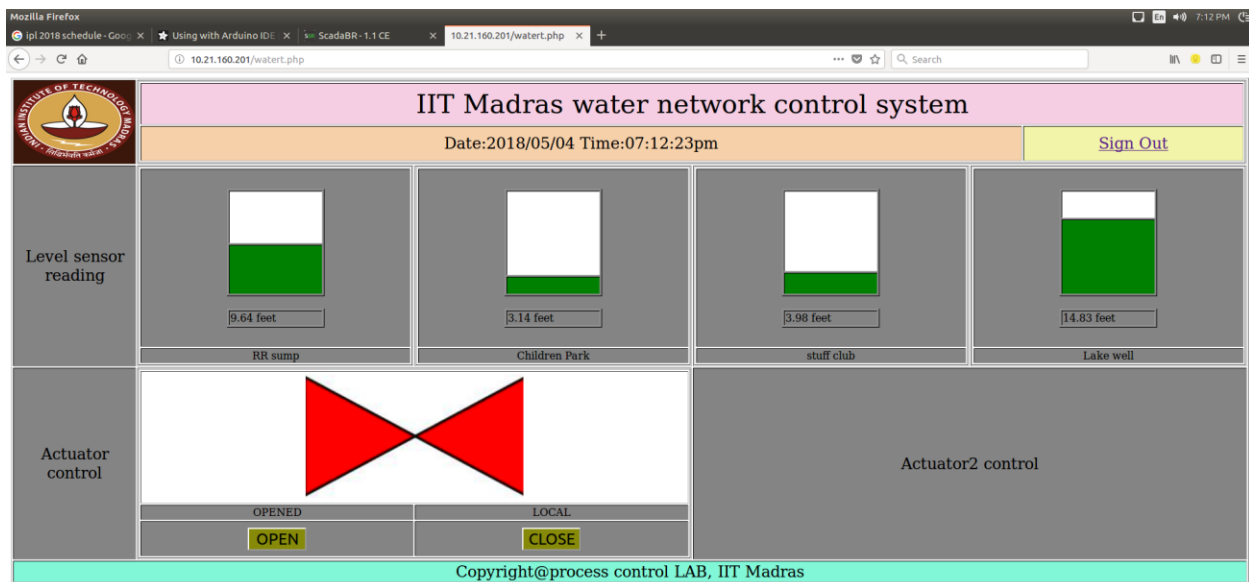


Figure-5.7 RR sump reading 9.64 feet on web application

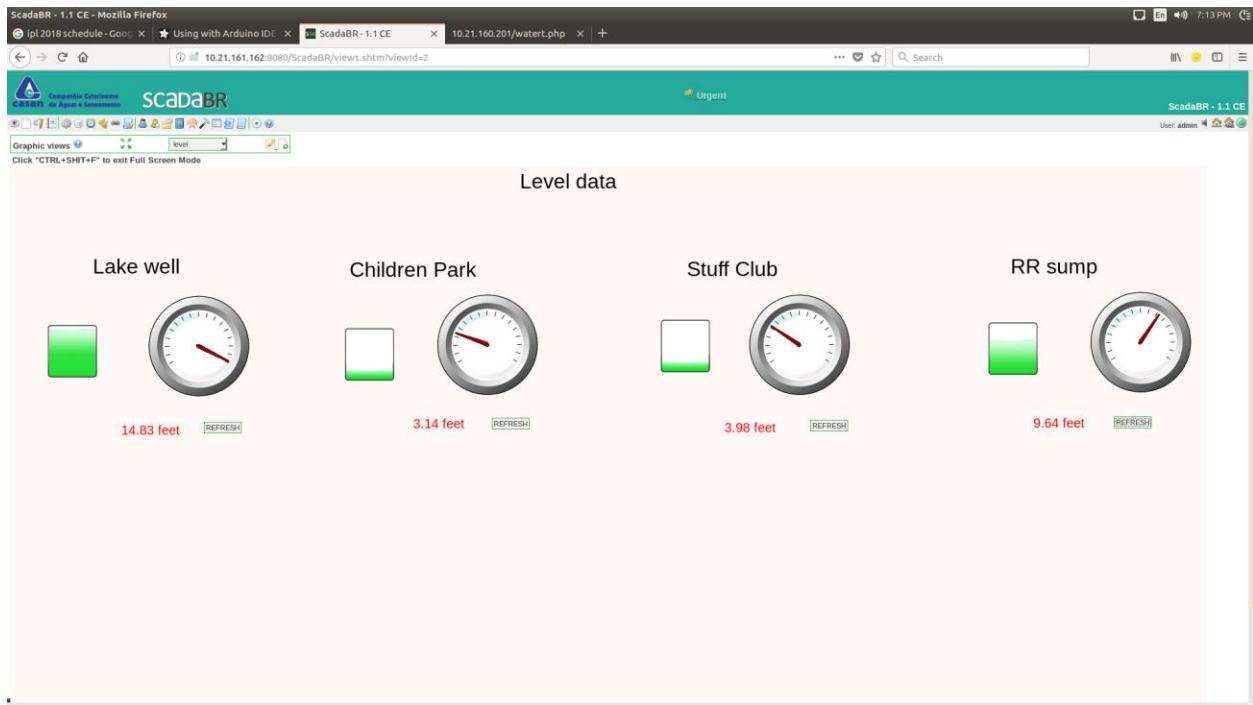


Figure-5.8 RR sump reading 9.64 feet on SCADABR

water_network

IIT MADRAS water network control

4 May 2018 19:14:09

Level readings

Lake well	14.83 ft
Children Park	3.14 ft
Stuf club	3.98 ft
RR sump	9.64 ft

Actuator status

Mode	IN LOCAL
status	FULLY OPENED

REFRESH READING

Actuator control

OPEN

CLOSE

Readings updated successfully

Figure-5.9 RR sump reading 9.64 feet on android application

CONCLUSION

This project presents a low cost IoT based solution using LoRa/GSM for monitoring and control of campus water distribution network. GSM is used as a redundant system to ensure continuous monitoring and control. Initial deployment results are encouraging. LoRA can easily be interfaced with industrial PLC (24 V DC) system for DI and DO data. Our GSM implementation can easily be used to implement 3G/4G using IoT protocol(MQTT/COAP). All codes have been uploaded on <https://github.com/tansquire>

REFERENCES

VavaliyaJaladhi, Bhavsar Dhruv, KavadiUtkarsha, Mohammad Mahroof, “OnlinePerformanc 5 essessment System for Urban Water Supply and Sanitation Services in India”, Aquatic Procedia, Volume 6, August 2016, Pages 51-63.

I. Stoianov, L. Nachman, S. Madden, T. Tokmouline, and M. Csail, “PIPENET: A Wireless Sensor NetwoRKfor Pipeline Monitoring,” Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on, pp. 264–273, 2007.

http://www.dragino.com/downloads/downloads/UserManual/LG01_LoRa_Gateway_User_Manual.pdf

<https://www.cookinghacks.com/form/viewtopic.php?f=43&t=5919><https://electronics.stackexchange.com/questions/274065/gsmmodule-gets-into-reboot-loop>

<https://arduino.stackexchange.com/questions/33214/is-the-a6-module-broken>

<https://www.youtube.com/watch?v=6snnAE3Pp9U>

<https://github.com/MarcoMartines/GSM-GPRS-GPS-Shield>

<https://www.youtube.com/channel/UC59K-uG2A5ogwIrHw4bmlEg>

<https://www.learnpython.org/>

<https://www.androidhive.info/2012/05/how-to-connect-android-with-php-mysql/><https://www.androidhive.info/2012/05/how-to-connect-android-with-phpmysql/>

<https://www.androidhive.info/2012/05/how-to-connect-android-with-php-mysql/>
<https://github.com/andresarmiento/modbus-arduino>