**Personal AI assistant using Python**

**Project submitted for the**

**Award**

**of Diploma**

**in**

**Electronics and Telecommunication Engineering**

**By**

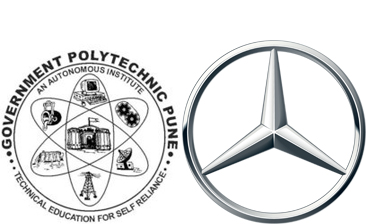
**Shubhankar R. Karajkhede 1903030**

**Tanmay H. Ladhe 1903038**

**Pranav S. Madane 1903040**

**Prathmesh H. Shimpi 1903053**

**Under the Guidance of**

**Mrs. P.V. Lengare**

**Government Polytechnic, Pune**

**June 2022**

## Government Polytechnic, Pune

## (An Autonomous Institute of Government of Maharashtra)



**Certificate**

**Shubhankar R. Karajkhede 1903030**

**Tanmay H. Ladhe 1903038**

**Pranav S. Madane 1903040**

**Prathmesh H. Shimpi 1903053**

**Has completed the necessary project work and prepared the bonafied on**

**“Personal AI assistant using Python”**

**In a satisfactory manner as a partial fulfilment of requirements of**

**Third Year Diploma In**

**Electronics and Telecommunication Engineering**

**For the Academic Year**

**2021-22**

**Dr. Vitthal B. Bandal**

**(Principal)**

**Dr. S.P. Narote**

**(HOD)**

**Smt. Pallavi V. Lengare**

**(Guide)**

Acknowledgement

In completing this project report on project titled **Personal AI assistant using Python**, I had to take the help and guideline of a few respected people, who deserve my greatest gratitude.

The completion of this project report gives me much Pleasure. I would like to show my gratitude to **Smt. Pallavi V. Lenagre and Dr. S.P. Narote** for giving me a good guideline for project throughout numerous consultations. I would also like to expand my deepest gratitude to all those who have directly and indirectly guided us in writing this project report.

Many people, especially my classmates and friends themselves, have made valuable comments and suggestions on this proposal which gave me inspiration to improve my project. Here I thank all the people for their help directly and indirectly to complete this project.

**List of Figures**

4.2 (a) Code Main Flowchart………………………………………………………………………12

4.2 (b) Code Main Flowchart ……………………………………………...………………………13

4.4.1 Flowchart of if statement in Python programming……..……………………….…………...18

4.2.2 Data flow diagram of voice engine and take\_command()…………...………………………20

4.5.2 Wikipedia module working………………………………………………………………….23

4.5.4 Youtube search flowchart……………………………………………………………………24

4.6.1 Pinout Diagram of Arduino UNO……………………………………………………………26

4.6.2 (a) working of relay……………………………………………………………………….....27

4.6.2 (b) Relay module………………………………………………………………………….....28

4.6.2 (c) Actual circuit diagram of 4 channel 5v relay module……………………………………29

4.6.3 Data flow of pyfirmata function…………………………………………………………......31

4.8 (a) Schematic circuit diagram………………………………………………………………….40

4.8 (b) Actual Circuit………………………………………………………………………………40

**List of Tables**

3.3 Hardware components……………………………………………………………………….10

4.6.1 Specifications of Arduino UNO……………………………………………………………25

4.6.2 Pin Specifications of 4 channel relay module……………………………………………...27

**Contents**

1. Certificate
2. Abstract
3. Acknowledgement
4. List of Figures
5. List of Tables

Chapter 1………………………………………………………………………………..1

Introduction……………………………………………………………………………..1

1.1 Background……………………………………………………….…………2

1.2 Objectives………………………………………………………...…………4

1.3 Purpose, Scope and Applicability…………………………………………..5

Chapter 2………………………………………………………………………………..6

Survey of Technology…………………………………………………………………..6

Chapter 3………………………………………………………………………………..8

Requirement and Analysis……………………………………………………………...8

3.1 Problem Definition………………………………………………………....8

3.2 Requirement Specification………………………………………………....9

3.3 Software and Hardware Requirement…………………………….……….10

Chapter 4………………………………………………………………………………11

System Design ……………………………………………………………………...…11

4.1-Algortihm for CLARE ……………………………………………………11

4.2-Flowchart ……………………………………………………………...….12

4.3-Librabries and their functions …………………………………………….14

4.4-Fundamentals for command handling……………………………..……..18

4.4.1 if...else statement in Python…………………………………….19

4.4.2 Voice engine, command and initial function…………….…….20

4.5 Functions……………………………………………………………...….22

4.5.1 Reading and Sending Emails…………………………………...22

4.5.2 Search on Wikipedia ……………………………………….…..22

4.5.3 Opening web URL…………………………………...…………23

4.5.4 Search a video on youtube………………………………..…….23

4.5.5 Other functions…………………………………………….……24

4.6 Automation using Voice …………………………………………………25

4.6.1 Arduino UNO ………………………………………………….25

4.6.2 Relay Module…………………………………………………..27

4.6.3 Application of pyfirmata for automation………………………30

4.7 Source code ……………………………………………………………...32

4.8 Circuit Diagram…………………………………………………………..40

1. Errors Faced..……………………………………………………………………….…41
2. Future Development.………………………………………………...……………….42
3. Conclusion…………………………………………………………….…………..….43
4. References..................................................................................................................44

**Chapter 1**

**Introduction**

In today’s era almost all tasks are digitalized. We have Smartphone in hands and it is nothing less than having world at your fingertips. These days we aren’t even using fingers. We just speak of the task and it is done. There exist systems where we can say Text Dad, “I’ll be late today.” And the text is sent. That is the task of a Virtual AI Assistant. It also supports specialized task such as playing songs, or searching information from Wikipedia and sending / reading emails, discovery and other operations.

Personal AI Assistants are software programs that help you ease your day to day tasks, such as showing date and time, creating mails, making hand free texts etc. They can take commands via text (online chat bots) or by voice. Voice based intelligent assistants need an audio input which will act as the command. For our project, we named our AI assistant as CLARE. We have so many virtual assistants, such as Apple’s Siri, Amazon’s Alexa and Microsoft’s Cortana.

This system is designed to be used efficiently on desktops. Personal assistant software improves user productivity by managing routine tasks of the user and by providing information from online sources to the user. CLARE is effortless to use. Once you start the program your voice command will be recognize by her. And within seconds, it gets executed.

Voice searches have dominated over text search. Web searches conducted via mobile devices have only just overtaken those carried out using a computer and the analysts are already predicting that 50% of searches will be via voice by 2020.Virtual assistants are turning out to be smarter than ever. Allow your intelligent assistant to make email work for you. Detect intent, pick out important information, automate processes, and deliver personalized responses. This project was started on the premise that there is sufficient amount of openly available data and information on the web that can be utilized to build a virtual assistant that has access to making intelligent decisions for routine user activities.

**1.1 Background**

There already exist a number of desktop virtual assistants. A few examples of current virtual assistants available in market are discussed in this section along with the tasks they can provide and their drawbacks.

**SIRI from Apple**

SIRI is personal assistant software that interfaces with the user thru voice interface, recognizes commands and acts on them. It learns to adapt to user’s speech and thus improves voice recognition over time. It also tries to converse with the user when it does not identify the user request.

It integrates with calendar, contacts and music library applications on the device and also integrates with GPS and camera on the device. It uses location, temporal, social and task based contexts, to personalize the agent behavior specifically to the user at a given point of time.

**Supported Tasks**

* Call someone from my contacts list
* Launch an application on my iPhone
* Send a text message to someone
* Set up a meeting on my calendar for 9am tomorrow
* Set an alarm for 5am tomorrow morning
* Play a specific song in my iTunes library
* Enter a new note

**Drawback**

SIRI does not maintain a knowledge database of its own and its understanding comes from the information captured in domain models and data models.

**ReQall**

ReQall is personal assistant software that runs on smartphones running Apple iOS or Google Android operating system. It helps user to recall notes as well as tasks within a location and time context. It records user inputs and converts them into commands, and monitors current stack of user tasks to proactively suggest actions while considering any changes in the environment. It also presents information based on the context of the user, as well as filter information to the user based on its learned understanding of the priority of that information.

**Supported Tasks**

* Reminders
* Email
* Calendar, Google Calendar
* Outlook
* Evernote
* Facebook, LinkedIn
* News Feeds

**Drawback**

Will take some time to put all of the to-do items in – you could spend more time putting the entries in than actually doing the revision.

**1.2 Objectives**

Main objective of building personal assistant software (a virtual assistant) is using semantic data sources available on the web, user generated content and providing knowledge from knowledge databases. The main purpose of an intelligent virtual assistant is to automate the daily tasks over voice command and answer questions that users may have. This may be done in a educational environment, for example, in staff room of a college a teacher may ask for turning on the light near their desk or fetch some important mails and information over the internet using only their voice

Virtual assistants can tremendously save you time. We spend hours in online research and then making the report in our terms of understanding. CLARE can do that for you. Provide a topic for research and continue with your tasks while she does the research. Another difficult task is to check the important mails daily. If you are from a corporate background you might know how important an email can be, just give a command to CLARE and she does the job for you.

One of the main advantages of voice searches is their rapidity. In fact, voice is reputed to be four times faster than a written search: whereas we can write about 40 words per minute, we are capable of speaking around 150 during the same period of time. In this respect, the ability of personal assistants to accurately recognize spoken words is a prerequisite for them to be adopted by consumers.

**1.3 Purpose, Scope and Applicability**

**Purpose**

Purpose of virtual assistant is to being capable of voice interaction, music playback, making emails, setting date-time, streaming videos, playing audio, and providing weather, traffic, sports, and other real-time information, such as news. AI assistants enable users to speak natural language voice commands in order to operate the device, apps and appliances.

There is an increased overall awareness and a higher level of comfort demonstrated specifically by millennial consumers. In this ever-evolving digital world where speed, efficiency, and convenience are constantly being optimized, it’s clear that we are moving towards less screen interaction.

**Scope**

Voice assistants will continue to offer more *individualized* experiences as they get better at differentiating between voices. However, it’s not just developers that need to address the complexity of developing for voice as brands also need to understand the capabilities of each device and integration and if it makes sense for their specific brand. They will also need to focus on maintaining a user experience that is consistent within the coming years as complexity becomes more of a concern. This is because the visual interface with voice assistants is missing. Users simply cannot see or touch a voice interface.

**Applicability**

The mass adoption of artificial intelligence in users everyday lives is also fueling the shift towards voice. The number of IoT devices such as smart thermostats and speakers are giving voice assistants more utility in a connected user’s life. Smart speakers are the number one way we are seeing voice being used. Many industry experts even predict that nearly every application will integrate voice technology in some way in the next 5 years.

The use of virtual assistants can also enhance the system of IoT (Internet of Things). Twenty years from now, Microsoft and its competitors will be offering personal digital assistants that will offer the services of a full-time employee usually reserved for the rich and famous.

**Chapter 2**

**Survey of Technology**

**Python**

Python is an OOPs (Object Oriented Programming) based, high level, interpreted programming language. It is a robust, highly useful language focused on rapid application development (RAD). Python helps in easy writing and execution of codes. Python can implement the same logic with as much as 1/5th code as compared to other OOPs languages.

Python provides a huge list of benefits to all. The usage of Python is such that it cannot be limited to only one activity. Its growing popularity has allowed it to enter into some of the most popular and complex processes like Artificial Intelligence (AI), Machine Learning (ML), natural language processing, data science etc. Python has a lot of libraries for every need of this project. For CLARE, libraries used are speechrecognition to recognize voice, Pyttsx for text to speech, webbrowser for web automation etc.

Python is reasonably efficient. Efficiency is usually not a problem for small examples. If your Python code is not efficient enough, a general procedure to improve it is to find out what is taking most the time, and implement just that part more efficiently in some lower-level language. This will result in much less programming and more efficient code (because you will have more time to optimize) than writing everything in a low-level language.

**Pyttsx**

Pyttsx stands for Python Text to Speech. It is a cross-platform Python wrapper for text- to-speech synthesis. It is a Python package supporting common text-to-speech engines on Mac OS X, Windows, and Linux. It works for both Python2.x and 3.x versions. Its main advantage is that it works offline.

**Speech Recognition**

This is a library for performing speech recognition, with support for several engines and APIs, online and offline*.* It supports APIs like Google Cloud Speech API, IBM Speech to Text, Microsoft Bing Voice Recognition etc.

**Sapi5**

SAPI5 also known as Microsoft Speech API is the technology for voice recognition and synthesis provided by Microsoft. It can be used to convert Text into Speech. Microsoft Speech API (SAPI5) is the technology for voice recognition and synthesis provided by Microsoft. Starting with Windows XP, it ships as part of the Windows OS.

**Pyfirmata**

PyFirmata is basically a prebuilt library package of python program which can be installed in Arduino to allow serial communication between a python script on any computer and an Arduino. This python package can give access to read and write any pin on the Arduino.

**Arduino UNO**

Arduino UNO is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. You can tinker with your UNO without worrying too much about doing something wrong, worst case scenario you can replace the chip for a few rupees and start over again.

**Relay Module**

The relay module is an electrically operated switch that can be turned on or off deciding to let current flow through or not. They are designed to be controlled with low voltages like 3.3V like the ESP32, ESP8266, etc, or 5V like your Arduino.

**Chapter 3**

**Requirement and Analysis**

System Analysis is about complete understanding of existing systems and finding where the existing system fails. The solution is determined to resolve issues in the proposed system. It defines the system. The system is divided into smaller parts. Their functions and inter relation of these modules are studied in system analysis. The complete analysis is followed below.

**3.1 Problem definition**

Usually, user needs to manually manage multiple sets of applications to complete one task. For example, a user trying to read emails from a particular mail ID There is need of a system that can manage tasks effortlessly.

We already have multiple virtual assistants. But we hardly use it. There are number of people who have issues in voice recognition. These systems can understand English phrases but they fail to recognize in our accent. Our way of pronunciation is way distinct from theirs. Also, they are easy to use on mobile devices than desktop systems. There is need of a virtual assistant that can understand English in Indian accent and work on desktop system.

When a virtual assistant is not able to answer questions accurately, it’s because it lacks the proper context or doesn’t understand the intent of the question. Its ability to answer questions relevantly only happens with rigorous optimization, involving both humans and machine learning. Continuously ensuring solid quality control strategies will also help manage the risk of the virtual assistant learning undesired bad behaviors. They require large amount of information to be fed in order for it to work efficiently.

Virtual assistant should be able to model complex task dependencies and use these models to recommend optimized plans for the user. It needs to be tested for finding optimum paths when a task has multiple sub-tasks and each sub-task can have its own sub-tasks. In such a case there can be multiple solutions to paths, and the it should be able to consider user preferences, other active tasks, priorities in order to recommend a particular plan.

**3.2 Requirement Specifications**

Personal assistant software is required to act as an interface into the digital world by understanding user requests or commands and then translating into actions or recommendations based on agent’s understanding of the world.

CLARE focuses on relieving the user of entering text input and using voice as primary means of user input. Agent then applies voice recognition algorithms to this input and records the input. It then use this input to call one of the personal information management applications such as task list or calendar to record a new entry or to search about it on search engines like Google, Bing or Yahoo etc. Focus is on capturing the user input through voice, recognizing the input and then executing the tasks if the agent understands the task. Software takes this input in natural language, and so makes it easier for the user to input what he or she desires to be done.

Voice recognition software enables hands free use of the applications, lets users to query or command the agent through voice interface. This helps users to have access to the agent while performing other tasks and thus enhances value of the system itself. CLARE also have ubiquitous connectivity through Wi-Fi or LAN connection, enabling distributed applications that can leverage other APIs exposed on the web without a need to store them locally.

Virtual assistants must provide a wide variety of services. These include:

* Providing information such as weather, facts from e.g. Wikipedia etc.
* Send or read emails over voice command.
* Play music from streaming services such as YouTube and local storage.
* Automate web sites that you visit more often.
* Handle electronic appliance like lights and fan over voice.

**3.3 HARDWARE AND SOFTWARE REQUIREMENTS**

The software is designed to be light-weighted so that it doesn’t be a burden on the machine running it. The main hardware requirement is a microcontroller and a relay channel which can handle electrical appliances. This system is being build keeping in mind the generally available hardware and software compatibility. Here are the minimum hardware and software requirement for virtual assistant.

**Hardware:**

* Pentium-pro processor or later.
* RAM 512MB or more.
* Arduino UNO board
* 4 channel 5v Relay board
* 4 channel power board
* Jumper wires and general purpose copper wires

Actual used Hardware components

Table 3.3: Hardware components

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr No.** | **Component** | **Quantity** | **Price** |
| 1 | Arduino UNO | 1 | 780 |
| 2 | 4 channel 5v Relay board | 1 | 420 |
| 3 | copper wires | 3 meter | 150 |
| 4 | jumper wires | 10 | 50 |
| 5 | Male Female connectors | 4 | 100 |
| 6 | 220v power supply (with case) | 1 |  |
| 7 | LED strip and bulb | 1 | 265 |

**Software:**

* Windows 7(32-bit) or above.
* Python 2.7 or later
* Chrome Driver
* Arduino IDE

**Chapter 4**

**System Design**

**4.1Algorithm used for CLARE**

CLARE can be categorised as intelligent virtual assistant. Intelligent Virtual Assistants aim at understanding what the user is telling them. This happens through Natural Language Understanding (or NLU for short). NLU deals with understanding meaning from what users have said. Essentially it is an ensemble of learning algorithms that try to identify, learn various patterns and make decisions or predictions on their own, relying purely on data instances and at times on human input.

We can categorize learning Conversational AI algorithms into two types: classifiers and language models. In order for the classifiers to work (i.e. learn and establish connections), there needs to be an existing compilation of phrases, which are already categorized into groups or classifications. This method requires human input. A common classifier for example is a spam detection system inside your mailbox. There are 2 classes, spam and not spam, which are regularly filled with new inputs either labeled as „spam“ or „not spam“ by the users.

Language models, on the other hand, can do its work of learning and predicting possible next words on a raw text without the assistance of humans. When starting out with a particular machine learning project, there are usually not that many categorized phrases (classifications) available for the Intelligent Virtual Assistant but plenty of raw text data. Unless you are using products with Conversational Intelligence functionality to distill raw text data into insight, you need language models to proceed.

**4.2 Flowchart**

Flowchart is a pictorial representation of a program. For our personal AI assistant following flowchart can be followed:

****

Fig. 4.2 (a) Code Main Flowchart

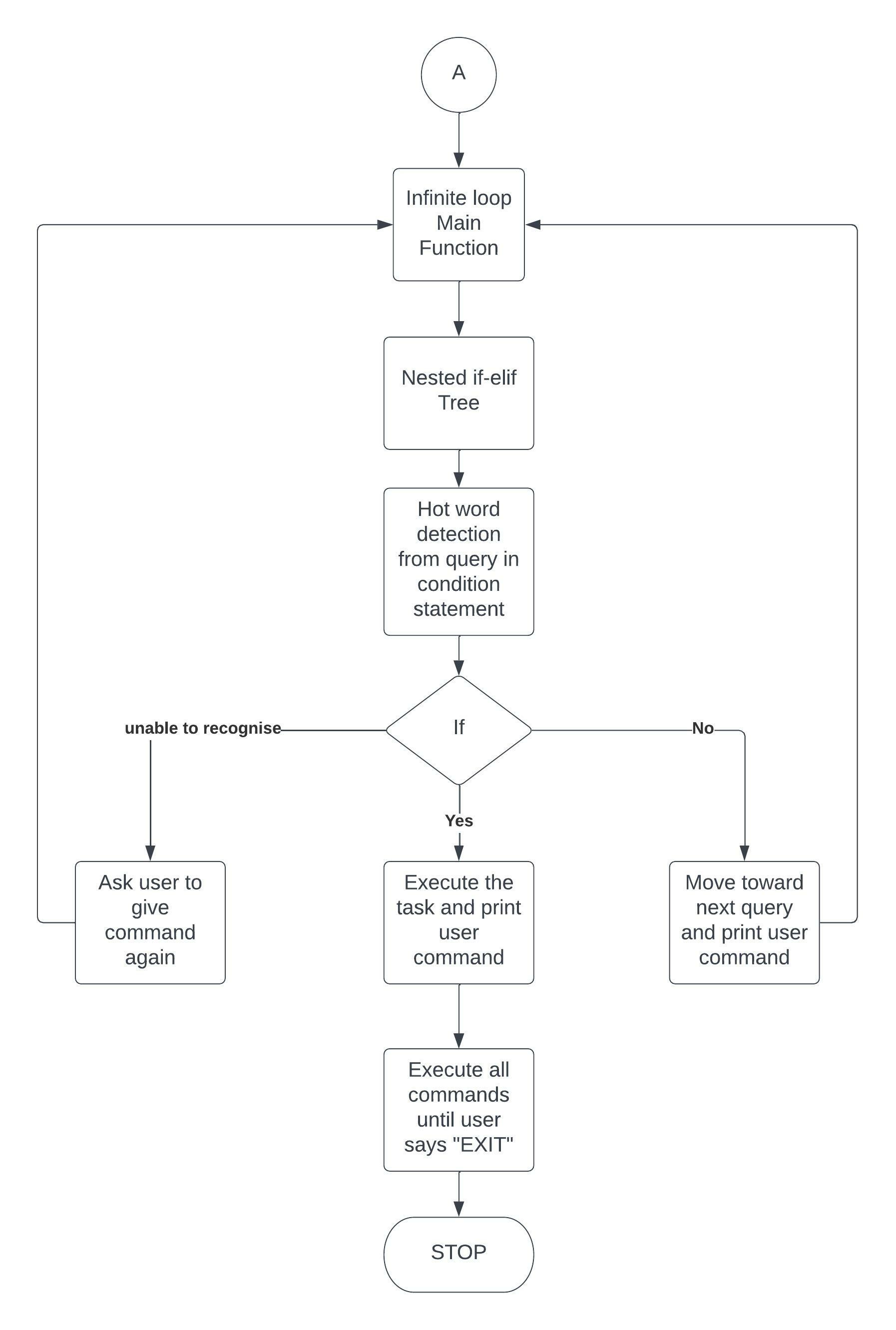
****

Fig. 4.2 Code Main Flowchart (b)

**4.3 Libraries and their Functions:**

Normally, a library is a collection of books or is a room or place where many books are stored to be used later. Similarly, in the programming world, a library is a collection of precompiled codes that can be used later on in a program for some specific well-defined operations. Other than pre-compiled codes, a library may contain documentation, configuration data, message templates, classes, and values, etc.

A Python library is a collection of related modules. It contains bundles of code that can be used repeatedly in different programs. It makes Python Programming simpler and convenient for the programmer. As we don’t need to write the same code again and again for different programs. Python libraries play a very vital role in fields of Machine Learning, Data Science, Data Visualization, etc. To develop CLARE we used different libraries which can be installed using pip. Libraries used are:-

* pyttsx3
* datetime
* speech\_recognition
* wikipedia
* pyfirmata
* pywhatkit
* imaplib
* email
* webbrowser
* os
* smtplib
* request

**1. pyttsx3 :**

pyttsx3 is a text-to-speech conversion library in Python. Unlike alternative libraries, it works offline and is compatible with both Python 2 and 3. An application invokes the pyttsx3. init() factory function to get a reference to a pyttsx3.

**2. datetime :**

Python Datetime module supplies classes to work with date and time. These classes provide a number of functions to deal with dates, times and time intervals. Date and datetime are an object in Python, so when you manipulate them, you are actually manipulating objects and not string or timestamps.

**3. speech\_recognition :**

Speech recognition is a machine's ability to listen to spoken words and identify them. You can then use speech recognition in Python to convert the spoken words into text, make a query or give a reply. You can even program some devices to respond to these spoken words.

**4. wikipedia:**

Wikipedia is a Python library that makes it easy to access and parse data from Wikipedia.Search Wikipedia, get article summaries, get data like links and images from a page, and more. Wikipedia wraps the MediaWiki API so you can focus on using Wikipedia data, not getting it.

**5. pyfirmata:**

pyFirmata is a Python interface for the Firmata\_ protocol. It is fully compatible with Firmata 2.1, and has some functionality of version 2.2. It runs on Python 2.7, 3.6 and 3.7. PyFirmata is basically a prebuilt library package of python program which can be installed in Arduino to allow serial communication between a python script on any computer and an Arduino. This python package can give access to read and write any pin on the Arduino.

**6. pywhatkit**

pywhatkit is a Python library for sending WhatsApp messages at a certain time, it has several other features too. Following are some features of pywhatkit module: Send WhatsApp messages. Play a YouTube video. Perform a Google Search.

**7. imaplib**

Python's client side library called imaplib is used for accessing emails over imap protocol. IMAP stands for Internet Mail Access Protocol. It was first proposed in 1986. Key Points: IMAP allows the client program to manipulate the e-mail message on the server without downloading them on the local computer.

**8. email:**

The email package is an email message management library. The email package's overall structure can be broken down into three basic components, plus a fourth component that regulates the behaviour of the others. The package's main component is an "object model" that represents email messages.

**9. webbrowser:**

In Python, webbrowser module is a convenient web browser controller. It provides a high-level interface that allows displaying Web-based documents to users. webbrowser can also be used as a CLI tool.

**10. os:**

The OS module in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules. This module provides a portable way of using operating system-dependent functionality. The \*os\* and \*os. path\* modules include many functions to interact with the file system.

**11. smtplib:**

The smtplib module defines an SMTP client session object that can be used to send mail to any internet machine with an SMTP or ESMTP listener daemon. For details of SMTP and ESMTP operation, consult RFC 821 (Simple Mail Transfer Protocol) and RFC 1869 (SMTP Service Extensions). class smtplib.

**12. request:**

The requests module allows you to send HTTP requests using Python. The HTTP request returns a Response Object with all the response data (content, encoding, status, etc).

Each library mentioned above is used for a specific purpose and it is applicable for a dedicated function. When the query contains Hot Word like “open google” or “search on wikipedia” the function is called and the libraries are used to carry out the task.

**4.4 Fundamentals for command handling:**

**4.4.1 if...else statement in Python.**

Decision making is required when we want to execute a code only if a certain condition is satisfied. The if…elif…else statement is used in Python for decision making.

Python if Statement Syntax:

if test expression:

statement(s)

Here, the program evaluates the test expression and will execute statement(s) only if the test expression is True. If the test expression is False, the statement(s) is not executed.In Python, the body of the if statement is indicated by the indentation. The body starts with an indentation and the first unindented line marks the end. Python interprets non-zero values as True. None and 0 are interpreted as False.

Python if Statement Flowchart:

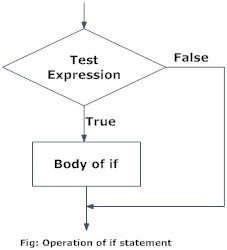


Fig. 4.4.1 Flowchart of if statement in Python programming

**4.4.2 Voice engine, command and initial function**

To setup the voice engine pyttsx3 library is used. Pyttsx3 library converts the text or string into the machine voice which is in built in our operating system.sapi5 is microsoft Speech API (SAPI5) is the technology for voice recognition and synthesis provided by Microsoft.

For initializing the voice setup and speed of the AI assistant voice the code below is used. CLARE uses a female voice for communication by using command setProperty(), we can set inbuilt voices for our virtual AI assistant.

engine = pyttsx3.init('sapi5')

voices=engine.getProperty("voices")

# print(voices[0].id)

engine.setProperty('voice', voices[0].id)

newVoiceRate = 145

engine.setProperty('rate',newVoiceRate)

take\_command() function is used to take voice command from microphone and convert them into strings by using speech recognition and pyttsx3. ‘def’ is keyword used to create a user defined function which can be called in main function. To overcome the effect of noise we have used timeout of 5 seconds and phrase limit of 3 phrases we can also use ambient noise function for setting threshold value as per the noise present in the environment.

The take\_command() function is very important as it is used to fetch the query from the user which can be later treated by CLARE as command as dedicated tasks are performed. The code for take\_command() function is shown as below:

def take\_command():

    r = sr.Recognizer()

    with sr.Microphone() as source:

        print("Listening...")

        r.pause\_threshold = 1

        audio = r.listen(source,timeout=5,phrase\_time\_limit=3)

        # r.adjust\_for\_ambient\_noise(source, duration = 1)

    try:

        print("Recognizing...")

        query = r.recognize\_google(audio, language='en-in')

        print(f"User said: {query}\n")

    except Exception as e:

        # print(e)

        print("Say that again please...")

        return "None"

    return query

Here we like to highlight “r.recognize\_google(audio, language='en-in')” this particular line of code. The recognize\_google() function is used to recognize the voice command which is decrypted using google speech recognition API. Language is set as English-Indian which will recognize the voice in indian accent.

Try and Except statement is used to handle these errors within our code in Python. The try block is used to check some code for errors i.e the code inside the try block will execute when there is no error in the program. Whereas the code inside the except block will execute whenever the program encounters some error in the preceding try block.

All the above steps can be verified by simple initial function as wishMe(). In this function CLARE greets the user according to time.

def wishMe():

    hour=int(datetime.datetime.now().hour)

    if hour>=0 and hour<12:

        speak('Good morning')

    elif hour>12 and hour<18:

        speak('Good afternoon')

    else:

        speak('Good evening')

    speak('I am Clare sir. please tell me how may I help you')

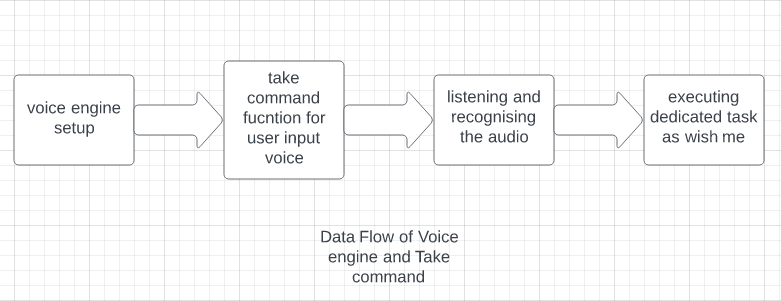


Fig. 4.2.2 Data flow diagram of voice engine and take\_command()

**4.5 Functions**

**4.5.1 Reading and Sending Emails**

Clare can send plain text emails to particular mail ID and read plain text email from the same

Python comes with the built-in [smtplib](https://docs.python.org/3/library/smtplib.html) module for sending emails using the Simple Mail Transfer Protocol (SMTP). smtplib uses the RFC 821 protocol for SMTP. The examples in this tutorial will use the Gmail SMTP server to send emails, but the same principles apply to other email services. Although the majority of email providers use the same connection ports as the ones in this tutorial, you can run a quick Google search to confirm yours.(code is shown in source code)

**4.5.2 Search on Wikipedia**

Wikipedia is the largest platform on the internet, which contains tons of information. It is an open-source platform which manages by the community of volunteer editors using a wiki-based editing system. It is a multi-lingual encyclopedia.

Python provides the Wikipedia module (or API) to scrap the data from the Wikipedia pages. This module allows us to get and parse the information from Wikipedia. In simple words, we can say that it is worked as a little scrapper and can scrap only a limited amount of data. Before we start working with it, we need to install this module on our local machine. Code used for Wikipedia search is:

if 'wikipedia' in query:

            speak('searching wikipedia...')

            query=query.replace("wikipedia","")

            results=wikipedia.summary(query,sentences=2)

            speak('According to wikipedia')

            speak(results)

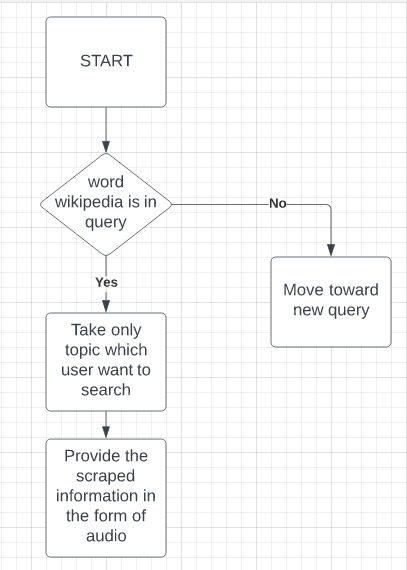


Fig. 4.5.2 Wikipedia module working

**4.5.3 Opening web URL**

The [webbrowser](https://docs.python.org/3/library/webbrowser.html" \l "module-webbrowser" \o "webbrowser: Easy-to-use controller for web browsers.) module provides a high-level interface to allow displaying web-based documents to users. Under most circumstances, simply calling the [open()](https://docs.python.org/3/library/webbrowser.html#webbrowser.open) function from this module will do the right thing.

CLARE can open any web URL. For convenience we implemented websites like youtube, stackoverflow, google and our college official web page.

webbrowser.open('https://gppune.ac.in/gpp/gpp\_s20/userindex.php')

webbrowser module is so easy to implement that we can see in above code. It takes only one function which is open() to open desired website.

**4.5.4 Search a video on youtube.**

By using pywhatkit CLARE can provide desired video search on youtube.

def youtube ():

    print(f'user want to watch {topic} ')

    pywhatkit.playonyt(f'{topic}')

if 'on youtube' in query:

            speak("what you want to watch")

            topic=take\_command()

            youtube()

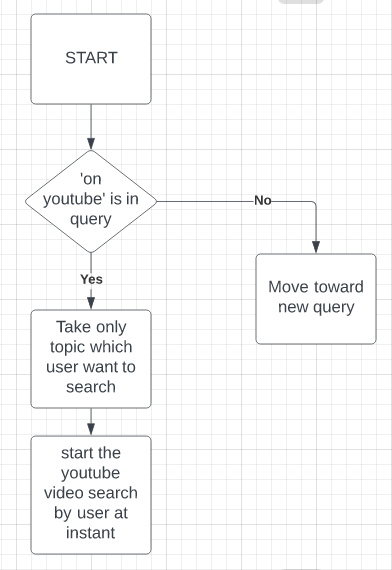


Fig. 4.5.4 Youtube search flowchart

**4.5.5 Other functions**

Besides these functions there are features like:

* Ask time
* Tell jokes or advice
* Open software applications
* Terminate software applications
* Play songs
* Home automation (turn on fan, turn on light, etc.)

CLARE can perform numerous tasks only the sky is the limit for it. AS we keep increasing the functions and hot words in query we can add the task as per our convenience.

**4.6 Automation using Voice**

**4.6.1 Arduino UNO**

**Main Features of Arduino:**

Arduino UNO is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. You can tinker with your UNO without worrying too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again.

**Specifications:**

Table 4.6.1: Specifications of Arduino UNO

|  |  |  |
| --- | --- | --- |
| Board | Name | Arduino UNO R3 |
| SKU | A000066 |
| Microcontroller | ATmega328P | |
| USB connector | USB-B | |
| Pins | Built-in LED Pin | 13 |
| Digital I/O Pins | 14 |
| Analog input pins | 6 |
| PWM pins | 6 |
| Communication | UART | Yes |
| I2C | Yes |
| SPI | Yes |
| Power | I/O Voltage | 5V |
| Input voltage (nominal) | 7-12V |
| DC Current per I/O Pin | 20 mA |
| Power Supply Connector | Barrel Plug |
| Clock speed | Main Processor | ATmega328P 16 MHz |
| USB-Serial Processor | ATmega16U2 16 MHz |
| Memory | ATmega328P | 2KB SRAM, 32KB FLASH, 1KB EEPROM |
| Dimensions | Weight | 25 g |
| Width | 53.4 mm |
| Length | 68.6 mm |

**Pinout Diagram**

Pin configuration of Arduino UNO R3

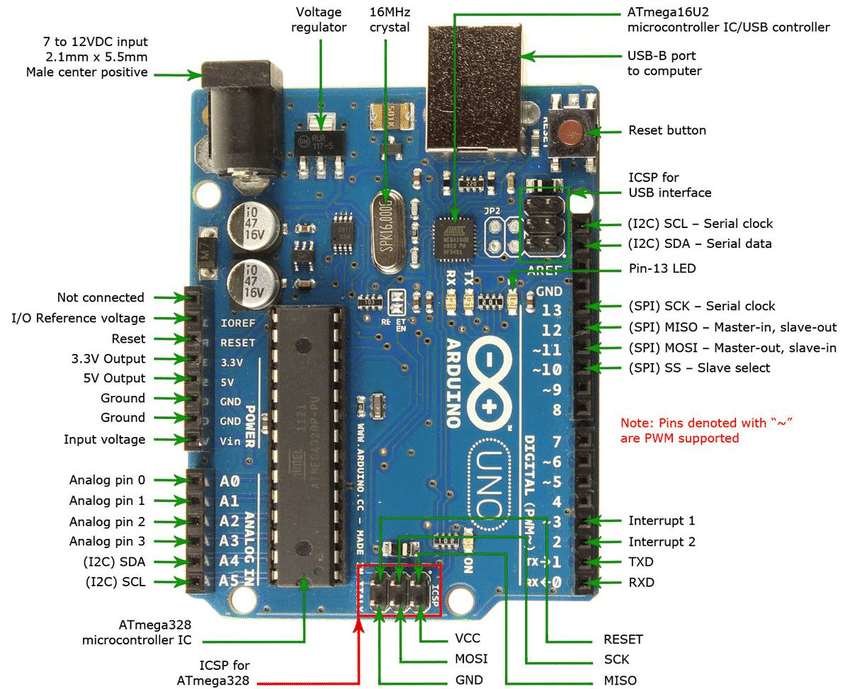


Fig. 4.6.1 Pinout Diagram of Arduino UNO

**4.6.2 Relay Module**

The four-channel relay module contains four 5V relays and the associated switching and isolating components, which makes interfacing with a microcontroller or sensor easy with minimum components and connections. The contacts on each relay are specified for 250VAC and 30VDC and 10A in each case, as marked on the body of the relays.

**Four-Channel Relay Module Pinout:**

Table 4.6.2: Pin Specifications of 4 channel relay module

|  |  |  |
| --- | --- | --- |
| **Pin Number** | **Pin Name** | **Description** |
| 1 | GND | Ground reference for the module |
| 2 | IN1 | Input to activate relay 1 |
| 3 | IN2 | Input to activate relay 2 |
| 4 | IN3 | Input to activate relay 3 |
| 5 | IN4 | Input to activate relay 4 |
| 6 | VCC | Power supply for the relay module |
| 7 | VCC | Power supply selection jumper |
| 8 | JD-VCC | Alternate power pin for the relay module |

**Working principal of Relay:**

The relay function can be better understood by explaining the following diagram given below:

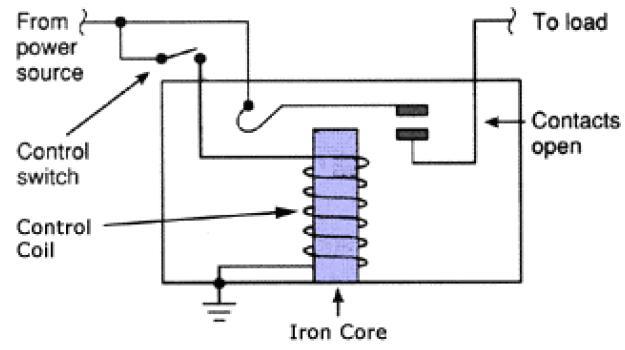


Fig.4.6.2 (a) working of relay

The diagram shows an inner section diagram of a relay. An iron core is surrounded by a control coil. As shown, the power source is given to the electromagnet through a control switch and through contacts to the load. When current starts flowing through the control coil, the electromagnet starts energizing and thus intensifies the magnetic field. Thus the upper contact arm starts to be attracted to the lower fixed arm and thus closes the contacts causing a short circuit for the power to the load.

On the other hand, if the relay was already de-energized when the contacts were closed, then the contact move oppositely and make an open circuit.As soon as the coil current is off, the movable armature will be returned by a force back to its initial position. This force will be almost equal to half the strength of the magnetic force. This force is mainly provided by two factors. They are the spring and also gravity.

**Relay Module Schematics:**

Real life relay module is shown in fig below. Following are the major components present on the four-channel relay module, we will get into the details of this later in the article.

5V relay, terminal blocks, male headers, transistors, [optocouplers](https://components101.com/optocouplers), diodes, and LEDs.

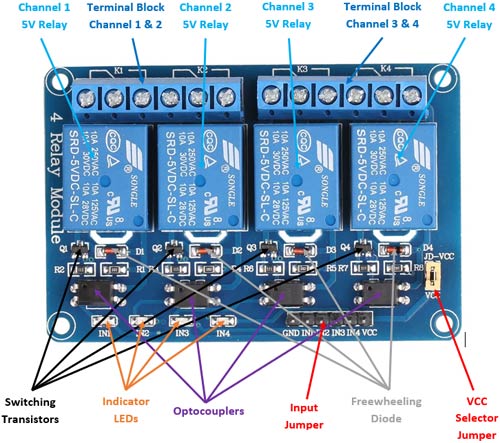


Fig. 4.6.2 (b) Relay module

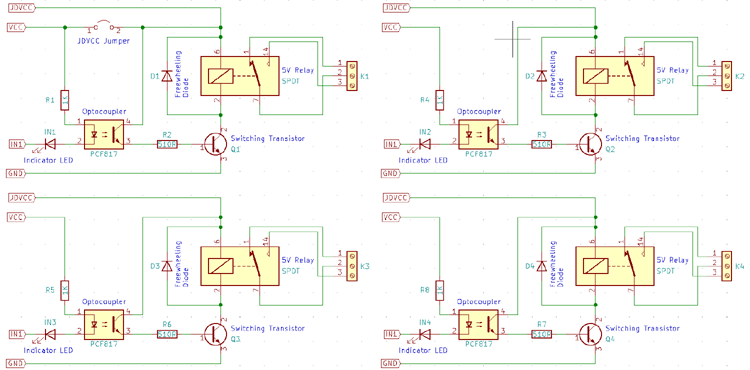


Fig. 4.6.2 (c) Actual circuit diagram of 4 channel 5v relay module

**4.6.3 Application of pyfirmata for automation**

[Firmata](http://firmata.org/wiki/Main_Page) is an intermediate protocol that connects an embedded system to a host computer, and the protocol channel uses a serial port by default. The Arduino platform is the standard reference implementation for Firmata. The Arduino IDE comes with the support for Firmata.

This could work perfectly with [intel-X86](https://www.seeedstudio.com/ODYSSEY-X86J4105864-p-4447.html) with its onboard Arduino Core meaning that you can control the Arduino Core simply using Firmata protocol with different programming languages too! In this tutorial, it's based on Windows OS.

CLARE uses digital functions of pyfirmata which allows user to use the python code to read or write the digital values (0 or 1) at any digital pin of arduino uno. First of all we need to initialize the COM port at our desktop at which the Arduino UNO is connected. The board specification have to defined like which board we are using in our case we are using Arduino UNO R3. Later, we have define variables containing digital read/write function, pin number and input/output configuration.

comport='COM3'

board=pyfirmata.Arduino(comport)

led1=board.get\_pin('d:8:o')

fan=board.get\_pin('d:9:o')

port1=board.get\_pin('d:10:o')

port2=board.get\_pin('d:11:o')

We have used digital pin 8,9,10,11 as output pins which are connected directly to the 4 channel relay module.

The function as ‘light on’ or ‘turn on fan’ are performed using pyfirmata. When the user give command containing hot word ‘light on’ or ‘turn on fan’ the program will check for the query string to call the respective function.

For example. If user said ‘light on’ the function led\_switch() will be called. In this function parameters are provided called val. At the instant program calls this function the value of val becomes:

1=light on

0= light off

This value is used to write the digital pin at arduino for turing light on or off.

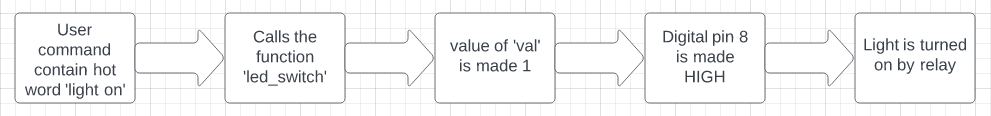


Fig 4.6.3 Data flow of pyfirmata function.

**4.7 Source code**

import pyttsx3

import datetime

import speech\_recognition as sr

import wikipedia

import pyfirmata

import pywhatkit

import imaplib

import email

from email.header import decode\_header

import webbrowser

import os

import smtplib

import requests

comport='COM3'

board=pyfirmata.Arduino(comport)

led1=board.get\_pin('d:8:o')

fan=board.get\_pin('d:9:o')

port1=board.get\_pin('d:10:o')

port2=board.get\_pin('d:11:o')

def led\_switch(val):

    if val==1:

        led1.write(1)

    elif val==0:

        led1.write(0)

def fan\_switch(val):

    if val==1:

        fan.write(1)

    elif val==0:

        fan.write(0)

def port1\_switch(val):

    if val==1:

        port1.write(1)

    elif val==0:

        port1.write(0)

def port2\_switch(val):

    if val==1:

        port2.write(1)

    elif val==0:

        port2.write(0)

engine = pyttsx3.init('sapi5')

voices=engine.getProperty("voices")

# print(voices[0].id)

engine.setProperty('voice', voices[0].id)

newVoiceRate = 145

engine.setProperty('rate',newVoiceRate)

def speak(audio):

    engine.say(audio)

    engine.runAndWait()

def wishMe():

    hour=int(datetime.datetime.now().hour)

    if hour>=0 and hour<12:

        speak('Good morning')

    elif hour>12 and hour<18:

        speak('Good afternoon')

    else:

        speak('Good evening')

    speak('I am Clare sir. please tell me how may I help you')

def take\_command():

    r = sr.Recognizer()

    with sr.Microphone() as source:

        print("Listening...")

        r.pause\_threshold = 1

        audio = r.listen(source,timeout=5,phrase\_time\_limit=3)

        # r.adjust\_for\_ambient\_noise(source, duration = 1)

    try:

        print("Recognizing...")

        query = r.recognize\_google(audio, language='en-in')

        print(f"User said: {query}\n")

    except Exception as e:

        # print(e)

        print("Say that again please...")

        return "None"

    return query

def youtube ():

    print(f'user want to watch {topic} ')

    pywhatkit.playonyt(f'{topic}')

def readMail():

    #credentials

    username ="raspberry02pi2003@gmail.com"

    #generated app password

    app\_password= "shubhankar@2003"

    # https://www.systoolsgroup.com/imap/

    gmail\_host= 'imap.gmail.com'

    #set connection

    mail = imaplib.IMAP4\_SSL(gmail\_host)

    #login

    mail.login(username, app\_password)

    #select inbox

    mail.select("INBOX")

    #select specific mails

    \_, selected\_mails = mail.search(None, '(FROM "raspberry02pi2003@gmail.com")')

    #total number of mails from specific user

    print("Total Messages from user :" , len(selected\_mails[0].split()))

    for num in selected\_mails[0].split():

        \_, data = mail.fetch(num , '(RFC822)')

        \_, bytes\_data = data[0]

        #convert the byte data to message

        email\_message = email.message\_from\_bytes(bytes\_data)

        print("\n===========================================")

        #access data

        print("Subject: ",email\_message["subject"])

        print("To:", email\_message["to"])

        print("From: ",email\_message["from"])

        print("Date: ",email\_message["date"])

        for part in email\_message.walk():

            if part.get\_content\_type()=="text/plain" or part.get\_content\_type()=="text/html":

                message = part.get\_payload(decode=True)

                print("Message: \n", message.decode())

                a=(message.decode())

                speak(a)

                print("==========================================\n")

                break

def sendEmail(to,content):

    server=smtplib.SMTP('smtp.gmail.com',587)

    server.ehlo()

    server.starttls()

    server.login('raspberry02pi2003@gmail.com', 'shubhankar@2003')

    server.sendmail('raspberry02pi2003@gmail.com', to, content)

def get\_random\_advice():

    res = requests.get("https://api.adviceslip.com/advice").json()

    # print(res)

    return res['slip']['advice']

def get\_random\_joke():

    headers = {

        'Accept': 'application/json'

    }

    res = requests.get("https://icanhazdadjoke.com/", headers=headers).json()

    # print(res)

    return res["joke"]

if \_\_name\_\_ =='\_\_main\_\_':

    # speak('Good afternoon')

    wishMe()

    while True:

        query=take\_command().lower()

        if 'wikipedia' in query:

            speak('searching wikipedia...')

            query=query.replace("wikipedia","")

            results=wikipedia.summary(query,sentences=2)

            speak('According to wikipedia')

            speak(results)

        elif 'light on' in query:

            print('light on......')

            speak('light on........')

            led\_switch(1)

        elif 'light off' in query:

            print('light off......')

            speak('light off........')

            led\_switch(0)

        elif 'turn on fan' in query:

            print('turning on fan.....')

            speak('turning on fan')

            fan\_switch(1)

        elif 'turn off fan' in query:

            print('turning off fan.....')

            speak('turning off fan')

            fan\_switch(0)

        elif 'light up' in query:

            print('turning on desk light')

            speak('turning on desk light')

            port1\_switch(1)

        elif 'darkness' in query:

            print('turning off desk light')

            speak('turning off desk light')

            port1\_switch(0)

        elif 'turn on device' in query:

            print('Your device is alive')

            speak('Your device is alive')

            port2\_switch(1)

        elif 'turn off device' in query:

            print('Your device is off')

            speak('Your device is off')

            port2\_switch(0)

        elif 'on youtube' in query:

            speak("what you want to watch")

            topic=take\_command()

            youtube()

        elif 'open Youtube' in query:

            speak('opening youtube')

            webbrowser.open('https://www.youtube.com/')

        elif 'open google' in query:

            speak('opening google')

            webbrowser.open('https://www.google.com/')

        elif 'open gpp' in query:

            speak('opening login page')

            webbrowser.open('https://gppune.ac.in/gpp/gpp\_s20/userindex.php')

        elif 'play music' in query:

            speak('hope this will entertain you')

            dir='C:\\Users\\Public\\Music\\Sample Music'

            songs=os.listdir(dir)

            print(songs)

            os.startfile(os.path.join(dir,songs[-1]))

        elif 'the time' in query:

            strTime=datetime.datetime.now().strftime("%H hours %M minutes and %S seconds")

            speak(f'sir the time is{strTime}')

        elif 'stop music' in query:

            os.system("TASKKILL /F /IM wmplayer.exe")

        elif 'read my mail' in query:

            speak('reading your latest emails sir')

            readMail()

        elif 'send email to' in query:

            try:

                speak('what should I say..')

                content=take\_command()

                to='raspberry02pi2003@gmail.com'

                sendEmail(to,content)

                speak('email has been sent')

            except Exception as e:

                print(e)

                speak('sorry sir I am not able to send the mail')

        elif 'advice' in query:

            speak("Here's an advice for you sir")

            advice=get\_random\_advice()

            speak(advice)

            print(advice)

        elif 'joke' in query:

            speak('hope you will like this one')

            joke=get\_random\_joke()

            speak(joke)

            print(joke)

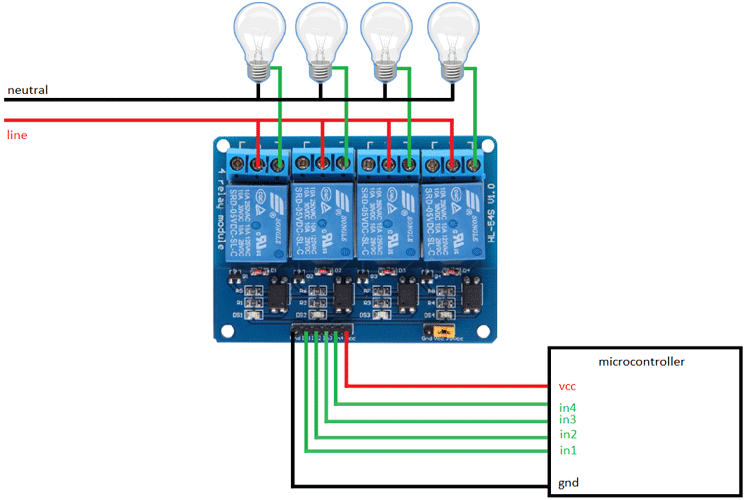
        elif 'exit' in query:

            speak("well see you soon sir Thank you")

            break

**4.8 Circuit Diagram**

The schematic circuit diagram for our project is shown in fig below:



ATMega328p

Fig 4.8 (a) Schematic circuit diagram

Actual final circuit is shown below:

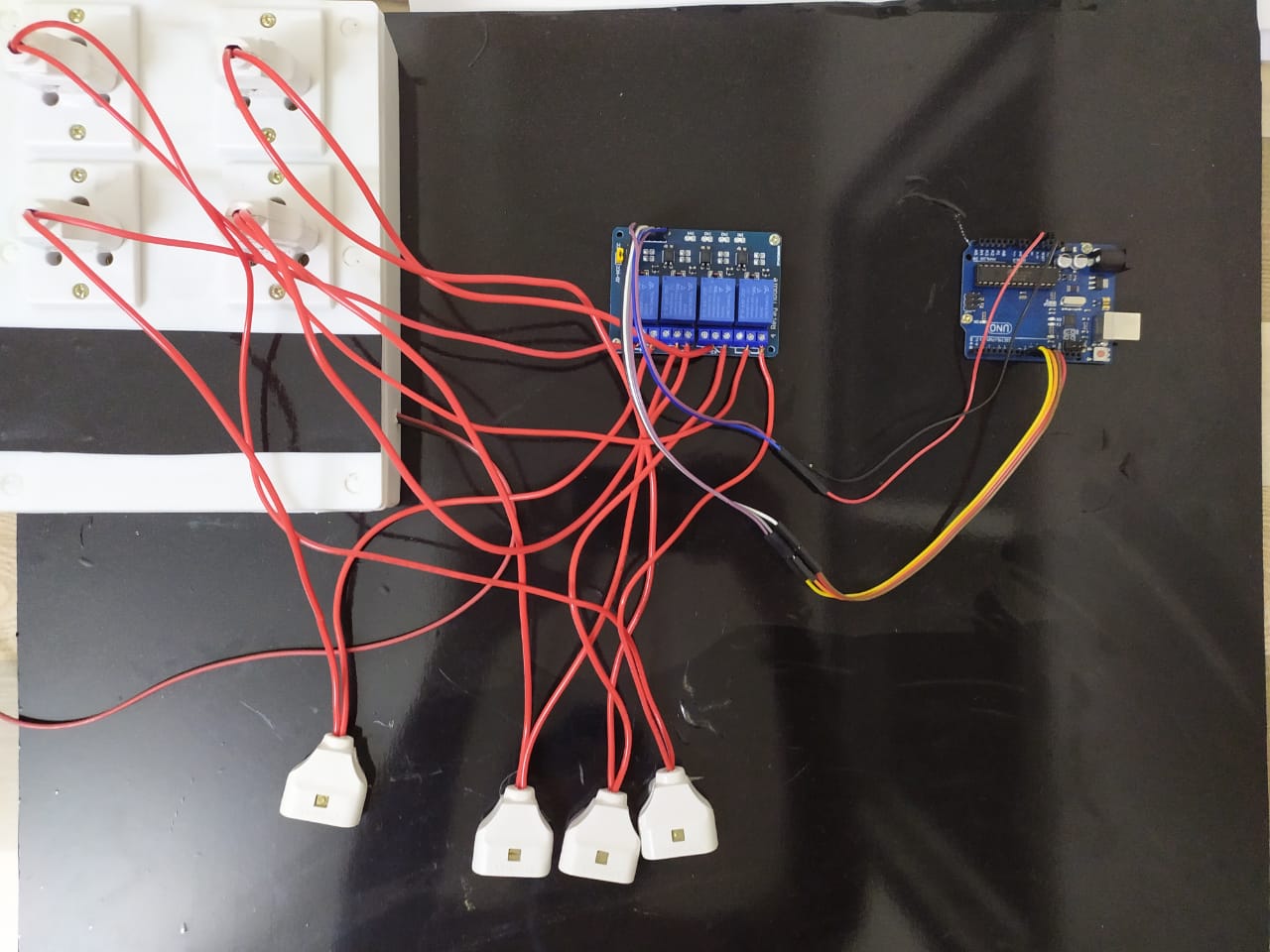


Fig 4.8 (b) Actual Circuit

**Errors Faced**

During the development of this AI assistant we face a lot of errors. Coding of simple task like if-else conditions function definition was an easy task but integrating python code with a microcontroller which doesn’t even compile python code was a challenge. Operating high voltage appliances with arduino and relay module is a bit of risky factor. Besides if the voice in not properly fetched it can lead to misalignment between the user desired task and final performance done by clare.

Sending and reading mails was also a difficult task as it need level decryption skills. Thanks to our respected HOD and mentor we were finally able to retrieve this task.

Combining and assembling was a tricky part cause we have to place power line wires at enough distance to distinguish the NO NC cables and also to avoid the short circuit hazard.

**Future Development**

CLARE is at very naive stage of development and there is lot of potential we can add to this AI assistant.

Our future goals are mentioned below:

* To convert the whole system into Real-Time-System using raspberry pi 4
* To introduce the wake word facility
* From 21 may 2022 google has banned the less secire app permission so we need to create fully automatic mail bot to overcome the problems introduced in send/read email fucntions
* To add selenieum
* For hardware we would like to attach speakers which will increase the volume of virtual assistant

**Conclusion**

Through this voice assistant, we have automated various services using a single line command. It eases most of the tasks of the user like searching the web, retrieving details using Wikipedia, home automation over voice. We aim to make this project a complete server assistant and make it smart enough to act as a full-fledged desktop plus home AI assistant. The future plans include integrating CLARE with mobile using React Native to provide a synchronised experience between the two connected devices. Further, in the long run, Jarvis is planned to feature auto deployment of desktop wakeup and shutdown, raspberry pi integration and all the development mentioned in previous chapter

**References**

* + Websites referred
    - [www.stackoverflow.com](http://www.stackoverflow.com/)
    - [www.pythonprogramming.net](http://www.pythonprogramming.net/)
    - [www.codecademy.com](http://www.codecademy.com/)
    - [www.tutorialspoint.com](http://www.tutorialspoint.com/)
    - [www.google.co.in](http://www.google.co.in/)
  + Books referred
    - Python Programming - Kiran Gurbani
    - Learning Python - Mark Lutz
  + YouTube Channels referred
    - edureka!
    - CodeWithHarry.
  + Documents referred
    - Designing Personal Assistant Software for Task Management using Semantic Web Technologies and Knowledge Databases
      * Purushotham Botla
    - Python code for Artificial Intelligence: Foundations of Computational Agents
      * David L. Poole and Alan K. Mackworth