REAL-TIME SYSTEM FOR BIRD SOUND RECOGNITION

by

Okan Küçüktopcu

Submitted to Graduate School of Natural and Applied Sciences
in Partial Fulfillment of the Requirements
for the Degree of Master of Science in
Electrical and Electronics Engineering

Yeditepe University

2017

REAL-TIME SYSTEM FOR BIRD SOUND RECOGNITION

APPROVED BY:

Prof. Dr. Cem Ünsalan (Thesis Supervisor)	[illegible]
Assist. Prof. Dr. Engin Maşazade (Thesis Co-supervisor)	[illegible]
Prof. Dr. Duygun Erol Barkana	[illegible]
Prof. Dr. Lütfiye Durak Ata	[illegible]
Assoc. Prof. Dr. İlker Bayram	[illegible]

DATE OF APPROVAL: .... /.... /2017

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my family for always being supportive of my endeavours.

I would like to express my deep gratitude both personally and professionally, to my advisor, Prof. Dr. Cem Ünsalan and my co-advisor Assist. Prof. Dr. Engin Maşazade. The completion of this study would be impossible without their guide and encouragement. I will remember every word of their valuable advice through all my future life and studies.

I would like to thank Prof. Dr. Duygun Erol Barkana for being a committee member and her support throughout my study at Yeditepe University. I would like to thank Prof. Dr. Lütfiye Durak Ata and Assoc. Prof. Dr. İlker Bayram also for being a committee member.

My sincere thanks also goes to Prof. Dr. Pramod K. Varshney for his support and guidance throughout this study. I am very grateful to him for offering me the summer internship opportunity in Syracuse University and leading me to get great a experience with study abroad.

I am very grateful being a part of research assistants of EE department. Their great friendships and my three years at Yeditepe University together with them will always stay as a valuable experience in my life. Thanks!

Finally, last but by no means least, I would like to thank Mustafa Murat Bilgiç, my project manager at work for always being supportive, very understanding and patient throughout this thesis study.

ABSTRACT

REAL-TIME SYSTEM FOR BIRD SOUND RECOGNITION

Environmental sound processing is one of the main research areas in biodiversity preservation studies. One of the most important components of these studies is the sound processing system to be used. In this study, we propose a stand-alone, low-level, custom-made, real-time environmental sound processing system concentrated on single-labeled bird calls and composed of a microphone circuitry, a Texas Instruments Tiva C Connected Launchpad (consisting of an ARM Cortex-M4F based microcontroller), and a storage unit. The proposed system enables data recording and on-board preliminary signal processing, feature extraction, classification and data storage. In the proposed system, we simultaneously record and process data. As the first processing step, we filter out steady background noise using spectral noise gating technique. Secondly, we detect necessary sound signal parts. We then extract mel frequency cepstrum coefficients (MFCCs) as features from the sound signal and classify features by minimum distance classifier with trained features. As the last step, store class labels after classification on an SD card. This is done by effectively defining and utilizing a ping-pong buffer structure on the microcontroller. The proposed system offers flexibility (both in hardware and software) for expansion.

ÖZET

KUŞ SESİ TANIMLAMASI İÇİN GERÇEK ZAMANLI SİSTEM

Çevresel seslerin işlenmesi, biyo-çeşitliliğin korunması için yapılan çalışmaların temel araştıma konularından biridir. Bu çalışmaların en önemli ögelerinden biri kullanılacak olan ses işleme sistemidir. Biz bu çalışmada kendi kendine çalışabilen, düşük seviyeli, amaca uygun tasarlanmış, gerçek zamanlı, tek etiketli kuş sesleri üzerine yoğunlaşmış, bir adet mikrofon devresinden, bir adet Tiva C Connected Lauchpad'ten (Üzerinde ARM Cortex M4F tabanlı bir mikrodenetleyici barındıran) ve bir depolama biriminden oluşan çevresel sesleri işleyen bir sistem ileri sürmekteyiz. Önerilen sistemde, aynı anda verileri kaydeder ve işleriz. İlk işlem adımı olarak, spektral gürültü kapılama tekniği kullanarak durumunu koruyan arka plan gürültüsünü filtreleriz. İkinci olarak, ses kaydı içindeki gerekli kısımları ayırırız. Daha sonra, ses sinyalinden mel frekanslı cepstrum katsayılarını (MFCC) çıkarıp, öğretilmiş özellikler kullanarak en düşük mesafe sınıflandırıcısı ile özellikleri sınıflandırırız. Son adım olarak, sınıflandırma sonunda oluşan sınıf etiketlerini bir SD karta kaydederiz. Bu işlemler, mikro denetleyici üzerinde ping-pong arabellek yapısının etkin bir şekilde tanımlanması ve kullanılması ile yapılır. Önerilen sistem yazılımsal ve donanımsal anlamda esneklik sunar.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

B	Filterbank Energies of a Frame
c_n	Mel Frequency Cepstrum Coefficients Values
C	Column Indexes of Non-zero Elements
D	Number of Mel Frequency Cepstrum Coefficients
E_k	Filterbank Energies
f	Frequency Value
F_{mcu}	Frequency of the Microcontoller
F_s	Sampling Frequency
$g\left(x\right)$	Euclidean Distance
i	Class Number
J	Signal Energy
L	Number of Frames
m	Mel Frequency Value
M	Size of Triangular Filterbank
N	Window Size
N_{buffer}	Ping-Pong Buffer Size
N_{cycles}	Total Cycles
P	Magnitude Frequency Response Array of a Frame
R	Number of Non-zero Elements in Row
$x[n]$	Sampled Signal
$X[k]$	Discrete Fourier Transform of Sampled Signal
y	Feature Vector of a Bird Sample
Z	Non-zero Elements in Triangular Filter Matrix
μ	Mean Point of a Class
μDMA	Micro Direct Access Memory
AC	Alternating Current
ADC	Analog to Digital Converter

CMSIS	Cortex Microcontroller Software Interface Standard
DCT	Discrete Cosine Transform
DC	Direct Current
DFT	Discrete Fourier Transform
DSP	Digital Signal Processing
FAT	File Allocation Table
FFT	Fast Fourier Transform
FN	False Negative
FP	False Positive
GPIO	General Purpose Input Output
GPTM	General Purpose Timer
IDE	Integrated Development Environment
IFFT	Inverse Fast Fourier Transform
IOT	Internet of Things
MFCC	Mel Frequency Cepstrum Coefficient
NPV	Negative Predictive Value
PPV	Positive Predictive Value
QSSI	Quad Synchronous Serial Interface
RFFT	Real Fast Fourier Transform
SD card	Secure Digital Card
SNR	Signal to Noise Ratio
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
STFT	Short-time Fourier Transform
TI	Texas Instruments
WSN	Wireless Sensor Network

1. INTRODUCTION

The growing rate of human domination and destructive human activities have originated massive damage on natural habitats and caused the earth to suffer irreversible environmental transformations for decades [1, 2]. These facts together with urbanization or contamination of resources such as water and air led to forest degradation, climate change, change of wildlife mitigation routes and even extinction of species. Therefore, biodiversity preservation is one of the main interests in environmental science in our rapidly changing world [3]. Methodologies are developed on environmental monitoring, in order to analyze and herewith preserve the biodiversity. In environmental monitoring, scientists are interested in observing wildlife where they analyze the measurements from the environment to make decisions and predictions on biodiversity, climate change, living species, or detect hazardous events like wildfire [4–6].

As an environmental monitoring method, acoustic wildlife monitoring is used. Acoustic monitoring is a passive animal activity observation method, which is done by listening animal sounds without disturbing them. One of the main research areas of acoustic monitoring is monitoring and counting vocalization of bird species and analyzing them to observe characteristics, diversity, environmental adaptation, seasonal population changes of species and identify the bird species in danger of extinction [7–9]. Because of human-based monitoring is costly and limited in time, researchers benefit from autonomous monitoring systems for classification of bird species, identification of animals of interest [10–13].

There are already sophisticated commercially available sensing units for various acoustic wildlife monitoring applications as in http://www.wildlifeacoustics.com/. These are ideal for long-term deployment, resistant to extreme weather conditions, and have intense data storage capabilities. On the other hand, low-cost, and low-energy wireless sensor nodes can be used together for the same purpose. Such a wireless sensor network (WSN) can be used in acoustic monitoring for inference [14–19]. As a consequence, it is important to design and develop new acoustic sensing platforms to add customized sensing modalities, signal processing capabilities, and additional modules to satisfy dynamic expert needs [18–20].

Within acoustic monitoring, classification plays an important role. Unfortunately, the ambient noise inherent in the environment decreases the performance of the system [9, 21]. Bardeli *et al.* [9] indicates that, experiments in a controlled environment (such as a laboratory) yield good bird sound detection performance. On the other hand, the performance degrades in an uncontrolled environment due to inherent background noise. Mporas *et al.* [21] applies different classificication methods on real-field bird sound recordings with different level of signal to noise ratio (SNR). They get best results with best SNRs. Therefore, it is important to implement a noise removal algorithm before processing environmental audio signals. One of the challenging issues in environmental monitoring is the automatic identification of bird sounds [22–24]. It comes into prominence due to the developments in machine-based recognition systems [25]. Accordingly, the sound-based solutions become adequate to identify the bird species, since the most of the bird sounds identify their species [26]. In machine-based recognition systems, there has to be a feature extraction step. Thus, the sound to be classified is defined by reduced amount of data. To be able to extract features in continuous-time, the data has to be detected first. For WSN applications, the sound signal or processed data has to be stored on the system or a central unit. For both conditions, the signal has to be pre-processed and classified on board to reduce the storage and data transmission load. Thus, only the classified data labels are stored. In addition, to classify any data and test the classification performance, there should be a labeled dataset containing training and test data related to classes.

To organize the bird vocalization in acoustical monitoring, the bird sounds are divided into two parts as bird calls and bird songs. Bird songs are long and complex set of sound signals produced by a male bird mostly. In a few species, some female birds also sing. Most of the bird songs can be heard in a certain time of the year, mostly in breeding time. On the other hand, bird calls are simpler and short sound signals and they are produced by both male and female birds every time of the year. The calls occur in a more functional pattern than a song [27]. Calls are actually produced for functions such as informing about food, synchronous flight, fighting and hawk alarms [8].

In this study, we propose a prototype real-time unit which can simultaneously acquire and process bird calls. The proposed design is a low-level, standalone system. It acquires the acoustic data from a microphone and remove noise elements via spectral noise gating. Then it detects the bird call in the denoised sound signal by energy thresholding and extracts MFCC features. Before the last step, it classifies bird calls by minimum distance classifier on a low-level microcontroller unit. Finally, it saves the labels produced by classification to a storage unit all in real-time. In order to evaluate the performance of the system, we construct a dataset by recordings obtained from the website, Xeno-Canto at at http://www.xeno-canto.org/. Our system and dataset focuses on only bird calls due to reaching sufficient amount of the same type of bird call samples are more available than reaching the same type of bird songs as mentioned in previous paragraph.

1.1. LITERATURE SURVEY

In order to collect bird sounds in the environment, there has to be used a sound recording unit. Wildlife acoustics at http://www.wildlifeacoustics.com/ presents multiple solutions. They propose their Song Meter SM3 and SM4 devices for bird and land animal sound recording. The SM3 and SM4 devices are compact, dual-channel sound recorders. They provide user-ready options such as adjustment of recording schedules, sampling frequency and filtering on devices. Their devices can save 1 TB of data. On the other hand, the devices are not customizable and they do not perform onboard processing. Instead, they present a program called Kaleidoscope to organize, process and classify data. Aide *et al.* [29] propose a recording unit with microphone, iPod Touch, and amplification unit for bird sound recording. They record one minute of every ten minutes of sound instead of real-time, continuous recording. Their recording unit, the iPod touch is not customizable for expansions. Acevedo *et al.* [30] used a Sennheiser ME-62 microphone with MSP430 microcontroller to get 7-minute recordings of every hour during 24 hours. Their microphone unit is high-cost and not customizable. In Aide's and Acevedo's method, they only recorded data for further offline processing instead of onboard processing.

In processing unit perspectives, the products of Texas Instruments (TI) and Linux-based Raspberry Pi models are reviewed. TI provides low-power solutions like MSP430G2553 with 16 Mhz system clock [31]. However, their processing capabilities are not sufficient for complex calculations. TI also presents ARM Cortex-M4F based MSP432 that can reach up to 48 Mhz clock speed with 64 KB SRAM [32]. It's also not sufficient for complex software environments in speed and memory perspective. TI's Tiva series presents TM4C123GH6PM and TM4C1294NCPDT microcontrollers [33, 34]. They provide up to 80 Mhz and 120 Mhz system clock speeds and 32 Kb and 256 KB SRAM memory respectively. There will be memory allocation suffering in TI's TM4C123GH6PM microcontroller. As high processing capability units such as Raspberry Pi are sufficient for real-time data processing purposes with their up to 1.2 GHz clock speeds and 1 GB memory unit. However, they have power consumption issues. The lowest power consumption is 160 mA among models when TI's TM4C1294NCPDT has 43.3 mA current consumption at 120 MHz with off-peripherals.

In order to improve feature extraction and classification capability of the system, the background noise should be removed [9]. Using a regular bandpass filtering will not be sufficient because the noise on the sound signal is distributed on the whole frequency spectrum. Boll [35] proposed the spectral subtraction technique. In their method, the magnitude spectrum of pre-constructed noise profile is subtracted from magnitude spectrum of the noisy signal. The halfwave rectification is applied on output attenuated magnitude spectrum. Berouti *et al.* [37] modified Boll's method by adding weighted subtraction mechanism. Kamath *et al.* [36] modify the Berouti *et al.*'s method by using different weights for different SNR values of the noisy signal. Kiapuchinski *et al.* [38] used spectral noise gating to filter out the noise part in an acoustic signal. They used their method to remove noises on bird sound recordings. In the spectral noise gating, the noise profile is used for gating instead of subtraction. The signal power is greater than noise profile, the signal is kept without any operation. Otherwise, the signal is smoothed taking previous smoothing operations into account, instead of zeroing as in spectral subtraction techniques. To note here, Kiapuchinski *et al.* used their method in real-time by recording and processing data on a high-level embedded unit. However, their method cannot simultaneously record and process data.

Several feature extraction methods from bird sounds have been proposed for recognition of the species. Mel-frequency cepstral coefficients (MFCCs) are the most common feature extraction technique for acoustic based classification [39, 40]. Kogan *et al.* [22], Fagerlund *et al.* [40] and Kwan *et al.* [41] used MFCCs for automatic recognition of bird species. Furthermore, Juang *et al.* [42] used linear predictive coefficients (LPC) as features. The spectrographic features were used to classify bird sounds in [43]. Lee *et* *al.* [26] extracted two-dimensional cepstral coefficients for automatic classification of bird species. Fagerlund *et al.* [44] calculated the permutation transformation coefficients for feature extraction.

In order to distinguish the animal species using their sound, several classification methods have been proposed. Acevedo *et al.* [10] compared three methods, linear discriminant analysis (LDA), decision tree (DT) and support vector machines (SVM) with pre-recorded 12 bird species sounds on a computer. Vilches *et al.* [45] used data mining algorithms with a naive Bayes classifier on bird songs. They used pre-recorded 154 bird songs from 3 species. Cai *et al.* [46] studied on pre-recorded bird calls from 14 species with neural networks (NN). They use cepstrum coefficients with linear and mel frequencies as features. Briggs *et al.* [47] proposed the classifier chains with the random forest method for bird sound classification. They also used pre-recorded bird sounds and their platform for processing was a computer. Fagerlund *et al.* [48] used kNN classifier using NN classification on user-ready dataset. They process and classify species on a computer offline. The developed methodologies on bird sound classification have three common characteristics in our review. Firstly, they used pre-recorded data to classify bird species. Secondly, they realized all the processes for classification on a computer system. Thirdly, they performed their methods offline, instead of real-time processing.

1.2. LIST OF CONTRIBUTIONS

In this thesis, bird sound pre-classification and classification methods are investigated. We develop standalone, low-level, custom-made, real-time environmental sound processing system concentrated on bird calls. Our contributions are listed below.

In our system, we perform onboard recording and processing simultaneously instead of recording data for further offline processing [10,30,45–48]. All the recording and processing steps are performed on a low-level microcontroller unit which is limited in processing power and memory space. In order to achieve simultaneous recording and processing, we implement ping-pong buffer structure on our limited microcontroller unit. Thus, we can process real-time discrete sound signals without sacrificing any sound data on a low-level system.

In order to record sound data, we develop a microphone circuitry for our implementation which is customizable for further studies such as optimizations on current consumption issues and sound quality. Furthermore, we develop our microphone circuitry so that it can easily be assembled on embedded platforms.

We perform five processing steps as noise removal, detection, feature extraction, classification, and data storage while recording sound signal in our system. As a noise removal method, we implement spectral noise gating [38]. We use energy thresholding for detection of sound. Then we extract MFCCs as features. Finally, we classify features by minimum distance classifier and store only class labels on an SD card instead of storing all the recorded sound signals.

To be able to run all these steps in real-time with our low-speed unit, we make optimizations on sub-steps of the processing steps. Overall, on-board, real-time and standalone classification and storage of bird species by their calls are achieved on a low-level unit in the proposed system.

1.3. THESIS ORGANIZATION

The rest of the thesis is organized as follows. In Chapter 2, we introduce the design details of the proposed system in separate sections as hardware and software details. The hardware units which are microcontroller unit, microphone circuitry, and storage unit are elaborated in separate subsections. In Chapter 3, we provide the detailed explanation of the implementation processes in 6 steps as sound recording, noise removal, detection, feature extraction, classification, and storage. In each part, we introduce the methods used to achieve corresponding steps of the system and explain how we implement them in detail. In Chapter 4, we express the experimental studies in two sections. In the first section, we evaluate the implementation performance of the system step by step and mention how we constructed the dataset to analyze the classification performance. Finally, we give a detailed explanation of classification performance on noisy and denoised sound signals. In the second section, we give an analysis on system hardware as memory usage, speed and power consumption. In Chapter 5, we summarize our proposed prototype system and make inferences and mention possible future works to improve the study.

2. DESIGN DETAILS OF THE PROPOSED SYSTEM

In this section, we consider the basic design of the proposed system. We handle hardware and software parts in separate sections. The aim here is getting familiar with the strengths and weaknesses of the proposed system. To note here, we propose a custom-made sound processing system such that new modules and properties can always be added as hardware allows.

2.1. HARDWARE DETAILS

Our real-time environmental sound processing system is composed of three basic hardware modules. We provide the hardware layout of our system in Fig. 2.1. The first module is the microphone circuitry. This module is responsible for acquiring and conditioning the environmental sound. The second module is the microcontroller which is responsible for the computation in the system. The third module in our system is the storage unit. This module is responsible for storing the processed audio data. Next, we explain each module in our proposed real-time system.

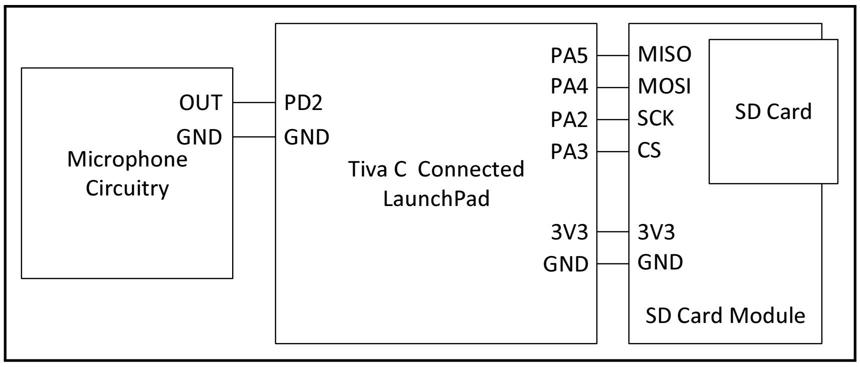


Figure 2.1. Hardware layout of the proposed system.

2.1.1. Microphone Circuitry

The microphone circuitry in our prototype system consists of two parts as a mono channel microphone and an amplifier part. The mono channel electret microphone is specifically picked for this application since it is suitable for embedded platforms [49]. The amplifier part has a low voltage audio power amplifier, LM386 [50]. We design the circuitry in Fig. 2.2 for amplification taking the design with voltage gain of 200 in LM386 datasheet as reference. The R1 value can be changed corresponding to the microphone used. It is chosen as 10

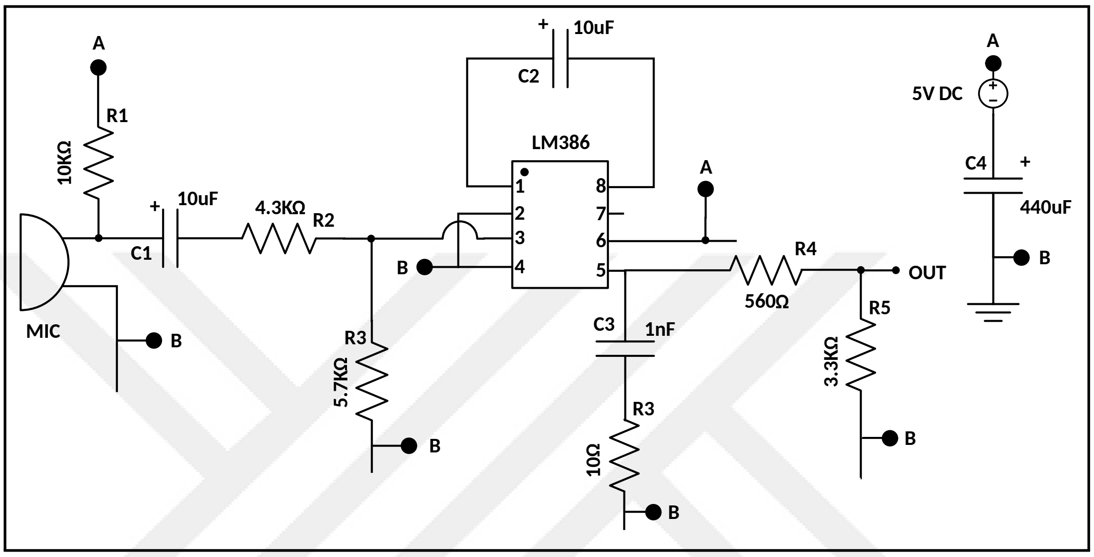


Figure 2.2. Microphone circuitry schematic of the proposed system.

KΩ because maximum current for the electret microphone is 0.5 mA and supplied voltage level is 5 V. C1 capacitor is used to remove the DC terms on the signal since the sound signal is AC signal. Its value is determined experimentally. The R2 and R3 values provide the tune of sound volume. Because of the noise elements originated from microphone, it is not desired to amplify the noise in the original signal. Besides, the amplitude of sound signal coming from the microphone can cause distortion on the output signal if the input is not adjusted effectively. Therefore, to help to decrease or removing the low noise elements and set the maximum value of the sound signal after amplification to an effective value, the volume of the sound signal coming from the microphone is decreased by fine-tuning the R2 and R3 values. In the beginning, a potentiometer is used to determine the optimal values for R2 and R3. The C2 is picked as 10 μF to set the voltage gain of the LM386 to 200 as indicated in LM386 datasheet. C3 acts as a current supplier for the output to help providing the demanded current against deficiency of current. R3 ensures limiting the current while C3 is being charged. C3 and R3 are chosen with reference to design with voltage gain of

200 in the LM386 datasheet. R4 and R5 are used as voltage dividers to make the output of the circuitry suitable for analog input pin of the microcontroller used. Source to ground capacitors like C4 are used to handle the noise originated from the source and the value of C4 is also determined empirically.

2.1.2. Microcontroller Unit

Our system is based on a cheap and low power ARM® Cortex®-M4F based Tiva™ TM4C1294NCPDT microcontroller produced by TI [34]. The microcontroller can reach 120 MHz clock speed. It has 1024 KB flash memory and 256 KB single-cycle static random access memory (SRAM). It also presents functional on-chip peripherals such as two 12-bit analog-to-digital converter (ADC) modules with 20 channels, eight 16/32-bit general-purpose timer (GPTM) blocks and 15 physical general-purpose input/output (GPIO) blocks with up to 90 programmable pins. The microcontroller has communication interfaces including eight universal asynchronous receivers/transmitters (UARTs), four quad synchronous serial interface (QSSI) with bi-, quad- and advanced SSI support. In our prototype system, we used the Tiva C Connected LaunchPad (EK-TM4C1294XL) to benefit from the mentioned microcontroller [51].

2.1.3. Storage Unit

Unfortunately, the memory of the picked microcontroller is not sufficient for our operations. Moreover, there is a necessity for an external storage unit for mobility of data. Therefore, we added an external memory unit to our system. To do so, we picked the 8 GB SanDisk Ultra® microSDHC™ UHS-I SD card [52]. We used an SD card socket to interface the SD card to our microcontroller unit. This setup allows us to store large amounts of data in real-time.

2.2. SOFTWARE DETAILS

We developed our real-time system on Code Composer Studio v6.1.0. This is the integrated development environment (IDE) that supports TI's microcontroller and embedded processors. We added three software libraries to our project. These are as follows. First, we included the TivaWare™ Peripheral Driver Library by TI. The functions in this library are extremely important to access the peripheral units of the Tiva C microcontroller. Second, we added the Cortex Microcontroller Software Interface Standard (CMSIS) DSP library offered by ARM. The functions in this library can be used to utilize DSP functions such as FFT in a fast and optimal manner. Third, we added the generic FAT file system, FatFS software library. As mentioned in the previous section, we store the processed data to an SD card in our prototype system. The processed data can be saved either by a file system or without using any format. However, data can be available on computer systems if it is formatted by a file system. Therefore, FatFS library allows us to store data by Fat32 file format. This also allows storing data in an SD card up to 32 GB in real time. We analyzed the input sound data and simulated noise removal operations and presented the outputs of the operations by Audacity®. Audacity is a free, open source, and cross-platform audio software for recording and editing audio files. To simulate the operations on the proposed system as non-real time system, we used MATLAB® as the software platform. We use Weka data mining software to use classification algorithms offline [60]. Weka provides user-ready machine learning algorithms and the algorithms can be applied on datasets directly.

3. SYSTEM IMPLEMENTATION

Our real-time environmental sound processing system has six implementation steps. These are as follows: sampling and recording the sound signal; removing background noise from the sampled signal by spectral noise gating; detecting the interested sound parts in recordings; extracting the features for classification by MFCC method; classifying the bird species by minimum distance classifier and storing the output of classification. To observe the performance of sound recording, noise removal, detection processes and feature extraction, we store and reconstruct the signals obtained from the first three steps and store extracted features for the end user apart from actual operations mentioned above. We provide the actual system implementation in Fig. 3.1.

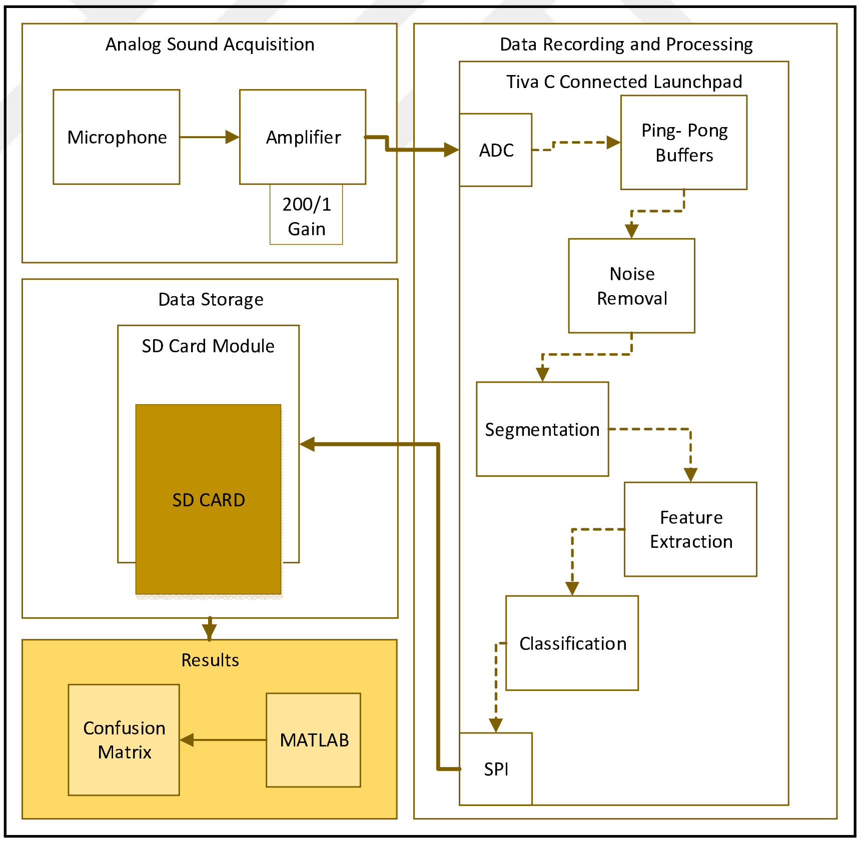


Figure 3.1. Implementation steps of the proposed system.

In the implementation, the sound recording is a continuous process. Therefore, it must not be interrupted by any other processes in real-time. Otherwise, samples may be missed during processing phases. To overcome this problem, we use the ping-pong buffer technique to handle all these operations simultaneously [53]. In ping-pong buffering, when the samples are recorded to the ping buffer, data already filled in the pong buffer is processed and stored. Thereby, we can achieve recording and processing in real-time simultaneously. This implementation uses memory with 16-bit integer values to save instant recorded data because of the 12-bit ADC of microcontroller and the fact that nearest integer size to 12-bit is 16-bit integer values. Although the recorded data is saved in 16-bit length, the ping-pong buffers are set to 32-bit length since all the process in implementation is performed by float values with 32-bit length in the proposed system. The type conversion from 16-bit integer to 32-bit float is done while assigning the recorded data to ping-pong buffers.

3.1. SAMPLING AND RECORDING

As the first step of environmental sound processing, we applied analog signal sampling and recording. We record the analog audio signal by the microphone circuitry of our prototype system. We also amplify the signal before feeding it to the microcontroller. The microcontroller samples the analog signal with its 12-bit ADC (one channel only) by a 20480 Hz sampling rate since most of the birds are in frequency range 100 Hz - 10 Khz [41, 54]. Thus, according to Nyquist sampling theorem, we can obtain sound samples up to 10240 Hz. As the ADC of microcontroller is 12-bit, the recorded data is in range of 0 - 4095. For the possibility of overflow in variables in the further steps of the implementation, the values of recorded data are normalized after type conversion. As the last step for the recording phase, we transfer the samples on 32-bit ping-pong buffers to implement an effective real-time progress. Thus, none of the samples in the sequence of time are sacrificed for further processing. In this step, the recorded sound signals are stored in SD card apart from the actual implementation in order to analyze the recording phase of the system implementation.

3.2. NOISE REMOVAL

The second phase of implementation is noise removal. Unfortunately, the sampled sound signal may contain undesired noise terms besides the actual sound signal. To eliminate the noise, we implemented the spectral noise gating technique in our prototype system. This technique is performed in two steps [38]. The first step is extracting the noise profile. The second step is gating and smoothing. The noise profile is constructed from the obtained signals. This is an offline learning part of the system. The estimated noise profile is further used to remove noise in the signal of interest in real-time. Noise removal is implemented by first constructing the noise profile and then applying spectral noise gating as follows.

3.2.1. Constructing the Noise Profile

Prior to actual environmental sound acquisition operation, the ambient sound is recorded frame by frame with a frame size of 2048 (100 ms) samples over L frames. We first obtain the magnitude spectrum of each frame by taking 2048 channel FFT of data. Then, we obtain the noise gate per each frequency subchannel by averaging the magnitude spectrum over L frames. Thus, we obtain mean frequency spectrum of noise.

In our implementation, ambient sound data is sequentially saved in ping-pong buffers in an alternating fashion. This enables simultaneous data recording and processing. While ping or pong buffer is filled by recording, the magnitude spectrum of the other buffer is extracted by using the FFT functions of the CMSIS library. The average magnitude spectrum is stored in a different noise gate buffer with size 2048 samples.

3.2.2. Applying Spectral Noise Gating

Noise profiling process leads to noise removal operation by spectral gating. Here, the noise profile stored in the noise gate buffer serves as a threshold to gate the magnitude spectrum of the sound data. The spectral noise gating process is implemented likewise in the noise profile construction by using the same ping-pong buffers. Fig. 3.2 shows real-time data recording and processing scheme using ping-pong buffers. As shown in Fig. 3.2, we use a ping-pong buffer technique by copying the first half of the ping-pong buffers to each other in an overlapping and alternating process.

Here, each data frame is considered to be composed of 1024 samples. Data frames are written to ping and pong buffers in an alternating fashion where the size of ping and pong buffers are still kept as 2048 samples (100 ms). According to Fig. 3.2, at time step one, data frame

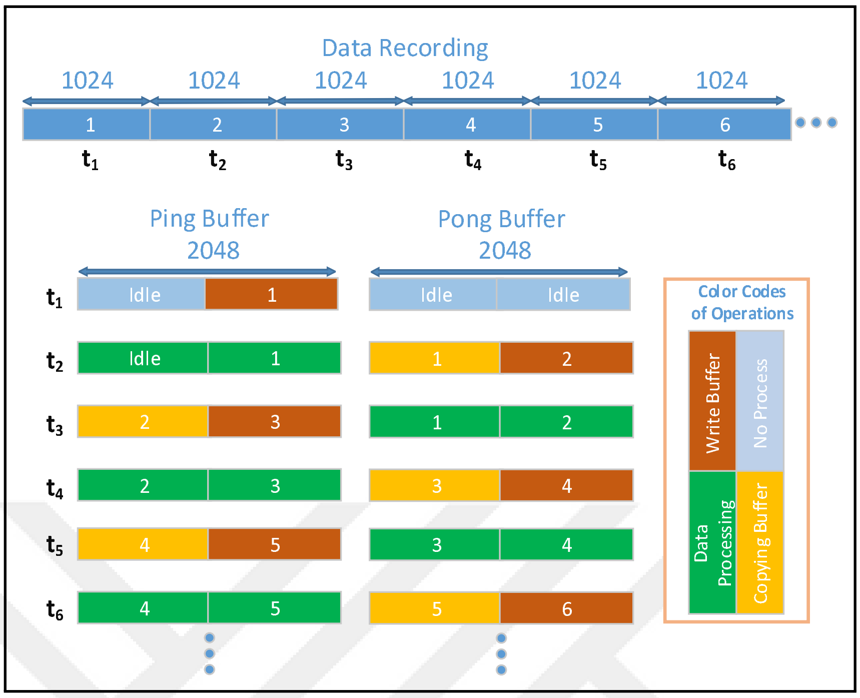


Figure 3.2. Real-time data recording and processing scheme using ping-pong buffers.

one is written to the ping buffer, where the rest of the buffers are empty. At the beginning of time step two, frame one is copied to the pong buffer. Then frame two is recorded to the pong buffer. At the beginning of time step three, frame two is copied to the ping buffer. Then at the pong buffer, spectral noise gating process is executed over the consecutive data frames one and two. Simultaneously, while processing the pong buffer, the incoming data frame three is recorded to the ping buffer. In a similar fashion, at the beginning of time step four, frame three in the ping buffer is copied to pong buffer and the spectral noise gating process is executed in the ping buffer over the consecutive data frames two and three. While processing frames two and three, frame four is written to the pong buffer simultaneously. The process continues this way until the last data frame.

Fig. 3.3 shows the spectral noise gating process in detail. The 2048 (100 ms) sampled frame members either in the ping or pong buffers are first multiplied by a 2048 sample Hamming window. This reduces the leakages caused by overlapped windowing [55]. Here, we save the parameters of the Hamming window in a different constant buffer to reduce processing load. We then take the 2048 channel FFT of the windowed signal. The magnitude at each

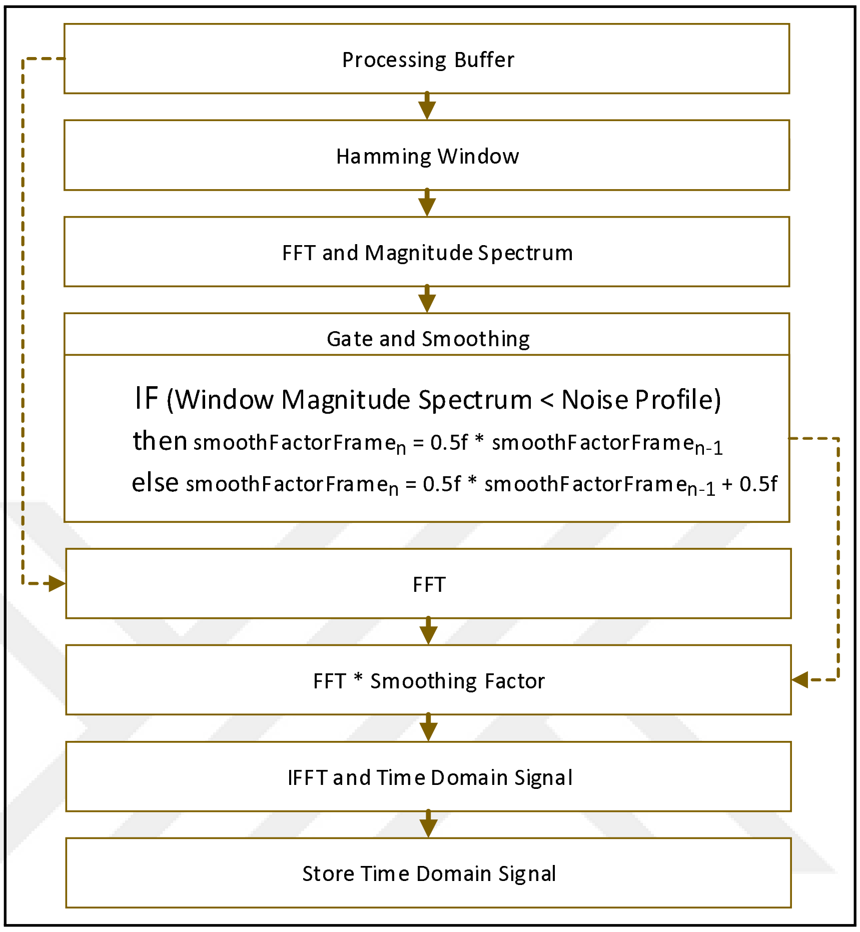

Processing Buffer
Hamming Window
FFT and Magnitude Spectrum
Gate and Smoothing
IF (Window Magnitude Spectrum < Noise Profile)
then smoothFactorFrame$_n$ = 0.5f * smoothFactorFrame$_{n-1}$
else smoothFactorFrame$_n$ = 0.5f * smoothFactorFrame$_{n-1}$ + 0.5f
FFT
FFT * Smoothing Factor
IFFT and Time Domain Signal
Store Time Domain Signal

Figure 3.3. Spectral noise gating steps.

frequency channel is compared with its noise gate obtained in the previous section. If the magnitude spectrum value at a given channel is less than its noise gate value, the smoothing factor is decreased. Otherwise, the smoothing factor is increased as shown in the fourth step of Fig 3.3. Here, at the beginning of the operation, smoothing factor is selected as one. Having obtained the smoothing factor for all frequency channels, the FFT of each channel is multiplied by its corresponding smoothing factor as shown in 5th and 6th step of Fig. 3.3. To note here, the mentioned FFT of each channel is the FFT calculated from ping-pong buffer frame without Hamming windowing. The resulting frequency domain signal is later used in detection and feature extraction operations. It is also converted to time domain signal by Inverse FFT (IFFT) operation using CMSIS apart from the actual system implementation and the final data is stored in an external storage unit in time domain to observe the performance of noise removal.

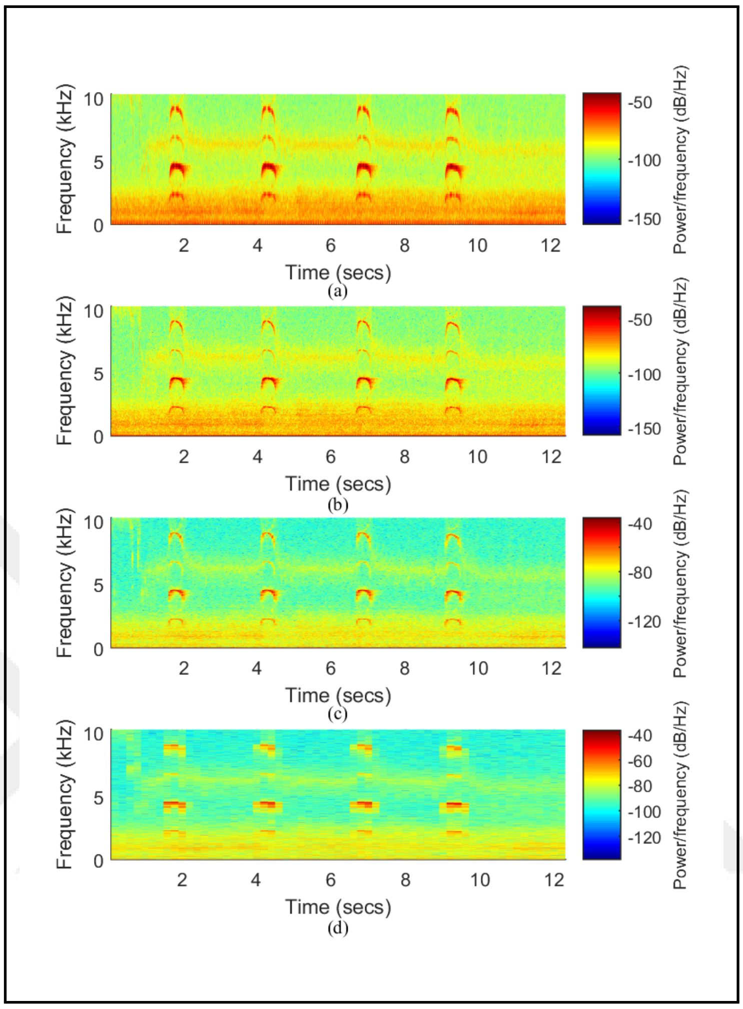


Figure 3.4. Bird calls with different N values, (a)128, (b)512, (c)2048, (d)8192

To be more precise why we implemented the window and FFT sizes as 2048, we analyzed the bird sound recordings with different N values supposing N is window and FFT size. We saw that there is a trade-off between noise terms and bird sound patterns. In Fig. 3.4, there are four spectrograms of recorded four calls of Eurasian sparrowhawk with different N values 128, 512, 2048 and 8192 respectively [56]. The N values are selected in 4-fold intervals to understand the difference visually. As can be seen, samples where N is 128 and 512 have distinct noise terms and sample with size of 8192 has less indistinct patterns than the sample where N is 2048. Using high N values causes memory shortage in our implementation. Increasing the N, it can be supposed that the noise terms are blurred and become easy to remove. In other words, the increasing the size N mimics a low pass filter on the spectrogram.

On the other hand, the clearness and distinction of sound patterns are decreased as can be seen in Fig. 3.4 while increasing the size N. Furthermore, using a small value for N means small window and FFT size. Small window size causes distinct noise terms in recordings and small FFT size causes distortion on bird call pattern. So, we determined the optimum value of N for noise removal and further steps in our implementation as 2048.

To mention here, we implement Real Fast Fourier transform (RFFT) provided by CMSIS Library referred in Section 2.2 for FFT operation to reduce the processing load and memory usage by 2 times for FFT operation. Beacuse RFFT works with the real terms only and calculates only the first half of the FFT bins. Thereby, we only calculate first 1024 bin of 2048 bin FFT for windows where N is 2048 and keep 2048 32-bit memory elements for real and complex part of FFT values instead of 4096 32-bit memory elements for 2048 bin FFT.

3.3. DETECTION OF SOUND PARTS IN RECORDINGS

The third operation of the implementation is the detection of sound parts in the recordings. There are two reasons to implement this section. The first one is to be able to focus on only the necessary sound parts in recordings. After noise removal operation, the recordings may contain silence and residual noise parts that are not necessary for feature extraction and storage. Therefore, we neglect these parts by detection. The second reason is to be able to observe the beginnings and the ends of the meaningful parts in recordings for feature extraction. As mentioned in Section 3.2.2, we work with the frames with the size of 2048 (100 ms). However, a meaningful part of an environmental sound may contain more than 2048 samples (100 ms). Therefore, we observe the beginnings and the ends of the meaningful parts in the recordings by detection.

The detection operation is applied after the sixth part in Fig. 3.3. After smoothing operation mentioned in Section 3.2.2, we obtain the FFT of denoised frame of recording. To detect the interested parts, we use the signal energy. According to Parseval's Theorem,

$$\sum_{n=0}^{N-1} |x[n]|^2 = \frac{1}{N} \sum_{n=0}^{N-1} |X[k]|^2 \tag{3.1}$$

where $X[k]$ is the DFT of the $x[n]$ and N is the window size, the energy of the signal can be calculated by FFT of the signal. To get rid of one more floating point division, we use,

$$J = \sum_{n=0}^{N-1} |X[k]|^2 \tag{3.2}$$

where $X[k]$ is the denoised FFT of recorded frame; J is the energy of the denoised frame; and N is the denoised FFT size with value of 2048.

The detection operation executes as shown in Fig. 3.5. First of all, the magnitude square of

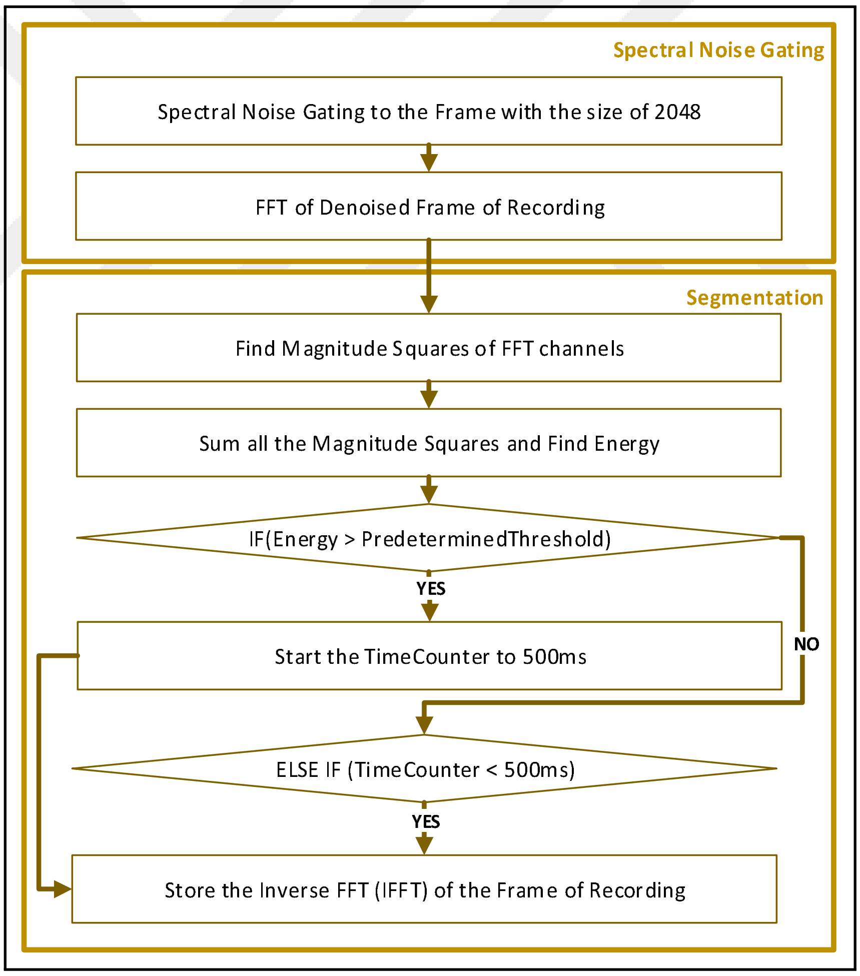

Spectral Noise Gating
Spectral Noise Gating to the Frame with the size of 2048
FFT of Denoised Frame of Recording
Segmentation
Find Magnitude Squares of FFT channels
Sum all the Magnitude Squares and Find Energy
IF(Energy > PredeterminedThreshold)
YES
NO
Start the TimeCounter to 500ms
ELSE IF (TimeCounter < 500ms)
YES
Store the Inverse FFT (IFFT) of the Frame of Recording

Figure 3.5. Implementation steps of the detection of sound parts.

the FFT channels of the denoised signal is calculated. Then, all the magnitude squares are added to find the energy as shown in Eq. 3.2. The energy of the signal is compared with predetermined threshold value. If the energy of the frame is greater than the predetermined threshold, a timer with 500ms period is started, since a meaningful part of environmental sounds like bird calls may contain separate chirp signals. In Fig. 3.6, two seconds recording of a bird species, Common Redstart with one call obtained from Xeno-canto is shown [28]. As can be seen in Fig. 3.6, the call consists of five chirps and there is a silence gap between

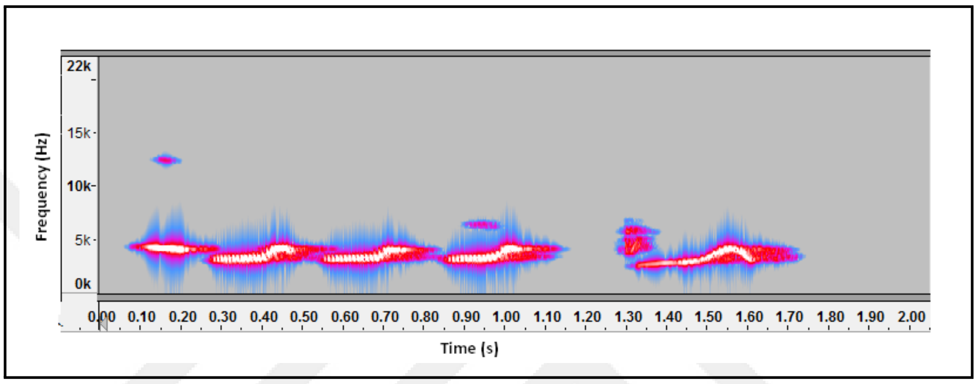


Figure 3.6. A call of a bird, Common Redstart.

fourth and fifth chirp. Accordingly, to process all the chirps or any other sounds like chirps in the same part, the silence signals between chirps in 500ms period are included the same meaningful part of the bird call. After all, IFFT of the frames fulfilled the conditions specified in Fig. 3.5 is used by further steps of the proposed system. It is also stored apart from actual implementation to observe the performance of the detection phase of the overall system.

As can be seen in Fig. 3.5, the detection process operates as the continuation of the noise removal process in the overall real-time implementation. Thereby, the detection phase of the implementation proceeds next to the processing of spectral noise gating in the ping pong buffer technique used in Section 3.2.2.

3.4. FEATURE EXTRACTION

The fourth section of the proposed system is the feature extraction from denoised and detected recordings. As a feature extraction method, MFCCs which mimics the human hearing is used.

The extraction of MFCCs is achieved by six steps [39]. Fig 3.7 illustrates the steps of extraction of MFCCs. As shown in Fig. 3.7, the first process of the MFCCs method is the pre-

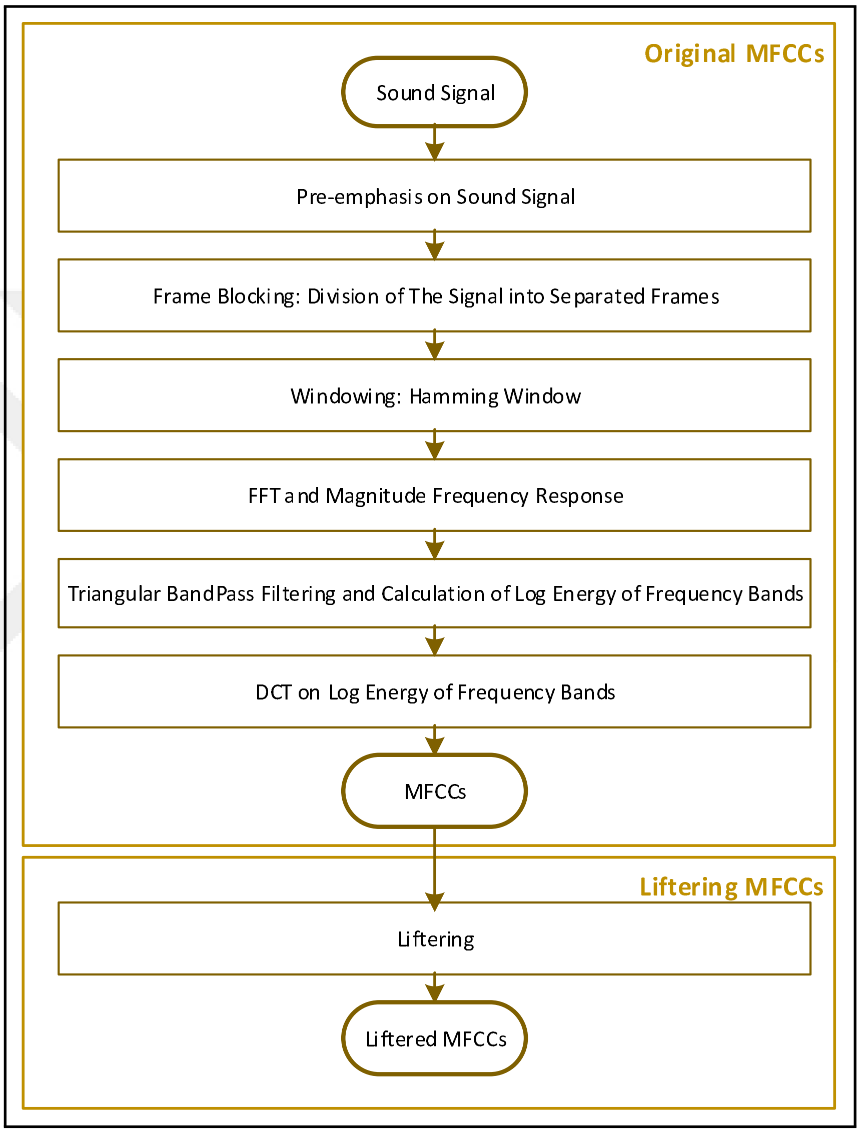

Original MFCCs
Sound Signal
Pre-emphasis on Sound Signal
Frame Blocking: Division of The Signal into Separated Frames
Windowing: Hamming Window
FFT and Magnitude Frequency Response
Triangular BandPass Filtering and Calculation of Log Energy of Frequency Bands
DCT on Log Energy of Frequency Bands
MFCCs
Liftering MFCCs
Liftering
Liftered MFCCs

Figure 3.7. Steps of extraction of MFCCs and liftered MFCCs.

emphasis of the input signal. By pre-emphasis, the high-frequency parts, that are suppressed during sound production mechanism of humans, are compensated. On the other hand, in our proposed system, this procedure is skipped, since we are focused on only the environmental sounds. As the second step, the frame blocking separates the signal into overlapped frames to be able to calculate the short-time fourier transform (STFT). This step is already achieved in ping pong buffer structure as mentioned in Section 3.2.2. The next step is the windowing the signal with Hamming window to reduce the leakages caused by overlapped windowing as mentioned in Section 3.2.2. The fourth step of the procedure is the calculation of FFT to find magnitude frequency response of each frame. As shown in Fig. 3.3, the resulted magnitude FFT calculation at fourth steps of MFCCs procedure is already performed by spectral noise gating. Therefore, we reach the fifth step, the triangular bandpass filtering by skipping first four steps of MFCCs procedures to reduce the processing load.

Triangular bandpass filtering mimics the human perception by using triangular overlapped filters distributed at even intervals in mel frequency domain. To calculate the triangular filters, the frequency domain is converted to mel frequency domain by,

$$m = 2595 \log_{10} \left(1 + \frac{f}{700} \right) \tag{3.3}$$

where m is the mel frequency and f is the regular frequency value. Then, the center points of the triangular filters are determined by distributing them at even intervals in mel frequency spectrum. Then, half-overlapped triangular filters are constructed. The number of center points determines the size of filterbank. The suggested number of filterbank size is in range 20 - 40 [57]. Therefore, supposing M is the filterbank size, M is set to 20 which is the suggested minimum size of filterbank to have minimum processing load on the microcontroller. The magnitude frequency responses of the frames are multiplied by M triangular bandpass filters to calculate energies of each frequency bands in the perspective of human hearing. Thus, we obtain M filterbank energies. The triangular bandpass filtering is used together with logarithm operation since human hearing is prone to logarithmic scale than linear scale [57].

As the last step of extraction of MFCCs, the discrete cosine transform (DCT) is applied on the log filter-bank energies as in,

$$c_n = \sum_{k=0}^{M-1} \log_{10}(E_k) \cos(n \frac{\pi}{M} (k + 0.5)), 0 \leq m \leq D - 1 \tag{3.4}$$

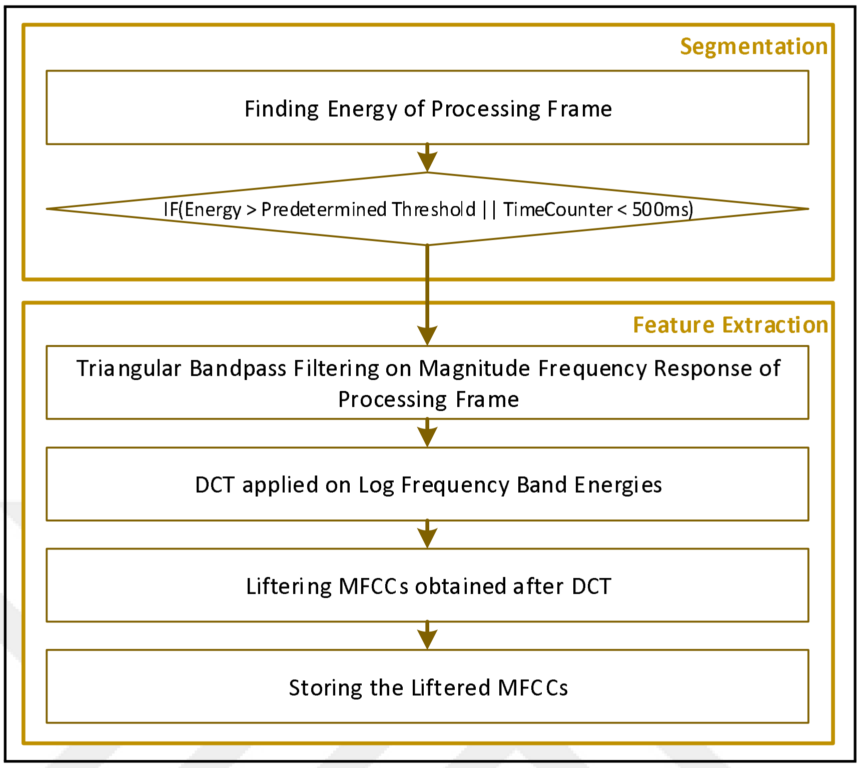

Segmentation
Finding Energy of Processing Frame
IF(Energy > Predetermined Threshold || TimeCounter < 500ms)
Feature Extraction
Triangular Bandpass Filtering on Magnitude Frequency Response of Processing Frame
DCT applied on Log Frequency Band Energies
Liftering MFCCs obtained after DCT
Storing the Liftered MFCCs

Figure 3.8. Feature extraction proceeding in proposed system.

where c_m is the MFCCs; D is the number of MFCCs with value of 13; M is the number of triangular filters; and E_k is the filterbank energies. Thus, we obtain D MFCCs where D is 13 in our proposed system. The DCT is applied for two reasons [57]. All the filterbank energies are quite correlated since filter-banks are overlapped. The DCT decorrelates the filter-bank energies. The higher DCT coefficients represent the fast changes in the filter-bank energies. To remove these high changes, a limited number of DCT coefficients are used. In our implementation, we keep D coefficients of DCT as features.

As an extra step, the sinusodial liftering operation is applied as in Paliwal [58]. Because, liftering operation gives less weight to both higher and lower ceptral coefficients; and it provides better recognition performance.

Looking from the real-time perspective, the connection between the feature extraction operation and the detection operation is similar to the connection between the detection and the noise removal operations. Hence, as shown in Fig. 3.8, the steps of feature extraction implementation run after detection operation in the time intervals of data processing of the ping pong buffer structure mentioned in Section 3.2.2. If the conditions mentioned in Section 3.3 are fulfilled, all the operations after fourth step in Fig. 3.7 is executed and the extracted MFCCs of the current frame are stored in a buffer. All the MFFCs with size of $(L \times D)$ in L frames of a detected part are kept until the detected part ends. If the detected part ends, the mean values of all MFCCs for L frames are calculated and kept as features. To note here, as in the previous steps, the outputs (features) obtained from the feature extraction operation are stored in SD card apart from the actual implementation.

To avoid from extra operating loads in real-time execution, the triangular filter matrix with the size of $M \times \frac{N}{2}$ where M is 20 and N is 1024, DCT coefficients matrix with the size of $D \times M$ where D is 13 and M is 20, and liftering coefficients with the size of D are calculated offline and integrated to the memory of the system, since they are constants. Then, the triangular filtering, applying the DCT on log energies calculated by triangular filtering, and liftering MFCCs procedures are achieved by only multiplication operations in real-time.

As mentioned, the size of the triangular filter matrix is 20×1024 which needs 20480 32-bit floating-point memory elements. However, the matrix of triangular filter contains lots of zeros in it. Therefore, only the non-zero elements, the number of non-zero elements in each row, and the column indexes of first non-zero elements in each row are stored in the memory of the system. Thus, instead of 20480 floating-point numbers, 1902 32-bit floating-point and 40 16-bit unsigned integer numbers are placed in the memory. Correspondingly, the multiplication operations for triangular filtering is realized using the number of non-zero elements in each row and the column indexes of first non-zero elements in each row as shown in Algorithm 3.4.

Algorithm 3.4. Triangular Filter Matrix Multiplication

```
Begin
$M \leftarrow 20$
$k \leftarrow 0$
for i $\in$ 1,...,$M$ do
   for j $\in$ 1,...,$R_i$ do
      $B_i \leftarrow Z_k * P_{C_i + j}$
      $k \leftarrow k + 1$
   end for
end for
End
```

where M stands for the number of triangular filters; R stands for number of non-zero elements in rows; B stands for filterbank energies of a frame; Z stands for non-zero elements in triangular filter matrix; C stands for column indexes of non-zero elements in rows and P stands for magnitude frequency response array of a frame.

3.5. CLASSIFICATION

The next and the fifth step of the proposed system is the classification of the bird species in the recordings. In our system, the minimum distance classifier technique is used for the classification purpose due to the low-weight structure of the classifier in our limited system. The minimum distance classifier uses the Euclidean distance as,

$$g_i\left(y\right) = \sqrt{\left(y - \mu_i\right)^T \left(y - \mu_i\right)} \tag{3.5}$$

where $g_i\left(y\right)$, i, μ_i, y represents Euclidean distance of the features to ith class, class number, the mean point of the i^{th} bird species in D dimensional space as mentioned and the feature vector of test bird records respectively.

To note here, because the existence of the square root operation in Eq. 3.5 does not affect the designation of minimum distance, we remove it on our implementation to decrease the process load on microcontroller.

The classification operation has three parts as train, cross-validation and test. The train part for the minimum distance classifier in our proposed system is the calculation and storage of the mean points of features of each bird species in D dimensional space. The features mentioned are the features of the sample recordings determined as training samples of each class. D is the size of the features extracted by MFCCs method as mentioned in Section 3.4.

In the cross-validation part, maximum possible Euclidean distances from mean points of each bird class are obtained. To be able to calculate maximum possible distance of each class, first all the samples in cross-validation set of recordings are labeled by hand, then the Euclidean distances from labeled samples to mean points of the classes with the same labels are calculated and stored. The maximum distances for each class label are kept as maximum possible distances for each class.

The test part of classification is actually the labeling of the features of sample recordings separated for testing purposes. The calculation of Euclidean distance from a test feature vector to trained feature vector of each class enables us to determine the minimum distance from test sample to classes. After reaching the minimum distance, the possible label of the test is determined as the label of the class which has the minimum distance. Before labeling, a threshold comparison of the reached minimum distance against maximum possible distance of determined possible label is performed. If the minimum distance is in range of maximum possible distance, the test sample is labeled as the label compared. Otherwise, the test is labeled as exclusive class.

The training and the cross-validation parts of the classification are carried out apart from the actual system implementation as offline processes. The features of training set is extracted and stored by microcontroller. The mean point calculation of these features are realized in MATLAB offline. The maximum possible distance determination processes are done in a similar way. First, the features of cross-validation set are stored by microcontroller. Then, the maximum distances are determined by MATLAB offline. The actual process in the implemented system is the operation in the test part. The trained features and maximum possible values for each class are buffered in an array in the microcontroller, in order to use in classification of the test samples.

3.6. STORAGE

As mentioned in Section 2.1.3, we store the processed data on an SD card. This card communicates with the microcontroller by serial peripheral interface (SPI) protocol. We use the QSSI module on the microcontroller as bi-SSI that serves SPI communication. FatFS library allows us to write the data to the SD card byte by byte. On the other hand, as mentioned in Section 3, the ping-pong buffers in our system have 32-bit word length. Therefore, the buffer elements have to be sent as four separate parts. To overcome this difficulty, we use byte based memory addresses of the buffers to send to the SD card without any extra processing load in real-time as shown in Fig. 3.9. Then, all the bytes in byte based

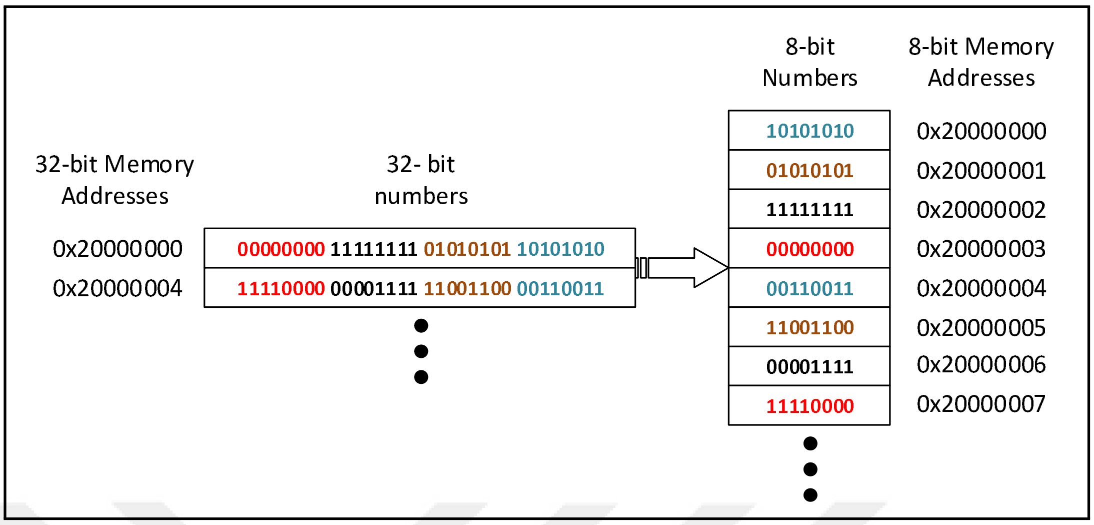


Figure 3.9. 32-bit to 8-bit memory adress to write on SD card.

memories are written on SD card sequentially. Otherwise, we have to separate all the 32-bit buffer elements into 8-bit buffer elements. This puts an additional processing load to the microcontroller. Besides, it affects the memory usage in a negative way.

4. EXPERIMENTAL STUDY

In order to evaluate the system, experimental studies are conducted in two steps. First, the performance of the implementation steps of the system is observed to understand how correctly and effectively the system works. Then, hardware related analysis are done to see how flexible the system is for expansion and how much energy is consumed by the hardware units.

4.1. PERFORMANCE ANALYSIS

In this part of the experimental study, we perform two distinct implementation experiments. The first one is the visual examination of noise removal operation, detection steps. This experimental step is performed apart from the actual real-time operations. The second study is the evaluation of the real-time classification capability of the system using the constructed dataset. As an extra step to classification capability, an offline classification performance evaluation is realized to be able to ponder on how the system will achieve with more complex classification algorithms.

4.1.1. Evaluation of Pre-classification Operations

The system is executed to store signals obtained by microphone circuitry and then store output of the noise removal, detection apart from the actual proposed system. Because we separate the samples after original sound recording, noise gating, and detection into bytes to store on the SD card, we have to recover them to obtain time-domain sound samples. The recovery is performed offline using MATLAB. After recovery of the sound samples, the half-overlapped windowing is performed using Hamming window function to exclude the leakages caused by windowing. Thus, we obtain the original, denoised and detected signals.

We next provide a working example of sound recording, noise removal and detection parts of our prototype system. Within this example, we use the actual bird sound signal recordings. The recorded bird sound signal is played from the speaker of a PC in a house with highway noise. The microphone of the proposed system is located approximately 30 cm away from the speaker of a regular laptop PC. To illustrate the sound volume while recording, the sound level in dB is measured by a smart-phone LG G2. The phone's microphone is placed beside the microphone in proposed system and the original record of the sound in Fig. 3.4 is played by PC. Likewise, the noise level in the house environment is measured. The measured sound and noise levels are maximum at 60 dB and 50 dB respectively. When we compare the measured sound level with the actual speech sound level, it is seen that both are very close to each other. The performances of the spectral noise gating and detection in our prototype real-time system are shown in Fig. 4.1 that illustrates the recorded, denoised and detected spectrograms of the played twelve seconds bird sound signal having four calls. Prior to

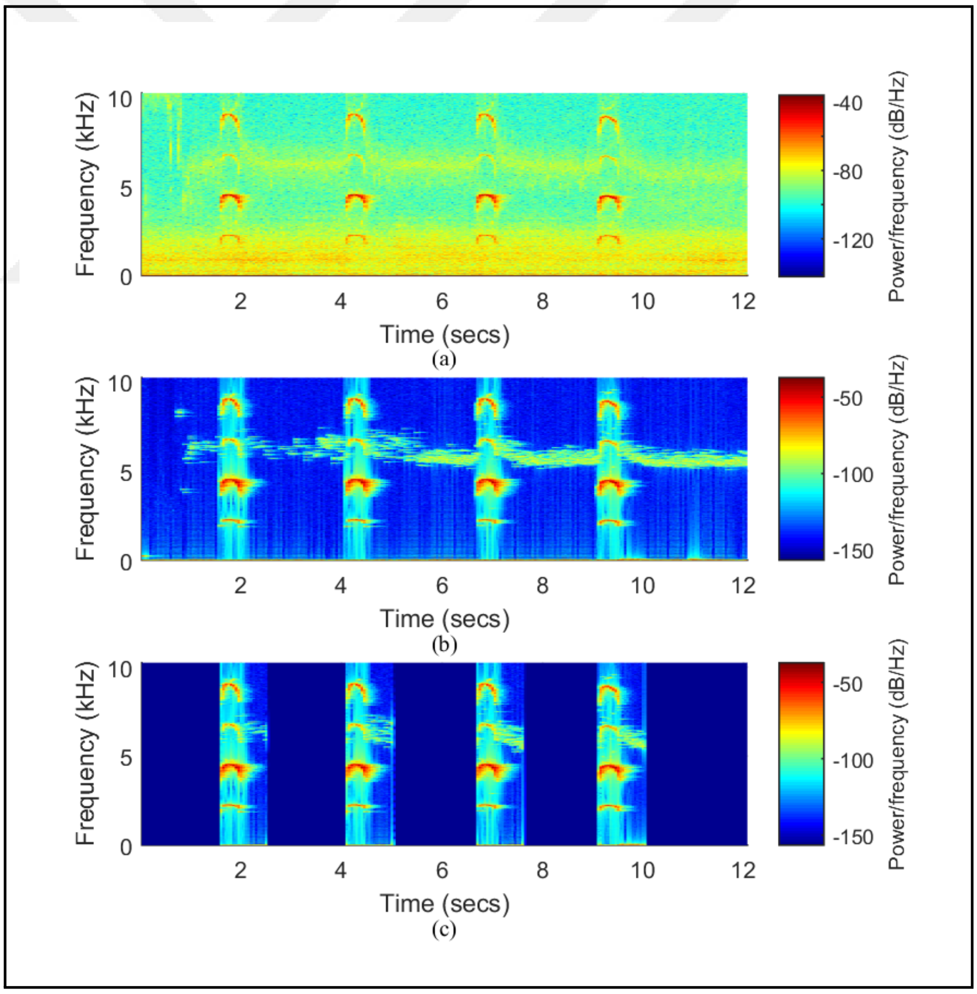


Figure 4.1. Bird calls (a)Recorded, (b)Denoised, (c)Detections

spectral noise gating, the noise profile is constructed by listening to the ambient sound for

30 sec. In other words, the noise profile is constructed by averaging L frames where L is 300.

The first window Fig. 4.1(a) represents the sound signal obtained by the microphone circuitry in our system. As can be seen, the system has a noticeable amount of background noise. This background noise is originated from the environment where recording is performed, non-ideal microphone circuitry and also non-ideal voltage source of the system.

The second window Fig. 4.1(b) is the spectrogram of the reconstructed output signal of spectral noise gating process. As can be seen, the undesired background noise is removed without disrupting the original signal as possible. Accordingly, we can conclude that the noise removal operation provides an independent signal from the sound recording technique and the environment at a sufficient level. As can be seen in Fig. 4.1(b), our implementation of spectral noise gating on our proposed system works properly.

The detection result of after noise removal operation is shown in Fig. 4.1(c). In spectrogram presented in Fig. 4.1(c), the parts outside of the detected parts are the zero power parts after detection. The bird calls in denoised sound are detected properly for this recording into four parts. It provides a proper feature extraction and classification. The implementation of detection on system works as expected, as can be seen in Fig. 4.1(c).

4.1.2. Classification Capability

As the last part of the implementation analysis, the classification results are obtained to be able to evaluate the classification capability.

In order to obtain the results, first a dataset is constructed. The dataset construction is achieved in 3 steps. The first step is determining which conditions are going to be issued while choosing the data. The second step is downloading and merging sound data respectively. The third and the last step is dividing the data into train, cross-validation and test sets.

Four conditions are evaluated for selection of recordings included in the dataset as follows,

- Using recordings with different level of SNR,

- Using single labeled recordings or recordings including one dominant label and other weak labels,
- Using recordings including only bird calls.
- Using world-wide recordings of bird species seen around Turkey.

We obtained all the recordings from the website, Xeno-Canto at http://www.xeno-canto.org/ since we have found the recordings in Xeno-Canto keeping all four conditions. Xeno-Canto is a database website for bird sound collections. It already contains 352682 recordings of 9724 species with different subspecies and its dataset is getting larger day by day. To be clear about using Xeno-Canto for the first condition, the website contains recordings with different sound qualities and there are quality labels A, B, C, D, E for the recordings. So constructing the dataset, we've used recodings A to E.

In Xeno-Canto, the labels of both dominant and weak species are provided for user in recordings. The sound can be listened online and the spectrogram of the recordings can be seen online. Thereby, we keep the second condition for our recordings just before the downloading thorough listening and visualizing them one by one. This website uses a filter for sound types of birds. We determine our recordings by this filter as bird calls. To keep our last condition, we filtered recordings by region and selected around Turkey and found the species used and got the samples of these species in recordings all over the world. To note here, we used Turkey as a region for real environmental experiments as a future work and we used world-wide recordings of species because the data from Turkey is not sufficient in the website yet.

After obtaining recordings, we downloaded 237 recordings and ignored the species with few samples and extracted bird sounds in the recordings and merged them together. In order to evaluate how the system behaves against the sounds outside the trained classes, we also added some other types of sounds such as urban-originated sounds, other species of birds, strong noise. Table 4.1 presents the bird species with class labels and number of samples for each class. As can be seen in the last column in Table 4.1, we have had 1116 samples with 21 species and other types of sounds. To get the total number of samples in merged set, the sound finder tool in Audacity is used. The sound finder tool finds the sound parts in recordings and labels them with a counter value. Fig. 4.2 shows the setting window of

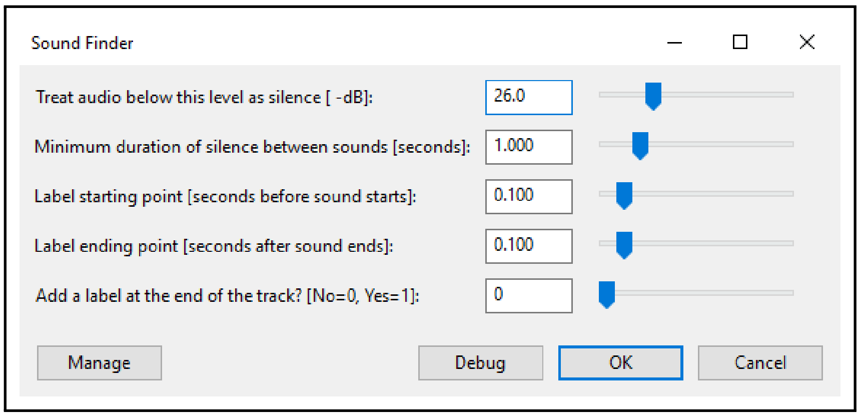
Table 4.1. Bird species with labels and number of samples in dataset

Species	Labels	Samples
Black Frankolin	B1	53
Black RedStart	B2	37
Black Woodpecker	B3	57
Common Blackbird	B4	40
Common Chiffchaff	B5	43
Common Nightingale	B6	69
Corn Bunting	B7	69
Corn Crake	B8	72
Eurasian Magpie	B9	48
Eurasian Scops Owl	B10	59
Eurasian Sparrowhawk	B11	47
Eurasian Wryneck	B12	38
European Greenfinch	B13	52
European Robin	B14	52
Green Warbler	B15	58
Little Bittern	B16	55
Red-billed Chough	B17	50
Rose-ringed Parakeet	B18	41
Ruddy Shelduck	B19	39
Tawny Owl	B20	38
White-throated Kingfisher	B21	53
Other	O	46
Total		1116

Audacity sound finder tool. All the settings presented in Fig.4.2 is used as shown.

As the last step of the dataset construction, we divide the dataset as 20% train, 20% cross-validation and 60% test set for each species. Table 4.2 shows the class labels with their number of train, cross-validation and test samples. Consequently, we have 222 train, 222 cross-validation and 672 test samples in total as can be seen in Table 4.2. As can be seen at last column of Table 4.2, the other type of sounds are not included in train and cross-validation set beacuse they are not samples of specific types and they are mix of sound types.

The next step after dataset construction is the experimental setup. The same experimental setup in Section 4.1.1 is provided for recordings of training, cross-validation, and test sets. We repeat the experiments for two conditions.


Sound Finder

Treat audio below this level as silence [-dB]: 26.0

Minimum duration of silence between sounds [seconds]: 1.000

Label starting point [seconds before sound starts]: 0.100

Label ending point [seconds after sound ends]: 0.100

Add a label at the end of the track? [No=0, Yes=1]: 0

Figure 4.2. Setting window of Audacity sound finder tool.

Table 4.2. Class labes with number of train, cross-validation and test samples

Labels	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15	B16	B17	B18	B19	B20	B21	O	Total
Train (20%)	11	8	12	8	9	14	14	15	10	12	10	8	11	11	12	11	10	9	8	8	11	0	222
CV (20%)	11	8	12	8	9	14	14	15	10	12	10	8	11	11	12	11	10	9	8	8	11	0	222
Test (60%)	31	21	33	24	25	41	41	42	28	35	27	22	30	30	34	33	30	23	23	22	31	46	672

As the first condition, the system is operated regularly as mentioned in Chapter 3. Pre-classification steps in implementation are realized and features of the samples in train and cross-validation sets are extracted by the microcontroller. Then, the class mean points and maximum possible distances of class labels are calculated by MATLAB. As the last step, the test results are obtained by recording and processing samples in the test set by the real-time proposed system. The confusion matrix for this experiment is provided in Table 4.3. As can be seen, 669 samples in the test set are captured with 3 miss. Total 557 samples are labeled correctly. To emphasize here, the false labeling among classes mostly occurs in 10th class, Eurasian Scope Owl to 11th class, Eurasian sparrowhawk with 12 false labeling. Investigating both classes, it is observed that the test set of the 10th class contains samples which have strong bird sounds with the frequency band of 11th class. Therefore, we trained, cross-validate and test the class with excluding that specific samples, the confusion in labeling for 10th class is disappeared. So, the background species are causing the labeling error. As can be calculated in Table 4.3, the most of the false labeling is done to other class label (o) with 54 false labeling.

Table 4.3. Confusion matrix of minimum distance classifier with SNG

		PREDICTED CLASS																					
	Species	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15	B16	B17	B18	B19	B20	B21	O	Actual Total
ACTUAL CLASS	B1	19	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	5	30
	B2	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	21
	B3	4	0	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	33
	B4	0	0	0	19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	24
	B5	0	0	0	0	21	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	24
	B6	0	0	0	0	0	36	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	3	41
	B7	0	0	0	0	0	0	40	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	41
	B8	0	0	0	0	0	0	0	39	0	0	0	0	0	0	0	0	0	0	0	0	0	3	42
	B9	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	1	0	0	0	2	5	28
	B10	0	0	0	0	0	0	0	0	0	23	12	0	0	0	0	0	0	0	0	0	0	0	35
	B11	0	0	0	0	0	0	0	0	0	0	19	0	0	0	0	0	0	2	0	0	0	6	27
	B12	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	4	2	22
	B13	0	0	0	0	0	0	0	0	0	0	0	0	26	0	0	0	0	0	0	0	0	4	30
	B14	0	0	0	2	0	0	0	0	0	0	1	0	0	25	0	0	0	0	0	0	0	2	30
	B15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	33	0	0	0	0	0	0	1	34
	B16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	31	0	0	0	0	0	2	33
	B17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28	0	1	0	0	1	30
	B18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	18	0	0	0	4	22
	B19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	23	0	0	0	23
	B20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	2	22
	B21	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	27	2	31
	O	1	0	0	0	0	0	1	0	0	0	0	0	1	0	6	0	0	3	4	0	0	30	46
	Predicted Total	24	20	24	21	21	36	43	39	21	25	33	22	27	25	40	31	29	23	28	20	33	84	669

In the second condition, we exclude the spectral noise gating technique in our implementation and repeat rest of the steps likewise in the first condition in order to evaluate the effect of the noise removal. The confusion matrix of the results is provided in Table 4.4. As can be seen, 649 different samples are captured with miss 23 samples. Total 471 samples are labeled correctly. The higher miss value here is caused by noise in the recording. Because there may be high background noise, the detection is not achieved properly and some detected parts are captured together and perceived as one detection. In addition, we increase the detection threshold to samples since the noise removal operation is excluded. This causes some misses for detection. Likewise in experiments in the first condition, the most of the false labeling is done to the other class label (o) with 78 false labeling as can be calculated in Table 4.4

To evaluate the classification performance of the two experiments in detail, the test statistics

Table 4.4. Confusion matrix of minimum distance classifier without SNG

		PREDICTED CLASS																					
	Species	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15	B16	B17	B18	B19	B20	B21	O	Actual Total
ACTUAL CLASS	B1	**22**	0	0	0	0	0	0	0	0	0	0	4	2	0	0	0	0	0	0	0	0	2	**30**
	B2	0	**11**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	**20**
	B3	0	0	**21**	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	7	**32**
	B4	0	0	0	**22**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	**23**
	B5	0	0	0	0	**17**	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	6	**25**
	B6	0	0	0	0	0	**36**	0	0	0	0	1	0	0	0	0	0	0	0	2	0	0	0	**39**
	B7	0	0	0	0	0	0	**34**	0	0	0	1	0	0	1	1	0	0	0	0	0	0	1	**38**
	B8	0	0	0	0	0	0	0	**36**	0	0	0	0	0	0	1	0	0	0	1	0	0	3	**41**
	B9	0	0	0	0	0	0	0	0	**20**	0	0	0	1	0	0	0	1	0	0	0	2	4	**28**
	B10	0	0	0	0	0	0	0	0	0	**22**	4	0	0	0	0	0	0	0	0	0	0	9	**35**
	B11	0	0	0	0	0	2	0	0	0	0	**16**	0	0	0	0	0	0	0	0	0	0	8	**26**
	B12	0	0	0	0	0	0	0	0	0	1	0	**15**	0	0	0	0	0	0	0	0	0	5	**21**
	B13	0	0	0	0	0	0	0	0	0	0	0	0	**21**	0	0	2	0	6	0	0	0	0	**29**
	B14	0	0	0	0	0	0	0	0	0	2	5	0	0	**17**	0	1	0	0	0	0	0	4	**29**
	B15	0	0	0	0	0	0	2	3	0	0	0	0	0	0	**27**	1	0	0	0	0	0	0	**33**
	B16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**32**	0	0	0	0	0	0	**32**
	B17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**19**	0	1	0	0	9	**29**
	B18	0	0	0	1	0	1	2	0	0	0	0	0	0	0	0	0	0	**14**	1	3	0	2	**24**
	B19	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	**17**	0	0	1	**21**
	B20	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	**15**	0	3	**20**
	B21	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	**22**	4	**30**
	O	1	0	0	1	0	2	3	0	0	0	0	0	7	0	5	3	0	3	4	1	0	**15**	**45**
	Predicted Total	**23**	**11**	**21**	**24**	**17**	**41**	**42**	**40**	**24**	**26**	**28**	**23**	**32**	**18**	**34**	**40**	**20**	**24**	**26**	**19**	**24**	**93**	**650**

below,

- true positive (TP) which is total number of test results that correctly indicate the presence of a condition,
- false positive (FP) which is total number of test results which wrongly indicate that a particular condition is present,
- true negative (TN) which is total number of test results that correctly indicate the absence of a condition,
- false negative (FN) which is total number test results which wrongly indicate that a particular condition is absent,

are calculated for each species from two confusion matrices in Table 4.3 and Table 4.4 and presented in Table 4.5. As can be seen in Table 4.5, the true positive values for experiment

Table 4.5. TP, FP, FN, TN for confusion matrices in Table 4.3 and Table 4.4

Labels	TP w/ SNG	TP w/o SNG	FP w/ SNG	FP w/o SNG	FN w/ SNG	FN w/o SNG	TN w/ SNG	TN w/o SNG
B1	19	22	5	1	11	8	634	619
B2	20	11	0	0	1	9	648	630
B3	24	21	0	0	9	11	636	618
B4	19	22	2	2	5	1	643	625
B5	21	17	0	0	3	8	645	625
B6	36	36	0	5	5	3	628	606
B7	40	34	3	8	1	4	625	604
B8	39	36	0	4	3	5	627	605
B9	20	20	1	4	8	8	640	618
B10	23	22	2	4	12	13	632	611
B11	19	16	14	12	8	10	628	612
B12	16	15	6	8	6	6	641	621
B13	26	21	1	11	4	8	638	610
B14	25	17	0	1	5	12	639	620
B15	33	27	7	7	1	6	628	610
B16	31	32	0	8	2	0	636	610
B17	28	19	1	1	2	10	638	620
B18	18	14	5	10	4	10	642	616
B19	23	17	5	9	0	4	641	620
B20	20	15	0	4	2	5	647	626
B21	27	22	6	2	4	8	632	618
O	30	15	54	78	16	30	569	527
Total	557	471	112	179	112	179	13937	13471

with SNG and experiment without SNG are 557 and 471 in total respectively. By using SNG, 86 more bird samples are correctly labeled. The false positive values for experiment with SNG and experiment without SNG are 112 and 179 in total. So, total 67 more samples in experiment without SNG are labeled as positives for species when there is no sample of the species.

After calculation of Table 4.5, the test statistics below,

- sensitivity or true positive rate (TPR) which is capability measure of tests to indicate the condition when condition is present and which is formulated as,

$$TPR = \frac{TP}{TP + FN} \tag{4.1}$$

- specificity or true negative rate (TNR) which is capability measure of tests to correctly exclude the condition when the condition is absent and which is formulated as,

$$TNR = \frac{TN}{TN + FP} \tag{4.2}$$

- positive predictive value (PPV) which is the proportion of tests indicating true presents to tests indicating both true and false presents and which is formulated as,

$$PPV = \frac{TP}{TP + FP} \tag{4.3}$$

- negative predictive value (NPV) which is the proportion of tests indicating true absences to tests indicating both true and false absences and which is formulated as,

$$NPV = \frac{TN}{TN + FN} \tag{4.4}$$

are calulated by using the values in Table 4.5 and presented in Table 4.6 As can be seen, the noise removal does not affect the performance of true negative rates but the true positive rates are 83% and 70% for two experiments respectively. The removal of noise provides making 13% more correct labeling in our classification dataset. In addition, the noise removal does not affect the performance of negative prediction, however positive prediction is 7.1% more in experiments with SNG.

As an extra step to classification capability, an offline classification is performed on the features used in online classification. To be able to use the features used in online classification, all the features are stored in SD card beside the labels of online classification results.

The neural network algorithm is used as the offline classification algorithm. To use the algorithm, Weka data mining software is used [60]. To be able to use the neural network algorithm, the multilayer perceptron classifier is selected among the classifier algorithms in

Table 4.6. TPR, TNR, PPV, NPV for Confusion Matrices in Table 4.3 and Table 4.4

Labels	**TPR w/ SNG**	**TPR w/o SNG**	**TNR w/ SNG**	**TNR w/o SNG**	**PPV w/ SNG**	**PPV w/o SNG**	**NPV w/ SNG**	**NPV w/o SNG**
B1	0.61	0.71	0.99	1.00	0.79	0.96	0.98	0.99
B2	0.95	0.52	1.00	1.00	1.00	1.00	1.00	0.99
B3	0.73	0.64	1.00	1.00	1.00	1.00	0.99	0.98
B4	0.79	0.92	1.00	1.00	0.90	0.92	0.99	1.00
B5	0.84	0.68	1.00	1.00	1.00	1.00	1.00	0.99
B6	0.88	0.88	1.00	0.99	1.00	0.88	0.99	1.00
B7	0.98	0.83	1.00	0.99	0.93	0.81	1.00	0.99
B8	0.93	0.86	1.00	0.99	1.00	0.90	1.00	0.99
B9	0.71	0.71	1.00	0.99	0.95	0.83	0.99	0.99
B10	0.66	0.63	1.00	0.99	0.92	0.85	0.98	0.98
B11	0.70	0.59	0.98	0.98	0.58	0.57	0.99	0.98
B12	0.73	0.68	0.99	0.99	0.73	0.65	0.99	0.99
B13	0.87	0.70	1.00	0.98	0.96	0.66	0.99	0.99
B14	0.83	0.57	1.00	1.00	1.00	0.94	0.99	0.98
B15	0.97	0.79	0.99	0.99	0.83	0.79	1.00	0.99
B16	0.94	0.97	1.00	0.99	1.00	0.80	1.00	1.00
B17	0.93	0.63	1.00	1.00	0.97	0.95	1.00	0.98
B18	0.78	0.56	0.99	0.98	0.78	0.58	0.99	0.98
B19	1.00	0.74	0.99	0.99	0.82	0.65	1.00	0.99
B20	0.91	0.68	1.00	0.99	1.00	0.79	1.00	0.99
B21	0.87	0.71	0.99	1.00	0.82	0.92	0.99	0.99
O	0.65	0.33	0.91	0.87	0.36	0.16	0.97	0.95
Total	0.83	0.70	0.99	0.99	0.88	0.80	0.99	0.99

Weka. The parameters of the algorithm are used as in default settings.

In this evaluation, we use the same dataset features as mentioned. All the extracted features with and without spectral noise gating are used for offline classification. In this part, we use 10-fold cross validation technique to obtain the results on offline analysis instead of dividing data into train, cross-validation and test set. In 10-fold cross validation, all the dataset is divided into 10 separate parts. Then, the training and test operations are repeated 10 times by using nine parts as the training and one part as the test set. In every iteration, the unused one part of 10 parts is used as the test part. After 10 iteration, the mean results are calculated for the experiments with and without spectral noise gating.

The mean numbers of correctly labeled samples of both experiments with and without spectral noise gating for neural network classifier are presented in Table 4.7. As can be seen in the Table 4.7, the mean number of correctly classified samples in experiments with noise removal are 60 samples more than in experiments without noise removal.

Table 4.7. Number of correctly labeled and total samples for experiments

	Exp. w/SNG	Exp. w/o SNG
Correctly Classified Samples	1039	979
Total Number of Samples	1116	1116

As can be calculated from the Table 4.7, the true positive rates (accuracy) of both experiments are presented in the first row of Table 4.8. The second row of Table 4.8 shows the accuracies

Table 4.8. Classification accuracies

	Exp. w/SNG	Exp. w/o SNG
Neural Network	0.93	0.87
Minimum Distance Classifier	0.83	0.70

of minimum distance classifier. As can be seen in Table 4.8, the accuracy of neural network algorithm is better for both experiments. In addition, the difference between accuracies of experiments with and without noise removal for neural network classifier is 6%. The same difference is 13% for minimum distance classifier. We can conclude that using more complex algorithms provide us to obtain more robust systems in noisy environments. When using low-weight algorithms, the noise removal operation is essential. However, the noise removal operation can be excluded when using complex algorithms.

4.2. HARDWARE ANALYSIS

In this part of the experimental study, we explain the hardware related analysis. This section is separated into two parts. First, speed and memory analysis of the system is presented, then the power consumption analysis of the system is detailed.

4.2.1. Speed and Memory Analysis

In order to interpret the flexibility of the system to future expansions, system hardware is investigated in both speed and memory perspective. The system uses 116558 (11%) bytes of 1048576 bytes of flash memory. The proposed system with window and FFT size N where N is 2048 uses 80317 bytes (30%) of 262144 bytes of SRAM memory. If necessary in the future, the size, N can be 4096 at maximum. The size 8192 or more causes fail in memory allocation with the same implementation steps as in proposed system.

The determination of the system speed is essential for the stability of ping-pong buffer technique and power consumption. More CPU speed means more processing capability but more energy usage at the same time. For example, the microcontroller in the system consumes 8.78 mA, 12.4 mA, 29.4 mA, 43.3 mA at 1 MHz, 16 MHz, 60 Mhz and 120 MHz respectively when none of the peripherals is on as can be seen in Fig. 4.3 Therefore, we

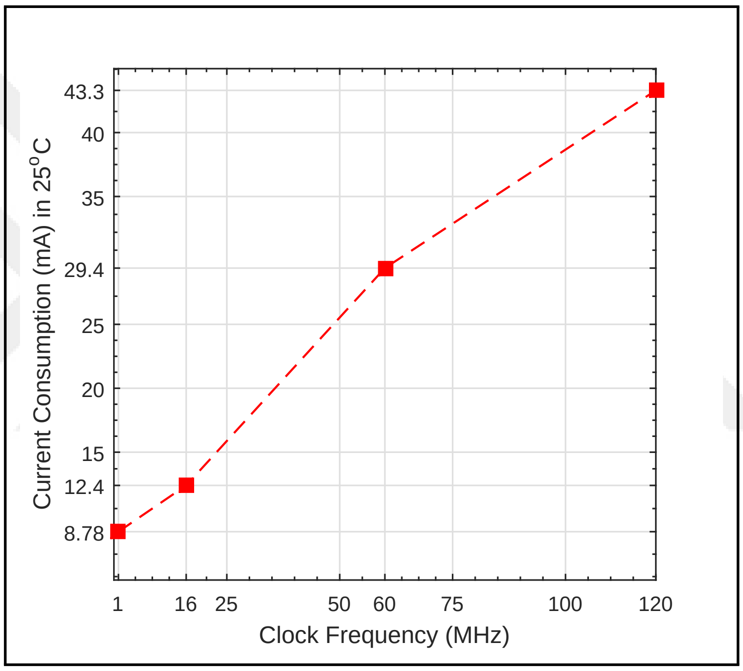


Figure 4.3. The current consumption against clock frequency of the microcontroller

determine the minimum speed ensuring the ping-pong buffering stability. In order to find out the minimum necessary speed, we measured the necessary clock cycles to process one ping or pong buffer. Table 4.9 shows the maximum approximated measured clock cycles for the implementation steps which are noise removal, detection, feature extraction, and classification. In our proposed system, feature extraction and classification steps are never executed together in ping or pong time of ping-pong buffering. Thereby, the maximum necessary cycles are calculated as 2980000 by summing cycles of noise removal, detection, and feature extraction steps. According to necessary clock cycle, the minimum necessary

Table 4.9. Clock cycles of Implementation Steps

Implementation Step	**Clock Cycles**
Noise Removal	1060000
Detection	60000
Feature Extraction	1860000
Classification	14100

clock speed is calculated as,

$$F_{mcu} = \frac{2 \times N_{cycles} \times F_s}{N_{buffer}} \tag{4.5}$$

where N_{cycles} is the total cycles of implementation steps, F_s is the sampling frequency, N_{buffer} is the ping-pong buffer size and F_{mcu} Hz is the necessary clock speed. As can be calculated by Eq. 4.5, the F_{mcu} is 59.6 MHz. Hence, we execute our system at 60 MHz, the closest frequency provided by MCU.

4.2.2. Power Consumption

To be able to find out maximum uninterrupted run-time of the system, the current consumption of each part of the system is measured. The system has three hardware parts as mentioned in Section 2.1. These are microphone circuitry, microcontroller, and storage unit. To use the microcontroller in our system, Tiva Connected C Launchpad is used. Tiva is run at 5 V DC however, the source voltage of the microcontroller is 3.3 V DC. The microphone circuitry is also run at 5 V DC.

The current consumptions of the hardware units are shown in Table 4.10. Minimum current value of the microphone circuitry is obtained under silence and normal conditions. The maximum value is obtained under high level sound volumes like shout.

The current measurement for microcontroller is done on microcontroller's source, not on the Tiva Launchpad at run-time execution. In microcontoller, an analog to digital converter (ADC) and 2 timer modules are activated at run-time.

Table 4.10. Current consumption of hardware units

Hardware Unit	Current Consumption (mA)	
	Min.	**Max.**
Microphone Circuitry	4.66	6.00
Microcontroller Unit	31.50	31.50
Storage Unit	2.88	3.20

The read-write current consumption of the storage unit is measured at run-time of the system. The no read-write current consumption is measured after system stops the implementation steps.

As can be calculated from Table 4.10, average current consumption of whole system at run-time is 39.1 mA. To ponder on the system lifetime at uninterrupted run-time, the proposed system with Duracell MN1500 AA battery is exemplified. Fig. 4.4 presents life-time of Duracell MN1500 AA battery under constant current consumption values, 5 mA, 10 mA, 25 mA and 50 mA is shown. Fig 4.4 is obtained by datasheet of Duracell MN1500

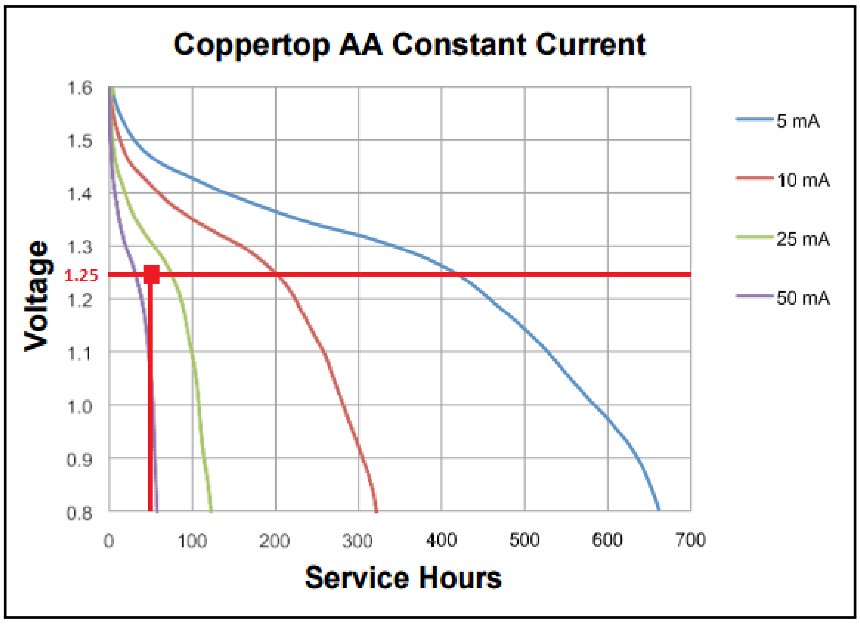


Figure 4.4. Duracell MN1500 AA battery current drain [59]

AA battery [59]. We approximate the lifetime of the system since the exact curve for 39.1 mA is not provided. To supply necessary voltage level, 5 V to proposed system, minimum four of Duracell MN1500 AA batteries is necessary. The horizontal red line in Fig 4.4 indicates the minimum voltage for one of four batteries. The vertical line indicates the approximate intersection point of 39.1 mA curve and red line. As can be seen, the approximate uninterrupted run-time of the proposed real-time system will be around 50 hours.

5. CONCLUSION

In this thesis, we presented a custom-made hardware unit for real-time environmental sound processing and bird sound classification consisting of a microphone, a microcontroller, and a data storage unit. The proposed system can handle data recording and onboard preliminary signal processing, feature extraction, and classification simultaneously by effectively utilizing the ping-pong buffers on the microcontroller unit.

To realize the presented system, we first handled the sound recording in real-time both in hardware and software perspectives. To get analog sound from environment, we designed a microphone circuitry and recorded the outputs of microphone circuitry by a microcontroller. Then, we practised and implemented a solution to remove background noise elements distributed in frequency spectrum of the sound. To reduce noise, we implemented the spectral noise gating method on the microcontoller.

After noise removal, we obtained the denoised signals of the sound. Then, we used energy thresholding method to detect the necessary sound parts in continuous recordings. To be able distinguish the bird species by a reduced amount of data, we implemented the MFCC feature extraction method and extracted the MFCC feature vectors of detected sound samples. Then, we implemented the minimum distance classifier for classification of bird calls. To be able to store the data in file system, we used and updated FatFS library according to our system.

In conclusion, we implemented six steps of data processing which is able to buffer and process data without any loss. We implemented these steps on the limited microcontroller by doing optimizations on memory usage and processing load. To evaluate the system performance, we constructed a dataset which contains only bird calls, test and observe our system in both software and hardware parts. In our tests, we observed that the noise removal enhanced the system performance in both detection and classification steps. In addition, we concluded that using such as system, the power-efficient, long-living and standalone systems can be achieved by code and hardware optimizations.

This study is expandable to more advanced methodologies and optimization techniques. In hardware perspective, the system can be upgraded to more powerful platforms or some other processing units can be added to realize more complex calculations. As an example, there can be a unit to process classification techniques which need more processing loads. In addition, some other sensing modalities (such as temperature, humidity, pressure) can easily be included the system. Thereby, the system can be adaptive to environment using the information obtained by sensing modalities. For example, different noise profiles can be used for different weather conditions. Furthermore, the detection method can also be changed according to environmental events. Addition of a wireless transmission unit to the system make the system capable for remote environmental monitoring. Using WSN and IOT technologies, localization and classification of species can be achieved together. To overcome the power consumption issues and extend the runtime of the system, a comparator unit can be used before recording step on processing unit. The system can be run for only necessary sound events by low power modes. Consequently, the study can be improved by enhancements on both hardware and software.

REFERENCES

1. P. M. Vitousek, H. A. Mooney, J. Lubchenco and J. M. Melillo, Human Domination of Earth's Ecosystems. *Science*, 277:494-499, 1997.

2. W. B. Meyer, *Human Impact On The Earth*, Cambridge University Press, 1996.

3. T. Di Battista, F. Fortuna and F. Maturo, Environmental Monitoring Through Functional Biodiversity Tools. *Ecological Indicators*, 60:237-247, 2016.

4. V. Devictor, R. Julliard, D. Couvet and F. Jiguet, Birds Are Tracking Climate Warming, But Not Fast Enough. *Proceedings of The Royal Society of London B: Biological Sciences*, 275:2743-2748, 2008.

5. K. H. Frommolt, R. Bardeli and M. Clausen, Computational Bioacoustics For Assessing Biodiversity. *Proceedings of The International Expert Meeting On IT-based Detection of Bioacoustical Patterns*, 234, 2008.

6. I. Potamitis, S. Ntalampiras, O. Jahn and K. Riede, Automatic Bird Sound Detection In Long Real-field Recordings: Applications and Tools. *Applied Acoustics*, 80:1-9, 2014.

7. C. J. Ralph, J. R. Sauer and S. Droege, Monitoring Bird Populations by Point Counts. DIANE Publishing, Albany California, 1995.

8. P. Marler, Bird Calls: Their Potential For Behavioral Neurobiology. *Annals of The New York Academy of Sciences*, 1016:31-44, 2004.

9. R. Bardeli, D. Wolff, F. Kurth, M. Koch, K. H. Tauchert and K. H. Frommolt, Detecting Bird Sounds In A Complex Acoustic Environment and Application To Bioacoustic Monitoring. *Pattern Recognition Letters*, 31:1524-1534, 2010.

10. M. A. Acevedo, C. J. Corrada Bravo, H. Corrada Bravo, L. J. Villanueva Rivera and T. M. Aide, Automated Classification of Bird and Amphibian Calls Using Machine Learning: A Comparison of Methods. *Ecological Informatics*, 4:206-214, 2009.

11. W. Chu and D. T. Blumstein, Noise Robust Bird Song Detection Using Syllable Pattern-based Hidden Markov Models. *IEEE International Conference On Acoustics, Speech and Signal Processing*, 345-348, 2011.

12. K. Kaewtip, L. N. Tan, A. Alwan and C. E. Taylor, A Robust Automatic Bird Phrase Classifier Using Dynamic Time-warping With Prominent Region Identification. *IEEE International Conference On Acoustics, Speech and Signal Processing*, 768-772, 2013.

13. A. Henríquez, J. B. Alonso, C. M. Travieso, B. Rodríguez Herrera, F. Bolaños, P. Alpízar, K. López de Ipina and P. Henríquez, An Automatic Acoustic Bat Identification System Based On The Audible Spectrum. *Expert Systems With Applications*, 41:5451-5465, 2014.

14. V. Berisha, H. Kwon and A. Spanias, Real-time Acoustic Monitoring Using Wireless Sensor Motes. *Proceedings of IEEE International Symposium On Circuits and Systems*, 2006.

15. C. Meesookho, U. Mitra and S. Narayanan, On Energy-based Acoustic Source Localization For Sensor Networks. *IEEE Transactions On Signal Processing*, 56:365-377, 2008.

16. H. Wang, C. E. Chen, A. Ali, S. Asgari, R. E. Hudson, K. Yao, D. Estrin and C. Taylor, Acoustic Sensor Networks For Woodpecker Localization. *Optics and Photonics*, 2005.

17. F. Ingelrest, G. Barrenetxea, G. Schaefer, M. Vetterli, O. Couach and M. Parlange, SensorScope: Application-specific Sensor Network For Environmental Monitoring. *ACM Transactions On Sensor Networks*, 6:17, 2010.

18. C. E. Chen, A. M. Ali and H. Wang, Design and Testing of Robust Acoustic Arrays For Localization and Enhancement of Several Bird Sources. *Proceedings of The 5th International Conference On Information Processing In Sensor Networks*, 268-275, 2006.

19. L. Girod, M. Lukac, V. Trifa and D. Estrin, The Design and Implementation of A Self-calibrating Distributed Acoustic Sensing Platform. *Proceedings of The 4th International Conference On Embedded Networked Sensor Systems*, 71-84, 2006.

20. M. Allen, L. Girod, R. Newton, S. Madden, D. T. Blumstein and D. Estrin, Voxnet: An Interactive, Rapidly-deployable Acoustic Monitoring Platform. *Proceedings of The 7th International Conference On Information Processing In Sensor Networks*, 371-382, 2008.

21. I. Mporas, T. Ganchev, O. Kocsis, N. Fakotakis, O. Jahn, K. Riede and K. L. Schuchmann, Automated Acoustic Classification of Bird Species From Real-field Recordings. *Tools With Artificial Intelligence (ICTAI), 2012 IEEE 24th International Conference On*, 1:778-781, 2012.

22. J. Kogan and D. Margoliash, Automated Recognition of Bird Song Elements From Continuous Recordings Using Dynamic Time Warping and Hidden Markov Models: A Comparative Study.. *The Journal of The Acoustical Society of America*, 103:2185, 1998.

23. A. Harma, Automatic Identification of Bird Species Based On Sinusoidal Modeling of Syllables. *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference On*, 5:545-548, 2003.

24. H. Tyagi, R. M. Hegde, H. Murthy, A. Prabhakar, Automatic Identification of Bird Calls Using Spectral Ensemble Average Voice Prints. *Signal Processing Conference, 2006 14th European*, 1-5, 2006.

25. L. N. Tan, G. Kossan, M. L. Cody, C. E. Taylor and A. Alwan, A Sparse Representation-based Classifier For In-set Bird Phrase Verification and Classification With Limited

Training Data. *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference On*, 763-767, 2013.

26. C. H. Lee, C. C. Han and C. C. Chuang, Automatic Classification of Bird Species From Their Sounds Using Two-Dimensional Cepstral Coefficients. *Audio, Speech, and Language Processing, IEEE Transactions On*, 16:1541-1550, 2008.

27. S. Fagerlund, Acoustics and Physical Models of Bird Sounds. *Laboratory of Acoustics and Signal Processing, HUT, Finland*, 2003.

28. P. Åberg, XC153203. Xeno-Canto's Homepage, http://www.xeno-canto.org/153203, [retrieved 27 April 2017].

29. T. M. Aide, C. Corrada Bravo, M. Campos Cerqueira, C. Milan, G. Vega and R. Alvarez, Real-time Bioacoustics Monitoring and Automated Species Identification. *PeerJ*, 1:103, 2013.

30. M. A. Acevedo and L. J. Villanueva Rivera, Using Automated Digital Recording Systems As Effective Tools For The Monitoring of Birds and Amphibians. *Wildlife Society Bulletin*, 34:211-214, 2006.

31. Texas Instruments, "MSP430G2x53 MSP430G2x13 Mixed Signal Microcontroller", http://www.ti.com/lit/ds/symlink/msp430g2453.pdf, [retrieved 27 April 2017].

32. Texas Instruments, "MSP432P401R, MSP432P401M SimpleLinka™ Mixed-Signal Microcontrollers", http://www.ti.com/lit/ds/symlink/tm4c1294ncpdt.pdf, [retrieved 27 April 2017].

33. Texas Instruments, "Tiva™ TM4C123GH6PM Microcontroller Data Sheet", http://www.ti.com/lit/ds/spms376e/tm4c123gh6pm.pdf, [retrieved 27 April 2017].

34. Texas Instruments, “Tiva™ TM4C1294NCPDT Microcontoller Data Sheet”, http://www.ti.com/lit/ds/symlink/tm4c1294ncpdt.pdf, June 2014, [retrieved 27 April 2017].

35. S. F. Boll, Suppression of Acoustic Noise In Speech Using Spectral Subtraction. *IEEE Transactions On Acoustics, Speech and Signal Processing*, 27:113-120, 1979.

36. S. Kamath and P. Loizou, A Multi-band Spectral Subtraction Method For Enhancing Speech Corrupted by Colored Noise. *ICASSP*, 4:44164-44164, 2002.

37. M. Berouti, R. Schwartz and J. Makhoul, Enhancement of Speech Corrupted by Acoustic Noise. *Acoustics, Speech, and Signal Processing, IEEE International Conference On ICASSP’79*, 4:208-211, 1979.

38. D. M. Kiapuchinski, C. R. Erig Lima and C. A. Alves Kaestner, Spectral Noise Gate Technique Applied To Birdsong Preprocessing On Embedded Unit. *IEEE International Symposium On Multimedia*:24-27, December 2012.

39. J. S. R. Jang Audio Signal Processing and Recognition, Jyh Shing Roger Jang’s Homepage, http://mirlab.org/jang/, [retrieved 27 April 2017].

40. S. Fagerlund and A. Harma, Parametrization of Inharmonic Bird Sounds For Automatic Recognition. *Signal Processing Conference, 2005 13th European* 1-4, 2005.

41. C. Kwan, K. Ho, G. Mei, Y. Li, Z. Ren, R. Xu, Y. Zhang, D. Lao, M. Stevenson, V. Stanford, An Automated Acoustic Systemto Monitor and Classify Birds. *EURASIP Journal On Applied Signal Processing*, 2006:52-52, 2006.

42. C. F. Juang and T. M. Chen, Birdsong Recognition Using Prediction-based Recurrent Neural Fuzzy Networks. *Neurocomputing*, 71:121-130, 2007.

43. L. N. Tan, K. Kaewtip, M. L. Cody, C. E. Taylor and A. Alwan, Evaluation of A Sparse Representation-Based Classifier For Bird Phrase Classification Under Limited Data Conditions. *Interspeech*, 2522-2525, 2012.

44. S. Fagerlund and U. K. Laine, New Parametric Representations of Bird Sounds For Automatic Classification. *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference On*, 8247-8251, 2014.

45. E. Vilches, I. A. Escobar, E. E. Vallejo and C. E. Taylor, Data Mining Applied To Acoustic Bird Species Recognition. *Pattern Recognition, 2006. ICPR 2006. 18th International Conference On*, 3:400-403, 2006.

46. J. Cai, D. Ee, B. Pham, P. Roe and J. Zhang, Sensor Network For The Monitoring of Ecosystem: Bird Species Recognition. *Intelligent Sensors, Sensor Networks and Information, 2007. ISSNIP 2007. 3rd International Conference On*, 293-298, 2007.

47. F. Briggs, X. Z. Fern and J. Irvine, Multi-label Classifier Chains For Bird Sound. *arXiv Preprint ArXiv:1304.5862*, 2013.

48. S. Fagerlund, *Automatic Recognition of Bird Species by Their Sounds*, Ph.D. Thesis, Helsinki University of Technology, 2004.

49. CUI Inc, "Electret Condenser Microphone", https://cdn-shop.adafruit.com/datasheets/CMA-4544PF-W.pdf, [retrieved 27 April 2017].

50. Texas Instruments, "LM386 Low Voltage Audio Power Amplifier", http://www.ti.com/lit/ds/symlink/lm386.pdf, [retrieved 27 April 2017].

51. Texas Instruments, "Tiva™ C Series TM4C1294 Connected LaunchPad Evaluation Kit User's Guide", http://www.ti.com/lit/ug/spmu365b/spmu365b.pdf, [retrieved 27 April 2017].

52. SanDisk, “SanDisk Ultra® MicroSDHC™/microSDXC™ UHS-I Card”, http://www.sandisk.com/products/memory-cards/microsd/ultra-class10-for-android/?capacity=8GB, [retrieved 27 April 2017].

53. J. Youngmi and N. McKeown, Doubling Memory Bandwidth For Network Buffers. *Proceedings of INFOCOM 98*, 2:808-815, 1998.

54. The Cornell Lab of Ornitology All About Birds, “Do Bird Songs Have Frequencies Higher Than Humans Can Hear?”, http://www.allaboutbirds.org/do-bird-songs-have-frequencies-higher-than-humans-can-hear/, [retrieved 27 April 2017].

55. L. Group, Understanding FFT Windows. *LDS Group Manual*, 2003.

56. P. Boesman XC281001, Xeno-Canto’s Homepage, www.xeno-canto.org/281001, [retrieved 27 April 2017].

57. Practical Cryptography, “Mel Frequency Cepstral Coefficient (MFCC) Tutorial”, http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/, [retrieved 27 April 2017].

58. K. Paliwal, On The Use of Filter-bank Energies As Features For Robust Speech Recognition. *Signal Processing and Its Applications, 1999. ISSPA ’99. Proceedings of The Fifth International Symposium On*, 2:641-644, 1999.

59. Duracell, “Duracell MN1500 Alkaline-Manganese Dioxide Battery”, https://d2ei442zrkqy2u.cloudfront.net/wp-content/uploads/2016/03/MN1500_US_CT1 .pdf, [retrieved 27 April 2017].

60. Machine Learning Group at the University of Waikato, “Weka 3 Data Mining Software in Java”, http://www.cs.waikato.ac.nz/ml/weka/, [retrieved 5 June 2017].