Project report on

Running ChatGPT using ESP32 (Text To Speech)

Submitted by:
LostAndFound io

Group Members:

1. Jjateen Gundesha BT22ECI002
2. Ayush Ambatkar BT22ECI005
3. Darshan Tate BT22ECI011

A report submitted for the partial fulfilment of the requirements of the course
ECL-108 IoT Workshop - 2

Submission Date: 21/08/2023

Task Number # 2

Under the guidance of:
Dr. Tushar Muratkar
Department of Electronics and Communication Engineering

भारतीय सूचना प्रौद्योगिकी संस्थान, नागपुर
Indian Institute of Information Technology, Nagpur

Table of Contents:

Chapter 1: Introduction

The "Speech to Text and Text to Speech Integration on Microcontroller" project represents an innovative endeavor in the realm of human-machine interaction and communication. In this report, we document the design, development, and implementation of a versatile system that seamlessly integrates both Speech-to-Text (STT) and Text-to-Speech (TTS) capabilities within a microcontroller environment.

As technology continues to evolve, natural language processing and speech recognition have become integral components of human-computer interaction. This project was conceptualized to bridge the gap between spoken language and digital data processing, enabling real-time conversion of spoken words to text and vice versa.

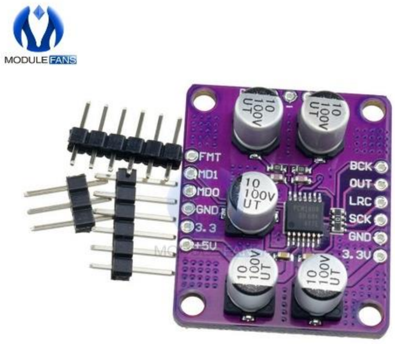
The primary objectives of this project include:

1. **Speech Recognition**: Implementing a reliable speech recognition system that can capture and transcribe spoken words accurately.
2. **Text-to-Speech Conversion**: Developing a Text-to-Speech module capable of converting text data into natural-sounding speech.
3. **Integration**: Seamlessly integrating both STT and TTS functionalities on a microcontroller platform to create a cohesive and interactive system.
4. **User Interface**: Providing a user-friendly interface for triggering and interacting with the system, enhancing accessibility and usability.
5. **Cloud Integration**: Leveraging cloud-based services to enhance the accuracy and versatility of speech recognition, as well as accessing cloud-based Text-to-Speech engines for lifelike speech synthesis.

Throughout this report, we will delve into the technical details of our implementation, highlighting the hardware and software components utilized, the challenges encountered, and the solutions devised to address them. Additionally, we will discuss the potential applications and future enhancements of this system, which could range from smart home automation to assistive technologies for individuals with disabilities.

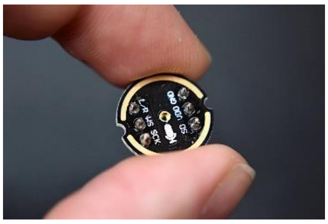
The project represents a convergence of cutting-edge technologies, from IoT connectivity and audio processing to natural language understanding, with the ultimate goal of enhancing human-machine communication. Through this report, we aim to provide a comprehensive overview of our journey in realizing this goal, from inception to implementation.

ESPRESSIF
ESP32-WROOM-32UE
3V3 EN VP VN 34 35 32 33 25 26 27 14 12 GND 13 D2 D3 CMD 5V
GND 23 22 TX RX 21 GND 19 18 5 17 16 4 0 2 15 D1 D0 CLK
EN
Boot



[11] Fig. 1: ESP32 board

[5] Fig. 2: I2S Amplifier



[6] Fig. 2: INMP 441 Mic.

Chapter 2: Code

// Speech To Text

```
//////Running code with board maneger verstion 1.0.6


#define button 23 //IR Sensor


#define led_3 4


#define RXp2 16
#define TXp2 17
#include "Audio.h"
#include "CloudSpeechClient.h"
int i=0;
void setup() {
pinMode(button, INPUT_PULLUP);


pinMode(led_3,OUTPUT);
  Serial.begin(115200);
 Serial2.begin(115200, SERIAL_8N1, RXp2,TXp2);
   Serial2.println("Intialising");
//  Serial.println(My_Data);
}


void loop() {
```

```
//      digitalWrite(led_3, 0);


   if(i==0){
   Serial.println("\nPress button");
   i=1;
   }
 //  if(i==1){delay(1);}


   delay(500);
   if(digitalRead(button)==0){
     Serial2.println("\r\nPlease Ask!\r\n");
     delay(2100);
  Serial.println("\r\nRecord start!\r\n");
  //Serial2.println("\r\nRecord start!\r\n");
  Audio* audio = new Audio(ADMP441);
 //Audio* audio = new Audio(M5STACKFIRE);
  audio->Record();
  Serial.println("Recoding Complited Processing");


  CloudSpeechClient* cloudSpeechClient = new
CloudSpeechClient(USE_APIKEY);
  cloudSpeechClient->Transcribe(audio);
  delete cloudSpeechClient;
  delete audio;
  i=0;
  }
  if(digitalRead(button)==1){
    delay(1);

    }
}
```

// Text To Speech

```
#include "Arduino.h"
#include "WiFi.h"
#include "Audio.h"
#define uart_en 15
#define RXp2 16
#define TXp2 17
#define I2S_DOUT      25
#define I2S_BCLK      27
#define I2S_LRC       26

Audio audio;

void setup()
{
  Serial.begin(115200);
    Serial2.begin(115200, SERIAL_8N1, RXp2,TXp2);

  WiFi.disconnect();
  WiFi.mode(WIFI_STA);
  WiFi.begin( "jjj", "12345678");
    Serial.print("Connecting");

  while (WiFi.status() != WL_CONNECTED){
    Serial.print(".");
    delay(1500);
  }
  Serial.println();
  audio.setPinout(I2S_BCLK, I2S_LRC, I2S_DOUT);
  audio.setVolume(150);
  audio.connecttospeech("Welcome back master! How can I help you
today? ", "en-US"); // Google TTS
}

void loop()

{
if (Serial2.available()){
  String Answer = Serial2.readString();
  Serial.println(Answer);
audio.connecttospeech(Answer.c_str(), "en-US");
  }
  audio.loop();


}


void audio_info(const char *info) {
  Serial.print("audio_info: "); Serial.println(info);}
```

Chapter 3: Working

Working Mechanism

The working mechanism of our Speech to Text and Text to Speech Integration on Microcontroller project can be divided into two main components: Speech-to-Text (STT) and Text-to-Speech (TTS). Below, we provide an overview of how each component functions:

Speech to Text (STT):

1. **User Input**: The system is initiated when a user presses a designated button (IR Sensor) connected to the microcontroller.
2. **Audio Recording**: Upon button press, an audio sensor (e.g., ADMP441) is activated to record the user's spoken words. The audio sensor captures sound data, which is then processed by the microcontroller.
3. **Cloud-Based Speech Recognition**: The recorded audio data is sent to a cloud-based speech recognition service using an API key. The service transcribes the spoken words into text.
4. **Text Output**: The transcribed text is returned to the microcontroller, which can then be used for various applications, such as voice commands, text-based data entry, or any other text-based processing.
5. **User Feedback**: Feedback is provided to the user, indicating that the recording has been completed, and the transcribed text is ready for use.

Text to Speech (TTS):

1. **User Input**: Text data is obtained from a user source, typically via a user interface.
2. **Text-to-Speech Conversion**: The text data is sent to a Text-to-Speech module within the microcontroller. This module converts the text into audible speech.
3. **Audio Output**: The generated speech output is directed to an audio output interface, which could be a speaker or headphones.
4. **User Feedback**: The synthesized speech is played back to the user, providing a natural and human-like auditory response.

Integration:

The seamless integration of STT and TTS functionalities ensures a two-way communication system. Users can speak to the system, which transcribes their speech into text for processing. Conversely, users can input text, which is transformed into speech for auditory feedback.

Overall Workflow:

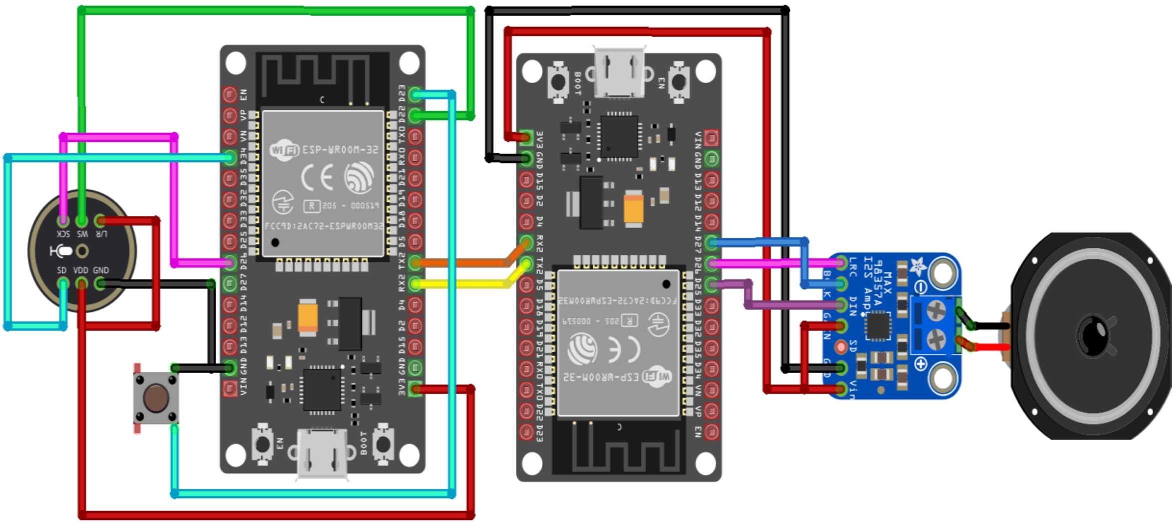
1. User interaction triggers either the STT or TTS functionality, depending on the mode (input or output).
2. The respective module processes the input data (audio or text).

3. Cloud services are utilized for enhanced accuracy in speech recognition (STT) and lifelike speech synthesis (TTS).

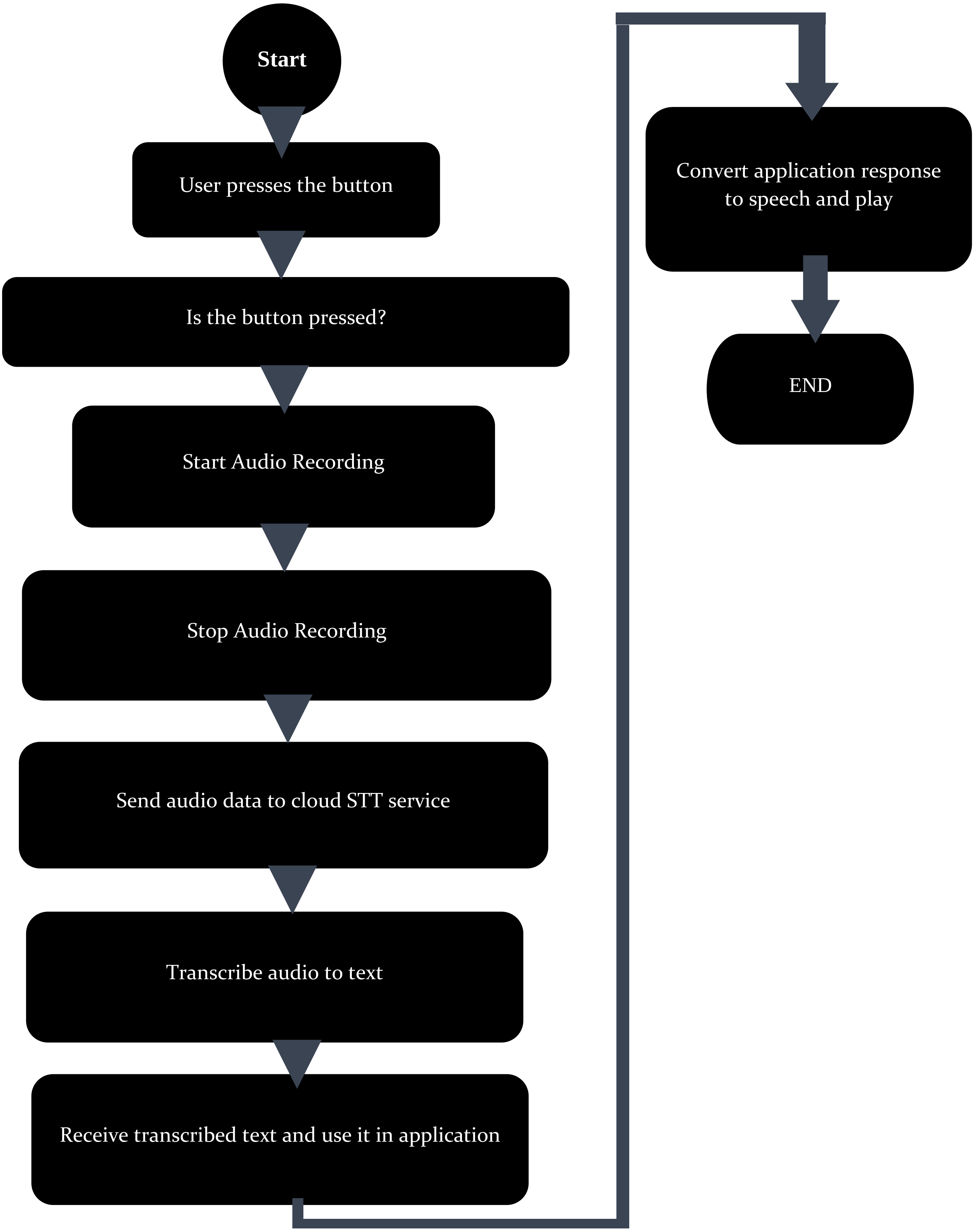
4. Users receive feedback in the form of transcribed text or synthesized speech, creating a seamless human-machine interaction experience.

The working mechanism of this project not only empowers users with voice-based commands and feedback but also opens up possibilities for applications in voice-controlled automation, accessibility tools, and interactive human-robot interfaces.

Circuit Diagram:



Flowchart:


Start
User presses the button
Is the button pressed?
Start Audio Recording
Stop Audio Recording
Send audio data to cloud STT service
Transcribe audio to text
Receive transcribed text and use it in application
Convert application response to speech and play
END

References:

[1] http://raphaeljs.com/

[2] https://www.npmjs.com/package/justgage

[3] https://www.chartjs.org/

[4] https://www.w3schools.com/css/

[5] https://www.google.com/url?sa=i&url=https%3A%2F%2Fsharvielectronics.com%2Fproduct%2Finmp441-omnidirectional-microphone-i2s-interface-digital-output-sensor-module%2F&psig=AOvVaw31HjVTKDI3Y9GFyBToKMUN&ust=1693885761493000&source=images&cd=vfe&opi=89978449&ved=0CBAQjRxqFwoTCLDNrKuGkIEDFQAAAAAdAAAAABAZ

[6] https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.mouser.in%2Fnew%2Ftexas-instruments%2Fti-tas5760m-amplifier%2F&psig=AOvVaw3nv-6XeO_5es4HGdgr7nox&ust=1693885684081000&source=images&cd=vfe&opi=89978449&ved=0CBAQjRxqFwoTCIDPteCLkIEDFQAAAAAdAAAAABAI

[7] https://cloud.google.com/text-to-speech

[8] https://cloud.google.com/speech-to-text

[9] https://www.youtube.com/watch?v=VoanFTpCTU4

[10] https://www.youtube.com/watch?v=R-CZLimCcW8

[11] https://esp32.com/viewtopic.php?t=7294