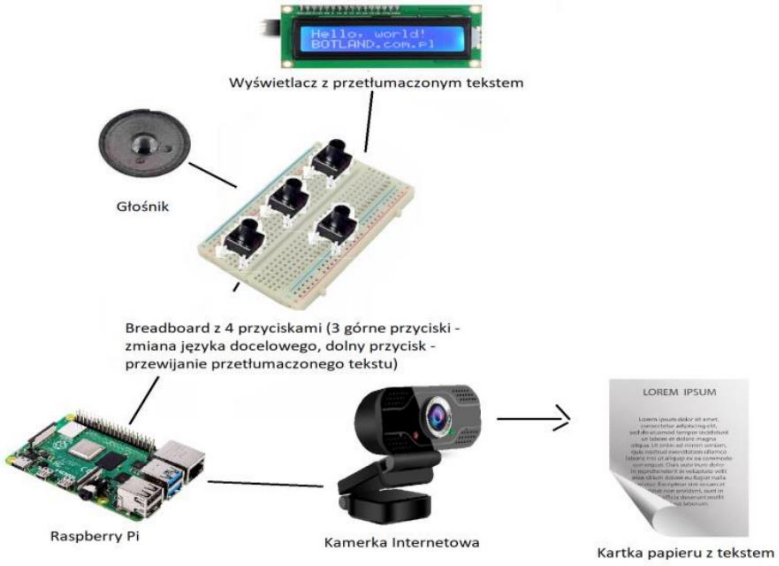
[Z7] KARTA PROJEKTU

Grupa laboratoryjna: L5 **Podgrupa: 2**	Mateusz Wyrzykowski (155 883) Kacper Molenda (156 014)	
	Prowadzący zajęcia:	**dr inż. Ariel Antonowicz**
Interaktywny Translator Obraz-Tekst-Głos		
CEL PROJEKTU	*Opracowanie interaktywnego systemu tłumaczenia tekstu z użyciem kamery internetowej oraz API do tłumaczenia i syntezatora mowy. System umożliwi użytkownikowi zarejestrowanie tekstu z kartki papieru, wybranie języka docelowego za pomocą trzech dedykowanych przycisków, a następnie przetłumaczony tekst zostanie wyświetlony na ekranie oraz odczytany przez moduł syntezy mowy.*	
SCHEMAT POGLĄDOWY (UPROSZCZONY)		
		
WYKORZYSTANA PLATFORMA SPRZĘTOWA, ELEMENTY POMIAROWE I WYKONAWCZE	*Raspberry Pi, Breadboard, Wyświetlacz LCD I2C 2x16, Głośnik okrągły z komunikacją sygnałem sterującym PWM, Kamerka internetowa, Przyciski*	

1. Cel i zakres projektu.

Celem naszego projektu było użycie dostępnego sprzętu domowego oraz wypożyczenie sprzętu laboratoryjnego, aby opracować projekt interaktywnego systemu tłumaczenia tekstu wykorzystującego kamerę internetową oraz API do tłumaczenia i syntezatora mowy. System pozwoli użytkownikowi uchwycić tekst z kartki papieru za pomocą kamery, wybrać język tłumaczenia za pomocą trzech dedykowanych przycisków, a następnie wyświetli przetłumaczony tekst na ekranie i odtworzy go za pomocą modułu syntezy mowy. Projekt opiera się na Raspberry Pi 5 i na dostępnych bibliotekach, tak aby poprawnie skonfigurować głośnik, wyświetlacz i API do tłumaczenia oraz syntezy mowy.

API użyte do tłumaczenia tekstu to DeepL API. Jest to darmowe API pozwalające na tłumaczenie do 500 000 znaków miesięcznie, co wpasowuje się w ramy naszego projektu. Drugim darmowym i pozwalającym na zamianę „tekstu na mowę” do 10 minut jest ElevenLabs. Klucze API są ukryte w pliku .env dostępnym tylko dla twórców projektu.

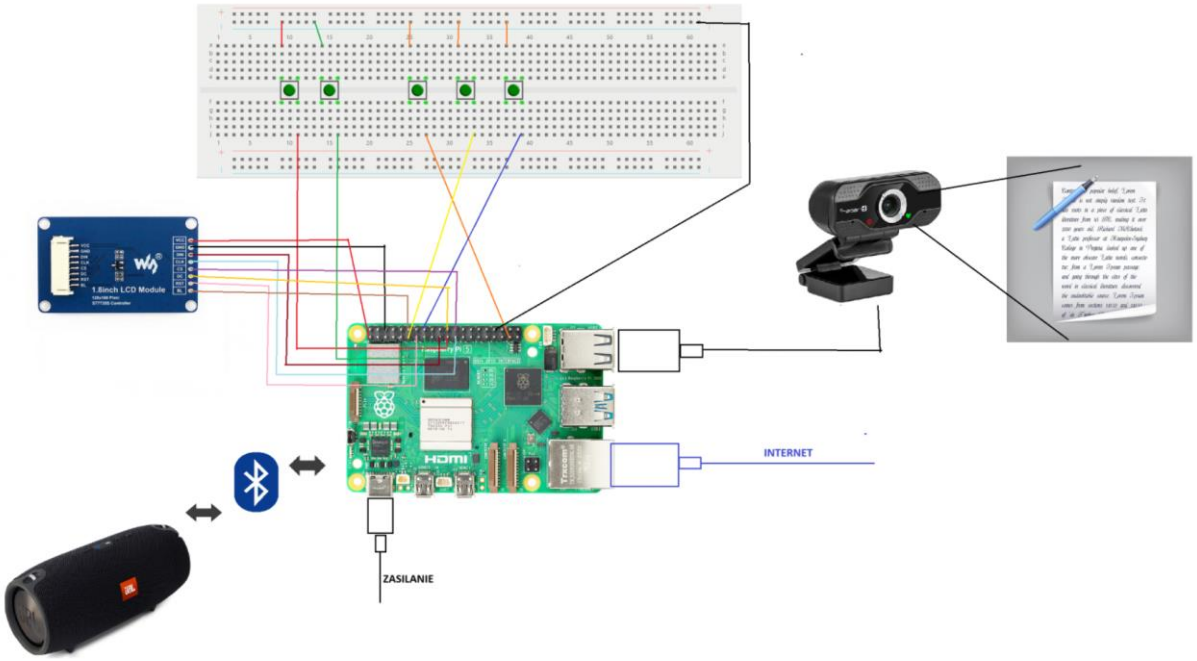
Głośnik wykorzystany do odtwarzania dźwięku to JBL XTREME, który podłączony jest do sprzętu przez Bluetooth. Pierwotnie wykorzystany miał być głośnik okrągły z komunikacją sygnałem sterującym PWM, doszliśmy jednak do wniosku, że taki głośnik nie nada się do odtwarzania dźwięków mp3.

Kamerka internetowa wykorzystana do robienia zdjęć tekstu to kamerka Tracer otrzymana przez nas od prowadzącego zajęcia Systemy Wbudowane. Do projektu wykorzystane zostały trzy kamerki, jednak ta dawała najostrzejsze zdjęcia, z których OCR bezproblemowo odczytywał tekst. Dwie poprzednie kamerki miały problem z oświetleniem kartki – miały one o wiele niższą rozdzielczość wyświetlania.

Wyświetlacz wykorzystany do wyświetlania tekstu to 1.8inch LCD Module ST7735S. Pierwotnie mieliśmy wykorzystać wyświetlacz LCD I2C 2x16, jednak problemy, jakie ze sobą niósł, wykluczyły go z projektu. Głównymi problemami była niewielka ilość tekstu mieszcząca się na wyświetlaczu oraz brak możliwości wyświetlania innych znaków niż znaki alfabetu łacińskiego oraz znaki interpunkcyjne. Taki wyświetlacz nie nadawał się do projektu, w którym korzysta się z języków zawierających znaki inne niż łacińskie, jak np. cyrylica.

Językiem programistycznym wykorzystanym do stworzenia projektu jest Python3. Python3 sprawdza się dzięki czytelności kodu, prostocie pisania oraz wykorzystaniu wielu sprawdzonych bibliotek.

2. Schemat.



3. Założenia projektowe a ich realizacja.

- Co chcieliśmy zrobić (założenia):

System tłumaczenia tekstu z użyciem kamerki internetowej, wyborem języka poprzez trzy dostępne przyciski, wyświetlanie każdego z kroków na wyświetlaczu i odtwarzanie przeczytanego tekstu za pośrednictwem głośnika

- Co udało się zrobić:

System tłumaczenia tekstu z użyciem kamerki internetowej, możliwością akceptacji lub odrzucenia zdjęcia poprzez dwa dostępne przyciski, wyborem języka poprzez trzy dostępne przyciski, wyświetlanie każdego z kroków na wyświetlaczu i odtwarzanie przeczytanego tekstu za pośrednictwem głośnika.

- Czego nie udało się zrobić (i dlaczego):

Przy zmianie wyświetlacza odpuściliśmy pomysł przewijania tekstu. Nowo wykorzystany wyświetlacz jest na tyle duży, aby wyświetlać tekst składający się do paru zdań.

- Możliwości rozwoju zrealizowanego projektu:

Zbudowanie gotowej „stacji” z odpowiednim oświetleniem pod którą można podsuwać kartki i system będzie czytał całe strony tekstu i składował je w bazie danych tak, aby był stały dostęp do przetłumaczonych plików.

Ulepszenie kamerki internetowej na kamerę, która będzie obsługiwała większe objętości tekstu.

Wyświetlać tłumaczony tekst „dynamicznie” – w tym samym momencie.

Dodanie szerszej możliwości wyborów języków.

Tłumaczenie tekstu na kilka języków na raz.

Użycie większego wyświetlacza, aby była możliwość wyświetlania dłuższych tekstów.

4. Listing kodu.

*tutaj umieść najważniejsze fragmenty kodu **wraz z komentarzami**.*

```
# Importowanie wymaganych bibliotek
from gpiozero import Button  # Obsługa przycisków GPIO na Raspberry Pi
import cv2  # Obsługa kamery i obrazów
import pytesseract  # OCR - rozpoznawanie tekstu z obrazu
from dotenv import load_dotenv  # Ładowanie zmiennych środowiskowych
import os
import sys
import requests  # Wysyłanie zapytań HTTP
from elevenlabs import ElevenLabs  # API ElevenLabs do syntezowania mowy
import pygame  # Obsługa dźwięku
import time
import re
import logging
import spidev as SPI
sys.path.append("..")
from lib import LCD_1inch8  # Biblioteka do obsługi wyświetlacza LCD
from PIL import Image, ImageDraw, ImageFont  # Obsługa obrazu i tekstu
import textwrap  # Zawijanie tekstu

# Konfiguracja wyświetlacza
RST = 27
DC = 25
BL = 18
bus = 0
device = 0
logging.basicConfig(level=logging.DEBUG)

disp = LCD_1inch8.LCD_1inch8(spi=SPI.SpiDev(bus, device),
spi_freq=10000000, rst=RST, dc=DC, bl=BL)

disp.Init()
disp.clear()
disp.bl_DutyCycle(50)

#Funkcja rysująca tekst na wyświetlaczu
def display_text_to_lcd(text):
    try:
        disp.clear()

        image1 = Image.new("RGB", (disp.width, disp.height), "BLUE")
        draw = ImageDraw.Draw(image1)
```

```
        Font2 = ImageFont.truetype("../Font/Font00.ttf", 10)

        max_chars_per_line = 30
        line_height = 14

        manual_lines = text.split('\n')

        wrapped_lines = []
        for line in manual_lines:
            if line.strip() == "":
                wrapped_lines.append("")
            else:
                wrapped_part = textwrap.wrap(
                    line,
                    width=max_chars_per_line,
                    break_long_words=False,
                    break_on_hyphens=False
                )
                if not wrapped_part:
                    wrapped_lines.append("")
                else:
                    wrapped_lines.extend(wrapped_part)

        logging.info("Drawing text with forced and auto line-breaks")

        for i, line in enumerate(wrapped_lines):
            draw.text((1, 1 + i * line_height), line, font=Font2,
fill="WHITE")

        disp.ShowImage(image1)
        time.sleep(2)

        logging.info("Display finished successfully")

    except IOError as e:
        logging.error(f"IOError: {e}")
    except KeyboardInterrupt:
        logging.info("Exiting...")
        exit()

#Inicjalizacja miksera pygame dla głośnika
pygame.mixer.init()

#Konfiguracja przycisków
GPIO_PIN_CZERWONY = 9
GPIO_PIN_ZIELONY = 22
GPIO_PIN_1 = 26
GPIO_PIN_2 = 17
GPIO_PIN_3 = 23

button_czerwony = Button(GPIO_PIN_CZERWONY, pull_up=True)
button_ziolony = Button(GPIO_PIN_ZIELONY, pull_up=True)
button_1 = Button(GPIO_PIN_1, pull_up=True)
button_2 = Button(GPIO_PIN_2, pull_up=True)
button_3 = Button(GPIO_PIN_3, pull_up=True)

# Ładowanie zmiennych środowiskowych
load_dotenv()
```

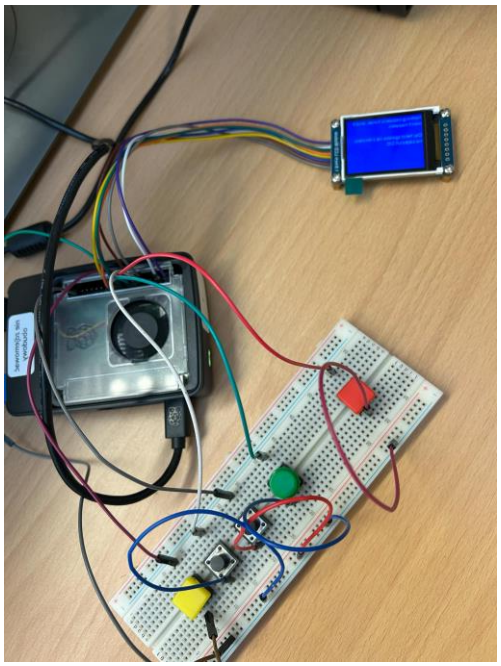
```
deepl_api_key = os.getenv('DEEPL_API_KEY')
elevenlabs_api_key = os.getenv('ELEVENLABS_API_KEY')

voice_ids = {
    'en': os.getenv("ENGLISH_ID"),
    'pl': os.getenv("POLISH_ID"),
    'ru': os.getenv("RUSSIAN_ID"),
    'de': os.getenv("GERMAN_ID")
}

#Robienie zdjęcia za pomocą kamery.
def capture_image():
    cam = cv2.VideoCapture(0)  # Kamera podłączona do Raspberry Pi
    display_text_to_lcd("Przechwytywanie zdjęcia...")
    ret, frame = cam.read()
    if ret:
        cv2.imwrite("captured.jpg", frame)
        display_text_to_lcd("Zdjęcie zapisane!")

    else:
        display_text_to_lcd("Brak zdjęcia!")
    cam.release()


#Konwersja obrazu na tekst za pomocą OCR.
def image_to_text():
    myconf = r"-l pol+eng+deu+rus"
    display_text_to_lcd("Wyciąganie tekstu ze zdjęcia...")
    text = pytesseract.image_to_string(Image.open("captured.jpg"),
config=myconf)
    display_text_to_lcd("Tekst wyciągnięty!")
    text = re.sub(r'[^\x20-\x7E]+', '', text)
    text = text.strip()
    print(f"Extracted text: {text}")
    return text


#Tłumaczenie tekstu za pomocą API DeepL.
def translate_text(text, target_language):
    display_text_to_lcd("Tłumaczenie tekstu...")
    url = "https://api-free.deepl.com/v2/translate"
    params = {
        'auth_key': deepl_api_key,
        'text': text,
        'target_lang': target_language
    }
    response = requests.post(url, data=params)
    if response.status_code == 200:
        translation_result = response.json()['translations'][0]
        translated_text = translation_result['text']
        detected_source_language =
translation_result['detected_source_language']
        return translated_text, detected_source_language
    else:
        raise Exception(f"Translation error: {response.status_code}")


#Syntezowanie mowy z tekstu za pomocą ElevenLabs.
```

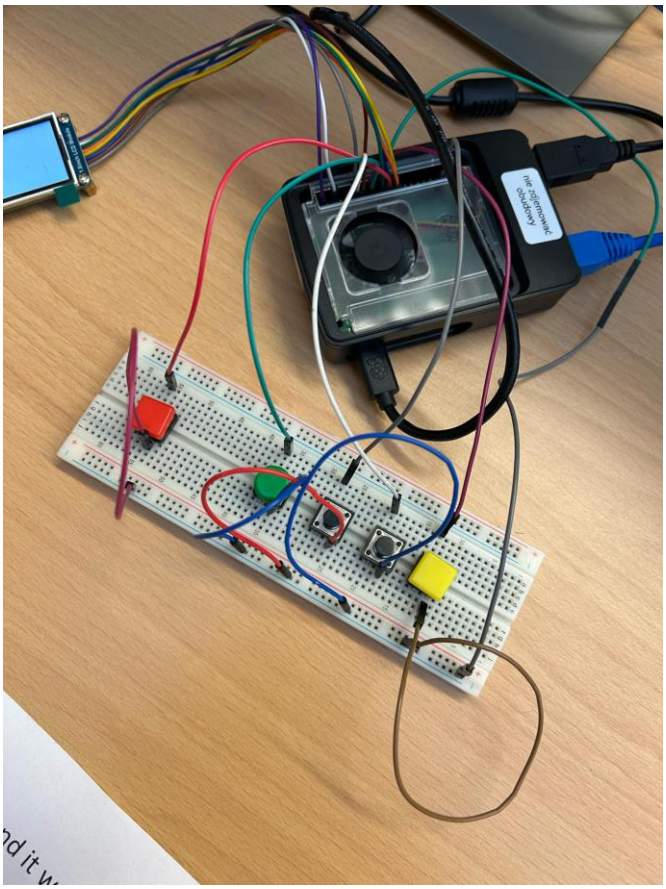
```python
def text_to_speech(text, language, elevenlabs_api_key, voice_ids):
    if not text:
        raise ValueError("Text cannot be empty or None")

    text = str(text)
    print(f"Text type: {type(text)}")

    display_text_to_lcd("Generowanie audio...")

    voice_id = voice_ids.get(language)
    if not voice_id:
        raise ValueError(f"Voice ID for language '{language}' not
found!")

    print(f"Using voice ID: {voice_id}")

    url = f"https://api.elevenlabs.io/v1/text-to-speech/{voice_id}"
    headers = {
        "Accept": "audio/mpeg",
        "Content-Type": "application/json",
        "xi-api-key": elevenlabs_api_key
    }

    data = {
        "text": text,
        "model_id": "eleven_multilingual_v2",
        "voice_settings": {
            "stability": 0.75,
            "similarity_boost": 0.85
        }
    }

    response = requests.post(url, json=data, headers=headers)
    print(f"Response status code: {response.status_code}")

    if response.status_code == 200:
        with open('output.mp3', 'wb') as f:
            for chunk in response.iter_content(chunk_size=1024):
                if chunk:
                    f.write(chunk)
    else:
        print(f"Error: {response.status_code}, {response.text}")
        raise Exception(f"Error: {response.status_code} -
{response.text}")


#Odtwarzanie pliku mp3 przez głośnik
def play_audio(audio: str):
    mp3_file = audio
    pygame.mixer.music.load(mp3_file)

    pygame.mixer.music.play()

    while pygame.mixer.music.get_busy():
        time.sleep(1)


#Detekcja języka tekstu.
def detect_language(text):
```

```
    display_text_to_lcd("Szukanie języka tekstu...")
    _, detected_language = translate_text(text, 'EN')
    return detected_language


available_languages = {
    'PL': {'ENG': 'en', 'RU': 'ru', 'DE': 'de'},
    'EN': {'PL': 'pl', 'RU': 'ru', 'DE': 'de'},
    'RU': {'PL': 'pl', 'ENG': 'en', 'DE': 'de'},
    'DE': {'PL': 'pl', 'ENG': 'en', 'RU': 'ru'}
}


def main():
    display_text_to_lcd("Start")
    button_ziolony_pressed = False
    button_czerwony_pressed = False

    while True:
        #1. Kamerka robi zdjęcie
        capture_image()
        #2. Ze zdjęcia wyciągamy tekst
        input_text = image_to_text()
        #3. Tekst przedstawiamy użytkownikowi dla weryfikacji, czy
zdjęcie zostało dobrze wykonane, a tekst dobrze wyciągnięty przez OCR
        display_text_to_lcd(f"{input_text}")

        while True:
            time.sleep(0.1)

            #3.1 Jeżeli zdjęcie jest poprawne, a tekst odwzorowuje to co
jest na kartce, to możemy przejść do kolejnych kroków programu
            if button_ziolony.is_pressed and not button_ziolony_pressed:
                print("Zdjęcie zaakceptowane!")
                button_ziolony_pressed = True
                break

            #3.2 Jeżeli zdjęcie jest niepoprawne, a tekst nie odwzorowuje
tego co jest na kartce, to ponawiamy robienie zdjęcia
            if button_czerwony.is_pressed and not
button_czerwony_pressed:
                print("Zdjęcie odrzucone. Powtarzamy.")
                time.sleep(0.5)
                button_czerwony_pressed = True
                break

        # Sprawdzamy, czy zdjęcie zostało zaakceptowane
        if button_ziolony.is_pressed:
            break  # Przerywamy główną pętlę po akceptacji zdjęcia

    #4. Szukamy języka, w którym napisany jest tekst
    detected_language = detect_language(input_text).upper()
    print(f"Detected language: {detected_language}")
    display_text_to_lcd(f"Język tekstu: {detected_language}")

    if detected_language not in available_languages:
        display_text_to_lcd("Język nie jest wspierany przez program.")
        sys.exit(1)
```

```
    possible_languages =
list(available_languages[detected_language].keys())

    #5. Dajemy użytkownikowi możliwość przetłumaczenia tekstu na jeden z
trzech możliwych wyborów
    display_text_to_lcd(f"Na jaki język chcesz przetłumaczyć
tekst?\n\n1.{possible_languages[0]} 2.{possible_languages[1]}
3.{possible_languages[2]}")
    while True:
        if button_1.is_pressed:
            target_language_code = possible_languages[0]
            break

        if button_2.is_pressed:
            target_language_code = possible_languages[1]
            break

        if button_3.is_pressed:
            target_language_code = possible_languages[2]
            break

        time.sleep(0.1)

    target_language =
available_languages[detected_language].get(target_language_code)

    #6. Tekst jest tłumaczony na wybrany język
    translated_text = translate_text(input_text, target_language)

    #7. Tekst jest zamieniany na synteze mowy
    text_to_speech(translated_text[0], target_language,
elevenlabs_api_key, voice_ids)

    #8. Przetłumaczony tekst jest wyświetlany wraz z odtwarzaniem się na
głośniku syntezy mowy.
    display_text_to_lcd(translated_text[0])
    play_audio("output.mp3")
    time.sleep(10)

if __name__ == "__main__":
    main()
```

5. Zdjęcia zrealizowanego układu.





6. Podsumowanie i wnioski.

Projekt ten nauczył nas pracy z systemem wbudowanym Raspberry Pi 5, jak i wykorzystania i instalacji dostępnych bibliotek oraz API do stworzenia integralnego systemu, który sprawdza się w swoich założeniach. Nauczył nas również pracy w zespole oraz przypomniał o zachowaniu spokoju w momentach kryzysowych.

Wykonany projekt zgadza się z naszymi początkowymi założeniami.

Projekty o dużej skali należy zaczynać najszybciej, jak to możliwe, ze względu na czasochłonność zadania, presję czasu oraz możliwość testowania i wymiany podzespołów, gdy jest to konieczne.

Po wielu próbach konfiguracji przeróżnych sprzętów doszliśmy do wniosku, że przed użyciem danej biblioteki trzeba sprawdzić kilka razy, czy na pewno jest ona kompatybilna z urządzeniem Raspberry Pi, ponieważ ten błąd zmusił nas do wykonywania kilkuset testów, które nie przynosiły oczekiwanego rezultatu.

Podobnie sprawa tyczy się wykorzystanego sprzętu, warto przyjrzeć się dokumentacji danego sprzętu przed wykorzystaniem go w projekcie, gdyż pozwoli to na zaoszczędzeniu czasu przy testowaniu i próbie implementacji niekompatybilnego komponentu.

Najważniejszym dla nas wnioskiem jest to, że nie wszystko da się przewidzieć i trzeba się liczyć z nagłymi zmianami. W fazie „burzy mózgów” niektóre rzeczy i rozwiązania mogą wydawać się trywialne. Trzeba je jednak brać pod uwagę, ponieważ może dojść do sytuacji, gdzie ta pozornie nieistotna kwestia okaże się dużym problemem.