Framework for reasoning with speech processing

Ahmed Laarfi
Framework for reasoning with speech processing

By

Ahmed Laarfi

A Dissertation submitted to
the College of Engineering and Science at
Florida Institute of Technology
in partial fulfillment
of the requirements
for the degree of
Doctor of Philosophy
in Computer Engineering

Melbourne, Florida
December 2020
We, the undersigned committee, hereby approve the attached dissertation,

Framework for reasoning with speech processing by Ahmed Laarfi

____________________________
Veton Z. Këpuska, Ph.D.
Associate Professor
Computer Engineering and Science
Major Advisor

____________________________
Marius C. Silaghi, Ph.D.
Professor
Computer Engineering and Science

____________________________
Josko Zec, Ph.D.
Associate Professor
Computer Engineering and Science

____________________________
Philip J. Bernhard, Ph.D.
Associate Professor and Head
Department of Computer Engineering and Science
Abstract

Title: Framework for reasoning with speech processing

Author: Ahmed Laarfi

Advisor: Veton Z. Këpuska, Ph.D.

It is known that programming languages are textual. We try here to take the advantages of Speech Recognition (SR) and employ them in creating a verbal Language, which takes its instruction from the voice. Because this work is a novel approach in the programming world, we could not find any resources.

This dissertation aims to make essential developments in Speech Recognition (SR) and Artificial Intelligence by constructing a new compiler that receives commands verbally and executes them.

That means entering data into the Computer by voice commands. This method of input means that we link several major computer topics with several subheadings. For example, there is a significant overlap here between several computer disciplines. The most important fields that we rely on in this topic are Artificial Intelligence applications, especially the distinction of speech by developing a Programming Language that will make a dramatic turning point, perhaps in the concept of programming. Instead of relying on writing, we turn to rely on voice commands.

This Compiler can perform many basic operations like any other compilers, but it receives the commands verbally. The instructions are given orally in a spoken
language such as English, Arabic, or French. By designing such a Compiler, two main themes emerge: 1) a careful study of the interaction between Artificial Intelligence (AI) and Speech Recognition (SR), and 2) a full understanding of how compilers are constructed and how all operations are driven.

This Compiler can be improved by adding a spoken language to the default implementation language, e.g., English. This Compiler can give the user/programmer choices to pick up the preferable spoken language that is supposed to interpret the commands.

Moreover, the Compiler gives the programmer a choice to choose the type of output, whether it is textual or conversational (audio), not just sound.
Many large companies have developed such Speech Recognition Systems (SRS), especially the companies producing Smartphones, Computers, and Laptops. If the translation is taken as a model application, they have not yet developed the perfect systems. The purpose of this paper is to add facilities to the Speech Recognition (SR) software so that it can deal with spoken languages. It gives application developers more flexibility because they can use their languages in programming verbally rather than textually. Any programming languages can be used to implement such a compiler, starting from "Go" to "Java." Those languages are classified into three main categories based on language-design: Adhoc (PHP and JavaScript as models), Copy&Delete (Java, Go, ...), Copy&Add (C#, for example).
# Table of Contents

Abstract ........................................................................................................................................ iii

Dedication .................................................................................................................................... xvi

Acknowledgment: ......................................................................................................................... xiii

List of Figures ................................................................................................................................. ix

The inverted pyramid ...................................................................................................................... 2

PART ONE Chapter One: General Introduction ................................................................. 4

1. 1.1. Introduction ....................................................................................................................... 6

I. “A small change could be a dramatic turning point in the programming world!” ................................. 6

II. “A computer, a stupid machine!” ................................................................................................. 6

III. (a) World Trends Towards Audio Applications .............................................................. 8

IV. (b) Speech ................................................................................................................................. 10

V. A summary .................................................................................................................................. 13

PART ONE Chapter Two: General Concepts ................................................................. 14

1. 2.1. Introduction ....................................................................................................................... 15

2. 2.1. Hardware and Software ...................................................................................................... 16

I. Computer .................................................................................................................................. 16

II. The Compiler: ............................................................................................................................ 18

III. Artificial Intelligence (AI): ..................................................................................................... 20

IV. Speech Recognition: ............................................................................................................... 23

V. A summary .................................................................................................................................. 25
PART ONE: Chapter Two: Anatomy and Physiology of the voice ...............27
  1.  2. 2. Anatomy& Physiology ..............................................................28

PART ONE Chapter Three: Physics&Mathematics.................................31
  1.  3.1. Physics and Mathematics ............................................................32

PART ONE Chapter Three: Accumulation ..........................................37
  1.  3.1. Past work .......................................................................................38
  2.  3.2. Speech Recognition .................................................................51
  3.  3.3. Novelty .........................................................................................55

PART TWO Chapter One: Design and implementation of a compiler includes a language supported by speech recognition in general .....................56
  1.  1. 1. Design a new language .................................................................57
         I. Introduction ....................................................................................58
         II. Why and how do we design Compilers in the middle of the significant congestion between programming language interpreters & compilers? ...60
  2.  1.2. Frog, Amphibious, Language .....................................................62
         I. System Definition ..........................................................................62
         II. Internal and external libraries ....................................................69
  0.1.3. Error handling ............................................................................73
  3.  1. 3. Case Study ..................................................................................75
         I. Saving the voice in an audio file ..................................................82
         II. Using the Buffers’ Technique .....................................................84
         III. Ambiguity ....................................................................................85
         IV. How to Solve Ambiguity .............................................................87

PART TWO Chapter Two: Constructing A verbal Compiler ....................88
1. 2.1. Constructing A verbal Compiler ......................................................... 89
   I. Introduction .......................................................................................... 89
   II. The difference between the Verbal Compiler and the app is a lock of hair!
       ........................................................................................................ 92
   III. A Model of a simple compiler that deals with the voice as an input 93
2. 2.2. How does the Calculator work .......................................................... 96
   I. What are the processes hidden by the example? ................................ 98
2. 2.3. The Programming Languages from Past to Present ........................ 100
   I. The Compiler, a programming language ........................................... 100
   II. How many programming languages are there? .............................. 100
   III. Why do we look at a calculator as a compiler? ............................. 101
PART THREE Chapter One: Future work ....................................................... 104
1. 1.1. Future work ...................................................................................... 106
   I. Extending the model to include large applications ....................... 106
   II. Changing the Programming Strategy ............................................. 109
   III. Creating an Acoustic Model of languages that do not have one! .... 113
   IV. Simplifying the Language Style ................................................... 114
PART THREE Chapter Two: Sum Up ............................................................... 116
2. 2.1. Sum Up .............................................................................................. 117
3. 2.2. Recommendations ............................................................................ 119
# List of Figures

Figure 1: From Electronic Numerical Integrator and Computer (ENIAC), a building, to where?* ................................................................. v
Figure 2: Bridging The Different Language* ........................................... xi
Figure 3: My Father receiving the Libya Championship Cup in 1965/1966 as the team captain ............................................................... xvi
Figure 4: Inverted Pyramid ................................................................... 1
Figure 5: PART ONE .............................................................................. 5
Figure 6: The Computer, a stupid machine* ........................................... 7
Figure 7: A display and control panel in .................................................. 11
Mission Control for the Shuttle program ............................................... 11
(NASA photos-80-6315) ...................................................................... 11
IBM 7094s in the Gemini Real-Time .................................................. 11
Complex. (IBM photo) ....................................................................... 11
Figure 8: Same as Figure 7 .................................................................... 11
Figure 9: ENIAC (1946), 28 ton of Devices .......................................... 13
Figure 10: An Isolated Machine. ........................................................... 17
Figure 11: A physiology of voice production ....................................... 28
Figure 12: Pascal, French Physicist, Mathematician, writer, and inventor
https://www.britannica.com/biography/Blaise-Pascal .......................... 33
Retrieved at 11:23 AM on 11/28/2020 .................................................. 33
Figure 13: Muḥammad ibn Mūsā al-Khwārizmī, algorithms’ developer ...... 34
Figure 14: Arabic Dictionary phonotics ............................................... 43
Figure 15: Grammar ............................................................................ 43
Figure 16: A part of the C++ main program to capture the voice using the SMFL library ................................................................. 45
Figure 17: A sample Code to design a compiler, not included the audio part .... 46
Figure 18a: A sample piece of code taken from Allen I. Holub, COMPILER DESIGN IN C .............................................................. 47
Figure 18b: A sample piece of code taken from Allen I. Holub, COMPILER DESIGN IN C ........................................................................ 48
Figure 19a: A sample code of DLL according to Allen I. Holub, COMPILER DESIGN IN C .................................................................... 48
Figure 19b: A sample code of DLL according to Allen I. Holub, COMPILER DESIGN IN C .................................................................... 48
Figure 19c: A sample code of DLL according to Allen I. Holub, COMPILER DESIGN IN C .................................................................... 49
Figure 20: The most common-used graph that is expressing on Communication via Spoken.................................................................52
Languages.................................................................52
Figure 21: PART THREE WORK IMPLEMENTATION..............................57
Figure 22: Recursive means that the Compiler is any high-level language that can program a set.................................................................59
of programs. By the new Compiler, another new compiler can be produced.......59
Figure 23: Basic Operation of the Speech Recognition System...................63
Figure 24: Compiler Construction Operation........................................64
Figure 25: Stages of an audio compiler..................................................65
Figure 26: For example, the part (processor) of an audio compiler could be broken down into two processors.................................................................66
Figure 27: More explanation about regular Compiler’s steps. .......................68
Figure 28: The internal and external libraries..............................................69
Figure 29: An example of an Internal Library; C#.....................................70
Figure 30: SFML, One of the external library that serves C++. Retrieved from: SFML (sfml-dev.org) on 11/30/2020 at 18:57 .................................................................71
Figure 31a: SMFL, an External Library. Retrieved from: Download (SFML) (sfml-dev.org) on 11/30/2020 at 19:11.................................................................71
Figure 31b: 1. Fig. 29, 30, and 31 clarifies how to download external library.
Retrieved from: SFML 2.5.1 (SFML / Download) (sfml-dev.org). On 11/30/2020 at 19:27 72
Figure 32: Error handling Menus. Prepared by author..................................73
Figure 33a. Prepared by author.................................................................75
Figure 33b. Prepared by author.................................................................76
Figure 34a. Prepared by author.................................................................77
Figure 34b. Prepared by author.................................................................78
Figure 34 c. Prepared by author.................................................................79
Figure 35: An example of how to capture a voice and use it as a command. Prepared by author. .................................................................80
Figure 36: The Ambiguity problem. Prepared by author.................................86
Figure 37: A verbal Calculator [A program Run]. Prepared by author..........93
Figure 38: The result of multiplying two..........................................................94
numbers [A program Run]. Prepared by author.................................................94
Figure 39: A verbal Calculator. Prepared by author........................................94
Figure 40: An error occurred [A program Run]. Prepared by author ..........95
Figure 41: General Idea about how the verbal Calculator works [Flowchart designed by the authors].................................................................97
Figure 42: Our Calculator is a compiler [Prepared by the Authors].............102
Figure 43: A Pyramid prepared by author ..................................................105
PREFACE

"Two Chinese speak Chinese, but they need a Chinese to Chinese interpreter process."

SR is defined as speaking to computers in any language. The process is described through complex technical operations, which will be addressed later. The number of software packages that utilize SR has been massively increased. Accordingly, software is commonly used. Their importance is growing because many devices use this new paradigm. The software has become an essential application of Laptops and Smartphones, and other critical electronic devices. The Compiler is a “virtual” part of an interpreter between two people who speak entirely different languages.

Figure 2: Bridging the Different Language*
Most of the traditional translators in programming languages are textual. Some changes have been made to enhance the translators by entering other types of files such as sounds, graphics, and videos in different forms and extensions. Known computer languages that are numbered in the tens, if not hundreds, accurately represent the Compiler. All compilers rely on a language's rules, which are fixed but mostly similar to each other. The compilation of the Artificial Intelligence of SR is quite like a person who speaks Japanese in a local dialect that is unknown even to the other Japanese. Many translators who have studied Japanese grammar are often unable to translate it correctly. The intended use of such applications to advance the most important modern technologies is advocated.
Acknowledgment:

“Your success is the greatest punishment for those who don't believe you!”

To my wife: Ismahan, who did the “impossible” to keep our family active and happy. She sacrificed for my autistic son, Yousef, and the rest of the family. She is a specialist doctor (Radiologist) who was supposed to be examined to obtain a practice license, and she preferred to take care of the family and include our son, a severely autistic child, with care. Since we came to the US a year ago, she drove me to all the places in and even about two hundred miles away for medical checks and consultations. When life became tight, she used to be a substitute teacher in our district's schools.

To my dear daughters, Aram, Elyaa, and Ebaa, who are excellent in their studies!

To my dear (loved) son, Youssef, because he is the youngest and sick.

My professor, Dr.Kepuska, accepted me with him after my previous supervisor's resignation while I have no scholarship left. Despite the radical difference between what I studied and worked on for three and a half years and his field, he accepted me. He did not require additional courses with him, although that was necessary, and he opened his library and gave me all the required references to understand the topic. The first time I heard about this topic was when I met him when he told me that I should change my major. He was the one who gave me complete freedom to work, fly high, publish, and create. I published nine papers when I was with him in a year, in which he participated in a couple of them.
To my companion for about thirty years, whom I have not seen for twelve years, we just have kept in touch, and he was a supporter in the hard days, Dr. Osama Ben Omran.

To my friend John Almasi wishing him a full recovery

To my brother, Dr. Fawzi, who runs my business in Libya despite his busyness

To the open-minded members of my Doctorate committee

To everyone who extended a helping hand to me in this hospitable country.

To all of those, I acknowledge my gratitude for their services, help, and support.
Acknowledgment:

Thanks and appreciation to Professor Earls, Susan, who was decommissioned. She was consistent with her convictions, and she stood with me in a delicate moment with moral support and advice. However, she did not accept me in the group that she supervised due to exceptional circumstances.
Dedication

To my Father, the "absent" present, who passed away months

To that great man, who lived honest, sincere, and beneficial to others and society throughout his life.

To that wise man who renounced violence throughout his life. He retired from politics in stages of his life and settled for the social role.

To my Father, who was a Kind, firm principle, who has raised us the best education and teaches the noble values, the most important of which is the love of knowledge and morality's virtues. To him, I dedicate this work.

Regarding a poet’s description of his Father’s lament

"My father .... his heart is a boy's heart ... and his patience is the patience of a Prophet ... may God have mercy on all of our fathers."

Figure 3: My Father receiving the Libya Championship Cup in 1965/1966 as the team captain.
Figure 4: Inverted Pyramid
The inverted pyramid.

In the world of journalism, this is one of the paths the editor takes. It starts with the most important, then the important, and ends with additional small details.

Scientific writing tends toward direct reporting. It is difficult to understand the topic if we start with the conclusion and finish with the introduction. Scientific writing is characterized by the sequence and introduction to what is later. So I will not start from the back, but rather as is usual in scientific writing. Nevertheless, the most obvious thing is that science is a cumulative process. When the search is done, we find an enormous surplus of information, and the range begins to narrow until we reach a conclusion, which is the size of a pin.

I was young, walking on my hands, struggling to keep my balance. It is usual for us to walk using legs. However, the inverted pyramid is also a scientific writing style with the correct understanding that everyone follows.

First, the shape caught my attention as I noticed that it looked like an inverted pyramid. First of all, I wanted to adjust it, and it is a matter of seconds. I kept it because, by pure chance, this figure represents reality with all its light and dark backgrounds: meaning the closer to the end, the narrower and fewer options.

We were young people using a game like this to spin a string around and throw it forcefully, holding the string’s head. The stronger and more accurate the throw is, the longer the game runs before it stops to fall.

Movement is essential for humans and in life in general. From personal experience, I can say that an organ in a person you do not use will lose it.
Here for me, it works in both cases, upside down or upside up. Nevertheless, it has profound implications that can be dropped on the reality in which we live. This is why I kept the pyramid upside down, not in the journalistic sense, of course.

What matters is that we try to simplify the explanation of the topic.
PART ONE

Chapter One: General Introduction
Figure 5: PART ONE
1. 1.1. Introduction

I. “A small change could be a dramatic turning point in the programming world!”

Artificial Intelligence is one of the first branches of computer science, and it is not new. What is new in our topic is the connection of Artificial Intelligence through Speech Recognition applications to create a new concept that may be a dramatic turning point in programming, which is audio programming. That is, giving commands by voice to be executed. By the way, I would like to remind the reader at the start that there are no sources for this work because the idea is unprecedented.

II. “A computer, a stupid machine!”

The Computer has no intelligence.

People have IQ levels ranging between eighty and one hundred and twenty for an average person. Computers are meaningless devices (hardware) without operating systems, software, and programming languages. Who creates these programs that control the Computer is a person, and at the same time, he is the one who manufactures and assembles computers. When someone describes human Intelligence, saying "like a computer," we fall into a logical problem expressed by the idiomatic one. Because a creature always acquires some, say, very little, of the attributes of the Creator. A human being created the Computer here. If natural intelligence ratios are minimal, then computer intelligence ratios are a small part of a person's Intelligence.
We agree that it is a very accurate machine, and when well programmed, it is more accurate than a human being. It does not deviate from the path prepared for it, which does not mean that it is a smart machine. A computer is a device that follows the same algorithm and does not create or add anything new. Man always changes his ways and methods to solve the same issue. The attribute of persistence gives the accuracy of the recipe of thinking gives Intelligence. There is a new branch in the field of Artificial Intelligence called artificial neural Intelligence. This branch is trying to develop the level of "intelligence" of the machine. This topic is a different big topic. However, after a year or a century, we may find a machine smarter than humans. In our time, most dreams come true. Hopefully, not!
III. (a) World Trends Towards Audio Applications

“We live the future in our present!”

One of the most important features provided by the Computer’s audio input is that it frees users from several activities. For example, there is no longer a need to use hands or any other type of input. Moreover, a quick insertion-input method of input frees the eyes from focusing on the input mode.

Of course, such applications have many problems since their launch four decades ago until today. The acceleration of technological developments leads to improved quality of such applications. Users rely more and more on the usage of such applications that utilize SR. The spelling of commands and the commonly used input units are utilized. A separate word may be stored in a variable of a specific type declared by the compiler designer. Symbols, erratic phrases, or related sentences can be stored as variables according to the entered data types. The storage is completed by the interaction between the Compiler and the human. Ultimately, the final confirmation should be affirmed by the yes or no question. Thus, we have variables stored in memory addresses. After storage, Artificial Intelligence is introduced, leading to continual updating, whether correcting previous errors or keeping the data available while storing any new information. The date is requested, 8 digits are expected to enter: (2 digits for the day, 2 digits for the month, and 4 digits for the year). In the 1990s, the Computer with 2 megabytes was considered a good machine, but 4 megabytes was excellent. At that time, video files were the most resource-intensive, followed by high-resolution image files. Audio files could not be ignored in terms of the required allocated space. The text files are the smallest in terms of memory size. Nowadays, computers have vast space available without considering the additions of hard disks and external storage devices. Although abundant memory space is available, the
concern is the files’ sizes being as small as possible. Therefore, the voice information is stored as text data; this feature allows users to handle input/output files as audio files simultaneously, such as byte/text files.

When a computer does a human's job, I do not know whether we think it is good luck or bad luck. Of course, there are a large number of people who will be dispensed. This dismissal is, of course, beneficial to the business owner. If the laid-offs are not rehabilitated, this will cause problems. We delight in progress, development, and innovation.

On the other hand, we find that it is harmful to some. An example is the Coronavirus pandemic. Many businesses have been abandoned or merged. Everything is now managed electronically by video, in several areas, the most important of education. In this situation, many lose their jobs.

On the other hand, it proved the effectiveness of these inventions, especially technology. When I defended my doctoral thesis in front of the ethical committee, we used the video conference technique. This fulfilled many physical location requirements, special display devices and computers, and possibly papers and secretaries. Yes, it became more accessible and comfortable.

On the other hand, this causes pain for some of those losing their jobs due to merging with others. However, these apps solve combining and isolate problems. This is a general example of technical development, and it includes Speech Recognition. SR, after the reduction in people's direct contact, has a significant dependence. Work is home. The conversation became between man and machine. Transfer to a representative is not done before all the available capabilities with the machine exhausted without a solution.
IV. *(b) Speech*

*“When you talk, we know clearly what you want!”*

It is not fully known how the ancient man communicated. Sign language could have been used at the time. Sign “Communications” is very limited in expressing the meanings, although it is widespread nowadays, communicating with the deaf and dumb and between them, and rules have been set for it. Sign languages remain deficient even if it is at the level of modernity it has. Indeed, the sign languages in the Stone Age were very primitive and had no rules and syntaxes. If a person is unable to speak in nature, he will not be able to write and read.

Nevertheless, speech is essential to humanity beyond what anyone imagines. Talking is communication, and without communication, life becomes difficult, complicated, and boring as well.

According to our elementary studies, the first language I remember was called the "cuneiform language." That was after which other spoken languages closer to perfection have been appeared and developed. The critical idea is that people are making understandable sounds, replacing the signals, of which the incomprehensible sound is one of them. These languages are organized with rules and syntaxes that govern them, linguistics, letters, pronunciation, spelling, reading, and writing. From here began the real life in which people communicate and understand. Man is no longer the same human being who previously resembled an animal and communicated exactly as an animal does. From here, the stage of knowledge and accumulation began, which is the honor of the man and his preference for other beings' reasons. It is knowledge, science, and reason that have led anthropologists to describe humans as "talking animals." They did not say an animal that thinks, and this distinguished speech and made it a priority. The thought results
from reason, logic, experience, and accumulation. The most crucial science tools, whose primary essence is speaking, reading, and writing.

The same idea can also be measured in the computer world. Without the communication networks that started with the four American universities sponsored by NASA in 1962, computers would not have the (strategic) importance today. Without communication, computers become almost isolated islands empty of people.

Figure 7: A display and control panel in Mission Control for the Shuttle program (NASA photos-80-6315) IBM 7094s in the Gemini Real-Time Complex. (IBM photo)

Figure 8: Same as Figure 7

Retrieved from

[https://history.nasa.gov/computers/Ch8-2.html]
at 8:28 pm, 10/29/2020.
Networks and communication media are what can be called "talk" between devices. Today is different from sixty years ago. We can reach every place and every device, making the writers call the world a "small global village."

We have our virtual-world side by side with our real-world and even intertwined with it. Our financial dealings became virtual through computers or mobiles, but they are accurate and reach the target in reality. There are social media, online markets, and cybercrime as well. Besides, a virtual currency (real) that appeared in its market in which it is used is the Internet. All sciences, especially computer science, have overlapped and interconnected to create these "smart" applications for us, despite my reservations about describing them as “intelligent.” The Computer got into everything that can be envisioned or even not think of it. One of the most important Artificial Intelligence that runs at us like a wild horse that seems to have not yet strained to stop running.

There are many Artificial Intelligence Applications (AIAs), and what we are particularly interested in here are the Applications of Speech Recognition (SRAs). Attempts to make audio applications appeared at the end of the seventies of the last second millennium. Today, after forty years of innovation and development, such applications have become familiar to everyone and even used daily, whether by our will or imposed on us. Most, if not all, parties are responding automatically, and dialogue continues between you and the machine until your problem is solved or you reach a representative to solve it for you. Sales and purchase transactions, hotel or clinic reservations, even paying bills or contacting companies, all of this is done automatically without human intervention.

I had an excellent opportunity, which is the honor of meeting one of the scientists in this field and innovator, Professor Kapuska, before supervising my Ph.D. thesis. We tried to create a new
topic that no one had previously touched upon: the Compiler's subject, which produces a speaking language that receives orders orally. This addition may be a turning point in programming languages, but it is an essential contribution to innovation and development.

V. A summary

![Figure 9: ENIAC (1946), 28 ton of Devices.](https://www.zdnet.com/article/eniac-one-of-the-first-computers-turns-60/)

Indeed, the world after 1943 is not yet the same as it was before it. This year is the year the Computer was invented. We will not exaggerate and compare speech recognition to human learning to speak, write, and languages. However, after the branch's evolution, I do predict ideas more imaginatively. At that time, speech and language were for dialogue between one person and another person. We are talking about dialogue and a conversation between a person and a machine, a device, or a robot.
PART ONE

Chapter Two: General Concepts
1. 2. Introduction

"All inventions begin to serve military purposes, and then we use them later in all aspects of life."

It is imperative to go through some of what is classified as computer science or general sciences related to our topic, which is the applications of speech recognition (SR). We took a calculator to simulate it, as it has limited tasks, and based on it, the project can be expanded to include larger applications such as databases and simulations.

Research usually leads to discovery. We found that canceling the first stage in the Compiler for manually entering data leads to a very large time-saving. The time-saving occurs in the use of symbols’ text in the first stage of Compiler Construction. If we use the high-level language to program with the same Compiler technique, the DLL will help speed up the run. That means any system is programmed recommended to use the programming technique used in the Compiler to build its library. We get rid of programmers’ mistakes repeating the code, create unnecessary loops, and get rid of whitespace. A new Programming Technique that avoids programming through high as far as it can.
1.2.1. Hardware and Software

I. Computer

"An ongoing race between hardware and software."

"A computer is an electronic device that manipulates information or data. It can store, retrieve, and process data."

Retrieved at 6:49 pm, 09/03/2020

https://edu.gcfglobal.org/en/computerbasics/what-is-a-computer/1/

The above description is the most straightforward and concise definition of this fantastic machine. In reality, computers are much more complicated than this meaning. In the beginning, it is viewed from two sides, and after the emergence of complete-performance operating systems, it has three aspects. The software, hardware, and Firmware might be considered as categorization. Each one is a world by itself that includes different sciences and applications of the degree of disparity. When the Computer's invented less than eighty years ago, it was a giant set of machines that might not have room to encompass them all together. Its capabilities were minimal, and in absolute terms, it was used for specific calculations. There were no operating systems or programming languages. The cost to build this machine was so huge that merely saving two digits in the memory was a great deal. For example, the date consists of eight digits. Still, in the beginnings and for the next twenty years, the engineers designing the hardware used to put the date in six numbers so that the year is removed from it, and the two places of the
century are considered constant (19). Fifty-seven years later, this resulted in the so-called 2000 (Y2K) crisis and the financial and physical damages.

![Figure 10: An Isolated Machine.](https://www.123rf.com/photo_96075680_old-computer-one-of-the-very-first.html)

Old Computers.


Until that date, a gigantic (different) machine was available with devices and chips considered advanced in its time. Still, this advanced industry in the (hardware) part did not keep pace with any suitable capabilities for operating (software). Yet, the race began between the hardware, which was always and until recently in software stages. The excellent point is that with the years' progress and the manufacturers' competition, the Computer's financial value has decreased, and its size has diminished. We hear a lot about a term that resembles a human being with a computer, which is a big mistake and confusion in concepts. A person is the one who made and programmed computers and has varying levels of Intelligence. Whereas we find that computer IQ is equal to zero, so it is a stupid machine unless they are programmed. Even they are
programmed, they are still restricted and unable to create. For example, if the description is "Computer-accurate," it is accepted. Currently, computers are available from all sides on converging capabilities in performance, and there are no more vast spaces in (the lead) separating them. The human-being forced the Computer to listen, speak, physically perform actions, control, simulate, and do everything imaginable in dreams. In light of the hardware's innovative capabilities, the designers of operating programs, programming languages, programs, and applications could reduce the difference in the hardware's interest, and the work is still going on.

II. The Compiler:

"Chatting with the computer!"

One of the most straightforward definitions of the Compiler is: "A compiler is a special program that processes statements written in a particular programming language and turns them into machine language or 'code' that a computer's processor uses."

09/09/2020 at 19:23 retrieved from https://whatis.techtarget.com/definition/compiler

In other words, it was previously defined as a translator interpreting between two, speaking the same language, which means that it is not translated in the linguistic sense of the word. Compiler writing has a standard style. As such, it is the programming language itself, in which we write its commands in a known language to receive the results in the same vocabulary.

The Compiler is a set of stages. Each stage accomplishes what is required of it and takes to the next step unless something is wrong. The Compiler is a programming language that is programmed by another programming language. Most of the popular programming languages, operating systems, and applications are programmed with C++. Any programming language can
be used to create a new programming language. It is possible to design a programming language other than the rules intended for building compilers. In studying advanced programming, a subject we studied in the B.Sc, we implemented a compiler for this course. It was in Pascal's language, which is now out of use. It was primitive and intended to implement a set of instructions by way of commands. The designer determined how to write them. For example, instead of writing the print order, whether on the screen or the printer. We change the command to send it to the screen or the printer with the understandable traditional commands. We process the order in rudimentary terms with what we have available to us from the commands and design libraries that support the commands in units that include procedures and functions. This method is chaotic and causes many problems in speed and the optimal use of the Computer's resources and devices. We do not notice the delay that it may cause, but in the programs that need time to be implemented, this time is doubled, and the matter becomes tedious, complicated, and not guaranteed. Standard wording of the compiler stages is the perfect way to save resources. Building libraries in a recognized programmatic manner organize them first and, secondly, making the implementation process smooth and problem-free.

Why do we need to design a new language amid this terrible crowd of programming languages, mainly if it does not serve a new programming branch? All programming branches have languages that they serve. So you do not need to spend much time designing this new language. The design and implementation of a new compiler take five to seven years by a specialist. If a work team implemented it, it would require zero to the last stages, including tests and modifications, which would require an average of eighteen months. If we do need to do such a duty, it will not be at the same level as the available languages. Our novel approach is to connect the Compiler to the Speech Recognition applications and coin a verbal compiler. We need to
keep abreast of developments in this new technology developing very quickly. We need a
speaking programming language as well as other applications. All applications today have a
voice command component added. Programming languages are still the only ones that are devoid
of communication and interference between the user and the Computer utilizing sound. There
have been attempts in writing commands into a programming language using voice instructions,
and they are partially succeeded. This feature can be used, but it is not the subject of this
research. We can use it to write programs as a verbal editor, not to give instruction. That is why
it was necessary to learn all the details of compiler programming. Here the resources are
available in abundance. To avoid perpetuation, we will only refer to the essential and useful
details of our work later. By studying the phases of the Compiler in-depth, I found that higher
languages pose many problems. Some of them lead to delays and the elimination of resources
that should not be taken.

III. Artificial Intelligence (AI):

"The simulation of human intelligence in machines that are programmed to think like humans
and mimic their actions. The term may also be applied to any machine that exhibits traits
associated with a human mind, such as learning and problem-solving." The definition is a simple
description retrieved at 09/12/2020, 9:54 am from:

https://www.investopedia.com/terms/a/artificial-intelligence-
ai.asp#:~:text=Artificial%20intelligence%20(AI)%20refers%20to,as%20learning%20and%20problem%2Dsolving.

It is necessary to emphasize that a person's intelligence is not comparable to that of a “stupid”
machine. That machine's IQ without human intervention is equal to zero. No matter how bright-
titles we put them, such as Artificial Intelligence (AI) or Natural Artificial Intelligence (NAI), an
adaptation between AI and Natural Intelligence, the Computer remains a deficient machine that only adheres to the rules set by the programmer. If it deviates from these rules, it will give the recipient any result other than the correct work. We are the ones who set the ground rules in all branches of computers and make it appear as though it is a thinking machine. Also, do not lose sight of the fact that artificial intelligence applications are endless simultaneously.

The first rule we learned in artificial Intelligence is that if Thomas is John's Father, and John is Michael's Father, Thomas is, by logic, the grandfather of Michael. Nevertheless, we found that such rules do not work in all cases! If we say that: Sarah is Michelle's sister and that Michelle is Jamal's sister, Sarah does not have to be Jamal's sister. Sarah may be Michela's sister from the mother's side, and Jamal is Michela's brother on the Father's side, so they are not brothers. By applying Mendel's heredity laws, we find that the proportions of truth-table between Jamal and Sarah being brothers or not being brothers are equal.

Based on the first inaccurate definition that may lead to an error, we must introduce another variable to increase accuracy. This variable determines whether the siblings are from one of the parents or the two parents. Here the percentage increases in the accuracy of the information.

Suppose we design a chess game between a human and a machine so that the device performs constant moves. Using Napoleon's plan, humans can defeat the machine in five to seven moves. Here we must improve the performance of the machine "with artificial intelligence." When the device recognizes through the prepared rules that the first moves of a person are the beginning of Napoleon's plan, it can disrupt the strategy and not make the Shah Cheek-mate in only five to seven moves. Other plans end the game in seven moves. These are elementary and primitive rules, but we can accumulate on them.
This is what we call the accumulation of the above information. The same applies to the fields of using Artificial Intelligence in medicine and other areas.

Artificial intelligence applications have become among the most common and overlap with all branches of computer science. In the field of robotics, many of us think of them as AI applications. Of course, yes, but robots are now a stand-alone science and a discipline. There are many branches of computer science related to artificial Intelligence. Most of them are robots, game programming, Cybersecurity, and Speech Recognition, but they have become independent disciplines. The curious idea about the issue is that when a person thinks about something, he finds it in front of him on a computer, smartphone, or he receives it via an email, such as an advertisement.

Artificial intelligence applications, of course, do not read the ideas, but you may have searched for what you thought using Google. Cookies are then used to fill you up with endless advertisements for several days on social media, email, or websites in general.

By Philip C. Jackson

Introduction to Artificial Intelligence: Third Edition

https://books.google.com/books?hl=en&lr=&id=vC-oDwAAQBAJ&oi=fnd&pg=PA33&dq=introduction+to+Artificial+Intelligence&ots=XLU-YBGCzq&sig=uDMOtAMUiVZgzGQ8R6JF5HtLQ#v=onepage&q=introduction%20to%20Artificial%20Intelligence&f=false
IV. Speech Recognition:

“Speech recognition, or speech-to-text, is the ability for a machine or program to identify words spoken aloud and convert them into readable text[…], and it may only identify these if they are spoken very clearly.”

Karolina Kiwak, SearchCustomerExperience: Retrieved from
[https://searchcustomerexperience.techtarget.com/definition/speech-recognition]

At 5:51 am, October 22, 2020.

“Speech recognition is the capability of an electronic device to understand spoken words. A microphone records a person's voice, and the hardware converts the signal from analog sound waves to digital audio. The audio data is then processed by software.”

TechTerms - The Tech Terms Computer Dictionary: Retrieved from
[https://techterms.com/definition/speech_recognition]

At 5:59 am, October 22, 2020.

• The world is advancing so fast that you are thinking of something new, and the next day it is on the market.

• Computer products and applications have accelerated our lives till the day passes as it was just minutes.

• The age of speed, age of the Computer, age of communication, or the age of knowledge, the multiple names are all overlapping.
• Speech Recognition has been introduced to be one of the computer branches.

Speech Recognition applications:

• With smartphones, tablets, and high-speed laptops, the use of Speech Precognition systems (SRS) has become widespread.

• It is difficult to limit the areas in which it is used, but it can be said that it exists in most sectors, industries, utilities, communication with any private, public, or even person, and ending with robots and space.

• The question is, what the Speech Precognition is?

• What does this title hide from details behind it that most users do not see?

• How does it work?

SR is a part of Artificial Intelligence, mainly its applications:

• SPEECH TO TEXT

• TEXT TO SPEECH

• SPEECH TO SPEECH

• These are the most common-used applications used in srs.

• The applications are then implemented using any programming languages, and many applications are provided free in stores or offers by the producers.

• There are many difficulties solved and not seen by the users.

• There is much science stands behind the interface that we deal with.

The Races between Computer Hardware and Software!
V. A summary

“The Compiler, the parent of applications!”

The group of computer science and other related to it has overlapped here to perform this simple
task. We have not talked about other sciences, such as physics, phonetics, mathematics, and
logic, which also exist in addition to an understanding of hardware and lower languages, at least
as a reasonable understanding.

The difference between the Compiler and application is a piece of hair!

Any system, in general, is a limited-edition compiler. For example, the payroll system, personnel
system, cost control, or other database systems, when they are programmed, come out to us in
the form of an application. This app is executing the limited commands available to it. Entry is a
command, amendment, cancellation, display, printing, backup, and reports designed by the
programmer, regardless of its shape, method of implementation, and language.

The Compiler is a programming language that could be considered a low-level language that
directly deals with hardware.

The programming language is a high-level language that needs a compiler or interpreter to work.
We can consider them as comprehensive applications!

We used the term "Software packages" before the new nickname, "Application."

Microsoft products are applications lookalike programming language but for specific tasks!

Excel can do many tasks, especially in the presence of "Macros."

25
Our system, like payroll and personal systems, are also applications.

We figure out that the Compiler base is also an application by the logic, but it is comprehensive!

One previous personal experience of mine was to design a simulation system in Pascal.

At that time, there were no "Menus" available for the programmer within the programming language.

We had to calculate the screen in pixels and the number of lines and columns and choose the point (0,0) to start and end with another point.

We divide the list into some drop-down Menus in the same way. We give the entire screen a color. We distinguish the Menu by a different color.

We indicate what we have chosen from them as a rectangle in another color, for example, red.

When we move from a rectangle to the one next to it, we face a very complex programming process that contains tens and perhaps hundreds of code lines. For example, we return the rectangle's distinctive color to the list's natural color by pressing the Move arrow. Go to the rectangle next to it and change its color to make it unique.

By the way, Pascal is a dead language even it is easier than C, but it works as C, so it is called the sister language to C. Because of the similarity, the more substantial stays alone.
PART ONE
Chapter Two: Anatomy and Physiology of the voice
1. 2. 2. Anatomy & Physiology

Anatomy and physiology of Speech

“Speech sound is a wave of air that originates from complex actions of the human body, supported by three functional units: generation of air pressure, regulation of vibration, and control of resonators.” *

*Figure 11: A physiology of voice production

Retrieved from: [https://www.google.com/books/edition/_/P-g3DwAAQBAJ?hl=en&gbpv=1&pg=PR1&dq=springer+handbook+of+speech+processing+pdf] in Nov., 5, 2020 at 3:30 pm
The importance of voice

An essential characteristic of a person is the five senses he possesses. The sound in the classification of the five senses of man does not exist as a unique sense. Because it is closely related to the sense of hearing, we can consider it among the tools of this sense.

Voice is critical to a person. The human being expresses himself through speech, an organized group of sounds that form a meaning.

All people communicate through sound. The expression of all the situations that appear on the human also is done through the voice. These situations are like sadness, love, and connection with others. Giving personal information is also done through voice. Human thought is produced through sound.

The sound is produced according to a fixed and regular mechanism. Humans use the organs responsible for producing sound. We find that vocal strings are the primary source of sound production. It controls the high and low volume of the sound. The tension and relaxation of the vocal strings lead to the diversity of sounds. The size of the larynx controls different sounds.

The larynx, pharynx, trachea, esophagus, diaphragm, and spine; all of these organs are involved in sound production. Air exits from the lungs through the trachea, and as it exists, it passes into the larynx, causing the amount of air flowing into the larynx to vibrate the vocal cords. The vibration of these vocal cords traps the air inside the larynx and releases it alternately. Each time air exits the larynx, a little of it goes into the throat, which is the beginning of a new sound wave. When air moves up the throat and exits the mouth, the sound is produced.
There is interesting information about the thyroid gland and its diseases. Most doctors refuse to perform surgery for singers on the thyroid gland. This is because the vocal cords are located in the same area as the thyroid gland. Umm Kulthum is the most significant Arab singer; all doctors refused to operate on her thyroid gland for fear that they would touch the vocal cords.
PART ONE

Chapter Three: Physics & Mathematics
1. 3.1. Physics and Mathematics

"Pascal, the great scientist, all he did was make a calculator!"

Is the accumulation
Figure 12: Pascal, French Physicist, Mathematician, writer, and inventor
https://www.britannica.com/biography/Blaise-Pascal

Retrieved at 11:23 AM on 11/28/2020
If we consider that the mother of all applied sciences is physics, there is no doubt that mathematics is the applied tool “Arm” for physics. We are not talking about physics applications, but what matters to us is that the computer combined physics and mathematics with seeing it in this way, whether it is hardware or software, operating systems, and programming languages.

All branches of computers start mainly with physics and mathematics. On the software side, we cannot design a system unless we study it mathematically and physically. We translate the study into algorithms, mostly mathematical equations, and finally reach the design stage. Algorithm referred to an Arabian Scientist called Muḥammad ibn Mūsā al-Khwārizmī.

![Muḥammad ibn Mūsā al-Khwārizmī](https://learnsomethinginteresting.com/2019/05/23/the-mathematical-inventions-of-muhammad-ibn-musa-al-khwarizmi/]

*Figure 12: Muḥammad ibn Mūsā al-Khwārizmī, algorithms' developer.*


Logic is the basis for programming, even if the computer logic is different from standard logic. Logic is all about mathematical relationships. Counting systems are one of the most critical uses
until we got to this current computer. All mathematical calculations are used in programming operations. On the other hand, we find that hard computer components are materials and have physical properties and mathematical calculations.

An isolated computer is a computer that is lifeless until connected to a communications network. This network is like the lung through which he breathes. The computer would not have had this (strategic) importance if a group of devices, each one containing little information that it did not share with others, would have been.

I like the famous saying about the apple and the idea. If you have an apple and an apple and exchange them, there will be only one apple. However, if you have an idea and an idea and we exchange ideas, then each of us becomes two ideas. Why don't you share information with millions if not millions of devices have walked? You then have a wealth of information. A paper was presented on the importance of electronic archiving eleven years ago. Its title was: Life is a Huge Archive. In it, she was exposed to the number of daily information that multiplied on the Internet, specifically on social media. It dealt with the fact that the amount of information since the inception of humanity to the seventies is less than the amount of information produced. I mentioned that a newspaper like the Washington Post, in one issue, is an imaginary amount of information. This information may be equivalent to the information an average person knew centuries ago in his lifetime. Thus, no one can imagine the world without computers and means of communication and communication. A university student today, when he graduates, will have read twice as many books as scholars of previous centuries, with the quality difference in the reader, of course.

All this was the result of scientific accumulation. It started with a mathematical equation and ended with something like science fiction. Nevertheless, it is continually accelerating.
I am talking here about physics and mathematics’ relationship to the related sciences in my topic in a hurry. When we talk about Speech Recognition applications, the first things we find before us are anatomy and physiology. The nature of the sound, how is it produced, who is responsible for it? These are functions performed through physical properties. This is when talking about the natural voice. Come to the process of picking it up and employing it. These have many mechanisms, starting from mathematics in all its sections. When specific sound capture methods are programmed, we go to mathematical equations. The same is the case in the compiler and other branches of computers. Through artificial intelligence, which is a science from which the distinction of speech was based.

The first thing that made computer hardware developers a talking machine started software developers' efforts to employ this qualitative development. The beginnings were difficult; however, specific models were adopted, but many things changed after Windows's advent. Finally, Smartphones have revolutionized the use of this application, which is the Speech Recognition. All details related to mathematics and physics.
PART ONE

Chapter THREE: Accumulations
1.3. 1. Past work

I. Past work in general

The development of the Computer came as a result of accumulating achievements and building upon them. It started with limited computer science capabilities and then expanded until it reached what it is today.

When selecting a computer in the past decades, the facilities were weak in memory, speed, and storage, both internal and external. The devices were large, expensive, heavy, with limited capabilities. The device and the systems that run it were considered a little old after a year, and after a couple of years, the device must be upgraded, which can only be done by purchasing a new device and new operating and programming systems. The assembly of the Computer took place in four stages. The highest level is assembly by components, such as a hard disk, a computer case, and a power supply, a series of graphics cards, printing, and others. We have moved to a more accurate assembly stage on the level of the components themselves until we got to the integrated circuits.

The same applies to the program. Languages were rudimentary and often used Interpreter rather than Compiler. A language like BASIC was running on the Disk Operating System on a black screen, forced to type, memorize, and accurately remember many commands. The programming was line by line.

A new operating system moved the Computer to new sites such as Windows and its many versions. "Smart" and visual programming languages have emerged, known as the "Drag and Click" languages. The devices and their capabilities became less and less in size, weight, and danger, greatly facilitating networking operations and lower their prices. As the capabilities develop, the advantages and innovations of companies, programmers, and users increase. We
have reached huge numbers in terms of memory, speeds, and storage sizes. Laptops appeared and evolved, then finally, mobile, and IPad became pocket computers.

We no longer need those screens that break the back due to their HEAVY weight, and the resulting radiation causes excellent health and visual damage. We no longer need to change our systems after two or three years. What we need now is to keep abreast of these tremendous developments.

We come to the compilers, and we find that they have greatly improved the level of ease of language and work from within the programming environment. However, it did not keep up with the developments that occurred with the advent of Speech Recognition and its spread in applications. We have never heard of a speaking language before. There are simple attempts that do not find adequate attention by applying SRS in some items that can be used and built upon. There are attempts by some amateurs to run the programs available in their machines by sound. As if you say a word, then the program that needed start-up. This applies to other applications. You can say Java and turn on the interface for the Java language. Language menus can also be controlled. You can also run the Editor to convert Speech to Transcript. There is indeed a problematic ambiguity, and we need to intervene with text writing. All this will be useful to some extent for us in our unprecedented project, which I expect will make a significant leap in programming. Build a voice compiler that receives verbal commands and transmits them to the application.

For these reasons, we did not find a single reference to serve us in this matter. All that is available until this moment is what we have published by Professor Kabushka and myself. We resorted to some unofficial programming sites, but we did not find anything similar.
Nevertheless, useful applications like the one we have mentioned can be developed and used in one or two-stage work, but not all of the work and their owners are usually non-professionals. I love these high-end amateurs, although they are not professional, and I appreciate their efforts. We do not find any book that talks about building a Verbal Compiler. There are no resources available that deal with audio managed programming language such as language for programming sounds. There is a shortage of audio libraries and no publications on them. Therefore, we encountered difficulties, and we are continuing to find solutions to them. I did what I designed in the first paper (1). We programmed a model that stores variables in text files. This model has Ambiguity, and if it works, it is prolonged. We improve our work to include the use of buffers. At this time, the Ambiguity rarely happens, and it has a faster response. The most difficulty we faced was building audio libraries that worked from inside the Compiler—in other words, internal programming libraries for sound in the Compiler. Even internal and external audio libraries that support some languages are task-limited. I believe it will be completed very soon, even though it is not required for my dissertation. Also, the compiler inputs were designed, and we stopped at the stage of producing an audio library.

So absolutely, we can confidently say that there is no prior work to this attempt. There is much work in SRAs, but not a speaking compiler. I have not seen all the work that my colleagues have done. This was because I joined a little late in SR teamwork. Undoubtedly, there are essential and milestone developments made by other colleagues. There are also tremendous developments in this field in general. My advisor is one of the inventors in this field. These developments can be observed in public life. In the laboratories of the SR, I saw some developments made by colleagues. Because some of them chose to make remarkable developments in areas preceded by others, they could learn about previous works. It
was an excellent opportunity for sure. It is exciting and stimulating to learn about other people's experiences. There are many development opportunities then. Good ideas sometimes come from one of these follow-ups. Later development can be made based on it. My Boss motivated me to find a new idea that I worked on. He sent me links to his researches and courses that he teaches related to the topic. I read this material.

I have turned to colleagues' research on the subject. Finally, I read a few papers from outside the university. Navigating and searching the Internet was also important. The searches led me to a site for localization models for spoken languages. I found that Arabic had no Accentric Model like some other languages. I tried to do a simple model as an example. I tried to design a mobile program so that the Calls keyboard works with Arabic orders. I submitted a proposal to use, but I understood that there is a deficiency or a problem. Then I suggested a compiler take orders orally. At first, I thought a little previous experience in this field would help me. I wanted to employ the potential of the SR in designing programming languages. We searched the Internet and found no research, study, or book on this topic. I started from scratch and thought that if I came up with a simple compiler as a template calculator, that would be a good start.

The beginning of the work was in C++. Compiler was designed to deliver audio input, which was hard. I did not find enough support in C++. Therefore, I switched to Java and previously worked in both languages. Audio storage is implemented as an audio file but transcribed into complex text, making text files connect very slowly. The best idea is to use buffers' techniques for "virtual" or physical storage. I had preferred it from the beginning. It was that I talked about it and the files in the first article paper I submitted. I also presented both options in the proposal defense more than a year ago.
I had previously worked on HTML to design a website for our company that did not meet with acceptance, so they contracted with a house of expertise. They took advantage of my previous designs, and some of them were incorporated into the final design. In 2015 I presented a different project in the Mathematical Methods course. The project was an essential modification in one of the methods to get a new numerical method—the professor required to program a Graph Model for it. At the time, I could only do algorithms because it was ten weeks for Semester, and I had eight units. After the Semester's end in the break week, I promised to complete the program and present it. I finished and submitted the program. The C program was not taken into account as they had a specific time to deliver the results, and the material was 77% (C+). I had created an account using the university email on the GitHub website. I uploaded that program in it. Also, I could not find an HTML calculator program at the time. It was a challenge to design a simple calculator program in this language. There is no audio introduced. At the time, I did not know about SR at all. Overall all these experiences have been beneficial to me. I would rule out programming this topic in HTML because my knowledge of it is not structured-language. Currently, HTML is structured after adding commands to it that reinforce this development.

I offered this to show that experience is also beneficial for innovation and development in addition to knowledge. What should we have done in the absence of previous references or work? We had not to give up and not stop.
II. My contributions:

When I listed some of my previous personal experiences, this was due to the lack of previous work or research in this field. I will continue to show some of my work, most of which are incomplete, despite exceeding 75% of the achievement. The stopping was for reasons that I will discuss, each in his narration—initially designed a Smartphone keyboard. I accessed a site specialized in the Acoustic model, tried to create an Arabic one. I started downloading the required languages and browsers for that work. I created my libraries and a small dictionary containing the Arabic words that would possibly be used in this model. Of course, the words were divided into phonetics. It can be used in the C++ program to pronounce Arabic words after I decided to record the pronunciation by more than one speaker. I paused here since I was running parallel to the verbal compiler.

![Figure 13: Arabic Dictionary phonotics](image)

Figure 13: Arabic Dictionary phonetics

![Figure 14: Grammar](image)

Figure 14: Grammar
A group of experiences on our project

First, choose C++ to do the project, as it is a rich language that provides many advantages. In parallel, it can translate its commands into assembly language. I am sure it has voice commands, but I could not find them. Finally, through research and investigation, I found external voice libraries that support them. This is certainly indicative of the failure of the language in programming sound compared to other languages. We figured out how to program a class in which the audio is captured and transcribed into an audio file. We stopped for several reasons:

Firstly, programming is inflexible and restricted to work on your computer, which you have set up in a particular way to work on such libraries.

Secondly, we encountered problems converting audio files into transcripts, although this is available in many other programming languages. Moreover, several text files are needed to store the selected words.

Third: I am sure that the solution to these problems may be in one line of code, but as long as I do not reach it, I still have an unresolved problem. This is due to the poverty of the sources.

Fourth: I decided to program this part in any language that allows this type of programming. Then we perform it as an executable file and import it for C++. Although this work consumes resources and time, it was an available solution in the absence of another solution.

This program requires much effort for the programmer and the device. It is a little far from what we would like. Although much of the work was done on it, in my view, it remained unsatisfactory to me. A programmer advised me to switch to Java, and I tried it later.

One of the main problems with this program is its very high Ambiguity. This problem also requires solutions. I found it difficult to continue using external libraries, which must determine
their program's path, which is a problem. Furthermore, in the presence of Ambiguity problems and storing files, it is suggested that it is better to stop and change the programming language.

In conjunction with this job, I was learning to program compilers. I designed a small compiler performs a program

All stages. In it, I defined reserved words as a token. I tried this program by reading the contents of the file, and it succeeded. It did a scan and showed it to me. I could build an audio library in

![Figure 15: A part of the C++ main program to capture the voice using the SMFL library.](image)
which to input and identify sounds. That was a nightmare for me. I stopped at this stage because there are no resources related to the subject.

Figure 16: A sample Code to design a compiler, not included the audio part.
```cpp
// MathLibrary.h
#pragma once

namespace MathLibrary
{
    class Arithmetic
    {
    public:
        // Returns a + b
        static double Add(double a, double b);

        // Returns a - b
        static double Subtract(double a, double b);

        // Returns a * b
        static double Multiply(double a, double b);

        // Returns a / b
        static double Divide(double a, double b);
    };
}
```

*Figure 18a: A sample piece of code taken from Allen I. Holub, COMPILER DESIGN IN C.*
```cpp
#include "MathLibrary.h"

namespace MathLibrary
{
  double Arithmetic::Add(double a, double b)
  {
    return a + b;
  }

  double Arithmetic::Subtract(double a, double b)
  {
    return a - b;
  }

  double Arithmetic::Multiply(double a, double b)
  {
    return a * b;
  }

  double Arithmetic::Divide(double a, double b)
  {
    return a / b;
  }
}
```

*Figure 18b: A sample piece of code taken from Allen I. Holub, COMPILER DESIGN IN C.*

```c
#define EXPORT __declspec(dllexport)

EXPORT void message (void);
```

*Figure 19a: A sample code of DLL according to Allen I. Holub, COMPILER DESIGN IN C.*
I thought about the languages related to Internet programming, thinking that audio libraries should be available in them. Although I do not have a good relationship with internet programming, I tried through the HTML language and found many audio libraries.

I started building a calculator program with basic operations, and an audio library was inserted into it. The program turned out to be very fast, easy to implement, and the rate of Ambiguity is low. I saw that I develop the program to give an audio output. There are audio libraries that do this job, but they conflict with the library I used to the point of contradiction so that the audio input and the (talking) result are lost. I tried to add scientific math operations that require

---

test_dll.c

```c
#include <stdio.h>
#include "test_dll.h"

EXPORT void message(){
    printf("Hello World");
}
```

*Figure 19b: A sample code of DLL according to Allen I. Holub, COMPILER DESIGN IN C.*

main.c

```c
#include "test_dll.h"

int main(){
    message();
    return 0;
}
```

*Figure 19c: A sample code of DLL according to Allen I. Holub, COMPILER DESIGN IN C.*
orientation to functions that you implement. We succeeded when entering by writing, and we did not succeed when entering by voice, which is the most important thing for me.

Someone may ask, Why I call what I did Compiler, which is regular programming that I previously criticized?

We repeated several times that we do not need to find more amid a language crowd. All we need is a change in the first character of the Compiler so we can do the PayPass over the text input and pass to the voice input.

If we do this on any app, it will surely apply to the rest of the apps. If we succeed in acting as an audio library inside the Compiler, we will have reached the end of this project. These simple commands are the modification of the first stage of the Compiler to make it work as we want it to. So it is a new compiler that has been modified from an earlier version. Since commercial companies produce the compilers, they do not provide Open Source for it.
1.3.2. Speech Recognition

Background

In the 1970s, ideas emerged about the possibility of talking to computers. The achievements at the beginning were not fast, but with the Windows operating system's development, capabilities were added to deal with sound, and from here was the actual start of this technology. In addition to that, the tremendous developments in hardware had a massive role in the spread of these applications. The occurrence of Smartphones makes the use of speech recognition widespread. Talking with a machine is not the same as talking to a person, of course. The machine only hears the words, and it must be known and distinguished. The pronunciation may be confused. Nevertheless, talking with a person completely differs. There is direct communication, knowledge, Intelligence in discerning what has been said, signs, eyes, and everything that facilitates communication.

Let us go simple without any complications in trying to understand speech recognition. We need to understand the mechanism that humans follow when speaking first, and this topic leads us to understand anatomy—specifically, the human head and what has to do with the exits of words. We will not discuss this topic here, as it is another topic that will move us away from our main topic.

However, the start is the generation of the voice. Understanding the mechanism, whatever the human organs involved in, such as lungs, vocal cords, etc., is very important for developing systems for distinguishing speech. Based on the sound source within the human being, algorithms can be created to distinguish speech. Physiology is used in this aspect.
After people became the languages through which they communicate, speech became the most important means of communication. This was in the past, and only the emergence of a new type of human-machine communication has changed.

However, the mechanisms for dealing with this new matter differed significantly. No longer is just what we talked about in a hurry about identifying the source of a sound. Mechanisms had to be found and translated into algorithms to capture the sound and try to distinguish it.

Recognizing speech by a machine is not the same as recognizing speech by a recipient of speech. The most significant thing is the man, and in his creation, there are miracles, but this man also performs a miracle by making a machine understand his words. Speech recognition is the most challenging priority in these stages of the arrangement. It needs to employ electrical pulses on and off and adapt them to receive sound pulses that the machine does not understand. Perhaps a person, with his senses and knowledge that helps him understand, is difficult to understand.

![Diagram of Communication via Spoken Languages](Image)

*Figure 20: The most common-used graph that is expressing on Communication via Spoken Languages.*
At first glance, according to the figure above, we find that there are two types of speech recognition, which are speech to text and text to speech. This is straightforward, and we have talked about a third type, which is Speech to Speech, but through a written medium. Currently, it is possible to capture the sound and store it in Buffer and use it later in all cases.

**Biometrics**

There are many biometric types, including fingerprint, eye print, face recognition, and voice, which is what we are interested in.

In addition to this, there is many other biometrics.

All these properties are physical characteristics that differ from one person to another. All of these characteristics can be used to verify a person's personality.

When we speak, there is information that is obtained. The most important to us are two parts. Identify the identity of the speaker and the content of the speech.

So the information that matters to us in distinguishing speech is the content of speech. There are more complex procedures through which lexical information is known, exploited, and employed in general.

Artificial Intelligence, in which there is a branch of Machine Learning, has a critical application called Pattern Recognition.

Through its use, pattern recognition enables us to learn patterns and how the sound is used. We mean here by the pattern one of the types of biometric or the speech signal itself.
Identification of the speaker is vital in certain situations, such as applications requiring confidentiality and reliability. In most of the Spanish Reconstruction applications, there is no need to know the identification. This helps distinguish speech. For example, the word's spelling from another person is different according to their natural characteristics. It is possible to take advantage of speakers' redundancy in some systems to reduce the error rate.

Some applications require a high degree of confidentiality. Such applications find it necessary to verify the identity of the speaker. This happens when compared to the databases of speakers. It is based on this comparison that the speaker is accepted or rejected.

Several techniques and methods are used to match speech. This modeling or matching is to recognizing the speaking or the speaker.

When we use the microphone to speak, it is an analog signal. We have to digitize everything analog so that we can deal with it later.

There are spaces between speech as a result of silence. When capturing sound, it must be checked that all of these voids that represent silence are removed. This is very important in later audio processing. Sometimes whoever said is more important than what was said.

https://www.biometricsinstitute.org/what-is-biometrics/types-of-biometrics/

Retrieved at 2:10 @ November 3, 2020.
1.3.3. Novelty

In this dissertation, what is new is speech recognition to present a language that receives orders orally. Never before in the computer programming languages branch has anyone mentioned this update. This innovation can be a turning point in programming. We proved here the seriousness of what we presented through a simple example, a calculator. This can be built upon in building more comprehensive compilers.
PART TWO

Chapter One: Design and implementation of a compiler includes a language supported by speech recognition in general
Figure 21: PART THREE WORK IMPLEMENTATION
2.1. 1. Design a new language

I. Introduction

“Programming languages produce other programming languages.”

In the late 1970s, a significant development took place in speech recognition systems (SRS), a vast improvement but still not enough to build a complete system that deals with SR. The improvements in this field have been continuously occurring, whereby software instead of hardware programming is utilized. In the second decade of the 21st century, SR is used everywhere. The SRs are easily programmed in C++ language in the Visual Studio group by Click & Drag through a box that acts as a voice recorder to save the data in any audio file format. Converting the text file to a voice file in the type of format needed is simple. In some Windows versions, recording voices using the speaker icon is found on the toolbar. Also, I can easily read text files as audio. Generally, most applications, whether audio or otherwise, have characteristics of converting files from a format to another format that can utilize applications like Excel, pdf, and word. Furthermore, software packages that perform a specific task, such as Laser and Photonics, can convert simulated files to data files in the form of images and vice versa to be used later as input either in the same or another software. An example is converting file extensions between Optiwave and Optic Studio.
As of today, the word processor accepts input in audio form in many languages. Numerous software packages prepare, write, pronounce, or convert what was read to a text file, making communication between people who cannot interact linguistically together much more relaxed. Suppose we try to translate from one language to another. Such software is still mostly incapable of translation even with languages of the same family. We are also faced with the problem of Ambiguity.

Figure 22: Recursive means that the Compiler is any high-level language that can program a set.
II. Why and how do we design Compilers in the middle of the significant congestion between programming language interpreters & compilers?

"Using these systems, our breaths become counted."

I have not heard or used a programming language in which the commands are uttered. Software to search for something or somewhere receives orders by voice. Automated voice systems control many activities, including schedules or cancelations and voice payment systems that give access to the software that monitors the calls by recording them. Numerous other reasons include quality assurance. SR usage represents examples of systems programmed by a specific language.

Compilers are similar to any other executive program that ends with the extension .com, or .exe gives programmers the environment to design and provide all programming needs based on the program’s purpose. Compilers orders and divides the files into packages (units) and performance-related tasks. Inside each is classes containing procedures, functions (methods), and all types of compiler stages (lexical, syntax, and semantic), and in order, they may express implementation types like defining variables, records, files, arrays, and libraries, and all kinds of saved files. In other words, the Compiler is also a program that uses a high-level source language to transform into a low-level target language. While running on its environment, the source compiler should detect errors, report them, and inform the programmer to make required corrections.

Second: The lack of SR applications in a language such as Arabic leads to the need to find solutions and alternatives, either by producing a spoken programming language or developing the few available applications. In this paper, all emphasis is placed on creating a spoken language by designing a compiler that achieves this purpose.
The main program, which stays resident in the memory, should have a minimum size, and be limited to few commands that apply the Dynamic Loading Technique, which calls the target program, bring it from its physical address to store temporarily in the main memory, performs its task, and returns to its location outside the memory. The commands that use this operating system technique should be available and implemented in the language used. Once the Compiler runs, it shows, edit, save and modify commands on the main screen. A user-friendly screen with menus helps programmers to perform tasks. Visual languages are preferable.

As discussed before, the SR files are used as input/output files or voice/text files, and the conversion between them. Most languages serve the call of such files either way and even the text, if it is not lengthy, can be edited in the instructor itself if programmers decide to write manuscripts to be read as voice files.
2.1.2. Frog, Amphibious, Language

“All that we are seeking is to command the computer system acoustically to obey.”

A New Generation of Language Programming: A programming language that receives voice commands based on the mechanisms of SR.

I. System Definition

Without detailing the nature of sound production, technical details of the relationship between humans and computers in SR are discussed. Mathematical calculations are vital to solving problems. The figure provided below demonstrates the process of how speech should be utilized. Designing a compiler to deal directly with the SRSs is vital, mainly due to the advancement of the Modern Generation of Programming Languages (MGPL) that support voice commands as an alternative to writing. Besides, such a system develops and simplifies programming languages so that words can be used from programmers' spoken language. Moreover, SR increases the speed of achievement, avoiding the incompatibility between the Speech Recognition Techniques (SRT) and the compilers. In many standard components, exchanging procedures may be needed for SR and Compiler Construction (CC). Instead of adapting the two systems to work in one environment, the standard framework would be appropriate for work together from the scratching with complete compatibility.
Figure 23: Basic Operation of the Speech Recognition System.
Figure 24: Compiler Construction Operation.
This verbal Compiler’s design phase has been broken down into six stages called processors to outline the subject in general and the programming processes specifically, as seen in Fig. 8, shown above.
**Stage 0.0. Programming from scratch**

A programming language is a compiler or interpreter that contains many rules and procedures, libraries, and auxiliary operations to control the Compiler. Yet, programmers can only identify problems and then program them. Our Compiler's innovation in stage 0 is unlike programming languages in the past that received commands through keyboards, mice, and input tools in general. The resulting programming language enables the programmer to design and implement systems by voice commands. Stage 0 requires constructing a complete Compiler entirely. Design a new compiler from scratch was one of the difficulties we faced. Implementing a compiler may take five years if only one programmer achieves it. We are not interested in developing a new programming language, but we need a modification to include sounds in the language for it to become more efficient.

*Figure 26: For example, the part (processor) of an audio compiler could be broken down into two processors.*
The dilemma is that sound-related programming mostly depends on capturing sounds to save it temporarily in the Buffers. Two significant problems must be confronted: A severe shortage of books that explain such types of programs and the limitation of the material on the Internet. Luckily, some fragmented material is provided by YouTubers who are interested in some applications. Buffers are audio streams stored for a while. A variable type of flowing data in the Buffer cannot be specified. Programmers must find a way to convert the buffers’ contents into variables of the String type. This method only enables programmers to compare the variables defined in the programs with data in the Buffers. Programmers always have alternative solutions, such as using text files. A Compiler acts as a calculator designed: The desired process is verbally commanded and stored in the text file A as a string or an integer variable. The numbers that are summed up through a regular program store in file B. Comparing the entered data in the text file “A” with the expected cases and matching with one of them is a required operation, and here is an example of an “addition” of two numbers. Different operations can be programmed, but in this case, the issue is the sophistication and need for more text files. The previous explanation is a solution but is stressful for programmers and takes time to process.
Figure 27: More explanation about regular Compiler's steps.
II. Internal and external libraries

One of the programming languages' essential features and advantages is the various libraries located within the language or programmed by the Compiler, and they are called internal libraries. Simultaneously, there are additional libraries that serve specific purposes that are not available in the programming language. Such libraries do not benefit all programmers in general. These libraries are called external libraries, and small software companies program them.

External libraries are downloaded from professional sites. Several steps are followed to install them. Internal libraries are not commonly available in languages but are added from within the language interface. The language mainly has the necessary libraries to run in general, but the performance of some additional operations, such as the use of sound commands, needs to be added.

*Figure 28: The internal and external libraries.*
Figure 29: An example of an Internal Library; C#.
Figure 30: SFML, One of the external library that serves C++. Retrieved from: SFML (sfml-dev.org) on 11/30/2020 at 18:57

Figure 31a: SMFL, an External Library. Retrieved from: Download (SFML) (sfml-dev.org) on 11/30/2020 at 19:11
Download SFML 2.5.1

On Windows, choosing 32 or 64-bit libraries should be based on which platform you want to compile for, not which OS you have. Indeed, you can perfectly compile and run a 32-bit program on a 64-bit Windows. So you’ll most likely want to target 32-bit platforms to have the largest possible audience. Choose 64-bit packages only if you have good reasons.

The compiler versions have to match 100%!

Here are links to the specific MinGW compiler versions used to build the provided packages:

- TDM 5.1.0 (32-bit), MinGW builds 7.3.10 (32-bit), MinGW builds 7.3.0 (64-bit)

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Visual C++</strong></td>
<td><strong>Visual C++</strong></td>
<td><strong>Visual C++</strong></td>
<td><strong>Visual C++</strong></td>
</tr>
<tr>
<td>12 (2013) - 32-bit</td>
<td>12 (2013) - 64-bit</td>
<td>5.1.0 TDM (SLLJ) - Code::Blocks - 32-bit</td>
<td>5.1.0 TDM (SLLJ) - Code::Blocks - 64-bit</td>
</tr>
<tr>
<td>GCC 7.3.0 MinGW (Dw2) - 32-bit</td>
<td>GCC 7.3.0 MinGW (SEH) - 64-bit</td>
<td>GCC 7.3.0 MinGW (Dw2) - 32-bit</td>
<td>GCC 7.3.0 MinGW (SEH) - 64-bit</td>
</tr>
</tbody>
</table>

On Linux, if you have a 64-bit OS then you have the 64-bit tools chain installed by default. Compiling for 32-bit is possible, but you have to install specific packages and/or use specific compiler options to do so. So downloading the 64-bit libraries is the easiest solution if you’re on a 64-bit Linux. If you require a 32-bit build of SFML you’ll have to build it yourself.

It’s recommended to use the SFML version from your package manager (if recent enough) or build from source to prevent incompatibilities.

<table>
<thead>
<tr>
<th>Linux</th>
<th>GCC - 64-bit</th>
<th>Download</th>
<th>2.31 MB</th>
</tr>
</thead>
</table>

Figure 31b: 1. Fig. 29, 30, and 31 clarifies how to download external library. Retrieved from: SFML 2.5.1 (SFML / Download) (sfml-dev.org). On 11/30/2020 at 19:27
Stage: 0.1.3. Error handling

In every language, when errors occur in one of the compiling stages, the number, type, and line of the errors are shown. In this modified language, errors are classified and given numbers and displayed in interactive screens with programmers.

Figure 32: Error handling Menus. Prepared by author.
PART TWO

Chapter Two: Case Study
2.2. 1. Case Study

In this case study, we analyze seven inputs and outputs of the six stages of translation for regular Compiler, that begin from the input and are translated by a compiler stage. As shown in Fig. 14.a., the statement is placed on a symbol table. Once the voice is captured, it is dealt with precisely as in the compiler stage. So, the input will become audio. In Fig. 14. b., the voice passes on the Syntax Analyzer that composes the tree if the capturing operation of the voice is correct; otherwise, an error occurs.

![Diagram showing the stages of translation in a compiler](image)

*Figure 33a. Prepared by author.*
Figure 33b. Prepared by author.
Figure 34a. Prepared by author.
Figure 34b. Prepared by author.
Figure 3

Case Study of audio Compiler(5)

```
t1=id3*17.0
Id1=id2+t1
```

Code Generator

```
LDF R2, Id3
MULF R2, R2, 17.0
LDF R1, Id2
ADDF R1, R1, R2
STF Id1, R1
```

Assembly Code

Figure 34 c. Prepared by author.
Fig. 33, 34 clarify all the compiler construction stages. Fig 34 a, b, and c show the other four steps to reach to Machine Code (Assembly Code), ultimately. This code deals with the contents of Microprocessors, Registers, and RAM and has access to other external resources such as memory. The example model of the Compiler that works as a calculator receives audio commands and then works as a regular compiler, information that will be explained in detail later.

![Diagram](image)

*Figure 35: An example of how to capture a voice and use it as a command. Prepared by author.*
The compiler model works as a calculator by receiving the instruction verbally.

To not confuse the users in a new language, all C++ symbols have been used. For model programming, several options are taken into consideration. Although C++ is complicated and there are no examples of all commands that deal with voice, I was chosen. The most crucial advantage of C++ is that it creates a code representing assembly language and has the required speed in dealing with the CPU. The most vital operating systems are programmed in C++. In Fig. 15, the graph is considered Stage Three, the third processor of the whole system, designed to implement its operation into two subdivision Stages (Processors). This Phase is the first stage after applying all the stages of the regular Compiler. Audio programming allows many ways to capture the voice — categories placed under two columns: First, save the captured sound into any audio file format to deal with later or reuse the audio saved for other purposes. Second, save the voice in a temporary Buffers for a while. Both methods are interchangeably used as needed.
I. Saving the voice in an audio file

An external library called SFML can be downloaded, and several procedures become executed. For instance, we downloaded some libraries from SMFL and stored them in the same folder where the language libraries are stored. The program will not work on another machine unless the same folders are stored in the same places and the same libraries have been transferred to these folders. Compile the file as an executed file to ensure that the program works correctly. Then the executed program is run in the objected file. Programmers are programming, and moving between different computers encounter complications that occur because of the non-presence of external libraries in their folders. When we finish the previous procedure, using voice commands can record any audio file at any length. Audio files are useful in that they can record clarifications and introductions to show results and help menus.

Furthermore, they declare system errors. One or two words can be recorded, transcribed, and stored in a text file to be used in the system. These text files can act as a database of the system while the column represents the “field,” and the line represents the “record.” The programmer can package classes to read data, and when the program runs, it deals with the DBs or text files. For example, if the “Addition” process is used, users command verbally to choose the “Add” procedure; the order is interpreted to be a variable such as an integer, a bit, or a string, and stored in an appropriate file. Based on the previous procedure, other functions are programmed that can claim variables to be saved in other text files, and recording audio files and playing them later in the proper program or typing the audio text inside the program to read can also be achieved.

Adding two numbers, users give the command that reflects the summation and declare how many numbers to be added. In this case, two files are used, and if the operation is complicated,
which means using more than a process like “Add” and “Square root,” more than two files are used. These operations are expressed within the codes that classify them and direct each process to their procedure. The verbal interaction between users and machines continues until the calculation is finished, and the result is given in the form of reading the text stored or written, and then the number shown is read. The machine alerts users that the process is completed. Users must obtain the results in style written primarily at valuable systems such as employee salary calculations. The input is done in the verbal form and under the control of the user. If the input is ambiguous, the use repeats to write what the command wants clearly. Ultimately, the input is stored in the database or text files. However, when calculating the results, they should be shown in reports and the paychecks.
II. Using the Buffers’ Technique

A buffer is a place for storing data. It has several uses, including memory storage, data transmission, and video and audio storage during streaming. Storage is usually for a limited time in RAM for acceleration. In audio programming, the stream is stored in Buffer without defining it as a variable, and this requires its definition later to be stored permanently. It is also known as a cash memory and most of its use in Internet and Internetworking.

Retrieved at 15:21 on Oct 31, 2020, from: [https://www.pcmag.com/encyclopedia/term/ ]

Inside such libraries, whether internal or external, there are instructions for buffering. The sound can be captured to the Buffer as a stream and then stored as any variable. Using buffers is much more comfortable, faster, and vital. One issue that requires attention is that the Buffer's sound does not become defined as a variable. Other commands address this problem by converting the sound into a String type variable. The sound in this format is very easily handled. The technique, identical to the previous stage's steps, has greater flexibility, faster speed, and no exit from the CPU to connect to files stored in other locations.
III. Ambiguity

Whatever the language spoke, recognizing the speech depends on listening to individual words taken from the context and watching the lips' movement and body language. The speaker uses his knowledge and logical repository to associate sentences and distinguish words individually or in a related context. Moreover, even if unable to understand a vocabulary used in a sentence can expect it and know the meaning. The understanding is when the full context of the sentence is taken. Furthermore, the use of compound words, dialects, and idioms should be taken into consideration. When a computer or mobile phone is used to Recognize the Speech (RS), none of those mentioned earlier can help the devices. The intelligence of computers is zero, but many people think that a computer is a smart machine. Programmers must define everything each step in a logical form acceptable to the Computer as a whole, and programming languages as a part can be followed as illustrated in our model, a verbal calculator, steps through which an integrated work can be created to deal with such an acoustic signal. Besides, noise and many different factors affect the quality of speech recognition techniques. Some of them are related to the Computer itself, like the sounds it makes. External factors in the room where we work are included, such as background sounds or noise.
Figure 36: The Ambiguity problem. Prepared by author.
IV. How to Solve Ambiguity

Many solutions can help users to specify the desired word. Programmers design a window in the program that gives matching options to the word. For example, if the word *two* is wanted, sometimes words *2, too, and tool* appears, but users can manually correct the ambiguity by choosing the needed word. The same thing with the word *four* can be written as 4, for, ford, forth, or four, but this operation leads to minor issues that arise delay. Another solution that gives more accurate results but is difficult to be applied. Always the English language dictionary depends on two pronunciations for each word or at most four. A lady’s and a man’s voice with slow and fast speech can be heard. Even though speakers record different pronunciations for the same word, composing a whole new dictionary is too hard. Instead, symbols that we use in the language and the accompanying dictionary can be employed. It can also be facilitated by some kinds of dialogue, like when the dictionary asks the user if a word is correct. If not, the wrong word is excluded while the user repeats pronouncing the word.
PART TWO

Chapter Three

Constructing A verbal Compiler
2.3.1. Constructing A verbal Compiler

I. Introduction

"We're not gonna reinvent what has already been invented!"

What is meant here is that the Compiler has been found since the invention of the computer. Its programming phases are known and specific, as well as compiler programming. So it would be kind of pointless if we were to start doing Compiler from scratch. Moreover, that takes a long time. This time should be directed to what is most beneficial. Compiler programming is useless as it will not be of the same quality as available. Also, our topic is not to program a compiler. Our topic is to insert voice so that the keyboard accepts verbal commands, which is a big deal. Any structured programming language can be taken. Our challenge is to maintain all stages and be adequate with modification only in the first stage. The programmer writes commands using any language that implements voice commands. In theory, this is easy, but applying it in some languages is tricky. Several reasons make it difficult, including the lack of references for audio programming languages. In terms of hardware, all audio sources are available. Operating systems, specifically Windows, support audio greatly. Here, we return to the race between hardware and software, so we do not see high-level programmers in audio compiler applications. However, some have programmed external audio libraries that support some languages. There is an urgent need to document the operation of audio programs in the language help details. In general, the programmer has to do the trick -In programming- to reach his goal. As they say, “All roads lead to Rome,” several techniques can be used. For example, I found it challenging to deal with audio in C++ in our project. There are solutions on the table. The sound is captured, and this is available in C++ by adding external libraries. It converts the audio into a transcript and
stores it in text files. We take whatever words we want and store them for use in system files.

The easiest of these turn around is the storage at Buffers. They are used directly as in a calculator Model or store sounds in files for later use. To store Buffer contents that are not defined as a variable, they must be defined as a variable, which requires action in the language. Another method I do not suggest is to use the easiest way to capture and deal with voice. It is converting commands to an application file that can be used in another Programming Language. A Calculator Programming Model, which changes the first stage, is a small program that does not exceed fifty lines. To write a code controlling the voice that takes very few. Of course, working with languages will require tremendous effort, but it can be accomplished. Once we got this calculator, we already have a compiler. We have found a way for the Compiler to receive our verbal commands.

In 1994, I studied operating systems course. Windows was not included at that time. We studied a section on dynamic loading of files into memory. When you read a large group of pages, you may think that the topic is hard to implement. We worked on a graduation project whose file went beyond five Mega when the four megabytes of memory were devices that only oil companies owned. We remembered Dynamic Loading, so we studied it and read a book on Turbo Pascal. We found that only two sentences perform this “Sophisticated” operation. However, before that, we had to reconsider our programming strategy. We benefited from most of what we wrote. We just changed the system to twenty-two units, which are libraries. We compressed them into twenty-two executable files ending with .com extensions. The main file was no more than ten lines with an infinite loop that kept us in the system until we decided to run out. It also contains the Dynamic Loading command.
In this stage, all the system is divided into phases. Some questions must be practically answered through preparing programs to execute the tasks that the Compiler performs. The first question that should be asked is what this Compiler looks like? In terms of programming, the question means designing the Compiler's main Menu and the other menus. Furthermore, the answer declares why it is called a compiler, not an application. Another question, for example, is how this system works. The solution gives details about how to merge the voice and text to run the procedures and classes. The third question is why it is a new compiler? Is it needed in the presence of this crowding of programming languages that it would not be better than them? Why confuses the programmer? The practical answer to this question will be through programming commands, from which the used language is an old language with its known commands to any programmer. It is possible to change the writing of language commands with new commands, but this will not benefit the programmer and cost him time and effort to learn the new language. Despite the scarcity of available audio programming resources, either in the language itself or in external libraries, only voice commands have been combined with language commands. Avoiding Ambiguity might lead to errors. The voice commands were restricted to specific words to be distinguished. These words are included in any language by adding them either by text file or by defining them in the Compiler's main program.
II. The difference between the Verbal Compiler and the app is a lock of hair!

In a previous paper [1], there is no need to design a new compiler but use one of the available languages while integrating the audio libraries.

Here, a simple example is a simple calculator, designed and programmed to receive verbal instructions and execute them, as will be explained later.

Simple Calculator, with basic operations, is not a comprehensive example, and only some limited processes are exposed through which an attempt is made to review some essential issues in the design of compilers.

For example, it displays error and error type when a basic calculation, such as division, crashes. Such an error is dealt with in the basic definitions or libraries defined for each operation. Yet, the Error is seen, and a letter or a voice message is given.

In much larger applications such as databases, databases can be set up as tables. These tables are loaded into records and fields, and each table has relationships with other tables. These properties are defined by voice commands and are also filled with data. While dealing with these databases, of course, many errors occur that need to be reviewed later. Here we find that there is a need to store voice commands to enter data. These files will then assist in modifications and cancellations. Indeed, storage is not a sound, but the voice command is transformed to its written counterpart and stored in text files for programming. In the case of a Simple Calculator, the storage feature is not used, although it is considered. This character's non-use is because there is no need to store a single command that can be restored phonetically. The many consecutive orders must be converted to their origin in the language used and stored in the program's writing file for later reference.
III. A Model of a simple compiler that deals with the voice as an input

Here, these figures present a simplified model for Compiler that performs mathematical operations by receiving voice commands with an error rate close to zero. The Ambiguity here is very low because we are specific to deal with numbers and operators. If there is ambiguity, we can add a definition of the word, an operator, or a number.

The programmer can choose the design according to his vision. This design was chosen because the user is used to this form of the calculator. There is no desire to confuse the user. These forms seem to perform a mathematical operation entering their data phonetically, which means that the matter is straightforward. On the surface, this simple design hides many program complications. The choice of a calculator as an example came so that we do not expand much by example, and such expansion has significant problems. There are many procedures and processes behind this example that will be discussed later.

![Figure 37: A verbal Calculator [A program Run]. Prepared by author.](image)
Figure 38: The result of multiplying two numbers [A program Run]. Prepared by author.

Figure 39: A verbal Calculator. Prepared by author.
Figure 40: An error occurred [A program Run]. Prepared by author.
### 2.3.2. How does the Calculator work?

A critical issue must be clarified at the beginning. In the paper [1], two design options were given. One of the two options is to use text files to store the input and output data and deal with them later. The most preferred choice is to capture the voice, place it in a buffer, use it directly, or it can be stored as well.

Following the second option, which deals directly with sound from within the Computer, has many advantages. First, the task can be performed very quickly without resorting to external communications. Secondly, contacting resources outside the machine, and thus outside the Computer's memory, complicates operations and increases slowness. If this connection is lost, it must be returned. Third, and most importantly, voice commands are faster to enter. In this program, numbers can be entered either number by number or group of numbers. Arithmetic operations are applied according to pronunciation. This Calculator simulates simple calculators, and many designs can be chosen. Some of them hide numbers and operations from the end-user interface. The user can make some adjustments to regulate the output status. The Calculator can only write the result, pronounce it, print it out, store it, or do it all together. This is one of the points where this calculator meets the Compiler.

Such processes, which take place in less than seconds, have high logic and processes' complexities hidden in the application.

Also, it is possible to control the time that the microphone remains open and receive orders. For example, it may be turned off after a second or two if it does not capture a sound.

Simultaneously, this short time distance can be increased in case of entering numbers and operations verbally. The Error percentage in the entry is almost non-existent unless it is a human error, such as not entering the amount that is supposed not to appear. We can control the time
taken by the calculator to display the result. Many operations can be controlled or added by the programmer. This program is designed to capture the voice; thus, we cancel the manual input throughout the input unit, which is our goal. Designing a compiler, we add a piece of code that makes all done by the audio input, and we do not need to design a Compiler from A to Z. Fig. 5 below explains how to control the time in this system. This chart gives an idea about controlling the time in more extensive systems or a verbal Compiler. Maybe a word or instruction or more, based on the language, apply this control. Many technical issues are hidden behind this instruction.

![Flowchart of verbal Calculator](chart.png)

*Figure 41: General Idea about how the verbal Calculator works [Flowchart designed by the authors]*
I. What are the processes hidden by the example?

The Compiler is a different programming language. One of the existing programming languages is used for computer management, through which mainframe systems represent the operating systems. Then there was a need to find programming languages to meet the demands of users. Companies competed to create new languages and develop the ones on the market. We have reached a point where the market has become saturated with languages capable of programming (everything). Creating a programming language requires excellent effort and consumes time and energy, and unnecessary in the presence of this crowd of programming languages and applications with macro features, making them almost “Open-source.” With the development of operating systems such that sound has become an indispensable programmatic component, many applications in the sound field have emerged, providing significant assistance to the user. Moreover, programming languages are now opened with a voice command, and voice commands execute all menu instructions. Furthermore, it was possible to write the code with the direct voice command and store what was written. We have all the supportive features and keep creating a compiler (programming language) to control A to Z by audio. This model is a simple attempt without being overwhelmed by the details of the details to invent such languages. If a programming language that deals with databases are chosen, it will be very complicated as the programming language focuses on the simulation. The Calculator of four basic operations is a good model that briefly meets most of the programming levels that we will show. I deliberately used a well-known programming language to make a (new) programming language, and I did not change anything in the programming language that I used to produce the programming language to add anything new with simple capabilities, whereas the goal is to employ sound in programming. The same libraries were used with the same (Token) of the language. An addition could be a simple modification in
the form of a series of steps for the compiler to have added audio libraries for the compiler stages in parallel.
2. 3.3. The Programming Languages from Past to Present

I. The Compiler, a programming language.

At the beginning of the invention of the Computer, it was a giant machine with limited capabilities. The hardware was not fit for anyone who wanted to use it. After several developments, the hardware became progressively irreproachable with software and operating systems. Now the hardware and software are advancing very quickly.

One of the most important uses of computers was programming languages. Simply, every programming language is a compiler. Programming languages have evolved and diversified to include all aspects. On the other hand, accompanied by significant developments in the hardware and many audio applications appeared. The emergence of these audio applications was not accompanied by programming languages that support voice commands. An attempt was made to find a programming language based on the voice command, not the textual; that what we try to do.

II. How many programming languages are there?

I got an idea to ask Google about the number of programming languages. It was revealed through the research that it ranges from 600 to 700, and some have identified it as 655 programming languages. There is no doubt that programming is imperative in our time for systems automation. Programming began with the invention of the first computer in 1941, and it was called machine language, which is a zero and one. I mean, the program is a long stream of Zeroes and Alons. It was a primitive machine that helped engineers in their work. Due to the difficulty of distinguishing it, machine language turned into an assembly language to be understandable. I remember that the most famous language was BASIC. The word basic is a well-known English word. BASIC was an acronym for words (Beginners' All-purpose Symbolic Instruction Code).
Meanwhile, the COBOL (Common Business Oriented Language), Fortran (Formula Translation), C, and later Pascal languages developed. Until we got to these long lines of languages, we only know ten or twenty of them. The BASIC language was used by an editor obtained from external commands for the black screen DOS operating system. I think she was working on Interpreter, not a compiler. She solved the problems of her time, and she was running out of even payroll systems. Now it is one of the top twenty languages in the world, Visual integrated, comprehensive language. From the '80s, all programmers would sit in front of a computer for hours and write. After the Speech Recognition techniques' emergence and development, we asked this question, which is why the programming language is not a voice command. This is our main research question. Then we set out to search for a practical answer. We wanted to design a verbal compiler. We seem to have come a long way.

III. Why do we look at a calculator as a compiler?

The source code is entered into the programming language used, and from it, we obtain the required program after a series of processes from the Compiler.

The Calculator is always a program. Here it was taken as a prototype for a simplified compiler. A user interface can be designed to make the programming easier; in this model, the Calculator user interface. In general, while creating a compiler, it should have a main menu with all the operations in other branched menus, starting from scratch when the user is asked to choose if that program is new or saved before, passing through the programming till the run.
IV. Summary

SIRI, Bixby, and Google Speech Recognition

Many audio applications depend on Speech Recognition, such as Siri, Alexa, and Google Speak Recognition. These and other applications are the use of audio capabilities in computers, smartphones, and more. In the example that we have programmed in HTML, we have never relied on these applications. Although I do not like to use the HTML language because it is difficult to maintain, I preferred it since it is mostly used in Internet applications. Internet applications are currently the most dependent on voice and audio input in general. After I had
difficulties in other attempts to see it as a temporary solution. Any problem that has not been solved remains challenging and involved in our view. When the problem is solved, it is then simple and easy. Solving programming problems may be in one or two lines of programming. Although when you read theoretically about the topic, you believe it is more complicated than you think. I previously talked about Dynamic Loading in operating systems. When we studied it theoretically, we read many pages. When we found that it was a solution to a programming problem that came our way, we were surprised that there is a definition for it in the main program and one command in Pascal. Most of what I started from programming duties, I’m going to finish them and achieved high percentages. This verbal Compiler keyboard is an idea that I believe is doable.

Pascal, that peerless French scholar of various sciences, was also an inventor. He tried to design a rudimentary calculator and did not implement the idea until later. There was a title for me: “Great Accomplishments Begin With Little Dreams.” This is a small dream, and perhaps one day, it will be implemented, and the University of Florida Institute of Technology will be a head start. "Positive" stubbornness is not a negative quality in a person. It is the insistence on reaching the goal. I was not disappointed when I saw my work copied, pasted, and presented, for example, as an invention or a doctoral thesis, even if my name did not appear in it. I take great pleasure in this and consider it a success for me. But at the same time, I do not give-away ideas and attribute every image to its owner. I believe that Science and Knowledge accumulate a lot of information and based on the above.

Appreciation: To Elyaa Issa, who painted the figures 1, 2, and 6.
PART THREE

Chapter One: Future work
Figure 43: A Pyramid prepared by author
3. 1.1. Future work

This work, which was presented throughout the previous two parts, led to the thinking of three main components of other outcomes that will accumulate.

A. This work expanded to include much larger and more comprehensive applications than presented as a model. For example, try to implement it in database applications and others.

B. Finding a new programming technique in which the higher languages are more dependent on the compiler programming model, so that we get rid of programmers' errors, whether those that stop the implementation of programs or errors that do not control the execution of what may cause a delay in it, and logically they are not errors, but they lead to problems.

C. With the Compiler programming model's help, it is possible to change the "reserved words." So that orders can be placed in Arabic, it is generally not very important to have an English interface. The most important topic is making an acoustic model for languages where voice work is not available; Arabic and other languages. As known, approximately twenty languages have an acoustic model.

I. Extending the model to include large applications

Currently, some websites encourage talented people to program and are creative. I do not know who it is really behind, but it has become a source of attraction for specialized companies to identify and attract people with good programming capabilities. On the other hand, it gives innovators and developers opportunities to review their work and demonstrate their capabilities on these sites to get job opportunities that suit their potential. When I was in
Denver, I opened an account with the email assigned to me from the university and installed two projects in it, but unfortunately, the email was lost, and thus the account. I rarely open these sites, but it may lead me to one of these sites if I search for a Google topic. Due to the lack of or say the lack of resources for voice programming, we only found these sites to provide us with assistance once from the many times we search. It is incomplete, but with the researcher's effort, he may pick up the thread head from it. YouTube is also a great social site that offers many useless things and provides useful information for science and education. Through my follow-ups, which increased somewhat after researching Speech Recognition, I found some valuable additions. Unluckily, we cannot use these materials or videos as references.

For example, I can contact Dr. Zec, Selaghi, Bernhard, or Kepuska to inquire about pure scientific information. When I get the answer, I may quote it or paraphrase it, and I consider the reply email as a reference for me and even put it among the references.

Those who provide the information we do not know what their status or qualifications are. Even the names by which they identify themselves may be borrowed. This is untimely.

Some of the attempts I quickly ran through were:

I find a translation or Transcript of a video, a prevalent topic, and the most effortless Speech To Text recognition. Now even the major broadcasters such as CNN, BBC, ABC, and others are making these translations in English to facilitate understanding for the deaf, for example, or those who cannot understand the broadcaster's accent, even the actor. This Transcript can be used to enter data for files or text editors.
Of course, audio files, although they are smaller than video files, are still gigantic compared to text files. Therefore, dealing with text files in systems processing is more comfortable, faster, and more accurate.
II. Changing the Programming Strategy

Another critical point is that Microsoft has provided tremendous assistance that supports voice applications, especially on Google. It is possible to voice control all programs and applications on your device through a simple connection process. For example, you can open Visual Studio, choose what is available in the menus, and even write programs in the way we talked about in the first paragraph.

It is challenging to visualize a programming language without a text editor. Our goal is not to cancel the text but rather to employ the voice on any topic related to our project.

In the same way, you can give a voice command to open Microsoft programs such as Access, Excel, and Word. All these massive programs can be manipulated for programming with voice inputs and voice commands as well. It can be stored as databases or inputs in general.

For example, an Access program can create structured database files. We give names to fields and input data as records, define single and compound keys, and arrange the necessary relationships. It can be entered by voice, as we explained previously. Excel files are widespread, so we may import an excel file and use it in the system, either as a database file or other types of entries. A word file in which we store simple data to act as an array, for example. Thus, we depend almost entirely on sound. Storage is in the form of text files for several considerations.

We are trying to make the whole project sound-based. For example, when writing code, I would manually and write text using traditional input methods.
In the end, just as the voice input has the advantages we mentioned earlier, it has its disadvantages.

We cannot sit down and listen to an undocumented audio report. One of the most valuable advantages of computer systems automation is that we have a written database available with access controls and saved to become essential documents that we can refer to later. We use written data for easy circulation, distribution, and access. Furthermore, it is the fastest solution if there is audio data in return.

We rely on written reports for our business, which are essential. It is easy to review and access specific information. Some reports result from individual systems that either have access to them in text form in their files or are printed and transferred to the interested party. For example, we cannot send a file contains details of numbers as a recorded file to employees that represent Pay-Check

Even if we use audio files on radio and television, we save them in the context of written files.

Compiler, after all, is a programming language that produces a programming language. It has steps and sequences that convert what we write as text into electrical impulses that machines can handle. In other words, the Compiler represents our programming language from a Low Level. The Low-level programming languages benefit because they deal directly with computer resources and components, so they are very fast as deer. High-level programming languages were developed to facilitate the programmer. So much so that they made programming very easy so that the programmer does not need more than setting the cursor and drawing, or in the usual sense, "Drug" and Click.
On the other hand, higher languages caused problems, difficulties, and slow access to hardware's required resources. The difficulty depends on how well the programmer understands the issue and whether it is smooth or fluid in translating it into algorithms. For example, Graph programmer programs and makes a mistake, so he estimates the machine's speed and capabilities. When executing the program from the speed, see nothing. He resorts to speed-blocking measures, and the simplest is making loops. The problem remains, and it loops within the rings. All this causes slowdown, poorly structured programming, and possibly a drain on resources. This is not apparent to anyone in simple applications, but it is noticed when you wait hours for the simulated result. Gigantic database systems, as well as distributed systems, may also have some delays.

Systems of copying or retrieving massive data from the cloud may be slow, regardless of the capabilities of the machines used or their speeds. Even excellent programmers may fall into this trap with the abundance of embellishments and documentation within the software that Whitespace is. Indeed, analyzing the system very well and outputting it in temporary files, quick relationships, etc., are topics that system analysts can accurately understand to help produce a sound system. All this in the presence of the easy high-level languages that the wizards do give us no guarantee of speed. I was hired in the mid-nineties to analyze and program a salary system, and I was a fresh graduate who did not understand database programming languages. I studied the old system well and met the specialists, and I understood what I did not understand from the previous system. I knew their requirements and what they wanted. I worked on the system, and the machines did not have many capabilities in speed, RAM, and storage media. There was no network, so we needed to get out what was entered in the form of reports sent to some departments and offices for review.
and auditing, and then re-entered. The process was like a nightmare. For example, in files, I found that the main file consists of more than one hundred fields. When you choose to "do the math," it takes hours. Printing And printers were primitive at the time, the least of which is the linear printer, which takes hours to print more than a thousand slip-on. We used to do the calculations when we came to work, and the process would end soon before the end of working hours, so we pressed the button to extract a detailed report and leave it and go back the second morning a day to see what happened. We are lucky if the printing work is not hampered in any way or by any problem.

Details for each employee were over thirty values. In backward countries, there are worthless details that are imposed and put into the report. For example, the basic salary of the employee or collaborator appears. Then appears deductions for absence alone and delay alone. Rewards bloom in one column and the extra in another. Besides, several types are calculated for each class as an hourly rate, either one hundred percent, one hundred fifty percent, or two hundred percent. The total salary is calculated from the above, and through it, some calculations are made. Social security is deducted, which is a fixed percentage that the employee pays part of. Then we get the remainder and use it to calculate segments in which income tax is deducted. There is another set of deductions. To reach the net, you have to read 30 fields.

What I did was better than what was found. We used to drink coffee and make some necessities, then start making calculations, extracting reports, and getting them before the end of work. However, one should not be held responsible only for the programmer for producing a slow system. Not getting to the point and insisting on packing extra details is also a reason. The programming language and the system as a whole are additional factors in the problem. This is a real example of being slow to get information.
If we pass the writing stage to the Second Stage in the Constructing compiler stages, we will save more than half the time. This bypassing can be either using the audio input or by simple modulation in the First Stage of the Compiler construction.

III. Creating an Acoustic Model of languages that do not have one!

If you design your Compiler, you can change as you like in "Reserved Words.” These words are words that the compiler reserves for use as language-specific commands. The Compiler does not allow them to be used for the purposes you have booked for. Here you can design the programming language screen in the spoken language you want to work in.” For example, Arabic. You can also change the commands to be in Arabic. This is if we want to design the regular Compilers. Nevertheless, this work is a waste of time. What does it mean for the language programmer to be in a language other than English? The programming specialist has become accustomed to working in the English language. This may confuse. I think the Japanese, the Chinese, and the French work on languages where orders are given in English.

As for audio programming, the topic is different and makes a difference. Commands can be in your language. Programmatically processing these commands in the English language, in which the words are reserved in the Compiler. However, it is first necessary to make sure that there is an Acoustic model for the language you speak. There is an official site that explains how to configure the Acoustic Model in general. By the way, it reviews the languages that own this model. I think the last time you browsed it over a year ago, there were nineteen languages. The Arabic language does not have an Acoustic model and all the efforts outside this model. If you log in to Microsoft Word and activate the Speech to Text process, the Arabic language is not present.
Initially, I wanted to design an Arabic Acoustic model with limited tasks, but I found someone working on a somewhat close job. The topic is not as complicated as the Compiler. The steps are precisely explained on that site, including all languages—the difference in the spelling of words. There are many possibilities for the letter to be pronounced with more than one sound. The letters must be cut according to the pronunciation and converted into strings of two letters for each letter to facilitate reading them from the program that scans and finds the desired letter. This is called phonetics. I first thought that the model should be a mobile device that takes voice commands. It was later agreed with Dr. Kapuska that we would work on the compiler project. For Compiler, there is no supporting material. We had to visualize and enact designs and then implement them.

The Acoustic model for languages is a business that specialized companies can conduct a feasibility study to implement or not. Nevertheless, it is a matter of fun and challenge for the researcher, although its steps are known. The challenge is to create a dictionary or to use an old dictionary for phonetics for the desired language if the dictionary exists.

Generally, this is an excellent existing project that can be researched. Moreover, in terms of importance apart from the commercial gains that will come later is vital.

IV. Simplifying the Language Style

I tried to use a writing style that had little change from the scientific patterns used. Indeed, my English remains a second language for me, at least for now, so mastering it is not the same as mastering it as a first language. It is known that only scientists and researchers are interested in scientific papers. Companies also care and have their specialists, but the top management or decision-makers may not understand what has been written. The researcher is
forced to make a presentation in which he explains in a simple way that is close to everyone's understanding. We can increase the number of readers for the research if it is closer to the reportage while preserving the scientific method whose most essential characteristics are references.
PART THREE

Chapter Two

Sum Up
3.2.1. Sum Up

Technology is continuously evolving. It is now the fastest in providing new achievements, significant developments, and great leaps. Discoveries are appearing to us periodically in succession. You cannot close your eyelid for a moment when a discovery or qualitative development may appear. In this work, which I consider a beginning, consequences will follow, which will take us to a turning point. This might be a dramatic turning point after it is entirely different from what was before it. This is in the area of the Compiler that takes orders orally. Meaning his input will be audio. A slight modification in one of the stages of building the Compiler leads us to fundamental changes that include replacing writing with the voice in giving commands. Of course, this gives the programmer as well as the user more freedom. Freedom is about freeing your hands.

Freeing eyesight from the harmful focus provided by these types of translators. As we know, sitting for a long time, causing trouble, so safety professionals are advised to stand every half hour and walk around for five minutes. Moreover, despite the screens’ availability more secure than the previous ones, the user needs to change his angle of view to other directions after focusing for five minutes. Furthermore, many tips can be avoided after breaking free from the keyboard and mouse and sitting. There are multiple methods of remedying Ambiguity’s problems, and over time better solutions will emerge. If the entry is "clean," then this type of entry will be high-speed. The same applies to applications as well. An application can be programmed to have 100 percent audio input. We go to the embedded systems and find that the matter can be applied to them. The control panels in large factories may have noise next to them, but this problem can be solved by placing the Compiler or the application in a soundproof room. The topic reaches robots; instead of programming
Microprocessors as an embedded system, the topic can be expanded. The system built into it could be a talking compiler. The designs for this Compiler are extensively presented here. We also carried out limited operations to validate it. We can build on what we have provided.
3.3.1. Recommendations

One: We recommend that this topic be presented as a patent to preserve the university's rights and the authors.

Two: This topic needs to be promoted after confirming its ownership and expanding its work, including other parties that may want to participate.

Three: A group of researchers and programmers could be assigned to develop this work. We can provide other research that is an extension of this work.

Four: Tests were conducted, for example, trying to create an audio library inside the Compiler. We found two difficulties that not all Google audio capabilities are qualified to work from within the Compiler. Also, we have the Compiler as a compressed file, not, for example, Open Source. Because the Compiler is not like a Linux operating system, it can be scaled. It is subject only to the development of the manufacturer. So despite attempts to move away from manual entry, we are forced to add pieces of code using High-Level Language. Another way is to make our Compilers from A to Z. This process is a waste of time because all we need to do is make changes to the first stage of the Compiler.

Five: From a scientific point of view, I have suggested two solutions for storing data. One of the solutions was the storage in Buffer, and we found it challenging to implement it in C++. We tried to use files, and there was a delay. In other languages, we have found the use of Buffer available. However, there is much necessary to use the files.

Six: In general, designing a Compiler to function as an application or a system, programming can be used by the keyboard itself. We prepare most of the programs from the High Level in
the form of DLL libraries. This dramatically simplifies and speeds up the tasks of the Compiler.

Seven: This subject is worthy of more research.
References


[2] Kepuska, Vetron website, retrieved on 02/28/19 @ 17:37 my.fit.edu/~vkepuska/web/


[28] https://cmusphinx.github.io/wiki/