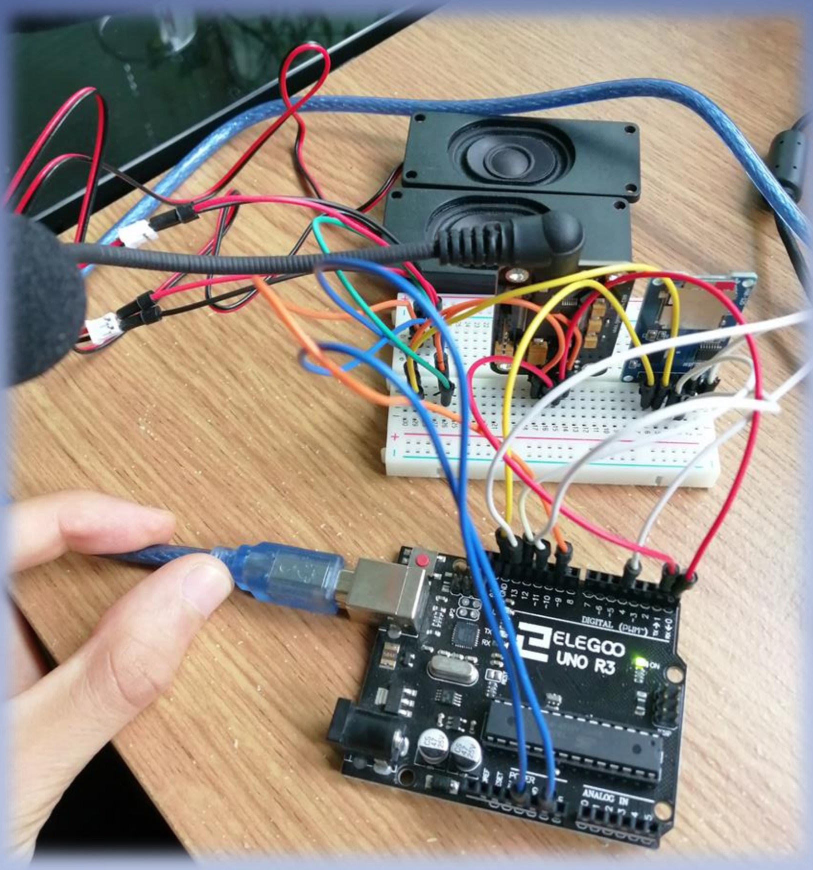


OTTO

Voice Recognition Virtual Assistant AI

06/01/2021 – Mario Portillo Hernaiz

Content

ANALYSIS

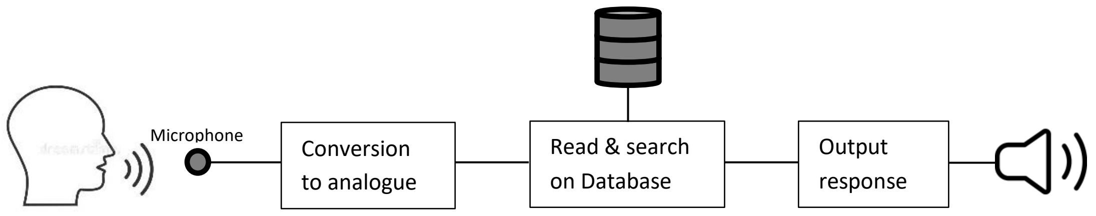
1.1 Background

During my gap year I was craving to do some sort of robot, or some sort of device that I could use on a daily basis. This got me thinking into what sort of things a device could take over so that the job is done as good as or even better than I do it. This is when I thought about “Alexa”, an amazon product. Alexa is used around the house for many non-physical things, it can be used to play music, to switch on/off some lights, to add reminders, to play the news and even play some games. This was the perfect device that I could re-create and re-program as it was small and didn’t need any physics. What I mean by physics is a robot that has to be moving around to achieve a task and, in this case, the device doesn’t need to move to hear my voice.

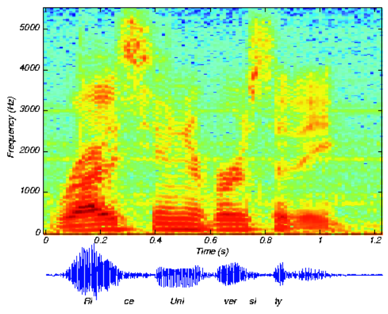
After what felt like an hour or two, I finally found the name I would give to this new device: OTTO.

1.2 Initial Research

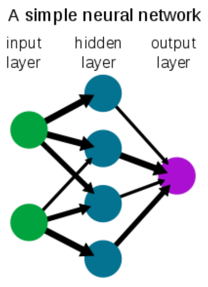
First of all, I had a look at how Alexa worked and if there were already other devices similar to Alexa created. Some other devices I found were Google assistant and Siri. Google assistant can be used both on phone and in a device whereas Siri, I believe, can only be used on phone. They all use a similar way of making things work. Below is a diagram showing it.



It seems pretty easy to record a voice, read it and find it on a database and then output the response. But in order to **read** the voice recording and **understand it**, search it in a database and output the required response, I would need to create an AI voice recognition from scratch. During my research I found quite a few ways in doing so;



1. Simple word patter matching → this method involves recognizing whole words based on their audio signature. It uses a spectrogram, a three-dimensional graph displaying time on the x-axis, frequency on the y-axis, and intensity is represented as colour. On the right is an example of this. The disadvantage is that it can only have a limited number of words it can understand; on the contrary, for this project I won’t use a large vocabulary, only a few words are needed.
2. Language model → using mathematics to analyse a language to find pattern. This method can understand large and complex phrases.
3. Artificial Neural Networks (ANNS) → a mathematical model that learns and is inspired by biological neural networks. It models a mathematical function from inputs to outputs based in the structure/parameters of the network. Using speech recognition with this concept is very useful as the algorithm will learn new concepts each time it runs. The only problem is that it’s very complex to create, especially from scratch. That’s why I will search potential programs or apps that already have that code implemented.



Listed below are the tools and libraries that could be potentially used for voice recognition so that I will not need to code it from scratch.

1. Google Speech API → This Google program converts speech to text which is perfect for my project as it would make it easier to read and reply with the correspondent answer.
2. Bing Speech API → This program is the same as Google speech API.
3. Amazon Alexa voice service → this allows you to create your own programs on an Alexa (not really what I plan on doing so I will discard this).
4. Facebook's Wit.ai → A natural language interface for applications that turns sentences into structured data, meaning that you can create apps you can talk or text to and it will keep learning with each interaction. Again, this may be useful, but it is not the objective of my project so I will also discard it.

Listed below are open sources of recognition libraries which can be more useful than the list above as it would not involve paying a fee.

1. CMU Sphinx → This is a group of recognition system developers at *Carnegie Mellon University*, it is written in Java, but it can be used in other programming languages such as Python or C#.
2. Kaldi → Easy to use and written in C++, Kaldi is an open-source recognition system.
3. HTK (Hidden Markov Model Toolkit) → is mainly made for statistical analysis modelling techniques owned by Microsoft but freely available and coded on C.

1.2.1 Hardware

The following open-source electronics prototyping platforms are potential hardware platforms I could use to develop this project.

1. **Arduino** → Easy to use hardware and software platform. This is the number one platform I should use as I already worked and created a few projects with it.

2. **Teensy 3.6** → Small and powerful bit of hardware that is compatible with Arduino. Specifications: 180-MHz Cortex M4F, 256 kB RAM, 1-MB flash storage, 4K EEPROM. Features: Teensyduino for Arduino IDE, SD card reader.

3. **Netduino N3 Wi-Fi** → Another alternative to Arduino and uses .NET MicroFramework and Netduino.Foundation Framework meaning it's programmed in a high-level language (C #). Specs: 168 MHz, 164+ kB RAM, 1408-kB flash storage. Features: .NET MicroFramework, Netduino.Foundation libraries, Wi-Fi, SD card reader, 22 GPIO.

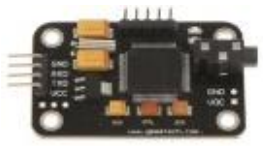
4. **Particle Photon** → Has a built-in Cypress Wi-Fi chip and comes with free connectivity with the Device Cloud and all its features. It even provides a Javascript and mobile SDKs along with its web and local IDE. Specs: 120-MHz ARM Cortex M3, 128 kB RAM, 1-MB flash storage. Features: Device Cloud, SDKs for mobile and ParticleJS, 18 GPIO.

5. **SparkFun Thing Plus** → Has built in Wi-Fi and Bluetooth and it is easy to program as it's compatible with Arduino IDE. It also contains sensors (described in the feature section below). Specs: 240 MHz, 520 kB SRAM, 16-MB flash storage. Features: Wi-Fi, Bluetooth, BLE, Li-Po charger, Hall-effect sensor, capacitive touch sensor, temperature sensor.

6. **SparkFun Red Board Artemis** → A better version of the Arduino Uno, comes with Bluetooth and ups the specs significantly as well as being compatible with the Arduino IDE. Specs: 48 MHz (96 MHz turbo), 384 kB RAM, 1-MB flash storage. Features: Arduino IDE compatibility, Bluetooth, 24 GPIO.

7. **Silicon Labs Wonder Gecko** → Contains an LCD which can be very helpful to display information but the only problem it is very expensive. It is quite easy to use and contains also a variety of sensors. Specs: 48-MHz Cortex M4, 32 kB RAM, 256-kB flash storage. Features: Integrated debugger, energy monitoring system, various sensors, LCD, Simplicity Studio.

In conclusion, Arduino is the number one platform I should work on as most of them are compatible with Arduino. I may also use the SparkFun Thing Plus as it has built in Wi-Fi and Bluetooth as well as being compatible with Arduino. Having a look at the "Silicon Lab Wonder Gecko" it gave me the idea of using an LCD to display data such as the time and date so that it won't have to be asked every time.



After a few hours of more research, I finally found the hardware that will help me with the voice recognition part of the project. The hardware is called "Voice Recognition Module" and it receives voice commands and responds through a serial port interface. The only problem is that this piece of hardware only has up to 15 voice commands, but the project being a prototype, for now I can work with only these. It is also compatible with Arduino. I will also need a "USB to TTL Serial Converter" in order to pass commands from a computer to the Voice Recognition Module.

Therefore the main piece of hardware I will work on is the "Arduino One" or another version of it, like "Elegoo Uno R3", which will be the main motherboard wich will control and connect all the pieces together in order to complete the project. I will also use a SD Card Reader Module to save and transfer the data to the Arduino so that it won't need to be connected to a laptop/computer. A speaker compatible with Arduino will also be used in order to output the voice replies.

1.2.2 Software

The following software platforms are related to the above hardware platforms.

1. **Arduino** → Arduino hardware is programmed in Arduino IDE in the language C/C++.
2. **Teensy 3.6** → As it is compatible with Arduino, the hardware is coded in the C language.
3. **Netduino N3 Wi-Fi** → Uses the C# language.
4. **Particle Photon** → This hardware is programmed in Arduino IDE and coded with the C++ language.
5. **SparkFun Thing Plus** → Compatible with Arduino IDE and so it will be coded in the language C/C++. I believe it can also by coded using python.
6. **SparkFun Red Board Artemis** → Compatible with Arduino IDE and coded with the C++ language. I also believe it can be coded using python.
7. **Silicon Labs Wonder Gecko** → Supports the C language.

In conclusion, the language I need to work on is the C language, be it C or C++ or C#. Some of the software platforms also use python which can be very helpful as I have done some previous work on it. The software I will use will be the Arduino IDE. In order to work with the Voice Recognition Module, I will have to manage a Serial Monitor software to send HEX commands from the computer to the USB Module, we will look into this in the next chapter.

1.2.3 Understanding the Voice Module

Understanding the Voice Module and how it works is more or less the most important part of this project. It is the key factor that allows me to speak with the Arduino and which manages all the voice recognition AI side. For now, I will write all the research. The testing and the actual coding of it will be done later on.

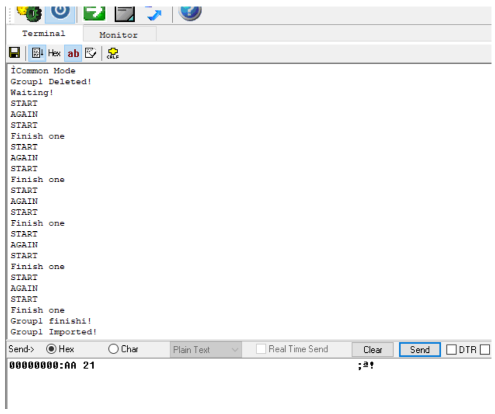
The Voice Module can store up to 15 voice commands split into 3 groups with 5 voice commands in each group. These voice commands need to be recorded group by group and then imported before it can recognise the voice command. Its recognition accuracy is of 99% under ideal environment, if someone else speaks the commands, they may not be recognised.

Its interface is 5V TTL. The serial format is: 8 data bits, no parity, 1 stop bit. The default baud rate is of 9600 but can be changed and the send format is HEX. Each voice instruction has a maximum length of 1300ms which allows for most words and even small phrases to be recorded. Once you start recording you can't stop the recording process until you finish all the 5 voice commands on one group. Also, the previous recordings of that group will be erased.

The command format is "Head + Key" where "Head" is a 0xAA and "Keys" are shown in the table below. I will write only the main ones I will need throughout the project, but there are a few more. It can get a bit confusing at first, but with the following examples in the next chapters, it'll become clearer.

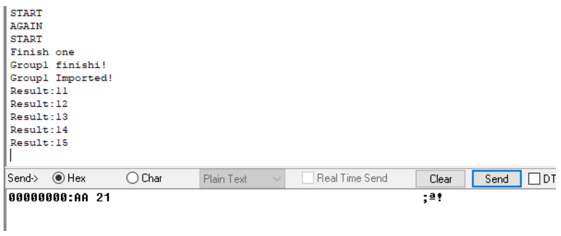
KEY	Description
0x00	Enters into the "waiting" state.
0x01	Deletes the instructions in group 1
0x02	Deletes the instructions in group 2
0x11	Begins to record instructions of group 1
0x12	Begins to record instructions of group 2
0x21	Imports group 1 and gets ready for voice commands
0x22	Imports group 2 and gets ready for voice commands
0x36	Switches to common mode (returns a string message for every command. For example, when deleting a group, it'll return "Group n deleted" where n is a number from 1 to 3)
0x37	Switches to compact mode (returns a byte message for every command. For example, when deleting a group, it'll return "0xcc" which means successful operation)

Now, in order to send the voice commands using HEX format into the voice module we will need to download an app that does that. In this case I found one called "Access Port" which does the job (the link to it can be found here: http://www.sudt.com/en/ap/). Once you get the hang of it, it is very simple to use so in the following chapters we will work with it and test it in order to send Voice Commands in HEX format to the Voice Module.



```
ÌCommon Mode
Group1 Deleted!
Waiting!
START
AGAIN
START
Finish one
START
AGAIN
START
Finish one
START
AGAIN
START
Finish one
START
AGAIN
START
Finish one
START
AGAIN
START
Finish one
Group1 finishi!
Group1 Imported!
```

```
00000000:AA 21                                    ;ª!
```



```
START
AGAIN
START
Finish one
Group1 finishi!
Group1 Imported!
Result:11
Result:12
Result:13
Result:14
Result:15
```

```
00000000:AA 21                                    ;ª!
```

1.3 Objectives

To fully complete my project, I will need to achieve the following objectives:

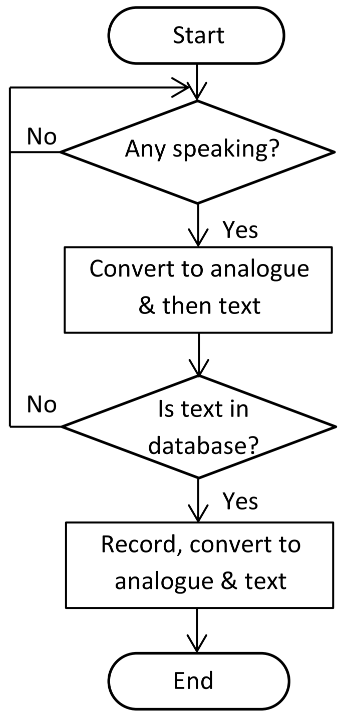
1. Research
    a. As I'm slightly new to programming and working with hardware, I should research potential hardware devices I will need in order to achieve the project's goals.
        i. What I will use for the input of the device.
        ii. What I will use as the main motherboard/main device.
        iii. What I will use for the output of the device.
    b. I should also look into what software I will need in order to work with the desired hardware.
        i. Research what software I will use to code a Voice Recognition AI.
        ii. Research what programming language I will use to code the hardware components.
    c. Look at how I will fit all the hardware parts in a 3D model in order to keep it clean and neatly together to finalise the product.
2. Creating OTTO
    a. Build the hardware so that it can input data, process it and then output desired data. The following parts I need are;
        i. Microphone – Record and save analogue data.
        ii. USB to TTL converter – Passes commands from the computer to the voice module.
        iii. Voice Module – Receives voice commands and understands them.
        iv. ROM – Save required data to output.
        v. Speaker – Output the required replies.
    b. Code the software.
        i. Program the hardware so that it works all together.
        ii. Convert analogue to text or other format of data.
        iii. Create database of my desired data I wish the program to understand and output.
        iv. Output reply through speaker.
3. Testing – Test and repeat trial and error until desired goal is achieved.
4. Updating the project:
    a. Making sure the "listening loop" is not on for ever. Make sure it only turns on when the name "OTTO" is called.

DESIGN

2.1 Flowcharts

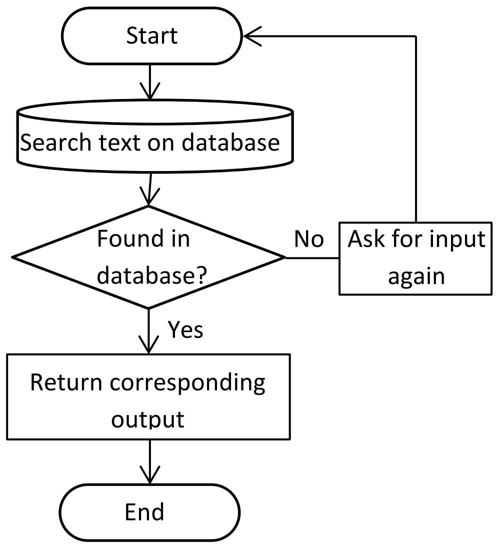
The following flowcharts are a breakdown of all the different sections in the project and how each will work in order to produce the outputs I need.

Section 1: Input and Conversion


Start
Any speaking?
No
Yes
Convert to analogue & then text
Is text in database?
No
Yes
Record, convert to analogue & text
End

Firstly, OTTO will await for any sort of speaking that the microphone can catch. Then it will see if any of those words are stored in the database. If they are it means OTTO has been called meaning someone is asking for a reply for a question. This means it should record the question and at the same time convert it to analogue and then text (or any other format). With this conversion it can now go to Section 2.

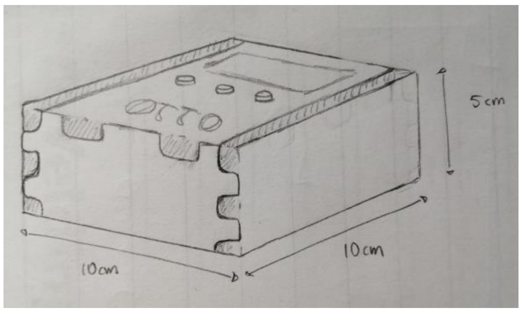
Section 2: Read & Search on Database



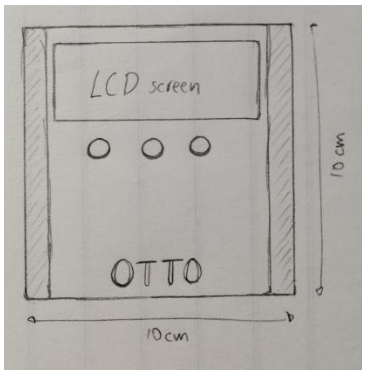
Next up, OTTO should be able to read the input, search it on the database and be able to output the corresponding reply. If the input is not found in the database, OTTO should be able to give a reply with this type of error and ask again for an input.

2.2 Sketches

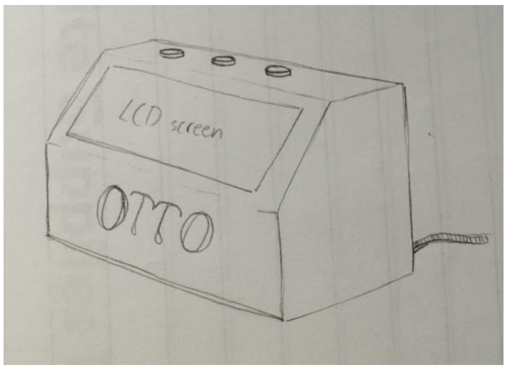
I might not be an incredible drawer, but in order to visualize the final design and to see what dimensions I want it to have, I drew the following sketches. These sketches do not represent the final design neither do the measurements, but they are the ideal numbers that I wish to keep.



From this angle you can see how I wish to fit the pieces so that it resembles a box. I wish to have OTTO with 5cm height and 10cm width and length.



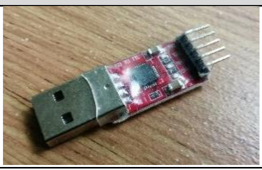
From bird's eye view, you can see how I wish to add an LCD as well as a few buttons (the 3 circles underneath the LCD). For a final touch I also add the name onto the design. If I don't add either buttons or an LCD to the final design then it will just be a wooden/plastic cube with the components inside.

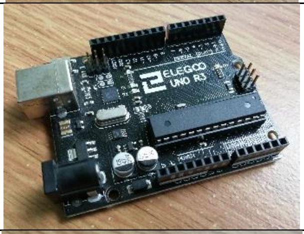
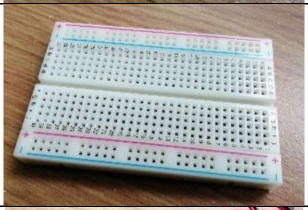
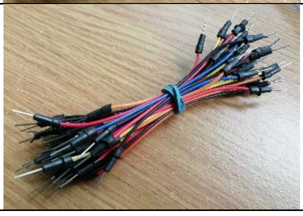


Another design I created would have a more comfortable way of looking at the LCD if it was possible to create that angle with the hardware. It would contain the buttons on top and also the little touch of the name onto the side.

2.3 Hardware Used

Before assembling and programming the project, let's look into what hardware I will be using and what they do.

Picture	Description
	The USB to TTL converter module will be used to convert any USB port into a 5V TTL signal. It will send data from the computer through it, to the voice recognition module.

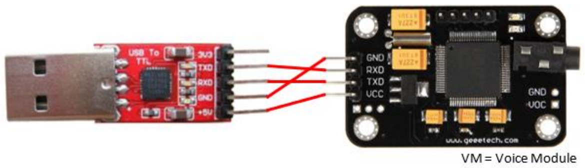
	Both, the voice recognition module (VM) and microphone are the pieces of hardware that allows several voice inputs to be recognised and understood from any other different voice inputs. It allows a recording of 15 voice commands and has a 99% recognition accuracy (under ideal environment).
	Arduino Uno, the main motherboard, is where all the work will be done in. All the hardware pieces will be connected to this and the software program will be uploaded to this as well.
	The SD Card reader with the SD Card will both be used to store all the voice replies and to transfer the data through to the Arduino Uno motherboard. The SD's capacity is 32GB which I believe will be more than enough for this project.
	The breadboard will combine all the pieces of hardware together using the cables. The diagram and description of how this will be done will be written and illustrated down below in the next section.
	The two speakers will output the voice replies from OTTO. They are compatible with Arduino. Maximum power: 3W.
	The cables will connect all the hardware pieces together and transfer the electricity through it.

2.4 Hardware Design & Cable Connections

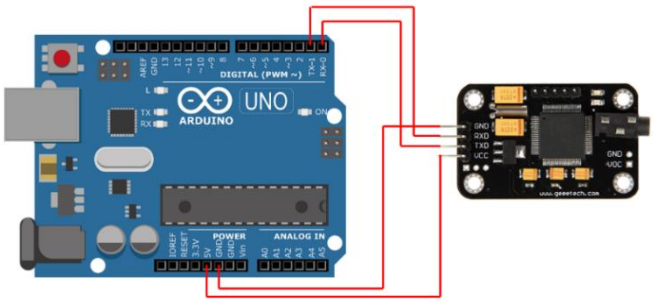
Now that I have all the pieces of hardware that I will be using, I can start designing the circuit diagram. I will do this in 3 separate parts, firstly I will create the circuit diagram of the voice module and the Arduino Uno to test it and see how it works. I will code it and then use the serial monitor in the Arduino IDE software, instead of the speaker, to output replies once I input a voice command. Once it has been tested, it works and I understand it, I will create the second circuit diagram of the speaker. Here I will work with the speaker and SD Card reader. Without using the microphone, I will code the program, input a text voice command and listen to the speaker's output. If after testing it works and I understand the concept of using these pieces of hardware I can then continue to the next part. The final part will consist of combining both sections together and thus OTTO will be completed.
Below are the diagrams with a brief description. The detailed information of all the connections will be shown in the next section.

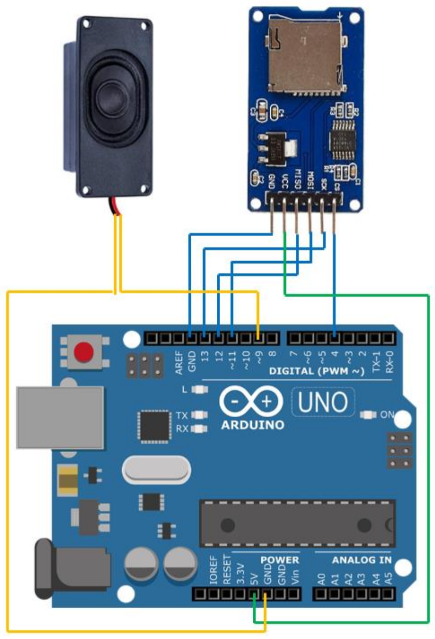
Part one:

Firstly, we will connect both the USB to TTL module with the Voice Module in order to transfer the voice commands from the computer to the Voice Module


VM = Voice Module

After transferring the data, we will connect the Voice Module with the Arduino Uno. Now we can work with the code and the Voice Module to test it.





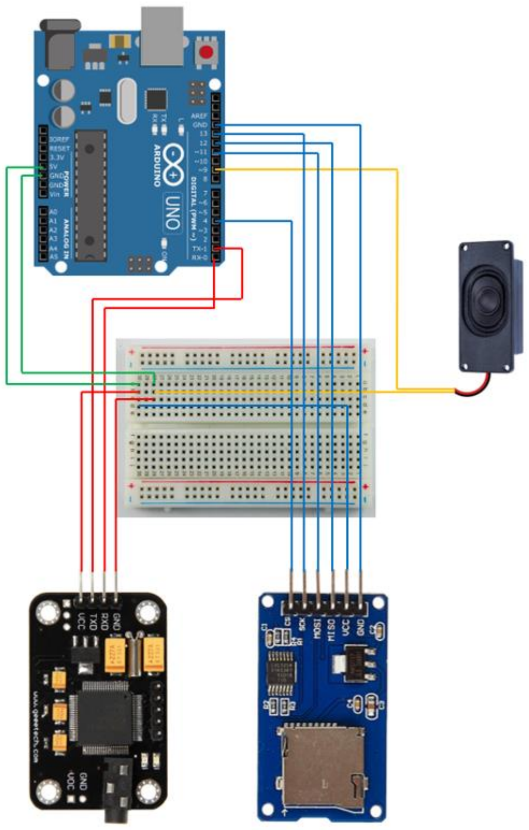
Part two:

On this part we will connect both the SD Card reader and the Speakers to the Arduino Uno. In this case I only showed one speaker but when connecting two or more speakers, the breadboard will be needed. Once the connections have been made, we can now code it and test it.

The SD Card will be used separately to transfer the voice responses into it.

Part three:

Finally, we will connect all the sections together in order to create OTTO. This time we will need the breadboard as pins such as the 5V and the GND need to be used more than once for all, Voice Module, SD Card reader and the speakers.



IMPLEMENTATION

3.1 Voice Recognition Section

How to create the voice recognition software and hardware:

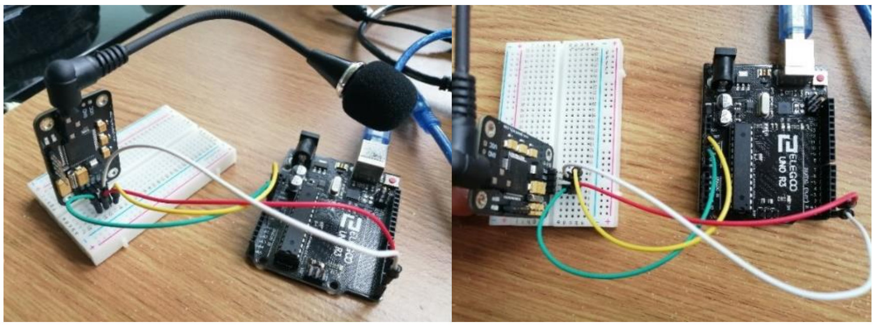
1. Firstly, connect the USB to TTL module with the Voice recognition module.
   a. RDX → TXD
   b. GND → GND
   c. VCC → +5V
   d. TXD → RDX
2. Use Access Port to record the voices and import them onto the voice recognition module.
   a. Step 1: In the software go to Tools>Configuration, then change the Port configuration and keep the settings to Baud rate: 9600 and send display as "Hex format". (In order to check what Port, the TTL module uses, search "Device Manager" in the windows tab and search the Ports "COMP". Find the comp that the "Silicone lab ___" uses)
   b. In the window at the bottom, refresh so that it inputs HEX (it should show: "00000000:"), then write the command to start the common mode ("AA 36") and press send in order to open the common mode. A text saying "Common Mode" should be displayed.
   c. Just in case delete all the previous files from the group you will record in by sending the "AA 01" command. Then send the waiting state command "AA 00" to make sure the voice module is ready to record.
   d. When you are ready, send another command in order to start recording ("AA 11").
   e. When we see "START", we say the command we want. When we see "START" again, we repeat the same command until we see "Finish one". We then say the next command when we see "START" and so on with all other commands until we see "Group finished". Make sure to give a fraction of a second when you see "START" so that it gives it time to start the recording.
   f. Once all voices are done and we see "Group Finished", import the voices to the board ("AA 21"). The text "Group Imported" will be displayed.
3. Disconnect the TTL and connect the voice recognition module with the board.
   a. GND → GND (Breadboard)
   b. VCC → +5V (Breadboard)
      i. Every electronic thing is connected with the GND and the +5V of the Arduino that then connects in the breadboard.
   c. TX → RX (Arduino)
   d. RX → TX (Arduino)
4. Upload the code. (Don't forget to change connection of the TX and the RX pins from the Voice Recognition Module when uploading the source code and then connecting them back as they were once the code has been uploaded. This changed connection is: TX → TX and RX → RX).
5. You are ready to test the voice commands once you upload the program.

```
byte com = 0;

void setup() {
 Serial.begin(9600);
 delay(2000);
 Serial.write(0xAA);
 Serial.write(0x00);
 delay(1000);
 Serial.write(0xAA);
 Serial.write(0x37);
 delay(1000);
 Serial.write(0xAA);
 Serial.write(0x00);
 delay(1000);
 Serial.write(0xAA);
 Serial.write(0x21);
}

void loop() {
while(Serial.available()) {
com = Serial.read();
 if(com == 0x11) {
   Serial.println("Yes Mario?");
   delay(100);
   com == 0x00;
   }
 if(com == 0x12) {
   Serial.println("My name is OTTO and I
was built by Mario Portillo during the
2020 Pandemic.");
   delay(100);
   com == 0x00;
   }
 if(com == 0x13) {
   Serial.println("You've got 3 tasks to
complete: x. y. z.");
   delay(100);
   com == 0x00;
   }
 if(com == 0x14) {
   Serial.println("The time is for you to
buy a watch");
   delay(100);
   com == 0x00;
   }
 if(com == 0x15) {
   Serial.println("5");
   delay(100);
   com == 0x00;
}}}
```

This is how it will look:



Source code:

The basic task of the code is to print a line in the monitor once a recognised word has been heard.

Firstly, it'll set up the serial connection and change the baud rate to 9600. It will then write the following commands which are formed as "Head + Key" where "Head" is the *0xAA* and the "Key" are as follows:

0x37 = Switching to compact mode (instead of common mode which we saw earlier).

0x21 = Importing group 1 (the more voice commands, the more groups that will be recorded and thus imported)

Once it's ready to use the group it'll run a loop where it'll print a sentence once it hears a voice command. For this test it will only print "Voice" and a number where the number (1 to 5) represents each voice command. Each 0x11 to 0x15 represents the voice commands from 1 to 5 respectively.

3.2 Speaker Section

How to make the speaker part of the project:

1. Connect the SD Card module with the Arduino board:
   a. VCC → +5V
   b. GND → GND (On digital)
   c. CS → Digital pin 4
   d. SCK → Digital pin 13
   e. MOSI → Digital pin 11
   f. MISO → Digital pin 12
2. Connect speaker with the Arduino board:
   a. One wire of the speaker jack → GND
   b. The other → Digital pin 9
3. Now we just have to create & convert the audio files. The type of audio we need is “WAV”. Record the voices you need and then convert them using an online website. The settings need to change for the audio file:
   a. Bit resolution: 8 bit
   b. Sampling rate: 16000 Hz
   c. Audio channels: mono
4. Normalise audio to get a clear output. Once done, save the recordings to the SD card. Then plug in the SD card in the SD card reader.

Source code:

```
#include <SD.h>                          //include SD module library
#include <TMRpcm.h>                       //include speaker control library
#define SD_ChipSelectPin 4               //define CS pin
TMRpcm tmrpcm;                           //create an object for speaker library

void setup(){

  tmrpcm.speakerPin = 9;                 //define speaker pin. Must use pin 9 of the Arduino Uno

  if (!SD.begin(SD_ChipSelectPin)) {      //see if the card is present and can be initialized
    return;                              //don't do anything more if not
  }

  tmrpcm.setVolume(6);                   //0 to 7. Set volume level
  tmrpcm.play("1.wav");                  //the sound file "1" will play each time the Arduino powers up,
or is reset
}
void loop(){}
```

3.3 Joining both sections (creating OTTO)

Now that we have created and tested the two sections above, we can complete the project.

1. In the breadboard connect both the GND and the 5V in different slots.
2. The order doesn't really matter but I'll start with the Voice Module, connect the following pins as follows:
    a. GND (Voice Module) → GND (Breadboard)
    b. VCC (Voice Module) → +5V (Breadboard)
    c. TX (Voice Module) → RX (Arduino)
    d. RX (Voice Module) → TX (Arduino)
3. Now let's connect the SD card reader pins:
    a. GND (SD Card reader) → GND (Breadboard)
    b. VCC (SD Card reader) → +5V (Breadboard)
    c. CS (SD Card reader) → Digital Pin 4 (Arduino)
    d. SCK (SD Card reader) → Digital Pin 13 (Arduino)
    e. MOSI (SD Card reader) → Digital Pin 11 (Arduino)
    f. MISO (SD Card reader) → Digital Pin 12 (Arduino)
4. Finally let's connect the speakers (As I have two of them, I will also need the breadboard for this):
    a. One wire of the speaker → On one slot of the breadboard
    b. The other wire → On another slot of the breadboard
    c. The first slot of the breadboard → GND (Breadboard)
    d. The second slot of the breadboard → Digital Pin 9 (Arduino)
5. Once everything is connected remember to change the connections of the TX and RX of the voice module before uploading the code and then reconnect them.
6. Once the code is uploaded and you've given it a few seconds for the project to load, you can now start to test it. Watch the magic happen!

Source code:

```
byte com = 0;
#include <SD.h>                    //include SD module library
#include <TMRpcm.h>                //include speaker control library
#define SD_ChipSelectPin 4         //define CS pin
TMRpcm tmrpcm;                     //crete an object for speaker library
```

```
void setup() {
  Serial.println("Start Up");
  Serial.begin(9600);
  delay(2000);
  Serial.write(0xAA);
  Serial.write(0x00);
  delay(1000);
  Serial.write(0xAA);
  Serial.write(0x37);
  delay(1000);
  Serial.write(0xAA);
  Serial.write(0x00);
  delay(1000);
  Serial.write(0xAA);
  Serial.write(0x21);
  Serial.println("First phase done");

  tmrpcm.speakerPin = 9;               //define speaker pin.
  if (!SD.begin(SD_ChipSelectPin)) {     //see if the card is present and can be initialized, don't do anything
more if not
    return;
  }
  tmrpcm.setVolume(6);
  Serial.println("Second phase done & READY TO SPEAK");
}
```

The above source code sets up the Arduino to make sure the voice input output are read correctly.

```
void loop() {
while(Serial.available()) {

com = Serial.read();
 if(com == 0x11) {
  Serial.println("Yes Mario?");
  delay(100);
  com == 0x00;
  tmrpcm.play("2.wav");
  }
 if(com == 0x12) {
  Serial.println("My name is OTTO and I was built by Mario Portillo during the 2020 Pandemic.");
  delay(100);
  com == 0x00;
  tmrpcm.play("5.wav");
  delay(12000);
  tmrpcm.play("6.wav");
  }
 if(com == 0x13) {
  Serial.println("You've got 3 tasks to complete: x. y. z.");
  delay(100);
  com == 0x00;
  tmrpcm.play("7.wav");
  }
 if(com == 0x14) {
  Serial.println("The time is for you to buy a watch");
  delay(100);
  com == 0x00;
  tmrpcm.play("8.wav");
  }
 if(com == 0x15) {
  Serial.println("Sorry for the rudeness.");
  delay(100);
  com == 0x00;
  tmrpcm.play("9.wav");
  }}}
```

In the loop of the program we go through each voice recognition so that when an input is created, we constantly check if its in the voice module recognition.

TESTING

	Objective	Completion
1	Microphone accepts voice input.	Success
2	Microphone outputs different logs depending on voice input.	Success
3	MP3 voice responses stored into SD card.	Success
4	Speaker outputs stored mp3 sounds.	🟩
5	Connecting all hardware components into one motherboard.	Success
6	Microphone accepts input and Speaker outputs mp3 response depending on voice input.	Success
7	3D model created	Unsuccessful

EVALUATION

5.1 Completion of Objectives

1. Research
   a. Hardware research has been proved efficient since I was able to find all the necessary pieces to build the project.
   b. Software research has also proven efficient since I was able to use the selected software on the hardware.
   c. In the end, the 3D model was not implemented due to the lack of resources.
2. Creating OTTO
   a. Building OTTO has been completed and all hardware pieces connected correctly.
   b. Programming OTTO has also been completed and all hardware pieces function with the software.
3. Testing – Testing was achieved which ticked off all objectives and functionalities in the project.

5.2 Useful websites

With the help of these few websites and videos online I was able to complete OTTO and have it functioning the way I wanted.

https://audio.online-convert.com/convert-to-wav

http://www.sudt.com/en/ap/

https://www. geeetech.com/wiki/images/6/69/Voice_Recognize_manual.pdf