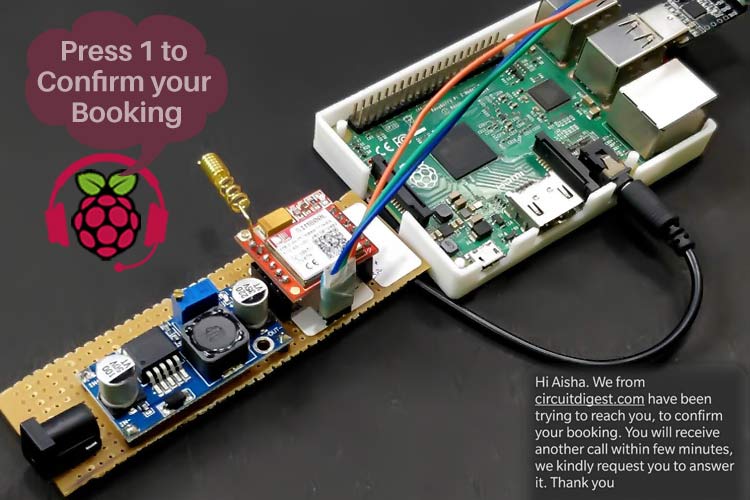
https://circuitdigest.com/microcontroller-projects/build-an-ivr-system-to-make-automated-phone-calls-and-messages-using-raspberry-pi-sim800l

[Build an IVR System to make Automated Phone calls and send Messages using Raspberry Pi and SIM800L](https://circuitdigest.com/microcontroller-projects/build-an-ivr-system-to-make-automated-phone-calls-and-messages-using-raspberry-pi-sim800l)

[RASPBERRY PI](https://circuitdigest.com/simple-raspberry-pi-projects-for-beginners)

By[Aswinth Raj](https://circuitdigest.com/users/aswinth-raj) Feb 11, 2021[2](https://circuitdigest.com/microcontroller-projects/build-an-ivr-system-to-make-automated-phone-calls-and-messages-using-raspberry-pi-sim800l#comments)



Raspberry Pi IVR System

Raspberry Pi, as we know is a powerful development board that provides a decent amount of computing power in the size of our palm. This combined with the flexibility of python could help us develop many small gadgets that can solve day to day problems.

**IVR system using Raspberry Pi**, I have to make phone calls and send messages using Raspberry Pi and when the receiver picks up the call, I should play a pre-recorded audio and get DTMF inputs as feedback.

We have previously built a [Raspberry Pi based Call and Text](https://circuitdigest.com/microcontroller-projects/raspberry-pi-phone-by-interfacing-gsm-module) project, and an [Automatic Call answering machine](https://circuitdigest.com/microcontroller-projects/automatic-call-answering-machine-using-arduino), you can also check them out if interested.

Why GSM Module? Why not make phone calls directly with API?

Soon I realized that making phone calls with Raspberry Pi using API can get very expensive in the long run. There are many service providers like Twilio, Nexmo, Plivo, etc. which can help you make phone calls easily using python and Pi. These services are great, but there are two main problems here,

* First is the cost, the charge per minute basis ($0.03 to $0.05) and making 100+ calls per day.
* The second problem is that the phone number from which we make these calls should also be purchased and maintained separately. Again, these numbers are virtual so you do not have the flexibility of using an existing number, and based on your location you might have to pay extra for getting a number that belongs to your country.

What is the Cheapest way to make phone calls using Raspberry Pi?

After looking into the pricing plans of all API-based phone call services, it was very clear that using a GSM module like SIM800L with Pi and making phone calls with network service provides like Airtel or Jio is definitely the most cost-effective way. So, in this project, we will be **interfacing SIM800L with Raspberry Pi** and we will be making automated phone calls and sending automated messages using it.

Which is the best GSM module to use with Raspberry Pi for making IVR calls?

With the craze for 5G and IoT around, there have been many new cool GSM modules launched in the market that can work with 5G networks with very low power. But my idea was to use something that is very popular and is easily available from a local hobby shop. So, I decided to try the Ai-thinker A9 and the popular SIM800L. The **Ai-thinker A9 GSM module** worked without any problem as the first power up and was butter smooth even when powered only with my Laptop USB port. But the biggest problem was Ai Thinker A9 GSM module does not support DTMF detection and also the in-line audio input did not work very well.

So, I had to move to the **SIM800L GSM Module**which is a power-hungry beast, but just feed it enough power and it works just perfectly for this application. It supports in-line audio for us to play automated voice and also supports in-built DTMF detection so no need for additional DTMF hardware to detect which keys the customer is pressing. Again, the major drawback of this module is that it needs high power, we will discuss more on this later.

Which SIM card can I use for SIM800L GSM module in India?

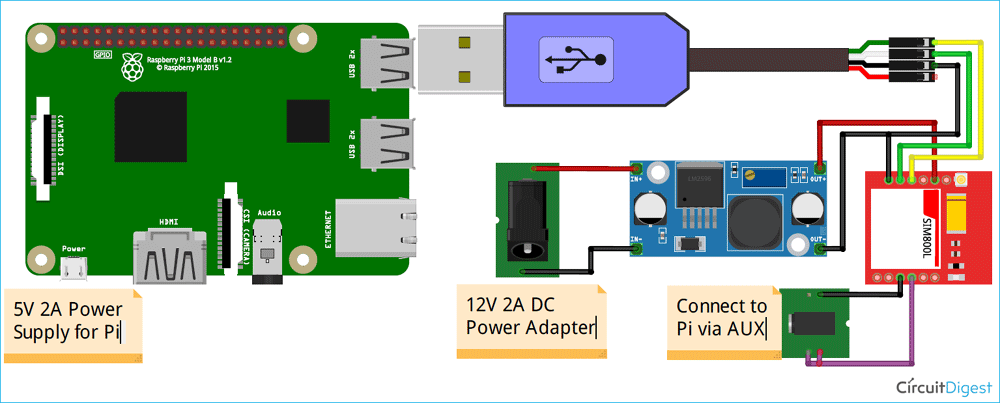
SIM800L supports only 2G networks and cannot work on 3G or 4G. Now, I am writing this in 2021 from India, and there are not many 2G options here to select from other than Airtel and BSNL. So, just buy or use an existing Airtel SIM card and we should be fine. I purchased a new Airtel SIM card and used it with my SIM800L after activating it using my mobile phone.

Materials Required to build our Raspberry Pi IVR system

* Raspberry Pi (2 or 3 or 4) running latest version of Buster
* SIM800L GSM Module
* 2G SIM Card (Airtel)
* Aux Cable
* LM2596 Buck Converter Module
* USB to TTL Converter
* 12V 2A Adapter
* Perf Board
* Berg Sticks
* Connecting wires
* Soldering Kit

**Note:**The IVR system that we build here only needs external hardware to execute Python code and provide Audio output,. The external hardware in this project is a Raspberry Pi, but you can also use a computer to do that.

Circuit Diagram to Build an IVR system using SIM800L

[](https://circuitdigest.com/fullimage?i=circuitdiagram_mic/Raspberry-Pi-IVR-System-Circuit-Diagram.png)

**Powering the SIM800L Module:** The most important thing to note here is how you power the SIM800L module. The SIM800L module operated between 3.7V to 4.2V (designed for Li-po batteries). The ideal operating voltage is around 4V. We have used LM2596 Buck converter to convert the 12V 2A input from the adapter to 4V as required by the SIM800L module. Care should be taken that the wires connecting the buck module with SIM800L are thick and short to carry high current easily. If your power connections are not made adequate, the module will reset itself when powered and will throw garbage valued on serial communication. So, make sure the SIM800L module is powered with a proper 12V 2A adapter and the wires are thick enough to carry high current without providing much resistance. If you face any problem, directly connect a fully charged Li-po battery to the Vcc and Gnd pins of SIM800L module, this will solve the power issues for sure.

**Serial Communication between SIM800L and Raspberry Pi:**If you have followed our previous [GSM based projects](https://circuitdigest.com/tags/gsm), you should already be aware that we have to communicate using “AT Commands” to the SIM800L module. AT commands can be used to make/receive calls, send/read messages, detect a keypress, and more. In this project, we will be using Python from Raspberry Pi to send these AT commands to the GSM SIM800L module. To do that, we have used a USB to TTL converter to connect the Rx and TX pin of the SIM800L module to the USB port of the Raspberry Pi.

**Audio input to SIM800L module from Raspberry Pi:**The SIM800L module supports microphone input through the MIC+ and MIC- pins on the module. Once the call is made, any audio input given to these pins will be played on the call receiver's phone. However, these pins are actually meant to connect a microphone and, in our case, we need to play a pre-recorded voice from the Raspberry. The audio output from the Pi through the 3.5mm jack is called line-out audio and we need to convert it to mic level audio to be able to receive it on the GSM module. Now, you can build a line to mic converter circuit to get this done professionally, but I just connected them directly and reduced the volume level on the Pi to just 2 points. I got it working this way without any problem.

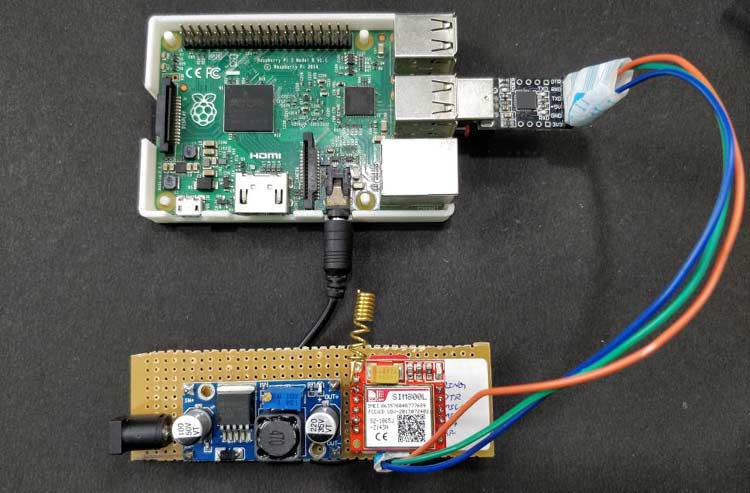
I built the complete circuit on a perf board and made sure the power connections had enough lead to provide low resistance connectivity. My board when soldered looks like this.

**Important:** LM2596 is a variable buck converter so before you use it, make sure you set the output voltage to 4V using the on-bord trimpot. Anything more than 4.2V can kill the SIM800L module permanently.

|  |  |
| --- | --- |
| IVR system using SIM800L | SIM800L IVR System |

Once you reach here, just power the board and insert the SIM card. Again make sure you have inserted the SIM card in the right orientation and the antenna is properly installed. If everything is done right you should notice the onboard led on the SIM800L board blinking once every 3 seconds. This means that the SIM800L module is able to set-up a network with our SIM card.

This ensures that our powering circuit is working properly. Now, we can connect the board without Raspberry Pi and start writing our Python script.



Raspberry Pi Python Code to Make Calls and Send SMS using SIM800L Module

The complete Python code for the Raspberry Pi IVR system can be found at the bottom of this page. The code is relatively big and hence it is not possible to explain every single step. However, the code is well-spaced with comments and methods for you to easily read and understand. The code along with the audio files can also be downloaded from the link below.

[***Python Code for Raspberry Pi IVRS System***](https://circuitdigest.com/sites/default/files/Python-code-for-Raspberry-pi-IVRS-system.zip)

We start the program by importing the required header files. We have used the serial package to enable serial communication between Pi and SIM800L, the *pygame* package is used to play music, in our case, play the audio files. Apart from that, we have the time package to create delays. All three packages are pre-installed in the Buster OS, so you need not install anything new.

import serial #for serial communication with GSM SIM800L

import time

import pygame #to play music

**def SIM800 (Command):**This function is used to send AT command from PI to SIM800L and get a response for that AT command. All the AT commands sent to SIM800L should end with “*\r\n*” and should be encoded in ASCII. This function appends all our AT commands with “\r\n” and encodes them into ASCII form before sending them to SIM800L. Then it also reads the response and decodes the ASCII values so that we can use them in our program.

#Speak with SIM800 -> gets AT command return as response

def SIM800(command):

AT\_command = command + "\r\n"

ser.write(str(AT\_command).encode('ascii'))

time.sleep(1)

if ser.inWaiting() > 0:

echo = ser.readline() #waste the echo

response\_byte = ser.readline()

response\_str = response\_byte.decode('ascii')

return (response\_str)

else:

return ("ERROR")

**def wait\_for\_SIM800():**This function is also very similar to the above function, but it does not send any value to SIM800, it just waits for the SIM800L to respond with something. When it gets a reply it returns it as a result.

#checks if SIM800L is speaking and returns it as response

def wait\_for\_SIM800():

echo = ser.readline() # waste the echo

response\_byte = ser.readline()

response\_str = response\_byte.decode('ascii')

return (response\_str)

**def Init\_GSM():** The initialize GSM function prepares the GSM module for IVR operations. It first checks for the module by sending an “AT” and waiting for an “OK” and then sends some specific AT commands to put the GSM module in messaging mode and DTMF receiver mode. Apart from that, it also disables all notifications so that we won’t be disturbed by text message notifications during a call.

#Checks SIM800L status and connects with ShopifyAPI

def Init\_GSM():

if "OK" in SIM800("AT"):

if ("OK" in (SIM800("AT+DDET=1"))) and ("OK" in (SIM800("AT+CNMI =0,0,0,0,0"))) and ("OK" in (SIM800("AT+CMGF=1"))) and ("OK" in (SIM800("AT+CSMP=17,167,0,0"))): # enble DTMF / disable notifications

print("SIM800 Module -> Active and Ready")

else:

print("------->ERROR -> SIM800 Module not found")

**def play\_wav (file\_name):** The next function is used to play the *wav files* once the call has been answered. We have stored pre-recorded *wav files* like, “*intro.wav*”, “*confirm.wav*”, “*cancel.wav*” etc. We have to play each of these files based on the keypad response from the user. This play\_wav function can be used to play any *wav file* of our choice. Make sure these *wav files* are saved in the same directory of your Python code.

#plays the given wav file #8000Mhz mono audio WAV works best on SIM800L

def play\_wav(file\_name):

pygame.mixer.init(8000)

pygame.mixer.music.load(file\_name)

pygame.mixer.music.play()

#while pygame.mixer.music.get\_busy() == True:

#continue

**def Call\_response\_for (phone\_number):**This is the most important function in the program. It gets the phone\_number to which a call has to be made and provides the response for that call. The response can be anything like *NOT\_REACHABLE, CALL\_REJECTED, CONFIRMED, CANCELED*, etc. The function uses AT commands to make the call to a given phone number and plays the recorded voice. Then it checks for the DTMF response from the caller and based on the response it plays the relative recorded voice and finally tells us what the caller has selected. If the caller rejected the call or was not reachable, the function will also provide that as a response.

# Makes a call to given number and returns NONE, NOT\_REACHABLE, CALL\_REJECTED, REJECTED\_AFTER\_ANSWERING, REQ\_CALLBACK,CANCELED, CONFIRMED

def Call\_response\_for (phone\_number):

AT\_call = "ATD" + phone\_number + ";"

response = "NONE"

time.sleep(1)

ser.flushInput() #clear serial data in buffer if any

if ("OK" in (SIM800(AT\_call))) and (",2," in (wait\_for\_SIM800())) and (",3," in (wait\_for\_SIM800())):

print("RINGING...->", phone\_number)

call\_status = wait\_for\_SIM800()

if "1,0,0,0,0" in call\_status:

print("\*\*ANSWERED\*\*")

ser.flushInput()

play\_wav("intro.wav")

time.sleep(0.5)

dtmf\_response = "start\_over"

while dtmf\_response == "start\_over":

play\_wav("press\_request.wav")

time.sleep(1)

dtmf\_response = wait\_for\_SIM800()

if "+DTMF: 1" in dtmf\_response:

play\_wav("confirmed.wav")

response = "CONFIRMED"

hang = SIM800("ATH")

break

if "+DTMF: 2" in dtmf\_response:

play\_wav("canceled.wav")

response = "CANCELED"

hang = SIM800("ATH")

break

if "+DTMF: 9" in dtmf\_response:

play\_wav("callback\_response.wav")

response = "REQ\_CALLBACK"

hang = SIM800("ATH")

break

if "+DTMF: 0" in dtmf\_response:

dtmf\_response = "start\_over"

continue

if "+DTMF: " in dtmf\_response:

play\_wav("invalid\_input.wav")

dtmf\_response = "start\_over"

continue

else:

response = "REJECTED\_AFTER\_ANSWERING"

break

else:

#print("REJECTED")

response = "CALL\_REJECTED"

hang = SIM800("ATH")

time.sleep(1)

#ser.flushInput()

else:

#print("NOT\_REACHABLE")

response = "NOT\_REACHABLE"

hang = SIM800("ATH")

time.sleep(1)

#ser.flushInput()

ser.flushInput()

return (response)

**def send\_message (message, recipient**): Apart from making calls and getting the response, this program also allows us to send a message, the send\_message function is used to do exactly that. It gets the message and recipient's phone number and sends the message.

#Receives the message and phone number and send that message to that phone number

def send\_message(message, recipient):

ser.write(b'AT+CMGS="' + recipient.encode() + b'"\r')

time.sleep(0.5)

ser.write(message.encode() + b"\r")

time.sleep(0.5)

ser.write(bytes([26]))

time.sleep(0.5)

print ("Message sent to customer")

time.sleep(2)

ser.flushInput() # clear serial data in buffer if any

**def incoming\_call():** The last function I made for this project is to detect the caller number of an incoming call. Since this SIM card is going to make calls to new people, some might try to call back to this number. In that case, this function can be used to check from which number we are receiving an incoming call and then later send a message or call them back as required.

def incoming\_call():

while ser.in\_waiting: #if I have something in the serial monitor

print ("%%Wait got something in the buffer")

ser.flushInput()

response = SIM800("ATH") #cut the incoming call

if "+CLCC" in response:

cus\_phone = response[21:31]

print("%%Incoming Phone call detect from ->", cus\_phone)

return (cus\_phone)

else:

print("Nope its something else")

return "0"

return "0"

Now that all the functions are defined, it's time to write the main code in which we will use all these functions to do something beautiful. Now for demonstration purpose, I am just going to hard code the customer name and customer phone number, but you can get it from an Shopify API call or read from a spreadsheet as required. I am entering the customer name as “AISHA” and number as “9877XXXXXX” for testing

cus\_name = "Aisha"

cus\_phone = "968837XXXX"

Inside the main infinite while loop, we will be starting a serial communication at 9600 baud rates with 15 seconds time out. Now, some SIM800L modules might work in different baud rates, make sure you enter the right COM directory and baud rate here.

# COM defanition for windows -> Should change for Pi

ser = serial.Serial("/dev/ttyUSB0", baudrate=9600, timeout=15) # timeout affects call duration and waiting for response currently 30sec

print("Established communication with", ser.name)

Next, we will make the phone call and get the required response from the customer, then based on the response, we will send a message to the customer. For example, if the response is confirmed, we will send a message regarding that, similarly, we can change the message for a different response from the customer.

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_IVR START\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

response = Call\_response\_for(cus\_phone) #place a call and get response from customer

print ("Response from customer => ", response)

if response == "CONFIRMED":

    text\_message = "Hi " + cus\_name + ". Your booking has been confirmed.Thank you. -Circuitdigest"

    send\_message(text\_message, cus\_phone)

if response == "CANCELED":  # if the response was to cancel

    text\_message = "Hi " + cus\_name + ". Sorry that you have decided to cancel your booking. If you cancled by mistake, kindly contact us through phone. -Circuitdigest"

    send\_message(text\_message, cus\_phone)

if ((response == "CALL\_REJECTED") or (response == "REJECTED\_AFTER\_ANSWERING")):  # if the response was rejected

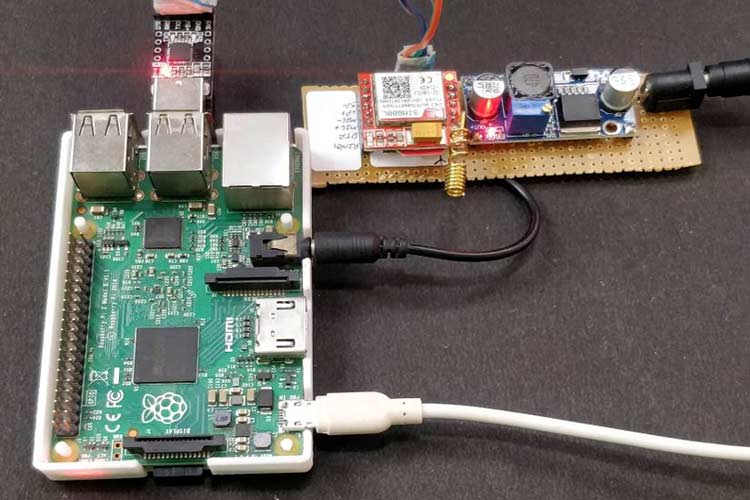
    text\_message = "Hi " + cus\_name + ". We from circuitdigest.com have been trying to reach you, to confirm your booking. You will receive another call within few minutes, we kindly request you to answer it. Thank you"

    send\_message(text\_message, cus\_phone)

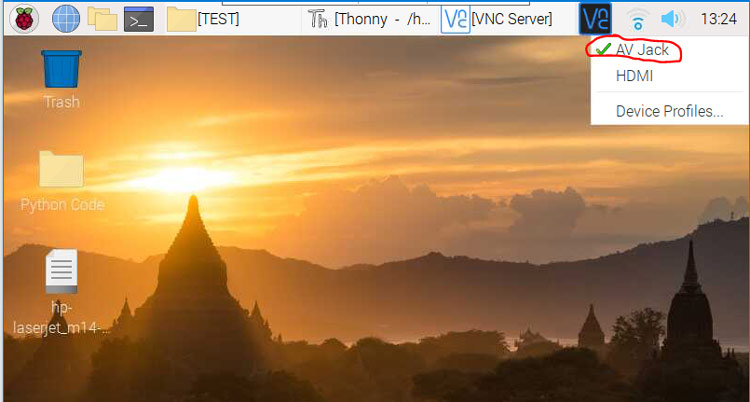
print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_IVR END\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

Testing our Pi IVR System

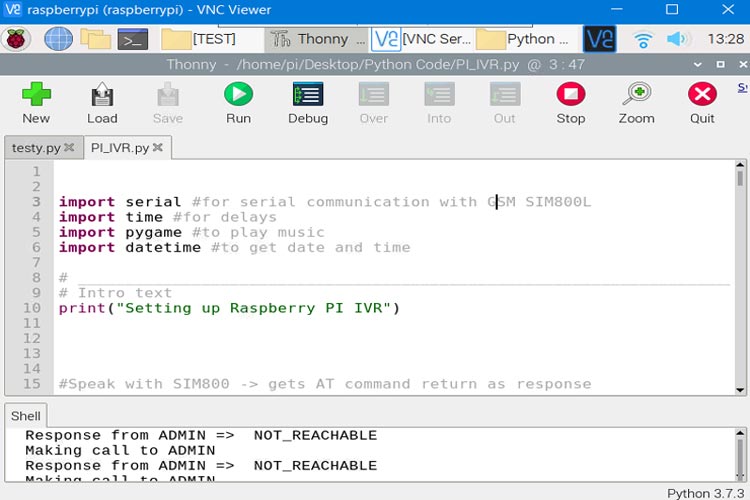
The complete working can be found in the video at the bottom of this page. Just make sure the connections are made correct and power up both the Raspberry Pi and GSM module board.



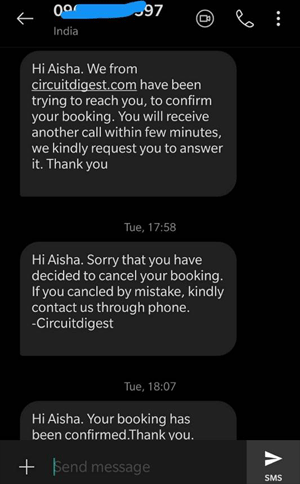
Before launching the program, make sure the correct COM port is mentioned in the code in my case it was “*/dev/ttyUSB0*”. Then make sure the audio is set to AV jack by right-clicking on the speaker icon and also make sure the audio volume is set low.



In the next step, just change the phone number and customer name as per your choice or tweak the program to fetch the number and name from an excel or cloud API, and our automated IVR is all set for action.



The program will make calls to the given number and get dtmf feedback from the receiver. It will also play related audio files and send a text message based on the response given by the receiver. Few template messages received by my phone is shown below.



You can easily change the audio files and messages as required by your application. Hope you enjoyed the project and learned something useful. The complete working of this project is demonstrated in the video attached below. If you have any questions, please leave them in the comment section below or use the [forums](https://circuitdigest.com/forums).

Code

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142 | import serial #for serial communication with GSM SIM800L  import time  import pygame #to play music  # \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_#  # Intro text  print("Setting up Raspberry PI IVR")  #Speak with SIM800 -> gets AT command return as response  def SIM800(command):      AT\_command = command + "\r\n"      ser.write(str(AT\_command).encode('ascii'))      time.sleep(1)      if ser.inWaiting() > 0:          echo = ser.readline() #waste the echo          response\_byte = ser.readline()          response\_str = response\_byte.decode('ascii')          return (response\_str)      else:          return ("ERROR")  #checks if SIM800L is speaking and returns it as response  def wait\_for\_SIM800():      echo = ser.readline()  # waste the echo      response\_byte = ser.readline()      response\_str = response\_byte.decode('ascii')      return (response\_str)  #Checks SIM800L status and connects with ShopifyAPI  def Init\_GSM():      if "OK" in SIM800("AT"):          if ("OK" in (SIM800("AT+DDET=1"))) and ("OK" in (SIM800("AT+CNMI =0,0,0,0,0"))) and ("OK" in (SIM800("AT+CMGF=1"))) and ("OK" in (SIM800("AT+CSMP=17,167,0,0"))):  # enble DTMF / disable notifications              print("SIM800 Module -> Active and Ready")      else:          print("------->ERROR -> SIM800 Module not found")  #plays the given wav file #8000Mhz mono audio WAV works best on SIM800L  def play\_wav(file\_name):      pygame.mixer.init(8000)      pygame.mixer.music.load(file\_name)      pygame.mixer.music.play()      #while pygame.mixer.music.get\_busy() == True:          #continue  # Makes a call to given number and returns NONE, NOT\_REACHABLE, CALL\_REJECTED, REJECTED\_AFTER\_ANSWERING,  REQ\_CALLBACK,CANCELED, CONFIRMED  def Call\_response\_for (phone\_number):      AT\_call = "ATD" + phone\_number + ";"      response = "NONE"      time.sleep(1)      ser.flushInput() #clear serial data in buffer if any      if ("OK" in (SIM800(AT\_call))) and (",2," in (wait\_for\_SIM800())) and (",3," in (wait\_for\_SIM800())):          print("RINGING...->", phone\_number)          call\_status = wait\_for\_SIM800()          if "1,0,0,0,0" in call\_status:              print("\*\*ANSWERED\*\*")              ser.flushInput()              play\_wav("intro.wav")              time.sleep(0.5)              dtmf\_response = "start\_over"              while dtmf\_response == "start\_over":                  play\_wav("press\_request.wav")                  time.sleep(1)                  dtmf\_response = wait\_for\_SIM800()                  if "+DTMF: 1" in dtmf\_response:                      play\_wav("confirmed.wav")                      response = "CONFIRMED"                      hang = SIM800("ATH")                      break                  if "+DTMF: 2" in dtmf\_response:                      play\_wav("canceled.wav")                      response = "CANCELED"                      hang = SIM800("ATH")                      break                  if "+DTMF: 9" in dtmf\_response:                      play\_wav("callback\_response.wav")                      response = "REQ\_CALLBACK"                      hang = SIM800("ATH")                      break                  if "+DTMF: 0" in dtmf\_response:                      dtmf\_response = "start\_over"                      continue                  if "+DTMF: " in dtmf\_response:                      play\_wav("invalid\_input.wav")                      dtmf\_response = "start\_over"                      continue                  else:                      response = "REJECTED\_AFTER\_ANSWERING"                      break          else:              #print("REJECTED")              response = "CALL\_REJECTED"              hang = SIM800("ATH")              time.sleep(1)              #ser.flushInput()      else:          #print("NOT\_REACHABLE")          response = "NOT\_REACHABLE"          hang = SIM800("ATH")          time.sleep(1)          #ser.flushInput()      ser.flushInput()      return (response)  #Receives the message and phone number and send that message to that phone number  def send\_message(message, recipient):      ser.write(b'AT+CMGS="' + recipient.encode() + b'"\r')      time.sleep(0.5)      ser.write(message.encode() + b"\r")      time.sleep(0.5)      ser.write(bytes([26]))      time.sleep(0.5)      print ("Message sent to customer")      time.sleep(2)      ser.flushInput()  # clear serial data in buffer if any  def incoming\_call():      while ser.in\_waiting: #if I have something in the serial monitor          print ("%%Wait got something in the buffer")          ser.flushInput()          response = SIM800("ATH") #cut the incoming call          if "+CLCC" in response:              cus\_phone = response[21:31]              print("%%Incoming Phone call detect from ->", cus\_phone)              return (cus\_phone)          else:              print("Nope its something else")              return "0"      return "0"  cus\_name = "Aisha"  cus\_phone = "96883XXXXX"  while (1): #Infinite loop      # COM defanition for windows -> Should change for Pi      ser = serial.Serial("/dev/ttyUSB0", baudrate=9600, timeout=15)  # timeout affects call duration and waiting for response currently 30sec      print("Established communication with", ser.name)      Init\_GSM() #check if GSM is connected and initialize it      print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_IVR START\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")      response = Call\_response\_for(cus\_phone) #place a call and get response from customer      print ("Response from customer => ", response)      if response == "CONFIRMED":          text\_message = "Hi " + cus\_name + ". Your booking has been confirmed. Thank you!!. -Circuitdigest"          send\_message(text\_message, cus\_phone)      if response == "CANCELED":  # if the response was to cancel          text\_message = "Hi " + cus\_name + ". Sorry that you have decided to cancel your booking. If you cancled by mistake, kindly contact us through phone. -Circuitdigest"          send\_message(text\_message, cus\_phone)      if ((response == "CALL\_REJECTED") or (response == "REJECTED\_AFTER\_ANSWERING")):  # if the response was rejected          text\_message = "Hi " + cus\_name + ". We from circuitdigest.com have been trying to reach you, to confirm your booking. You will receive another call within few minutes, we kindly request you to answer it. Thank you"          send\_message(text\_message, cus\_phone)      print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_IVR END\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")      ser.close()      time.sleep (5) |