POLITECNICO DI TORINO

DIPARTIMENTO DI ELETTRONICA E TELECOMUNICAZIONI (DET)

Master Degree in Electronic Engineering

Master Degree Thesis

A Low-Power Embedded System for Real-Time EMG based Event-Driven Gesture Recognition

Supervisors
Prof. Maurizio Martina
Dr. Paolo Motto Ros
Prof. Guido Masera

Candidate
Andrea Mongardi

Torino, July 2019

Abstract

Gesture recognition is an important topic in modern IoT applications, being used to control mobile apps, robotics and also videogames. Many technologies are in use to detect gestures and make them suitable to digital processing and machine learning classifier. One widely used way to collect data, especially in biomedical researches, is based on surface ElectroMyoGraphic (sEMG) signals, obtained simply applying non-invasive electrodes on the skin of the area of interest. This approach makes gesture recognition suitable for Human-Machine Interface (HMI), like prosthesis and robotic limb control.

This thesis work studies a possible implementation of hand gesture recognition, using a system based on the Average Threshold Crossing (ATC) event-driven feature of the forearm sEMG signals. This feature is obtained averaging on a predefined time window the events generated when a sEMG signal goes above a voltage threshold; the obtained value is an index of muscle activation.
The proposed system is composed of three acquisition boards (which acquire sEMG signals and process them to obtain the ATC values) and an Apollo2 MicroController Unit (MCU) with an ARM Cortex M4F microProcessor (µP). For demonstration purpose two Bluetooth 3.0 modules have been added to communicate with an Arduino-based tank able to execute performed movements. The thesis work focuses on firmware optimization on the ARM µP as well as software on MATLAB® environment, in order to obtain the lowest power consumption possible, with a latency suitable for real-time applications (< 300 ms).

Dataset creation has involved 25 healthy people, each one performing five movements within five repeated sessions. The neural network has been trained using the holdout validation method, implemented at low computational level, to exploit ARM library capabilities. Power consumption analysis have been performed on both acquisition channel and MCU, obtained values are 0.7 mW and 0.8 mW, respectively; overall power consumption results in 2.9 mW. Maximum latency of the classifier has been measured 8.5 ms, that added to the acquisition windows, bring to a latency of 268.5 ms from the gesture to the effective movement of the actuator. The above results make the system suitable for wearable real-time applications.

Summary

The goal of this thesis is to implement a low power system, based on event-driven approach, which would be able to classify six hand gesture. The system is composed by three Analog Front Ends (AFEs), which acquire and pre-process the surface ElectroMyoGraphic (sEMG) signal obtained by the forearm skin, one ultra-low power microcontroller unit, which implements the classifier and manage all the useful peripherals, two Bluetooth modules to communicate and an actuator, which is used to show the correctness and fastness of the system.

Six chapters compose this document.

In the first chapter, the *Introduction*, a general information about possibly useful idea to understand this thesis are given, taken by some literature works, mainly articles. The chapter is divided in four sections: *The Skeletal Muscular System* (sec. 1.1) introduce some basic notions about how the skeletal muscle is made and how it behaves when it is stimulated from the nervous system; then, *ElectroMyoGraphic (EMG) signal* (sec. 1.2) explains the different types of EMG acquisition, focusing on the surface one and listing some possibility of feature extraction; *The Average Threshold Crossing Technique* (sec. 1.3) introduces the feature extraction process used in this thesis, explaining how it is obtained; last, *State of the Art* (sec. 1.4) hake a summary about some previous work on gesture recognition, from some older initial work to other more recent and performing.

After the introduction, the narration tries to follow a logic path, starting describing the hardware, then the data acquisition, last the software part.
Chapter two, called *System Configuration*, describes in detail all the hardware by which the system is composed. This chapter is divided in four sections too: *Acquisition Channels* (sec. 2.1) illustrate the whole structure of the AFEs, explaining in detail what each part does and why, and it also describes which are the electrodes used and related advantages and disadvantages; the *Apollo2 board* (sec. 2.2) is then introduced, explaining the motivation for its employment and the most interesting functionality, including a brief list of the code routines; then the *Bluetooth Modules* (sec. 2.3) are described, showing their capabilities, and last the *Zumo*

Robot (sec. 2.4) is depicted, including some technical details and the behavior of its firmware.

In the third chapter, *Data Acquisition*, all the process required to acquire data is described. Three sections composed this chapter: *Performed Movements* (sec. 3.1) explain how the final set of movements has been obtained and by which gestures is composed; *Electrodes Placement* (sec. 3.2) describes in the most accurate possible way the muscles on which the electrodes have to be placed, including some initial test; *Acquisition Protocol* (sec. 3.3) then introduces the complete procedure to be followed to acquire the data, including some placement issues and the way to solve them in the easy way.

The forth chapter, *Classification Algorithms*, describes the software that has been used to implement the classifier both offline and online. The *Offline Training* (sec. 4.1) has been performed on a Matlab platform, using some backpropagation algorithms and finally obtaining the desired parameters, while the *Online Prediction* (sec. 4.2) has been made directly on the Apollo2 board, enhancing its computational capabilities thanks to an ARM native library.

The fifth chapter, *Experimental Results*, makes an overview on the performances obtained by the classifier and on the ones of the overall system. The division in section is straight-forward: *Classifier Accuracy* (sec. 5.1) takes in account the accuracy performances of the classifier, analyzing some results; the *System Latency* (sec. 5.2) has been measured on the board and is divided between the classifier one and the one of the whole system; last, *Power Consumption* (sec. 5.3) is analyzed, considering both the Apollo2 and the three acquisitions channels.

Last, the *Conclusions* chapter closes the paper, summarizing the work described in the previous chapters, and introduce some possible *Future Works* (sec. 6.1).

Acknowledgements

A sincere thanks to Prof. Maurizio Martina for giving me the opportunity to work on this thesis project, to Dr. Paolo Motto Ros for constantly helping me when I needed and to Prof. Danilo Demarchi for immediately welcoming me into MiNES group family.
I also want to thank Fabio and Rossana, together with the friends of the VLSI and the various thesis mates at the Civera lab, for making me feel at home at all times.
Thanks to my family for supporting me throughout my studies, to my beloved Serena, who has put up with me in recent years, and to Stefano, Paolo and Filippo for having shared with me the first far-from-home years.
A special thanks goes then to Fabio, Luca, Martina, Eva and Valentina, to all the university colleagues and to the friends of the high school, without whom I would not be here and I would not be the same.

Un ringraziamento sincero al Prof. Maurizio Martina per avermi dato l'opportunità di lavorare a questo progetto di tesi, al Dr. Paolo Motto Ros per avermi seguito costantemente e al Prof. Danilo Demarchi per avermi da subito accolto nella grande famiglia del MiNES group.
Voglio inoltre ringraziare Fabio e Rossana, insieme agli amici del VLSI e ai vari compagni di tesi al Civera, per avermi fatto sentire a casa in ogni momento.
Grazie alla mia famiglia per avermi supportato nell'intero percorso di studi, alla mia compagna di vita Serena, che mi ha sopportato in questi ultimi anni e a Stefano, Paolo e Filippo per aver condiviso con me le prime esperienze fuori casa.
Un ringraziamento particolare va inoltre a Fabio, Luca, Martina, Eva e Valentina, a tutti i compagni di studi universitari e agli amici del liceo, senza i quali non sarei qui e non sarei lo stesso.

Andrea

Contents

List of Figures

List of Tables

List of Acronyms

ADC	Analog to Digital Converter
App	Application
ATC	Average Threshold Crossing
CNS	Central Nervous System
DAC	Digital to Analog Converter
EMG	ElectroMyoGraphic
GPIO	General Purpose Input/Output
IAP	Intracellular Action Potential
IEMG	Integrated EMG
LPM	Low Power Mode
MAV	Mean Absolute Value
MCU	Micro Controller Unit
MDF	Median Frequency
MNF	Mean Frequency
MUAP	Motor Unit Action Potential
RMS	Root Mean Square
sEMG	Surface ElectroMyoGraphic
SPI	Serial Peripheral Interface
SSI	Simple Square Integral
UART	Universal Asynchronous Receiver Transmitter
ULP	Ultra Low Power
VAR	Variance
WL	Waveform Length

Chapter 1

Introduction

Nowadays tech market is becoming day by day more competitive. Continuous technology improvements regarding power consumption and product area have brought embedded systems to a new era. Customers are requesting increasingly performing products, with smart objects like watches and glasses that are on top of their popularity [1].
Researchers and developers have recently focused their work on these topics, trying to continuously increase computational capability of their processor, while keeping small the final size of the device. According to this, wearable devices have been involved in multiple type of applications, including gesture recognition.
Using the correct type of data, acquired in the proper way, gesture recognition can be employed in many field [2], from unblocking the mobile phone smiling at it, to simulating movements in a video game, or also to command a prosthetic hand, or whatever.

In biomedical applications the most used technology to acquire gestures is the surface ElectroMyoGraphy (sEMG), which is made applying non invasive electrodes on the skin. Typically, for upper limb prosthesis or to control an external device, the electrodes are placed on the forearm, in order to collect muscular activities when the hand is moved. The recognition of the gestures is usually made by means of a machine learning technique, often a Support Vector Machine (SVM). Unfortunately, to provide enough data to the classifier, the sEMG signal has to be entirely sent to a processing unit, in order to extract the needed features, or it can be processed directly on the acquisition boards, but it would take a lot power consumption [3], and the so made device would not be suitable for wearable applications.

A possible solution is to implement the Average Threshold Crossing (ATC) event-driven technique. This particular feature extraction is obtained configuring a threshold above the rest value of a signal and raising the output every time the threshold is crossed. The resulting *quasi-digital* signal contains the information

about the muscular activation in the time domain, but its digital shape makes it of easy interpretation by a microcontroller.
It has been demonstrated in previous works that the number of generated TC events is proportional to the effective muscular activation [4]. Moreover, this event-driven approach considerably relaxes hardware and software resources and decreases the power consumption of the entire system [5], extending the life time of an external battery supply.

In this thesis work the ATC technique is implemented with three acquisition channels, directly on the subject forearm. The acquired TC events are sent by wire to a microcontroller, to avoid excessive power consumption. The processor used to elaborate the data is an ARM Cortex-M4F, selected due to its ultra low power capability and its optimized floating multiplication unit. A fully-connected Neural Network (NN) has been chosen to classify acquired TC events and predict the performed hand gesture.
An *in vivo* study has been made, involving 25 healthy people, of different genre and age. Six hand gestures have been executed: Wrist Extension, Wrist Flexion, Radial Deviation, Ulnar Deviation, Grasp and Idle (i.e. the rest position of the hand). The experimentation has been divided in two steps: first, twenty people have been employed to acquire TC data to train the classifier, then the remaining five have tested the online system, configured with the already trained NN.
The output class is then sent to a small tank, via Bluetooth 3.0 modules, to demonstrate the effective correlation between the hand movement and the obtained output. The tank has an Arduino shield on board and it has been configured to execute simple commands.

In the following sections a brief introduction about the background of the thesis is provided to the reader. In the first two sections fundamentals about the muscular system and sEMG signals are introduced. Then an overview about the ATC technique and how it has been applied in the recent past is given. Last, gesture recognition and machine learning technique are illustrated.

1.1 The Skeletal Muscular System

The main actors in the execution of our movements are the skeletal muscles, which generate action potential when stimulated by the *Nervous System*. The tension produced in the muscle by this stimula induces the effect which is usually call *muscular force*. The entity of this force is dependent by multiple factors, but it is possible to denote that it is directly proportional to the cross section of the muscle which has taken part in that movement.

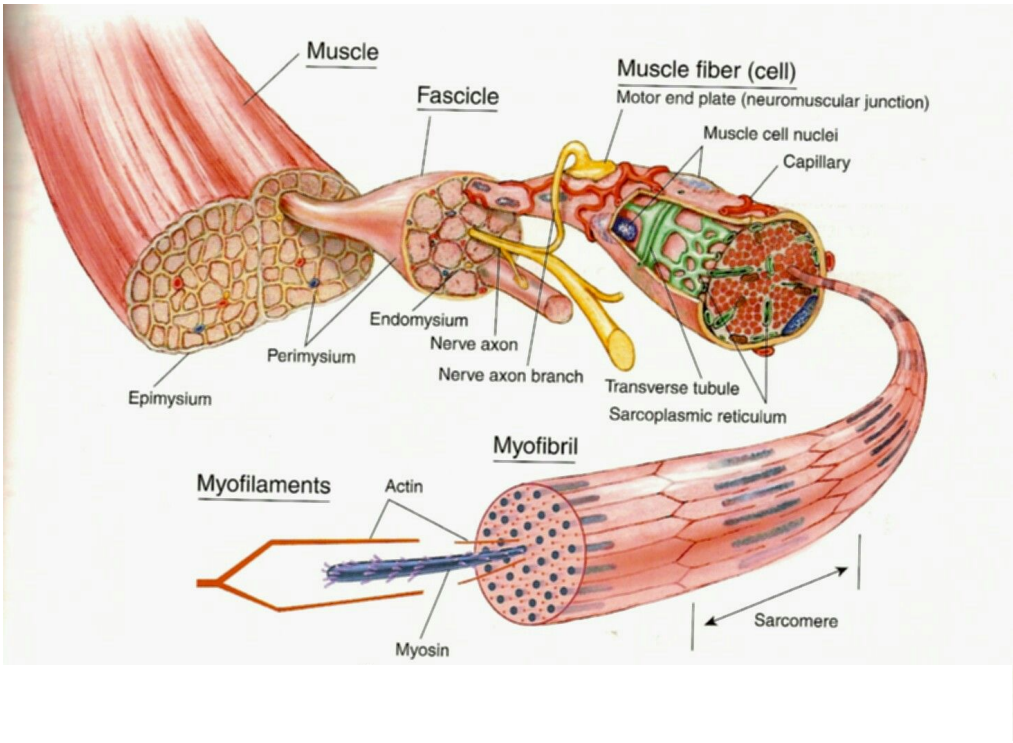


Figure 1.1: Structure of the skeletal muscles. [6]

All skeletal muscles are voluntary, they belong to the striated tissues family and are tied to the skeleton bones with their tendons, which allow them to transmit their power and produce movement. Their structure is not so elementary, in fact it is composed by a lot of subunits (shown in Figure 1.1). Starting from the macroscopic level, and going towards the inner part, it is possible to distinguish the muscular fascicles, protected by a connective membrane called *epimysium*, which is the appendix that connect the whole muscular body to the tendons. Fascicles, which are divided one from the others by the *perimysium* membrane, are in turn composed of muscular fibers, each one protected by the *endomysium*, the inner membrane of the muscle. This last membrane also contains the extracellular fluid and all the nutrients needed by the fibers to survive.

In the inner part of the muscle, inside each fiber, multinuclear cells, sarcolemma and sarcoplasm, which is actually the cytoplasm of the muscular cells, are located. Quality, quantity and allocation of fibers internal biomaterial depend on the nucleus type, while muscular characteristic, like reactiveness, resistance and power, are related to the different proteins they produce [7].
In the cytoplasm, in particular, mitochondria are in charge of producing the necessary energy, transforming the substances that arrive via the circulatory system into the ATP (*Adenosine TriPhosphate*). Quantity of mitochondria can vary depending on the function that the fiber has in the body, up to a 20% of the total volume when the fiber is particularly oxidative.

Going into the fiber, it is possible to found the myofibrils, organized in a parallel scheme. They are composed by the sarcomeres, which are instead placed serially one to the other and are considered the basic functional units of the skeletal muscle. Sarcomeres are in turn made up of parallel overlapping matrices of myosin and actin, which are proteins shaped in thick and thin filaments, respectively.

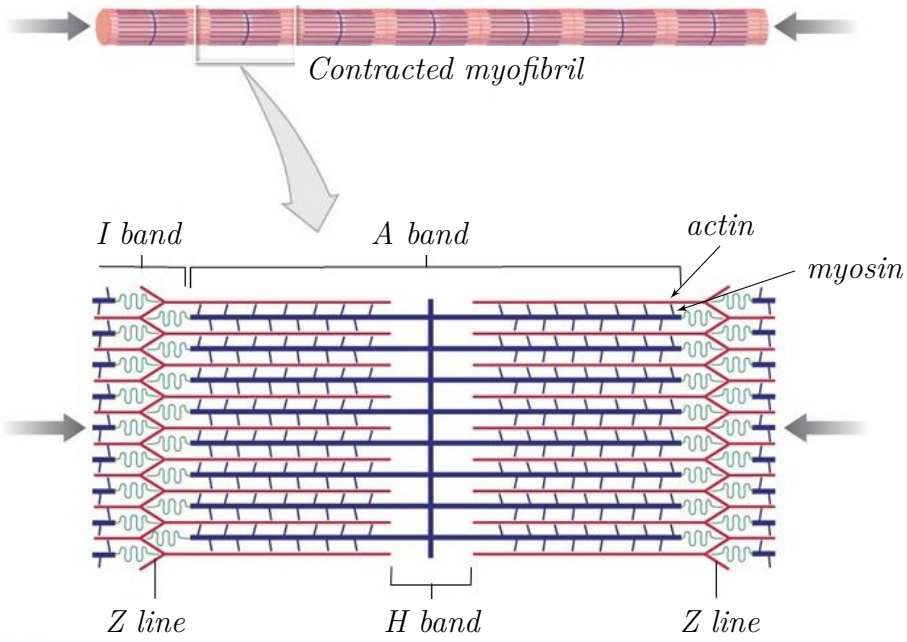


Figure 1.2: Inner view of a sarcomere. [8]

The sarcomeres (Figure 1.2) are studied by means of the polarized light microscopy and five different areas can be distinguished:

- **Z line** (from german *Zwischen-Scheibe*): it is a dark protein-based line, which connects actin filaments of one sarcomere to its neighbors' ones.

- **I band** (from Isotropy): it is the outer band of the sarcomere, made only of actin filaments, and it is the lighter one, when seen with the microscope.
- **A band** (from Anisotropy): it is the darker band of the sarcomere, due to the presence of both actin and myosin filaments.
- **H band** (from german *Helle-Scheibe*): it is the central area of the sarcomere and it is slightly lighter than the A band, due to the absence of actin.
- **M line** (from german *Mittel-Scheibe*): it is a dark line right in the middle, made of cytoskeletal proteins, which connect myosin filaments of the two side.

In mammalian muscles each sarcomere can be long 2–3 mm. When a contraction is performed the I band and the H band become thinner, while the A band preserve its width.

1.1.1 Skeletal Muscle Physiology

Muscular fibers are activated by the Central Nervous System (CNS), which sends signals through the motor neurons. Inside the spinal cord, α-motoneurons receive signal from the CNS and send the activation order to groups of muscular fibers, through their axons. This functional unit, composed by one single motor neuron and one group of fiber, is called Motor Unit (MU, in Figure 1.3).
Each MU works on a continuous cycle of polarization and depolarization. Without any conduction, the transmembrane differential voltage is $-70\,\text{mV}$, also called rest potential. This voltage is kept constant thanks to the sodium-potassium pump, a special protein with control the traffic through the membrane of the axon. This protein makes the Na^+ ions go out and accepts only the K^+ ions in.
When the neuron is is stimulated, the action potential propagates through the axon, opening the sodium doors (allowing the entry of Na^+ and increasing the differential voltage to 40 mV). The subsequently opening of the potassium doors (which allows the output of K^+), allows the repolarization of the membrane, which reduce again its potential, reaching the resting value of $-70\,\text{mV}$. During this process, the neurotransmitters stocked in the neuron are released and can reach the muscle fibers. Though, after a short time interval, the protein responsible for the transmission closes the path and ends the process.
As soon as the transmitters reach the terminal head of the axon, they directly stimulate the fibers innervated from that neuron, causing immediately a contraction.

Each muscles has all the α-motoneurons gathered together. The number of MUs is dependent by the force precision necessary to perform a defined movement and, concerning humans, can vary between 100, for a small muscle (e.g. in the hand), and 1000 in a limb bigger muscle.

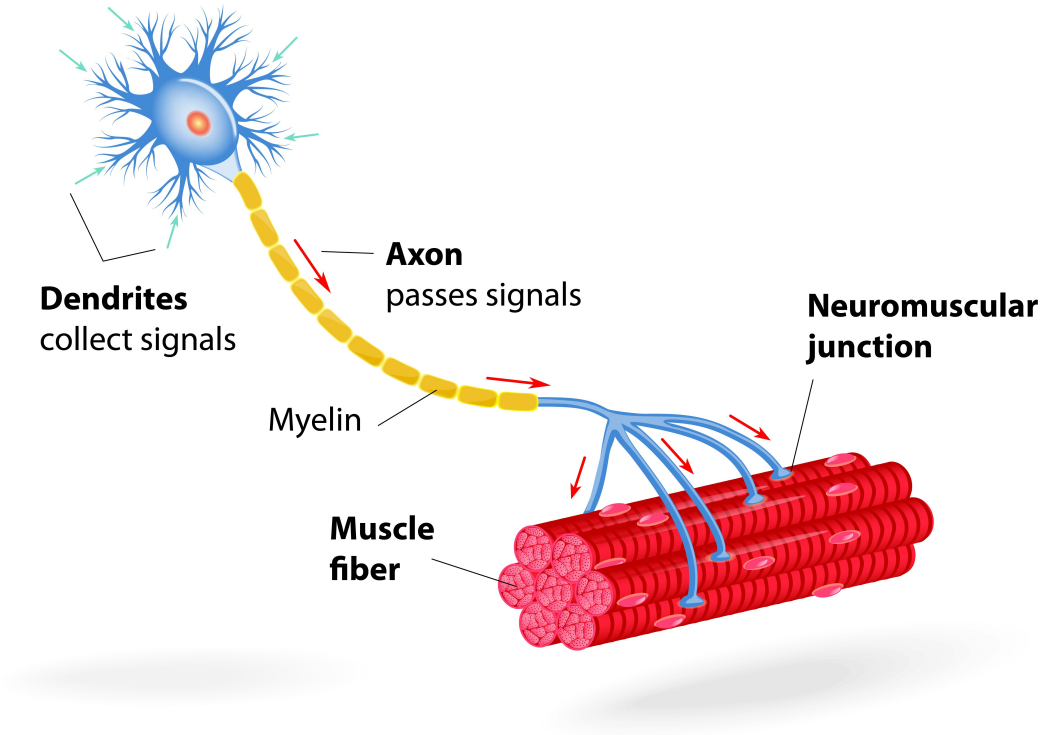


Figure 1.3: Motor Unit anatomy. [9]

All the fibers work together, both in terms of generated force and of activation timing. They can not produce a small or intermediate action, if they activate they have to release all their power. Once the potential get over the voltage threshold, the contraction is active, lasts few seconds and then stops.
The difference between an harder and a softer exercise does not produce a difference in the force of the activation, but causes instead a repeated stimulus in the muscle. If the stimula are really one next to the other, the muscle suffers the maximum activation possible, called *tetanus*. After a certain time of activation, even if the stimulus is still active, muscular fatigue starts to arise and the muscle has to relax itself.

1.2 ElectroMyoGraphic (EMG) Signal

The action potential caused in the MU by the CNS, which has been discussed in Sec. 1.1.1, is responsible of some fluctuations in the electromagnetic field around the muscle. The discipline that studies and collects these fluctuations is called ElectroMyoGraphy (EMG) and can be performed using invasive or non-invasive methods:

- the **Deep EMG** signal represent the activity of one single MU, using needles or thin wires electrodes, inserted in the skin. This technique is generally preferred when the aim of the acquisition is to study the physiology and pathology of MUs. At a central level it can be use to evaluate MUs recruitment and activation path through the body, while at a peripheral level it is possible to study injury effects, loss of muscular innervation or effects of neurodegenerative diseases.
- **surface EMG (sEMG)** is performed using superficial electrodes, applied on the skin, and it is so able to acquire signals from more than one MU. This technique is more useful when the aim is to:
  - characterize a movement on a temporal base, studying the duration of the muscular activation;
  - apply biofeedback techniques to inform a patient of how one of its muscle is contracted;
  - obtain information about the global activity of a muscle, or group of muscles;
  - control external actuator;
  - measure the EMG signal in all the cases when it is not considerable to put electrodes in the patient, in order to perform fast and non invasive application, like in rehabilitation or in medical visit for sport practice.

1.2.1 The surface EMG signal

The surface ElectroMyoGraphic (sEMG) signal can vary its amplitude between 0 mV and 10 mV and has a frequency spectrum in the range 6–500 Hz, with the major power contribution between 20 Hz and 150 Hz.
Unfortunately, this particular signal is affected by multiple noise sources:

- the friction between the skin, the immediate under skin substrates and the electrode generates a noise with a frequency usually lower than 10 Hz. This contribution, called **movement artifact**, can easily be filtered in the preprocessing phase, being out of the sEMG spectrum;

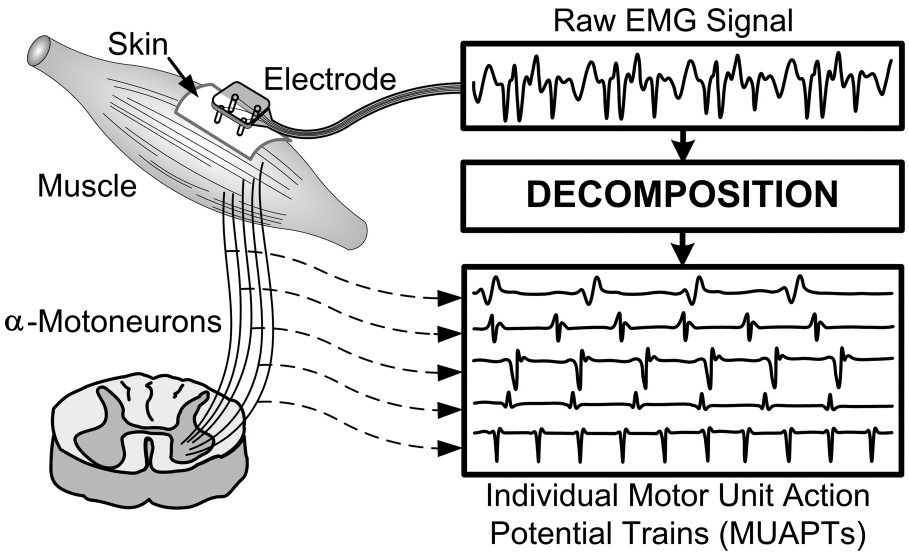


Figure 1.4: *Decomposition technique* of sEMG signal. [10]

- the presence on a body area of multiple muscles, like in the forearm, can make difficult the acquisition from a single muscle. This **cross-talk** can not be filtered, unfortunately, but can be lowered with an accurate placement of the electrodes;

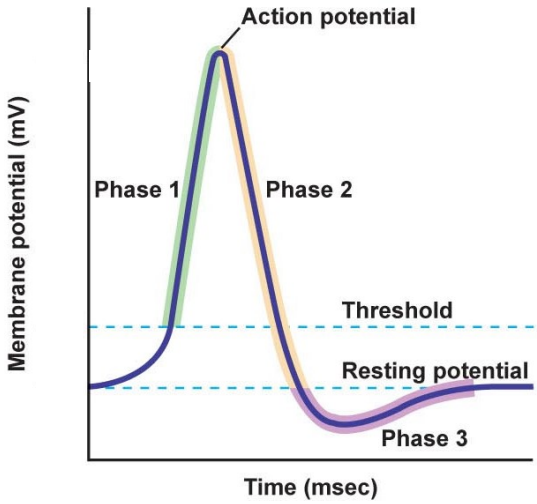
- **muscular fatigue** is another contribution to take in account. In fact, as written in Sec. 1.1.1, early or later the muscle activity will start o decrease, causing the sEMG to have a lower frequency and a falsified amplitude;

- all the **external contributions** can be dangerous for a clean acquisition. In fact, the voltage supply itself would introduce in the system a 50 Hz noise, which is often maintained in the circuit because is in the frequency spectrum.

Electrical Model

The signal acquired with superficial electrodes is the summation of the action potentials generated by the depolarization of each fiber of a MU, and it is called MUAP (Motor Unit Action Potential). Starting from the nerve insertion point of each muscular fiber, two depolarization area propagate towards the opposite extreme with a speed of 3–5 $\mathrm{m\,s^{-1}}$. These areas are not spatially aligned, neither in the time they reach the acquisition electrodes, due to different point of innervation and different propagation speed between one fiber and the other. The resulting signal is the summation of all these contribution (as shown in FIgure 1.4). A repeated activation of a MU generates a train of action potential, called MUAPT.

The behavior of a single Intracellular Action Potential (IAP) can be approximated adding together simple contributions (as shown in Figure 1.5). Generally speaking, IAP is structured in an initial depolarization phase, followed by a repolarization phase and ending with a longer iperpolarization phase. Its shape can vary depending on muscle conditions, in particular, when the fatigue arises it is difficult to distinguish among the different phases. In fact, the repolarization phase become slower, bringing to a more wide peak.



Three phases of IAP:
Phase 1: *Depolarisation*
Phase 2: *Repolarisation*
Phase 3: *Hyperpolarisation*

Figure 1.5: Intracellular Action Potential. [11]

1.2.2 sEMG Feature Extraction

In sEMG analyses it is possible to distinguish three different features group: in the time domain, in the frequency domain, or in the time-frequency domain. The last domain is not considered in this introduction, due to its high complexity that make it not suitable for the training of a classifier. Only the features of the first two domains are then reported below.

Features in the time domain

This type of features extraction is largely use in literature, thanks to their straightforward calculation, without needing any transform. Mayor disadvantage is that they assume no variations in the signal frequency.

- **Root Mean Square: RMS** is modeled as amplitude modulated Gaussian random process whose RMS is related to the constant force and non fatiguing

contraction.

$$RMS = \sqrt{\frac{1}{N}\sum_{n=1}^{N} x_n^2}$$

- **Mean Absolute Value: MAV** is calculated by taking the average of the absolute values of sEMG signal.

$$MAV = \frac{1}{N}\sum_{n=1}^{N} |x_n|$$

where the terms have the same meaning as above.

- **Mean Absolute Value Slope: MAVSLP** is defined as the difference between the MAVs of adjacent segments of EMG signal.

$$MAVSLP_i = MAV_{i+1} - MAV_i$$

where i is the index of considered MAV.

- **Integrated EMG: IEMG** is defined as the area under the curve of the rectified EMG signal.

$$IEMG = \sum_{n=1}^{N} |x_n|$$

where N denotes the length of the signal and x_n represents the sEMG signal in a segment.

- **Variance of EMG: VAR** uses the power of the sEMG signal as features. In fact, it is calculated using the following formula:

$$VAR = \frac{1}{N-1}\sum_{n=1}^{N} |x_n|^2$$

- **Simple Square Integral: SSI** is the summation of absolute square energy of sEMG signal in time domain.

$$SSI = \sum_{n=1}^{N} |x_n|^2$$

- **Waveform Length: WL** is the cumulative length of the waveform over time segment and can be mathematically represents as:

$$WL = \sum_{n=1}^{N} |x_{n+1} - x_n|$$

All of these features above (RMS, MAV, MAVSLP, IEMG, VAR, SSI, WL) are computed based on sEMG signal amplitude.

Features in the frequency domain

- **Median Frequency: MDF** is the frequency value that divides signal power spectrum in two equal areas:

$$\sum_{j=1}^{MDF} P_j = \sum_{j=MDF}^{M} P_j = \frac{1}{2}\sum_{j=1}^{M} P_j$$

  where P_j represents signal power spectrum value corresponding to j frequency and M is the maximal spectrum frequency.

- **Mean Frequency: MNF** is an average frequency which is calculated as the sum of product of the sEMG power spectrum and the frequency divided by the total sum of the power spectrum:

$$MNF = \sum_{j=1}^{M} f_j P_j \Big/ \sum_{j=1}^{M} P_j$$

  where terms have same definition of previous case.

Both these frequency domain features describe fatigue in an appropriate way. MDF is usually less affected by noise than MNF, but MNF has an higher value.

1.3 The Average Threshold Crossing Technique

Acquisition systems already on the market have usually a standard structure:

- the electrodes placed on the skin surface allow the acquisition of the sEMG signal with the bipolar configuration;
- an analog circuit is responsible of the preprocessing of the signal, by means of amplifiers and filters, which let pass only the useful frequency contribution, blocking the undesired ones;
- an Analog to Digital Converter (ADC) converts the signal from its analog shape to a digital information, readable by common logic;
- last a microcontroller takes the digital information from the ADC and can process it onboard or just transmit it to a computer.

However, the IAP shape and behavior have made researchers think about a possible acquisition system based on a threshold. In particular, the Average Threshold

Crossing (ATC) technique has been formalized. The ATC is an event-driven technique: when the myoelectric signal crosses a predefined threshold an output signal is arisen. This technique, proposed by [4, 5, 12] is completely different with respect to literature. It is in fact implemented directly in hardware by means of a voltage comparator Two signals are compared to determine the output state: one is the threshold, externally provided, and the other is the sEMG signal, already filtered and amplified. The output signal of the comparator is a *quasi-digital* signal, in fact contains the information in the time domain and not on its analog value. In particular, the output would be high when the sEMG value is above the threshold, and low otherwise. A more detailed scheme of a possible acquisition channel is described in Sec. 2.1

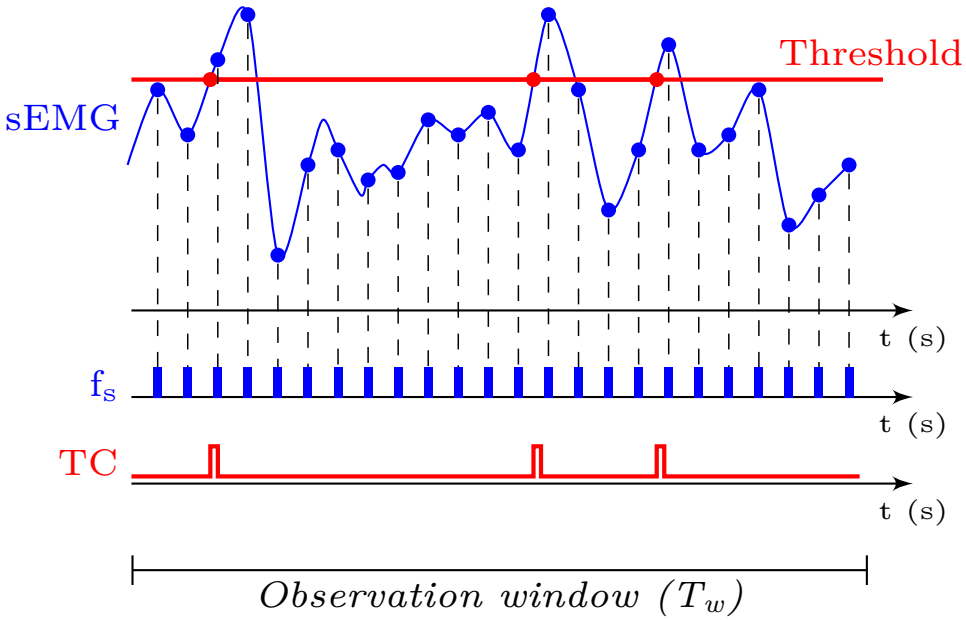


Figure 1.6: Payload comparison between ATC and standard sampling.

The ATC parameter is then the number of times the sEMG signal crosses the threshold, in a defined time period, divided by the duration of the period itself. Considering that the sEMG signal is usually wireless sent to an external processing node, this approach allows to drastically lower the power consumption of the transmission. In Figure 1.6 a comparison between the two payload is shown. The difference is clear, the number of packets sent using this approach is far lower with respect to the constant payload of the standard sampling.
In this representation the threshold is fixed, but it has already been demonstrated that a dynamic threshold could fit as well, being adaptable to each environment

and further reducing the power consumption. The advantages of this event-driven transmission can be summarized in two main contribution:

- a lower power consumption, thanks to a reduced and not constant sampling frequency, and a resulting lower necessary throughput.
- a simpler acquisition hardware; in fact, the ADC becomes useless and the hardware can be reduced to a size suitable for wearable applications [13, 14].

Although, this technique is not perfect and actually can not be used for diagnostic reasons or similar. In fact, the original information contained by the sEMG is completely destroyed and no reconstruction is possible. However, an earlier study as demonstrated the correlation between the ATC parameter and the strength of a performed movement, allowing some comparison on that research field. The ATC is now during a validation phase for what concern more complex tasks, like gesture recognition, which are based on the strength applied but require a certain amount of precision.

1.4 State of the Art

Gesture recognition ia an hot topic in recent days researcher work, with a wide application range: depending on the aim of the implementation, it is possible to found in literature a lot of different approaches. They all have something in common, though: the modules to optimize most, in order to improve the classifier and have an effective gesture recognition, are data preprocessing, features extraction and classification methods. In the following pages are shown the most important literature works which use sEMG signals applied to gesture recognition.

Momen et al. [15] group has implemented a classifier which permit the user to make voluntary movements, without any restrictions or predefined set of gesture. They used only two sEMG features to obtain a result: the RMS and its logarithm, both calculated in real time on a 200 ms window. In the testing phase they demonstrated that the classifier worked fine with the movements provided by the user, having obtained a final average accuracy of 87%, with a standard deviation of about 13%. The algorithm used in this work was the *fuzzy clustering C-means*.

In a later work, *Shenoy* et al. [16] have reduced the dimensions of the features vector, by using only the RMS, but to obtain a good result they had to assign a predefined set of movements. Features extraction has been performed on a windows made of 128 samples, with a sampling frequency of 2048 Hz. The acquisition window is slightly wider than 60 ms, being really suitable for a robot control in real time. This work has also introduced a new type of classifier, different from the one

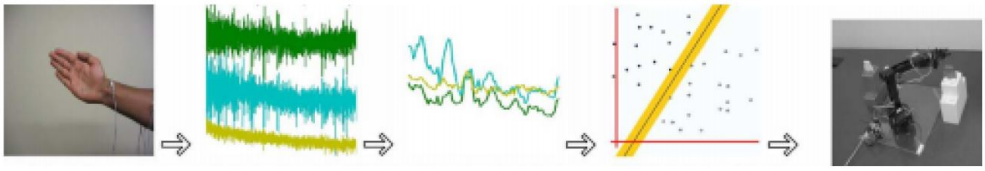


Figure 1.7: Acquisition protocol with standard sEMG features. [16]

used in the previous work. This new classifier is called Support Vector Machines (SVM), and is able to disclaim between classes using complex non linear functions. Using a cross validation method this classifier can reach an accuracy of 90%. In Figure 1.7 the system flow of this work is shown.

From these good results, the literature continue mainly on this road. In particular, the work of *Lucas* et al. [17] is interesting because slightly differs from the main flow, using the Discrete Wavelet Transform (DWT) as unique feature. This approach improve the use of the standard DWT by introducing a minimum error classifier, which enhance the general performances, but reduces the flexibility of the system and increases the computational time. The used classifier is again the SVM, with an overall performance fo 95%, slightly better with respect to earlier works.

Another work that perform a classification using as feature the DWT is the work of *Kartsch* et al. [18]. They implement a electrode wristband and exploit low power possibility by using a ultra-low power microcontroller as processing unit and adding a solar harvester. The classifier used is always the SVM, which in this case classifies among five gestures from four input channels. The feature extraction process is performed directly on the wristband, but some elaborating unit are still needed, consuming extra power with respect to the needs of the classifier alone.

Also a commercial article [19] has used a wristband to perform gesture recognition, but with worst results than the previous one (in fact, it is now out of production). They implemented a wristband with eight couple of electrodes, with embedded processor and transmitter, useful for example to virtually control the mouse of a computer. They use a Radial Basis Functions (RBF) Neural Network (NN), reaching only an accuracy of 66% classifying six gestures. They had also other features, like accelerometer and gyroscope, but the product had not enough success.

In this thesis work, the SVM has not been used, thinking that its high performances are not needed for this simple task, while it has been decided to implement a simpler fully-connected NN, which has been considered suitable to enhance the ATC capabilities of having a small payload and generating few data. In the following sections, the system will be introduced.

Chapter 2

System Configuration

The system used in this thesis is composed by four main parts, as shown in Figure 2.1: three acquisition channels, which filter and amplify the sEMG signals acquired from the surface electrodes; the ultra low power AmbiqMicro Apollo2 board, with an ARM Cortex-M4F processor that is in charge of all the computations; two Bluetooth 3.0 modules which allow the communication between the Apollo2 board and the last part of the system, a Zumo mini-robot, used to verify the effective correctness of the classified movements.

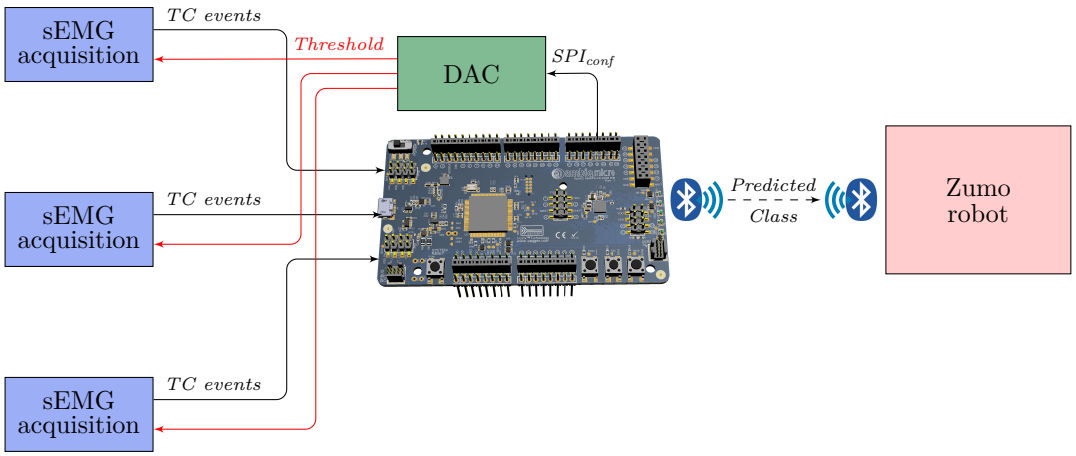


Figure 2.1: System schematic.

2.1 Acquisitions Channels

The acquisition of sEMG signals is performed applying standard surface electrodes on the skin of the forearm; in particular, we used the pre-gelled Ag/AgCl H124SG

Covidien electrodes, with circular shape and a diameter of 24 mm. The so obtained signals are brought to the channels using shielded wires with a clip head, directly connected on the acquisition board.

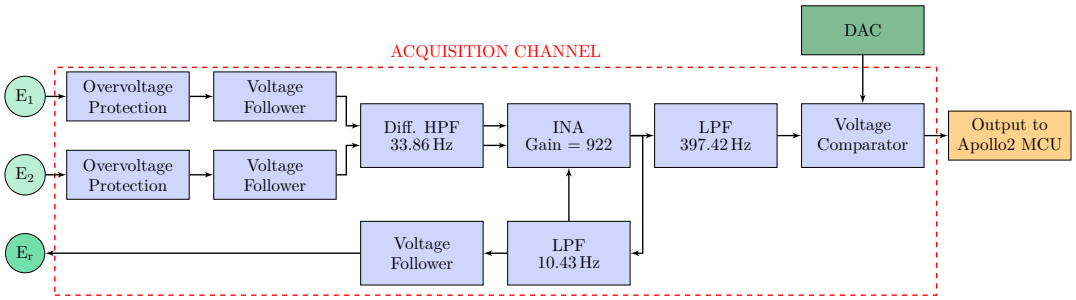


Figure 2.2: Structure of the acquisition board.

The channel itself is a full-custom Printed Circuit Board (PCB) made only with off-the-shelf components, obtained improving an older version, composed of four channels and a microcontroller on a single PCB [14]. As shown in Figure 2.2, the channel has eight distinguishable stages:

- Overvoltage protection obtained with four different components:
  - two ESD5Z3.3 ElectroStatic Discharge (ESD) suppressor Zener diodes (Texas Instrument™ (TI));
  - two CMAD6001 ultra low leakage switching diodes (Central Semiconductor Corp.™);
  - two BLM15AX102SN1 noise suppression ferrite beads (Murata);
  - one TPD2E001 two-channel Transient Voltage Suppressor (TVS) protection diode array (TI).
- Impedance decoupling using an OPA2347 dual voltage follower integrated circuit.
- A differential high-pass filter to reduce movements artifacts, with cutoff frequency of 33.86 Hz.
- Differential amplification of the two sEMG signals, referencing them to a voltage of 1.65 V, by means of an INA321 with a gain of 922.
- A low-pass filter on the negative feedback of the INA321, with a cutoff frequency of 10.43 Hz, necessary to reject eventual noise inserted by the INA itself.

- Another voltage follower, between the 1.65 V source and the reference electrode, needed to prevent noise injection from the arm.
- One more low-pass filter, made with an OPA333, used to reduce high frequency noise, above the desired spectrum.
- Last, a TLV3691 voltage comparator, generates the TC events, comparing the pre-processed sEMG signal with an adjustable threshold that is usually set to 1.8 V. A 30 mV hysteresis ensures a stable commutation between the digital high and the digital low state, letting us to obtain a *quasi-digital* signal.

The reference voltage needed by the comparator is provided by an external Digital-to-Analogue converter. The chosen module is a Maxime MAX5742 which provides a compatible Serial Pheripheral Interface (SPI) to easily connect with the µP and four output voltage channels to set any different threshold value for each acquisition board.

Acquisition channels have two output signals:

- EMG obtained after last filter stage, usable for visual purpose or to check the correct behavior of the board;
- *quasi-digital* signal obtained after the voltage comparator, connect on an input of the microcontroller, with basic interrupt interface.

2.2 Apollo2 board

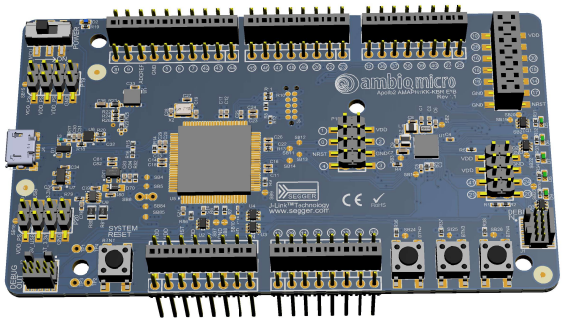


Figure 2.3: The AmbiqMicro Apollo2 evaluation board.

The main core of the whole system is certainly the AmbiqMicro Apollo2 EVB (Figure 2.3), which supplies power to the three AFEs, to the external DAC and to the BT module, as well as doing all the computations needed to define the right class

depending on the inputs. The processor embedded in the Apollo2 EVB is an ARM Cortex-M4F, which is a 32-bit microprocessor (µP) "designed to enable developers to create cost-sensitive and power-constrained solutions" [20].

The Apollo2 has been chosen for its really competitive technical characteristic, especially regarding low power consumption. In Table 2.1 main features are reported [21].

Table 2.1: Apollo2 technical sheet

Max operating frequency	48 MHz
MCU	32-bit ARM Cortex-M4F
MCU min power	10 µA MHz^{-1}
Flash/SRAM	1 MB/256 kB
VDD	1.8–3.6 V
I/O	I^2C/SPI (6x) UARTS (2x)

For this application a frequency of 24 MHz has been chosen for the main clock, while the low frequency, more precise, 32.768 kHz crystal has been used for the time window implementation needed by the ATC. The buck converters have been enabled to guarantee a really low power consumption by the µP.

2.2.1 Firmware

The firmware for the MCU has been written using the Keil µVision IDE v5.24.2.0, as suggested from AmbiqMicro developers due to its high compatibility with the ARM products family. In fact, a lot of specific ARM libraries are available directly in Keil Pack Installer, making possible to add or delete any library package. For this application the Digital Signal Processing (DSP) library has been chosen, according to its low computational cost and to its easy usage; this library is an ARM-native part of the CMSIS package [22] and provides very useful support for matrix calculations. Moreover, lots of board support packages are supplied by AmbiqMicro itself, permitting the developer to use high level functions to interact with each single small component of the system, even performing complex tasks.

The program is as usual divided in three main parts: first all the needed constants and variables are defined, then useful routines are implemented and last the main function executes the desired commands. Here there is a brief list of the

routines used in the main functions, in order as they are called:

- Initial configuration: some basic function are called depending on the value of environmental constants; if high frequency is required, the MCU would be set to 48 MHz instead of 24 is enabled if needed, low power mode is entered, activating the buck converters and the debug interface is configured, if the mode is needed.

- Led initialization: LEDs are configured according to library functions, they are made controllable from the µP and then switched off.

- GPIO configuration: the three pins required to collect data from the three AFEs are activated as input, with interrupt on the rising edge. For each of them an appropriate function is registered in the interrupt service table. No hazards occur in case of multiple concurrent interrupt are received. The interrupts are then cleared for security reasons and the GPIO master is enabled.

- Timer configuration: a timer is set according to ATC requirements. The best trade-off to generate a 130 ms time window without dissipating to much energy it is to use the external high precision low frequency clock (F = 32.768 kHz). A interrupt is then set to a compare value of 4260, obtaining a window of 130.005 ms, and corresponding service function is registered. *REPEAT* mode is activated, so the timer restarts from the beginning after the interrupt occurs.
  If a time measurement is required for performance reasons, another timer would be activated, with a frequency a quarter of the selected Hard Clock (HCLK), tipically 6 MHz.

- SPI configuration: the SPI interface is configured to communicate with the external DAC.

- UART initialization: if the tank is in use, uart is set to communicate with BT module. Parameters are reported in Sec. 2.3.

- Matrices initialization: if online mode is selected, all the matrices needed to store intermediate value and parameters of the NN are paired with their relative DSP instance.

- Threshold setting: this routine takes as input a number of channel and a value in V and sends data to the DAC through the SPI. If the channel value is set to '4', all the channels are configured together. Two data packets are used (8 b each) according to parallelism of MAX5742, one containing control characters and few bits of the threshold value and the other with only the threshold.

- Movement loop: a *for* loop is implemented and used if the acquisition mode is active. Depending on the number of acquisitions that have been selected, the loop will separate each movement from the other, letting the user understand when a change in the gesture has to be performed.

- Running loop: the program enters then in a *while* loop, basically evaluating the NN output and going to deep sleep mode till the next interrupt(s). In case time or current measurements are desired, some variables are set in the appropriate way.
  NN behavior will be debated in Sec. 4.2.

The board could be powered via USB cable or with an external supply, like a battery or a voltage generator, to be connected directly on the power pin on the power header of the board.

2.3 Bluetooth modules

In order to satisfy the will to show the output class of the classifier in a more visible way than the embedded leds, a small friendly tank (named Zumo) has been used, constructed with the intent to participate to robot sumo battles, but used in this thesis in an harmless way. Connecting this small robot with the Apollo2 board using two Bluetooth modules allows to show in a fast efficient way the behavior of the system.

Figure 2.4: The SPBT3.0DP1, by STMicroelectronics

Two SPBT3.0DP1 bluetooth modules (shown in Figure 2.4) have been chosen to satisfy the constraints. In particular, these modules, made by STMicroelectronics, use the old Bluetooth 3.0 technology, which includes a *SPI-transparent* usemode, permitting a fast effective transmission, with few constraints. The modules are connected to the host through an UART interface, both on the Apollo side and on the Zumo side, configured with the parameters listed in Table 2.2.
Their serial profile reach a throughput of $1500\,\mathrm{kbit\,s^{-1}}$, resulting in a very fast

Table 2.2: UART configuration

Baud rate	$115\,200\,\mathrm{s}^{-1}$
Packet length	8 bit
Stop bit	1 bit
Parity	none
Flow control	none

transmission compared with the latency of the system. Moreover, the configuration is really simple (it's possible to do it also with a free BT mobile app) and the data transfer does not require any control packet or server configuration, typical of standard BT, allowing to normally send data, as if the connection was wired.

2.4 Zumo robot

As mentioned in 2.3, a small tank called Zumo, made by Pololu Robotics & Electronics [23], has been chosen to show to a possible audience the correct behavior of the system, and it is possible to power it using four standard batteries, making it free to move. The product is delivered together with an Arduino Leonardo, making the motor shield easily configurable and controllable. Arduino Leonardo is one among few products of the Arduino family that has two Serial profile on board, making possible to communicate with the computer and with the BT module at the same time.

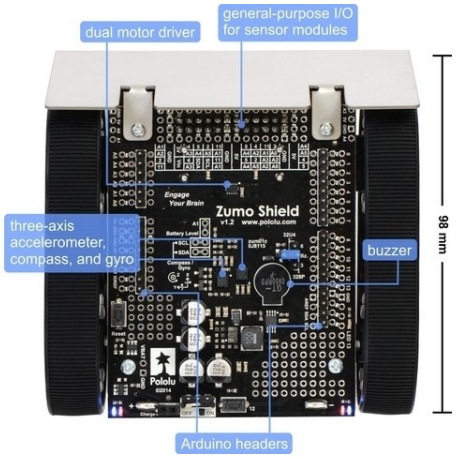


Figure 2.5: Zumo robot top view.

The robot, whose top view is shown in Figure 2.5, is composed of:

- Integrated 75:1 HP micro metal gearmotors
- Mounted array of six IR reflectance sensors
- Integrated DRV8835 dual motor drivers
- A piezo buzzer
- Integrated LSM303D 3-axis accelerometer and 3-axis magnetometer
- Integrated L3GD20H 3-axis gyroscope
- Optional user pushbutton
- 7.5 V boost regulator
- Convenient access to Arduino I/O lines
- Arduino Leonardo board.

For the purpose of this thesis only few features have been used, directly controlling the two motors and not considering any additional possibility, but for future improvements Zumo will be suitable for security checks and more sophisticated maneuvers.

2.4.1 Arduino firmware

All the commands needed by the Zumo robots are delivered from the Arduino Leonardo assembled on top of the motor shield. Arduino IDE v1.6.12 has been used to write the control code. Pololu Electronics provides a specific library (*ZumoShield.h*) to control all the Zumo peripheral, from edge detection IR to motor driver. Another library is then added, to allow communication between the Arduino and the BT module (*Wire.h*).

In a first phase of the work, the Arduino has been used to configure the two BT modules, being programmable in an easier way and at higher programming level with respect to the Apollo2 MCU. A small routine has been written to set some parameters on the modules Flash memory, in order to make them controllable from a remote device and continuing then the configuration from the personal mobile phone.

The main program has then been written, including useful control command for Zumo motors as well as a routine to play music from the on-board buzzer. Like all Arduino scripts, the file is composed of four different parts: the global scope,

where some useful variables are set, the **Setup** function, which is called only at the reset of the µC, the **Loop** function, which is continuously called during the execution of the program, and some other custom routines, used to enhance main code readability.

- In the global scope different types of variables are set: some velocity constants, used to control the speed of the motors and to adjust a small traction difference between the two track (*MAX_SPEED*, *MAX_L*, *REV_DELTA* and *MOTOR_STEP*), a lot of boolean control variables, to manage program flow (*playing*, *leftRun*, *rightRun*, *reverse*, etc.) and a character array to store all the information needed for the song.

- The **Setup** routine enables the Zumo led as output and configures the two serial connection with the computer (*Serial*) and the BT module (*Serial1*), setting them to a Baud rate of $9600\,s^{-1}$ and $115\,200\,s^{-1}$ respectively.

- The **Loop** routine initially waits for a string to be sent on the *Serial1* connection. It continues the execution either when the termination character ';' is received or when the waiting time of 500 ms is elapsed. At this point, it takes the first n-1 characters of the acquired string and it compares them to the previous value: if they are identical, it means that the selected action has not changed and it has nothing to do on Zumo, while if they are different a *switch case* is performed to choose the right routine to call:

  case 0: *turnRight*;
  case 1: *turnLeft*;
  case 2: *goForward*;
  case 3: *goBackward*;
  case 4: *playMusic*;
  case 5: *stopMoving*.

  Default state execute a reset of all the environment variables of the Zumo robot, stopping it from moving and playing, if needed. Last, the state of the variable *blinky* is checked, to decide if the led as to blink or not, used mainly for debugging reasons. The routine is then terminated and the loop restart from the begins, waiting for a string to arrive.

- The custom routines are the six reported above plus others secondary ones, used to keep the code flow clear. Some of them are related, others are independent; most of them are movement command, so they check the variables that define the current state of the two motors and they optimize the action, in order to make a gradual acceleration of the tank, without skidding. Moreover, if a needed track is already moving in the right direction, the action ignores it and acts only on the other one.
  The only routine different from the others is *playMusic*, which simply checks if

the buzzer is playing and switches its state. Every time the music is stopped, it will restart from the beginning.

The whole code is normally charged on the Arduino flash using serial USB connection.

Chapter 3

Data Acquisition

The whole system is based on the acquisition of the sEMG signal from the forearm skin surface. The placement of the electrodes has to be really accurate, to prevent misleading values to be acquired. An initial study has been performed, concerning where the electrodes have to been placed, according to biomechanics of the forearm, as well as how many of them are really needed, trying to minimize their number. Once the placement has been defined and the movements are chosen, the data acquisition could finally start.

3.1 Performed movements

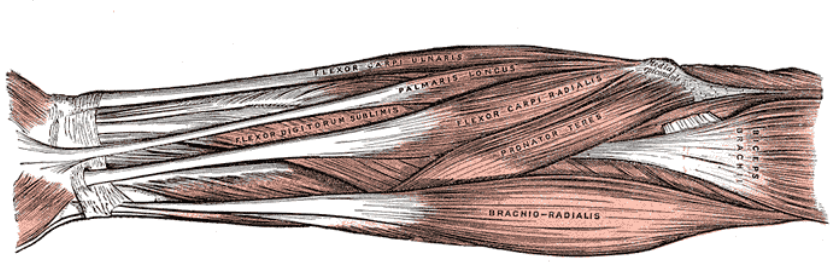
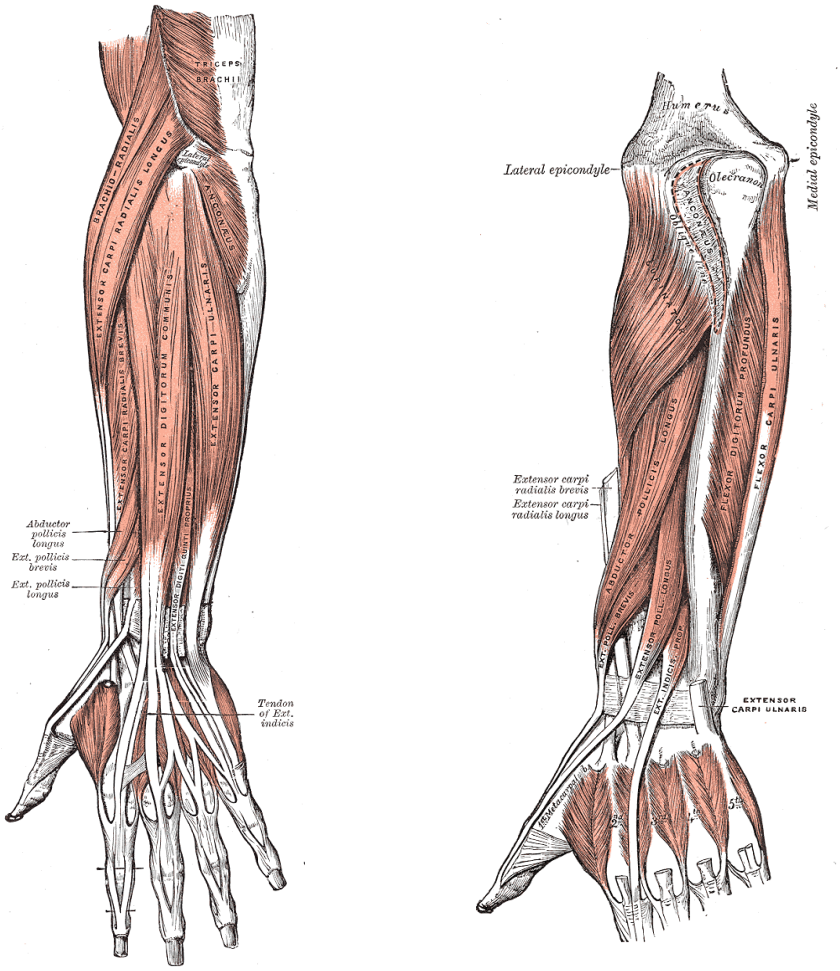


Figure 3.1: Superficial muscle on medial section of the forearm.

The list of gestures to be executed from the volunteers has been selected according to some recent literature works [16, 24] and in a way that would be suitable for controlling an external actuator, without requiring an excessive effort from the subject. The final list of movements, obtained considering that the actuator would have been a tank (2D direction are necessary), is the following: *Wrist extension* to perform right turns, *Wrist flexion* to turn left, *Wrist radial deviation* to make the tank go forward, *Wrist ulnar deviation* to make it go backward and *Hand grasp*

to execute some special task, included in order to be competitive with respect of other similar works. A sixth gesture has then been considered, the *Idle state* or *Rest position* of the hand, necessary to consider as a class when no movements are done as well.



(a) Superficial (b) Deep

Figure 3.2: Muscles position in forearm distal section.

The muscles necessary to perform the desired movements are hand extrinsic muscles (called extrinsic because they are in the forearm, out of the hand area), generally originated in the elbow area and terminating in the metacarpal area. They are divided in superficial and deep ones; the seconds, reported as an example in Figure 3.2b, are useless for this work because their position in the forearm is not suitable for using surface electrodes. Only the superficial muscles have been then taken into account, both the medial 3.1 and the distal section 3.2a.

All the below match movement-muscles have been deduced from the *Eaton Hand* online book [25].

Wrist Extension
The wrist extension is the act of move the back of the hand towards the distal forearm. The mainly used muscles are the *extensor carpi radialis lonugus*, the *extensor carpi radialis brevis* and the *extensor carpi ulnaris*, together with some deep muscles.

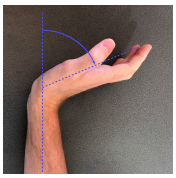


Figure 3.3: Wrist extension.

Wrist Flexion
The wrist flexion is the movement of the hand palm, towards the inner arm. The involved muscles are *flexor carpi radialis*, *palmaris longus* and *flexor carpi ulnaris*, as well as *flexor digitorum superficialis* and *profundus*.

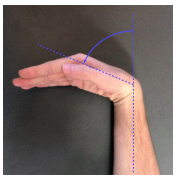


Figure 3.4: Wrist flexion.

Wrist Radial Deviation
The hand is moved up, following the thumb direction, in order to perform the radial deviation. Muscles involved are *abductor pollicis longus*, *flexor carpi radialis*, *extensor carpi radialis longus* and *brevis*.

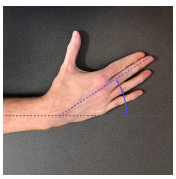


Figure 3.5: Radial deviation.

Wrist Ulnar Deviation
The hand is moved down, in the little finger direction, to perform ulnar deviation. The useful muscles are *extensor carpi ulnaris* and *flexor carpi ulanis*.

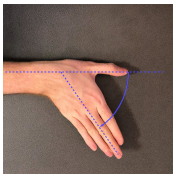


Figure 3.6: Ulnar deviation.

Hand Grasp
Hand grasp is the action of closing all the finger towards the hand palm. *Flexor digitorum* and *palmaris longus* are the most used, together with many intrinsic muscles of the hand.

Figure 3.7: Grasp.

Idle state
The idle state is performed trying to relax all the above described muscles, keeping the hand in a steady position, without contrasting gravity.

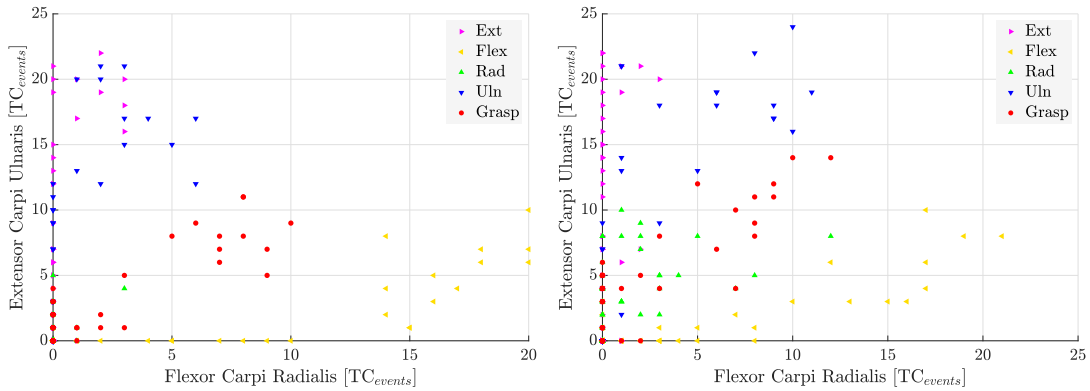
Figure 3.8: Rest position.

3.2 Electrodes Placement

The electrode placement is a critical aspect of this thesis. In fact, even using relatively small electrodes, like the H124SG, the muscles are to close one to the other

to not have *crosstalk* between them. Moreover, some muscles are so thin to be even smaller than the 24 mm electrodes. Thus, an initial study on the optimal placement has to be done, as well as continuously being wise when a new subject is considered, to adapt to the different morphology of the forearm.

An initial study considered the number of channels to be used, according to the five active movements determined in section 3.1. A first try with only two channels has involved six different positions around the forearm, all of them near the elbow, following a circular path. Signals have been recorded with an early version of the Apollo2 firmware and then plotted on a MATLAB® graph.
Data recorded individually from each of the six position brought to a configuration that seemed acceptable, with the electrodes placed on the *flexor carpi radialis*, for what concerns the medial section of the forearm, and on the *extensor carpi ulnaris* on the distal section. Two different trials have been made, one setting the threshold to 1.80 V and the other trying to recognize more activations using a threshold of 1.75 V. Both of them results in a bad overlap of different classes, due more to noise contribution than to effective muscular activation, as shown in Figure 3.9.

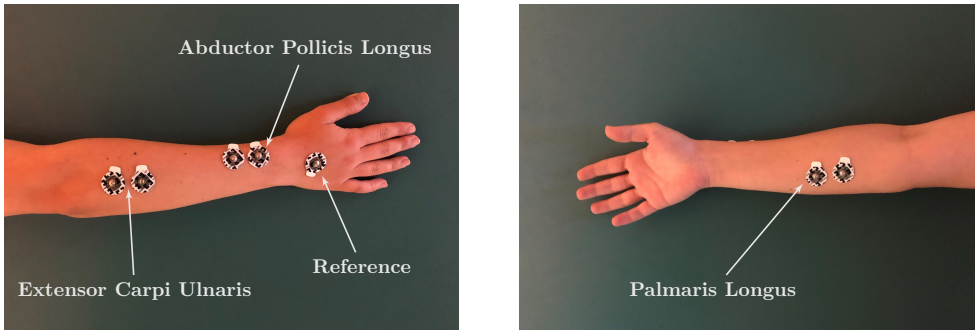


(a) 1.80 V threshold (b) 1.75 V threshold

Figure 3.9: Initial trial with two channels.

Watching the graphical results, it seems the data are compressed on the 2D plane and nothing could divide them appropriately. The choice to add a third channel has then been made, considering that the power and computational cost of the overall system would not have been affected so much.

Comparing the results obtained from the initial study with a previous thesis [26], it seemed reasonable to slightly modify the placement of the already existing electrodes pairs and to insert the third couple in a suitable way to take care of the radial deviation movement, that was almost neglected by the first configuration.



(a) Forearm back (b) Forearm front

Figure 3.10: Electrode placement on both sides of the forearm.

Tests realized on this set up resulted in a good discernment between classes and the configuration has been made definitive. Final placement, shown in Figure 3.10 is the following:

- First couple of electrodes is placed on the *palmaris longus*, which originates from the *medial epicondyle* and has its insertion on the proximal *superficial palmar fascia*. The electrodes should be positioned on the lower area of the *palmaris*, near to the *flexor carpi ulnaris*, in order to take into account some good cross action from it. The main contribution is to the *Flexion* movement, but some useful value are recorded also during *Ulnar Deviation* and *Grasp*.

- Second pair is placed on the superficial area of the *abductor pollicis longus*, which has its origin in the *radioulnar interosseous membrane* and terminates on the radial dorsal base of the *thumb metacarpal*. Placement of the electrodes should be made near the wrist, where the muscle become superficial. These electrodes are mainly used in *Radial Deviation*, but have nice effects on *Extension* and *Grasp*.

- The third and last pair has to be placed on the *extensor carpi ulnaris*, which has the origin in the lateral border of the *distal humerus* and the insertion on the dorsal base of the *small metacarpal*. Main effects are obviously on the *Extension* movement, but they are also necessary for *Ulnar Deviation*.

- Last, the reference electrode has been placed on the hand back, near the wrist, in a bony electrical neutral area, in a way that does not prevent correct execution of the movements.

As it is possible to see, the channels are used combined with others to obtain a possible movement, allowing to reduce the number of electrodes used, while keeping high the number of movements that are classified.

3.3 Acquisition Protocol

After the initial test phase, an *in vivo* experimentation has been launched, in order to collect from different people enough data for the classifier to be trained and tested. In this part 25 people have been involved, 16 males and 9 females (with an age between 23 and 37 years old). They have all been exhaustively informed about the experiment and possible risks. They received the informed consent regarding the study and they accepted to participate signing the form, redacted according to the local bio-ethical committee regulations.

Volunteers have been divided in two different groups: 20 people have been destined to the classifier training, while the remaining 5 have been enrolled in the online testing phase. The two sessions have been performed subsequently, in different days, without in any way replicating environment conditions, in order to guarantee that training data and testing ones were not dependent one to the other. The two groups protocols differ for few procedures, but an initial calibration phase is in common. Each subject has to sit in a comfortable way, in a way that the right arm could stay above the table of interest, in an horizontal position, and that the hand is free to move. After the explanation about the study, the subject has the electrodes placed on the forearm. This placement is really critical, as written in Sec. 3.2, because of the narrowness of the muscles and their small cross section. A bad positioning could bring to a very bad quality of the acquired data, with an accuracy degradation up to 30% [27, 28]; to prevent this issue, a preliminary analysis is performed, acquiring sEMG signals for a small time period and visualizing how the results would be with that electrodes configuration.

- The volunteer execute one movement at a time, starting from the rest position and keeping the gestures once reached. The acquisition lasts 6.5 s, during which 50 values are acquired over the 130 ms time window.
- The hand returns in the rest position and a pause of 5 s is performed to avoid muscular fatigue.
- A different movement is acquired on the available 6.5 s period. The routine is repeated until all the five active movements are performed.
- Obtained data are then saved to a file and moved to MATLAB® environment. Here are 3D plotted and some visual observation takes place.
- If necessary, some electrodes could be placed in a slightly different way, to enhance classifier performances.

The above protocol could be repeated in case of drastic conditions.
Once the calibration has finished, the real acquisition could begin.

3.3.1 Training protocol

During training each individual had to perform only the five active movements, always in the same order. The acquisition period is doubled with respect to the calibration phase, to ensure enough data is collected and to allow the subject to perform the movement in a clean way, without being too fast so no more noise is introduced. The acquisition is made with the Apollo2, using the debug mode of the board to stop and restart the flow when necessary. No skin treatment is performed at the begin, to ensure robustness of the system even with not optimal condition on the forearm.

1. The supervisor remembers to the subject which is the movement to be performed.
2. A Start command is given to the volunteer and the debug is told to continue the program.
3. The person reaches the desired gesture, then comes back to the rest position and executes the movement again.
4. When the 13 s period is finished and all the 100 sets of data are acquired, a Stop command is given to the subject.
5. A rest of 5 s is observed. If there are still movements to be execute, the flow goes back to point 1.
6. If all the movements are done, a pause of 1 min is performed, letting the person to lay the arm on the table.
7. Data is saved on the computer. Flow restarts from 1, unless five session have already been finished.

During the acquisition it was not possible to display any output, since the classifier is not trained yet, so the volunteers had to perform with their most capabilities. The electrodes are then removed from the forearm.

3.3.2 Testing protocol

After the classifier is trained, the remaining five people are called and the testing phase begins. The acquisition period is set to 5.2 s (40 windows of 130 ms), in order to keep the execution low and not let the subject arm become tired, as well as to have 1000 acquisition packet for each subject, at the end. The debug mode is used like in training protocol. The output is not visible even if the classifier is now trained, in order to do not let the volunteers adapt their gestures, in case of mistake. All the six movements, including *Idle*, are requested now, to value also the idle performance.

1. The supervisor remembers to the subject which is the movement to be performed.
2. A Start command is given to the volunteer.
3. As soon as the person reaches the desired gesture, the debug is told to continue.
4. The gesture has to be kept steady for the 5.2 s period.
5. When the 40 sets of data are acquired, a Stop command is given to the subject.
6. A rest of 5 s is observed. If there are still movements to be execute, the flow goes back to point 1.
7. If all the movements are done, a pause of 1 min is performed, letting the person to lay the arm on the table.
8. Data is saved on the computer. Flow restarts from 1, unless five session have already been finished.

The debug window immediately returns the accuracy value of the session and both the confusion matrix and the statistics related to that subject.

Chapter 4

Classification Algorithms

The data acquired during training acquisition phase have to be processed in order to be useful for the classification process. Once they are saved in the computer, they have been uploaded to MATLAB® to be elaborated. The output label is added, depending on the movement that was performed, to tell the classifier which the desired value is. Then, for each person, data from the five consecutive sessions are added together and plotted on a 3D graph. Here a custom routine creates the idle data, transforming all the values below a predefined threshold, according to 4.1

$$Idle = \sqrt{x_i^2 + y_i^2 + z_i^2} \leq N \tag{4.1}$$

where N is the user-defined value and x_i, y_i and z_i are the three TC values that constitute an acquisition set.
Once obtained the six-classes dataset, it is possible to delete some points, in case they appear to be generated from a environment problem, in order to not let them influence the classifier. If the obtained results seems good, the dataset is saved and could now be used as training input for the classifier.

The aim of the system is to keep low computational effort and power consumption, so a fully-connected Neural Network (NN) has been chosen as classifier architecture. This NN is the simplest implementation possible, regarding small datasets like the one used in this work. Moreover, its forward propagation prediction is made of just matrices multiplication, making it suitable for the CMSIS-DSP usage, as explained in Sec. 2.2.1. The training of the network has been performed offline on a computer, while the prediction has been done online on the Apollo2 MCU.

4.1 Offline Training

The training of the NN classifier has been made offline on the MATLAB® environment. This implementation allows a relatively fast and repeatable training routine and provides the usual powerful tools of the software. The back-propagation function needed to optimize the NN cost has been studied in [29].

Initially, some considerations and trials have been made, looking for the best implementation possible. With fewer data than the final ones, which are too many and it would have taken too much time, a preliminary study on the NN has been performed. A special routine, taking into account all network possibilities, has been written. The program had trained and tested 63 different NN, each one considering 10 different regularization values, starting from 1 hidden layer made of only 10 nodes and ending with 3 hidden layers with 30 nodes each. The top five results are shown in Table 4.1.

Table 4.1: NN preliminary study (250 iterations)

Layers #	Nodes #	Regularize (λ)	Val. Error	Accuracy (%)	Training time (s)
2	26	0.300	0.321	94.98	63.104
2	27	0.300	0.330	94.70	63.666
3	28	0.001	0.332	94.65	94.054
2	23	0.001	0.335	94.93	59.189
2	25	0.010	0.336	94.61	62.386

Since this characterization, the NN structure (shown in Figure 4.1) has always been defined according to the best result obtained, with 2 hidden layer and 26 nodes each. However, the regularization parameter λ has been always changed, trying every time the whole vector of ten values (i.e., ranging from 0 to 10, with a logarithmic progression), in order to perform a subtle adjustment and improve the overall performance.

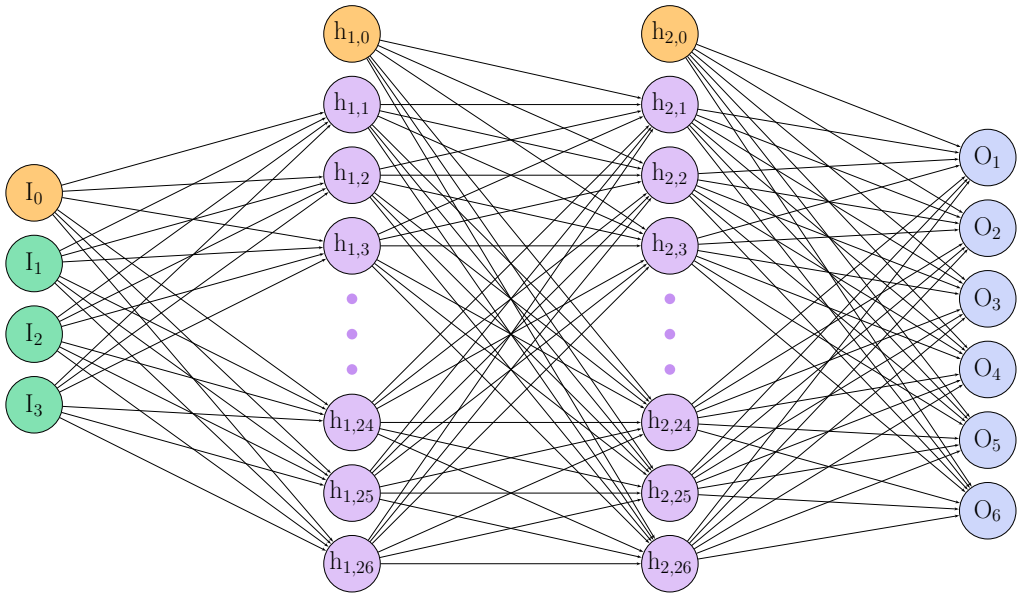


Figure 4.1: Neural Network structure.

4.1.1 Matlab Firmware

A MATLAB® script is used to train the NN. In the top of the code, global variables specify how the NN is structured. The code flow is the following:

- The input datasets (e.g. the one in Figure 4.2) are loaded separately, granting that no correlation occurs between them. Data matrices are divided into train set and validation set. They are then shuffled to enhance the performance of the back propagation. No standardization has been made, in fact data are already evenly distributed and the range of the three features is almost the same (0–30 $\mathrm{TC}_{\mathrm{events}}$).

- A regularization λ is picked from the vector and the training starts. The external routine *NNtrain* performs as many iterations as configured from the user. Of course, the more you iterate, the more the NN will be precise, but after a certain value improvements are no more significant (final implementation has been iterated 1500 times).

- The computational error of the network, computed on the validation test, is saved in a vector. Preceding task is repeated till the end of the λ vector.

- Once terminated, best result is picked among the ten trained network, according to the lower computational error.

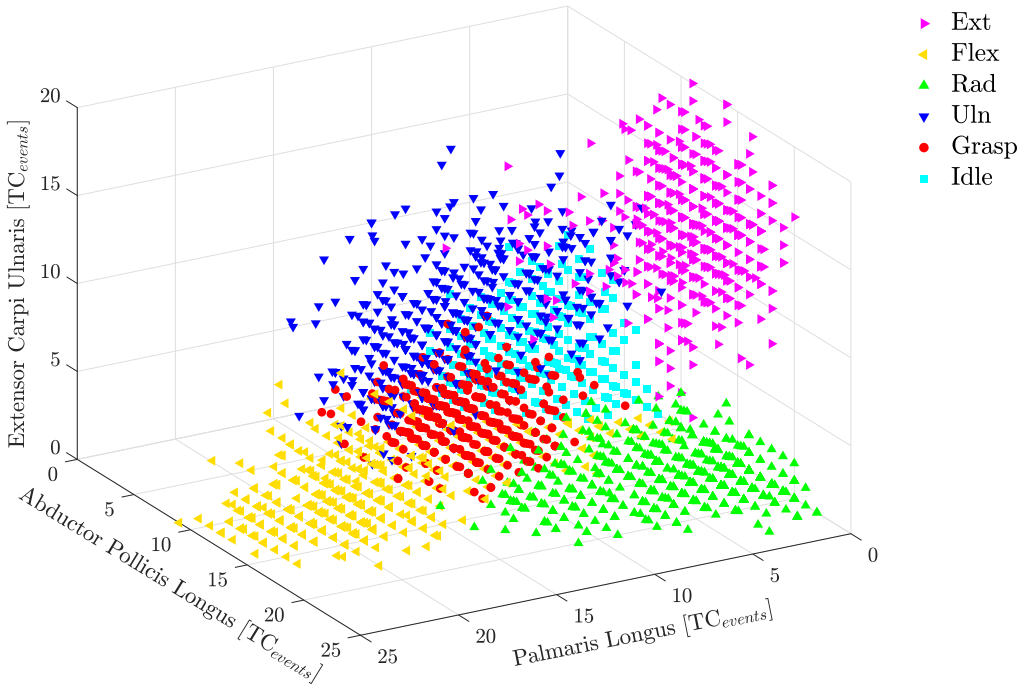


Figure 4.2: Dataset from one subject.

- Theta parameters obtained with corresponding λ are then picked and saved in specific variables.
- The user is asked whether he wants to save theta values, statistics or other parameters of interest.

The NN that has been used at the end of this work, is reported in Table 4.2.

Table 4.2: NN final implementation (1500 iterations)

Layers #	Nodes #	Regularize (λ)	Val. Error	Accuracy (%)	Training time (s)
2	26	0.010	0.630	92.31	2110

4.2 Online Prediction

The obtained network is then transferred to the Apollo2 MCU, copying the related theta parameters to a configuration file of the Apollo firmware. This values are then put in the specific matrices, as explain in 2.2.1.
Every time a 130 ms time window ends, the timer interrupt arises and the board exits the deep sleep mode. The service routine takes the TC values counted by the GPIO interrupts transform them into the NN inputs. Two routine are then involved, one after the other, to define the output class:

- the *actuate* routine is in charge of taking the NN and forward propagate them to the output, using the theta provided by the training phase. Practically, for each layer, the values of the preceding one are multiplied by the correspondent parameters and are then normalized using the sigmoid activation (sigmoid is a function limited between 0 and 1) in order to not let the network diverge. Every layer a node is added, the so called *bias* node, needed to guarantee always a known value in the network. Once the output layer is reached, the predicted class is defined as the class which has the greater activation value at the end. The so obtained class is passed as a parameter to the second routine.

- The *checkState* routine takes the predicted class as input and it is in charge of controlling if the class is within the possible ones and if it is a long or a short change with respect to the previous one, and finally sends the value out on the UART to the BT module, if needed. First a delay slot is implemented: the class is taken into account only if it is equal to the previous one and different w.r.t. the one before. If it does not respect these constraints it means either it has not changed its value in the last two time windows or the value is changed since the present window only. Single window glitch are neglected in order to enhance the robustness of the system. Then the class is passed in a switch statement and if it matches one of the options the output value is displayed on the LEDs (if enabled) and the related character is written into the transmitting string. Last, if the TANK mode is enabled, a string containing the class and the terminator character ';' is sent towards the Zumo robot via the UART interface.

Chapter 5

Experimental Results

At the end of the work, some measurement of interest have been taken. In following sections, accuracy of the classifier, latency of the system and power consumption of the boards are analyzed in details. An overall view of the results, including a comparison with others SoA works, is shown in Table 5.1.

Table 5.1: Comparison with existent EMG-based classifier

Work	Features	Classifier	Accuracy (%)	Channels #	Gestures #	Power [mW]
[19]	multiple	RBF[1]	66	8	6	n.d.
[30]	n.d.	HD[2]	90-96	64	5	n.d.
[24]	ATC	SVM[3]	93	3	5	20.2
[18]	DWT[4]	SVM	94	4	5	5.1
This	ATC	NN	96	3	6	2.9

[1]Radial Basis Functions, [2]High Dimensional classifier, [3]Support Vector Machine
[4]Discrete Wavelet Transform

5.1 Classifier Accuracy

The accuracy of the classifier has been measured directly on the Apollo2 MCU, implementing specific routines able to take count of the prediction and to understand if it has been correct or not. Saved values have then been printed on the results files, being accessible for later discussions. Table 5.2 shows the confusion matrix of the whole testing phase. The predicted values of the five people that

have taken part in the *in vivo* experiment are added together. The value on the main diagonal of the matrix should be 1000, if all the movements were performed correctly. Unfortunately it is possible to see some consistent variations from the theoretical value.

Table 5.2: Validation Confusion Matrix

		Predicted					
		Ext	Flex	Rad	Uln	Grasp	Idle
Actual	Ext	992	0	4	4	0	0
	Flex	0	883	67	0	42	8
	Rad	0	29	913	12	40	6
	Uln	180	6	0	749	27	38
	Grasp	0	60	11	35	804	90
	Idle	0	0	0	0	0	1000

The statistics obtained from this confusion matrix are reported in Table 5.3. The average accuracy is 96.34%, reaching a comparable, if not higher, value with respect of other SoA works. It is possible to observe that *Ulnar Deviation* and *Grasp* are the worst performing movements. Those gestures, due to the few electrodes used, have an overlap in features space and are difficult to classify. An accurate placement of the electrodes during the initial calibration can solve part of those issues, keeping the accuracy performances over a good value of confidence.

Table 5.3: Statistical Results

	Accuracy(%)	Precision(%)	Recall(%)	F_1-score(%)
Ext	96.87	84.64	99.20	91.34
Flex	96.47	90.29	88.30	89.28
Rad	97.18	91.76	91.30	91.53
Uln	94.97	93.63	74.90	83.22
Grasp	94.92	88.06	88.40	84.06
Idle	97.63	87.57	100.00	93.37
Avg	**96.34**	**89.32**	**89.02**	**88.80**

5.2 System Latency

Latency has been measured on the Apollo2 board, implementing a timer with a frequency of 6 MHz. The timer is started when the end of the time window occurs. It continues running through all the computations and it is stopped after the output class is defined. In this way it takes into account the whole computational phase, without neglecting anything. The average value obtained from the measurement is 8.5 ms. The MCU is then active with a duty cycle of 6.5%, obtained considering as full period the acquisition time window ($8.5/130 = 0.065$).

Though, to calculate the overall latency, from when the user performs the desired movement to when the actuator (in this case the tank) executes it, it is necessary to add together two time windows. In fact, with the implementation of the delay slot during prediction, the first acquisition period is always neglected, being not enough to determine the correctness of the inputs. Fortunately the second time window starts immediately, without waiting for the calculations to be done, resulting in a global latency of 268.5 ms (130 + 130 + 8.5 at the end).This value is still suitable for real-time applications, as it is lower than the usual limit of 300 ms.
The throughput of the BT module does not significantly influence the latency of the system, because the ATC computing is far slower than 1500 kbit s^{-1}.

5.3 Power Consumption

Power consumption analysis has been performed using an external equipment. Regarding the MCU consumption, the measurement has been made removing a small jumper on the board, usually inserted between board power and MCU `VDD`, and applying an instrument on its extremities.

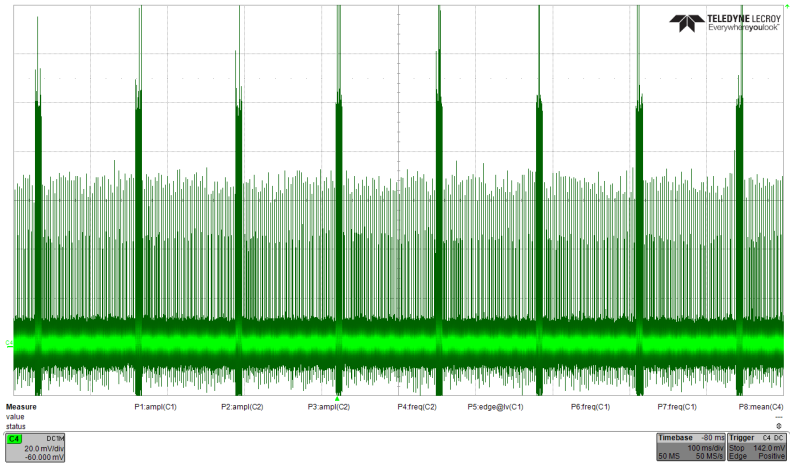


Figure 5.1: Current absorption, measured with the active current probe.

At the beginning an active current probe has been chosen to perform the measure. The probe is connectible to the oscilloscope and provides on the output a voltage ten times greater than the related current one (1 mA -> 10 mV). The obtained measurements are shown in Figure 5.1. Even if the generic behavior is represented, the displayed waveform is not satisfying, having a lot of noise and sometimes even a negative average value.

Considering this issue another instrument has been taken into account and the final measurements have been made using an INA126P. The measure is made taking the loss of voltage on a 10 Ω resistance and configuring the INA to amplify by a value of 85.1. As it is possible to see in Figure 5.2, current consumption follows exactly the behavior of the µP, being low during the acquisition window and having a higher step during the 8.5 ms calculation.
Obtained deep sleep power consumption is 0.70 mW, while in active mode 2.05 mW are measured. Considering the duty cycle of 6.5% an average power consumption of 0.80 mW is obtained.

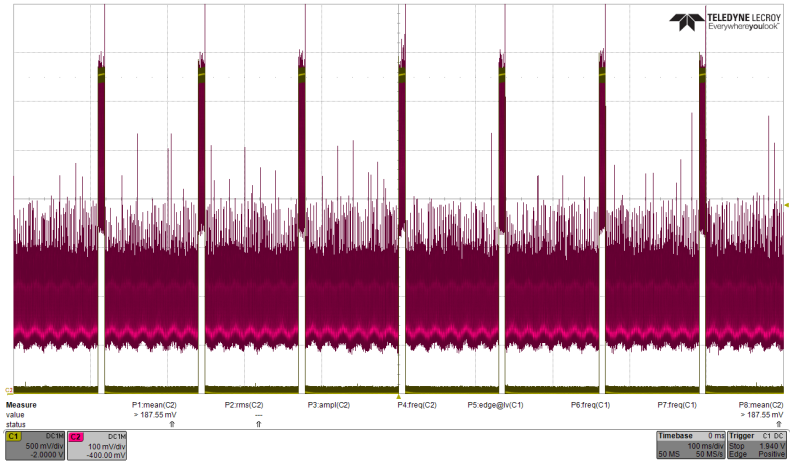


Figure 5.2: Current absorption, measured with the INA126P.

Current absorption of the acquisition channels has then been analyzed. During a previous work [14] a power consumption of 0.635 mW was measured, but since in this version of the PCBs some more components have been added, the measurement has been done again. Obtained value is 0.701 mW for each channel.
Adding together the different contributions, a total power consumption of 2.9 mW is obtained, making this system suitable for wearable applications and competitive with other SoA works (e.g. [18]).

Chapter 6

Conclusions

This thesis proposed a low power system able to recognize hand gestures, by means of an embedded classifier, acquiring surface electromyographic signals from the forearm. The Average Threshold Crossing (ATC) event-driven technique has been used to preprocess data. The final implementation results suitable for wearable real-time applications, thanks to low latency and power consumption.

Three sEMG acquisition channels are able to acquire muscular activity from the forearm, giving as output a filtered sEMG signal, to check the health of the system, and a TC *quasi-digital* signal, sent to an Apollo2 microcontroller to process the data. The signal elaboration is performed with negligible delay and with a power consumption of only 0.701 mW for each channel.
The TC events sent to the microcontroller are counted with a simple interrupt routine and then averaged over a time window of 130 ms. The obtained values are initially saved on the computer and used as training data for a fully-connected Neural Network (NN). In a second phase, obtained parameters are then used to classify data in real-time directly on the board.
The predicted class is then sent to a small tank, via a Bluetooth 3.0 module, to show the effective correctness of the onboard classifier.

Final solution implements a NN with 3 input features, 2 hidden layer, with 26 nodes each, and 6 output classes. An *in vivo* validation has been made, with 25 people involved, in order to test the accuracy of the classifier, obtaining a value of 96.34%. Power consumption of the overall system is measured of 2.9 mV, using an external instrumentation amplifier. Latency of the classifier, measured with a firmware routine, results in 8.5 ms.
These statistics make this work comparable with other SoA works, or even more competitive.

6.1 Future Works

Future improvements could take different directions: an optimization of the µP firmware is possible, in order to reduce power consumption and enhance computational capabilities, as well as a system rethinking, addressing different problems or adding new features.

Regarding the firmware upgrades, an opportunity to reduce computational effort on the microcontroller is to implement approximate computing methods. This would bring to a faster computation and possibly to a lower power consumption too. It would concern both the neural network sigmoid activation, which could be rectified to avoid the exponential, and the type of data used, evaluating if a different data type would be better for this application.
Another improvement could be the introduction of a reinforcement learning online, in particular for what concerns the initial manual calibration of the electrodes placement. A software routine could train the last layer of the network to improve adaptability of the system to each individual.

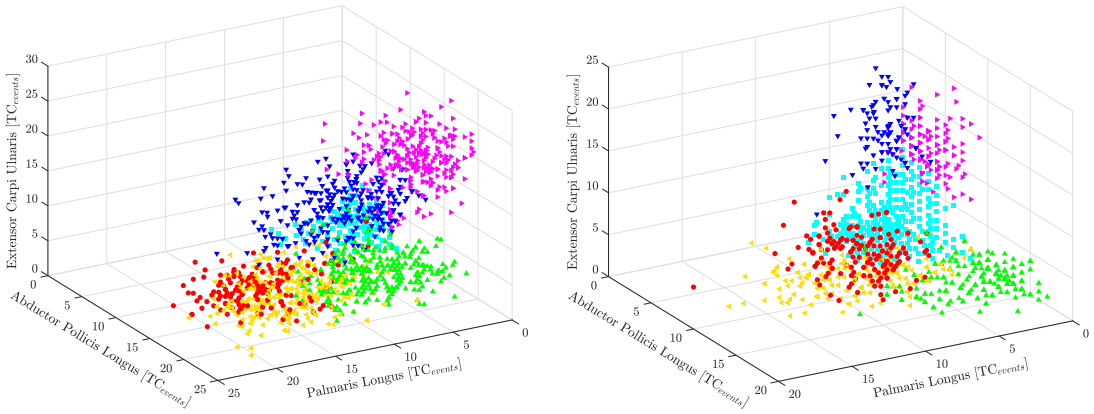
On the hardware side a lot of upgrades can be made, depending on the desired application. Changing the number of the electrodes as well as their shape, could bring to a more powerful system, with possibly a classifier able to distinguish among more and more different gestures, even more similar to each other.

Appendix A

3D Datasets Representation

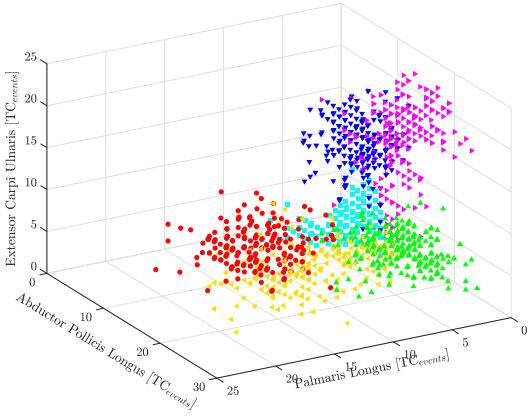
In this first appendix, the graphical representations of the twenty datasets used for the training phase are displayed. Each figure contains 2500 points, obtained by one single person in a session. Each color corresponds to a different class and the three axis are equivalent to the three acquisition channels used in this work. The idle class has already been generated and it is displayed in cyan.
From the graphics it is possible to notice that from one person to another there could be even big differences in the class displacements. Though, the classifier has been able to distinguish between them obtaining good results as shown in 5.1.

Figure A.1: Legend of the movements.

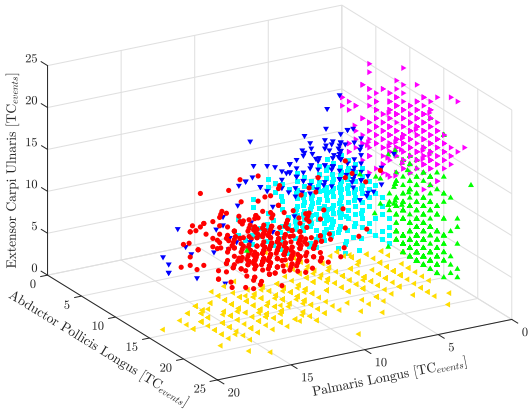


(a) Subject 1 (b) Subject 2

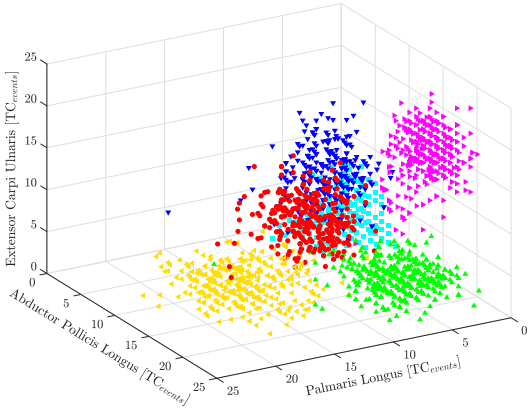
Figure A.2: Datasets of the 20 people involved in the training phase (1 of 4).



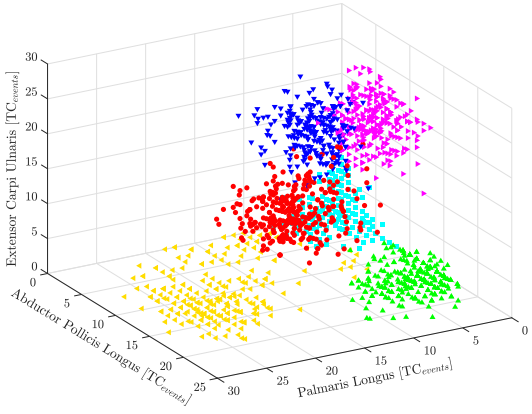
(c) Subject 3



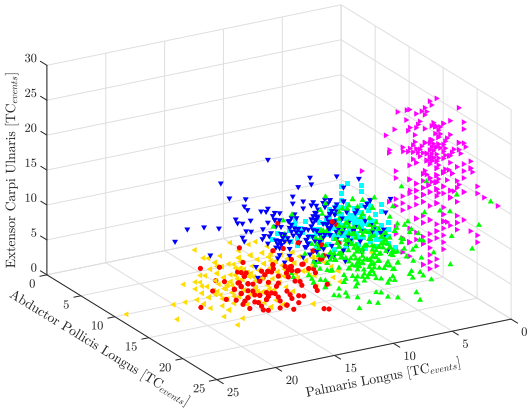
(d) Subject 4



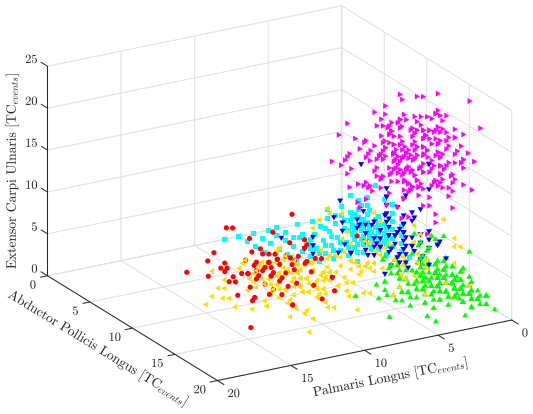
(e) Subject 5



(f) Subject 6

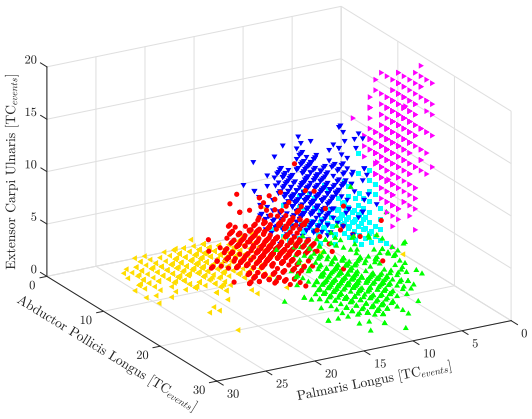


(g) Subject 7

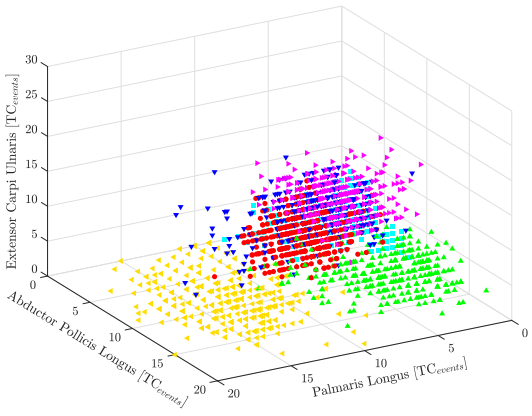


(h) Subject 8

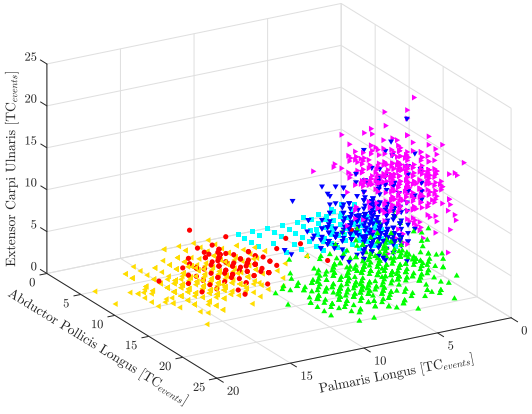
Figure A.2: Datasets of the 20 people involved in the training phase (2 of 4).



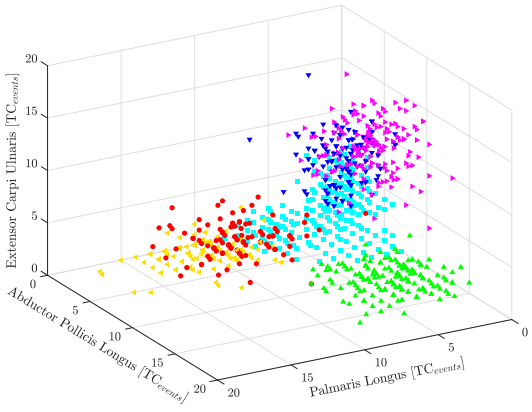
(i) Subject 9



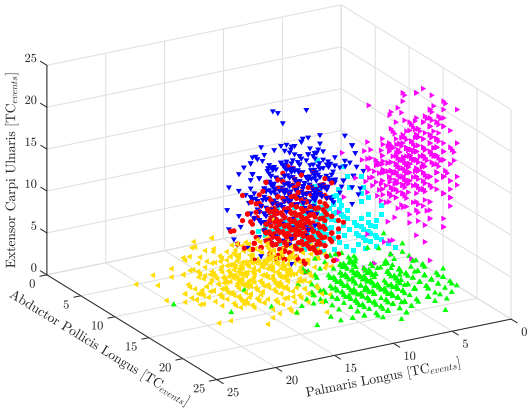
(j) Subject 10



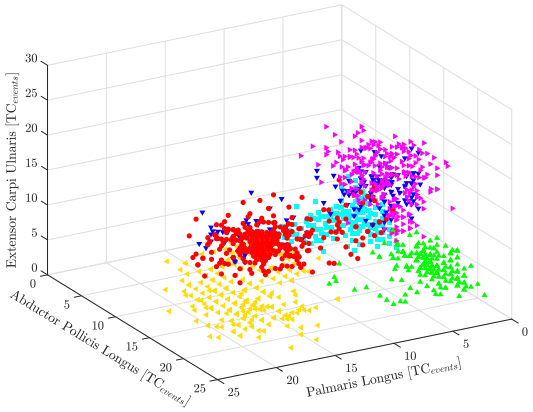
(k) Subject 11



(l) Subject 12

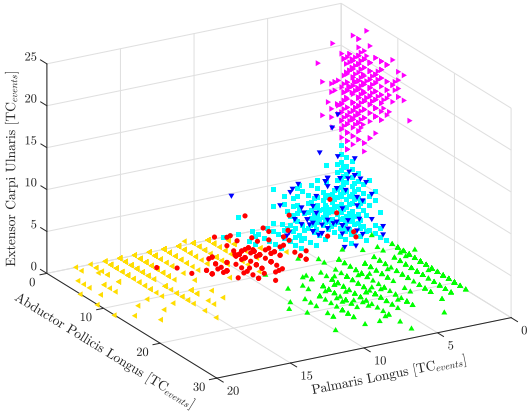


(m) Subject 13

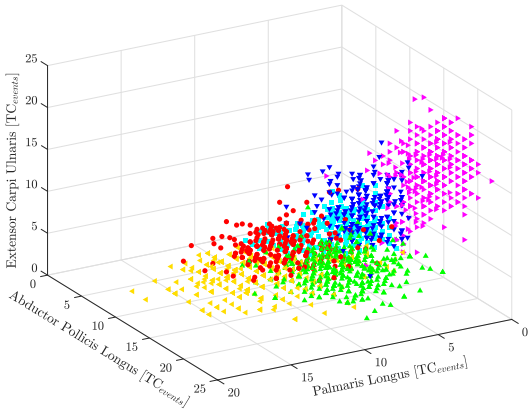


(n) Subject 14

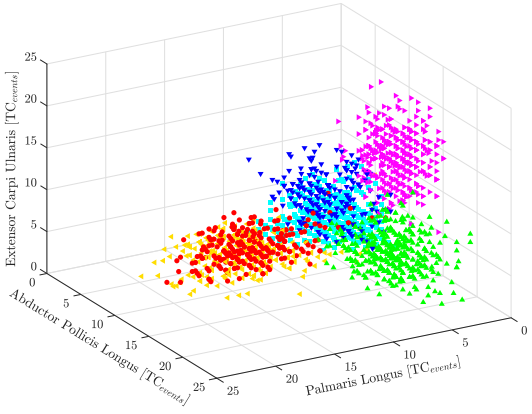
Figure A.2: Datasets of the 20 people involved in the training phase (3 of 4).



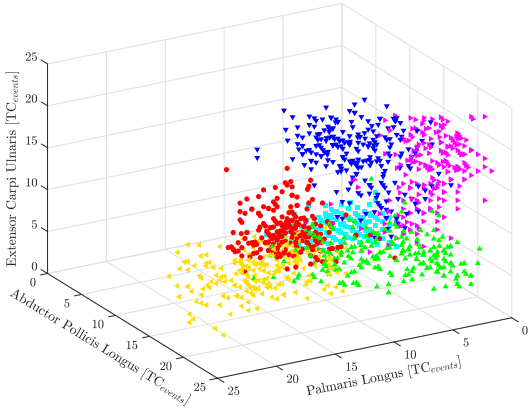
(o) Subject 15



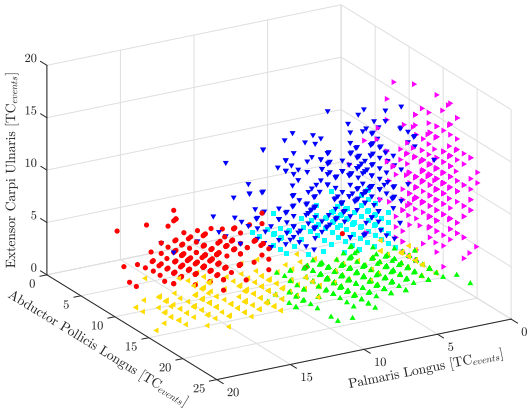
(p) Subject 16



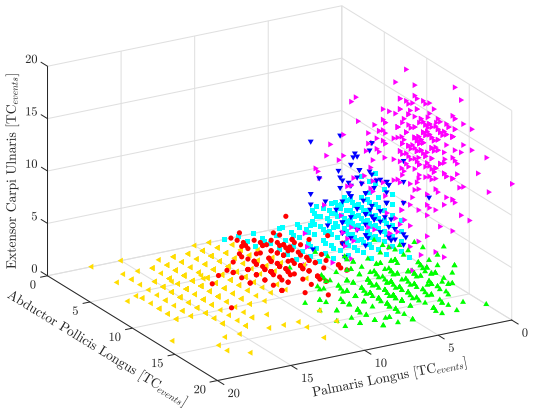
(q) Subject 17



(r) Subject 18



(s) Subject 19



(t) Subject 20

Figure A.2: Datasets of the 20 people involved in the training phase (4 of 4).

Bibliography

[1] F. Conti, D. Palossi, R. Andri, M. Magno, and L. Benini, "Accelerated visual context classification on a low-power smartwatch," *IEEE Transactions on Human-Machine Systems*, vol. 47, no. 1, pp. 19–30, 2016.

[2] S. Mitra and T. Acharya, "Gesture recognition: A survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 3, pp. 311–324, 2007.

[3] J. McIntosh, C. McNeill, M. Fraser, F. Kerber, M. Löchtefeld, and A. Krüger, "Empress: Practical hand gesture classification with wrist-mounted emg and pressure sensing," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pp. 2332–2342, ACM, 2016.

[4] M. Crepaldi, M. Paleari, A. Bonanno, A. Sanginario, P. Ariano, D. H. Tran, and D. Demarchi, "A quasi-digital radio system for muscle force transmission based on event-driven IR-UWB," in *2012 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pp. 116–119, IEEE, 2012.

[5] S. Sapienza, M. Crepaldi, P. Motto Ros, A. Bonanno, and D. Demarchi, "On integration and validation of a very low complexity ATC UWB system for muscle force transmission," *IEEE transactions on biomedical circuits and systems*, vol. 10, no. 2, pp. 497–506, 2015.

[6] https://multiple-sclerosis-research.org/2015/03/mitochondria-again-but-this-time-in-muscle/, 2019.

[7] H. Ishikawa, "Fine structure of skeletal muscle," *Cell and muscle motility*, vol. 4, pp. 1–84, 02 1983.

[8] Martini, Nath, and Bartholomew, *Foundamental of Anatomy & Physiology*, ch. 10. 10 ed., 2015. Chapter name: Muscle Tissue.

[9] Muay Thai Scholar contributors, "The motor Unit." Retrieved: Feb, 18 2016. [Online]. Available: `http://www.muaythaischolar.com/wp-content/uploads/2016/02/Motor-Unit-Anatomy.jpg`.

[10] C. J. De Luca, A. Adam, R. Wotiz, L. D. Gilmore, and S. H. Nawab, "Decomposition of surface EMG signals," *J Neurophysiol*, no. 96, pp. 1646–1657, 2006. doi: 10.1152/jn.00009.2006.

[11] E. Marieb and K. Hoehn, *Human Anatomy and Physiology*. 10 ed., 2016.

[12] A. Shahshahani, P. Motto Ros, A. Bonanno, M. Crepaldi, M. Martina, D. Demarchi, and G. Masera, "An all-digital spike-based ultra-low-power IR-UWB dynamic average threshold crossing scheme for muscle force wireless transmission," *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2015.

[13] P. Motto Ros, M. Paleari, N. Celadon, A. Sanginario, A. Bonanno, M. Crepaldi, P. Ariano, and D. Demarchi, "A wireless address-event representation system for ATC-based multi-channel force wireless transmission," in *5th IEEE International Workshop on Advances in Sensors and Interfaces IWASI*, pp. 51–56, IEEE, 2013.

[14] D. A. Fernandez Guzman, S. Sapienza, B. Sereni, and P. Motto Ros, "Very low power event-based surface EMG acquisition system with off-the-shelf components," in *2017 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pp. 1–4, IEEE, 2017.

[15] K. Momen, S. Krishnan, and T. Chau, "Real-time classification of forearm electromyographic signals corresponding to user-selected intentional movements for multifunction prosthesis control," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 15, no. 4, pp. 535–542, 2007.

[16] P. Shenoy, K. J. Miller, B. Crawford, and R. P. Rao, "Online electromyographic control of a robotic prosthesis," *IEEE transactions on biomedical engineering*, vol. 55, no. 3, pp. 1128–1135, 2008.

[17] M.-F. Lucas, A. Gaufriau, S. Pascual, C. Doncarli, and D. Farina, "Multi-channel surface emg classification using support vector machines and signal-based wavelet optimization," *Biomedical Signal Processing and Control*, vol. 3, no. 2, pp. 169–174, 2008.

[18] V. Kartsch, S. Benatti, M. Mancini, M. Magno, and L. Benini, "Smart wearable wristband for EMG based gesture recognition powered by solar energy harvester," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, IEEE, 2018.

[19] T. Phienthrakul, "Armband gesture recognition on electromyography signal for virtual control," in *2018 10th International Conference on Knowledge and Smart Technology (KST)*, pp. 149–153, IEEE, 2018.

[20] ARM, "Cortex-M series." https://www.arm.com/products/silicon-ip-cpu/cortex-m/cortex-m4, 2019.

[21] AmbiqMicro, "Apollo MCUs." https://ambiqmicro.com/mcu/, 2019.

[22] ARM, "CMSIS Library." https://developer.arm.com/tools-and-software/embedded/cmsis, 2019.

[23] Pololu Robotics & Electronics, "Zumo robot for Arduino." https://www.pololu.com/product/2510, 2019.

[24] S. Sapienza, P. Motto Ros, D. A. Fernandez Guzman, F. Rossi, R. Terracciano, E. Cordedda, and D. Demarchi, "On-line event-driven hand gesture recognition based on surface electromyographic signals," in *2018 IEEE International*

Symposium on Circuits and Systems (ISCAS), pp. 1–5, IEEE, 2018.

[25] E. Hand, "The electronic text book of hand surgery." http://www.eatonhand.com/mus/mus126.htm, 2019.

[26] E. Cordedda, "Riconoscimento real time di gesti applicato a segnali EMG superficiali basato su ATC."

[27] F. Palermo, M. Cognolato, A. Gijsberts, H. Müller, B. Caputo, and M. Atzori, "Repeatability of grasp recognition for robotic hand prosthesis control based on sEMG data," in *2017 International Conference on Rehabilitation Robotics (ICORR)*, pp. 1154–1159, IEEE, 2017.

[28] S. Benatti, E. Farella, E. Gruppioni, and L. Benini, "Analysis of robust implementation of an EMG pattern recognition based control," in *BIOSIGNALS*, pp. 45–54, 2014.

[29] A. Ng, "Machine Learning *by Stanford University*." https://www.coursera.org/learn/machine-learning/home/welcome, 2018.

[30] A. Moin, A. Zhou, A. Rahimi, S. Benatti, A. Menon, S. Tamakloe, J. Ting, N. Yamamoto, Y. Khan, F. Burghardt, *et al.*, "An EMG gesture recognition system with flexible high-density sensors and brain-inspired high-dimensional classifier," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, IEEE, 2018.