

École polytechnique de Louvain

Vagus Nerve-Based Seizure Detection Algorithm Analysis and Ultra-Low-Power Implementation

Author: Roman HARDY

Supervisors: Prof. David BOL, Prof. Antoine NONCLERCQ

Readers: Prof. Martin ANDRAUD, Sylvain FAVRESSE

Academic year 2023–2024

Master [120] in Electrical Engineering

Abstract

According to the World Health Organization, more than 50 million peoples have epilepsy, making it one of the most common chronic brain diseases. As such, many medications to treat epilepsy are already available. Proper medication and diagnosis would make up to 70% of people with epilepsy seizure-free, mostly with antiepileptic drugs. However, as drugs are not always conclusive, research is conducted to improve them and bring new solutions. Amongst theses new ideas, a new type of closed-loop neurostimulator is investigated, using the vagus nerve to both predict the seizures and counter them. This work looks at the feasibility of a new algorithm to detect epileptic seizures based on the vagus nerve. The method was partially tested before, but never from an implementation point of view. This work will then put a strong focus on the challenges, possibilities and requirements coming from this new algorithm. To this end, the algorithm is implemented in C and characterized on an ultra-low-power microcontroller, allowing real measurements of power consumption and showing the memory limitations of an embedded device.

Acknowledgements

I would like to thank Prof. David Bol and Prof. Antoine Nonclercq who offered me the opportunity to glance at the complex yet fascinating research in neurobiology. More than a thesis topic, they gave me the chance to look at other applications of engineering that I had not encountered before.

Alongside Prof. David Bol I express my gratitude to Sylvain Favresse, they both guided me through our numerous meetings this year. They were always available and listening and despite their busy schedules, were often faster to reply to emails than I was. Their advices proved me more than once that after five years at university, you still have a lot to learn.

Thanks to Rémi Dekimpe for his recommendation and help with the configuration, programming and usage of Apollo development board, which I shall not forget to return to him by the end of the summer.

At last, a kind word for my friends and relatives, who helped me during these fives, sometimes long, years. Group C of the master's project in electrical engineering was the best team I ever worked with. My parents, who supported and encouraged me along the way. My sisters, showing a relentless curiosity in the progress of my work, yet facing often the same answer. And, despite her moderate interest in this thesis, my girlfriend.

Contents

Introduction	VII
1 About Refractory Epilepsy	1
1.1 Epilepsy	1
1.1.1 Neurons and Action Potentials	1
1.1.2 Seizures	2
1.1.3 Monitoring Epilepsy	3
1.2 Treatments	4
1.2.1 Antiepileptic Drugs	5
1.2.2 Surgery	5
1.2.3 Stimulators	6
1.2.4 Ethical concerns	10
1.3 Motivation for closed loop VENG stimulation	11
2 Analysis of the seizure detection algorithm	12
2.1 Analysis framework	12
2.2 Experimental data	13
2.3 Action potentials detections	13
2.3.1 Algorithm description	14
2.3.2 Parameters	16
2.3.3 Clusters	17
2.4 Seizure detection	21
2.4.1 Detection metric	22
2.4.2 Classifier	22
3 Ultra-low-power implementation	25
3.1 Design choices	26
3.2 Fixed point arithmetic	28
3.2.1 Operations on fixed point number	30
3.2.2 Programming with fixed point arithmetic	32

3.3	Implementation details	32
3.3.1	Global functioning	32
3.3.2	SoC configuration	33
3.3.3	Memory usage	34
3.4	Optimizations	37
4	System characterization	40
4.1	Experimental setup and methods	40
4.2	Resources analysis and bottlenecks	41
4.2.1	Execution time	42
4.2.2	Power consumption	45
4.2.3	Comparison with other works	48
4.3	Further improvements	51
	Conclusion	53
	A List of implemented Make targets	55
	B Detection metric of the six subjects	58

Acronyms

ADC analog to digital converter

AED antiepileptic drug

AFE analog frontend

ANT–DBS deep brain stimulation of the anterior nucleus of the thalamus

AP action potential

CMSIS common microcontroller software interface standard

CNS central nervous system

DBS deep brain stimulation

DMA direct memory access

DSP digital signal processing

ECG electrocardiography

EEG electroencephalogram

EVB evaluation board

FPR false positive rate

GPIO general purpose input/output

iEEG intracranial electroencephalography

ISR interrupt service routine

NCC normalized cross-correlation

RE refractory epilepsy

RNS responsive neurostimulation

SMU source and measurement unit

SoC system on a chip

TPR true positive rate

VENG vagus nerve electroneurogram

VNS vagus nerve stimulation

Introduction

Epilepsy is one of the most common chronic brain diseases, affecting more than 50 million people of all ages, worldwide [1]. Epileptic patients are subject to unexpected seizures, during which their sight, hearing, awareness, and motion can be affected, causing them discomfort and possibly leading to physical injuries. Besides, some patients suffer from superstitious beliefs associating epileptic seizure with demonic possession, witchcraft, sorcery, or similar and tend to be marginalized, having more difficulties finding a job or having a driver's license.[2].

Drugs successfully remove these chronic events in 70 percent of the affected people [3]. For the ones where medicines are ineffective, the remaining options are either brain surgery or neuromodulation devices [4]. The latter works by sending a small electric current to the nervous system. Continuous stimulation in the brain or on a nerve reduces the battery life and may cause several side effects to the patient, including paresthesia, implant site pain, shortness of breath, cough and headache [5], [6]. Therefore, closed-loop systems are developed to only stimulate when deemed necessary. The seizures are either manually triggered by the patient or detected by a real-time analysis conducted on the implant. The current standard for automatic detection of seizures is the electroencephalogram (EEG), a signal often sensed directly in the brain. To avoid this invasive procedure, another option is to have electrodes placed on the vagus nerve. Moreover, it is a deliberate choice to avoid using EEG or other brain sensing interfaces to avoid the ethical challenge that may arise from an abuse of these technologies [7].

An implanted device to monitor epilepsy, such as considered in the context of this master thesis, is composed of precise cuff electrodes, a low noise analog frontend (AFE), and an embedded platform. The electrode and AFE have high constraints, as the signal sensed is in the μV range [8]. On the embedded platform, it involves the development of new algorithms to detect seizure onsets based on the vagus nerve electroneurogram (VENG) signal. The requirements for such a program are multiple and should all be optimized for the well-being of the patient.

It needs to be accurate while avoiding false positives that cancel the benefits of closed-loop systems. The detection delay is important, as a late stimulation could fail to counter the seizure [9]. Finally, as it aims to fit inside a medical implant, the power consumption should be reduced to its minimum, as well as the memory usage. Drug pumps, neurostimulators, or pacemaker all fall below the mW limit and power sources for implanted devices can be as low as 1 μ W [10].

In [11] Raffoul et al. described a template matching algorithm to detect action potentials (APs) on the vagus nerve and provided a MATLAB [12] implementation of it. This algorithm generates new templates, obtained upon the clustering of the detected APs, and therefore adapts to the electrode setup and patient. Moreover, the clustering allows to discriminate the APs based on their form and the nerve fiber they originate from. Using the APs rate and amplitude, a detection metric is obtained as proposed by L. Stumpp et al. in [13]. This work aims at evaluating the possibility of detecting epileptic seizures by sensing the vagus nerve based on the template matching technique and detection metric mentioned above. Previous works were focused on the algorithmic part, overlooking the real-time, embedded aspect of the project. Here, the latter is thoroughly investigated to highlight the requirements of such a system and the need for improvements or weaknesses of the algorithm, from a computing performance perspective. Experimental data of induced seizure in rats is provided by the Université Libre de Bruxelles to assess this pipeline functionality.

This master thesis is divided into four chapters, between this introduction and a conclusion.

- Chapter 1 details the medical background of this master thesis: epilepsy, vagus nerve and APs. It also includes an overview of the treatments available, with an emphasis on the approved neuromodulation therapies, so that the reader can better understand the context of this work.
- Chapter 2 is about the algorithm studied in this work. The analysis framework used to obtain the classifier performances is introduced, then the algorithm detection algorithm and the detection metric. Parameters influencing the detection metric are discussed.
- Chapter 3 conducts the ultra-low-power embedded implementation. Starting with an explanation of the design choices and the development platform selected, followed by a resource analysis of the low-level implementation.
- Chapter 4 targets the system characterization of the experimental program on an ultra-low-power microcontroller. From the power consumption to

the algorithm complexity (theoretical and observed), the implementation is scanned, with a focus on each stage as well as the application end-to-end. The bottlenecks and consumption key factors are described. It ends with a comparison with other works and possible improvements.

Chapter 1

About Refractory Epilepsy

This first chapter lays the foundation and motivation for a closed loop VENG device. It introduces the main concepts and mechanisms of epilepsy, alongside the possible treatments.

1.1 Epilepsy

To understand what is epilepsy, one first needs to know what a seizure is and what triggers them. This first section focuses on the neurobiological and clinical aspect of epilepsy.

1.1.1 Neurons and Action Potentials

Figure 1.1 depicts a neuron, with the main features of interest labelled: the cell body, the axon and the dendrites. Ion concentrations in the neuron and in the extracellular medium create a potential of around -60 mV between both sides of the cell membrane [14]. If the potential in the cell crosses a certain threshold, the membrane first depolarizes, rising its potential, then repolarizes to regain its original state. This transient electrical pulse is called an action potential (AP). It travels on the membrane, including the axon, and reach other neuron dendrites, themselves being stimulated, leading to a possible new AP. The junction between a neuron's dendrite and another neuron's axon is called a synapse, where neurotransmitters are passing through from one neuron to another. Neurotransmitters either excite the next neuron or inhibit it, influencing its behavior. This allows information to flow inside the neural network.

The transmembrane potential at rest is crucial for good operation. Would

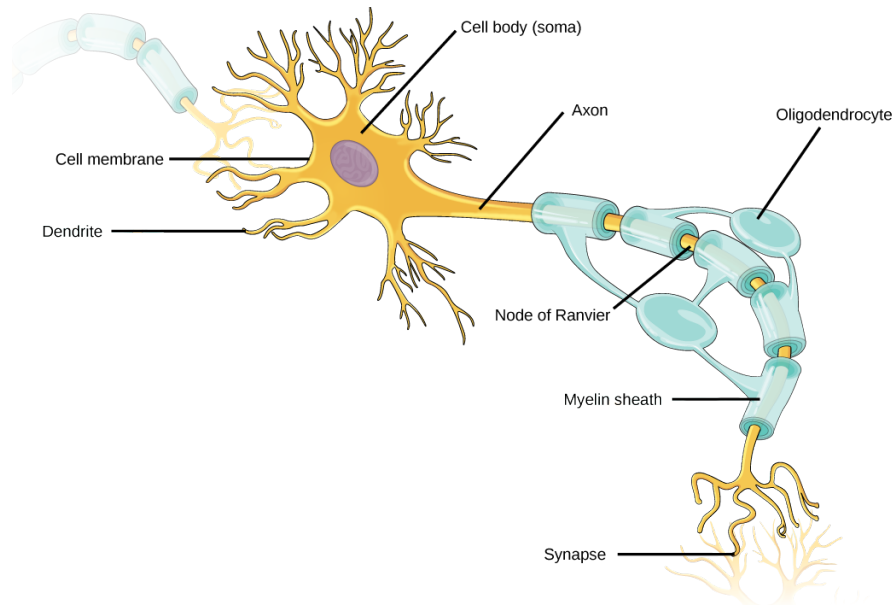


Figure 1.1: Schematic of a neuron [15]

it be too low, the threshold would increase and prevent the depolarization process. On the contrary, a higher potential would decrease the threshold leading to an excessive amount of AP. Similarly, defects in the membrane could lower down the threshold [14].

1.1.2 Seizures

A seizure is characterized by an abnormal electrical activity in the brain. It overreacts and fires an excessive number of AP. There is often a physical manifestation (motor symptom) such as twitching of limbs, spasms, or repeated body movement. Non-motor symptoms includes changes in sensation, changes in emotions, stillness, goosebumps, increase of heartbeat [17].

In 2017 the International League Against Epilepsy introduced a new classification scheme for the seizure types. The root of this classification is the seizure onset, that can be focal, generalized or unknown if one was unable to identify it. Focal onset refers to seizures emerging from one side of the brain, but is not restrained to it and can spread to both sides. Generalized onset is localized in both hemispheres of the brain at the same time. Then, further characterization is made based upon a list of motor and non-motor symptoms. In addition, focal seizure includes whether the person is aware of itself and his surroundings (retained awareness) or not (impaired awareness). In many documents, including this one,

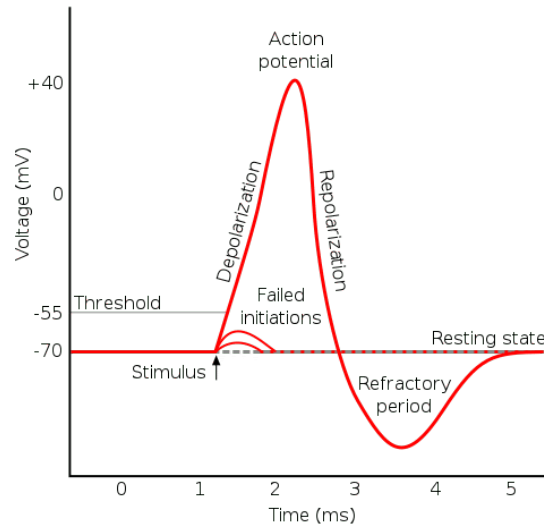


Figure 1.2: Action potential illustration [16]

focal onset (resp. generalized onset) is shortened to focal (resp. generalized). The terms generalized and focal epilepsy are then used.

Seizures occur when the inhibitory and excitatory circuits are unbalanced. For someone to be diagnosed as epileptic, they need to have suffered from two unprovoked epileptics seizures, or one with a high probability of a second one. Epilepsy put an emphasis on the *unprovoked* characteristic of the seizure. Under special circumstances, such as brain injury, intoxication or high fever [18], anyone could encounter a seizure. Epileptic seizures are unpredictable, making them difficult to counter and understand. Also, the term epilepsy encompass a variety of manifestations, syndromes and causes with as many possibles treatments. It is not one single disease but rather a family of various conditions.

1.1.3 Monitoring Epilepsy

In order to pinpoint the type of epilepsy or detect oncoming seizures, one should be able to monitor it. The first step, important in the classification process, is the visual witness. A relative of the patient, present during a seizure can relate the symptoms he saw to help the physician.

To further analyze the patient condition, an EEG is usually performed. This non-invasive procedure consists of placing a mesh of electrodes, following a well-defined placement, on the patient's head and record the electrical activity. A typical recording is at least half an hour long, during which the patient keeps his

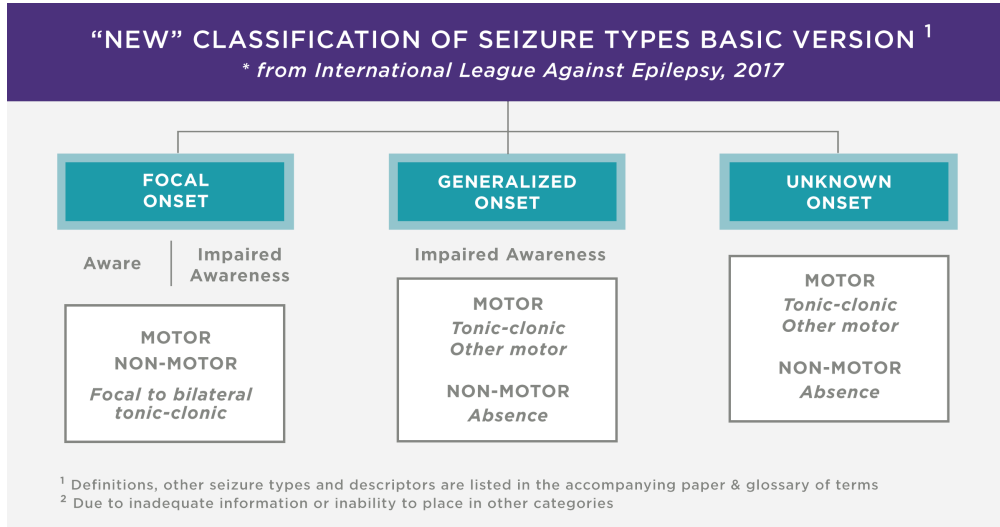


Figure 1.3: 2017 International League Against Epilepsy classification scheme [17].

eyes closed most of the time. If the results are inconclusive, the recording may be done while the patient is asleep [19]. Upon results evaluation, anomaly in the signal can be associated with epilepsy or help to make the distinction between focal or generalized epilepsy. However, this tool alone cannot establish an epilepsy diagnosis, as non-epileptic people can show abnormalities in the EEG while some epileptic patients do not [20].

Intracranial electroencephalography provides a better spatial resolution than EEG, but requires a surgical procedure to place the electrodes. It is an effective method to localize with precision the seizure onset zone, alongside neuroimaging [21]. However, the surgery can lead to negative side effects, such as hemorrhages or infections [5].

VENG consist of cuff electrodes placed around the nerve. The vagus nerve, also named cranial nerve X, contains efferent and afferent fibers. Efferent neurons send motor information from the central nervous system (CNS) to the body and afferent neurons transmit sensations from the body toward the CNS. The vagus nerve afferent fibers carry information from the cardiovascular, respiratory and digestive functions and is therefore prone to seizure detection [13].

1.2 Treatments

There is no cure for epilepsy. Except for surgery, all other available treatments are palliative, unable to provide seizure freedom for most people. The first option is

almost always antiepileptic drugs (AEDs), used by around 70 % of patients. In 2015, a survey the United States showed that only 44 % of adult under AEDs were seizure free in the past year [22]. For peoples unsatisfied by medications or the ones unresponsive to it, neuromodulation device or surgery might be an option. The Epilepsy Foundation also mentions the following therapies, not discussed here: dietary therapies, seizure rescue therapies, alternative therapies and clinical trials [3].

1.2.1 Antiepileptic Drugs

The term *antiepileptic* might be inappropriate, as it does not prevent epilepsy but the seizures. They are indeed often called antiseizure medication. These drugs interfere with the excitatory and inhibitory mechanism, to keep them balanced. One way to achieve this is through the neurotransmitters in the synapses. The medications are swallowed and enters the bloodstream in the stomach, then flowing to the brain. They are gradually removed by the liver and kidneys.

Based on the seizure type, syndromes, lifestyle, comorbidities, other medications, and patient preference a specialist tries to prescribe the most adequate AED from the broad variety of existing ones [23]. This choice must be tailored for each patient and if not conclusive, it is worth trying other AEDs, even two simultaneously. After two nonsuccessive trials of AEDs suggested by a neurologist, the patient is diagnosed with refractory epilepsy (RE), also named drug-resistant epilepsy. From there, further investigation in an epilepsy center is advised.

1.2.2 Surgery

The goal of epilepsy surgery is to free the patient from seizures with a brain resection¹ of the seizure onset zone. This treatment is only available for focal onset seizures and when the seizure onset zone is clearly identified. Risks associated with the procedure include bleeding, infection, brain injury and cognitive deficits [3], [23]. These advert effects and general concern about brain surgery result in its underutilization. A 2015 review of epilepsy surgery reported a seizure freedom of 34 % to 76 % after the operation [24]. This great variation comes from the lack of cohesion in the studies reviewed. Seizure freedom is not always properly defined, studies are subject to patient selection bias and are not conducted over the same follow-up periods.

¹Resection: the act of removing any part of the body in a medical operation (Cambridge Dictionary)

1.2.3 Stimulators

Neuromodulation therapy involves a device to send small electrical current to the nervous system to prevent or abort the seizures. There is a few neurostimulators available on the market, approval depending on the country. Three devices, with clearance in the United States by the U.S. food and drug administration are described here, to provide an overview of the current situation.

Table 1.1 summarizes the characteristics of the presented devices, alongside expected results. Table 1.2 presents the seizure reduction for patient with focal epilepsy. The efficacy of responsive neurostimulation (RNS) and deep brain stimulation of the anterior nucleus of the thalamus (ANT-DBS) is similar and above vagus nerve stimulation (VNS), reporting a median reduction in seizure frequency at three months of 43 % for RNS and 42 % for ANT-DBS versus 31 % for VNS. However, VNS efficacy might increase with time and does not require brain surgery. In three years, seizure reduction for focal epilepsy increased by 20.6 % for VNS, versus 2.1 % for RNS and 5.4 % for ANT-DBS. To select a neuromodulation device, one can also take into account the type of stimulation, the stimulation target and the data collected, to better understand the patient situation. Moreover, large-scale comparison between the system are still needed to provide better evidence in the choice of therapies [5].

VNS Therapy

VNS Therapy, developed by LivaNova, was the first device approved by the food and drug administration to treat RE in 1997, with the first patient implanted almost ten years prior, in 1988. In 2021, more than 125,000 patients have been treated with the VNS Therapy system [26] through five generations of implants (figure 1.4). It was originally designed as an open-loop system programmed to typically stimulate for 30s every 5 min, this schedule being adjustable by a doctor to better accommodate to the patient. New version includes an automatic detection of heart rate changes, making the system closed-loop and there is also a manual activation by passing a magnet in front of the detector. The device is implanted in the upper-left chest, with an electrode on the left vagus nerve (figure 1.5).

Surgical complications have been reported in 8.6 % of case in studies, with mainly infections, hematoma, and vocal cord palsy [27]. They are usually related to the stimulation, meaning that reprogramming the stimulation schedule can help reduce them. They are expected to improve over time and does not cause CNS side effects such as fatigue, psychomotor slowing, soreness and stress [28].

Criteria	VNS Therapy	ANT-DBS	RNS
Stimulation target	Left vagus nerve	Anterior nucleus of the thalamus	Ictal onset zone
Stimulation type	Scheduled, closed loop (heart rate), on demand	Scheduled	Closed loop (ictal iEEG)
Data collected	Activation, prone position, heart rate	Local field potentials (live streaming in clinic)	Seizure frequency, ictal iEEG discharge
Median reduction in seizure frequency at 3 months	31 %	44 %	42 %
50 % responder rate at end of double-blind phase	26 %	30 %	29 %
50 % responder rate at 1 year	37 %	43 %	44 %
Seizure free at 2 years of follow-up	0 %	13 %	9 %

Table 1.1: Comparison of stimulator device characteristics and results. Adapted from [5]. The 50 % responder rate is defined as the proportion of subjects with a reduction of at least 50 % of their baseline seizure frequency.

Year	RNS	ANT-DBS	VNS Therapy
1	66.3 %	58.4 %	32.9 %
2	56.0 %	57.5 %	44.4 %
3	68.4 %	63.8 %	53.5 %

Table 1.2: Seizure reduction for stimulator devices on focal epilepsy [25]

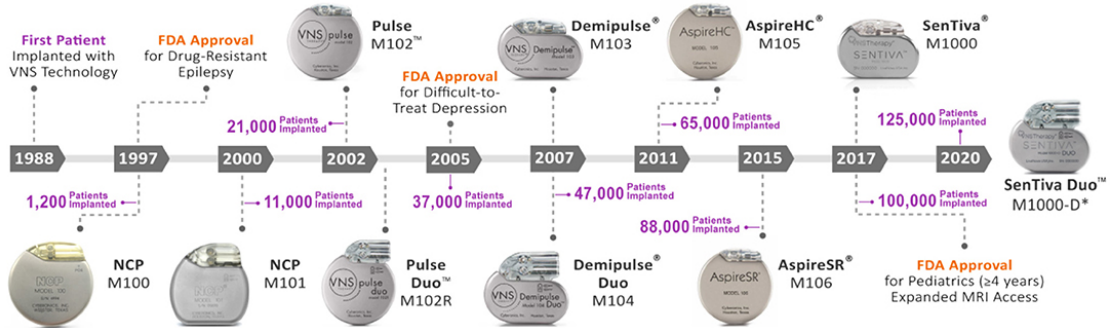


Figure 1.4: Evolution of the VNS Therapy system. From [29]

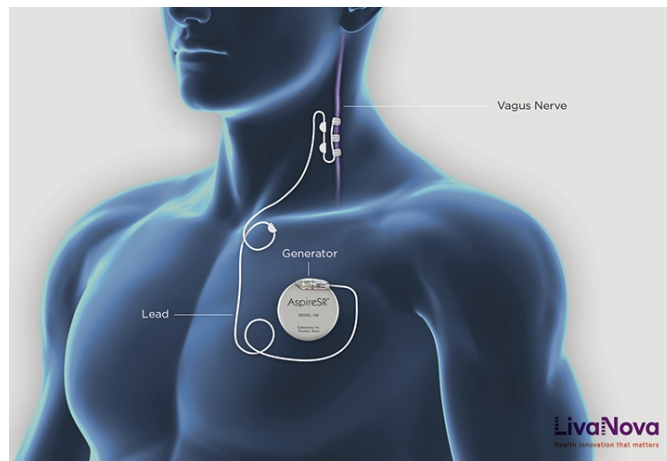


Figure 1.5: LivaNova AspireSR implant for VNS Therapy [30]



Figure 1.6: Medtronic Percept PC implant [32]

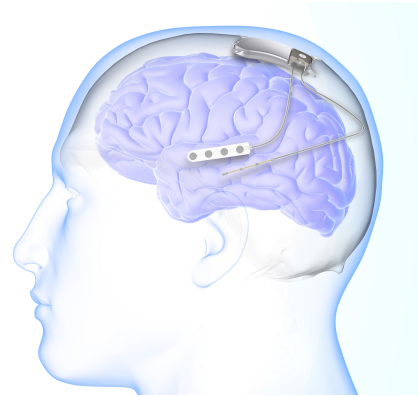


Figure 1.7: NeuroPace RNS System [33]

Deep Brain Stimulation

The mechanisms of ANT-DBS are not completely known but are attributed to the interference with depolarization process, neurostimulators and synchronization of brain activity and epileptic network [5], [6]. Medtronic has released a open loop ANT-DBS device for RE, constituted by a chest or abdominal implant connected to electrodes in the brain. Medtronic BrainSense technology allows the recording of local field potentials during stimulation for analysis by a clinician.

The most common adverse effect is implant site infection with about 20 % of individuals reporting implant site pain. A long-term study on ANT-DBS for RE also reported depression and memory problems, but a direct correlation with ANT-DBS is difficult to establish due to the high prevalence of these side effects with people suffering from RE in general [31].

Responsive Neurostimulation

NeuroPace RNS System aims to monitor brain activity, identify seizure patterns to counter them and record EEG data to report to a clinician. The device is implanted in the cranium with one stimulating electrode deep in the brain and one or two sensing electrodes at identified seizure onset zones (figure 1.7).

After a 9 years follow-up study of prospective efficacy and safety on RNS for focal epilepsy reported a risk of infection associated with the procedure of 4.1 %. Adverse effect of depression and suicidality over the 9 years were high (25 % and

9.8 %) but the majority of these events (82 % for depression) were not considered as related to the device due to prior history of the subjects [34].

1.2.4 Ethical concerns

The usage of neurostimulators, especially implanted in the brain, raises many ethical concerns. In this section, deep brain stimulation (DBS) is not related to RE but treatment of other neurological disorders, such as Parkinson's disease, as it was the scope of the study in [35].

The first concern is inherent to any medical procedure: the risk-benefit balance. Are the expected risk and adverse effects proportional to the expected benefit ? For the devices listed above, the risk-benefit balance can be considered valid. However, this implies that if other treatment showing a better balance are available, they should be considered in priority. Second, the patient must be consent to the operation while having realistic expectations for the outcome, not biased by unrealistic reports. The bias of a "last resort" solution should also be avoided, selecting a procedure out of despair [35]. This again applies to all listed treatments.

Other ethical questions targets specifically the usage of DBS or EEG and iEEG. For instance, side effects of DBS includes changes in behavior, mood or cognition may alter the patient "personal identity". Drawing conclusion on this point is difficult, as the concept of "personal identity" in itself is still up to debate [7], [35].

Outside the scope of medical treatment, device recording and trying to understand brain activity introduces agency² concerns, although it may at first be an empowerment tool, allowing people to gain independence through EEG assistive technologies. The main issue addressed in [7] concerning agency is that research and improvement on EEG device could allow the understanding of human decision-making process. For marketing purpose, it means that companies could use extremely powerful cognitive biases to trick the consumer on an unprecedented level of manipulation.

All of this remain hypothetical, as research on this matter is still scarce. From a certain point of view, it makes it even more dangerous, as the possible deviations of this technology has undefined and unpredictable consequences. Until further debate and conclusion are drawn, one's take on the matter could be to

²"Agency is understood by ethicists as the ability of the individual to choose its own actions" [7].

avoid this field completely. This is the approach taken at UCLouvain in the context of this work.

1.3 Motivation for closed loop VENG stimulation

VNS therapies already showed successful results in its commercial applications. The procedure is less invasive than intracranial stimulation and avoid any ethical concern about iEEG nor brain stimulation.

However, it still uses a tachycardia-based sensor for its closed-loop application. In [36] L. Stumpp demonstrated the possibility to detect induced seizure in rats based on the vagus nerve activity. Moreover, they obtained superior detection performance compared to a heart rate-based metric. This work further investigates a vagus-nerve seizure detection in perspective of a VNS, removing the need for another sensor, thus simplifying the surgery and associated risks.

Chapter 2

Analysis of the seizure detection algorithm

This chapter describes the algorithms presented in [11] and [13]. The analysis is conducted from a functional point of view, without entering into the details of the MATLAB implementation. The influence of some parameters is discussed, as neither this processing pipeline nor dataset was experimented before.

The computational complexity, speed, processor and memory usage is not considered. The MATLAB codes are designed for research purposes, prioritizing ease of configuration and development over sheer performances. A resource analysis is then irrelevant.

First the goal to reach with the algorithm and metrics to quantify its quality are introduced as well as the experimental data used to test the pipeline. Then, the APs detection algorithm and its parameters are described. Since it is the most demanding part of the pipeline, possible solutions to alleviate this computational burden are discussed. At last, the seizure detection algorithm is defined and combined with the previous APs detection, to complete the detection process. The impact of some parameters on the classification quality is observed and a more general statement about the whole pipeline result is finally drawn.

2.1 Analysis framework

The objective is ideally to predict seizure onsets, or at best to determine if one is occurring. A program matching input data into categories is called a classifier. For this application, only two options are considered: a seizure is taking place, or it

is not. Such a classifier is called a binary classifier, as it can either report *true* or *false*.

The main metrics to assess the classifier performances are the true positive rate (TPR) and false positive rate (FPR), defined as

$$\text{TPR} = \frac{\text{true positives}}{\text{interval with seizures}} \quad (2.1)$$

$$\text{FPR} = \frac{\text{false positives}}{\text{interval without seizures}} \quad (2.2)$$

in [13]. The intervals with seizures correspond to the number of seizures but could be interpreted as the total seizure duration. Then, the intervals without seizures correspond to the number of seizure plus one. False (resp. true) positive are declared if at least one seizure is reported during an interval without (resp. with) seizure. These metrics are useful if one wants to achieve a perfect classification, with a TPR of 1 and FPR of 0, as is done in [13]. However, when perfect classification is not possible, it does not give a lot of feedback. Two classifiers can obtain the same FPR even though one reports twice as many seizures outside the expected timestamps. Also, in this context, a random classifier would have both a TPR and FPR of 1. Each interval contains multiples samples of the detection metric and thus the classification, meaning that a classifier choosing a random output on each iteration would report multiples seizure in each interval.

2.2 Experimental data

The Université Libre de Bruxelles provided six recordings of induced seizures in animals subjects. Each recording last approximately one hour and contains between one and fives seizures. A short overview of the experimental data and subjects is available in table 2.1. The sampling frequency for the VENG data is 80 kHz, but was later downsampled to 20 kHz with MATLAB’s `resample` function, for the embedded application. It should not impact the results, as the first step in the processing pipeline is to filter the signal in a frequency band lower than 10 kHz, meaning that the Nyquist frequency is respected. Alongside the VENG, electrocardiography (ECG) and EEG recordings are also available.

2.3 Action potentials detections

The algorithm proposed by Raffoul et al. in [11] is based on template matching. This technique identifies a specific sequence, the template, in a larger signal, here

Animal [number]	Age [week]	Weight [g]	Seizures [#]	Seizures duration (mean) [s]	Seizures duration (std) [s]
P0036	12	350	5	52.9	17.2
P0048	8.5	260	3	57.1	26.2
P0049	8.5	215	3	31.2	0.7
P0058	10	320	4	70.8	7.6
P0059	11	306	5	40.2	9.4
P0062	9	330	3	54.8	8.8
<i>Average</i>	<i>9.8</i>	<i>297</i>	<i>4</i>	<i>51.2</i>	<i>11.7</i>

Table 2.1: Characteristics of the experimental data subjects

the VENG data. The main features of this algorithm are the adaptive templates based on the recording setup and the ability to classify the APs based on their shape, reflecting the type of fibers activated. This method was validated on one experimental subject, comparing the detected APs frequency and amplitude modulation with the subject respiratory rate, derived from the ECG recording.

2.3.1 Algorithm description

The algorithm’s execution is divided into six sequential steps, detailed below. To better understand the relation of the different blocks, a flowchart of the algorithm is presented in figure 2.1.

1. The input signal is processed by two independent filters, resulting in two filtered VENG signals, denoted as the generic and specific signals. In the current configuration, both filters are the same bandpass filter between 300 Hz and 3000 Hz.
2. By default, a generic template is formed based on the system parameters. The user can also add their own template(s).
3. A first APs detection (generic detection) is performed over the generic VENG

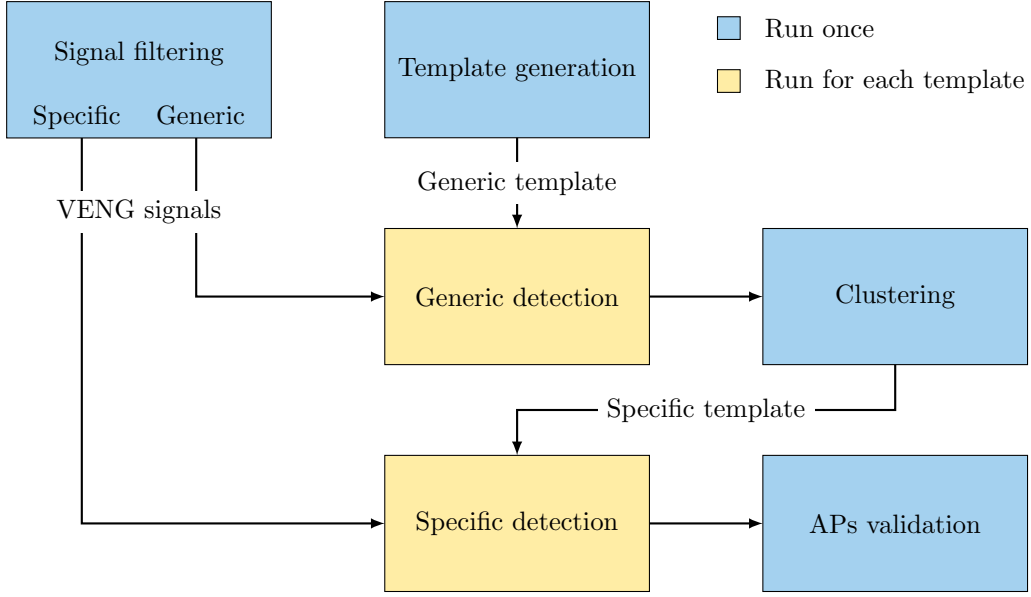


Figure 2.1: Flowchart of the AP detection algorithm

signal, based on the template(s) defined at step 2. The normalized cross-correlation (NCC) between each template and the signal is computed. To do so, both the template and the sub-part of the signal being correlated are normalized in amplitude. This operation can be reduced to

$$\text{NCC}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{|\mathbf{x}| |\mathbf{y}|} = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2} \sqrt{\sum_i y_i^2}} \quad (2.3)$$

where $\mathbf{x}$ is the template and $\mathbf{y}$ a moving window in the signal, the same size as $\mathbf{x}$.

The correlation's absolute value is retained to have a value between 0 and 1. From this new signal, peaks (local maxima) are extracted. They need to pass a threshold in amplitude and be spaced enough from each other. These peaks define the location of the AP in the filtered signal. At last, APs amplitudes are compared with the local RMS amplitude on a 0.5 s window, rejecting the ones too small (considered as noise) or too large (considered as artifact).

4. APs are clustered in the time domain to obtain new templates, with the K-means method and the metric set to correlation. The initial number of clusters is set to 2, but increases if the mean discorrelation inside one of them is too high. It is an iterative process, rising the number of clusters by

1 and starting over each time the decorrelation threshold is passed. Once done, clusters with less than 1 % of the total population are discarded and not considered to be physiological.

5. A second APs detection (specific detection) is executed, based on the clusters found during the clustering process. The method is similar, but the settings (thresholds, minimal distances) may differ.
6. APs can be assessed over other criteria and their characteristics are stored. The main features of interest are their temporal location and amplitude.

The MATLAB implementation is publicly available on GitHub¹. Running this version on the hour-long recordings would be impossible, or highly impractical at best, due to the memory and time needed. Thus, Sylvain Favresse, PhD student at UCLouvain, who had previously worked on the algorithm, segmented the program execution. Instead of working on the full signal, windows of 30 min are used, including a 5 min overlap on each side. At the edges of the signal, data is copied and mirrored.

2.3.2 Parameters

The AP detection algorithm is highly customizable through various parameters, acting at different stages of the process:

1. The cross-correlation threshold for the generic detection.
2. The cross-correlation threshold for the specific detection.
3. The generic template for the first detection.
4. The mean decorrelation threshold for the clustering.
5. The minimum and maximum threshold amplitude-to-RMS ratio imposed on the APs.
6. The bandpass filter range for the generic detection.
7. The bandpass filter range for the specific detection.

The **first four parameters** are mentioned as the major parameters in [11] and are not discussed in this work, leaving their values untouched. The **fifth parameter**, concerning the requirement in the amplitude-to-RMS ratio for the APs is the one of interest in this section, as it showed the biggest impact on the results. The

¹<https://github.com/BEAMS-Biomechatronics/AP-Detection-Algorithm>

sixth and seventh parameters are not considered, keeping the 300 Hz – 3000 Hz range suggested in [11].

The initial value for the minimum amplitude-to-RMS ratio was 0.75 and 5 for the maximum. However, these were too constraining for this application, as the number of detected APs during a seizure was often lower than expected. An example of this is presented in figure 2.2, on the upper graph. On the second, third and fifth seizure, the number of APs drops significantly, presumably because of their amplitude, greater than expected during a seizure. This was confirmed by removing the constraint on the amplitude, effectively keeping all detected APs, regardless of their amplitude. The average number of detected APs per second rises from 105 to 119, corresponding to a 13% increase. During the three seizures mentioned above, the number of APs detected is now above the baseline, as expected. However, high spikes are present in the signal plotted, around 1500 s, 1700 s and 2400 s, assumed to be due to artifacts misdetected as APs. At last, the first seizure, that stood out in the first case, is now hidden in the average number of detections. This could be an unwanted side effect or a true reflection of the reality.

Figure 2.3 shows that there is indeed many APs detected with an amplitude greater than five times the background RMS value. Although the mean values during and outside seizures tends to be similar, the amplitude shows a more important variation across the subjects during seizures, meaning that for some cases, but not all, it could be profitable to keep APs with greater amplitudes. The range of accepted amplitudes, with respect to the RMS value, is fixed from 1 to 16. The sharp spikes mentioned can be filtered by the next step of the detection process.

2.3.3 Clusters

The clustering in the MATLAB code runs with dozens of thousands of APs. It is obvious that having in memory that many arrays to represent the APs will not be feasible on the embedded platform. Consequently, the duration the clustering will be reduced. It is then interesting to look at the shape and variations among the inferred centroids and try to estimate if reducing the amount of data available for the clustering will impact the results. To this end, centroids are compared between each other in the default configuration and when the clustering windows are reduced.

As the APs can be pointed upward or downward, the centroids often comes in pairs, one being the opposite of the other. To compare the centroids, the absolute Pearson cross correlation coefficient is used, the same metric as for the clustering, except for the absolute value, added here to account for the opposite centroids. Each

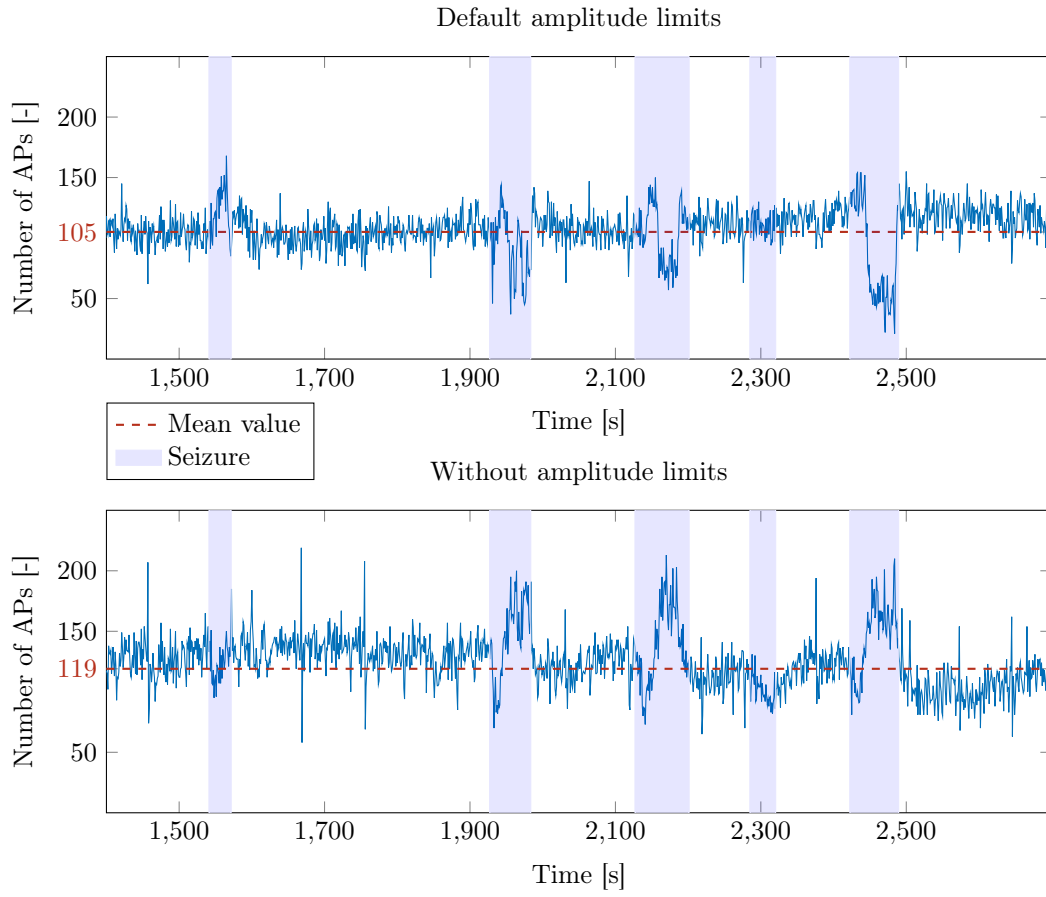


Figure 2.2: Effect of the APs amplitude limits on the number of spike detected. The time axis is restrained to period where seizures are occurring, and the subject used is P0036.

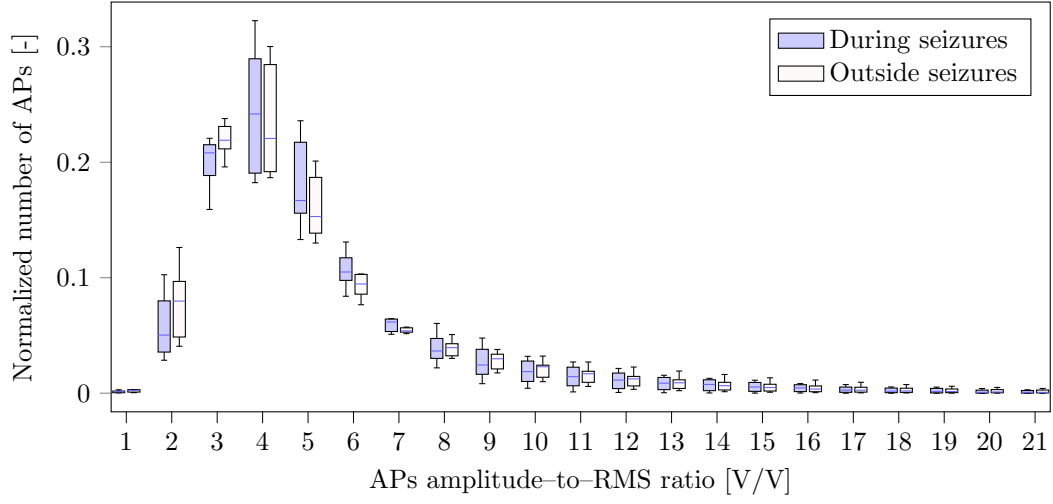


Figure 2.3: Illustration of APs' amplitude-to-RMS ratio distribution during seizure and outside, across all animal subjects. For each animal subjects, the AP detection is performed without amplitudes limits, the detected APs are divided by the local RMS value, and APs present during seizure are separated from the others. They are counted in unity bins from 1 to 21 and normalized to compare the distribution during seizures and outside. Boxplots are drawn to represent the variations across animal subjects inside each bin.

subject's recording is divided in three windows, leading to a total of 18 windows, possibly with all different centroids. None of the obtained centroids match closely the original template, with a maximum correlation coefficient of 0.97. However, one centroid is very common, appearing on half of the total windows. This centroid is represented in figure 2.4 alongside the original template. Table 2.2 compares all centroids with this most common centroid, and 9 windows contains only two centroids, with a mean absolute Pearson correlation coefficient of 1.00 ± 0.00 with the most common centroid.

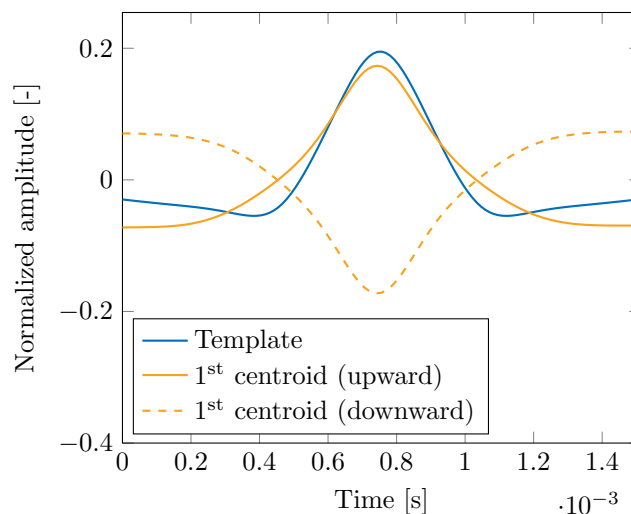


Figure 2.4: Original template and the most common centroid, normalized in amplitude. Their absolute Pearson correlation coefficient is 0.94 (and 1.00 between the upward and downward centroid).

If the goal of using one generic and one specific detection is only to accommodate to each electrode and patient setup, it could be performed offline, as for half of the data, changing the original template with this centroid is enough to skip the first generic detection. To make the system more sensitive to the different APs shapes, the mean discorrelation accepted within a cluster can be lowered, to motivate the creation of new clusters. Another option is to shorten the clustering windows, to increase the time precision. Using 4 min windows, corresponding to 16 per subject and 96 in total, still 52 % of the centroids are considered to be equal. In addition, their distribution among the windows and subjects, tends to be the same. From P0036 to P0062, the percentage of centroids considered equal to the most common one are 87 %, 100 %, 6 %, 62 %, 50 % and 0 %. It corroborates with the proportion of data highlighted in blue in table 2.2.

Subject	Number of centroids (1 st , 2 nd , 3 rd window)	1 st window (mean $\pm$ std)	2 nd window (mean $\pm$ std)	3 rd window (mean $\pm$ std)
P0036	2,2,2	1.00 $\pm$ 0.00	1.00 $\pm$ 0.00	1.00 $\pm$ 0.00
P0048	2,2,2	1.00 $\pm$ 0.00	1.00 $\pm$ 0.00	1.00 $\pm$ 0.00
P0049	4,4,4	0.96 $\pm$ 0.01	0.96 $\pm$ 0.02	0.96 $\pm$ 0.01
P0058	5,2,2	0.95 $\pm$ 0.02	1.00 $\pm$ 0.00	1.00 $\pm$ 0.00
P0059	3,2,4	0.98 $\pm$ 0.02	1.00 $\pm$ 0.00	0.96 $\pm$ 0.00
P0062	5,5,4	0.94 $\pm$ 0.01	0.94 $\pm$ 0.01	0.95 $\pm$ 0.01
<i>Average</i>	<i>3</i>	<i>0.97 $\pm$ 0.01 (on all windows)</i>		

Table 2.2: Pearson correlation coefficient between all centroids and the most common one.

To reduce the memory and computation burden of the clustering two solutions are possible: either remove this part from the embedded application and perform it online or drastically reduce the clustering duration. To keep a more complete analysis of the algorithm, the latter is selected. Besides, centroids found inside one window are reused on the next one, as the results tends to be the same over consecutive windows. This means that the clustering is not performed over each window, but less frequently, at a rate that is yet to be determined.

2.4 Seizure detection

In [13], Stumpp et al. use an anterior iteration of the same template matching algorithm to obtain the frequency and amplitude of APs. From there, a detection metric is developed to be used as a basis for the classifier. They also describe a complete and precise methodology to tune the parameters, using machine learning cross-validation techniques.

2.4.1 Detection metric

The detection metric is computed based on the APs frequency and amplitudes. They are grouped in two, adjacent, sliding windows, called foreground (fg) and background (bg) window. They move through the signal and at each step, the ratios of the APs amplitude and frequency they convey are computed. Intuitively, it will describe how the frequency and amplitude varies, similarly to what a derivative would do. The background window duration is always longer than one of the foreground window. The time resolution is set to 0.5 s.

The mean spike amplitudes within the window, U_{bg} and U_{fg} are defined as the average absolute amplitude of all detected spikes within the window. Similarly, the mean spike frequencies within the window, f_{bg} and f_{fg} are defined as the number of spikes within the window, divided by the window length. The relative frequency and amplitude slopes, f_r and U_r , are defined as

$$f_r = \frac{f_{\text{fg}}}{f_{\text{bg}}} \quad U_r = \frac{U_{\text{fg}}}{U_{\text{bg}}} \quad (2.4)$$

corresponding to the ratios previously mentioned. The normalized relative slopes, $\tilde{f}_r$ and $\tilde{U}_r$ correspond to the relative slopes, divided by their baseline standard deviation,

$$\tilde{f}_r = \frac{f_r - 1}{\sigma_{f,\text{baseline}}} \quad \tilde{U}_r = \frac{U_r - 1}{\sigma_{U,\text{baseline}}} \quad (2.5)$$

At last, they are combined into one detection metric, m ,

$$m = (\tilde{f}_r + 1) (\tilde{U}_r + 1) \quad (2.6)$$

that will react to both changes in the amplitude and frequency. At rest, its value is expected to be one.

2.4.2 Classifier

The classifier introduced in [13] from the detection metric m is straightforward. For each subject, a threshold τ is defined based on the mean and standard deviation of the detection metric baseline,

$$\tau = \mu_{m,\text{baseline}} + k\sigma_{m,\text{baseline}} \quad (2.7)$$

k is the baseline standard deviation coefficient, one of the two classifier parameter. The other, n_τ , is the number of samples that need to cross the threshold in order to report a seizure.

Parameters

In addition to the classifier’s parameters, the foreground and background window size of the detection metric are also considered here. A sweep on these parameters is performed to investigate how they affect the results. The obtained TPR and FPR are averaged over all subjects, for the same set of parameters. Each averaged result is plotted on a FPR–TPR axis in figure 2.5 to show the distribution of the classifier performances, with the point’s color representing one parameter values.

It is clear that this setup will not achieve the perfect classification reached in [13]. None of the tested combination reached a TPR of 1, detecting all seizure for all subjects. The few configurations having a TPR above 0.75, reporting correctly 3 out of the 4 seizure in average, have a TPR above 0.8. It means that if they successfully detect seizures, it is because they report everything as one. They have a low k (figure 2.5a) and n_τ (figure 2.5b) value, meaning that the threshold is easy to cross and once crossed, does not take long to be classified as a seizure. It is impossible to get a high TPR and low FPR, and they are in fact highly correlated, with a correlation coefficient of 0.7.

Results

When looking at the detection metric in figure B.1, one could understand why a perfect classification is difficult. For instance, P0062 presents larges peaks in the detection metric at the beginning. On other subjects (P0058, P0059, P0062) nothing stands out during the seizures. Further research in the first step, the AP detection, should be conducted to have better results before treating the data with the seizure detection algorithm. Especially, comparing the data obtained from the VENG signal with the EEG and ECG recordings could help to validate the results of the APs detection. For this work, results were leave as it is, to move on the ultra-low-power implementation and also avoiding overfitting the model on this limited set of data. Conclusion drawn from the implementation are still valid if the algorithm’s parameters change, and their values can be easily modified in the source code.

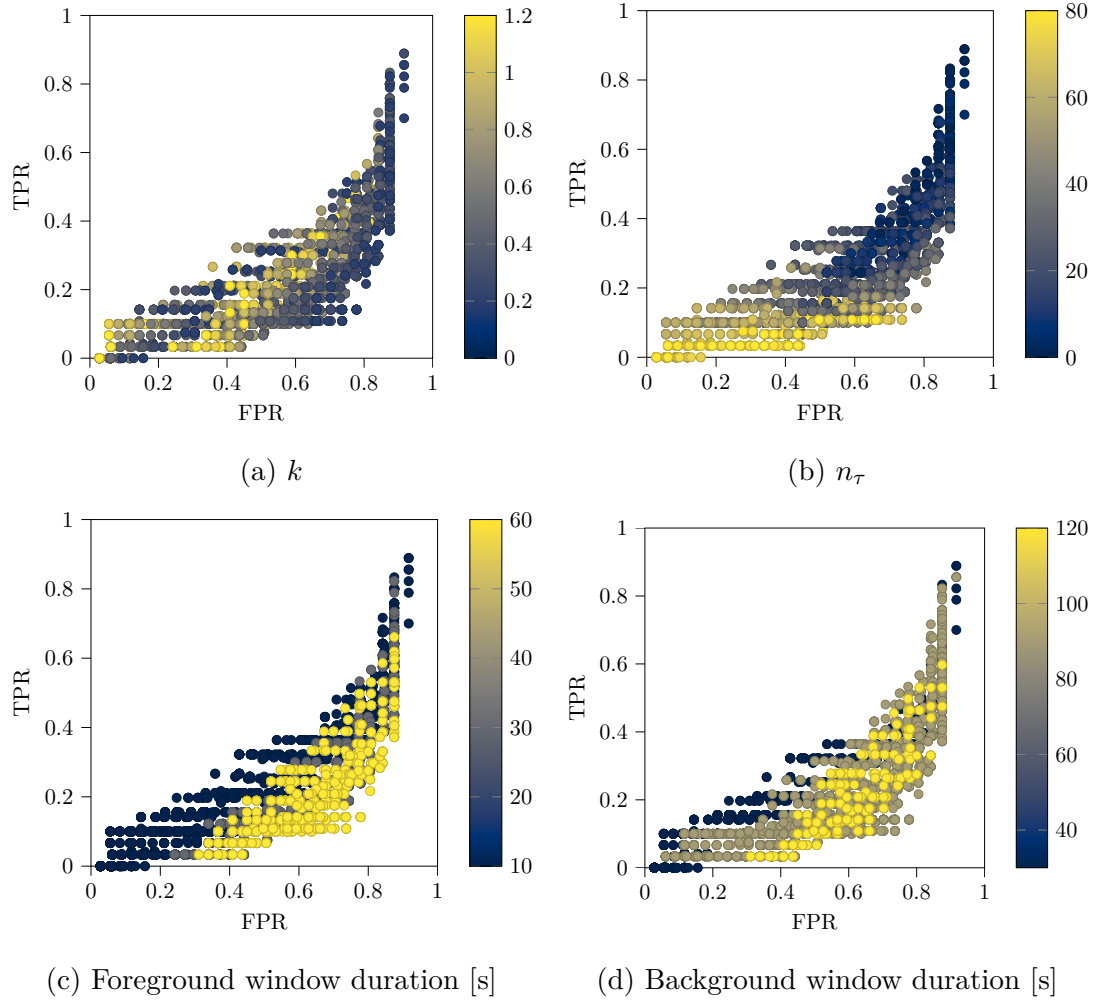


Figure 2.5: FPR–TPR plot of the sweep on parameters. Each graph highlight a specific parameter through the color scheme. The best classifier would be in the top left corner (TPR of 1 and FPR of 0), and the worst in the bottom right (TPR of 0 and FPR of 1).

Chapter 3

Ultra-low-power implementation

The targeted platform is an Apollo 3 Blue system on a chip (SoC) (referred as the Apollo SoC), designed by ambiq micro, and mounted on an evaluation board (EVB) [37]. The EVB includes a variety of tools to ease the implementation and testing process: female jump wire connectors, power source management, USB connector, user switches and LEDs, reset button and a Segger J-Link debugger, as seen in figure 3.1. The Apollo SoC is based on the Arm Cortex-M4 microcontroller and present a rich set of peripheral and function, resumed on the block diagram in figure 3.2.

In perspective of being implemented on the SoC, the algorithm is first modified to perform a real-time detection, since it is currently only running offline. This new version is written in C code and tries to be as efficient as possible. During the development process, the code is compiled on both the macOS and Ubuntu operating system, to take advantages of faster compilation time, powerful debugging and testing tools and easy access to the experimental data. Once done, the program is adapted to run on the microcontroller, with the least amount of change possible and keeping both versions running on the SoC and personal computer synchronized. The personal computer version is referred as the *desktop program* and the SoC one as the *embedded program*. The desktop program can process up to 2 MS/s per template on a standard device¹, treating one data recording in less than 20 s per template.

To assess the code's results, the CUnit testing library is used [38]. CUnit is a unit testing library for C, providing functions to create test suite and asserts variables' values. Unit testing consist of validating the software behavior, either

¹2.5 GHz Intel Core i5, 16 GB DDR3 on a mid 2012 MacBook Pro

as a whole or by isolating functions. Both approaches are taken. For most of the implemented function in C, the equivalent code in MATLAB is identified and some input and output data is saved. The input data is fed to the analogue C function and the C and MATLAB output are compared. Depending on the situation, either a strict equality is required or with some margin of error.

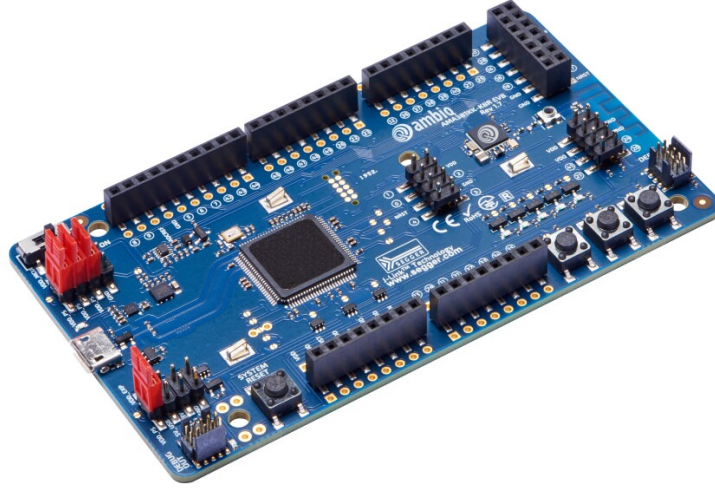


Figure 3.1: Apollo 3 Blue EVB. (source: [37])

3.1 Design choices

To fulfill real-time requirements and continuous acquisition, the algorithm's core needs modifications. Also, even if the current signal is filtered to stay in the 300 Hz – 3000 Hz range, the sampling rate is imposed at 20 kHz to allow future adjustments of the signal bandwidth, up to 10 kHz. In MATLAB, the algorithm processed 30 min long data, corresponding to 36 MS at 20 kHz, something impossible to achieve on the Apollo SoC, due the memory restrictions (figure 3.2).

The proposed modified algorithm works as followed: each 0.5 s starts one new APs detection process. Then APs are stored for the clustering or deleted, depending on the user's preferences. There is no more specific and generic detection, as each part of the signal is analyzed only once. On the first detection rounds, the detection can be considered as generic. Periodically, at a user-defined rate, the clustering process starts and overwrite the generic template with the new ones. The following detection rounds are now considered as specific. The detection metric is computed each time, based on the number of detected APs and their amplitude. This new processing pipeline's flowchart is represented in figure 2.1. Two new

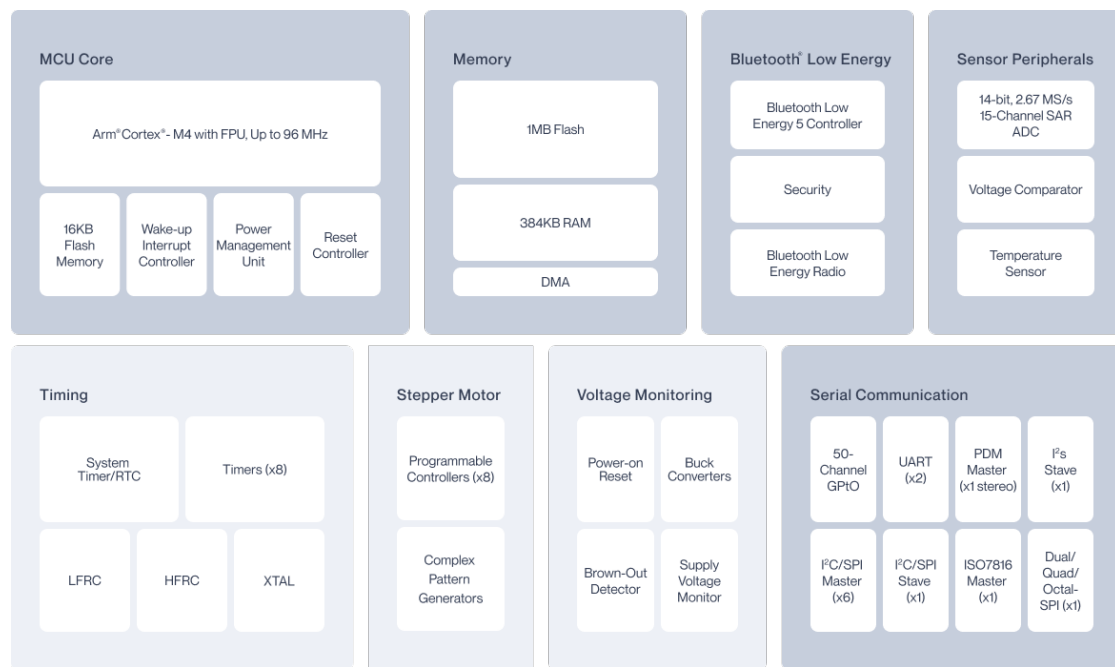


Figure 3.2: Block diagram of the Apollo 3 Blue SoC (source: [37]). The blocks of interest are the MCU core, memory, sensor peripheral (analog to digital converter (ADC) only), timing (one timer and the high frequency RC oscillator) and serial communication (for testing purposes).

parameters are introduced.

1. The clustering duration, in seconds, defining the time span of the considered APs, answering the question *Saving APs ?* in figure 2.1. A value of 30 would store APs over 30 s worth of data.
2. The clustering period, in seconds, defining at which rate the clustering is started, answering the question *Clustering ?* in figure 2.1. A value of 60 would start the K-means clustering every minute.

Combining the two values given as example, the algorithm would first store the APs on the first 30s of data, then start the clustering and update the templates. During the next 30s only the detection metric will be computed, disposing the detected APs. Then, the cycle restart, and the APs are stored again.

The detection metric computation is performed as introduced in section 2.4.1. Since it requires few operations and only to store the APs amplitudes and frequency in for the foreground and background window, it is already suited for a real-time application. Other changes, with respect to the MATLAB implementation are listed below.

- All data are in 32-bits fixed point format, instead of double-precision floating point.
- The filtering is done with two cascaded second order IIR biquad filter, instead of zero-phase filtering.
- The clustering use the root mean square error distance, instead of the correlation.

3.2 Fixed point arithmetic

The entire processing pipeline (figure 3.3) is using *fixed point* arithmetic. Using fixed point arithmetic instead of simulating floating point operations significantly increase the performance of many algorithms when there is no floating point unit hardware accelerator [39]. The Apollo SoC does come with a floating point unit but using integer calculation is also a step forward a future implementation taking advantage the single instruction multiple data extension, a form of parallelism. For instance, the Cortex-M4 allows simultaneous computation on 2×16 bit or 4×8 bit operand, without noticeable increase in the power consumption [40].

In this paradigm, any rational number x is approximated by a pair of

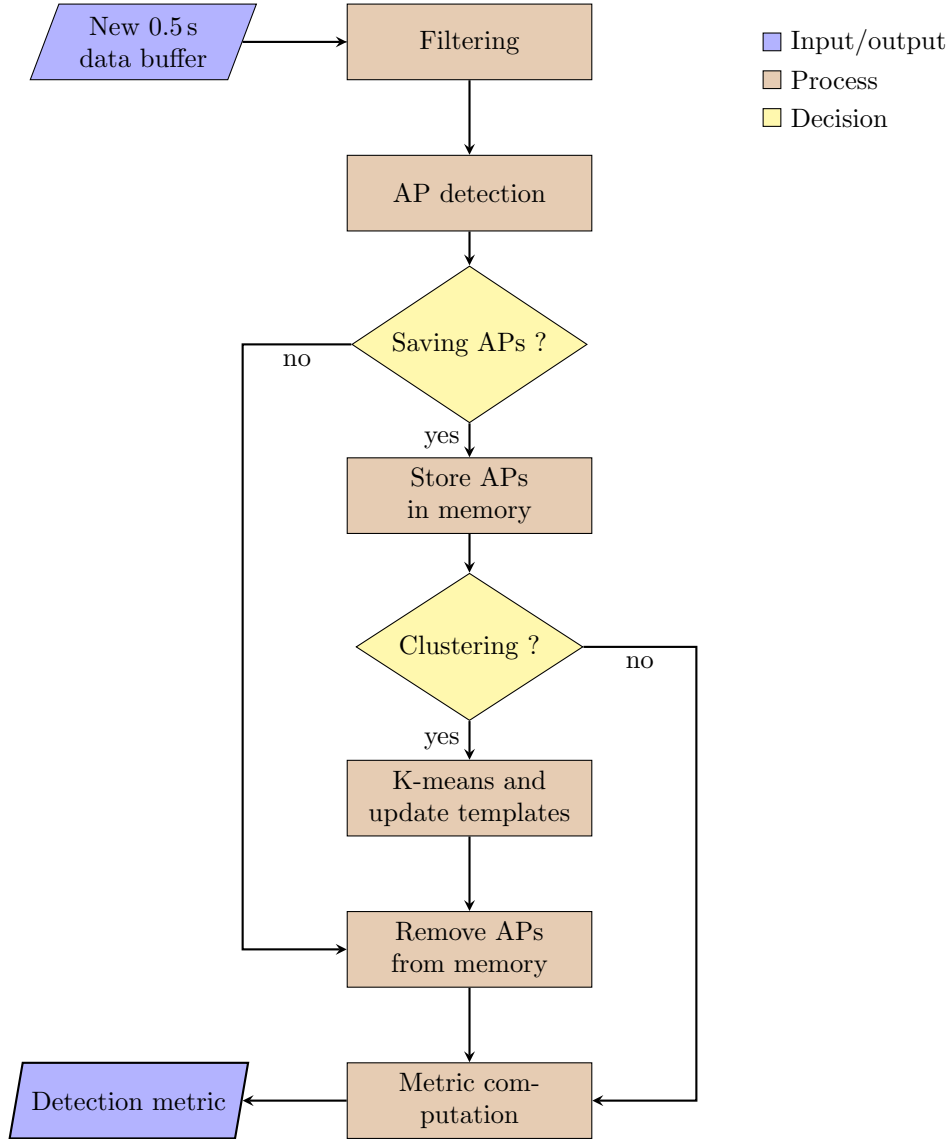


Figure 3.3: Flowchart of the proposed algorithm's pipeline, for one acquisition cycle. Decision are simplified to binary "yes-no" answers, but depend on parameters set by the user.

integers (n, e) , such that

$$x \approx n2^{-e} \quad (3.1)$$

with n being the *mantissa* and e the *exponent*. If both the mantissa and exponent are stored in memory, with e being a variable quantity, the number is said to be a *floating point* number, because the exponent can change, "moving" the decimal point. In fixed point, the exponent is known at compile time and implicit, leaving to the programmer the responsibility to handle it accordingly. This work will refer to the Arm notation, where fixed point number are described as $Qm.n$, where m represent the number of integer bits (including the sign bit) and n the number of fractional bit.

3.2.1 Operations on fixed point number

This short subsection presents how the four basic operations are performed on fixed point arithmetic. Throughout the examples and explanations, a, b, c are defined as

$$a = x2^{-p} \quad b = y2^{-q} \quad c = z2^{-r}. \quad (3.2)$$

Addition and subtraction

Addition and subtraction are straightforward, as long as the exponent are equals ($p = q$)

$$a + b = x2^{-p} + y2^{-q} \quad (3.3)$$

$$= x2^{-p} + y2^{-p} \quad (3.4)$$

$$= (x + y)2^{-p} \quad (3.5)$$

$$= c. \quad (3.6)$$

The result, c has the same exponent as the operand ($r = p = q$). Shifting left and right by n allows respectively a multiplication by 2^n and a division by 2^n , allowing the modification of a and b 's exponent in the first place. Using $Q1.31$ format, setup is required. To perform an addition or subtraction with an integer, the integer should first be converter to fixed-point format, or the other way around.

Multiplication

The product of operands a and b will impact the exponent, as

$$ab = x2^{-p} \times y2^{-q} \quad (3.7)$$

$$= xy \times 2^{-(p+q)}. \quad (3.8)$$

To stay in the same format, a bit shift is necessary to change the exponent. It can either be done before the multiplication on the two operands or afterward, using an intermediate result and then downshifting. The latter allows a greater precision, using full operand bits during the multiplication.

First downshifting the operands resumes to, with $p = q$ for simplicity,

$$ab = xy \times 2^{-2p} \quad (3.9)$$

$$= (x2^{-p/2})(y2^{-p/2}) \times 2^{-p}, \quad (3.10)$$

$$(3.11)$$

where the result is indeed with the same exponent p .

Multiplication with an integer results in the same exponent as the fixed point operand, as the integer has a null exponent,

$$xy \times 2^{-(p+0)} = xy \times 2^{-p}, \quad (3.12)$$

assuming y is the integer.

Division

In contrast to multiplication, the division will decrease the exponent of the result (w.r.t the operands). To end up with c having an exponent r ,

$$\frac{a}{b} = \frac{x2^{-p}}{y2^{-q}} \quad (3.13)$$

$$= \frac{x}{y} 2^{q-p} \quad (3.14)$$

$$= \frac{x}{y} 2^{q-p} 2^r 2^{-r} \quad (3.15)$$

$$= \left(\frac{x}{y} 2^{r+q-p} \right) \times 2^{-r} \quad (3.16)$$

$$(3.17)$$

the numerator must be multiplied by 2^{r+q-p} .

If the denominator is an integer, the result will follow the numerator's exponent, since $q = 0$,

$$\frac{a}{b} = \frac{x}{y} 2^{q-p} = \frac{x}{y} 2^{-p}. \quad (3.18)$$

```

1 /**
2  * @brief 32-bit fractional data type in 1.31 format.
3  */
4 typedef int32_t q31_t;

```

Listing 3.1: CMSIS–DSP $Q1.31$ type format declaration [40]

3.2.2 Programming with fixed point arithmetic

The common microcontroller software interface standard (CMSIS) digital signal processing (DSP) software library, used in this work, provides a broad variety of function for $Q1.31$ variables. The CMSIS library provides a consistent and efficient interface for Arm’s Cortex–M processors and some Cortex–A processors, and the DSP pack specialize in signal processing functions. From the computer’s perspective, fixed point variable are treated as signed integer variables. This is confirmed in the source code of the CMSIS–DSP library, at the `q31_t` declaration (listing 3.1). The `q31_t` is an alias to `int32_t`, a signed 32-bit integer. The `q31_t` type does not ensure that the data is in $Q1.31$ format it is up to the programmer to keep track of the format. It is more a confirmation that the variable is treated as $Q1.31$ by the function, but never guaranteed, as some functions return non $Q1.31$ data in `q31_t` variables. Sometimes, a more flexible approach is taken, and the variable is returned as a floating point, represented by a $Q1.31$ variable (the mantissa) and a shift (the exponent).

3.3 Implementation details

The following section will provide an in-depth description on how the algorithm is implemented on the Apollo SoC: the processing pipeline’ steps are enumerated, the peripheral and parameters of the SoC are detailed and the resulting memory usage is analyzed.

3.3.1 Global functioning

The processing pipeline is divided into four stages, each implementing a key step of the algorithm: the signal filtering, the APs detection, the clustering and the metric computation. The filtering and clustering are straightforward, respectively being a simple call to a CMSIS–DSP function and a reimplement of MATLAB’s k-means algorithm.

The APs detection is further detailed here, as it is the most time-consuming

and complex part of the algorithm. It can be decomposed into the following steps.

1. The CMSIS–DSP `arm_correlate_q31` function is called to obtain the correlation between the signal and the template.
2. The signal is squared to later compute the norms and RMS value.
3. Each point in the correlation is divided by the norm of the signal over the spike duration.
4. The absolute value of the normalized correlation is computed and the maximum between this template normalized correlation and the previous ones is stored.
5. Peaks in the (absolute) normalized correlation, over all templates, are detected. This is a reimplementation of MATLAB’s `findpeaks` functions. Peaks have a minimum distance between them and must be greater than a defined threshold. The peaks are the locations of the detected spikes.
6. The signal RMS value is computed.
7. The spikes’ amplitudes are extracted and if they lie within the accepted range, the spikes are registered as APs.

The metric computation starts by computing the relative frequency and amplitude. A structure stores all the foreground and background window’s element in two circular buffer, one for the amplitude and one for the frequency. While the data is still considered to be in the baseline range, the detection metric is not computed. When the baseline index, a defined parameter, is reached, the baseline standard deviations are computed once. Then, the detection metric is computed for each new 0.5 s buffer.

3.3.2 SoC configuration

This subsection will briefly describe the peripherals used on the Apollo SoC. Everything not mentioned can be considered unused and powered off, if possible. To perform a continuous monitoring of the signal, two buffer are used. While computations are performed on one of them, the other is being filled by the direct memory access (DMA) in the background.

The MCU Core frequency is set to 48 MHz, with all memories enabled. The flash cache controller is enabled for both data and instruction caching. The MCU is set to go into deep sleep while waiting for an interrupt.

CTIMER3 is used to trigger the analog to digital converter (ADC) at 20 kHz and use a division (16) of the high frequency RC oscillator clock as a source, running at $48/16 = 3$ MHz. The counter compare value is set to $(3 \cdot 10^6 / 20 \cdot 10^3) - 1 = 149$ in repeated count mode, to automatically restart the timer after each trigger. Finally, the timer is linked to the ADC trigger input.

The ADC precision is set to 14 bit, without average measurement. The Apollo SoC's ADC is able to perform up to 16 measurements and average them to increase precision, but this feature was not investigated here, as it lays more in the AFE domain. A single input is selected and linked to pin 16 of the EVB. The ADC power mode is set to low power mode 0, suited for continuous acquisition. Low power modes 1 and 2 have respectively a maximal entry latency of 70 μ s and 660 μ s, not compatible with the 50 μ s period of this application. The DMA is used to automatically transfer samples from the ADC FIFO to SRAM and trigger an interrupt when the 10 kS, corresponding to 0.5 s at 20 kHz are all treated. The DMA clears the 18 most significant bits of the variable, containing information used by the ADC peripheral, to only transfer the 14 bits of data.

The ADC interrupt service routine (ISR) reads the interrupts status for error and completion of the transfers, setting boolean global variables accordingly. It then switches the DMA target address, to change the output buffers. The DMA transfer count is reset to 10 kS and the interrupts are cleared. Upon return of the ISR, the main code will resume where it stopped, on the *wait for interrupt* call.

3.3.3 Memory usage

For this application, most variables were statically allocated. *Static allocation*, by opposition to *dynamic allocation*, refers to memory allocated at compile time. Dynamic allocation is made during the program's execution, usually with the `malloc` function in C or automatically as variables are declared. Dynamically allocated variables of the desktop and embedded program are local variables. The desktop program includes as well other allocations, for system calls to read and store the data. Those are not considered. The *memory usage* will thus refer to the global variables, statically allocated.

Static allocation is read from the compiled files, with tools such as `size`² or `nm`³. Large sections of data reported by these utilities can be examined in the source code, where they are declared. A special Make target is implemented to perform a short and quick memory analysis, see appendix A.

²<https://linux.die.net/man/1/size>

³<https://linux.die.net/man/1/nm>

Recall that the input data is stored on 32-bit variables, at 20 kS/s for 0.5 s, resulting in 40 kB buffer (10 kS of 4 B). Reducing the data length (i.e. using 16-bit integer) everywhere will roughly divide the memory usage by 2. Changing the sampling frequency will also have a great impact on most of the operations.

Memory breakdown

The memory usage can be divided into three main categories: instructions, uninitialized data and initialized data. The instructions, also called code segment, text segment or simply text contains the code (instructions) of the program executed. The uninitialized data may be referenced as bss or .bss, for block starting symbol and the initialized data as data.

The uninitialized data is the most memory-consuming part, at 348 kB, while the instruction account for 14 kB and the rest for 0.3 kB. Figure 3.4 shows the proportion of each memory category and describes how the uninitialized data is divided among different functions. The correlation uses the most memory due to the `arm_correlate_q31` functions, requiring an array twice as large as the maximum input. Among the different categories presented in figure 3.4, the correlation, DMA buffers and signal are fixed by the sampling frequency and buffer duration. Spikes, metric and clustering are limited by the users parameters, setting an upper limit on

- the maximum number of spikes stored (Spikes, Clustering),
- the maximum size of both foreground and background window (Metric),
- the maximum duration of the baseline (Metric),
- and the maximum number of clusters (Clustering).

Indeed, the onboard memory is limited to 1 MB of flash memory and 348 kB of RAM, requiring to select the upper limits wisely.

Limitations

Expecting around 100 of APs per second [8], the current system can only perform clustering over 5 s. One way to overcome this would be to reduce the size of the input, either by shortening the DMA buffer and triggering the AP detection more frequently or by using 16-bit data instead of 32-bit. The latter would also result in faster computation times. Reducing the sampling frequency is not considered, as having a 20 kHz sampling frequency is one of the system's requirement.

Reducing the size of the input would directly affect the correlation, DMA

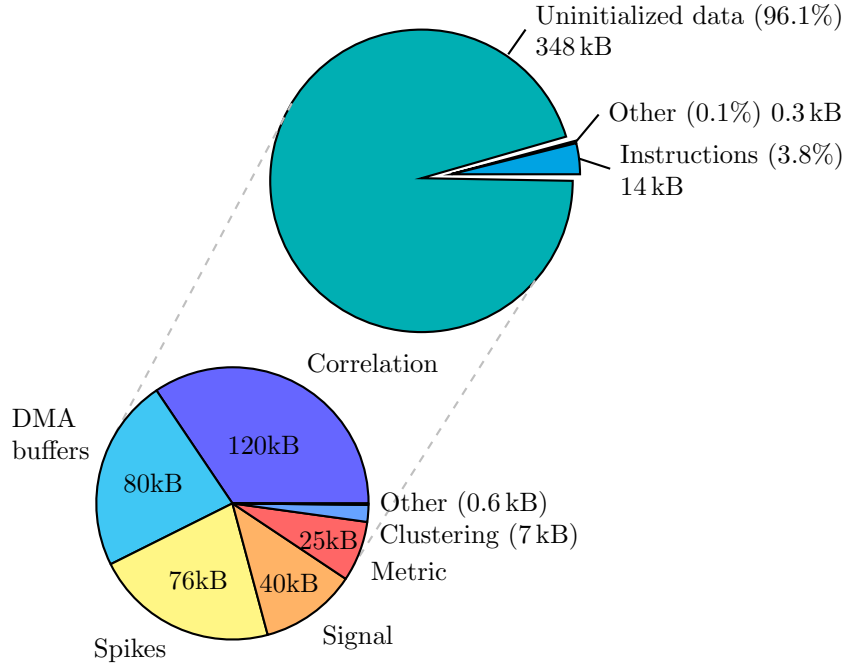


Figure 3.4: Memory usage of the embedded program. Top graph represent the entire memory usage. Bottom graph represent the memory breakdown of the uninitialized data. *Correlation* is the `correlation_t` structure, containing mainly a 80 kB array for the output of the `arm_correlate_q31` function and the squared input signal (40 kB). *DMA buffers* are the two alternating 40 kB input buffers, reused later to store the maximum correlation across all templates. *Spikes* is the `spikes_t` structure, storing the APs location, amplitude and shapes. *Signal* is the filtered signal. *Metric* is the `metric_t` structure, with two circular buffers for the APs frequency and amplitude and two buffers to compute the relative frequency and amplitude standard deviation. *Clustering* is the `kmeans_t` structure, used to store the clusters, the distances between each point and each cluster and the points assignments (to which clusters they are related). The point are the APs in the `spikes_t` structure. *Other* includes the `templates_t` structure (0.44 kB) and structures related to the digital filter. Note that the memory used for the clustering could have been avoided by reusing memory inside the DMA buffers or correlation space, but it was not worth the amount of complexity added, for only 7 kB.

buffers and signal categories of figure 3.4, as said earlier. They sum up to 240 kB, corresponding to 69 % of the uninitialized data. This is therefore the first point to consider in future improvement for the memory usage.

3.4 Optimizations

Optimizations described here are compared to the MATLAB original implementation and only deliberate changes are considered. The use of a simple IIR biquad filter instead of a zero-phase filter is an example of unintentional change made to the algorithm, as it was used because the CMSIS-DSP library did not offer an implementation of a zero-phase filter. The main idea is to avoid computing twice the same results or unnecessary calculations, and rely on the already optimized CMSIS-DSP functions. This section illustrates this philosophy with a few examples.

Reordering operations

Operations are done on the smallest array without affecting the result. For instance, instead of multiplying the template and signal norms together, multiply once the threshold by the template norm, and compare this product to the correlation. Recall the NCC from equation 2.3, compared to a threshold τ_{RMS} it gives

$$\frac{\mathbf{x} \cdot \mathbf{y}}{|\mathbf{x}||\mathbf{y}|} \stackrel{?}{>} \tau_{\text{RMS}}, \quad (3.19)$$

or by writing out τ'_{RMS} as $\tau_{\text{RMS}} |\mathbf{x}|$, considering $\mathbf{x}$ as the template, we get

$$\frac{\mathbf{x} \cdot \mathbf{y}}{|\mathbf{y}|} \stackrel{?}{>} \tau'_{\text{RMS}}, \quad (3.20)$$

Since the templates have a constant norm, it can be done once when initializing the threshold, avoiding one multiplication ($|\mathbf{x}||\mathbf{y}|$) for each element in the input signal. It means that the NCC is never truly computed as defined since it lacks one final division, but the result is unchanged.

The input signal is squared once when computing the normalized correlation and reused later to obtain the signal RMS value. This is a trade-off between memory usage and computation time, as for the normalization there is no need to have the entire input array squared. And when the signal RMS value needs to be computed, memory space used earlier can be overwritten, saving 40 kB of memory but at the cost of twice the computation.

Rolling sums

In the normalization process and the metric computation, rolling sums are used. A rolling sum first makes a classical sum of element over the array for the first iteration, but then only subtract the last element of the previous array and add the first element of the new one. This avoids summing the overlapping element between the two consecutive array, making only two operations.

Optimized library

At last, usage of the CMSIS–DSP library is also to be considered. The library is highly optimized for Cortex processors and can perform loop unrolling. Loop unrolling aims at reducing the loop overhead induced by the branching and testing instructions. A typical case of loop unrolling from the CMSIS–DSP library is presented in listing 3.2. The first `while` loop is the loop unrolling, where instead of performing one addition instruction per loop iterations, it does four. It is also compatible with Arm’s vector instruction set extension that provides up to 5 times performance uplift for DSP functions [41]. However, this instruction set is available for Armv8–M architecture, and not the Armv7–M used on the Apollo SoC.

The CMSIS–DSP code is included in Ambiq’s software development kit as a pre-compiled library. No information regardless the compiler optimization flags nor the use of loop unrolling is included in the software development kit. The exact compilations settings for the library are then unknown but assumed to be the best suited for the Apollo SoC.

```

1  uint32_t blkCnt;    /* Loop counter */
2
3  #if defined (ARM_MATH_LOOPUNROLL)
4
5  /* Loop unrolling: Compute 4 outputs at a time */
6  blkCnt = blockSize >> 2U;
7
8  while (blkCnt > 0U) {
9
10 /* Add and store result in destination buffer. */
11 *pDst++ = __QADD(*pSrcA++, *pSrcB++);
12 *pDst++ = __QADD(*pSrcA++, *pSrcB++);
13 *pDst++ = __QADD(*pSrcA++, *pSrcB++);
14 *pDst++ = __QADD(*pSrcA++, *pSrcB++);
15
16 /* Decrement loop counter */
17 blkCnt--;
18 }
19
20 /* Loop unrolling: Compute remaining outputs */
21 blkCnt = blockSize % 0x4U;
22
23 #else
24
25 /* Initialize blkCnt with number of samples */
26 blkCnt = blockSize;
27
28 #endif /* #if defined (ARM_MATH_LOOPUNROLL) */
29
30 while (blkCnt > 0U) {
31
32 /* Add and store result in destination buffer. */
33 *pDst++ = __QADD(*pSrcA++, *pSrcB++);
34
35 /* Decrement loop counter */
36 blkCnt--;
37 }

```

Listing 3.2: Excerpts of CMSIS-DSP `arm_add_q31` function, highlighting the loop unrolling [40]. This function add element-wise the array `pSrcA` and `pSrcB` of size `blockSize` into the `pDst` array. Modifications have been made to improve readability.

Chapter 4

System characterization

This chapter will uncover a detailed power consumption and execution time report of the algorithm, running on an Apollo 3 Blue EVB from Ambiq Micro. As a real-time test with the VENG recording was not possible, classification results are not discussed here.

After an introduction to the experimental setup and methods used to measure the power consumption and execution time, the results are detailed and analyzed. First the execution time is mentioned, to verify if the system can indeed be real-time, and under which constraints. The timing is studied by separating each core function of the processing pipeline to highlight the more time-consuming ones and to investigate the effect of some parameters on the execution speed. Afterwards, a similar approach is taken on the power consumption and the subsection is ended by a comparison with other works, from the power consumption point of view. At last, several possible improvements are suggested based on the results obtained before.

4.1 Experimental setup and methods

Two measures were of interest: the average supply current and the execution time of different functions. The main setup includes a source and measurement unit (SMU) connected to the 1.8 V power supply pin of the Apollo EVB, a 4 channel digital oscilloscope connected to four general purpose input/output (GPIO) pins of the board and a signal generator feeding the ADC. Precise reference of the instruments used is in table 4.1, with a reminder of their usage in this setup.

The SMU output voltage was set to 1.9 V¹, above the 1.8 V usual voltage, but still in the recommended 1.755 V - 3.60 V range [42]. The measurement was average over time, to provide an accurate estimation of the power consumption through multiple acquisition and computation cycles.

The signal generator was set to output a sine wave, with an amplitude of 120 mV, an offset of 800 mV and a frequency in the 500 Hz - 1300 Hz range. Changing the frequency is an effective and easy way to influence the number of detected spikes.

With the wave measurement functions of the oscilloscope, it is easy to obtain the duty cycle or duration of one function linked to a GPIO, setting the GPIO level to high when entering the function and putting it back to the ground upon return.

When numerical data were to be precisely monitored, the universal asynchronous receiver transmitter interface, provided with the Ambiq Micro hardware abstraction layer. It was for instance used to ensure that the results obtained on one (small) set of data fetched from memory were identical from the one obtained with the C code running on a computer. This peripheral was never enabled when measuring the power consumption, in order to keep unbiased results. When timing and data were needed, each time a proper separation between first the function execution and then the universal asynchronous receiver transmitter activation and transfer was made.

4.2 Resources analysis and bottlenecks

The three metrics of interest concerning the embedded resources' analysis are the execution time, the power consumption and memory usage. Execution time will highlight where to focus further optimizations and ensure that the real-time constraints are respected. Power consumption, linked to the execution time to some extent, reflects the overall quality of the system and is effective to compare against similar works. Memory usage is interesting as power consumption can be reduced by disabling part of the on-chip memory. It also indicates what rooms for improvements there is left, in case additional code or data would be loaded on board. The memory usage was previously discussed in section 3.3.3.

¹This slight overvoltage was used to account for the voltage drop on the jump wires, preventing a brown out reset.

Name	Reference	Utility
Source and measurement unit	Keithley SMU 2450	Supply power for the EVB while measuring the average current
Digital oscilloscope	Tektronix MSO 2024B mixed signal oscilloscope	Meter functions time execution and duty cycles
Signal generator	Gene Agilent 33250A	Generate a sine wave acting as the output of the AFE for the ADC

Table 4.1: Devices used in the characterization process at the Wallonia electronics & communications measurements laboratory.

4.2.1 Execution time

In order to have a continuous monitoring of the patient, the acquisition buffer of 0.5s must be processed in at most 0.5s. Otherwise, upon completion of the DMA ISR, the central processing unit will resume where it stopped somewhere in the computation. The rendezvous is missed and one (or more) acquisition buffer are ignored. This requirement is a hard real-time constraint: the buffer computation's frequency must be less than or equal to the buffer acquisition's frequency. It is the only time constraint of the program.

The call and return of the successive functions are represented in figure 4.1, showing the state of the corresponding GPIOs. This informative plot confirms that the real-time constraint is not violated, all functions starting every 0.5s. The AP detection in this default configuration is made with one template. The rest of this subsection will investigate the effect of some parameters on the execution time, to conclude with an approximated constraint to respect in function of the said parameters values.

Number of templates

The APs detection runtime with one template is 289ms. It is then clear that when using two templates in the detection process, the real-time constraint is not respected, as the AP detection alone takes 567ms. This is illustrated in figure 4.2, comparing both execution time with one and two templates and highlighting the

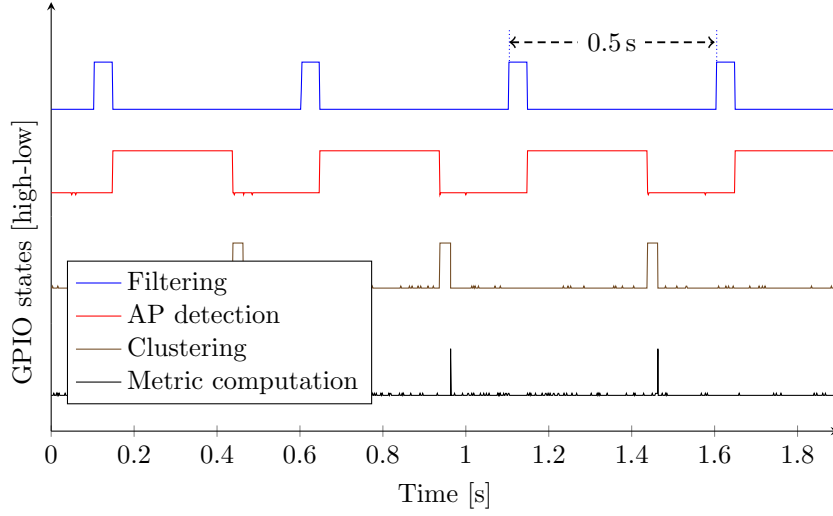


Figure 4.1: Execution time of each stage in the processing pipeline, spanning on three full acquisition cycles and the start of a fourth one

hard real-time constraint.

Number of detected APs

Although the number of detected APs is not a user's parameter, it is greatly influenced by others and therefore makes a good synthesis of them. Said main parameters of influence are the minimum distance between two APs, the normalized cross-correlation threshold and the amplitude-to-RMS ratio limits. Also, the current memory setup constraint the number of stored APs to 500, and the timing requirement to the usage of only one template.

The only affected functions are the APs detection and the clustering. The filtering is considered to be constant in time, as well as the metric computation, except for trivial cases where no APs are detected multiples times in a row. In this case, some divisions are skipped, and the result is undefined, to avoid division by zero. This situation is not considered, as it should not arise in a proper configuration.

Both the APs detection and the clustering execution time are expected to grow linearly with the number of detected APs. For the APs detection, it is due to the last step, where APs are validated based on their amplitude and copied to store them for the clustering. The clustering growth linearly due to the distances between each point and cluster that need to be computed. This was validated experimentally, with the results displayed in figure 4.3, where the linear relationship

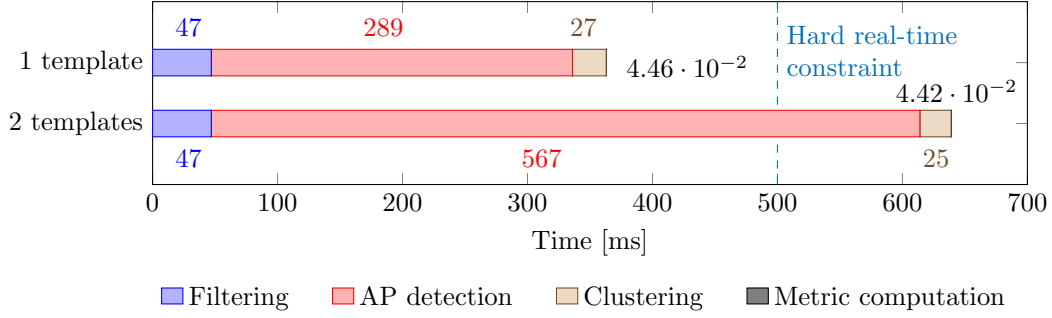


Figure 4.2: Execution time of the pipeline stages with 1 and 2 templates. The hard real-time constraint is the completion of a new ADC-DMA cycle. Note that although the metric part being hardly visible, its value is still displayed (on the far right of each bar).

is clearly visible on the top right and bottom graph.

Normalization

To better understand which part of the AP detection is the longest, it is divided in two main parts. First the normalized cross correlation, second the peak detection and AP validation. The two top graph in figure 4.3 shows execution time for these two parts. The normalized cross correlation is what takes the longest, especially the normalization and the correlation.

Recall that for the normalization process, happening after the correlation and squaring of the input signal, a rolling sum is used to compute the signal square norm over the current window. This induces two operation, one addition and one subtractions, as the squared values are already in memory. Then, the square root is computed, and the correlation divided by the norm. The CMSIS-DSP `arm_sqrt_q31` function uses the Newton-Raphson iterative technique to get an approximate result. To compute the square root x of y , the algorithm solves $x^2 = y$ with

$$x_0 = \frac{y}{2} \tag{4.1}$$

$$x_{n+1} = \frac{x_n + y/x_n}{2}, \tag{4.2}$$

where x_0 is the initial guess. Its main advantage is the quadratic convergence rate and simple implementation [43].

A review of fast square root approximation algorithm for fixed point micro-controller estimated that the Newton–Raphson method converge in two or three iterations and using the same amount of division, depending on the precision required [43]. It means that for each point in the input signal, there is three or four division to perform, including the final division by the norm. This large amount of division is believed to take up most time in the normalization process, highlighted in figure 4.3 (top left graph).

A solution to alleviate this task would be to use a faster method to compute the square root. In [43], a method that converges in one iteration with a maximum error of 0.04 % is introduced, while they report a maximum error of 5×10^{-6} % for the Newton–Raphson method. This technique only use one division. Going from four or three division to only two could give substantial improvements, greatly reducing the execution time. An even faster but less precise version of their algorithm is also proposed, with a maximum error of 2.86 %.

4.2.2 Power consumption

To have a precise insight of each part of the code consumption, function were incrementally enabled to see the added power of each functionality. The total power consumption is of 1.92 mW and half of it is consumed by the ADC and DMA running in the background (0.96 mW). About the active computing functions, the longer they take to execute, the more power they consume, as it can be seen in figure 4.4. This is expected, as they all are executed with the same configuration for the Apollo SoC, meaning that the instantaneous power should be the same, and the average power will depend on their runtime.

The instantaneous power is measured using a shunt resistor (38.46Ω) and two oscilloscope probes to obtain the voltage drop on it. From there, the instantaneous current and finally the instantaneous power is derived. Obtaining goods results is difficult, as we want both a high precision (in the mV range) with a high offset (around 1.9 V). The measured instantaneous power is visible in figure 4.5, with the expected instantaneous power, derived from measurement with the SMU (figure 4.4) and the functions' duty cycles (figure 4.2).

Recall that the ADC and DMA have a consumption of 0.96 mW and the processing pipeline also 0.96 mW, with an active duty cycle of 70 %. Then, the baseline consumption, for the ADC and DMA should be of 0.96 mW and the processing pipeline should add 1.37 mW for a total of 2.33 mW. This is how the orange line was obtained in figure 4.5, showing the expected instantaneous power.

When processing the data, the instantaneous power is relatively close to

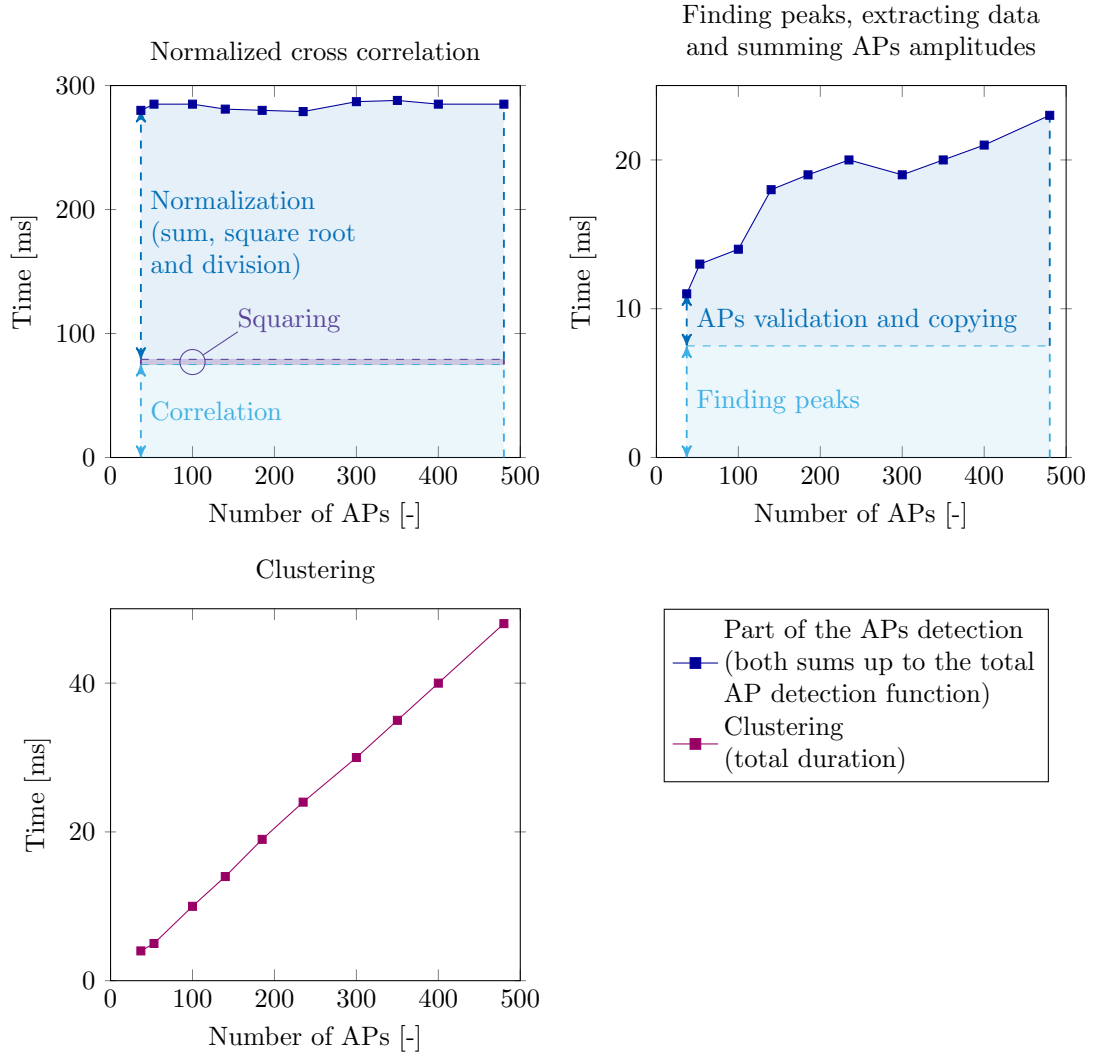


Figure 4.3: Duration of the pipeline steps in function of the number of spikes recorded. Top graphs represent part of the APs detections, the addition of the two lines giving the total runtime of the function. This way, the normalized correlation, running in 283.5 ± 3 ms, can be decomposed into 75 ms for the correlation, 4.16 ms for computing the square of the input signal and the rest is for the normalization. Similarly, on the top right graph, the other part of the APs detection function is divided into a constant runtime of 7.5 ms used to find the peaks in the correlation and the time used to make the final validation and copy data for further clustering. Values are considered constant when their standard deviation across measurements is smaller than 1 ms. Bottom graph represent the runtime of the clustering, set for two centroids and ending in one iteration. All vertical axes ranges are different.

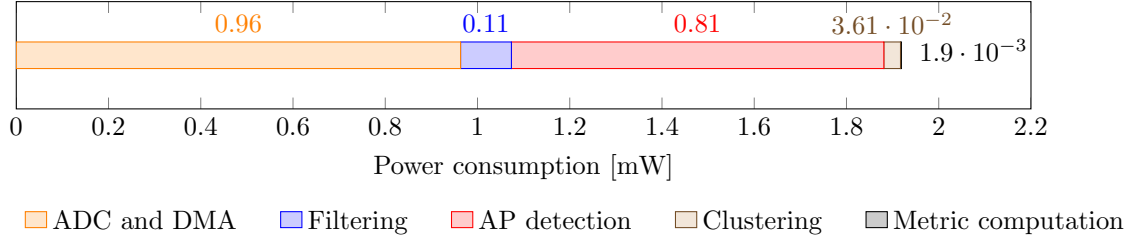


Figure 4.4: Power consumption of each stage in the processing pipeline.

what is expected, at 2.15 ± 0.23 mW. However, when the core is in sleep mode and only the ADC and DMA are running, the power consumption is 0.41 mW lower than expected. The measured conducted with the SMU are considered as more reliable, as they are average over a longer time and more precise, while the measure on the shunt resistor suffers from imprecision mentioned above. Moreover, the results were really sensitive to the oscilloscope configuration, with significant variations across different measurements, adding or subtracting up to 1.5 mW to the inferred power.

The variation in the instantaneous power are however always visible, at the same timestamps. In figure 4.5, the power consumption at the beginning of the processing pipeline is slightly higher. This part correspond to the filtering and the start of the normalized correlation that relies on calls to CMSIS–DSP functions. Later, another spike is visible, again when CMSIS–DSP functions are called. Between these two, it is mainly the normalization process (the norm rolling sum, square root computation and division). At the end, when the power consumption rises again, APs are copied and distances are computed in the clustering. Three propositions are made to answer this behavior.

1. The CMSIS–DSP functions are better optimized, and suffer less from stalls, where the processor have to wait before executing another instruction. During these stalls, more frequent outside the CMSIS–DSP functions, the power consumption is reduced as fewer operations are performed over the same amount of time.
2. The CMSIS–DSP functions use more frequent access to the program or data memory and relies less on the cached array memory. This is plausible since the considered functions works on large array quickly. The filtering, for instance, uses the whole 40 MB input array.
3. The CMSIS–DSP functions performs more complex operations, where the

content of the registers changes more frequently. A higher power consumption could be associated with more switches in the registers bits.

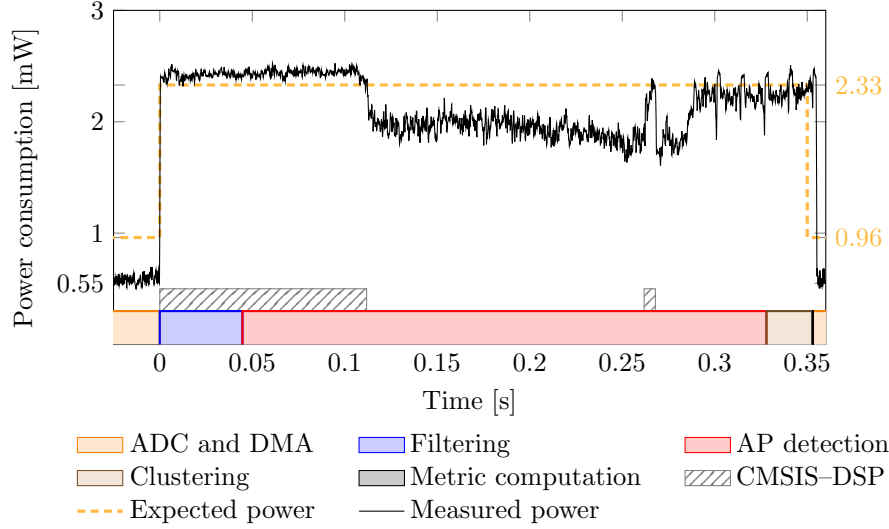


Figure 4.5: Instantaneous power consumption of the Apollo SoC. At the bottom, colors indicates which function is running. Part where the execution relies exclusively on CMSIS–DSP functions are also indicated.

4.2.3 Comparison with other works

Comparison with other works is difficult, due to the lack of similar applications available with enough information to make a comparison. Two anterior works are selected: one previous master thesis from the UCLouvain [44] and a one paper published in the symposium on VLSI circuit [45], also from the UCLouvain. This section will mainly focus on the power consumption.

Introduction to other works

The master thesis implemented an analog front-end for VENG sensing with a digital back-end to perform an adaptive tripole, based on the Apollo 3 Blue. The Apollo SoC uses two channels on the ADC and transfer 2048S with the DMA, so 1024S per channel. Once the DMA transfer is completed, the samples are formatted, to remove information left by the ADC in the samples. It is unclear why this is not performed by the DMA, as this is a feature is available and used in this work. It could be to ensure that the samples are alternating from the first channel and the second, having all even indexes related to one channel and odd

indexes to another. The processing is using floating point arithmetic to compute the imbalance correction.

The VLSI article introduce a new custom Cortex-M4 microcontroller, named *SleepRider*. In a use case on ECG monitoring, it is compared to a commercial ultra-low-power microcontroller, the Apollo 3 Blue. As the accent is put on the description of the SleepRider chip, little information about the algorithm or Apollo SoC configuration are available. The DMA is not involved, as samples are treated sequentially, upon each new ADC conversion. Only the measured power of the ADC is considered.

Processing

As the application greatly differ between this work and the two other introduces, it may not be relevant to compare them based on their processing execution time nor power consumption. For instance, this works consumes almost four times the amount of power compared to previous master thesis [44] but performs a full seizure detection and not only an imbalance compensation. Also, the scale of the data differs, this works using 10 kS buffers versus two 1024 S buffers. We can assume that it lays in the same range of values, but it is difficult to validate this works performance based on this comparison alone.

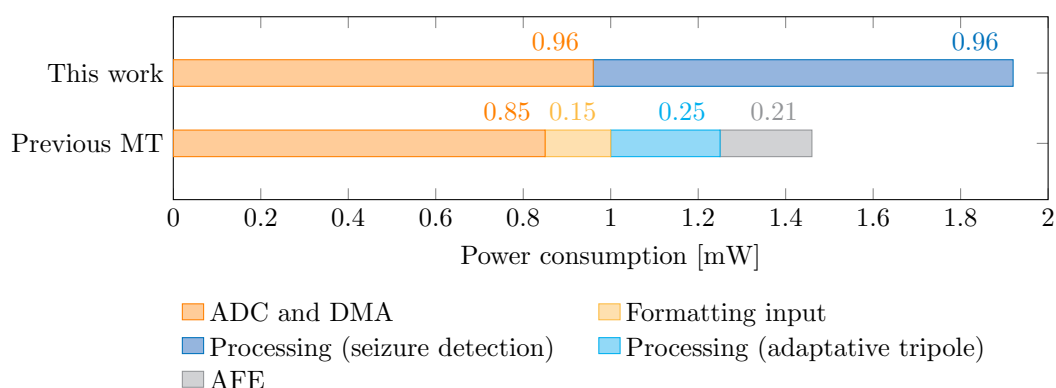


Figure 4.6: Comparison between the power consumption obtain in this work and in previous master thesis [44].

ADC power consumption

The ADC power consumption’s characterization relies on many parameters. To compensate for the various configurations a few hypotheses are made concerning the ADC power consumption. It is considered to be directly proportional to the number of channels, the sampling frequency and the precision (number of bits). At last, a

differential channel is approximated by one channel. Under these assumptions, the power measured by Dekimpe in [45] (1.02 mW) is similar to what is observed in this work (0.96 mW) and about twice as much as what is observed by Jaminon-De Roeck in [44] (0.5 mW), see table 4.2. This is rather unexpected, as the configuration and usage of the ADC in this work and in [44] is similar, except that in [44] two channels are used and the DMA does not mask the unwanted bit on the samples. On the other hand, the situation in [45] is very different, as the DMA is not enabled and there is no continuous acquisition.

The greater power consumption in this work compared to the previous master thesis of Jaminon-De Roeck [44] may come from the usage of the DMA feature to mask bits on the data, and from the memory power consumption, as this work uses the full 348 kB of RAM, while the previous master thesis only enabled 64 kB. The source code of Jaminon-De Roeck work is not available, making it difficult to draw conclusions.

Criteria	This work	Previous MT [44]	Dekimpe, 2021 [45]
Sampling frequency	20 kHz	20 kHz	250 Hz
Number of channel	1	2	1 (assumed differential) [♠]
Precision	14-bits	12-bits	12-bits
Power consumption	0.96 mW	0.85 mW	22 μ W
Power consumption at 20 kHz, for 1 channel and 14-bits of precision [♣]	0.96 mW	0.5 mW	1.02 mW

Table 4.2: Power consumption of the Apollo SoC ADC and DMA in different configurations. [♠] The exact configuration of the ADC is not mentionned. [♣] Under the assumptions mentionned in section 4.2.3.

4.3 Further improvements

Keeping the same configuration, there are two major software aspects that need improvements: the normalized correlation and the data format. Beyond software, hardware acceleration is possible and should be envisaged for the filtering and correlation steps. To fasten the computation time, one solution could be to use the 96 MHz clock frequency for the Arm core. This is considered as a "last resort" solution, as a better alternative exists and are introduced hereunder.

Data format

The current implementation uses 32-bit fixed point numbers in most places. Choosing a high precision format eased the development process, as less consideration on overflows was necessary. However, a more thoughtful analysis of the precision and range of the data needed may allow the usage of smaller data types with appropriate scaling operations. This would result in lower memory usage, in faster computation using single instruction multiple data operations thus reducing the overall power consumption.

Normalized correlation

A faster way to compute the correlation is by going through the frequency domain. However, as stated by Lewis in [46], the *normalized* correlation has no simple expression in the frequency domain, and when the searched feature (the template) is much shorter than the time series (the signal), using the direct form is more efficient. Lewis introduces an efficient method to perform the normalized cross correlation for image processing, using precomputed values of the search window to fasten the norms' computation. Kaso developed a normalized cross correlation algorithm working in the frequency domain, including an expression for the normalization, but is no more efficient than the one introduced by Lewis [47].

It could also be worth looking at performing both the filtering and correlation in the frequency domain, without going back to the time domain in between, as the operations are consecutive. It could mitigate the performances of doing the correlation in the frequency domain. Also, if the same templates are reused on multiple iterations, saving them in the frequency domain could further reduce the cost associated with the transformation.

Finally, recall the improvement suggested in section 4.2.1. Using a faster algorithm to compute the square root would already have an impact on the processing time, and would not require substantial changes in the code.

Hardware acceleration

Both firsts steps (the filtering and the correlation) of the detection process could be performed using specialized hardware acceleration. It could speed up the process and lower the power consumption, as the Arm core could stay in sleep mode during this time. It could also lower the memory usage if resources allocated to these tasks are no longer necessary.

Conclusion

Epilepsy is a neurological disorder disturbing the brain's behavior. As it is one of the most common brain disease, lots of various medications and treatment are available. One class of them are stimulator, trying to avoid and abort seizures. They have proven their efficiency, but comes with a cost: surgery adverse effects, implant side effects and on top of that, ethicists are raising awareness on the usage of brain implants and monitoring techniques. In this context, a new device, based on the vagus nerve for both the stimulation and detection is studied to alleviate the downside of the one already available.

This master thesis first investigated the implementation of a seizure detection algorithm based on the APs present in the vagus nerve. Although the two algorithms chained here were each validated in previous work, at a first glance, they seem reluctant to work together. Further investigation should be conducted to adapt them and find the root of the problem.

Nevertheless, the ultra-low-power implementation, written in C code and using fixed point arithmetic is made. It offers a fast, efficient and well-organized codebase, that can be reused later if modifications are applied to the MATLAB algorithm. It is implemented to run both on a personal computer and an Apollo 3 Blue SoC. Tools to analyze the code efficiency and memory usage are also provided, with a complete set of test functions to validate the program behavior. The Apollo SoC is configured with the memory-mapped registers, to have a clear and precise understanding of its behavior.

The memory usage of the embedded program reveals that the current implementation limits the number of APs that can possibly be stored on the device for the clustering. The large input size for the algorithm (40 kB) induces larges buffers in the next steps, taking most of the available memory. The execution speed highlighted that only one template can be processed in time, due to the long normalization process. Using more than one fails to meet the real-time requirements. The total power consumption of the Apollo SoC remains high, considering that the

AFE and stimulation part are not considered, at 1.92 mW. Half of this power is drawn by the ADC and DMA, which is almost twice as expected when compared to other works. To investigate it, more configuration of the ADC should be tested.

This work demonstrated the current software, hardware and classification weaknesses of the proposed algorithm to detect epileptic seizures and its current ultra-low-power implementation. It proved that for a fraction of the price, it is possible to achieve results similar to the one obtain with MATLAB. It opened the door to future improvement in order to pursue a vagus nerve-based closed loop system to treat refractory epilepsy.

Appendix A

List of implemented Make targets

To compile the desktop program, a *Makefile* was created with a variety of targets. A Makefile is a set of rules to compile files and handle dependencies. Some rules do not compile anything, but are used to execute certain commands. They are declared in the Makefile so that they can have prerequisites, triggering a new compilation of the program if the source code has changed. All make targets are commented and can be automatically listed with the `make help` commands. Most of the command were silenced, to only display relevant information in the terminal. For instance, instead of printing the entire compiler call command, only "[<status>] Building <filename>..." is displayed, with <status> changing from * to v or x in case of failure. It is visible in figure

Makefile targets are separated in five categories: general, build, runs, tools and cleaning.

1. General

`help` Displays a summary of the Makefile targets, similar to what is introduced here.

2. Build

`main` Build the main target, the executable file.

`embedded` Build the embedded target, a "simpler" version of the main target, very similar to the one running on the Apollo SoC.

`.../tests` Build the CUnit test executable

.../libCMSISDSP.a Build the CMSIS–DSP library by calling CMake in a subdirectory.

3. Runs

run Run the default full program.

plot Plot the last run with **gnuplot**.

4. Tools

cunit Build the test target and run it, displaying the output of the CUnit tests. Stores the program logs in a separates files, offering to print them afterward.

memcheck Run Valgrind’s tool memcheck on main

callgrind Run Valgrind’s tool callgrind on main and offer to launch KCachegrind to visualize the output.

size Print information about the program size (either the main, embedded or the one build by uVision).

```

~/Doc/U/M2/TFE/s/embedded main !25 ?25 > make main 18:11:09
[✓] Building metric.c.o
[✓] Building spikes.c.o
[✓] Building template.c.o
[✓] Building main
~/Doc/U/M2/TFE/s/embedded main !25 ?25 > make size file=uvision 18:11:40
[✓] Building embedded
Calling nm to uvision/keil/bin/vengApollo.axf ...
Type meaning
  B      Uninitialized data section
  R      Read only data section
  T      Text (code) section

Type      Name      Size Location
B      myCorrelation 118K
B      mySpikes      75K
B      signal        40K
B      g_DMABuffer2  40K
B      g_DMABuffer1  40K
B      myMetric      25K
B      myKmeans      6,9K
T      signalADC     4,0K
T      arm_correlate_q31 1,1K
T      arm_biquad_cas_df1_32x64_q31 896
[...] [...] [...]

Reading vengApollo.map ...

      Code (inc. data)  RO Data  RW Data  ZI Data  Debug
10600      728      4200      240      352672      86058  Grand Totals
10600      728      4200      240      352672      86058  ELF Image Totals
10600      728      4200      240      0      0      ROM Totals

Total RO Size (Code + RO Data)      14800 ( 14.45kB)
Total RW Size (RW Data + ZI Data)    352912 ( 344.64kB)
Total ROM Size (Code + RO Data + RW Data) 15040 ( 14.69kB)

~/Doc/U/M2/TFE/s/embedded main !25 ?25 > 18:11:50

```

Figure A.1: Example of the make command usage. First the main target is build, with the successive object files being compiled. Then, the size command is used and displays information about the files compiled by the uVision IDE.

Appendix B

Detection metric of the six subjects

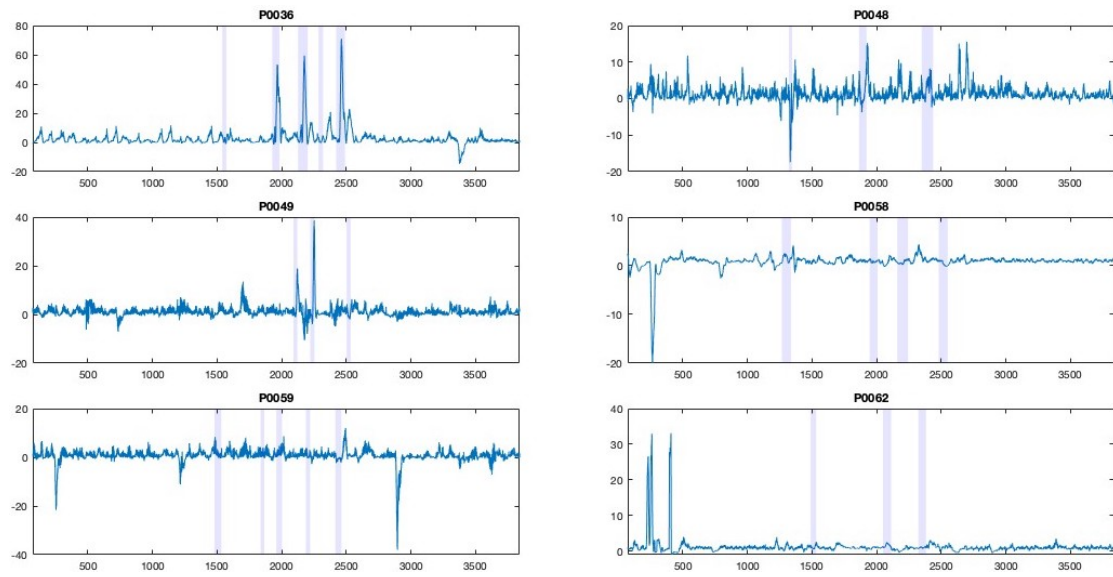


Figure B.1: Detection metric the six subjects. Computed with a 15 s foreground window and a 60 s background window.

Bibliography

- [1] World Health Organization, *Epilepsy*, 2024. [Online]. Available: <https://www.who.int/health-topics/epilepsy>.
- [2] H. De Boer, “Epilepsy; Social Consequences,” in *Encyclopedia of the Neurological Sciences*, Elsevier, 2014, pp. 157–163, ISBN: 978-0-12-385158-1. DOI: [10.1016/B978-0-12-385158-1.00292-X](https://doi.org/10.1016/B978-0-12-385158-1.00292-X).
- [3] *Treatment*. [Online]. Available: <https://www.epilepsy.com/treatment>.
- [4] Patricia O. Shafer and Patricia M. Dean, *Seizure Devices*, Nov. 2018. [Online]. Available: <https://www.epilepsy.com/treatment/devices>.
- [5] P. Ryvlin, S. Rheims, L. J. Hirsch, A. Sokolov, and L. Jehi, “Neuromodulation in epilepsy: State-of-the-art approved therapies,” *The Lancet Neurology*, vol. 20, no. 12, pp. 1038–1047, Dec. 2021, ISSN: 14744422. DOI: [10.1016/S1474-4422\(21\)00300-8](https://doi.org/10.1016/S1474-4422(21)00300-8).
- [6] N. Zangiabadi, L. D. Ladino, F. Sina, J. P. Orozco-Hernández, A. Carter, and J. F. Téllez-Zenteno, “Deep Brain Stimulation and Drug-Resistant Epilepsy: A Review of the Literature,” *Front. Neurol.*, vol. 10, p. 601, Jun. 2019, ISSN: 1664-2295. DOI: [10.3389/fneur.2019.00601](https://doi.org/10.3389/fneur.2019.00601).
- [7] C. A. Fontanillo Lopez, G. Li, and D. Zhang, “Beyond Technologies of Electroencephalography-Based Brain-Computer Interfaces: A Systematic Review From Commercial and Ethical Aspects,” *Front. Neurosci.*, vol. 14, p. 611130, Dec. 2020, ISSN: 1662-453X. DOI: [10.3389/fnins.2020.611130](https://doi.org/10.3389/fnins.2020.611130).
- [8] L. Stumpp, H. Smets, S. Vespa, *et al.*, “Recording of spontaneous vagus nerve activity during Pentylentetrazol-induced seizures in rats,” *Journal of Neuroscience Methods*, vol. 343, p. 108832, Sep. 2020, ISSN: 01650270. DOI: [10.1016/j.jneumeth.2020.108832](https://doi.org/10.1016/j.jneumeth.2020.108832).
- [9] R. S. McLachlan, “Suppression of Interictal Spikes and Seizures by Stimulation of the Vagus Nerve,” *Epilepsia*, vol. 34, no. 5, pp. 918–923, Sep. 1993, ISSN: 0013-9580, 1528-1167. DOI: [10.1111/j.1528-1157.1993.tb02112.x](https://doi.org/10.1111/j.1528-1157.1993.tb02112.x).
- [10] A. Amar, A. Kouki, and H. Cao, “Power Approaches for Implantable Medical Devices,” *Sensors*, vol. 15, no. 11, pp. 28889–28914, Nov. 2015, ISSN: 1424-8220. DOI: [10.3390/s151128889](https://doi.org/10.3390/s151128889).

- [11] R. Raffoul, J. C. Cerda, E. A. Reina, *et al.*, “Action Potential Detection Algorithm Adaptable to Individual Nerve and Recording Setup,” in *2022 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, Taipei, Taiwan: IEEE, Oct. 2022, pp. 655–659, ISBN: 978-1-66546-917-3. DOI: [10.1109/BioCAS54905.2022.9948681](https://doi.org/10.1109/BioCAS54905.2022.9948681).
- [12] T. M. Inc, *MATLAB version: 9.13.0 (R2023a)*, The MathWorks Inc. Natick, Massachusetts, United States, 2023. [Online]. Available: <https://www.mathworks.com>.
- [13] L. Stumpp, H. Smets, S. Vespa, *et al.*, “Vagus Nerve Electroneurogram Based Detection of Acute Pentylentetrazol Induced Seizures in Rats,” *International Journal of Neural Systems*, 2021.
- [14] H. E. Scharfman, “The neurobiology of epilepsy,” *Curr Neurol Neurosci Rep*, vol. 7, no. 4, pp. 348–354, Jul. 2007, ISSN: 1528-4042, 1534-6293. DOI: [10.1007/s11910-007-0053-z](https://doi.org/10.1007/s11910-007-0053-z).
- [15] *35.1 Neurons and Glial Cells - Biology | OpenStax*. [Online]. Available: <https://openstax.org/books/biology/pages/35-1-neurons-and-glial-cells>.
- [16] *The Action Potential - TeachMePhysiology*. [Online]. Available: <https://teachmephysiology.com/nervous-system/synapses/action-potential/>.
- [17] Elaine Kiriakopoulos, *Types of Seizures*, 2022. [Online]. Available: <https://www.epilepsy.com/what-is-epilepsy/seizure-types>.
- [18] J. Sander, “Epileptic Seizures,” in *Encyclopedia of the Neurological Sciences*, Elsevier, 2014, pp. 175–177, ISBN: 978-0-12-385158-1. DOI: [10.1016/B978-0-12-385157-4.00295-5](https://doi.org/10.1016/B978-0-12-385157-4.00295-5).
- [19] O. Markand, “Electroencephalogram (EEG),” in *Encyclopedia of the Neurological Sciences*, Elsevier, 2014, pp. 1084–1091, ISBN: 978-0-12-385158-1. DOI: [10.1016/B978-0-12-385157-4.00523-6](https://doi.org/10.1016/B978-0-12-385157-4.00523-6).
- [20] P. Camfield and C. Camfield, “Epilepsy, Diagnosis of,” in *Encyclopedia of the Neurological Sciences*, Elsevier, 2014, pp. 100–102, ISBN: 978-0-12-385158-1. DOI: [10.1016/B978-0-12-385157-4.00282-7](https://doi.org/10.1016/B978-0-12-385157-4.00282-7).
- [21] B. C. Jobst, F. Bartolomei, B. Diehl, *et al.*, “Intracranial EEG in the 21st Century,” *Epilepsy Curr*, vol. 20, no. 4, pp. 180–188, Jul. 2020, ISSN: 1535-7597, 1535-7511. DOI: [10.1177/1535759720934852](https://doi.org/10.1177/1535759720934852).
- [22] N. Tian, “Active Epilepsy and Seizure Control in Adults — United States, 2013 and 2015,” *MMWR Morb Mortal Wkly Rep*, vol. 67, 2018, ISSN: 0149-2195/1545-861X. DOI: [10.15585/mmwr.mm6715a1](https://doi.org/10.15585/mmwr.mm6715a1).
- [23] G. Liu and H. V. M. Associates, “Epilepsy: Treatment Options,” vol. 96, no. 2, 2017.

- [24] B. C. Jobst and G. D. Cascino, "Resective Epilepsy Surgery for Drug-Resistant Focal Epilepsy: A Review," *JAMA*, vol. 313, no. 3, p. 285, Jan. 2015, ISSN: 0098-7484. DOI: [10.1001/jama.2014.17426](https://doi.org/10.1001/jama.2014.17426).
- [25] H. C. Skrehot, D. J. Englot, and Z. Haneef, "Neuro-stimulation in focal epilepsy: A systematic review and meta-analysis," *Epilepsy & Behavior*, vol. 142, p. 109182, May 2023, ISSN: 15255050. DOI: [10.1016/j.yebeh.2023.109182](https://doi.org/10.1016/j.yebeh.2023.109182).
- [26] LivaNova, *Vagus Nerve Stimulation (VNS) Therapy for Epilepsy / VNS Therapy*. [Online]. Available: <https://www.livanova.com/epilepsy-vnsth-erapy/en-us>.
- [27] D. Révész, B. Rydenhag, and E. Ben-Menachem, "Complications and safety of vagus nerve stimulation: 25 years of experience at a single center," *PED*, vol. 18, no. 1, pp. 97–104, Jul. 2016, ISSN: 1933-0707, 1933-0715. DOI: [10.3171/2016.1.PEDS15534](https://doi.org/10.3171/2016.1.PEDS15534).
- [28] E. Ben-Menachem, "Vagus Nerve Stimulation, Side Effects, and Long-Term Safety," *Journal of Clinical Neurophysiology*, vol. 18, no. 5, pp. 415–418, Sep. 2001, ISSN: 0736-0258. DOI: [10.1097/00004691-200109000-00005](https://doi.org/10.1097/00004691-200109000-00005).
- [29] P. Afra, B. Adamolekun, S. Aydemir, and G. D. R. Watson, "Evolution of the Vagus Nerve Stimulation (VNS) Therapy System Technology for Drug-Resistant Epilepsy," *Front. Med. Technol.*, vol. 3, p. 696543, Aug. 2021, ISSN: 2673-3129. DOI: [10.3389/fmedt.2021.696543](https://doi.org/10.3389/fmedt.2021.696543).
- [30] LivaNova, *Vns-diagram*. [Online]. Available: <https://www.pacificneuroscienceinstitute.org/brain-tumor/wp-content/uploads/sites/2/2022/09/vns-diagram-labels-LivaNova-med.jpg>.
- [31] V. Salanova, T. Witt, R. Worth, *et al.*, "Long-term efficacy and safety of thalamic stimulation for drug-resistant partial epilepsy," *Neurology*, vol. 84, no. 10, pp. 1017–1025, Mar. 2015, ISSN: 0028-3878, 1526-632X. DOI: [10.1212/WNL.0000000000001334](https://doi.org/10.1212/WNL.0000000000001334).
- [32] *Medtronic DBS therapy*. [Online]. Available: <https://parkinsonseurope.org/i-have-parkinsons/treatment/medtronic-dbs-therapy/>.
- [33] Neuropace, *The RNS System - A patient guide*. [Online]. Available: <https://www.neuropace.com/wp-content/uploads/2021/02/neuropace-rns-system-patient-guide.pdf>.
- [34] D. R. Nair, K. D. Laxer, P. B. Weber, *et al.*, "Nine-year prospective efficacy and safety of brain-responsive neurostimulation for focal epilepsy," *Neurology*, vol. 95, no. 9, Sep. 2020, ISSN: 0028-3878, 1526-632X. DOI: [10.1212/WNL.00000000000010154](https://doi.org/10.1212/WNL.00000000000010154).
- [35] M. Schermer, "Ethical issues in deep brain stimulation," *Front. Integr. Neurosci.*, vol. 5, 2011, ISSN: 16625145. DOI: [10.3389/fnint.2011.00017](https://doi.org/10.3389/fnint.2011.00017).

- [36] Lars Stumpp, “Vagus nerve electroneurogram for seizure detection in the acute Pentylenetetrazol model in rats,” Ph.D. dissertation, UCLouvain, 2022. [Online]. Available: <http://hdl.handle.net/2078.1/264708>.
- [37] A. E. Team, *Apollo3 Blue*, Aug. 2020. [Online]. Available: <https://ambiq.com/apollo3-blue/>.
- [38] *CUnit Home*. [Online]. Available: <https://cunit.sourceforge.net/>.
- [39] C. Hinton and C. Cb, “Document Number: ARM DAI 0033A Issued: September 1996 Copyright Advanced RISC Machines Ltd (ARM) 1996 All rights reserved,” *Open Access*,
- [40] *CMSIS-DSP*. [Online]. Available: https://arm-software.github.io/CMSIS_6/latest/DSP/index.html.
- [41] A. Ltd, *ARM*. [Online]. Available: <https://www.arm.com/>.
- [42] Ambiq Micro, “Apollo3 Blue EVB Quick Start Guide,”
- [43] A. Dianov and A. Anuchin, “Review of fast square root calculation methods for fixed point microcontroller-based control systems of power electronics,” *IJPEDS*, vol. 11, no. 3, p. 1153, Sep. 2020, ISSN: 2722-256X, 2088-8694. DOI: [10.11591/ijpeds.v11.i3.pp1153-1164](https://doi.org/10.11591/ijpeds.v11.i3.pp1153-1164).
- [44] C.-T. Jaminon-De Roeck and D. Bol, “Implementation and optimization of an ultra-low power vagus nerve sensing system for epileptic seizures detection,” TFE, UCLouvain, 2021.
- [45] R. Dekimpe, M. Schramme, M. Lefebvre, *et al.*, “SleepRider: A 5.5 μ W/MHz Cortex-M4 MCU in 28nm FD-SOI with ULP SRAM, Biomedical AFE and Fully-Integrated Power, Clock and Back-Bias Management,” in *2021 Symposium on VLSI Circuits*, Kyoto, Japan: IEEE, Jun. 2021, pp. 1–2, ISBN: 978-4-86348-780-2. DOI: [10.23919/VLSICircuits52068.2021.9492365](https://doi.org/10.23919/VLSICircuits52068.2021.9492365).
- [46] J. P. Lewis, “Fast Normalized Cross-Correlation,”
- [47] A. Kaso, “Computation of the normalized cross-correlation by fast Fourier transform,” *PLoS ONE*, vol. 13, no. 9, Y. Li, Ed., e0203434, Sep. 2018, ISSN: 1932-6203. DOI: [10.1371/journal.pone.0203434](https://doi.org/10.1371/journal.pone.0203434).

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | www.uclouvain.be/epl