EXPT NO: 2
NAME: Rohit Murali
BATCH: D
ROLL NO: 2015120031

AIM: Study of GSM modem:

1. Install and configure minicom, wvdial and AT commands
2. Python scripting

SOFTWARE USED: Minicom, wvdial

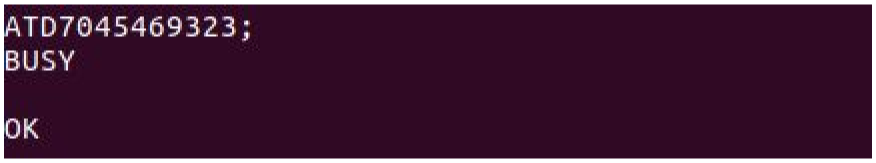
PROCEDURE:

1. Install and configure minicom, wvdial and AT commands
   STEPS:
   1. Open Terminal on the system and check if the minicom and wvdial is installed or not.
   2. If not installed then type on terminal:
      sudo apt-get install minicom -to install minicom
      sudo apt-get install wvdial -to install wvdial
   3. Now connect the modem and the adapter with the required sim either using USB or Serial port.
   4. Type on the terminal **sudo wvdial** to check the connectivity of the modem.
   5. If connected properly the terminal displays the modem connected and the port for the same.
   6. To configure the given modem, type on terminal **sudo minicom -s,** a configuration setup opens up.
   7. Go to Serial port setup and set the serial device according to your connection i.e ttyUSB or ttyS0
   8. Save setup as dfl and exit from the configuration setup.
   9. Type AT, this command is usually replied with an OK if the port and the module can connect correctly, else wise it comes back with a result code ERROR.

AT Commands:

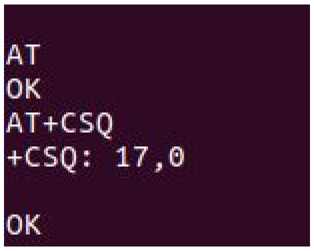
1. **AT**
   The "AT" or "at" prefix must be set at the beginning of each Command line. To terminate a Command line enter <CR>.

2. **ATD<N>**
   Originate call to phone number in memory



```
ATD7045469323;
BUSY

OK
```

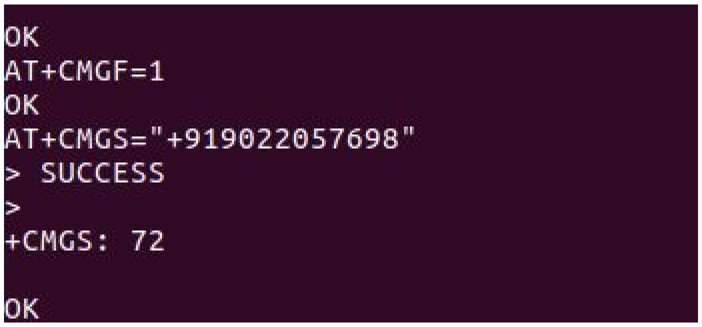
3. **ATZ**
   Resets Default Configuration
4. **AT+CSQ**
   Returns Signal Quality Report; Execution of Command returns received signal strength indication and channel bit error rate from the ME.



```
AT
OK
AT+CSQ
+CSQ: 17,0

OK
```

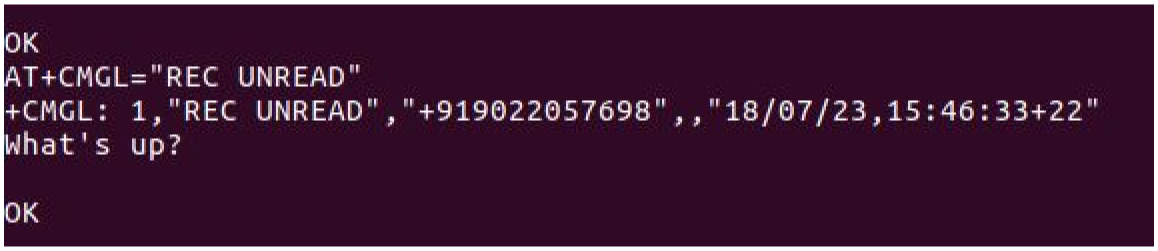
5. **AT+CMGF**
   Select SMS Message Format.0-PDU mode, 1- text mode

6. **AT+CMGS**
   Send SMS Message. TA sends message from a TE to the network. Message reference value <mr> is returned to the TE on successful message delivery.



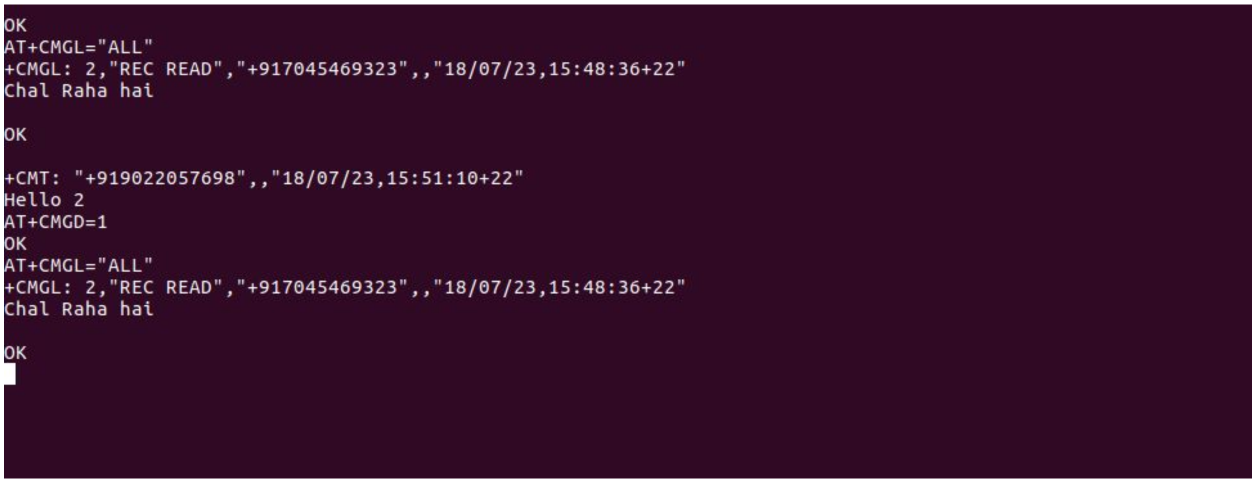
```
OK
AT+CMGF=1
OK
AT+CMGS="+919022057698"
> SUCCESS
>
+CMGS: 72

OK
```

7. **AT+CMGL**
   List all SMS messages present in TE.



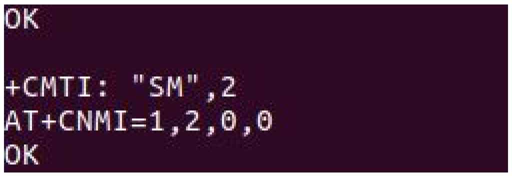
```
OK
AT+CMGL="REC UNREAD"
+CMGL: 1,"REC UNREAD","+919022057698",,"18/07/23,15:46:33+22"
What's up?

OK
```

8. **AT+CMGD**
   Delete Selected Message.



```
OK
AT+CMGL="ALL"
+CMGL: 2,"REC READ","+917045469323",,"18/07/23,15:48:36+22"
Chal Raha hai

OK

+CMT: "+919022057698",,"18/07/23,15:51:10+22"
Hello 2
AT+CMGD=1
OK
AT+CMGL="ALL"
+CMGL: 2,"REC READ","+917045469323",,"18/07/23,15:48:36+22"
Chal Raha hai

OK
```

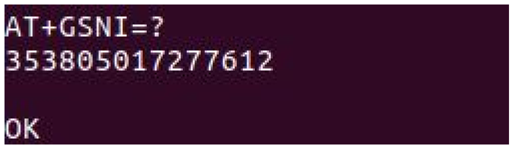
9. **AT+CNMI**
   New SMS Message Indications. TA selects the procedure for how the receiving of new messages from the network is indicated to the TE when TE is active. If TE is inactive, message receiving should be done as specified in GSM 03.38.



```
OK

+CMTI: "SM",2
AT+CNMI=1,2,0,0
OK
```

10. **AT+GSNI**

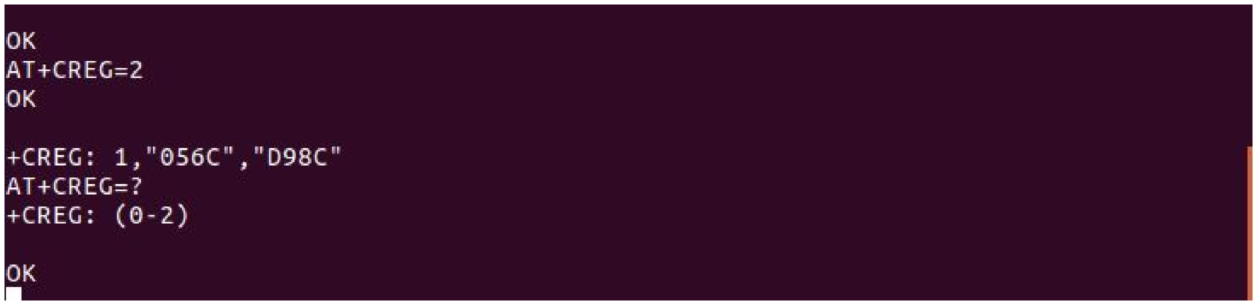
This command returns the SIM Identification Number unique to each TE.



```
AT+GSNI=?
353805017277612

OK
```

11. **AT+CREG**

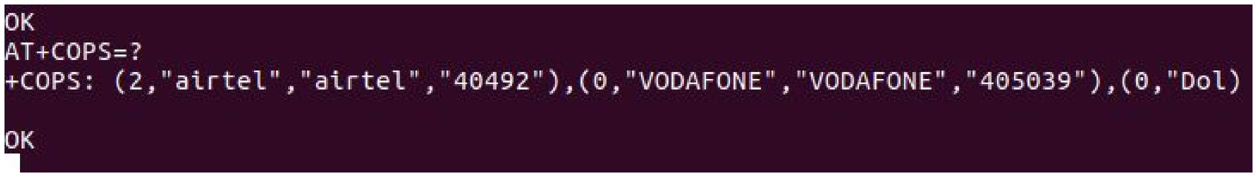
Network Registration. Returns List of supported networks.



```
OK
AT+CREG=2
OK

+CREG: 1,"056C","D98C"
AT+CREG=?
+CREG: (0-2)

OK
```

12. **AT+COPS**

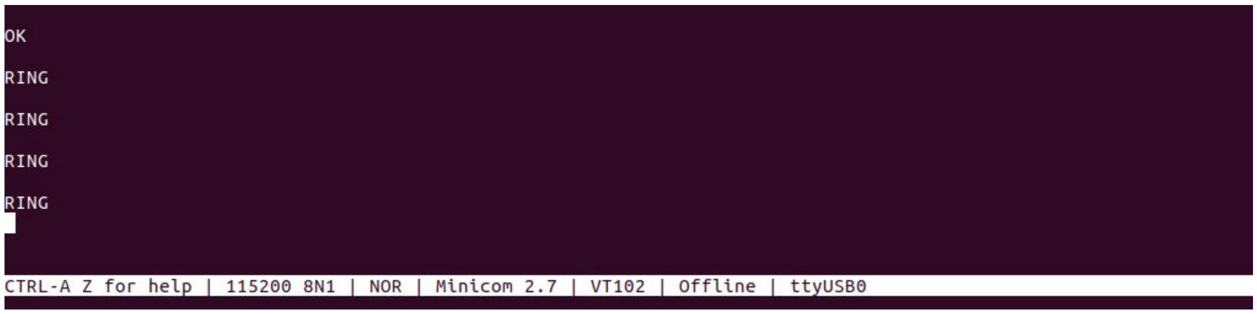
This commands allow you to read out the current network operator, display a list of available network operators or select a operator directly.



```
OK
AT+COPS=?
+COPS: (2,"airtel","airtel","40492"),(0,"VODAFONE","VODAFONE","405039"),(0,"Dol)

OK
```

13. **CALL**

Used for incoming calls.



```
OK

RING

RING

RING

RING


CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.7 | VT102 | Offline | ttyUSB0
```

Cell ID Finder:

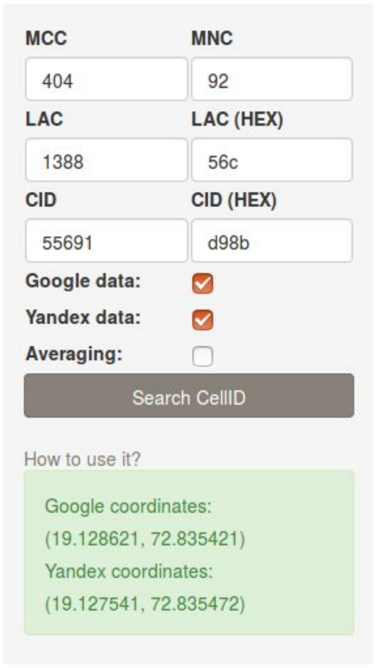
1. To check the current location of the user cell execute the following command:

AT+CREG? //Get current registration status If there is any error, CME error code is returned

AT+CREG=2 //Configure to return unsolicited result code when there is change in network registration status or change of network cell
For eg:
AT+CREG=2
OK
+CREG=1,"056C","D98B" //Returns "LAC" and "CID"

//List of supported responses
AT+COPS=?
+COPS: (2,"airtel","airtel","40492")
//Returns "40492" where "404" is "*MCC*" and "92" is "*MNC*"

2. To find the google coordinates of the user cell:
   Enter the following details in the given website:
   https://cellidfinder.com/
   MCC , MNC , LAC(hex value) and CID(hex value)
   It will then give the google coordinates of the user cell.

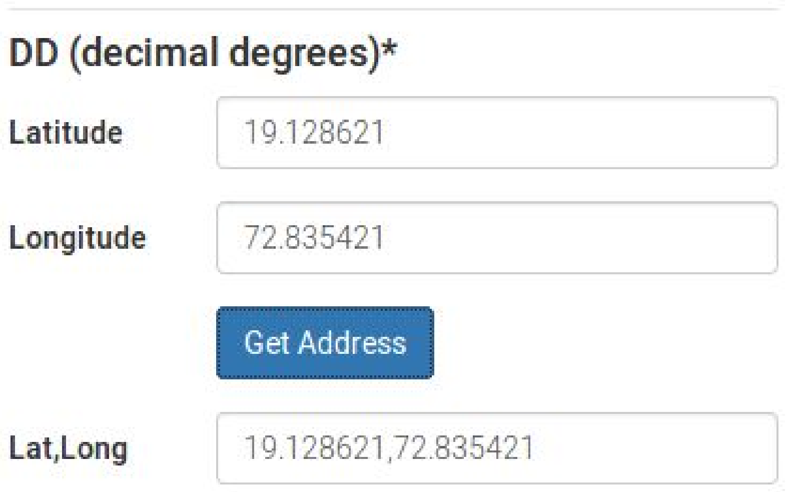


MCC	MNC
404	92
LAC	LAC (HEX)
1388	56c
CID	CID (HEX)
55691	d98b
Google data:	✓
Yandex data:	✓
Averaging:	

Google coordinates:
(19.128621, 72.835421)
Yandex coordinates:
(19.127541, 72.835472)

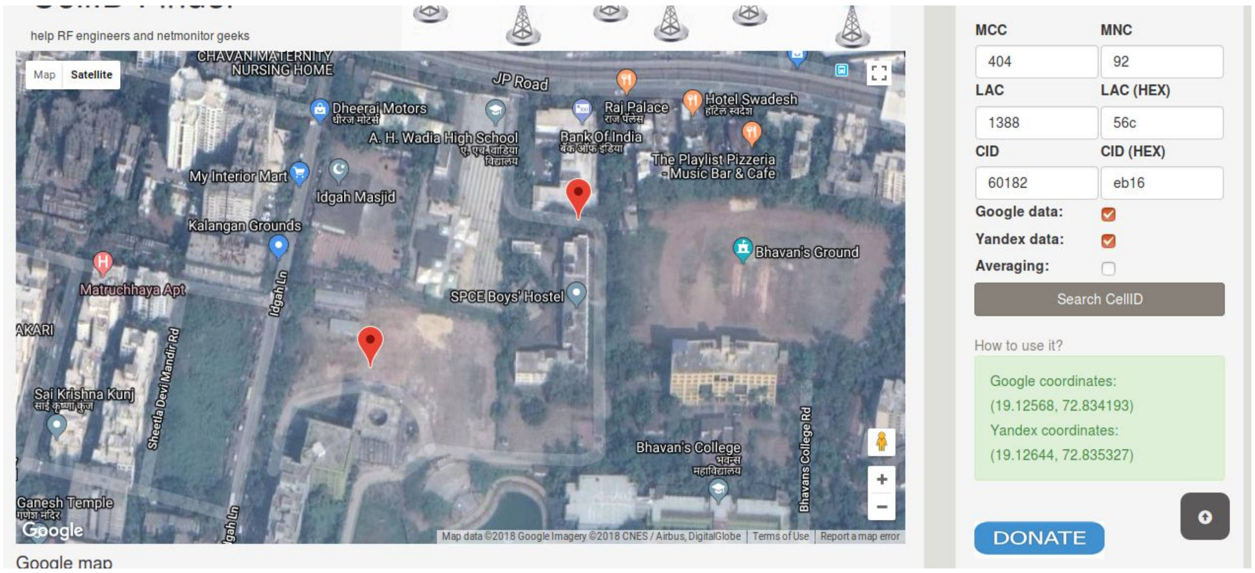
3. To get the address from the google coordinates:

Enter the google coordinates obtained from step 2 in the given website: https://www.gps-coordinates.net/


DD (decimal degrees)*

Latitude	19.128621
Longitude	72.835421
	Get Address
Lat,Long	19.128621,72.835421

You can get the address of the current cell user.



MCC	MNC
404	92
LAC	LAC (HEX)
1388	56c
CID	CID (HEX)
60182	eb16
Google data:	✓
Yandex data:	✓
Averaging:	☐

Search CellID

How to use it?

Google coordinates: (19.12568, 72.834193)
Yandex coordinates: (19.12644, 72.835327)

PYTHON SCRIPTING

PROCEDURE:

1. Importing required modules
    a. **Pyserial** - This module encapsulates the access for the serial port. It provides backends for Python running on Windows, OSX, Linux, BSD (possibly any POSIX compliant system) and IronPython. The module named "serial" automatically selects the appropriate backend. Since, we had python installed in **Anaconda** environment, the pyserial library can be installed using the command - **conda install -c anaconda pyserial**

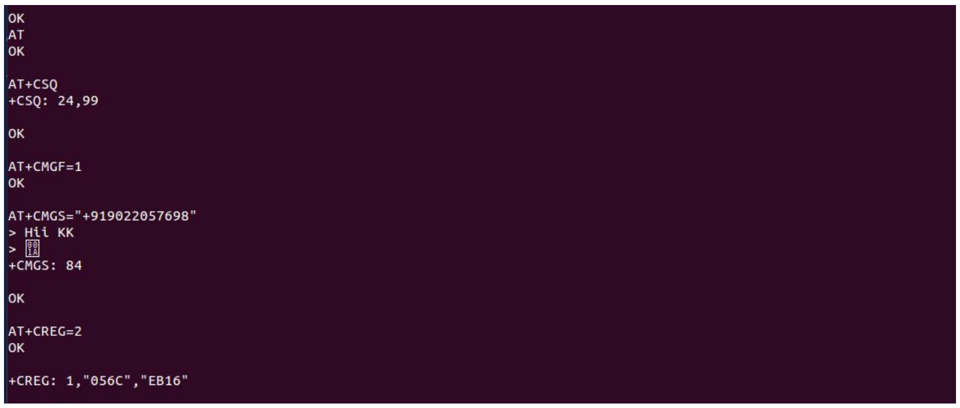
    b. Time - This module allows us to handle various operations regarding time, its conversions and representations, which find its use in various applications in life. The beginning of time is started measuring from 1 January, 12:00 am, 1970 and this very time is termed as “epoch” in Python.

2. Setting up a **serial connection** with the SIM900 module using the **Serial** method of the pyserial library which is imported using the **import** command of python along with the time library.
3. The parameters passed to the Serial method are device name, baud rate and timeout value.
4. Since, we are using the sleep function of time library we need to encase our entire operation in try….catch block as time.sleep() function throws exceptions.
5. We use the write() and read function of pyserial library for our serial communication with the GSM module.
6. We provide a waiting time after each query (after every write() function call in the program) for the GSM module to process the query and procure the output after which we read the output using the read() function for a specific amount of characters.
7. All the AT commands are fed sequentially to the GSM module and the output printed on the terminal.
8. The serial connection is closed explicitly in the finally block using the serial.close() function.

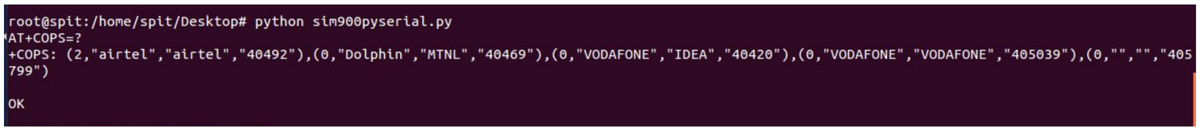
PYTHON SCRIPT:

```
import serial
import time

phone = serial.Serial("/dev/ttyUSB0",  9600, timeout = 5)

try:
        phone.write(b'AT\r')              # initial AT command
        time.sleep(0.2)
        print(phone.read(256))

        phone.write(b'AT+CSQ\r')      # AT command for getting the signal strength
        time.sleep(0.2)
        print(phone.read(256))

        phone.write(b'AT+CMGF=1\r')          # setting the texting mode as string mode
        time.sleep(0.2)
        print(phone.read(256))

        phone.write(b'AT+CMGS="+919022057698"\r')      #inputting the recipient phone number
        time.sleep(0.2)

        phone.write(b'Hii KK\r')         # SMS body
        time.sleep(0.2)

        phone.write(chr(26))              # Sending CTRL+Z command serially
        print(phone.read(256))

        phone.write(b'AT+CREG=2\r')          # gets the base staion details
        time.sleep(5)
        print(phone.read(256))

        phone.write(b'AT+COPS=?\r')          # gets the total information of the signal
        time.sleep(40)
        print(phone.read(2560))

finally:
        phone.close()
```

PYTHON OUTPUT:



```
OK
AT
OK

AT+CSQ
+CSQ: 24,99

OK

AT+CMGF=1
OK

AT+CMGS="+919022057698"
> Hii KK
> 
+CMGS: 84

OK

AT+CREG=2
OK

+CREG: 1,"056C","EB16"
```



```
root@spit:/home/spit/Desktop# python sim900pyserial.py
AT+COPS=?
+COPS: (2,"airtel","airtel","40492"),(0,"Dolphin","MTNL","40469"),(0,"VODAFONE","IDEA","40420"),(0,"VODAFONE","VODAFONE","405039"),(0,"","","405
799")

OK
```

CONCLUSION:

1. Wvdial and Minicom were used for modem initialisation and control and their various commands for serial communication setup were studied.
2. AT Commands for sending, receiving text messages and phone calls along with commands for network identification were studied and implemented on Minicom Terminal Emulator.
3. A script on Python programming language was developed for sending the AT Commands to the modem. The output response was redirected to Python console using methods from ‘serial’ library in Python.