

# BLUETOOTH AUDIO DEVICE

**USING PYTHON & RASPBERRY PI**

7/24/2018

SIEMENS TECHNOLOGY AND SERVICES PVT.LTD

Under the guidance of:

Mr. K.C.TEJASWI

By:

BIMAL PAUL  
NAGASHREE.B.S  
NEHA SINGH

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| SL.NO | CONTENTS | PAGE NO |
| 1 | Acknowledgements | 3 |
| 2 | Problem Statement And Abstract | 4 |
| 3 | Initial Proposals | 5 |
| 4 | The Prototype | 6 |
| 5 | Stages Of Development | 7 |
| 6 | Functionality | 20 |
| 7 | Running Script On Booting The Pi | 22 |
| 8 | Challenges Faced | 23 |
| 9 | Potential Improvements | 23 |
| 10 | References | 24 |
|  |  |  |

# Acknowledgements

We would like to express our gratitude to Mr. K. C. Tejaswi and Mr. Krishna Chaitanya M for having guided and encouraged us, and helped us get familiarized with technologies like the Raspberry Pi during the course of this project and our internship. We deeply value their inputs which helped us improve our work.

We would also like to thank Mr. Sanjeev Kumar and Mr. Ajith Unnikrishnan for all their support during our internship.

We extend our gratitude to the WinCC UA Cloud Services team for their advice and support during the course of our internship.

Finally, we would like to thank Siemens Technology and Services Pvt. Ltd and the HR Department for providing us this valuable opportunity.

# Problem Statement

Create an interactive audio device that can be installed in automation plants to provide the customer, information about each sub system that is a part of the plant.

# Abstract

The project aims at creating a device that will provide the customer, information that he desires about every sub system in an automation plant.

The device (Bluetooth headsets) speaks to the customer to ask him what information he desires to know. Based on the customer’s answer, it speaks out the information. The customer can control which sub system he want to know about first. Customer can traverse the systems in the order he likes.

The headsets can also be used by him to control the lighting in the room. The customer can set the lighting mode and then start acquiring information from the sub systems. Once he starts interacting with the sub systems of the plant, lighting modes cannot be changed in order to avoid interference and allow the user to listen to the information without any disturbance.

# Initial Proposals

1. The first proposal was using Arduino board. A text to speech converter module, EMIC-2 is interfaced with the Arduino for text to speech conversion. Sensors (PIR and Ultrasonic range finder) are used to detect human presence. An external mic is connected to Arduino to take audio input from user.

Requirements: Arduino Uno, Arduino IDE, Sensors and EMIC-2.

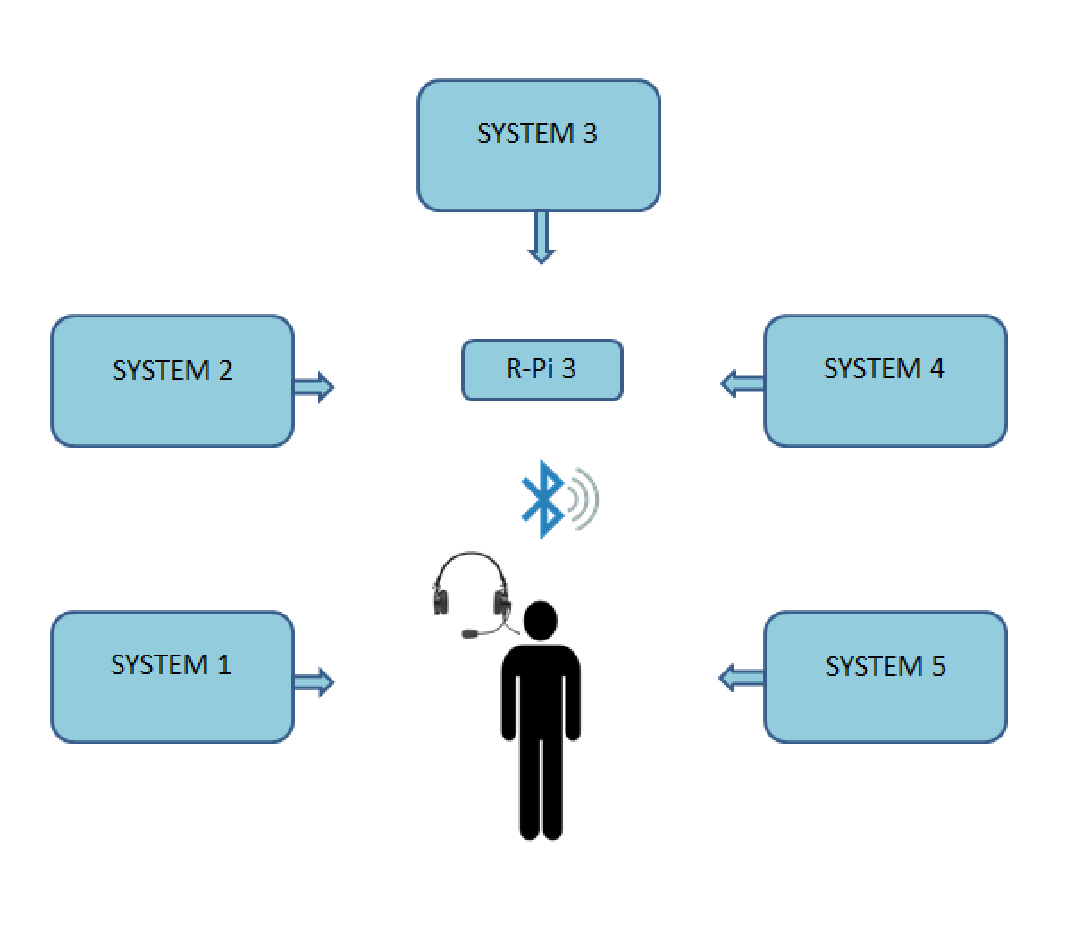
2. This proposal used Arduino board, EMIC-2, sensors as previous one. The Bitvoicer software is used to process speech on the Arduino board. The Arduino needs to be connected to a computer continuously. Speech processing happens entirely on the computer. An external mic with amplifier was required.

Requirements: Arduino Uno, Arduino IDE, Sensors, EMIC-2, Bitvoicer.

3. This proposal uses Raspberry Pi instead of Arduino. Windows 10 IOT core is used to interface pi to a computer. Pi can be programmed with C# in Visual studio. An SD card of 8GB or more is required. EMIC-2, sensors and an external mic are required and have same functionality as in previous proposals.

Requirements: Raspberry pi, Windows 10 IOT core, Visual Studio, Sensors, EMIC-2.

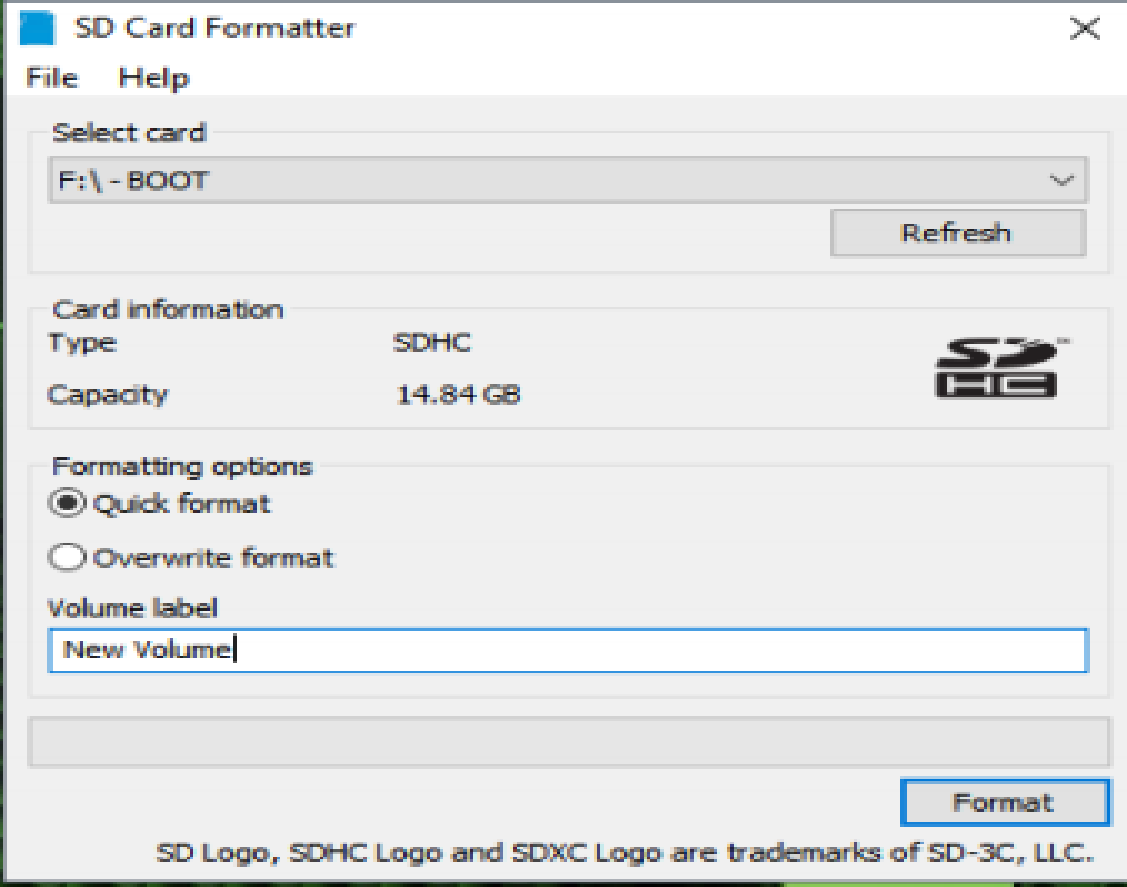
# The Prototype

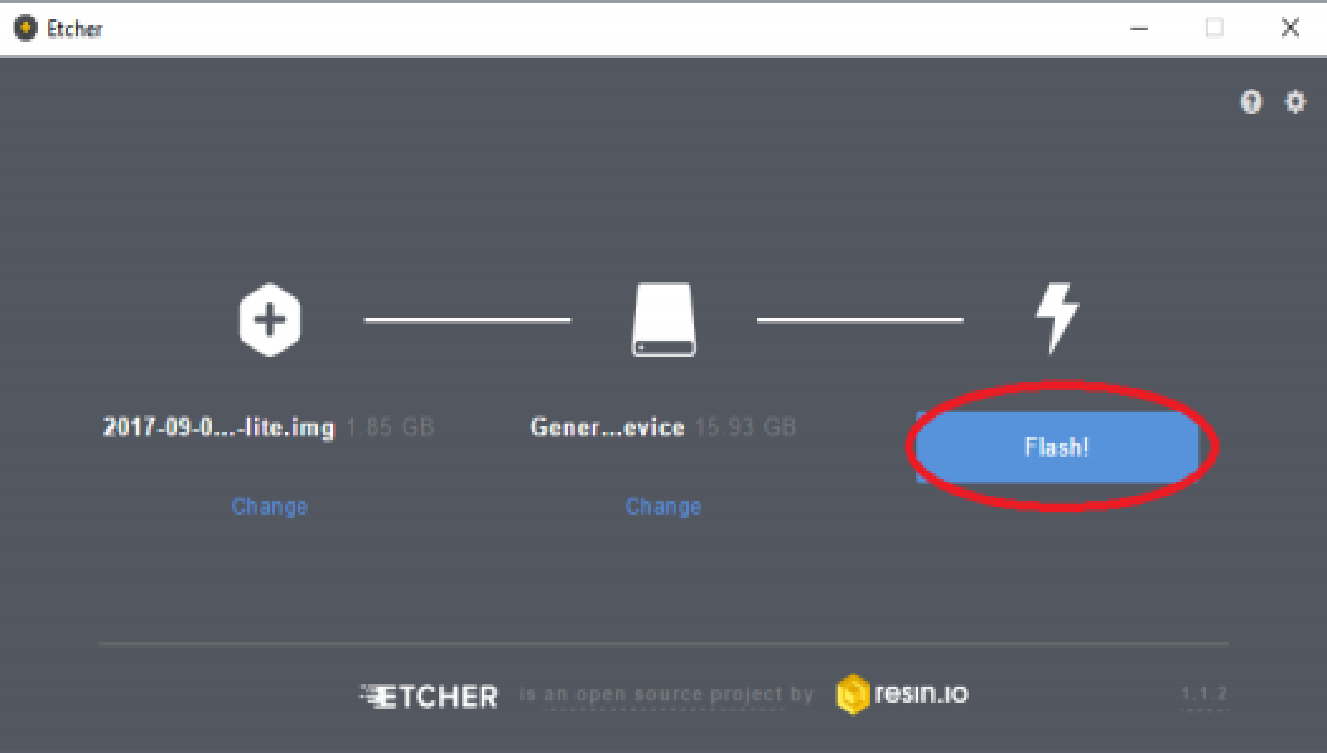


# Stages of Development

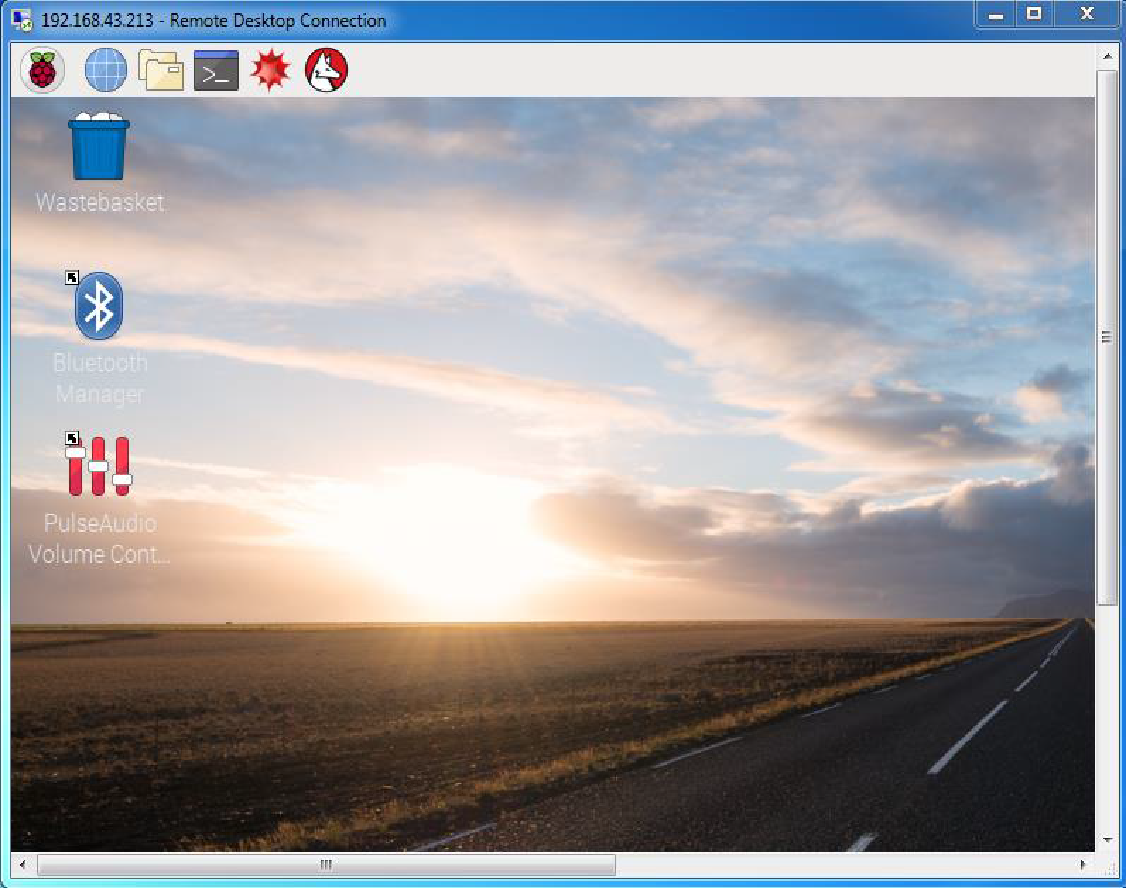
**SETTING UP RASPBERRY PI:**

* Download Etcher and install the .exe file on the computer.
* Download SD card formatter and install it.
* Download the .img Raspbian OS file from raspberrypi.org.
* Format the SD card using SD card formatter. The SD card needs to have a memory of atleast 8GB.





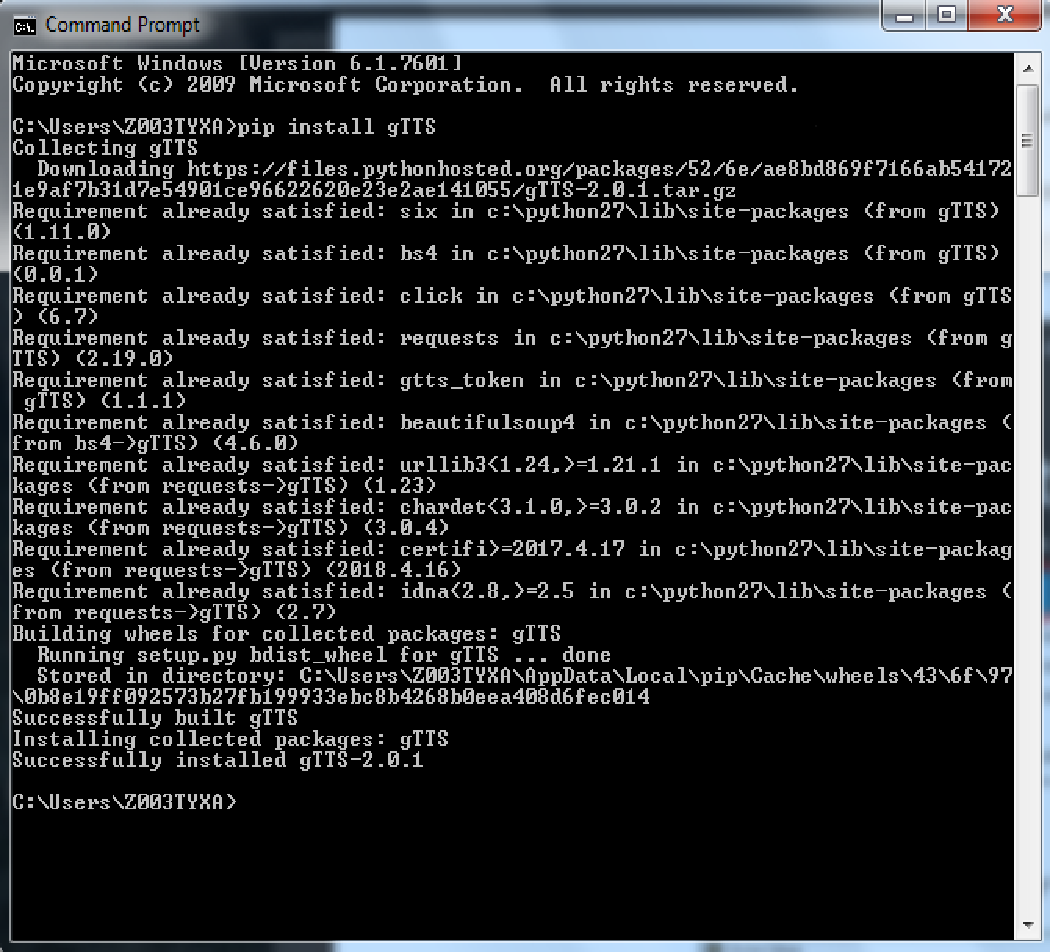
* The SD card is now ready to be inserted in the pi. The card is inserted in a slot on the underside of the pi.
* Connect pi and the PC to a Wi-Fi router through an Ethernet cable. This is done to get both pi and PC on the same network.
* Install Advanced IP scanner on the PC.
* Scan for the connected devices in the network to find IP address of the pi using Advanced IP scanner.
* The pi must be a live host on the scanner screen.
* Open Windows Remote Desktop connection and enter the pi’s IP address to remotely access the pi’s desktop.
* Enter username as “**pi**” and password as “**raspberry**”.
* The raspbian desktop now appears and can be used as a user interface to program the pi.

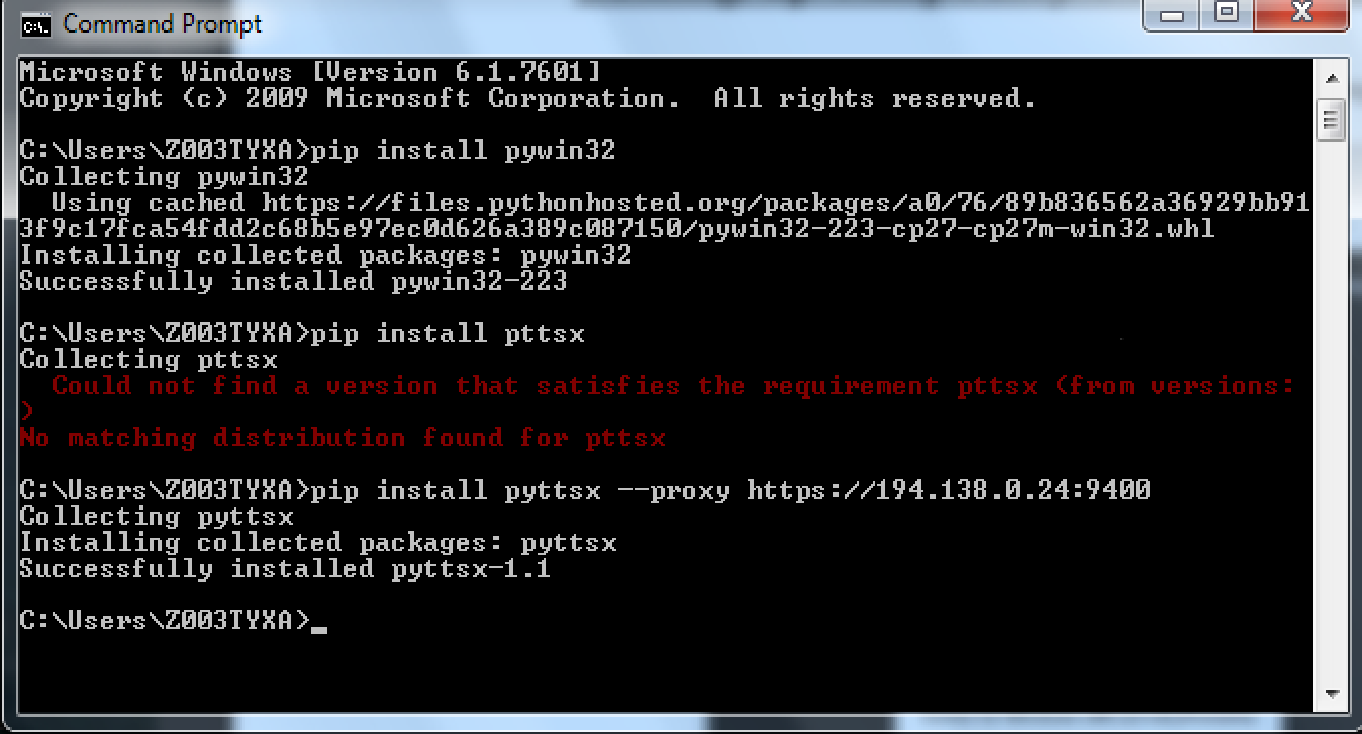


* Open command prompt on the raspbian desktop and type “sudo raspi-config”. In Advanced options, enable Expand File System.
* In Interfacing options, enable SSH, Serial, Remote GPIO.

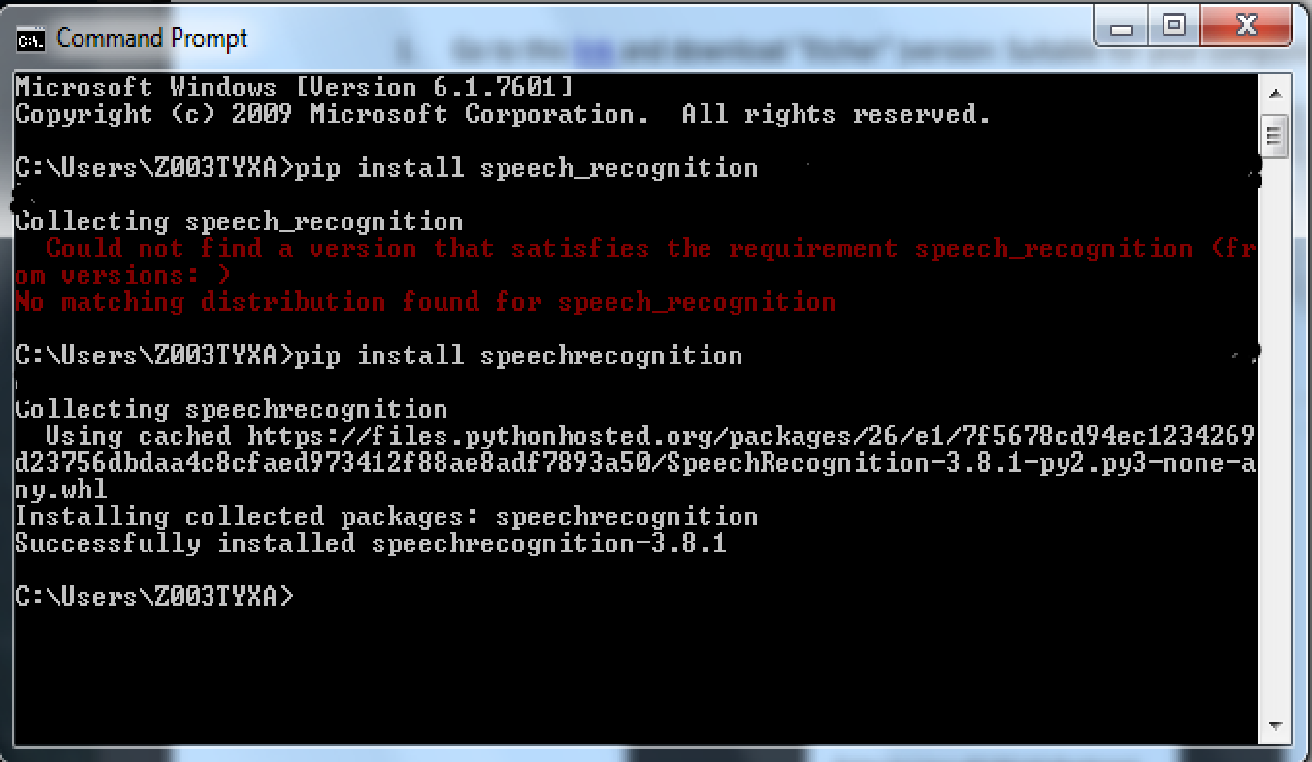
**INSTALLING PYTHON AND REQUIRED LIBRARIES:**

* This prototype requires python version 2.7 which was downloaded from python.org.
* Some libraries used for text to speech conversion are, pyttsx, gTTS, pygame.

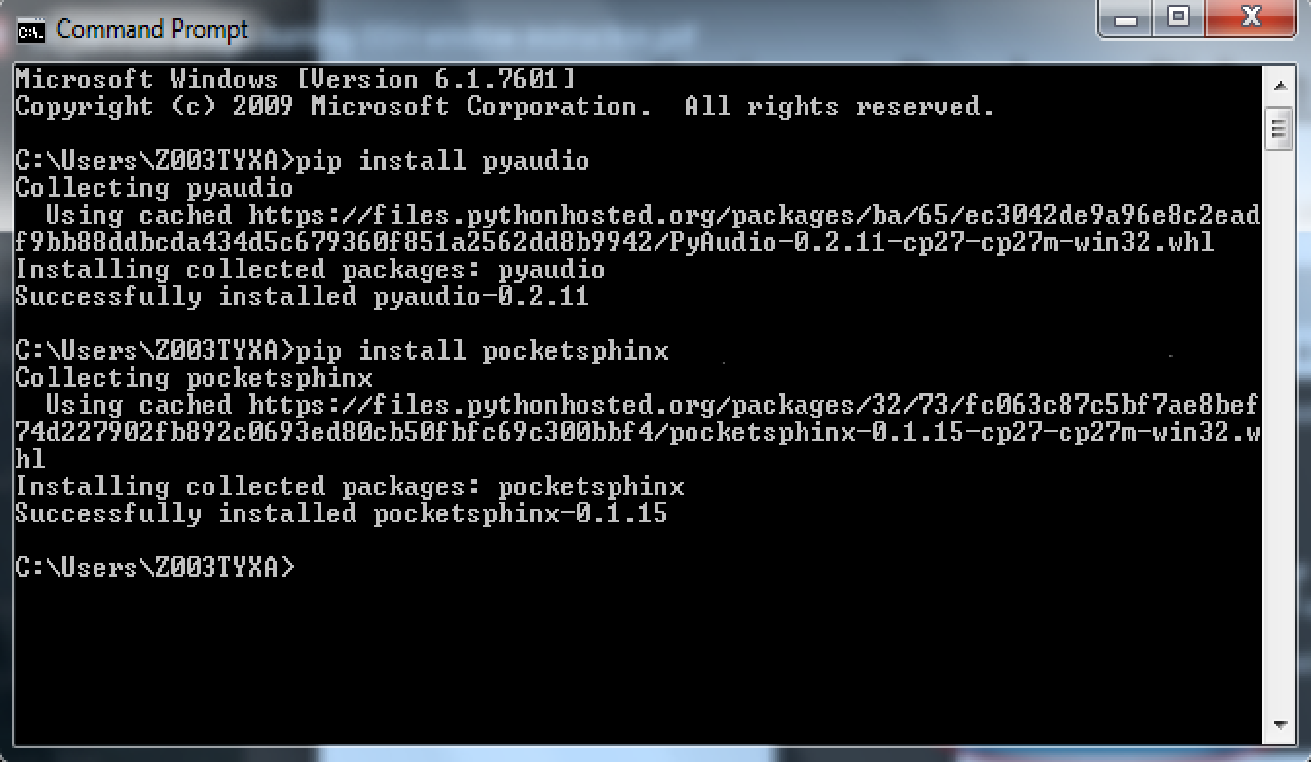




* The library used for speech to text conversion is speech\_recognition.



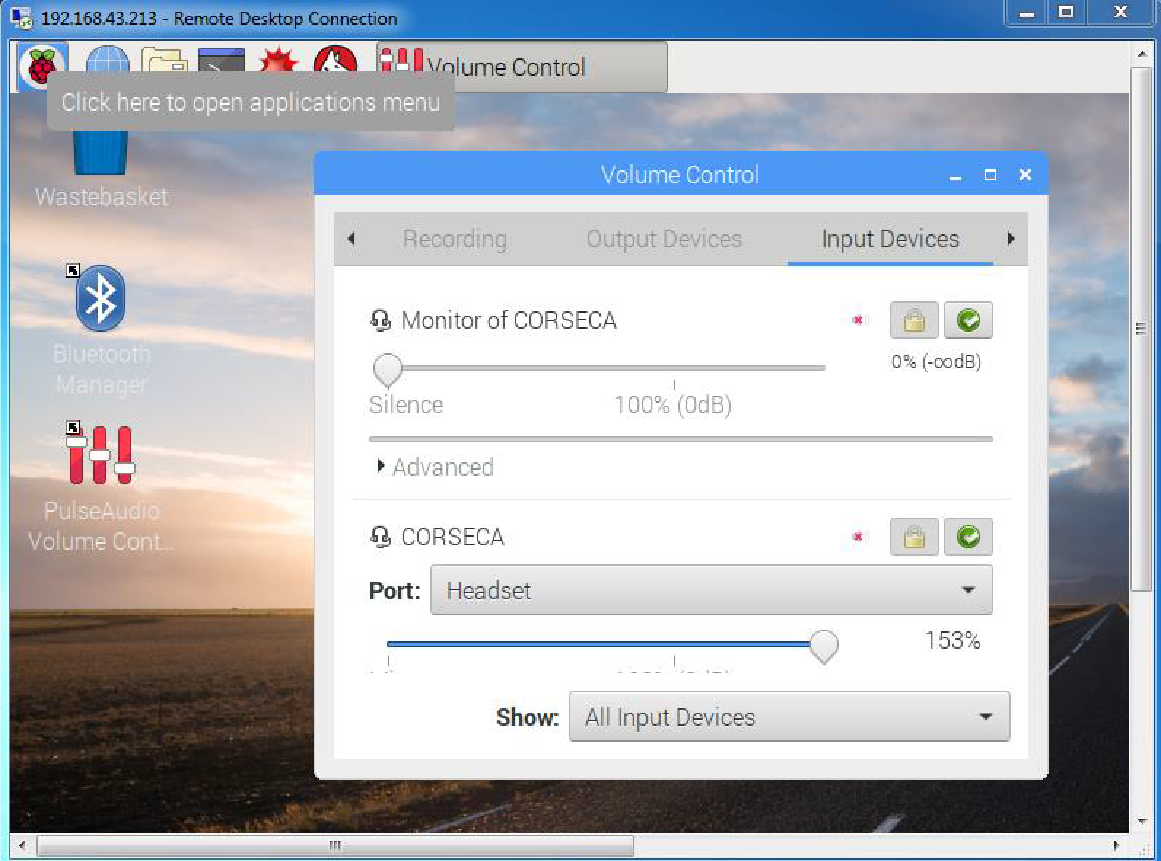
* The libraries that were used to aid the Bluetooth connection to pi were, pulseaudio, pybluez, pyaudio.

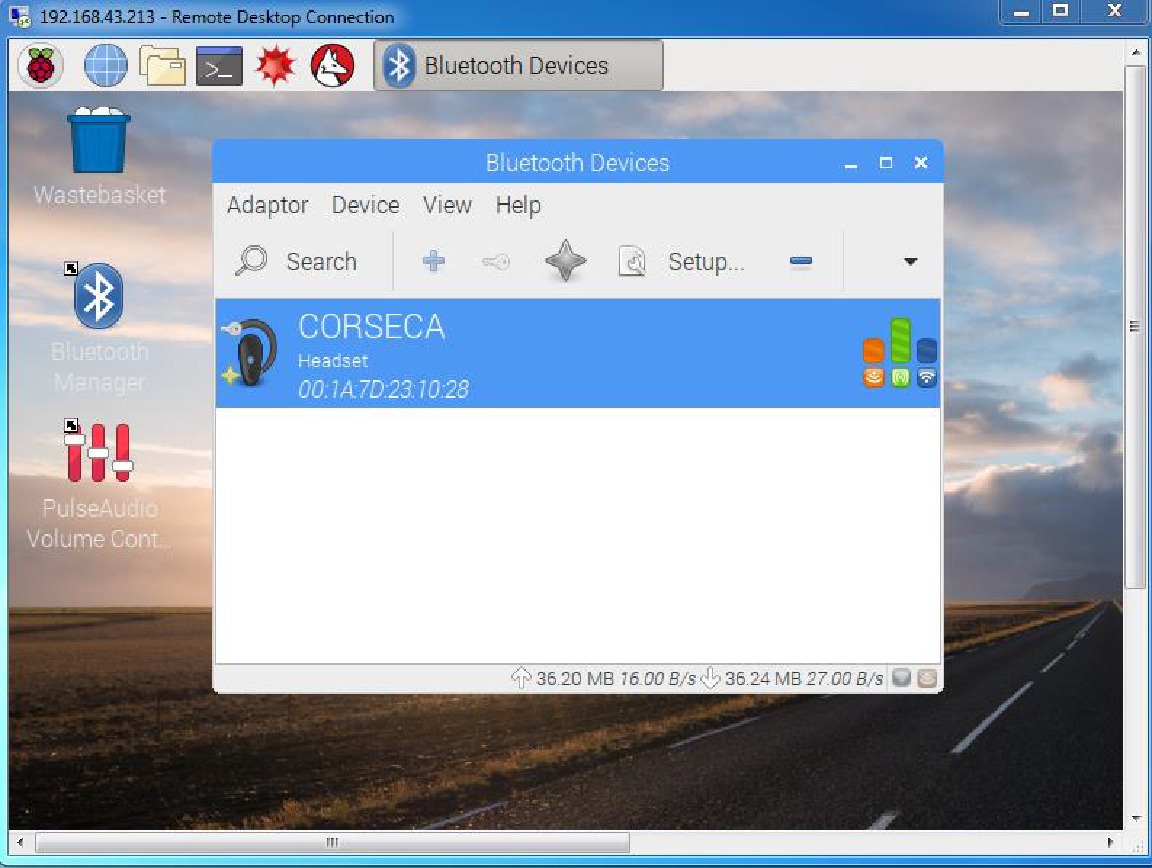


* These libraries were installed using pip installer.

**SETTING UP BLUETOOTH MANAGER AND PULSE AUDIO VOLUME CONTROL:**

* Open command prompt on the pi desktop and type”sudo apt-get install pi-bluetooth”.
* Open a terminal and type “sudo apt-get install blueman pulseaudio pavucontrol pulseaudio-module-bluetooth”.
* Restart the pi.
* Bluetooth Manager will now be enabled on the pi.
* Open a terminal and type “sudo apt-get install vlc”.
* Pulse Audio Volume control will now be enabled on the pi.
* Connect the Bluetooth device to pi through Bluetooth Manager.
* To set up HSP Bluetooth profile, which enables the headset to be used as a speaker and a mic simultaneously, go to Pulse Audio Volume Control, in the Configurations tab, set the profile of the inbuilt speaker “off”. The profile of the headset being used is set to “Headset Head Unit (HSP)”.





**Text to Speech with python:**

Python has several libraries through which speech processing can be done. The two important libraries include, pyttsx and gTTS.

gTTS is an online library. It is a library developed by Google. This library can be used to save a text as a voice mp3 file. It can be easily installed using the pip installer in python. This library has been used in the project to save the description of every system in an mp3 file.

Pyttsx can be used for direct conversion of text to speech. This library has three attributes namely, ‘rate’, ‘voice’, ‘volume’.

These are actually the attributes of the output voice that can be set by the user. Rate determines the speed of the speech, volume determines its loudness and voice determines the tone of the speech. There are few standard voices given in pyttsx that can be used. The age and gender of the voice can also be set.

Pyttsx supports text to speech in all the major languages of the world.

import pygame

from gtts import gTTS

from playsound import playsound

import os

tts = gTTS(text='text to mp3')

tts.save('intro.mp3')

**Playing an mp3 file with python:**

The pygame library is used to play music, mp3 and wav files in python. This library too can be installed easily using the pip installer.

It contains python classes and functions that allow the user to play audio and video output. This library was originally developed to develop video games in python.

An mp3 file can be played in three steps, initialize, load and play.

import pygame

pygame.mixer.init()

pygame.mixer.music.load("intro.mp3")

pygame.mixer.music.play()

**Speech to Text with python:**

This prototype uses the library called speech\_recognition for speech to text conversion in python. The library’s recognizer class can make use of several methods to convert text to speech. The most efficient one being the Google API. Google API requires a continuous internet connection to efficiently recognize speech.

The microphone needs to be initialized properly in order to correctly recognize speech. Pyaudio must be installed along with speech\_recognition to use microphones. The list of microphone names can be viewed by calling the list\_microphone\_names()static method of the Microphone class.

The device id of the mic is then obtained. It is the serial number of the mic name in the list of devices. This device id can be used to initialize the microphone and set it as the default input device.

import speech\_recognition as sr

while True:

r=sr.Recognizer()

with sr.Microphone() as source:

print("Say Something")

audio=r.listen(source)

try:

print(r.recognize\_google(audio),"\n")

except:

print("Couldnt recognise")

**GPIO Interrupts and polling on R-pi 3:**

The Raspberry Pi 3 has 40 GPIO pins. These pins can be configured as input or output pins. The push buttons provided for each system are connected to the GPIO pins of the pi.

The pins to which the push buttons are connected are configured as input pins. Initially all the input pins are pulled down to logic 0. When a button is pressed, 3.3V appears on the corresponding GPIO pin and it is recognized as logic 1.

The pins are programmed such that the pi must ask the customer what information he wants and speak the information of the correct system to the customer when he presses the button. The GPIO pins are continuously polled to check which button is pressed. The information of the system corresponding to that button is spoken to the customer.

GPIO interrupts can also be provided instead of continuously polling the pins. Interrupts use the processing power of the pi more efficiently. The interrupt trigger can be set to rising edge, this way an interrupt is triggered whenever the pi encounters a rising edge on any GPIO pin.

The RPi.GPIO package needs to be installed on the pi in order to program the pins with python.

# Adding male and female voice:

To make the Raspberry Pi speak in male or female voice, we need a software interface to convert text to speech on the headset. For this we need a **Text To Speech engine**. The voice may be a little robotic.

The code is as follow-

engine=pyttsx.init()

sound=engine.getProperty('voices')

rate=engine.getProperty('rate')

volume=engine.getProperty('volume')

engine.setProperty('rate',100)

engine.setProperty('volume',.5)

engine.say("say 1 for male and 2 for female")

time.sleep(5)

engine.runAndWait()

print("say 1 for male and 2 for female")

r=sr.Recognizer()

print("1")

with sr.Microphone() as source:

print("2")

energy\_threshold=3000

print("3")

#time.sleep(5)

audio = r.listen(source)

print("44")

worda = r.recognize\_google(audio)

print(worda)

print("5")

if worda=='1':

print("you chosed male voice")

if worda=='2':

print("you chosed female voice")

# Functionality

The complete functionality of this prototype has two parts:

**SYSTEM INFORMATION:**

In an automation plant with several subsystems, each system is provided with a push button. Every push button is connected to one GPIO pin of the pi.

When the customer wishes to know anything about a subsystem, he has to press the push button. The pi then speaks to the customer to ask what information is required. The customer then has to tell the pi what information he needs.

The pi processes and understands the speech and speaks out the required information. The customer also has an option of asking the pi to repeat the same information again by saying “1”. After the customer is satisfied with the information he heard, he can say “2” to stop and move on to any other system in the plant.

He can then repeat the same process and acquire information about all systems of the plant. The customer also has the option of listening to information about one system multiple times.

**LIGHTING AND DALI INTERFACE:**

The pi can also be used to set the lighting modes in the plant. The customer must tell the pi what mode he wants to set, example: presentation mode. The pi then commands the DALI through the SCADA and the DALI will set the lighting modes of the room.

These two tasks are enabled in the same system and can be performed simultaneously. Although, preference is given to the task of acquiring system information.

Once the task to acquire system information commences, the DALI interfacing task is stopped for the better experience of the customer. The DALI interface task restarts 30 minutes after the last button (system information task) is pressed. After this, both tasks will run simultaneously as before.

Coding for introducing delay between two tasks:

flag1=1

while(flag1==1):

t1=time.time()

if ((not prev\_input1) and input1):

print("Button 1 pressed")

t0=time.time()

system1()

if(t1-t0>=1800):

flag1=0

# Running Script on Booting The Pi

* Create a python script. Make sure that it is debugged. Find out the path in which the script is present on the pi.
* Say, the name of file is Audio\_Device and the path is home/pi. To double check this, type “cat/home/pi/Audio\_Device” in terminal. This command must show the contents of the file.
* Add a new cron job by modifying the cron tab. Use this command in the terminal, “sudo crontab –e”.
* Add the following line in terminal “@reboot python home/pi/Audio\_Device.py &” to run the selected script.
* To save these changes, type CTRL+X then Y then ‘Return’.
* Type “sudo reboot” to reboot and check if the script runs on boot.

# Challenges Faced

* Getting Raspberry pi and PC on the same network. Initial setup has to be done with a Wi-Fi router.
* Enabling HSP Bluetooth profile on raspberry pi to make the device interactive.
* Programming GPIO pins such that it detects the input from push buttons any number of times.
* Giving preference to the running of one task and giving a delay between the finishing of the preferred task (acquiring system information) and the other task (DALI interface).

# Potential Improvements

* The device can be made more interactive by extending the number of words that can be understood by it.
* The device can be made to understand more accents by training it over time using machine learning.
* The customer can be given an option to select the language of his choice to interact with the device
* The customer can be given an option to choose the trainer (voice) to take him through the task of acquiring system information.
* A display can be provided for every system to make the device more interactive.

# References

* https://pythonprogramminglanguage.com/text-to-speech/
* http://pyttsx.readthedocs.io/en/latest/engine.html#the-engine-factory
* https://stackoverflow.com/questions/30612298/text-to-speech-tts-module-that-works-under-python-3
* https://realpython.com/python-speech-recognition/
* https://raspi.tv/2013/how-to-use-interrupts-with-python-on-the-raspberry-pi-and-rpi-gpio-part-3
* http://youness.net/raspberry-pi/bluetooth-headset-raspberry-pi-3-ad2p-hsp
* https://raspi.tv/2013/how-to-use-interrupts-with-python-on-the-raspberry-pi-and-rpi-gpio
* https://www.youtube.com/watch?v=LRUZ668ZSVU
* https://www.codesofinterest.com/2017/03/python-speech-recognition-pocketsphinx.html
* https://www.raspberrypi.org/forums/viewtopic.php?t=109160
* http://www.instructables.com/id/Raspberry-Pi-Launch-Python-script-on-startup/
* https://www.dexterindustries.com/howto/run-a-program-on-your-raspberry-pi-at-startup/
* https://www.lfd.uci.edu/~gohlke/pythonlibs/
* https://plugable.com/2016/03/14/listening-to-bluetooth-audio-on-your-raspberry-pi-3-pi-2-or-pi-zero/
* https://plugable.com/2015/05/11/quick-fix-for-problems-using-bluetooth-from-the-raspberry-pi-desktop/