Code Documentation

1 Pinecone Code

Project folder: ./code/bl602_demo_wifi
Source code: ./code/bl602_demo_wifi/bl602_demo_wifi/main.c
Project is based on bl_iot_sdk/customer_app/bl602_demo_wifi

Function descriptions

bfl_main

- main entry method
- initializes the wifi connectivity
- starts our main task proc_hellow_entry

proc_hellow_entry

- initializes UART connection to arduino and for debugging
- initializes LED pins
- ARDUINO_TRIGGER_PIN == this pin forwards the signal from the connected button to the arduino. Button is not connected directly to the arduino as we needed to use the pinecone for all components.
- initialization of servo PWM and WIFI connection
- forwards button signal to arduino, arduino records until button is released
- waits for arduino to send data transfer start sequence (4x ‘\r’)
- receives file size in bytes of WAV file as character sequence (content-length)
- constructs http header with the content-length header field
- http packages are sent to a remote port forwarding server, which forwards the packages to our local notebook (to have a static hard coded IP in changing WIFI networks)
- receives the file, byte by byte and stores it into a buffer. Transfer complete if no byte is received for at least a second.
- function client_demo is called to submit the http request with the buffer (containing the http header and file data) and the the amount of bytes which are used in the buffer

client_demo

- submits the http request (sends the recorded WAV file to the server) using the write() method
- read the server response using the blocking read() method and the receive buffer “recv_buffer”
- search for the value after the sequence “res:” to receive the boolean server response e.g. “res:0” for a failed authentication
- open and close the the door using the servo motor on successful authentication

2 Arduino Code

Source code: ./code/arduino/arduino.ino

- initialisation of TMRpcm library for audio recording and storing the audio on the sd card
- constantly wait for the button to be pressed (signal from pinecones' GPIO pin)
- record audio while button is pressed, audio is stored as a WAV file on the SD card
- when button is released:
- get the file size in bytes as an integer, convert the integer into a character sequence/array
- send the data transfer start command (4x '\r')
- transfer the file size as a character sequence (bytes)
- send the file byte by byte
- wait for the button to be pressed again

3 Server Code

Source code: ./server/server.py

- Starts a simple python server using FastAPI
- uses nvidia's nemo ai model for speaker verification
- handles HTTP POST requests on server path '/auth'
- runs on localhost port 800
- local port is forwarded using the scripts ./server/lt.py or ./server/serveo.py (alternative services) to a static and public IP address
    - e.g. a POST request on ourvoiceproject1.serveo.net/auth is now forwarded to our local server on localhost:8000/auth

Function auth()

- file data is read from the request (audio WAV file) and written into the file ./test.wav
- the audio is transcribed with the nvidia nemo model
    - if the text is different from the passphrase "Open the door", the boolean result 0 is returned to the client
- Using another nemo model, the n-dimensional feature vector (pytorch tensor) is extracted from the incoming audio
- another feature vector is extracted for every audio sample from the 2 enrolled users Sven and Sayed
    - the incoming audio is compared to every stored audio sample and the cosine similarity is calculated between both vectors
    - if a similarity of at least 0.6 (60%) is achieved, the speaker is considered to be the same speaker as in the stored audio sample.