MINISTRY OF SCIENCE AND HIGHER EDUCATION OF THE RUSSIAN FEDERATION

Federal State Autonomous Educational Institution of Higher Education

“South Ural State University (National Research University)”

School of Electronic Engineering and Computer Science

Department of Computer Engineering

|  |  |
| --- | --- |
| THESIS IS CHECKED  Reviewer,  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_ \_. \_. \_\_\_\_\_\_\_\_\_  “\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. | ACCEPTED FOR THE DEFENSE  Head of the department,  Ph.D., Associate Professor  \_\_\_\_\_\_\_\_\_\_\_\_\_ D.V. Topolsky  “\_\_\_”\_\_\_\_\_\_\_\_\_\_ 2022 г. |

Methodical instructions for the implementation of graduate qualification works

GRADUATE QUALIFICATION WORK

SUSU – 09.04.01.2022.308-643.GQW

|  |  |
| --- | --- |
|  | Supervisor,  PhD, Associate Professor  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ D.V. Topolsky  “\_\_\_”\_\_\_\_\_\_\_\_\_\_\_2022 г.  Author,  student of the group: CE-228  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_. \_. \_\_\_\_\_\_\_\_\_  “\_\_\_”\_\_\_\_\_\_\_\_\_\_\_2022 г.  Normative control  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_. \_. \_\_\_\_\_\_\_\_\_  “\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_2022 г. |

Chelyabinsk-2022

MINISTRY OF SCIENCE AND HIGHER EDUCATION OF THE RUSSIAN FEDERATION

Federal State Autonomous Educational Institution of Higher Education

“South Ural State University (National Research University)”

School of Electronic Engineering and Computer Science

Department of Computer Engineering

|  |  |
| --- | --- |
|  | ACCEPTED FOR THE DEFENSE  Head of the department,  Ph.D., Associate Professor  \_\_\_\_\_\_\_\_\_\_\_\_\_ D.V. Topolsky  “\_\_\_”\_\_\_\_\_\_\_\_\_\_ 2022 г. |

**TASK**

**of the master graduate qualification work**

for the student of the group CE-228

First Name Last Name

in master direction 09.04.01

“Fundamental Informatics and Information Technologies”   
(master program “Internet of Things”)

1. **The topic** (approved by the order of the rector from \_\_.\_\_.2022): “Methodical instructions for the implementation of graduate qualification works”.
2. **The deadline for the completion of the work:** 01.06.2022.
3. **The source data for the work:**

*Example #1.*

* 1. GNU/Linux.
  2. Speech commands dataset. [Electronic Resource] URL: https://arxiv.org/abs/1804.03209.
  3. Detexify data base (http://detexify.kirelabs.org), MNIST base.
  4. LaTeX, MathCad® syntaxis, MathML, Wolfram Mathematica®.

*Example #2. Digital Instrument Transformer*

* 1. Voltage class 110 kV.
  2. Rated primary current 200...2000 A.
  3. Current measurement accuracy class for ACAS 0.2.
  4. Current measurement accuracy class for RPA 5.
  5. Voltage measurement accuracy class 0.2.
  6. Communication protocol IEC 61850-9-2.
  7. Dimensions of sensor 1540x320x450 mm.
  8. Weight 120 kg.

1. **The list of the development issues:**

*Example #1.*

* 1. To develop a Library that allows real-time classification of raw audio.
  2. To develop a library that runs on devices that consume low power and have low processing capabilities;
  3. To test the library and give an example of how to implement interfaces.
  4. To deploy the library in a version control environment.
  5. To deploy the library in at least one official Library managers for MCU hardware vendors.

*Example #2.*

* 1. Describe of the subject area.
  2. Design the Entity Relations diagram for the police man.
  3. Declare functional and non-functional requirements.
  4. Design use case diagram for the system.
  5. Design a database schema.
  6. Implementation of the application.
  7. Testing of the application.

*Example #3.*

* 1. Review existing software methods for sticker detection.
  2. Design a sticker detection system, including the design of a convolutional neural network.
  3. Implement a sticker detection system, prepare training data, and train the designed CNN.

1. **Issuance date of the task:** \_\_.12.2021.

Supervisor \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ *D.V. Topolsky* /

Student \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/*\_. \_. \_\_\_\_\_\_\_\_\_* /

CALENDAR PLAN

| Phase | Deadline | Supervisor’s signature |
| --- | --- | --- |
| Introduction and literature review | 10.03.2022 |  |
| Development of the model, design of the system | 21.03.2022 |  |
| Implementation of a system | 04.04.2022 |  |
| Testing and debugging of the system, experiments | 25.04.2022 |  |
| Full text, normative control | 16.05.2022 |  |
| Proposal defense | 24.05.2022 |  |

Supervisor \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ *D.V. Topolsky* /

Student \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/*\_. \_. \_\_\_\_\_\_\_\_\_* /

Annotation

|  |  |
| --- | --- |
|  | \_. \_. \_\_\_\_\_\_\_\_\_. Methodical instructions for the implementation of graduate qualification works. – Chelyabinsk: SUSU; 2022, 42 p., 8 pic., bibl. – 18. |

Annotation is a brief description of the work in terms of content, purpose, form and other features.

Annotation includes:

* description of the main topic;
* object problems;
* goals (and objectives) of the work;
* results of work;
* the novelty of the work in comparison with others, related in subject matter and purpose.

Annotation performs the following functions:

* make it possible to establish the main content of the document, determine its relevance and decide whether to refer to the full text of the document;
* provide information about the document and eliminate the need to read the full text of the document in case the document is of secondary interest to the reader;
* are used in information systems, including automated systems for searching documents and information.

The recommended average size of an abstract is 500 characters.

*Example.*

This thesis consists of five main chapters: Introduction, definition of requirements, design and Implementation, deployment and testing, conclusion, and references.

In the first chapter, we will have the subject area analysis briefly then have an overview of analogues and the main technological solutions that I will use will be featured. All the different software platforms to be used will be adequately described.

In the second chapter, there is a description of both functional and non-functional requirements as well as a description of how the various software components will interact with each other. We then finish by describing how documentation will be done for the device.

In the third chapter we describe the design and implementation of the software and how the different components will interact with each other as well as the algorithms for tackling the problem and a description of the data.

In the fourth chapter we will do deployment and testing for both the software accuracy and the hardware performance and interfacing with a real MCU. For this practice we will use an Arduino MCU.

Finally in the fifth chapter we will have a conclusion for the thesis, with future improvements to the solution being discussed as well as opportunities.

**CONTENTS**

[INTRODUCTION 9](#_Toc89692350)

[1. SUBJECT AREA ANALYSIS 12](#_Toc89692351)

[1.1. OVERVIEW OF ANALOGUES 14](#_Toc89692352)

[1.2. ANALYSIS OF THE MAIN TECHNOLOGICAL SOLUTIONS 19](#_Toc89692353)

[1.3. CONCLUSION 23](#_Toc89692354)

[2. DEFINITION OF REQUIREMENTS 24](#_Toc89692355)

[2.1. FUNCTIONAL REQUIREMENTS 24](#_Toc89692356)

[2.2. NON-FUNCTIONAL REQUIREMENTS 25](#_Toc89692357)

[3. DESIGN AND IMPLEMENTATION 28](#_Toc89692358)

[3.1. ARCHITECTURE OF THE PROPOSED SOLUTION 28](#_Toc89692359)

[3.2. ALGORITHMS FOR SOLVING THE PROBLEM 30](#_Toc89692360)

[3.3. DESCRIPTION OF DATA 31](#_Toc89692361)

[4. IMPLEMENTATION 33](#_Toc89692362)

[4.1. IMPLEMENTATION OF INTERFACES 33](#_Toc89692363)

[5. TESTING 34](#_Toc89692364)

[5.1. TESTING METHODOLOGY 34](#_Toc89692365)

[5.2. TEST PROCEDURE 34](#_Toc89692366)

[6. CONCLUSION 37](#_Toc89692367)

[REFERENCES 41](#_Toc89692368)

[APPENDIX A SOURCE CODE 42](#_Toc89692369)

# INTRODUCTION

The *introduction* reflects the following main points:

1. general formulation of the topic;
2. the relevance of the chosen topic, its theoretical and / or practical significance;
3. the degree of scientific elaboration of the research topic;
4. the purpose and objectives of the study;
5. an explanation of how, with the help of what research methods and in what order the author intends to solve the assigned tasks;
6. announcement of the structure of the work (titles of chapters of the work and their brief description);
7. characteristics of the main sources of information.

The introduction should be short (usually up to 3 pages) and clear. From the introduction it should be clear what the work is devoted to (the purpose of the work), what tasks and with the help of what methods it is solved, what results should be achieved.

A *research problem* is a theoretical or practical issue requiring research study and solution by means of science. Formulation of a scientific problem is a creative act that requires a special vision, special knowledge, experience, and scientific qualifications. The solution to the problem is usually the goal of following.

The *topic* is a laconic formulation of the problem. The most convincing basis defining the research topic is a contradiction in social practice, management practice, reflecting the most acute, socially significant issues that require urgent solutions.

The criterion of *relevance* indicates the need and timeliness of studying and solving the indicated problem. Actual research provides an answer to the most pressing questions at this time, reflecting the social order of society, business to modern science, point to the most important contradictions that take place in practice. The criterion of relevance is dynamic, flexible, depends on time and consideration of specific and specific circumstances. In its most general form, relevance is characterized by the degree of discrepancy between the demand for scientific ideas and practical recommendations (to meet a particular need) and the proposals that science and practice can provide at the present time.

*Example.*

Speech recognition, also known as automated speech recognition, machine speech recognition, or speech-to-text, is a feature that allows a program to convert human speech into written text. Although voice recognition and speech recognition are often mixed, speech recognition focuses on the conversion of speech from a spoken format to a text format, whereas voice recognition only attempts to recognize a particular user's voice.

As embedded device computing becomes more prevalent, it presents new and creative ways for humans and machines to cooperate, connect, and interact. Though data has received most of the attention in embedded systems, the incorporation of speech into embedded computer applications provides a flexible means of providing human interaction, connectivity, and control. Voice, when implemented properly and with respect for *relevance*, prices, and human factors, may offer a more versatile user interface at a lower cost than conventional approaches such as touch screens or data entry.

Analysis of *the degree of elaboration of the problem* demonstrates its elaboration and study in the relevant science and practice.

*The purpose of the research* is a mental anticipation of the result that will be obtained in the course of the research (as the researcher sees it). The goal is considered achieved if the leading idea reflected in the topic is formulated, substantiated, proven and tested in practice.

*Example.*

The purpose of the research:

To create a software complex to enable running of machine learning algorithms for speech recognition on small devices with limited power and computing capabilities.

*Research tasks* are ways to achieve a goal (what needs to be done to achieve the goal: study, describe, establish, identify, etc.). If a theoretical analysis of the literature is needed to solve a problem, then one of the tasks may be to identify the theoretical foundations of the problem, etc. It is necessary to formulate the tasks very carefully, since the description of their solution should constitute the content of the chapters and paragraphs of the study, and the assessment of the effectiveness of the study will depend on the description of their solution.

*Example.*

Tasks necessary to achieve the goal:

1. Analyzing the market for existing libraries.
2. Collect a keyword dataset to use to train a machine learning model for speech recognition.
3. Design a model architecture for MBEDSpeech.
4. Training the MBEDSpeech Model.
5. Testing and deploying the model to different embedded device Platforms in a way that can be used by other developers.
6. Integrating MCU code with deployed library as an interfacing example
7. Testing the integrated model.

*Research methods* are ways of solving research problems and obtaining a research result. Usually, theoretical methods are used (analysis, synthesis, comparison, generalization, modeling, etc.), empirical methods that ensure data collection (observation, study of products of activity, documentation, questionnaires, sociometry, conversation, method of independent characteristics, experiment, etc.). and mathematical methods (quantitative data processing, ranking, etc.).

*The novelty of the research* characterizes new theoretical and practical conclusions, patterns, content, principles and technologies that were not known and were not recorded in the literature at this point. The criterion of research novelty can have both theoretical and practical significance. The theoretical value of the research lies in creating a concept, describing a method, model, approach, concept, principle, etc. The practical significance of the study lies in its readiness for implementation in practice.

# 1. SUBJECT AREA ANALYSIS

This section describes the subject area, the statement of the problem is given in an expanded form in accordance with the customer's requirements, but formulated with the professional help of the student, existing analogues can be given and the software tools selected to achieve the goal can be characterized. All information provided here can be described in one section or broken down into subsections below.

The task can relate to a very complex subject area, with new concepts, with complex terminology, complex relationships between objects, therefore, it is necessary to describe this subject area, solved, unsolved or partially solved problems, a brief history of development, the contribution of predecessors, a description of concepts and terms, which the student knows.

The description of the subject area should touch upon a wider and more important range of problems than that of the problem solved in the final qualifying work. For example, if the problem of developing and creating a microcontroller with a software component is being solved, then it makes sense to describe the field of microelectronics.

*Example.*

An ultra-low-power embedded device is one that is inexpensive, runs on a few hundred kilobytes of RAM, has similar amounts of flash memory for persistent program and data storage, has a clock speed of just tens of megahertz, does not run a full operating system, and avoids using dynamic memory allocation functions like new or malloc() because they’re designed to be reliable and long-running, and it’s extremely difficult to ensure that if you have a heap that can be fragmented.

To take advantage of ultra-low-power embedded devices’ low power consumption, it is imperative that we merge the embedded software development with machine learning since data is the most important by-product of any system. One of the most anticipated problems that this combination of disciplines can solve through predicting things based on past observations is speech recognition.

To do speech recognition using deep learning, a programmer should feed data into learning algorithm that discovers the rules in the data, which then builds a model based on the data provided through a process called training and finally data is then run through this model to make predictions, a process called inference.

The past few years has seen products provide a voice user interface (UI) designed to give instant access to information without the need for a screen or keyboard. All these applications use speech recognition libraries in their development as will be discussed below. In most cases, speech recognition is done in the cloud, on powerful servers running large ML models bringing up privacy issues, efficiency, and speed due to latency and high-power consumptions sending a constant stream of data consumes a lot of energy.

This thesis will therefore focus on training a tiny model that listens for a wake words specified in the Speech Commands dataset and run it on a low-powered chip and does offline inference as well as give access to other embedded software developers to use it in various applications.

## 1.1. OVERVIEW OF ANALOGUES

In this section, it is necessary to describe the known analogues of the tools developed by the student. If there are many such works, then it is necessary to provide information on the most used and high-quality, at a given time, Software Systems or Software and Hardware Systems. Analogs may not be for the entire Software System or Software and Hardware System, but only for some of its parts, for example, only the technical part or software components. An example of a link to bibliographic sources: [1-4].

*Example.*

1.1.1. Kaldi NL

Kaldi is a speech recognition toolkit written in C++ intended for use by researchers whose main goal is to have modern and flexible code, written in C++, that is easy to modify and extend. Important features include:

* it can be used at code-level for integration as a library;
* it has extensive support for linear algebra;
* Kaldi provides algorithms in a way that can easily be extended in the most generic form possible;
* code for Kaldi is provided using an unrestrictive license: Apache 2.0 permitting and encouraging modifications and re-release;
* Kaldi's documentation is complete and highly accessible to everyone.

1.1.2. Speechmatics

Speechmatics is speech to text recognition software powered by machine learning with high accuracy. The software is available on both cloud and on-premise for users; it can also be embedded in devices. It uses a custom language build substructure i.e., automatic linguist, which lets the software learn new languages at a high pace.

Speechmatics boasts a high accuracy level that is further boosted by the custom dictionary feature. You can add new words to a language quickly. Speechmatic takes a new approach to recognize the English language, and it provides accurate speech recognition despite the user’s accent. Features of Speechmatics include:

* it gives a very accurate speech to text conversion;
* it constantly delivers a low word error rate across all the languages it is supporting; speechmatic provides frequent testing of the languages they are offering to keep a check on word error rate;
* it is an industry major in language coverage; Speechmatic is getting updated with new vocabulary to meet business-relevant needs.
* it can be installed with cloud services to avail its speech-to-text technology in real-time or, one can avail a pre-recorded (batch) files for on-premise installatio; Speechmatics provides adjustable deployment;
* it allows its users to add context-specific words to its dictionary; it enhances your transcription accuracy; this feature enables you to define the context of a conversation in advance; you can input variables like name, accents, abbreviations, acronyms, special, or industry-specific language, et cetera;
* it gives advanced punctuation, that is built over 2.5 billion words and holds an industry-leading set of supported punctuation marks; this optimizes the pace and ease of reading a transcript for human users;
* the software identifies a change of speaker within the user’s transcript; it adds a token automatically when the change in the speaker is noticed; this helps in easing the transcript modification for readability.

1.1.3. Google Speech API

Google Cloud Speech API is a programmatic interface to Google Cloud Platform services that allows a user to send speech recognition requests to Speech-to-Text in any programming language using the Google Cloud Client Libraries. It is a key part of Google Cloud Platform, allowing you to easily add the power of speech to everything from computing to networking to storage to machine-learning-based data analysis to your applications.

Speech-to-Text can process up to 1 minute of speech audio data sent in a synchronous request. After Speech-to-Text processes and recognizes all of the audio, it returns a response. A synchronous request is blocking, meaning that Speech-to-Text must return a response before processing the next request.

Speech-to-Text has three main methods to perform speech recognition. These are listed below:

* Synchronous Recognition which sends audio data to the Speech-to-Text API, performs recognition on that data, and returns results after all audio has been processed;
* Asynchronous Recognition which sends audio data to the Speech-to-Text API and initiates a Long Running Operation which can periodically be polled for recognition results;
* Streaming Recognition which performs recognition on audio data provided within a gRPC bi-directional stream. Streaming requests are designed for real-time recognition purposes, such as capturing live audio from a microphone. Streaming recognition provides interim results while audio is being captured, allowing result to appear, for example, while a user is still speaking.

After comparing the different transcribers Google Cloud Speech-to-Text, Speechmatics and Kaldi on Word Error Rate and hourly cost to see how they compare to each other, I found out what are their weaknesses are and it helped me to build up on my proposed solution.

First, when on comparing Google Cloud Speech-to-Text and Speechmatics, both these services are quite similar in their offering: cloud-based speech-to-text for many different languages with high performance. Both Google and Speechmatics continually update their language models to increase accuracy and introduce new words where applicable. This continuous development is a strong point for both, especially the introduction of new words which can help with new company names and other terms. Both services have a cost per minute of audio transcribed: Google uses a fixed price, the price for Speechmatics goes down as more minutes are purchased up front. Google has a few different transcription models available more than Speechmatics, however.

On the other hand, Kaldi which is an opensource speech recognition toolkit developed and maintained mainly by Daniel Povey with the help of about 70 other contributors so far has a lot of flexibility, especially since it’s open source and can be extended or improved by anyone who dares understand it. It does carry a lot of complexity however, requiring a lot of time and effort to fully learn the quirks.

Using Word Error Rate, which compares a reference with a hypothesis and is given by (Substitutions + Insertions + Deletions) / Number of words, where a substitution is when a word is replaced, an insertion is when a word is added that wasn’t said and a deletion is a word that is omitted, Google Speech-to-text scored 4.9%, Speechmatics 14.7% and Kaldi 8.01%.

I also compared them in terms of the cost of running for instance on an Amazon EC2 instance by dividing costs of the EC2 instance by the amount of audio it can transcribe per hour, for a c5.2xlarge instance it costs $0.384 per hour. There is an additional monthly fee for storage usage as well, however this does not significantly increase hourly costs.

With the Amazon instance running I also did some testing to determine how much audio could be transcribed per hour and determined that we could transcribe at a rate of 5 to 10% of the duration of the audio. I determined that it cost about $0.038 per hour of transcribed audio for Kaldi NL but $ 2.4 for Speechmatics and $ 1.4 for the Google Speech API. However, this does not include the cost of setup and maintenance!

Table 1 – Comparison of analogues

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Kaldi NL** | **Speechmatics** | **Google Speech API** |
| Language support | 60 Languages | 21 languages | 80 languages |
| Cost / min | $ 0.038 | $ 2.4 | $ 1.4 |
| Speaker detection | English (8Khz) | No | No |
| Audio Formats | FLAC, Siren, WAV, OGG, NULAW, Siren SR | FLAC, PCM, WAV, OGG, NULAW | FLAC, Linear16, MULAW, ARM, AMR\_W8 |
| Noise Friendly | Yes | No | Yes |
| Word Hints | Yes | No | No |
| Internet dependency | Yes | Yes | Yes |

## 1.2. ANALYSIS OF THE MAIN TECHNOLOGICAL SOLUTIONS

In this section, it is necessary to describe the known technical means, software, technologies that were chosen to solve this problem. Describe the advantages and disadvantages of the tools used and justify the need to use these particular software environments, tools and technologies. In particular, such advantages can be the low price of software or hardware or free software, the presence of a graphical interface, etc.

*Example.*

1.2.1. Speech Commands Dataset

The Speech Commands Dataset was created by the TensorFlow and AIY teams to showcase the speech recognition example using the TensorFlow API. The dataset has 65,000 clips of one-second-long duration. Each clip contains one of the 30 different words spoken by thousands of different subjects. This dataset will be combined with the Microsoft scalable Noisy Speech Dataset, which has a collection a variety of environmental noise files in .wav format sampled at 16khz. For the unknown category, I will use audio files with a collection of other words that are not considered in the dataset classes.

It has limited vocabulary but is still have enough variety for models trained on the data to potentially be useful for some applications. The dataset’s top ten common words as the core of our vocabulary that would be useful as commands in embedded systems or robotics applications; "Yes", "No", "Up", "Down", "Left", "Right", "On", "Off", "Stop", and "Go". I will be focusing my thesis on these.

In the second version of the dataset, there are four more command words; “Backward”, “Forward”, “Follow”, and “Learn”. One of the most challenging problems for keyword recognition is ignoring speech that does not contain triggers, so I also needed a set of words that could act as tests of that ability in the dataset. Some of these, such as “Tree”, were picked because they sound like target words and would be good tests of a model’s discernment. Others were chosen arbitrarily as short words that covered a lot of different phonemes. The final list was "Bed", "Bird", "Cat", "Dog", "Happy", "House", "Marvin", "Sheila", "Tree", and "Wow”. These extra words will be added in a future release of the library.

1.2.2. TensorFlow

TensorFlow is an open-source end-to-end platform for creating Machine Learning applications. It is a symbolic math library that uses dataflow and differentiable programming to perform various tasks focused on training and inference of deep neural networks. It allows developers to create machine learning applications using various tools, libraries, and community resources.

TensorFlow is the best library of all because it is built to be accessible for everyone. TensorFlow library incorporates different API to build at scale deep learning architecture like CNN or RNN. TensorFlow is based on graph computation; it allows the developer to visualize the construction of the neural network with Tensor board. This tool is helpful to debug the program. Finally, TensorFlow is built to be deployed at scale. It runs on CPU and GPU. To give a concrete example, Google users can experience a faster and more refined the search with AI. If the user types a keyword to the search bar, Google provides a recommendation about what could be the next word.

Some supported TensorFlow algorithms include:

* Linear regression;
* Classification;
* Deep learning classification;
* Deep learning wipe and deep;
* Booster tree regression;
* Boosted tree classification.

1.2.3. GitHub

GitHub provides combines the distributed version control and source code management features of Git with access control, issue tracking, feature requests, task management, continuous integration, and wikis for every project. These services are free of charge for individuals; however, the more complex professional and corporate services are commercial. It provides limitless private repositories to all plans, including free accounts, but only allows up to three contributors per repository. The following are some of the benefits of utilizing git:

* Git performs very strongly and reliably when compared to other version control systems; new code changes can be easily committed, version branches can be effortlessly compared and merged, and code can also be optimized to perform better; algorithms used in developing Git take the full advantage of the deep knowledge stored within, with regards to the attributes used to create real source code file trees, how files are modified over time and what kind of file access patterns are used to recall code files as and when needed by developers;
* Git is designed specially to maintain the integrity of source code; file contents as well as the relationship between file and directories, tags, commits, versions etc. are secured cryptographically using an algorithm called SHA1 which protects the code and change history against accidental as well as malicious damage; you can be sure to have an authentic content history for your source code with Git;
* Git offers support several kinds of nonlinear development workflows and its efficiency in handling both small scale and large scale projects as well as protocols; it is uniquely designed to support tagging and branching operations and store each and every activity carried out by the user as an integral part of “change” history; not all VCSs support this feature;
* Git offers the type of performance, functionality, security, and flexibility that most developers and teams need to develop their projects; when compared to other VCS Git is the most widely accepted system owing to its universally accepted usability and performance standards.
* Git is a widely supported open-source project with over ten years of operational history; people maintaining the project are very well matured and possess a long-term vision to meet the long-term needs of users by releasing staged upgrades at regular intervals of time to improve functionality as well as usability; quality of open-source software made available on Git is heavily scrutinized a countless number of times and businesses today depend heavily on Git code quality.

1.2.4. Google Colaboratory

Colaboratory, is a product from Google Research that allows anybody to write and execute arbitrary python code through the browser and is well suited to machine learning and data science. Colab is a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources including GPUs.

I will be using Colab to:

* Write and execute code in Python;
* Create/Upload/Share notebooks;
* Import/Save notebooks from/to Google Drive;
* Import/Publish notebooks from and to GitHub;
* Import external datasets e.g. from Kaggle;
* Integrate PyTorch, TensorFlow, Keras, OpenCV;
* Utilize free Cloud service with free GPU.

1.2.5. Programming Technologies

For this project we will use Python, C and C++.

Development that tools are required to develop and test/debug the code include:

* Compiler;
* Debugger.

## 1.3. CONCLUSION

Here the essence of the problem to be solved must be formulated clearly and clearly: is the problem a part of a complex large system or is it an autonomous problem. As a rule, the student's task is to develop and create a software tool or hardware with a software component for solving problems of physics, engineering, biology, genetics, geophysics, archeology, etc. The task in this case may consist of the following components:

* problems of the subject area, for example, the problem of creating a data structure for a more convenient and quick search for information, technical problems, etc.;
* tasks for the design and creation of the appropriate software.

*Example.*

The following components are being used for this project. It is urgent and under development right now:

* Google Speech Dataset;
* TensorFlow;
* Keras;
* GitHub;
* Google Colab.

# 2. DEFINITION OF REQUIREMENTS

When setting the task, the requirements for Software Systems or Software and Hardware Systems may be as follows:

* general or business rules;
* functional;
* non-functional (for example, quality and reliability, requirements for basic hardware and software).

*General requirements* (business rules) include requirements for the use of free tools, the use of specific or outdated hardware or software, increased requirements for the confidentiality of information use, etc.

## 2.1. FUNCTIONAL REQUIREMENTS

One of the most important points in setting a task is to define functional requirements, that is, the *functions required* by the user that need to be implemented to achieve the goal. Users can be divided into types (if necessary) and functions can be listed according to their purpose for each type. If we are talking about the requirements for Software and Hardware Systems, then these can be requirements, both for the software and for the technical component.

These requirements can be written in the form of short and clear sentences describing the functions of Software Systems or Software and Hardware Systems, or in the form of use case diagrams of the UML language.

*Example.*



Figure 1 – Use case diagram

## 2.2. NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements determine the conditions and environment for performing functions (for example, protection and access to the database, secrecy, etc.), they are not directly related to functions, but reflect user needs to perform functions. They describe the principles of interaction with environments or other systems, and also take into account uptime, data protection, as well as quality standards for achieving specific indicators or attributes of quality. These requirements reflect the needs of the customers of the system. For example, the requirements for the quality and reliability of Software Systems or Software and Hardware Systems can be formulated by the customer (manager).

It should be remembered that any additional requirement will affect the project of Software Systems or Software and Hardware Systems, costs and implementation time.

*Example.*

2.1.1. Speech Recognition

The MBEDSpeech recognition library allows 1 second word length of a 10 word vocabulary. It stores the "trained" word patterns used for recognition in internal memory. The main board has a charging port that can be hooked up to a battery to power the static ram when the main circuit is turned off. This keeps all the trained words safely stored in memory (SRAM) so the circuit does not have to be retrained every time it is turned on.

2.1.2. Continuous Recognition Style

It is the natural conversational speech people are accustomed to in everyday life. It is extremely difficult for a recognizer to shift through the text as the words tend to merge together. Isolated speech recognition system is another feature, of the IC that is used by MBEDSpeech.

2.1.3. Internet independence

The machine learning model used by MBEDSpeech should run on the MCU offline.

2.1.4. Noise

MBEDSpeech should be able to work well in noisy environments by automatically recognizing and filtering out the noise.

2.1.5. Processor power

MBEDSpeech should run on an edge device with a 32-bit ARM Cortex-M4F microprocessor running at 64MHz with 1MB of program memory and 256KB RAM.

2.2. Core Requirements

2.2.1. MBEDSpeech shall be able to classify audio in Realtime.

2.2.2. MBEDSpeech shall be compatible with edge Devices from Arduino, STM Electronics and NVDIA

2.3. Speech Requirements

The following guidelines are specific to speech:

2.3.1. MBEDSpeech SHALL use only approved keywords words found in the Google Speech Commands Dataset.

2.3.2. MBEDSpeech SHALL provide serial output to indicate when the Microphone Off.

2.3.3. MBEDSpeech shall recognize words in English only for the scope of this thesis

2.3.4. The audio used for MBEDSpeech training shall be 16KHz

2.4. Documentation Requirements

Documentation of this library will be delivered in various formats including pdf, ppt, html and docx and will be English language. This documentation will be the user manual for “MBEDSpeech” outlining its various features and how to use the Library for microcontroller development. It will contain all the options for deploying the service on various MCUs.

Error codes will also be explained and several solutions or how to avoid or solve them will also be outlined.

2.5 Conclusion

In this chapter, we defined the various core and speech requirements for MBEDSpeech. We also described how the various software components within it will interact with each other. Lastly, we gave the documentation requirements that described among other things the user manual and its contents.

# 3. DESIGN AND IMPLEMENTATION

In this section, it is necessary to describe the methods for the implementation of Software Systems or Software and Hardware Systems (with diagrams, in particular, a structural, functional, schematic diagram, a diagram of connections, connections, general, location), the mathematical algorithms used for solving the problem (if any) and the research part, if it is is necessary to solve the problem. All information can be described in one section, or broken down, for example, into the subsections below.

## 3.1. ARCHITECTURE OF THE PROPOSED SOLUTION

This section can fully or partially describe the structure of Software System or Software and Hardware System, i.e.:

1. what functional blocks (files, modules, procedures, functions, classes) Software System or Software and Hardware System consists of;
2. a description of each block with its name and purpose;
3. a graphical diagram of the interconnection of these blocks.

UML diagrams can be used to describe the operation of a software tool.

*Example.*

The MBEDSpeech Library is composed of various components, as shown in the image below. First, the microphone receives audio input then using MFCC (Mel frequency cepstral coefficients) extracts features and at the same time reducing the magnitude of the speech signal devoid of causing any damage to the power of speech signa. The inference is then run on the features outputting class probabilities as shown in the diagram below.

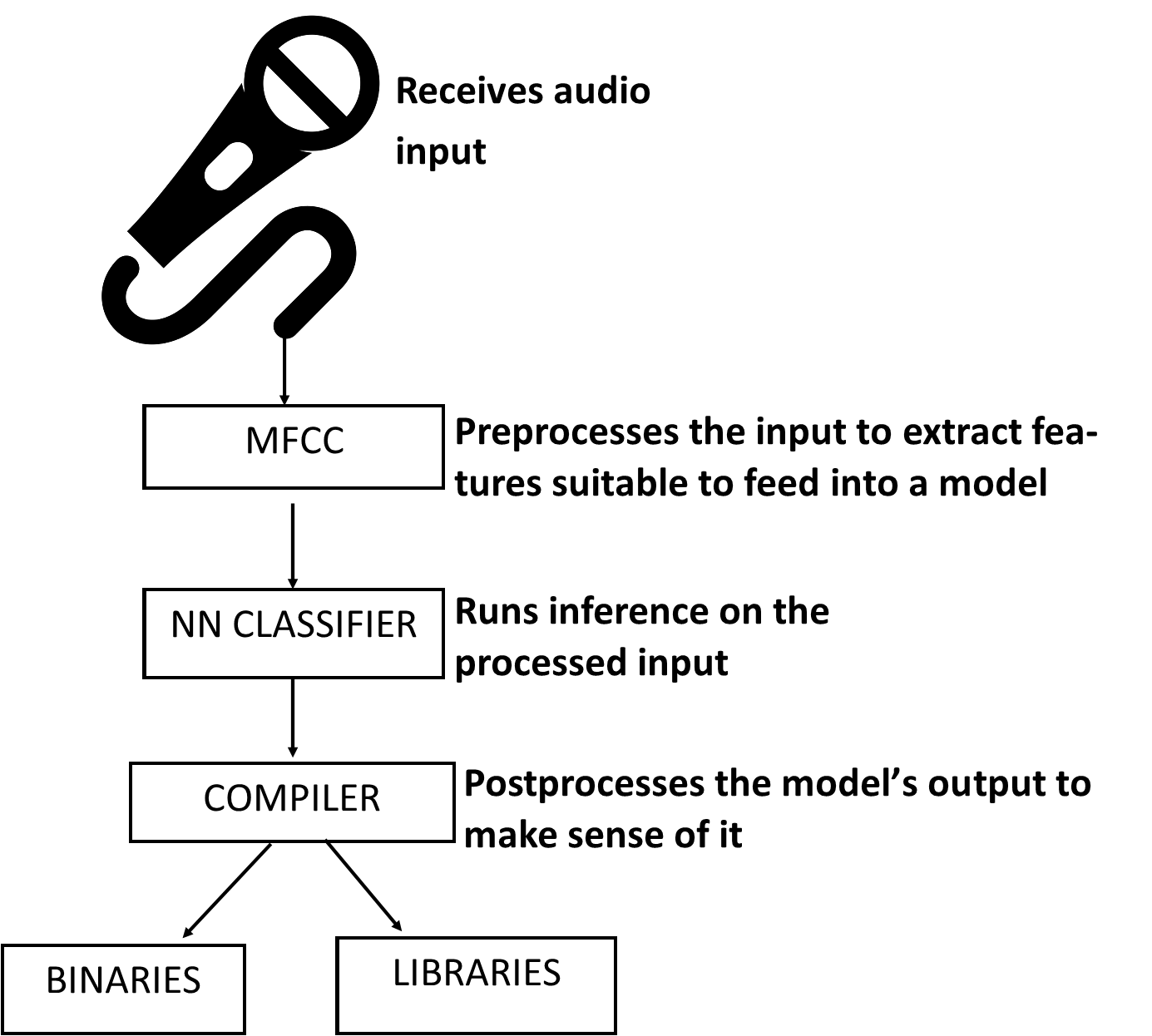


Figure 2 – MBEDSpeech Architecture

## 3.2. ALGORITHMS FOR SOLVING THE PROBLEM

*Research part.* The research part consists in studying and comparing the available methods for solving the problem, what advantages and disadvantages are in each of the methods, how to use the advantages and avoid the disadvantages in the student's proposed method for solving the problem.

*Mathematical algorithms.* In this subsection, it is necessary to describe the used mathematical algorithms for solving the problem with formulas and with links to sources.

*Example.*

While computing the word recognition rate (WRR) word error rate (WER) is used and the formula is (1)

|  |  |
| --- | --- |
|  | (1) |

where *H* is the number of correctly recognized words;

*I* is the number of substitutions;

*N* is the number of word references.

*Algorithms for the implementation of Software System or Software and Hardware System.* To describe the algorithm for the implementation of Software Systems or Software and Hardware Systems, two ways of describing algorithms can be used:

* verbal: in the form of a sequence of implementation steps with a description of classes, interfaces, procedures (it is possible with a program code that implements the most interesting algorithms), and if this is a technical part, then in addition to a verbal description, diagrams of technical devices can be given;
* graphic: in the form of block diagrams, necessarily with explanations.

## 3.3. DESCRIPTION OF DATA

This section describes the structure of the input, output and intermediate data. For example, the input data comes to the input of the program in the form of a file, which means that the structure of the file must be fully described. A similar requirement is met for intermediate and output data. For example, for sites, the input information can be text files (specify in which format), graphic information (specify the format), etc. Output information is HTML, PHP, etc. pages visible in the browser window (specify which ).

*Description of the database.* In this section, it is necessary to describe the structure of the database, if there is one in the project. If the application does not use a database, this item is omitted.

When describing the structure of the database:

* all tables are described in the form: *table name* and its purpose, i.e. what information this table is intended for storing;
* a description of all fields of the table with an indication of the type, purpose, primary and foreign keys.

For the object database, a description of the data structure of all classes is provided, similarly to how it is done for tables, only the description of methods is added.

If tools for working with a database have been developed and created in the final qualifying work, they can be described in this section.

*Example.*

A database is a collection of bulk data stored in a framework that makes it easy to find and examine relevant data.

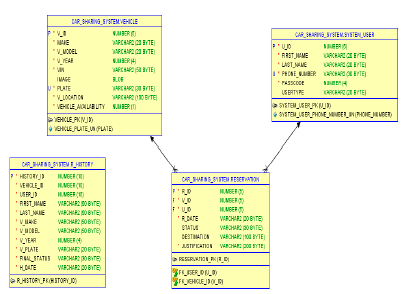


Figure 3 – Database scheme

The "System user" table contains information for users who log in to the system.

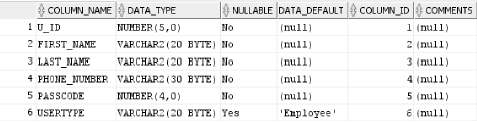


Figure 4 – Table structure “System user”

# 4. IMPLEMENTATION

## 4.1. IMPLEMENTATION OF INTERFACES

This section demonstrates and analyzes the main achievements obtained as a result of completing the final qualifying work. Examples of the program's work with different input data, you can give pivot tables and graphs. If the main task of the work was to conduct a study, then the methods and results of the study, classification of results, tabular comparative data are given.

In this section, you can provide screenshots with explanations for a better perception of the section or a photograph of the technical system.

*Example.*

To add the Library to Arduino IDE, the user has 2 options. The first option is by using the Arduino Library Manager. This can be found in the Tools > Manage Libraries menu or using the shortcut ctrl + shift + I in the resulting window, search for “MBEDSpeech”, then click on install when the Library is found as shown below

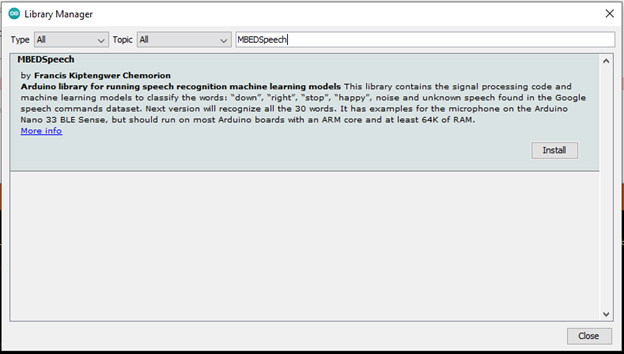


Figure 5 – EMBEDSpeech Library in Arduino Library Manager

# 5. TESTING

## 5.1. TESTING METHODOLOGY

In this section, it is necessary to indicate the degree of debugging of the software or hardware. For example, indicate that a software or hardware device is fully debugged and put into operation (for example, an experimental one). You can specify whether it passed alpha or beta testing, that is, it was tested only by the developer or "outside" colleagues. You can indicate on what data, how the testing of Software System or Software and Hardware System was carried out. If necessary, you can bring a test for the correct functioning of Software System or Software and Hardware System.

Depending on the system proposed by the student, he chooses the testing methodology:

* unit testing;
* integration testing;
* system testing;
* acceptance testing;
* load testing, etc.

## 5.2. TEST PROCEDURE

Tests are specially developed to check all possible situations of the developed PS or PTS, the test results are documented and attached. Based on the results of testing a PS or PTS, technical, algorithmic or software optimization can be carried out.

It is recommended to put test cases into the application, and in the case of automated testing – code examples.

*Example.*

5.2.1. Testing the Machine Learning model

For testing the neural network, 20% of the data used for training and 80% sample data that the CNN had not interacted with was used to evaluate how the model is performing. Accuracy of the model was measured as the percentage of windows of audio that were correctly classified

A Confusion matrix is a table showing the balance of correctly versus incorrectly classified windows. This is by comparing the values in each row, the On-device performance region shows statistics about how the model is likely to run on-device, inferencing time is an estimate of how long the model will take to analyze one second of data on a typical microcontroller, peak is how much RAM will be required to run the model on-device.

For EMBEDSpeech, the following were the results of testing the model: The accuracy is 87.6%, an inferencing time of 4ms, peak ram usage of 4.3k and ROM usage of 47.3k. The confusion matrix is as shown below.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **DOWN** | **GO** | **LEFT** | **NO** | **NOISE** | **OFF** | **ON** | **RIGHT** | **STOP** | **UNKNOWN** | **UP** | **YES** |
| **DOWN** | **73.30%** | 13.10% | 0.30% | 2.10% | 3.60% | 0.60% | 0.90% | 0% | 1.50% | 1.20% | 1.80% | 1.80% |
| **GO** | 8.40% | **74.00%** | 0.60% | 0.60% | 4.30% | 3.10% | 1.20% | 0.30% | 1.90% | 0.90% | 3.40% | 1.20% |
| **LEFT** | 0.60% | 0.60% | **79.60%** | 0.30% | 5.20% | 1.20% | 0% | 4.30% | 0.60% | 0.30% | 0.60% | 6.70% |
| **NO** | 4.00% | 18.00% | 2.10% | **67.00%** | 3.10% | 0.60% | 0.90% | 0.60% | 0% | 0.60% | 1.80% | 1.20% |
| **NOISE** | 0% | 0% | 0.20% | 0% | **96.60%** | 0% | 0.20% | 1.00% | 0.20% | 0.20% | 0.70% | 0.70% |
| **OFF** | 0% | 0.90% | 0.30% | 0% | 4.90% | **78.70%** | 0.90% | 0% | 0.90% | 0% | 13.00% | 0.30% |
| **ON** | 0.90% | 0% | 0.30% | 0% | 5.10% | 5.70% | **84.10%** | 0.90% | 0% | 0.60% | 2.40% | 0% |
| **RIGHT** | 0% | 0% | 2.50% | 0% | 3.90% | 0.60% | 1.10% | **91.70%** | 0% | 0% | 0.30% | 0% |
| **STOP** | 0.30% | 3.40% | 0% | 0% | 15.80% | 2.10% | 0% | 0% | **68.50%** | 0.30% | 9.60% | 0% |
| **UNKNOWN** | 2.00% | 10.70% | 6.10% | 3.10% | 11.20% | 4.10% | 18.90% | 14.80% | 4.10% | **18.90%** | 4.10% | 2.00% |
| **UP** | 0.30% | 2.10% | 0.60% | 0% | 14.10% | 6.50% | 0.90% | 0% | 0.30% | 0.30% | **75.10%** | 0% |
| **YES** | 0% | 0.30% | 4.70% | 0.30% | 4.00% | 0% | 0% | 0.30% | 0% | 0.30% | 0% | **90.10%** |
| **F1 SCORE** | 0.78 | 0.67 | 0.82 | 0.78 | 0.77 | 0.78 | 0.83 | 0.89 | 0.77 | 0.3 | 0.72 | 0.88 |

Figure 6 – EMBEDSpeech Confusion Matrix

5.2.2. Testing the library using Arduino Lint

The Arduino team created a tool to check Arduino projects for common problems. Arduino Lint runs over 175 checks on your sketches, libraries, and boards platforms which cover specification compliance, Library Manager submission requirements, and best practices.

When I run Arduino Lint on my Library, all checks are passed as seen in the outputs below.

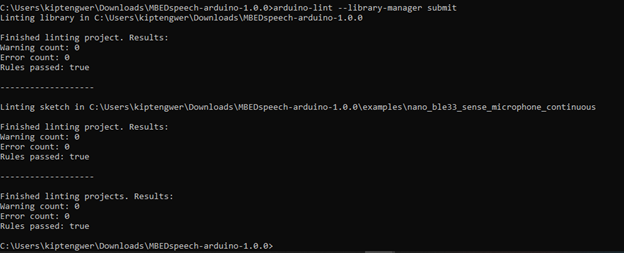


Figure 7 – Arduino Lint Test Results

5.2.3. Conclusion

In this chapter we have conducted testing for the machine learning model which returned an accuracy of 86% and tested the Arduino library using Arduino lint. The library is now ready for use by other developers.

# 6. CONCLUSION

*Conclusion* – a consistent, logically coherent presentation of the results obtained and their relationship with the general goal and specific tasks set and formulated in the introduction.

The conclusion should contain brief conclusions based on the research results, reflecting the novelty and practical significance of the work, proposals for the use of its results, technical and economic efficiency.

In the conclusion, the theoretical and practical conclusions are summarized, as well as the proposals that the author came to as a result of the research. The conclusions are formulated point by point as they should be announced at the end of the presentation at the defense.

The conclusion reflects the assessment of the work and includes recommendations for the practical use of its results.

This section requires:

* indicate whether the problem has been solved in full or in part;
* to sum up the work done - what has been done to solve the problem, it can be:
* Developed and implemented Software System or Software and Hardware System.
* Research work.
* List of solved model or real problems.
* New mathematical algorithms have been developed.
* A new approach to solving such problems is proposed.
* A new method for the implementation of Software System or Software and Hardware System has been proposed.
* A technology for solving such problems has been developed.
* to assess the practical significance of the work, whether it will have practical application;
* to mark possible points of growth (development) of Software System or Software and Hardware System. Example:
* Add new functions (blocks, operating modes).
* Extend to a new class of tasks, a different data type.
* provide comparative characteristics (briefly) of the work done with existing analogues, if any;
* list the types of work done and the results obtained;
* bring the volume of the created software (in any units, for example, in kilobytes, lines of code, classes, procedures, functions, etc.);
* to reflect the approbation of work: speeches at seminars, publications, speeches at conferences and competitions (received diplomas, certificates, etc.), acts of implementation, actually operating website (address), etc.

*Example.*

This project undertakes a viable solution for the need of machine learning at the very basic level, that is, in low powered embedded devices. The project will enable us to bring device with a microcontroller under the control of our voice without having connect to any speech recognition cloud services. It basically uses the commands in the speech command dataset, which is carefully chosen and ideal for robotic applications.

The tasks solved in this thesis include

1. Development of a Library that allows real-time classification of raw audio that can run in Arduino microcontrollers. This is by developing a machine learning model that can identify different words in the Speech commands dataset.
2. Development of a library that runs on devices that consume low power and have low processing capabilities by converting the machine learning model into a TensorFlow Lite model that can run in very small devices.
3. Testing of the library has been done using Arduino Lint and an example of how to implement interfaces is added into the Library and can be accessed using the Arduino IDE.
4. Deployment of the library in a version control environment (GitHub) has been done. The link to the repository is: https://github.com/\_\_\_\_\_\_\_.
5. This Library has been deployed into the Arduino library manager for all MCU architectures. Link to information about the Library is: https://www.arduinolibraries.info/libraries/\_\_\_\_\_\_\_\_.

Due to the successful running of the initial library release, the next steps will be to add the rest of the keywords present in the speech dataset with the 1.0.2 release. Optimizations as well will be done to tune the performance of the library as shown below:

6.1. Optimizing Latency

Designing model architectures is difficult and time-consuming, but there have recently been some advances in automating the process, such as MnasNet, using approaches like genetic algorithms to improve network designs. These are still not at the point of entirely replacing humans.

I am therefore looking forward to using ready services like AutoML that allow users to avoid many of the gritty details of training, be able to design the best possible model for your data and efficiency trade-offs solving latency issues.

6.2. Optimizing Power Usage

For this I will try to estimate how much power the model uses on different devices by measuring the latency for running one inference, and then multiplying the average power usage of the system for that period to get the energy usage. After knowing how many arithmetic operations a model requires, and roughly how many operations per second a processor can perform, I can roughly estimate the time that model will take to execute. I intend to get these device power usage numbers at a particular frequency and voltage from datasheets.

6.3. Optimizing Model and Binary Size

Currently during training, weights are usually stored as floating-point values, taking up 4-bytes each in memory. Because space is such a constraint for embedded devices, I will use the compression utility in TensorFlow Lite to reduce those values down to a single byte in a process called quantization. It works by keeping track of the minimum and maximum values stored in a float array, and then converting all the values linearly to the closest of 256 values equally spaced within that range. These codes are each stored in a byte, and arithmetic operations can be performed on them with a minimal loss of accuracy.

# REFERENCES

1. Kitainik L. Fuzzy Decision Procedures with Binary Relations. Towards a Unified Theory / L. Kitainik. – Boston: Kluwer, 1967. – 254 p.
2. Bandler, W. Fuzzy power sets and fuzzy implication operators / W. Bandler, L. Kohout // Fuzzy Sets and Systems. – 1980. – V. 4. – P. 13 – 30.
3. Topolsky, D.V Mining Explorer Robot / D.V. Topolsky, I.G. Topolskaya, I.G. Plaksina // Proceedings – 2021 International Russian Automation Conference, RusAutoCon 2021. – 2021. – P. 480-485.
4. Speech Recognition [Electronic Resource] http://www.programmersclub.ru (accessed: 20.03.2021).

# APPENDIX A SOURCE CODE

*Example.*

#!/bin/bash

# Generate 20 files with 10 rows

nflies = 25

nrows = 10

for ((i = 0; i<$nflies; i++)); do

filename = "dat\_"$(date + %F\_%H\_%M\_%S)"\_micst\_"$nrows"ent\_perdata\_"$i

echo - e $i": Next "$nrows" rows ... \n"

for ((j = 0; j<$nrows; j++)); do

echo - e $(shuf - i 0 - 2 - n 1)"."$(shuf - i 0 - 999 - n 1)" 1.000 1.000 1.000 0.000 3.000 0.000025\n" >> . / workunits / $filename

done

done

for fname in $(ls ./ workunits); do

cp ./ workunits / $fname `./ bin / dir\_hier\_path $fname`

#rm - f ./ workunits / $fname

# --target\_user 8

./bin/create\_work --appname non - lin\_ode\_analysis --min\_quorum 1 --target\_nresults 1 --wu\_name wu\_nloa\_$fname --wu\_template templates/default\_in --result\_template templates/default\_out $fname

done

rm - f . / workunits/\*

9