Politecnico di Torino

POLITECNICO DI TORINO

DIPARTIMENTO DI INGEGNERIA MECCANICA E AEREOSPAZIALE (DIMEAS)

Master's Degree Course in Biomedical Engineering

Embedded Machine Learning for Hand Gesture Recognition: Development and Validation of an ATC-based Armband

Supervisors

Prof. Danilo DEMARCHI

Ph.D. Paolo MOTTO ROS

M.Sc. Fabio ROSSI

M.Sc. Andrea MONGARDI

Candidate

Federica PASQUALI

TORINO, December 2021

Abstract

Gesture recognition refers to the mathematical interpretation of human motions using a computing device. With the increasing use of technology, hand-gesture recognition has become an essential aspect of Human-Machine Interaction (HMI), allowing the machine to capture the user's intention and respond accordingly. It is a fundamental tool to enable novel interaction paradigms in numerous applications, such as robotics, prosthetic control, healthcare and sign language. Several technologies are currently available to detect gestures. Data acquisition systems based on surface ElectroMyoGraphic (sEMG) signals, collected from non-invasive electrodes on the skin of the area of interest, are extensively used, especially in biomedical research.

This thesis work aims to develop and validate a seven-channel sEMG armband for hand gesture recognition. The embedded low-power system is based on an event-driven approach focused on the Average Threshold Crossing (ATC) feature, which reduces the complexity of the classification algorithms by transmitting a lower amount of data, hence diminishing the power consumption. This parameter is obtained by averaging the number of sEMG Threshold Crossing (TC) events in a pre-defined time window, reflecting muscle activation.

A first analysis has been conducted on a previously acquired dataset with seven acquisition channels involving 14 people, each performing seven gestures over three sessions. Different machine learning techniques were taken into consideration to create a system able to recognize and classify hand gestures in real-time. Neural Network (NN), Random Forest (RF), Support Vector Machine (SVM) and K-Nearest Neighbour (KNN) has been tested and implemented embedded. Software implementation carried out in the MATLAB® environment and firmware deployment for the ARM Cortex microProcessor (µP) has been conducted to obtain low power consumption, matching the requirements of a wearable battery-powered device, and a latency below 300 ms, suitable for real-time applications.

A model of the armband has been designed and 3D printed. The proposed system comprises seven bipolar acquisition channels consisting of dry electrodes, each acquiring the sEMG signal and providing the extracted TC signal. The research group has developed a PCB with an Apollo 3 Blue MicroController Unit (MCU) with an ARM Cortex M4F µP onboard, performing signal conditioning and the computation of the ATC data, fed to the embedded machine learning algorithms for the classification, and containing a Bluetooth Low Energy (BLE) module for data transfer.

The validation phase entailed the creation of a new dataset involving 20 people, each executing nine movements, repeated twice, within three sessions. The aforementioned machine learning algorithms have been trained and deployed on the MCU. Among them, the NN has reached an average classification accuracy of 95.09 %, and a maximum system latency of 1.037 ms, that summed to the acquisition window (i.e., 130 ms) is way below the 300 ms required for the online application, making the system suitable for wearable real-time use.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction to the Muscular System

The *Muscular System* is a set of anatomical structures comprising muscle cells, contractile elements, and linked connective tissues responsible for the execution of body movements. Muscle contraction also accomplishes other essential functions in the body, such as posture, joint stability, transport of blood and nutrients and heat production.
The Central nervous system tightly controls the functionality of the muscles through electrical stimuli, called *Action potentials* (AP).
The human body is composed of about 600 different muscles, classified into three morphological types [1]:

1. **Skeletal muscle**: The skeletal muscles account for 40% of the body weight in an adult. They are the only voluntary muscles of the body and are controlled by the peripheral component of the *Central Nervous System* (CNS). They are applied to the skeleton's external surface, and their contraction generates the movement of joints and associated structures. Skeletal muscles fibres have numerous peripherally located nuclei with organized striations in the sarcoplasm due to the alternating bands of actin and myosin filaments, confering them a striated appearance [2].

2. **Smooth muscle**: Smooth or visceral muscle is present in the walls of hollow internal organs and blood vessels, but not in the heart. Their fibres do not present striations having a single nucleus and are arranged in parallel lines. The smooth tissues are involuntary and controlled by the medulla oblongata in the brain, which guides involuntary activity throughout the body.

3. **Cardiac muscle**: Cardiac muscles or *myocardia* are involuntary muscles that form the heart's walls. Their primary function consists in making the right atrium contract permitting to impel blood into circulation. Cardiac muscle cells are striated single-nucleated fibres, also called myogenic fibres, because they contract rhythmically without fatigue independently of any direct nervous stimulation. This arrangement enables the cardiac muscles to contract as a unit.

1.1.1 The Skeletal Muscle

This study aims to perform gesture recognition based on the feature extracted from the sEMG signal collected from the forearm muscles. A deep understanding of skeletal muscles' main features permits comprehension of the movements and the signal thoroughly.

Each skeletal muscle is an organ that consists of multiple integrated tissues, including skeletal muscle and nerve fibres, connective tissue, and blood and lymphatic vessels.

The muscle fibres are large-sized multinucleated cells bundled into fascicles and enclosed by three layers of fibrous connective tissue, the *Mysia*, which primary aim is to provide structure to the muscle as shown in Figure 1.1.

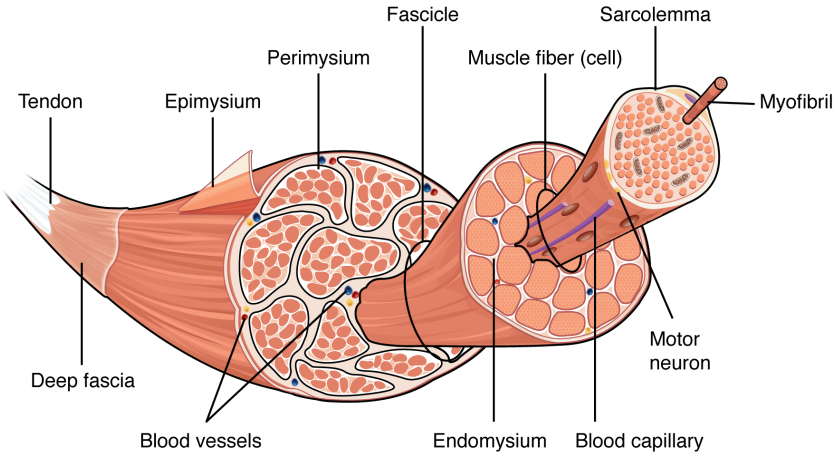


Figure 1.1: Skeletal Muscle structure [3].

A layer of irregular connective tissue called the *epimysium* envelopes individual muscles. It preserves their structural integrity while allowing the muscles to contract and move autonomously, separating them from other tissues in the area. The *perimysium* is a middle layer of connective tissue that encircles the bundles of muscular fibres called fascicles. The fascicular organization gives the ability to the nervous system to trigger a specified movement by recruiting a subset of muscle fibres within a muscle fascicle[3].

A thin layer of connective tissue called the *endomysium* encloses the individual muscle fibres of a fascicle. The endomysium plays a crucial role in transferring the force generated by the muscle fibres to the tendons. In order to produce a force on the bones, the skeletal muscles work with the tendons. At one end, the tendon's collagen tightly interlaces with the collagen of the three connective layers. At the other end, the tendon connects with the dense irregular connective tissue that coats the bone, called the *periosteum*. The contraction of the muscle fibres generates a tension, transmitted through the Mysia to the tendon and then to the periosteum, acting on the bones, resulting in the motion of the skeleton.

In other sites, the three connective layers can fuse with the *aponeurosis*, a broad flat sheet of dense fibrous connective tissue, or to the *fascia*, a band of connective tissue that supports the muscle and provides a pathway for the nerve, blood and lymphatic vessels.

Blood vessels richly supply the skeletal muscle for oxygen delivery, waste removal and nourishment. Moreover, the axon of somatic motor neurons is connected to each muscle fibre and provides the signals that enable the fibre to contract.

Skeletal muscle cells are multinucleated. This characteristic allows an abundant production of enzymes and proteins needed for the maintenance of muscle standard functionality. The nuclei are disposed below the plasma membrane called *sarcolemma*. The *sarcoplasmic reticulum* (SR) stores calcium ions (Ca_2^+) and then when the fibre is stimulated to contract, release and retrieve it.

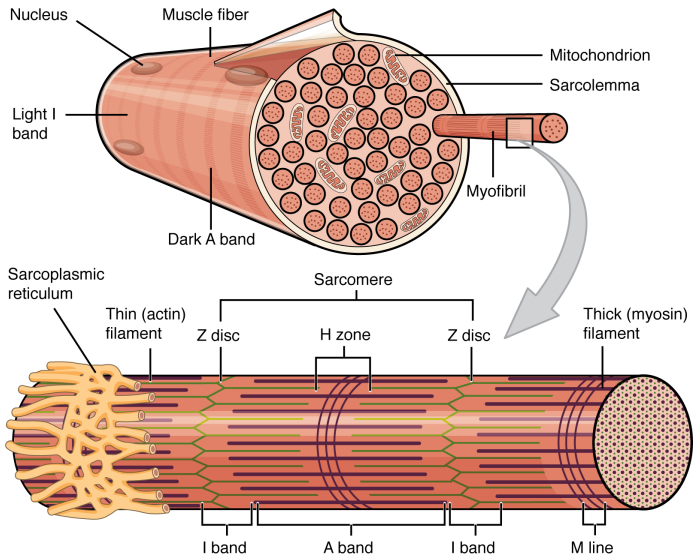


Figure 1.2: Muscle fibres [3].

The proteins inside the muscle fibres form structures called *myofibrils*, which contains sarcomeres connected in series. A highly ordered organisation of contractile, structural and regulatory proteins forms the *sarcomere*, the muscle fibre's smallest functional unit. The shortening of each sarcomere causes the contraction of the skeletal muscle fibres. The range of a myofibril enclosed between two cytoskeletal structures called **Z-lines** defines a sarcomere which presents an arrangement of thick and thin *myofilaments*. This configuration confers the striated appearance of the skeletal muscle. The thick filaments containing myosin compose the dark striated **A band**. This band covers the space between the centre of the sarcomere and the Z-lines. The protein myomesin anchor these thick filaments to the **M-line**, which indicates the middle of the sarcomere. The light regions are called **I bands** and comprise thin actin filaments secured to the Z-lines by a protein called α-actinin. The thin filaments partially overlap with regions of the thick filaments, extending toward the M-line into the A band. The **H zone** at the centre of the A band contains only thick filaments, as shown in Figure 1.2.

During a contraction, the myofilaments slide across each other, shortening the distance between the Z-lines. Due to this mechanism, the A band does not change in length while the I band and H zone shrinks. During contraction, filaments overlap increases consequencing a decrease of the region with no overlap.

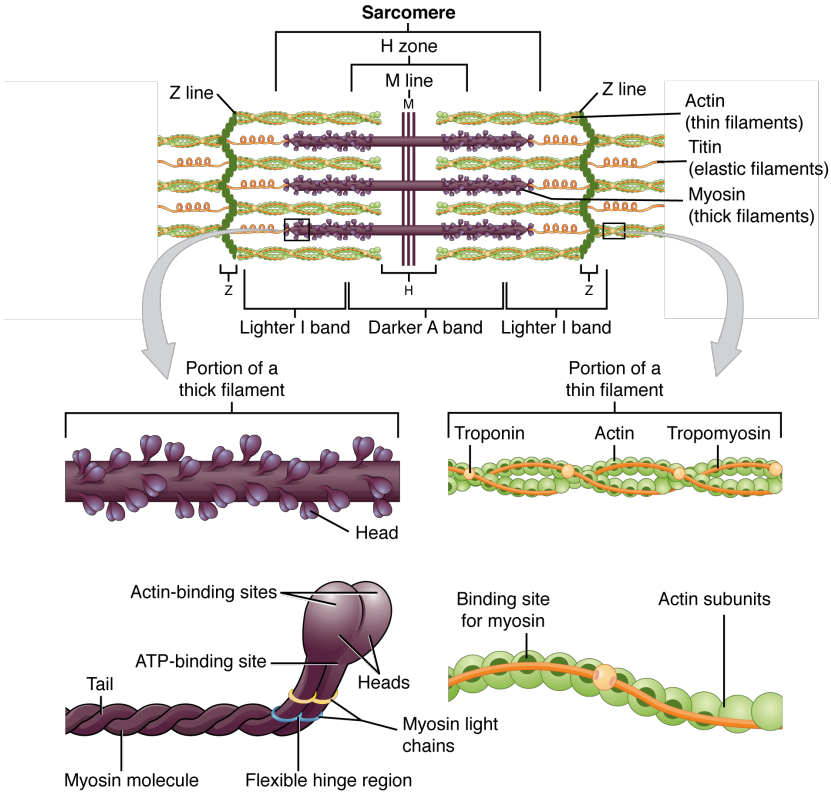


Figure 1.3: Sarcomere [3].

Two filiform actin chains (F-actin), composed of single actin proteins, form the thin filaments represented in Figure 1.3. The globular actin monomer (G-actin) within the filaments presents a binding site for myosin and is associated with two regulatory proteins, troponin and tropomyosin. These proteins control the exposure of the actin-binding sites to myosin. Myosin protein complexes form the thick filaments. These complexes comprise six different proteins, four light chains and two heavy chains molecules. The light chains present a regulatory role, while the heavy chains contain a globular head equipped with two binding sites, one for actin and another for ATP (*Adenosine TriPhosphate*). Other structural proteins have secondary roles in force production. Titin provides elasticity to the sarcomere and helps in the alignment of the thick filaments, while nebulin stabilizes the thin filaments and spans the length of the thick filaments.

The contraction of a skeletal muscle fibre takes place upon the reception of a signal from a motor neuron. This cause the pulling and sliding of the thin filaments past the thick ones in the sarcomere. The filament sliding process occurs only with the exposure of myosin-binding sites. The exposure process starts when the Ca_2^+ ions enter the sarcoplasm. This mechanism is known as *Excitation-Contraction Coupling*. Tropomyosin prevents actin from binding to myosin by twisting around the actin filaments and covering the binding sites. The troponin-tropomyosin complex use calcium ions to control the formation of cross-bridges between the myosin head and the actin filaments. When calcium is present, filament sliding and cross-bridge formation occur.

The *Somatic Nervous System*, is part of the peripheral nervous system and actuate the voluntary control of body motility trough the skeletal muscles. Each muscle fibre is innervated by a single motor neuron, which can connect to multiple muscle fibres. The complex composed of a single motor neuron and all the muscle fibres innervated by it is called *motor unit* (MU), as shown in Figure 1.4.

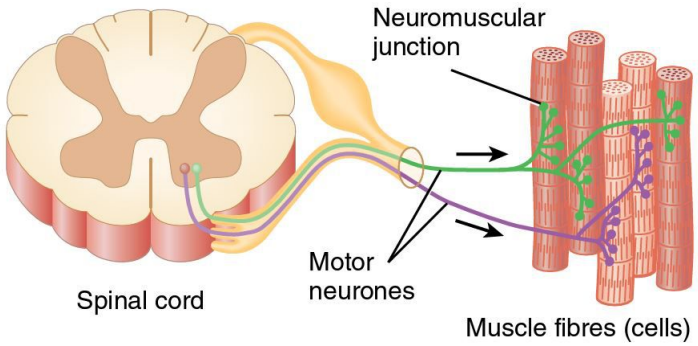


Figure 1.4: Motor unit anatomy [4].

The function of a motor unit is strictly related to its size. Small sized motor units allow precise motor control of a muscle, while larger ones are responsible for gross movements. Most human body muscles consist of a blend of small and large motor units, giving the nervous system a high grade of control over each muscle. The smaller motor units are the first to be activated during movements, while the larger ones are recruited if the movement requires higher strength levels. This mechanism that permits the regulation of the tension generated by the muscle is called *recruitment*. The nervous system can recruit different motor units during the contraction to prevent complete muscle fatigue and permit more prolonged muscle contractions [5].

Based on their speed of contraction and strength, skeletal muscle fibres can be classified into three types:

1. **Type I**: They use aerobic respiration (oxygen and glucose) to produce ATP and contract slowly. They are suitable for prolonged contractions and are the first fibres that are recruited, producing small forces.

2. **Type IIa**: They primarily use anaerobic glycolysis and have rapid contractions. They produce greater forces compared to the type I fibres.

3. **Type IIb**: They primarily use aerobic metabolism and are the most powerful muscle fibres. They are recruited for demanding and short-term efforts and have a high conduction speed that makes them fast to respond but more sensible to fatigue [6].

1.1.2 Forearm Muscles

The twenty muscles present in the forearm are split into two compartments: the anterior, constituted by the flexor muscles and the posterior, composed by the extending muscles [7].

The **anterior compartment** is the flexor compartment, in which muscles have the primary function to flex and pronate the digits and wrist. This compartment consists of a superficial, an intermediate and a deep layer.
In the **superficial layer**, there are four muscles: *Palmaris Longus*, *Flexor Carpi Ulnaris*, *Flexor Carpi Radialis* and *Pronator Teres*. The first three produce the extension of the wrist, while the last permits the pronation of the forearm. All these muscles originate at the medial epicondyle, which is the common flexor origin.
The *Flexor Digitorum Superficialis* is the only muscle present in the **intermediate compartment**. Its action generates the flexion of the metacarpophalangeal and proximal interphalangeal joints of digits II-V [8].
Lastly, the **deep compartment** contains three muscles: *Flexor Digitorum Profundus*, *Flexor Pollicis Longus*, and *Pronator Quadratus*. The *Flexor Digitorum Profundus* allows the flexion of the distal interphalangeal joints of the digits. *Flexor Pollicis Longus* is responsible for flexing the wrist, the metacarpal-phalanx and the interphalangeal joints of the digit. The *Pronator Quadratus*, together with the *Pronator Trees*, aid the pronation of the forearm. The *Flexor Digitorum Profundus* starts proximally on the ulna, the *Flexor Pollicis Longus* on the anterior radius and the *Pronator Quadratus* on the distal anteromedial ulna.

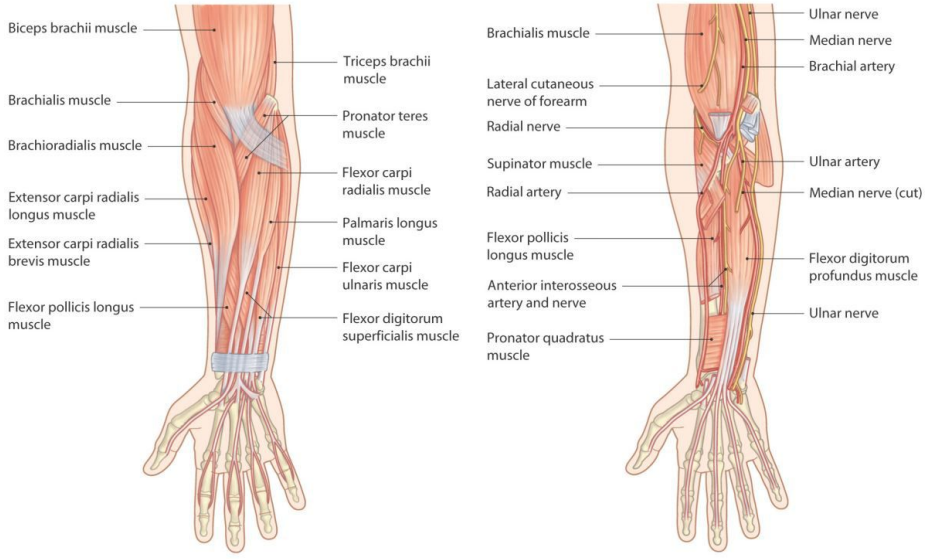


Figure 1.5: Anterior compartment muscle groups [9].

The **posterior compartment** is the extensor compartment since its prime function is to extend the wrist and the hand's digits and thumb abduction. This compartment consists of a superficial and a deep muscle group.
The **superficial compartment** comprises seven different muscles. The *Extensor Digitorum Communis* generates the extension of the wrist, the metacarpal-phalanx and the interphalangeal joints of the phalange. The *Extensor Digiti Minimi*, is

responsible for the extension of the wrist and the fifth digit. The *Extensor Carpi Ulanris* coordinates the extension and abduction of the wrist together with the *Extensor Carpi Radialis Brevis*. The *Extensor Carpi Radialis Longus* permits the abduction and extension of the hand and also hand grasping. The *Brachioradialis* is primarily an elbow fixator but also produces elbow flexion. Finally, the *Anconeus* grants support to the other muscles.
The five muscles contained in the **deep layer** all start from the posterior interosseous nerve. The *Supinator* origins from the lateral epicondyle of the humerus and assists the supination of the forearm. The *Abductor Pollicis Longus* consents to wrist extension, thumb abduction and the extension of the carpometacarpal joint. The *Extensor Pollicis Longus* and the *Extensor Pollicis Brevis* are both involved in the wrist and the thumb extension. They also control the extension of the metacarpophalangeal and the carpometacarpal joints of the first digit. The *Extensor Indicis* allows the extension of the second digit.

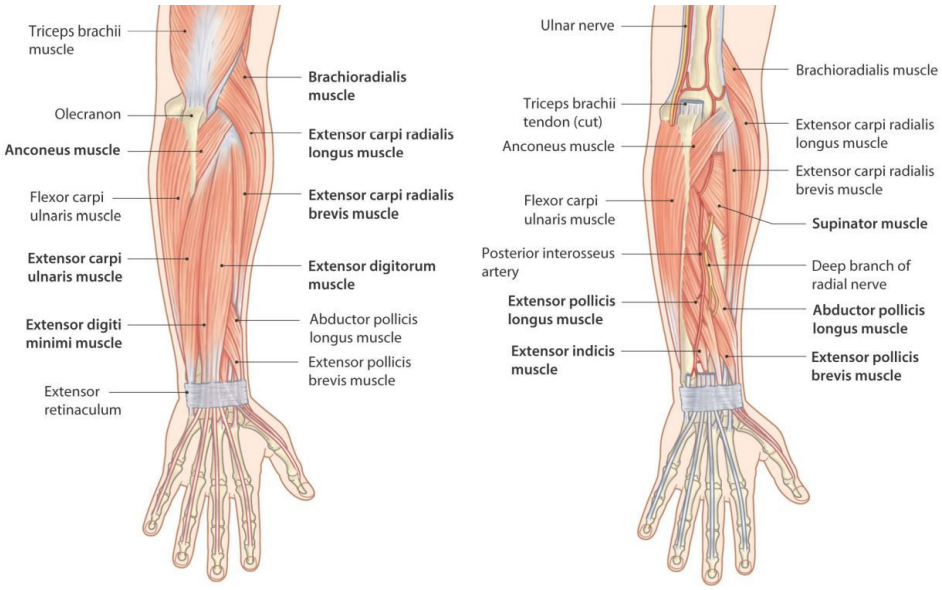


Figure 1.6: Posterior compartment muscle groups [9].

1.2 ElectroMyoGraphic (EMG) Signal

The EMG signal detects the electrical activity of the muscle fibres that activate during a contraction, presenting depolarizing and repolarizing zones. It gives essential information about muscle contraction reflecting neuronal and muscular activity. Biological tissue separates the signal sources from the recording electrodes, acting as a Spatio-temporal low-pass filter on the potential distribution [10].

The intracellular space of muscle fibre at rest condition presents a membrane potential of 70-90 mV, negative inside the cell, positive in the extracellular environment. The sodium-potassium pump (*NaK ATPase*) is responsible for the maintenance of the resting potential. This pump work against the concentration gradient of ions crossing the cell membrane. The action potentials generated by the motor neurons travel to the neuromuscular junctions, causing acetylcholine

release in the space between the muscle fibre membrane and the nerve terminal. Acetylcholine causes the excitement of the fibre membrane and the consequent generation of a local potential gradient, corresponding to an inward current density (depolarization zone). The generated *intracellular action potential* (IAP) propagates along the fibre down to the tendons ending, leading to an ionic transmembrane current profile that propagates along the sarcolemma [11].
The IAP consists of 3 phases:

1. **Depolarization**: When a stimulus occurs changing the membrane potential over the value of the threshold potential equal to -55 mV, an action potential arises. An action potential follows the *all-or-nothing law*, meaning that any subthreshold stimulus will cause no response. During the depolarization, the calcium channels open, causing the entry of calcium ions that bring the membrane potential to a value equal to +40 mV.

2. **Repolarization**: in this phase, the potassium channels open and lead to the entrance of potassium ions, bringing back the potential close to its resting state.

3. **Hyperpolarization**: membrane potential stabilizes to its resting state only after this stage in which its value descend below -90 mV.

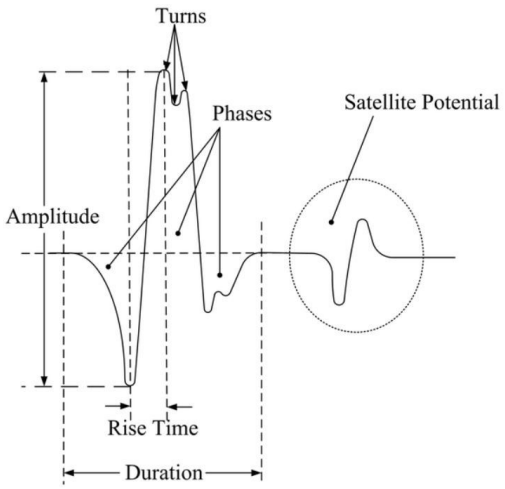


Figure 1.7: Characteristic parameters of the motor unit action potential (MUAP) [12].

As seen in chapter 1.1, the smallest functional unit controlled by the nervous system is the motor unit. The activation of a motor unit and the consequent generation of an action potential will lead to the excitation of all the fibres innervated by the motor neuron. The resulting signal consists of a spatial-temporal summation of all the action potentials produced by the depolarization of the muscle fibres. This cumulative potential is called MUAP (*Motor Unit Action Potential*). The amplitude of these potentials collected through the ElectroMyoGraphy will vary based on the position and characteristics of the electrodes used for the acquisition and the muscles' properties. Also, multiple MUAPs are usually collected by the

electrode in the detection zone, which will acquire the contribution of multiple motor units [10].

Electromyograms can be collected from the skin surface or from within the muscle [13].

1. **Surface Electromyography (sEMG)**: is a non-invasive technique in which the electrodes for recording the EMG signal are placed on the skin's surface, making this technique easy to use and suitable for dynamic acquisitions. Surface EMG is not specific to the muscle over which electrodes are placed due to its low selectivity. The recorded signal is the summation of the signals generated by all the motor units recruited for that specific movement. Bipolar surface EMGs provides global evidence on the extent and timing of muscle activity. In particular, the sEMG technique is mainly used to analyse muscle disorders or abnormalities and diagnose neuromuscular diseases. It is used in muscle rehabilitation and to control prostheses and orthosis [14].

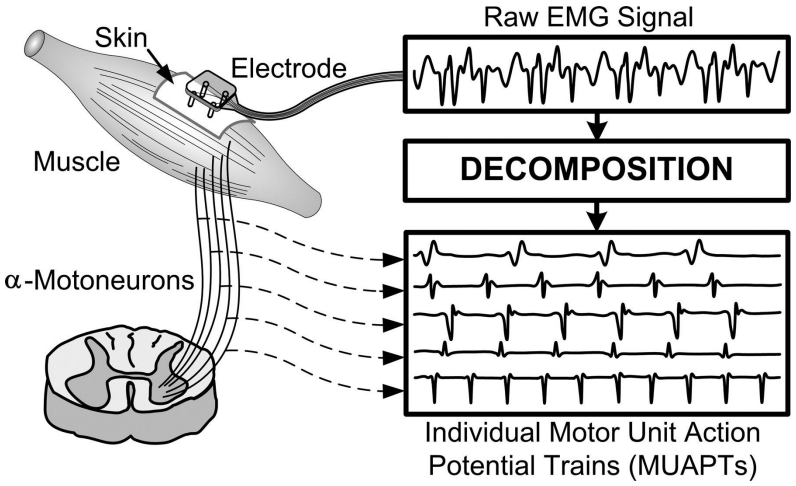


Figure 1.8: Surface Electromyographic signal (sEMG) decomposed in its constitutive MUAPs) [15].

2. **Intramuscular Electromyography (iEMG)**: It is an invasive technique. The electrodes used are needles inserted directly into the muscle of interest. Due to its high selectivity, the signal collected represent the activity of a single motor unit. Contrary to the sEMG, this technique doesn't present any filtering effect induced by the tissues, as the electrode is positioned directly inside the muscle. iEMG permits the study of physiological properties of motor units, such as recruitment threshold and fatigue. Due to its invasiveness and the inability to be used in dynamic conditions, it is commonly used to diagnose myopathies.

1.2.1 Surface ElectroMyoGraphy (sEMG)

The sEMG signal is a stochastic signal that can be approximated with a Gaussian distribution. Its peak-to-peak amplitude varies between 0 - 10 mV, and its frequency content is contained between 0 - 500 Hz. The spectral components that contain the most prominent information are those between 50 and 150 Hz [16].

A typical acquisition chain for the sEMG consists of four main blocks: detection, amplification, conditioning and digitalization. The signal acquisition requires the placement of electrodes on the subject's skin. Their positioning concerning the muscle is essential to obtain a good quality signal. After that, amplification adapts the sEMG signal to the ADC dynamics, used later in the digitalization step. The filtering blocks are introduced to obtain a recording in which only the band of the sEMG signal is present and to attenuate artefacts and noise.

In fact, during the acquisition of the signal, multiple sources of noise are present that afflicts the final quality of the electromyogram [17]:

- **Inherent electrical noise**: electronic components introduces intrinsic white noise throughout the band of interest. The use of properly designed circuits and quality electronic components can reduce its influence. However, its contribution cannot be completely eliminated.
- **Power line interference**: it is also a type of electrical noise. Parasitic capacitances with the ground and power line, form voltage dividers that lead to the generation of interferences with amplitudes more significant than that of the EMG signal. The dominant frequency is 50 Hz or 60 Hz, respectively, for the EU and the USA. The interference should be rejected by the detection system avoiding saturation of the amplifier.
- **Cross-Talk**: the low selectivity of the sEMG signal, together with the simultaneous activation of several muscle fibres during the actuation of a movement, generates cross-talk in the recorded signal. During the recording, the sampling system detects the activity of neighbouring muscles. Cross-talk is always present but can be improved by controlling the positioning of the electrodes and their inter-electrode distance.
- **Motion artefacts**: is a low-frequency noise caused by the relative movement of the electrodes sliding on the skin or the movement of the amplifier system cables. Because the noise is below 20 Hz, it can be removed with proper circuit design.
- **Electrocardiographic artefacts**: the activity of the heart is detected by the sEMG acquisition system. This biological noise cannot be removed, but it can be reduced using bipolar recording with high CMRR channels.
- **Muscular fatigue**: fatigue can appear during a prolonged or strenuous muscle contraction, leading to decreased conduction speed and features in the frequency domain and increased values of the features in the temporal domain, as reported in the next section.

There are many types of electrodes on the market. The electrodes shape, size, inter-electrodic distance and material will influence the acquisition system. The

most used are silver chloride (AgCl) electrodes thanks to their property of not being polarizable, but also silver/silver chloride (Ag / AgCl), gold (Au) and silver (Ag) are common. These materials are conductive, making it possible to observe the electrical activity of the muscles. **Wet electrodes** use a conductive electrolytic gel on the metal surface interfacing the skin surface. The gel permits good electrode adhesion, diminishing the electrode-skin impedances, reducing the noise caused by friction and increasing the mechanical stability. Disposable electrodes with a built-in gel layer reduce the application time and are accessible in different shapes and sizes where these criteria influence the spatial resolution. Nevertheless, these electrodes have some drawbacks. Being disposable, they cannot be reused, and they are difficult to recycle. Also, the use of gel requires a preliminary skin preparation step. Finally, throughout the acquisition, the gel will dry, causing skin irritation that leads to variations in its electrical characteristics, such as impedance.

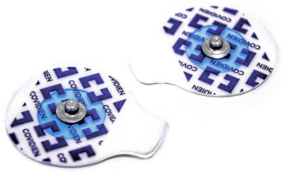


Figure 1.9: Example of gelled electrodes on the market [17].

Dry electrodes do not require the use of conductive gel, making them reusable. They are also suitable for long term acquisition or continuous monitoring through direct contact with the skin. The disadvantages, in this case, are the necessity to have the electrodes well pressed over the skin, avoiding their misplacement during dynamic acquisitions and the high electrode-skin impedance that must be taken into consideration in the electronic circuitry fabrication.

Figure 1.10: Example of dry electrodes on the market [18].

For the sEMG, two sampling techniques can be used:

- **Monopolar configuration**: it uses two electrodes, an active one placed on the skin above the muscle to be examined and a reference one placed in a neutral position. A differential amplifier receives the acquired signals, returning their difference. Even though it is easy to implement, this technique is usually not used because of its noise.

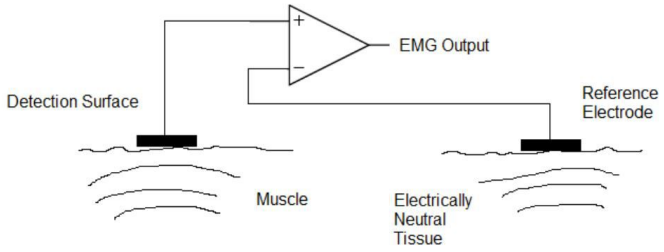


Figure 1.11: Monopolar configuration.

- **Bipolar configuration**: it uses a pair of active electrodes, both set up on the muscle of interest, and a third reference electrode, again positioned in a neutral position. The standard inter-electrode distance is about 2 cm. This recording grants better resistance to disturbances and noise but is less selective and requires a complex positioning for small muscles. The signals deriving from the active electrodes are the input for the instrumentation amplifier that amplifies their difference, rejecting the common noise. This configuration is called the *single differential*.

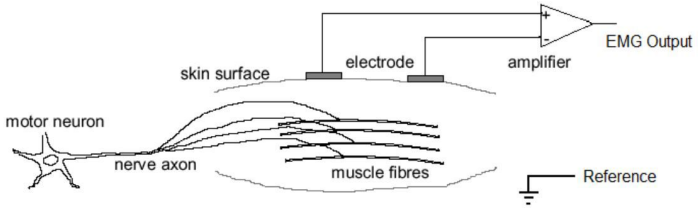


Figure 1.12: Bipolar configuration.

The double differential can also be used, but it is less common. It uses, in this case, three active electrodes with three instrumentation amplifiers, as shown below in Figure 1.13. This mode permits the creation of a spatial filter making it further selective to detect more superficial signals.

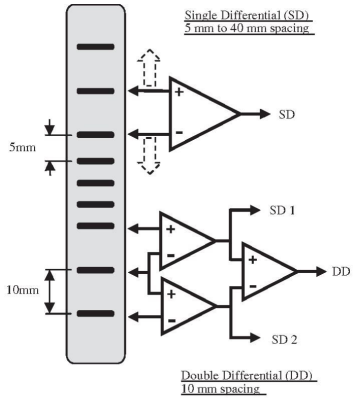


Figure 1.13: Double differential.

1.2.2 sEMG Feature Extraction

The information in the sEMG signal is decoded using *feature extraction*, which aims to eliminate irrelevant and noisy data and retain relevant and informative data. From literature, it can be seen that three are the main domains from which the features can be extracted: time, frequency and time-frequency. In the last years also the fractal domain started to be investigated. In this dissertation, the last two domains are not considered due to their high complexity [19].

Time-domain

Time-domain characteristics are easy and fast to implement, being computed on the raw EMG signal. However, the main drawback resides in the stationary assumption over the signal. The most used are:

- **Root Mean Square (RMS)**: this parameter provides the most insight on the amplitude of the signal, giving a measure of the signal power.

$$RMS = \sqrt{\frac{1}{N}\sum_{n=1}^{N} x_n^2} \tag{1.1}$$

- **Variance of EMG (VAR)**: it employs the power of the sEMG signal as a feature.

$$VAR = \frac{1}{N-1}\sum_{n=1}^{N} |x_n|^2 \tag{1.2}$$

- **Mean Absolute Value (MAV)**: it gives information about muscle contraction level, and it is computed taking the average of the absolute value of the signal.

$$MAV = \frac{1}{N}\sum_{n=1}^{N} |x_n| \tag{1.3}$$

- **Mean Absolute Value Slope (MAVSLP)**: is determined as the difference of the MAVs computed over adjacent signal segments.

$$MAVSLP = MAV_{i+1} - MAV_i \tag{1.4}$$

- **Integrated EMG (IEMG)**: it is the integral sum of the sEMG signal. It is normally used as an index of muscular activation, and it is given by the sum of the absolute values that the signal acquires in a temporal window.

$$IEMG = \sum_{n=1}^{N} |x_n| \tag{1.5}$$

- **Waveform Length (WL)**: It measures the signal complexity, defined as the cumulative length of the waveform over a time window.

$$WL = \sum_{n=1}^{N} |x_{n+1} - x_n| \tag{1.6}$$

- **Simple Square Integral (SSI)**: is equal to the summation of the absolute square energy in the time domain.

$$SSI = \sum_{n=1}^{N} |x_n|^2 \tag{1.7}$$

Frequency domain

The characteristics in the frequency domain are estimated from the *Power Spectral Density* (PSD) of the sEMG signal. This step is time-consuming, making these features not suitable for real-time applications.

- **Mean Frequency (MNF)**: it represents the mean value of the PSD. It is obtained as the sum of the product between the frequency and the PSD of the signal divided by the sum of the PSD.

$$MNF = \frac{\sum_{i=1}^{M} f_i PSD_i}{\sum_{i=1}^{M} PSD_i} \tag{1.8}$$

- **Frequency Median (MDF)**: corresponds to the PSD median, dividing the sEMG signal power spectrum into two equal areas.

$$MDF = \frac{1}{2} \sum_{i=1}^{M} PSD_i \tag{1.9}$$

1.3 Average Threshold Crossing Technique (ATC)

The acquisition system for the sEMG signal on the market presents a standard circuitry. The electrodes positioned on the skin surface permits the acquisition of the ElectroMyoGraphic signal in a bipolar configuration. In the amplification chain, we can distinguish two main parts. The first part is the analogic circuit which comprises an amplifier and one or more filters. The second is the digital part formed by an *Analog to Digital Converter* (ADC) and a microcontroller. The amplification stage amplifies the signal while the filters remove undesired or noisy frequency components. The ADC and the microcontroller convert and send the data extracted from the EMG signal to the computer digitally.

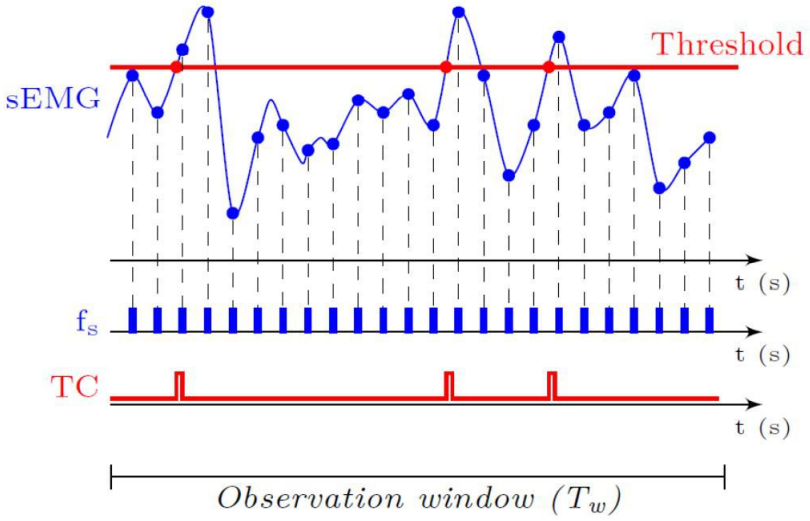


Figure 1.14: Implementation of the ATC technique [12].

The detection of the muscular contraction can be efficiently obtained using an event-driven technique based on a threshold. This bio-inspired approach is founded on an engineering paradigm that mimics the human body's highly efficient voluntary control strategies, resulting in the minimization of the sent information. The *Average Threshold Crossing* (ATC) technique is a transmission technique based on the generation of events when the sEMG signal surpasses a threshold. The implementation is obtained via hardware providing two signals to a voltage comparator: the sEMG signal and a threshold voltage with which the comparison is made [[20],[21],[22]].
The signal exiting from the comparator is quasi-digital, containing its information in the time domain and maintaining a high logical state when the sEMG signal exceeds the threshold and a low logical state otherwise. The reading of the analog signal is managed by the Interrupt Capture Unit (ICU) since the TC signal has only two logic levels. So, the ATC is equal to the number of times the sEMG signal surpasses the established threshold within a predefined time interval, divided by the length of the interval itself. The addition of a comparator to the standard acquisition chain permits the extraction of the ATC parameter from the electromyographic signal. The microcontroller has the possibility to transmit both the ATC and the sEMG

signal. The ATC data can be combined with wireless transmission, triggered by the threshold exceeding, obtaining a quasi-digital signal that permits the attainment of a device with low power consumption and good to interferences. The number of transmitted events drastically reduces with a threshold-based transmission, as clearly seen in Figure 1.14. In the current application, the threshold is maintained fixed during the acquisition. However, it is possible to make it dynamic to augment the selectivity in the detected events, further reducing the power consumption.

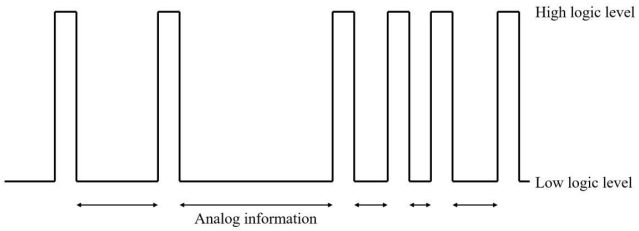


Figure 1.15: Quasi-digital signal.

Methods using the features introduced in 1.2.2 for the detection of the onset and offset of a muscular contraction result in high power consumption, low real-time performances and high computational complexity. This technique has been selected as the most appropriate for this application because of three main advantages [23]:

1. Simplification of the information broadcasting thanks to the event-driven sampling, which reduces the complexity of the classification algorithms.

2. Low power consumption, due to the narrowband required for the transmission with not continuously acquired data.

3. Simplification and reduction in the dimension of the hardware making it suitable for wearable devices.

Nevertheless, the primary challenges related to this technique's use are the difficulty in choosing the threshold for event revelation. The loss of information in the original sEMG signal that cannot be recovered, in particular the signal morphology, is completely lost, precluding the possibility to extract characteristic features of the sEMG signal.

1.4 Machine Learning Classification Algorithms

Different classifiers have been tested and ultimately used in the final system to perform hand gesture recognition. This section covers a brief introduction to machine learning and the theory behind the classifiers used.

Through training, machine learning algorithms find correlations and patterns in large data sets and make predictions based on that analysis. Their prediction ability improves the more data they have access to and with use [24]. When building a machine learning model, we can choose among different algorithms based on the problem at hand. However, for the *no free lunch theorem* [25], it cannot be stated that if an algorithm has worked well in a specific field, it will also fit other problems belonging to the same area. Moreover, algorithm performance is strictly related to the quality of the data passed as inputs to it in the training phase. It is proved that any model will reach the same accuracy if supplied with enough data [26]. For this reason, particular attention in the development of a machine learning model is reserved for the collection and processing of data.

A learning algorithm deployment has two main phases. The first is the model generation, in which the algorithm is trained over data passed to it. In this phase, the algorithm is able to create a data representation that will later be used to predict new outputs. The second phase is inference, consisting of passing unseen data to the trained model and evaluating the correctness of the provided output. In a few algorithms, these outputs are used to improve or rebuild the learning model. The training and the inference blocks are neatly separated. Therefore they can run on different machines. The learning phase requires high computational and memory resources, increasing with the data complexity. The model training involves splitting the dataset into three, where 60% of the data is used for learning, 20% for validation, and the remaining 20% for testing. The validation and testing stages show the accuracy reached by the algorithm.

Machine Learning is a branch of *Artificial Intelligence* that teaches computers to learn from data without explicitly programming.

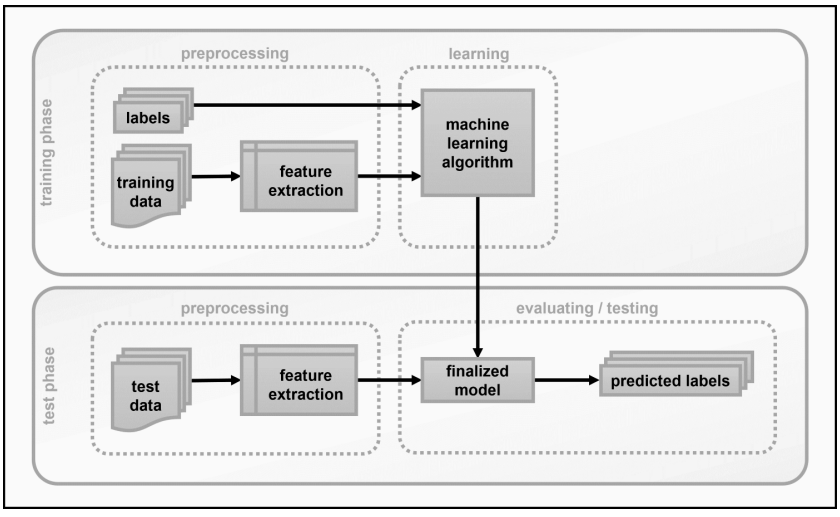


Figure 1.16: Machine learning working flow [27].

Depending on the nature of the data, four different learning models can be used: supervised, unsupervised, semi-supervised and reinforcement learning [28].

- **Unsupervised learning**: employs machine learning algorithms to analyse and cluster unlabelled datasets, discovering data groupings or hidden patterns. The primary function of this learning method is mostly data clustering, but it can also be used to reduce the number of features through dimensionality reduction.
- **Supervised learning**: generates predictive models based on input and output data pairs. Based on the nature of the data, supervised learning algorithms are categorized into *Classification* when the inputs are categorical or *Regression* if the inputs are continuous.
- **Semi-supervised learning**: uses a smaller labelled dataset in the training phase to guide the classification and extraction of features from a larger unlabelled dataset. It is commonly used when there are few labelled data disposable.
- **Reinforcement learning**: is a behavioural learning model, similar to supervised learning. It develops a system called *agent* that betters its performance by interacting with its environment. The agent takes action on each time step and receives a reward value, which measures how good the action is towards the goal to be achieved. The system interacts with the environment in order to take action. The best action is chosen based on the reward. Through the interaction with the environment, the agent uses an exploratory trial-and-error method to learns acts that lead to the maximization of the reward.

Different algorithms have distinct learning dynamics, providing insight into the model plasticity and how it handles data. In *batch learning*, the algorithm trains on a single batch, and once it is generated, it cannot learn anything new. Contrarily, the algorithm is constantly rebuilt on new data, learning on the fly in *online learning* [29]. Finally, the data representation modelled by the algorithms during the training step has different natures, connected to how much memory the model will use. *Instance-based* models store the entire dataset and compare the new inputs with it during inference with a measure of similarity. *Model-based* deduces a function to obtain the output from the new feature vectors.

1.4.1 Neural Network

An Artificial Neural Network (ANN) is a supervised classification method which takes inspiration by how the human nervous system process information. Artificial NN comprises multiple layers of nodes or neurons. The neuron is the basic information processing unit that collects inputs and generates an output [30]. It consists of:

- A set of connecting links, each defined by a numeric weight that defines the input's strength to that specific neuron. These interneuron connection strengths are used to store the acquired information. The weights are modified

to represent the specific learning task based on the characteristics of the training examples.

- An adder function that calculating the weighted sum of the inputs.
- A non-linear activation function that limits the output of the neuron in terms of amplitude. Different activation functions can be chosen: *step function*, *sign function*, *sigmoid function* and *Rectified linear unit function* (ReLu). Each of them emulates the typical response of the biological neuron, which only fires if the total input of the incoming synapses is higher than a threshold potential.

$$y = h_{W,b}(x) = f(b + w^T x) = f(b + \sum_{i=1}^{n} w_i x_i) \tag{1.10}$$

where $f : R \to R$ is called the activation function.

The main components of a neural network are the neurons, but the behaviour of the network is related more to how they are organized and connected.

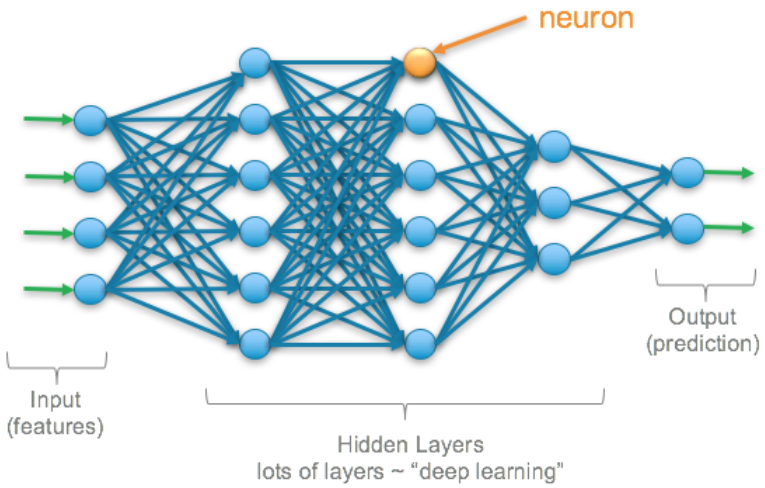


Figure 1.17: Neural network structure [31].

The most common architecture is the feed-forward neural network. In this organization, the connections can have only one possible direction. The neurons are organized in sequences of multiple layers linked from the inputs to the outputs. The input layer contains a certain number of input units that represent the inputs of the network and do not perform any calculation. The number of inputs is equivalent to the number of elements of the feature vector. The output layer provides the neural network's output, taking a certain number of inputs and returning one single output. In between them, there can also be hidden layers. Their amount and the number of nodes per layer dictate the complexity of the learning model. In *fully connected layers*, each input is provided to each neuron of the network [32].

Training an NN is about finding values of the network parameters (W, b) to make the training error small, minimizing the cost function $J(W, b)$, using *Backpropagation algorithm*, where the cost function is:

$$J(W,b) = \frac{1}{m} \sum_{t=1}^{m} \frac{1}{2} (h_{W,b}(x^{(t)}) - y^{(t)})^2 + \frac{\lambda}{2} \sum_{l=1}^{n_{l-1}} \sum_{i=1} \sum_{j=1} |W_{ij}^{(l)}|^2 \tag{1.11}$$

Where b is the bias, W are the weights and λ, the regularization parameter, that penalizes the magnitude of the wights, m represents the training point quantity and l the considered layer.

The initialization of all the parameters W and b is performed with small random values, serving the purpose of symmetry breaking and after, an optimization algorithm is used. Usually, batch gradient descent is selected even though it is susceptible to local optima, being the cost function non-convex. All the parameters W, b for all the neurons are updated in one iteration. Backpropagation algorithm is an efficient recursive way for computing the above derivatives, based on the chain rule. The gradients are calculated backwards, starting from the output going to the network's input. In each pass, we update the parameters accordingly to the gradient.

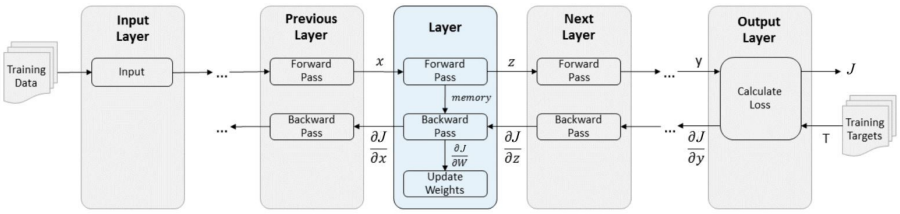


Figure 1.18: Training of a NN: Backpropagation algorithm.

1.4.2 Support Vector Machine

A Support Vector Machine (SVM) is a supervised algorithm applied for regression and classification problems. SVMs seek the hyperplane that best divides a dataset into two classes. A *Hyperplane* (H) is a discriminative surface that linearly separates a set of data and it is used for their classification. Geometrically, H is the zero sublevel set of an affine function that separates the whole space R into two half spaces.
Data points are classified based on staying on one side or the other of this decision boundary. If the point lies further from H, there is more confidence in its correct classification.

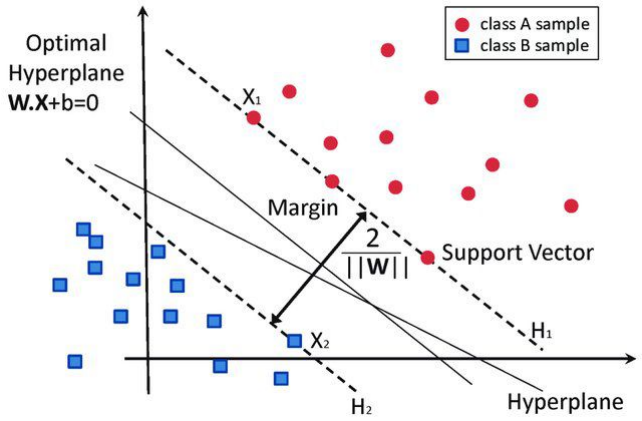


Figure 1.19: Hyperplane in support vector machine algorithm [33].

The algorithm aims to construct the separating hyperplane based on a penalty measure equal to the distance of the point from it. Data points that lay in the correct space do not receive any penalty; the misclassified ones are given a penalty equal to their distance from *H*. The hyperplane selected is the one that gives the maximum separation margin between the two classes and that minimize these misclassification errors. This class of models is called liner *Linear Support Vector Machine* (L-SVM).

$$min\ C\sum_{i=1}^{N}[1 - y_i(w_0 + w^T x_i)]_+ + \frac{1}{2}||w||_2^2 \tag{1.12}$$

Here, C is the regularization parameter that penalizes the misclassification error and N is the number of data points.

However, in binary classification problems, the data is often far from being linearly separable. The problem is overcome by introducing a nonlinear feature map that transforms the feature vector non-linearly. Additionally, the feature map $\phi : R_n \rightarrow R_d$ maps the n-dimensional input space into a d-dimensional feature space, typically with $d >> n$. Data non-linearly separable in the primary input space may become linearly separable when brought in a higher dimensional feature space. The decision boundary in the primary input space is now nonlinear and determined by the equation:

$$w^T\phi(x) + w_0 = 0 \tag{1.13}$$

Usually, this nonlinear function is not known. Constructing the dual formulation from the primal (1.12) we can solve it without explicitly knowing the feature map but using the kernel matrix related to it. A *kernel* is a function that operates a transformation in the primary feature space on the input data with the advantage of not increasing the dimensional space [34]. The most used kernel functions are *sigmoid*, *polynomial*, *radial basis function* (RBF). After choosing the most suitable kernel, SVM classifiers require careful tuning of the parameter, mainly C and gamma.

- The regularization parameter C controls the penalization of misclassified points. Increasing its value leads to smaller margins and lower misclassification rates.
- *Gamma* defines the influence of the single training examples. A lower value considers points further from the hyperplane, and a higher value only points closer to it.

1.4.3 Random Forest

Random Forest is an ensemble classifier based on Decision Trees, a non-parametric supervised learning method.

Decision Trees are simple supervised classifiers that operate a partition of the instance space recursively. The trees are built top-down, generating nodes, starting from an initial one called *root* that has no incoming terms. The other nodes receive only one input and are called internal nodes when they have outgoing edges or

leaves or decision nodes otherwise. *Iterative Dichotomiser 3* (ID3) algorithm is used for the generation of decision trees. The algorithm constructs them top-down, starting by finding the attribute that should be tested first. *Entropy*, a statistical test that assesses how well the feature alone classifies the training data, is commonly used. The entropy of an empirical probability distribution is defined as:

$$H = -\sum_{i=1}^{m} p_i log2 p_i \tag{1.14}$$

Where $p_1, ..., p_m$ is a general discrete distribution of the output values.
It gives a measure of the impurity of the samples. Low entropy means that members of a point set tend to belong to the same class; otherwise, they are well mixed. Together with entropy, decision trees use also another metric that measures the ability of an attribute to classify the training instances correctly. In ortìder to quantify the attribute efficacy, *Information Gain* (IG) is computed as the decrease in entropy obtained by dividing the instances of the dataset according to the considered attribute.

$$IG(y, A) = H(y) - H(y|A) \tag{1.15}$$

The selected attribute is the one with the highest IG and is used as the root of the tree. Starting from the root node, a branch for each possible value of the attribute is created, and the training data are sorted accordingly. This process is repeated for all the features.

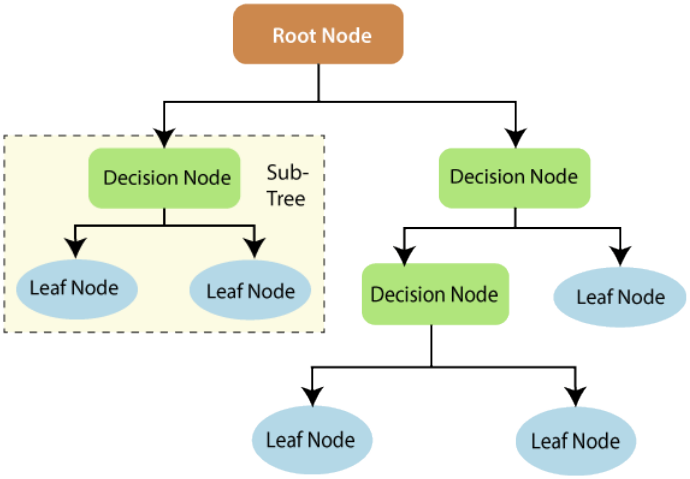


Figure 1.20: Example of a Decision Tree [35].

Random Forests (RF) are ensemble methods explicitly destined for decision tree. RF introduce two origins of randomness: bagging and random vector vectors:

1. **Bagging**: each tree is grown using a bootstrap sample of the training data. Given a training set D, containing N samples, Bootstrap sampling creates a set $D_i, i = 1, ..., k$ by drawing N examples at random with replacement from D.

2. **Random vector method**: the best split is chosen from a random subset of m features at each node instead of all features. The IG is not computed for all the attributes but only for this random subset.

For each D_i a different classifier is trained, and the new instances are classified by majority voting.

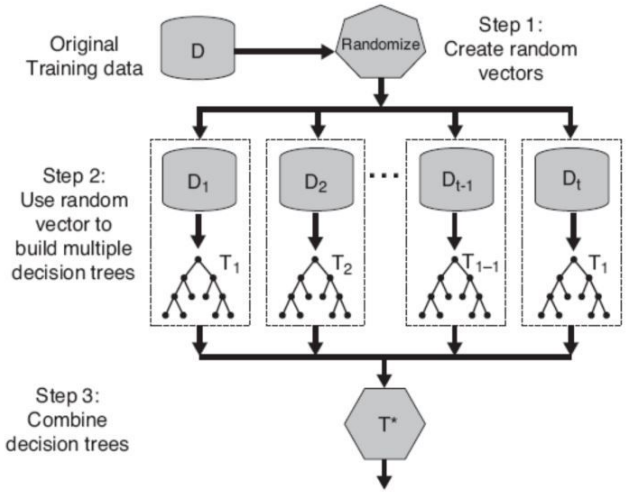


Figure 1.21: Random Forest algorithm.

1.4.4 K-Nearest Neighbour

K-Nearest Neighbour (KNN) algorithm is a non-parametric simple, supervised classifier used in the context of classification and regression [36]. K, the number of considered neighbours, is selected through cross-validation, choosing the one that minimizes the errors while preserving the algorithm ability to generalize to new unseen data. The algorithm assumes that data points belonging to the same class are close to each other. This idea of similarity or proximity is captured by computing the Euclidean distance between points on a graph. The computed distances between the current input and each training example are sorted in ascending order. The first K entries are picked, and the returned output corresponds to the mode of the K labels.
Attention must be paid when using KNN because the classification is time-consuming and computationally intensive with a large amount of data.

k-Nearest Neighbour Search Using a Kd-Tree

Kd-trees divide data into nodes, each containing a fixed amount of data selected by the user. The K-nearest points for a given input are first searched within the node in which the point resides. After, the algorithm chooses all other nodes having any area within the same distance, in any direction from the input point to the k-th closest point. In this identified region, the algorithm searches for points closer to the input. K-d trees are used with large datasets, being much more efficient than using the exhaustive search method, reducing the computational cost [37].

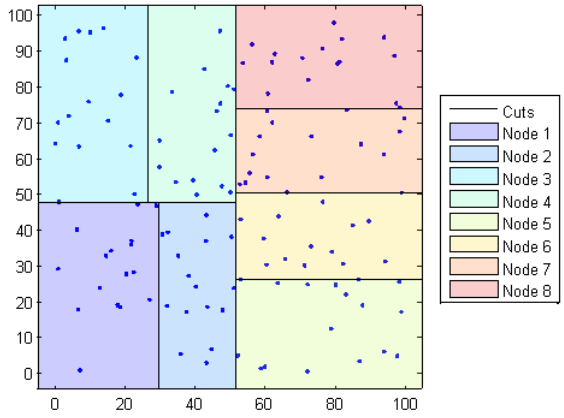


Figure 1.22: K-Nearest Neighbour implementation using Kd-tree [37].

1.4.5 Naive Bayes

Naïve Bayes is a supervised probabilistic algorithm that exploits *Bayes' theorem* under naive conditional independence assumptions. Bayes theorem provides the mechanism to update the prior knowledge based on the evidence of unseen data.

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)} \tag{1.16}$$

$P(\theta|D)$ = Posterior probability, probability of θ given data D.
$P(D|\theta)$ = Likelihood, probability of data D given θ.
$P(\theta)$ = Prior probability, describe the a-priori knowledge about θ.
$P(D)$ = Marginal probability, probability of data D.

The Posterior is computed as the product between the Likelihood and the Prior probability distribution and represents the updated state of knowledge about θ, exploiting the new information brought by the data D. As an underlined hypothesis, the algorithm considers the features conditionally independent given the class label. Moreover, the Likelihood follows one of the statistical distributions: *Gaussian*, *Multinomial* or *Bernoulli*. The Bayesian approach is inherently recursive, which means that it is suitable for online recursive inference.
The classifier works by assigning each input feature vector a class probability $P(x|C_k)$, and then deciding as output class the one that maximizes the above probability.

$$P(C_k|x) = \frac{P(x|C_k)P(C_k)}{P(x)} \tag{1.17}$$

given the assumption of class-conditional independence:

$$P(x_1, ..., x_m|C_k) = P(x_1|C_k)P(x_2|C_k)...P(x_m|C_k) = \prod_{j=1}^{m} P(x_j|C_k) \tag{1.18}$$

The output class if finally given by:

$$y = arg\, max_{k=1,...,k}\, P(C_k|x) \tag{1.19}$$

Chapter 2

State of the Art

2.1 Artificial Intelligence (AI) in embedded systems

Electronic systems have become widely used in several fields and indispensable for many daily activities, aiming to improve quality of life. Embedded systems acquire multiple users' data to adapt and respond to their needs. Connection to data networks allows these systems to share this information and retrieve the elements to formulate proper decisions. In this optic, machine learning provides the tools enabling the decisions introducing the concept of *intelligence* in embedded systems. A critical aspect is the optimization of their computational capabilities that must satisfy memory and battery constraints for practical use [38].

MicroControllers Units (MCU) are the central processor unit of embedded systems. They collect analog or digital data, process information and return a specific output through their peripherals. The rapid development of this field is strictly related to open-source hardware boards, which constitute a free-access platform with a manifold of data introducing a lower-cost strategy.

Systems On a Chip (SOC) recently became broadly diffused, enabling several microprocessors' integration and increasing their processing abilities, having all the system functional elements integrated into one chip. Their ability to acquire data in real-time and process it faster has also improved machine learning algorithms performances.

With this information, an embedded system (ES) is an electronic system designed to carry out a specific function and inserted into a more extensive system device, using sensors and actuators to interact with the external environment. It executes on highly specialized machines that are not primarily computers. Their design is based on the concept of codesign that brings together the software and the hardware components. During the system conception, computation capacity, memory, timing and number of external devices limitations must be considered. Software deployment is carried out according to the specific application. Traditional codesign is an iterative approach that starts from a system functional specification. Its completeness and precision are fundamental for the deployment of an efficient system. Proceeds with the hardware and software components subdivision, which result in three parallel flows [29]:

- **ES compilation**: programmed in C, or less frequently in assembly language.
- **Hardware synthesis**: specified using VHDL, or more recently using model of computations (MOC).
- **Hw/Sw interface realization**

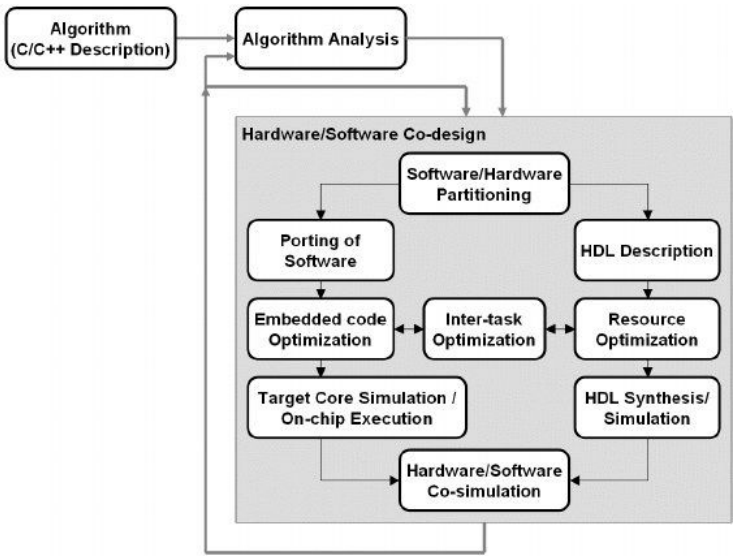


Figure 2.1: Hardware/Software codesign of an embedded system [39].

Integrated and tested in a co-simulation environment. The standard programming language for Hw/Sw design is SystemC, an extension of C++, supporting many levels of abstraction. Together with a system-level description language (SLDL), three elements are required for the software:

- **Processor model**: describe it at various levels of abstraction.
- **RTOS**: supply the real-time timing, connectivity between tasks and synchronization.
- **Software generation tool**: produce code for the chosen RTOS.

ES can be centralized or networked. Networked ES represents a network of nodes that uses wired or wireless communication, Internet of Things (IoT), as an example.

Nowadays, ES has become more complex, networked and intelligent, which has brought to the appearance of a novel category of ES known as intelligent embedded systems (IES). It can be used for decision-making, learning or execution of intelligent algorithms due to its characteristics of self-learning and optimizing, including also AI-software systems.

In particular, Machine Learning and Deep Learning have seen the most development, thanks to technological progress and the uprising of intelligent networks, which has encouraged the creation of ML models and their use in problem solving. An intelligent network has three boundaries [40]:

1. End devices are resource-scarce devices that collect data through sensors and broadcast them to the edge device.

2. Edge devices use internet connections to transmit data to the Cloud.
3. The Cloud processes the received information for storage or as input for an ML model, returning the output that can be sent back to the end device.

This type of system depends entirely on an internet connection for data transfer, introducing delays due to the time required for the request to reach the Cloud and the response to be received and security concerns. In real-time applications, the chosen system's intelligence's location is the end device MCU.

End devices

End devices monitor their environment through sensors and actuators that employ a wide variety of communication protocols. They are usually battery-powered, so they require low power consumption and deep-sleep states. They are application focus aiming at energy saving, and are used in real-time, running an RTOS that allows multi-task and prioritization. MCUs in end devices are resource-scarce with small RAMs, memory storage, and processing power to accommodate these needs. Consequently, end-devices MCUs pose limits in software development. As presented in Section 1.4, ML models are built in two steps. The training phase, being computationally demanding, is carried out in a high-computational space in the Cloud, using high-level languages (Python, TensorFlow, Caffe, MATLAB). Then the inference code is implemented on the end device's MCU, with low-level languages (C, C++), passing through exporters for the code conversion. Examples of available converters are Sklearn-porter [41] a Python package that allows exporting a model trained using Scikit-Learn in different programming languages. MATLAB Coder is a transpiler for MATLAB to C/C++, which, however, cannot directly convert models to C and Weka-Porter that transpile Decision Trees only. These converters do not introduce code optimization and are highly generic to consent to run the code on any platform.

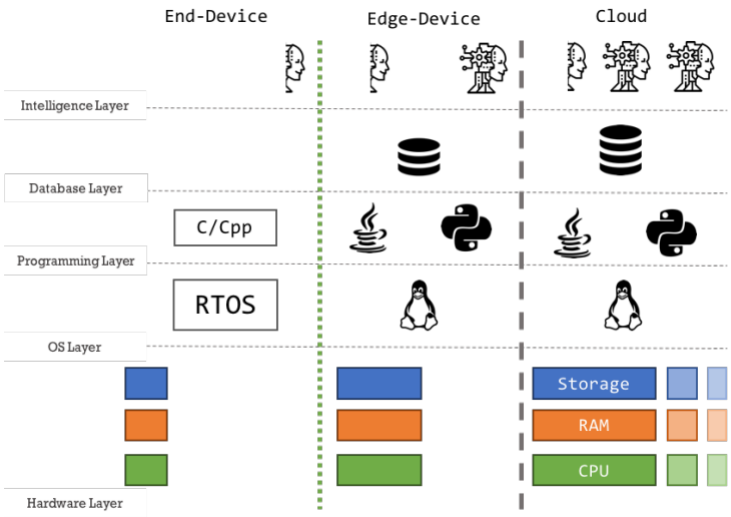


Figure 2.2: Cloud-based, edge-device and end device systems depiction.

In order to enhance these systems capabilities and ML methods portabilities, several technologies and algorithms have been introduced.

MCU with Enhanced ML Capabilities and Compatibility

- PIC32MZ DA family by Microchip (2017) [42]: MCU with high-performance 2D GPU improves execution time and introduce parallel computation, permitting the implementation of more complex models.
- ARMv8.1 MCUs (Cortex-M) with Helium technology (2019) [43]: higher performances in ML and DSP applications.
- ASIC + ARM Cortex-M3 ECMxx MCU by ETA Compute [44]: low-power device that can be integrated into battery-powered systems and used in applications such as healthcare, speech recognition and video. The use of ASIC technology improve performances and lower size and power consumption.

ML Algorithms for Embedded Systems

- ProtoNN [45]: algorithm that implements kNN, taking into consideration the memory constraints of embedded systems. The optimization is carried out with prototypes and low-d projections that build the model to fit the maximum size.
- Bonsai [46]: algorithm based on the creation of a single shallow tree with nodes that can make non-linear predictions. The output is given by summing the predictions provided at each node.
- CMSIS-NN [47]: Cortex-M processor cores software library for NN. Neural Networks generated with it improve 4x their performances and reduce their memory footprint. Its use requires fixed-point quantization.
- FastRNN and FastGRNN [48]: algorithms for the implementation of recurrent neural networks. Achieve significant reduction of these models by using low-rank, sparse quantized (LSQ) matrices and residual connections.

Resource scarce MCUs: ML implementation examples

Szydio et al. have implemented different ML algorithms (i.e. Multi-layer perceptron, Decision tree and Naive Bayes classifiers) into an ARM STM32F429. They executed the training phase in Python environment on a personal computer and exported the obtained models on the MCU. The code has been optimized to fit the memory constraints using compiler optimizations, such as GNU supplied [49].

Leech et al. [50] have implemented an algorithm to estimate room occupancy using an infinite Hidden Markov model and a system composed of a PIR sensor and an MCU. The model has been implemented in MATLAB and exported to C/C++ by making multiple optimizations to decrease the occupied memory. The trained model has been tested on two different microcontrollers (Cortex-M4 with floating-point unit and Cortex-M0), but it cannot satisfy real-time constraints. The model, stored into a 10.36 KB SRAM, shows an accuracy of 80% and a minimum computational time of 1.15 s, using Cortex-M4.

In their work, Haigh et al. [51] developed an SVM model on an ARMv7 and a PPC440 MCUs for decision making in mobile ad hoc networks (MANETs). To bring

and optimize the model in C++, the authors used different libraries, eliminated packages relying on malloc and included different optimization for data structures, algorithms development and numerical representation. They applied inlining to the most called functions and decreased the stack call by collapsing the object hierarchy obtaining a 20% reduction in runtime. They also tested the use of different variables types (integer 32-bits, float 32-bits, and 64-bits double), with the hybrid approach (integer + float) being the fastest.

Parker et al. [52] studied and deployed a distributed neural network using multiple Arduino Pro Mini (APM) MCUs, each representing a node of the NN. The new approach has been developed to brig parallelism into NN. The simple network comprises four APM: an input, a hidden layer with two neurons and an output layer. The communication between the MCUs has been obtained using an I^2C protocol. The model learning is based on the backpropagation algorithm and has been used to learn simple logical operations, achieving 43 s to learn the XOR.

The critical aspects of ML models implementation on resource-scarce MCUs can be briefly summarized as follow:

- **Memory footprint**: the majority of the algorithms requires the storage of data points, hyperplanes or thresholds that occupy a considerable amount of memory. Code optimization is essential to reduce memory footprint while maintaining the achieved accuracy.
- **Power consumption**: low power consumption became a constraint when working with battery-powered devices. Many factors influence this aspect, but regarding wireless devices more than 65% of the power is consumed in communication. In this case, the most effective strategy for power reduction is reducing the number of communication performed. In addition to that, modern systems provides low power and sleep states to further trim consumption down. Another important aspect is memory. Power consumption is higher as its size increases, and some memory types have higher consumes requiring a periodical refresh.
- **Execution time**: time constraints are critical in real-time applications. Execution time is connected to the clock cycle that determines how fast the processor can complete a task. Machine cycle must also be taken into consideration and is the time required by the processor to perform fetching, decoding, execution, and storage, variable among different MCUs. The algorithms must be implemented to run in the smallest amount of machine cycles as possible to complete the desired task.
- **Accuracy**: this aspect is deeply influenced by the data used in the training phase. The construction of a well-balanced dataset with a good representation of all the possible cases and low noise is the key to obtaining high accuracy.
- **Scalability and flexibility**: these two aspects are challenging to achieve. In ML, data from different sources may not be suitable for other platforms because of differences in data acquisition methods or data handling. However, flexibility renders a model scalable and fit for a set of platforms and data. Techniques such as data standardization and preprocessing can make the model more robust.

2.2 Gesture Recognition

Gesture recognition tries to confer a mathematical interpretation to the human movement using computing devices. With the increasing use of technology, hand-gesture recognition has become a primary field of Human-Computer Interaction (HCI) because it gives the machine to comprehend the user needs and respond consequently. The majority of the gesture recognition systems follows three steps:

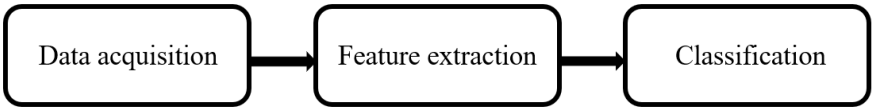


Figure 2.3: Implemented steps in a gesture recognition system.

The data acquisition systems vary among different studies and applications [53], with the most common being:

Wired gloves: in their work, Chouhan et al. [54] created a low-cost wired glove with a 3-axis accelerometer, four Hall effect, and seven bending sensors to map hand and fingers orientation. The sensors acquire the data stored in an array which is then sequentially transmitted to a computer using a UART connection and an automatic repeat request (ARQ) as a fault controlling setup. The preprocessed data is used to train a logistic regression model in the MATLAB environment. The validation phase assessed the correctness of the model classification by evaluating the output of the trained algorithm on new data that arrived on the PC. The reached average accuracy of the system is around 96%. Its intended application is the conversion of sign language to a more human-understandable form.

Motion capture: Bargellesi et al. [55] used a motion capture (MoCAP) system to perform hand gesture recognition. The MoCAP setup consists of 12 infrared cameras with a sampling frequency of 340 Hz, capturing 3D position information from 8 markers. This type of system requires a preliminary step of camera calibration to identify markers, setting up a coordinate reference system in the acquisition environment. The features fed to the Random Forest, chosen as the classification algorithm, have been extracted testing three different methods: time-series cropping (CROP) that reached an accuracy in gesture recognition of 93%, summary statistics (STAT) with an accuracy of 96% and time series resampling (RESAMP) reaching 97%.

Leap motion controller: in their work, Yang et al. [56] developed a Leap Motion Controller (LMC) system for dynamic hand gesture recognition based on a Bidirectional Recurrent NN. LMC represents dynamic gestures employing a set of feature vectors containing information such as coordinate, velocity and acceleration of the fingers, wrist and palm. The dynamic motions are considered as sequential frames obtained through the aggregation of different hand characteristics. LMC utilizes binocular RGB high-definition cameras to enhance gesture mapping efficiency, while infrared cameras lessen the background impact. A convolutional NN extracts the feature data, performing multi-layer convolutional filtering. Two

datasets containing 12 and 10 gestures were tested (using k-fold cross-validation with k = 5), reaching an average accuracy of 93.5% and 90%, respectively.

Stereo camera: Mohidul Alam Laskar et al. [57] exploited stereo cameras to pass from 2D to 3D gesture recognition. In order to be used, the system requires an initial calibration of the two RGB sensors and cameras to assess their geometric relationship. Features extraction is performed on the disparity maps produced from the images captured by the two cameras. The classification algorithm implemented has been the Conditional Random Forest (CRF) employing the OpenCV 2.0 library in C/C++. The proposed system reached an accuracy of 88%.

Wireless sensing: WiCatch is a novel device-free hand gesture recognition system developed by Zengshan et al. [58]. It bases hand motion recognition on the channel state information. This system works with hand reflected weak signals employing an inference elimination algorithm interference reduction produced by the reflection from unmoving objects creating a direct signal. The received signals are sampled in the time domain to extract the virtual antenna array on which the motion locus of the gesture is rebuilt. The classification is conducted using SVM, achieving an average accuracy of 96%.

sEMG signal: Benatti et al. [59] developed and validated an sEMG wearable device for real-time gesture recognition. The system uses an ARM Cortex M4 microcontroller for data extraction and classification, interfaced via SPI to a Cerebro Afe. The acquisition system presented eight channels and a sampling frequency of 1 kHz. The acquired data were transmitted via Bluetooth to a PC for the offline training of the SVM classifier in the MATLAB environment. After the training, the model has been deployed on the MCU to perform real-time classification. The averaged accuracy reached was 89.2% over seven movements to be classified and a computational time that satisfies real-time application constraints.

Relevant features are extracted from the data acquired and used for training and testing the classification algorithm that will ultimately return the output gesture. Classification enables the systems to recognize the executed movements. Hand gesture recognition systems have several application fields:

Robotic control. It is one of the largest and most diversified areas of applications. An application example is the work of Zhao et al. [60] who designed a hand gesture control model for mobile robots. Gestures are extracted from the background and recognized by applying image processing techniques. After that, gesture recognition is obtained by applying a matching algorithm based on the invariant moment matching method and used to control the mobile robots.

Gaming. This application aims to render the virtual gaming experience more immersive and interactive. S. Rautaray and A. Agrawal [61] combined image processing algorithms (e.g. Lucas Kanade and Camshift) implemented in C++ using OpenCV library and haar cascade classifier to create an easy to use system based on real-time gesture recognition. In the game, the user can control the movement of a character using four different gestures. The system showed an accuracy ranging from 93 to 80% based on the selected gestures.

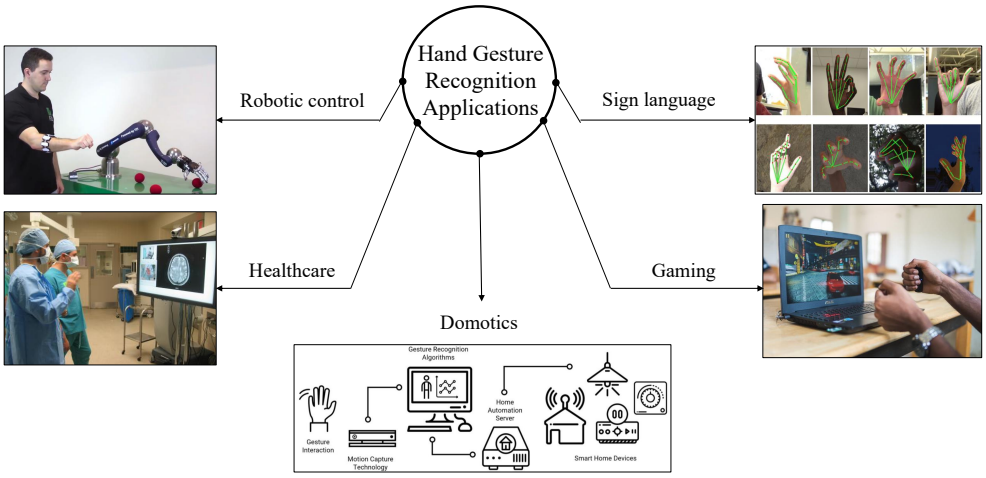


Figure 2.4: Applications areas of hand gesture recognition.

Domotics. Home automation systems create smart homes by monitoring several aspects such as lighting, security and heating. Nguyen et al. [62] developed a three-component system to control house appliances. A smartwatch equipped with accelerometers and gyroscopes (sampling frequency of 25 Hz) acquire data related to the execution of hand gestures and transmits them to a smartphone. Here the classification process occurs, and the output is sent to the home automation platform, connected to smart devices. A deep Convolutional NN (CNN) and a DeepConvLSTM have been tested, obtaining an F1-Score value of 73.7% and 75.8%, respectively, over the execution of 18 different gestures.

Sign language recognition. Sign languages use manual communication, including hand gestures and fingers orientation, arm, head and body movement, and facial expressions to deliver a specific word or meaning. Several works aim to overcome the communication barriers suffered by deaf people using gesture recognition systems. Canavan et al. [63] proposed an automatic system based on a random regression forest and a skeleton-based feature representation from data collected with a LMC system. The real-time setup was tested on the American Sign Language, removing dynamic letters and obtaining a classification rate of 98.36%.

Healthcare. In this field, gesture recognition is mainly used to control prosthesis and orthosis, movement of electric wheelchairs and manipulation of medical images. This last application was implemented by Wachs et al. [64] who proposed '*Gestix*', a video capture based system for the navigation of MRI images within medical records. The accuracy reached in recognition of eight gestures over ten subjects was equal to 96%. The system has also been tested in the surgical context during neurosurgical biopsies by a surgeon at Washington Hospital Center.

2.3 EMG based Armband for Gesture Recognition

2.3.1 Myo Armband by Thalmic Lab

Thalamic Lab company has developed the Myo Armband. This device is the most widespread sEMG armband for gesture recognition due to its ease of use and technical characteristics. The armband has a 9-axis Inertial Measurement Unit (IMU) and eight bipolar channels and recognizes eight gestures. It uses a BLE 4.0 (data transmission chip BLE NRF51882) transmission module, has a maximum sampling frequency of 200 Hz and an eight bits ADC. It also contains a vibrator motor for user feedback. The microcontroller used is the Freescale Kinetis ARM Cortex M4, and it has two rechargeable lithium batteries (3.7 V - 260 mA h) [65].

Figure 2.5: Myo armband [65].

Many articles have investigated its applications and evaluated the relative performances in fields like robot control interface, virtual reality gaming and control of prostheses. For this last application, the Department of Innovation Engineering of the University of Salento and BionIT Labs Company used Myo Armband to control the prototype prosthesis they developed, Adam's Hand [66]. Through Myo armband, an algorithm performs gesture recognition on the rectified EMG and the 9-axis IMU signals and establish the strength required for the movement execution. This time-averaged data are fed to a neural network which weights are obtained during the training stage when the system is started. The myoelectric prosthesis possesses 15 DoF and a single motor to reduce its weight and is provided with actuators and sensors. It permits the fingers' movement independently and generates an appropriate force based on the task to be performed.

2.3.2 gForce-Pro Armband by Oymotion

gForce-Pro is an sEMG hand gesture recognition armband, developed by Oymotion, that tracks arm position monitoring biometric forearm signals. The system consists of a 9-axis motion sensor and eight acquisition channels with bipolar configuration, and it can recognize eight gestures. Data transmission is handled via Bluetooth (BLE4.1), and it is compatible with Windows, Android and Arduino. Its maximum sampling frequency is 1000, and it has a built-in bandpass filter at 200-500 Hz. The ADC works on 8 bits to convert the analog signal to digital. ARM Cortex

M4 High-Performance microcontroller is used to achieve low power consumption (below 0.1 W). OTrain interface helps the user with the set-up of the armband. After establishing the device connection (for which gForce Dongle is necessary), a training phase in which the user can personalize the active gestures starts. After that, the armband can be used offline [67].

Figure 2.6: gForce-Pro Armband by Oymotion [67].

2.3.3 3DC Armband

Laval University's Biomedical Microsystems Laboratory designed the 3DC Armband, a 3D printed wireless system for gesture recognition [68]. The band has ten bipolar acquisition channels and a 9-axis IMU, and it can identify eleven hand and wrist movements. The transmission unit is based on the Enhanced Shockburst, a 2.4 GHz low-power customized protocol similar to BLE. The sampling frequency is 1,000 Hz, and the system contains a bandpass filter between 20-500 Hz. The device is powered by a 100 mA h LiPo battery and uses a low-power microcontroller unit and transceiver. Its performance has been evaluated and compared with the result obtained from the Myo Armband.

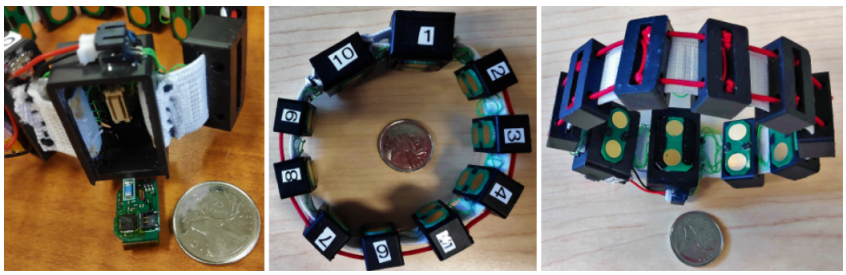


Figure 2.7: 3DC Armband [68].

The researchers collected a new dataset containing the signal from 22 subjects and tested different classification algorithms fed with three different data types: raw sEMG signal, baseline feature set and time/frequency domain features. The best accuracy was reached using the raw sEMG signal and a ConvNet classifier, attaining an average accuracy of 89.47% against the 86.41% of the Myo Armband.

2.4 ATC in Hand Gesture Recognition

The Istituto Italiano di Tecnologia (IIT) started to study the prospect to use ATC data in sEMG detection in their study [20]. They developed a wireless low-power system for miniaturized biomedical applications based on the ATC data with an Impulse Radio Ultra-Wide Band (IR-UWB) transmission technology. They validate the use of the ATC parameter to obtain information about muscle activation by computing its correlation with the muscle generated strength, measured with dynamometers, achieving a value of 0.95 ± 0.02 and comparing it to the correlation calculated between the sEMG signal and the ARV (i.e. 0.97 ± 0.02). Their work continued in [21]. The ATC transmission was extended to a multi-channel case using an Address-Event Representation (AER), and the system has been deployed on a full-custom chip. Further simulations on the ATC were also conducted to justify its final intended use in gesture recognition. They proved ATC parameter robustness against saturation and distortion effects caused by the amplifier and that 5-6 dB of Signal to Noise Ration (SNR) allows to the ATC correlation value to attain its maximum. Correlation also shows a 70% tolerance for lost events.

The MiNES research group employed the ATC feature in ML algorithms for hand gesture recognition in the last few years, starting from Sapienza et al. [69]. Their work developed a system to recognize four hand movements (wrist flexion, wrist extension, hand grasp, wrist radial) and the idle state using three channels positioned on the forearm. In the MATLAB environment, an SVM classifier was trained using the Statistics and Machine Learning Toolbox. The model performance and classification time reached fair values with a latency of 160 ms, while the average accuracy obtained after cross-validation is shown in Table 2.1.

Table 2.1: SVM classifier performance obtained by Sapienza et al.

	Accuracy(%)	Sensitivity(%)	Specificity(%)	Precision(%)
EX	88.25	73.00	93.33	78.49
GR	93.25	91.00	94.00	83.49
UD	92.00	84.00	94.67	84.00
FL	98.00	95.00	99.00	96.94
Avg.	92.87	85.75	95.25	85.73

Continuing on this application, Mongardi et al. [70] increased the number of active, distinguished gestures up to five, introducing the wrist ulnar deviation. NN was selected as classification model to reduce power consumption and move the classification process on an MCU. The Network architecture presented 2 hidden layers with 26 neurons each. A custom board was used for the collection of the sEMG signal and ATC data extraction. The selected MCU was the Apollo 2 by Ambiq. The validation phase required the acquisition of a new dataset using three couples of 24 mm electrodes. The obtained system latency is 268.5 ms while the performances are reported below:

Using the same dataset, Barresi [71], replaced the MCU used in the previous work with an Apollo 3 and tested two additional machine learning algorithms,

Table 2.2: NN performance obtained by Mongardi et al.

	Accuracy(%)	Sensitivity(%)	Specificity(%)	Precision(%)
EX	96.87	84.64	99.20	91.34
FL	96.47	90.29	88.30	89.28
RD	97.18	91.76	91.30	91.53
UD	94.97	93.63	74.90	83.22
GR	94.92	88.06	88.40	84.06
ID	97.63	87.57	100.00	93.37
Avg.	96.34	89.32	89.02	88.80

K-Means and SVM. The average accuracy obtained in the training phase was 83.35% and 95.3%, respectively, encouraging the algorithm's deployment on the MCU. The system latency was equal to 239.85 ms for the SVM and to 130.124 ms for the K-Means. Online prediction results are reported in Table 2.3 and 2.4.

Table 2.3: SVM performance obtained by Barresi et al.

	Accuracy(%)	Sensitivity(%)	Specificity(%)	Precision(%)
EX	96.79	78.84	79.34	79.13
FL	97.50	78.92	88.16	83.24
RD	98.86	99.38	86.09	92.26
UD	96.83	66.66	68.97	67.80
GR	98.08	80.00	81.36	80.67
ID	99.83	99.94	99.82	99.87
Avg.	97.98	83.95	83.05	83.07

Table 2.4: K-Means performance obtained by Barresi et al.

	Accuracy(%)	Sensitivity(%)	Specificity(%)	Precision(%)
EX	95.00	97.06	35.87	79.13
FL	93.71	52.65	100.00	83.24
RD	99.62	98.90	96.25	92.26
UD	92.79	40.00	98.28	67.80
GR	93.46	62.14	35.16	80.67
ID	96.25	99.74	94.72	97.16
Avg.	95.14	75.08	76.71	69.64

Finally, in his work Tolomei [72], started a further investigation on the previous dataset using different machine learning algorithms: NN, SVM, RF, Naive Bayes and Gaussian Mixture Modelling (GMM). Among them, only the best-performing ones were deployed on the MCU and compared as reported in Table 2.5.

Table 2.5: Online comparison of the ML algorithm tested by Tolomei et al.

	Global Accuracy(%)	Computational Time(µs)	Average Power Consumption(mW)
NN	88.15	2560	0.5442
SVM	84.55	54840	0.9324
RF	82.70	185.49	0.5131
K-Means	76.45	61.92	0.5126
NB	81.50	140.46	0.3758

In this study, the feasibility for the development of a wearable armband was also investigated. A new dataset was acquired, and two additional gestures were added (pinch grip and open hand). The channels were brought up to seven, and the sEMG signal was acquired using pre-gelled electrodes (24 mm) and the *g.HIamp-Research amplifier* by g.tec.The previously tested algorithms were trained on the new dataset, and a final comparison was made.

Protocol for Dataset Acquisition

Because this dataset has been used for further investigation, the acquisition protocol is briefly reported. The campaign recruited 14 people (11 males and 3 females) performing eight gestures (i.e. seven active and the idle). After giving their consent for participation in the study and signing the informed consent drafted following the regulations of the local bio-ethical committee, the session started. The participants sat in a comfortable position, and the electrodes were carefully placed following the standardization previously studied. A calibration check was made to verify proper muscle activation on all channels. The subjects performed the gestures sequentially and the supervisor remained the gesture to be performed, setting a timer.

The acquisition protocol followed was:

- The acquisition starts in idle condition.
- The gesture to be performed is maintained for 30s.
- Each gesture is followed by a rest interval of 5s.
- After the execution of all the movements, a 30s rest is observed.
- This acquisition is repeated three times.

The table 2.6 below presents the statistical results obtained on the dataset and represents the starting point for the investigation of this thesis work.

Table 2.6: Comparison of the ML algorithm tested by Tolomei et al. on the new dataset

Classifier	Global Accuracy (%)
NN	80.30
SVM	78.10
RF	78.90
K-Means	35.70
GMM NB	70.90

Chapter 3

Further Investigation on 7 Channel Dataset

3.1 Offline training

A preliminary investigation of the most suitable Machine Learning algorithms has been conducted using the dataset acquired in a previous work [72] to develop a wearable armband for Gesture Recognition.

In order to be helpful for the classification process, the data acquired have to be processed. The elaboration of the data, performed in MATLAB, adds the output label depending on the movement performed, required by supervised classifiers, and cleans the data from possible noise or involuntary movements during the recordings. Instead of considering only data between the 5^{th} and the 95^{th} percentile, a threshold is set in this work. The routine defines as idle all the values below the designated threshold.

$$Th = \sqrt{f_1^2 + f_2^2 + f_3^2 + f_4^2 + f_5^2 + f_6^2 + f_7^2} \leq N \tag{3.1}$$

The user selects the N value, and f_i represents the TC values that constitute an acquisition set.

This implementation permits the maintenance of more border values, helpful in discriminating among different classes during the classification process. The algorithms' implementation takes advantage of the computational efficiency of MATLAB environment, exploiting the *Statistics and Machine Learning Toolbox*.

The training phase considers 100 initialization of the ML algorithms and uses K-fold cross-validation, with K = 5. Each fold preserves the percentage of examples for each class, applying a stratified K-fold method. For the parametric classifiers, accurate tuning of the hyperparameters is performed to determine the optimal algorithm for the data at hand. The input data are divided into train and validation sets and loaded separately, avoiding the occurrence of correlation between them. The data matrices are not standardized, having that the data are evenly distributed, and the seven features belong to the same range (0-32).

3.1.1 Neural Network

The Neural Network algorithm implementation exploits the *Deep Learning Toolbox* and uses the backpropagation method described in Section 1.4.1. The optimization of the network parameters is carried out by a routine that examines different architectures, considering 2 or 3 layers with an equal number of nodes ranging from 16 to 64. For each tested architecture, the learning rate is varied from 0.001 to 1. Adam (*Adaptive Moment Estimation*) optimizer is used during the training sessions, setting the maximum number of epochs to 500 and using a mini-batch with 1/6 of the total number of observations of the training dataset at each iteration. A validation patience setting stops the iterations when the accuracy of the classifier does not increase significantly. The conventional gradient descent algorithm updates the network weights and biases by minimizing the loss function by moving in the direction of the negative gradient computed on the loss with a small step, which size is dictated by the learning rate value. The five best performances and their related architectures are reported in the table below.

Table 3.1: Top 5 results for NN preliminary study.

Layers	Nodes	Learning rate (α)	Accuracy(%)
2	40	0.01	87.25
3	63	0.01	87.11
3	59	0.03	87.04
2	56	0.001	86.91
2	36	0.01	86.77

The selected network presents the following structure:

1. **Input layer**: the number of nodes in the input layer is equal to the number of acquisition channels of the system, 7 in this case.

2. **Hidden layers**: the network includes two equally sized fully connected hidden layers, each comprising 40 nodes. The *Rectified Linear Unit function* is used as an activation function to avoid divergence. It sets to zero all the negative input instances and maintain unaltered the positive ones, applying a threshold operation.

3. **Output layer**: in the output layer, each recognized gesture corresponds to a node. The *Softmax layer* used for the output applies the softmax function to its inputs. This function can also be referred to as the normalized exponential, and it acts as the logistic function for a multi-class case.

The Learning rate value is equal to 0.01.

3.1.2 Support Vector Machine

The Matlab function *fitcecoc* fits multiclass models for Support Vector Machines and is used to implement the classifier. An Error-Correcting Output Codes (ECOC) model simplifies the multi-class classification problems into a set of binary ones. In

particular, this function train a multiclass ECOC model using SVM binary learners. The number of binary learners is 8, having set the coding as one vs all and for each of them, one class is considered as positive while the other is negative. A new input is assigned to the specific class that minimizes the cost functions for each binary classifier. Also, in this case, a routine optimizes the hyperparameter for the classifier.

Table 3.2: Top 5 results in hyperparameter selection.

Kernel	Regularization parameter (C)	Gamma	Accuracy(%)
'*rbf*'	100	'*scale*'	87.10
'*rbf*'	1000	'*scale*'	86.66
'*rbf*'	10	'*scale*'	86.46
'*rbf*'	1	'*auto*'	86.43
'*sigmoid*'	10	'*auto*'	85.89

The best model's parameters are:

- **Kernel**: radial basis function (RBF)
- **Regularization parameter (C)**: '100'. It is a positive value, inversely proportional to the strength of regularization. As the value of C increases, the number of generated support vectors decrease.
- **Gamma**: 'scale'. In this condition, its value is equal to $\frac{1}{n_{features} * variance}$

3.1.3 Random Forest

Fitcensemble function deploys Random Forest models. The algorithm grows shallow trees, which depth can be controlled by specifying the maximum number of splits and the minimum leaf size. Each tree grows on a bootstrapped dataset, which resamples the original with replacement, maintaining the same size. The classification ensemble for the prediction is created by using all the features in the data. Then, the other trained ensembles will use fewer features. The output class is chosen trough majority voting, comparing the in-sample accuracies.

Table 3.3: Top 5 results in Parameter selection.

Tress	Parent Size	Accuracy(%)
100	30	87.02
80	35	86.97
30	15	86.88
90	15	86.47
30	40	86.32

The selected parameters for the Random Forest classifiers are:

- **Number of Trees**: '30', the number of trees generated in the forest.

- **Parent Size**: '15', the maximum number of branch nodes observations.

This model has not the highest performance but offers an appropriate trade-off between the accuracy and the contained number of learners.

3.1.4 K-Nearest Neighbour

For the implementation of the K-Nearest Neighbour algorithm, the *fitcknn* function is used. It returns a KNN classification model based on the input features. fitcknn creates a Kd-tree by default to find the K-nearest neighbours, using a radius search. The proximity metric selected is the Euclidean distance, and the number K of neighbours has been chosen equal to 15. Also, the distance weight is equal for all the examples contained in the training set.

Table 3.4: Preliminary analysis for KNN algorithm.

Neighbour number	Distance metric	Accuracy(%)
15	'*Euclidean*'	87.32
30	'*Minkowski*'	87.27
17	'*Chebychev*'	87.17
13	'*Minkowski*'	87.13
27	'*cityblock*'	87.02

3.1.5 Naive Bayes

Fitcnb can implement different types of multiclass Naive Bayes models. In particular, it is possible to choose among different data distributions like the *multinomial*, the *multivariate multinomial*, the *Gaussian* and the *Kernel smoothing density estimate*. All tested on the training data. The algorithm does not require hyperparameter tuning because it estimates the probability densities of the predictor within each class on the labelled data. The results for the examined distributions are reported in Table 3.5:

Table 3.5: Preliminary analysis for NB algorithm.

Distribution	Accuracy(%)
'kernel'	74.02
'mn'	73.78
'mvmn'	73.62
'normal'	70.88

3.1.6 Offline performance comparison

Table 3.6 offers a comparison among the performance of the tested algorithms.

The examined classifiers show comparable results at a good usability level. Only the NB classifier exhibits overall lower performances, resulting in its exclusion from the following phase of formulating an embedded algorithm for the MCU. Among NN, SVM, RF and KNN, the Neural Network offers the lowest power consumption and computational effort. However, in this preliminary phase, all the above classifiers have been deployed. For the firmware design, memory constraints and maximum latency of 300 ms must be guaranteed. This part is discussed in the following section.

Table 3.6: Offline performance comparison.

Algorithm	Accuracy(%)	Precision(%)	Recall(%)	F1-Score(%)
NN	87,25	82,23	82,40	82,28
RF	86,88	82,39	82,41	82,42
SVM	87,10	83,54	83,56	83,54
KNN	87,32	83,08	83,21	83,15
NB	74,02	69,38	70,01	69,26

3.2 Firmware for Online prediction

Here, only the firmware related to the ML implementation is treated.

- **Neural Network**

  The NN implementation applies feed-forward propagation using the weights obtained in the training session. The hidden layers employ the ReLu activation function, while the output layer the SoftMax function. The use of the CMSIS DSP library optimizes the matrices computation without using other external libraries.

- **K-Nearest Neighbour**

  K-Nearest Neighbour model did not necessitate any specialized library, being based on the Euclidean distance computation from the datapoint of the training set. Due to the constraint in the memory available, instance reduction was applied to reduce the dimension of the dataset.
  Instance Selection (IS) is a data-mining pre-processing technique that chooses a representative subset of instances from the available dataset while trying to achieve the slightest loss in performance possible. After applying different algorithms presented in the literature, DROP3 (Decremental Reduction Optimization Procedure) was selected as the one offering the best balance between accuracy and dimensionality reduction.
  DROP3 is a decremental algorithm that removes both central and border points [73]. The pseudocode is reported below.
  Briefly, the algorithm starts by applying a noise filter that removes all the instances in the training set misclassified by their K nearest neighbour. For this new subset of features, the lists of nearest neighbours and associates

are calculated. An instance X is removed in the main loop if the number of associates correctly classified without X is greater than the number of associates correctly classified with X.

Algorithm : Decremental Reduction Optimization Procedure 3 (DROP3)

Input: A training set $X = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$, the number of nearest neighbours k
Output: The set of selected instances $S \subseteq X$
Noise filtering: remove any instance in X misclassified by its k neighbours
$S = X$

foreach *instance* $\mathbf{x} \in S$ **do**
  Find $\mathbf{x}.N_{1...k+1}$, the $k+1$ nearest neighbours of $\mathbf{x}$ in S
  Add $\mathbf{x}$ to each of its neighbour's list of associates
foreach *instance* $\mathbf{x} \in S$ **do**
  Let `with` = # of associates of $\mathbf{x}$ correctly classified by $\mathbf{x}$ as a neighbour
  Let `without` = # of associates of $\mathbf{x}$ correctly classified without $\mathbf{x}$
  if `without` $\geq$ `with` **then**
    Remove $\mathbf{x}$ from S
    foreach *associate* $\mathbf{a}$ *of* $\mathbf{x}$ **do**
      Remove $\mathbf{x}$ from $\mathbf{a}$'s list of nearest neighbour
      Find a new nearest neighbour for $\mathbf{a}$
      Add $\mathbf{a}$ to its new neighbour's list of associates
return S

- **Support Vector Machine**

  The SVM algorithm was deployed with the use of the LIBSVM library [74]. The routine takes as input from the trained model the kernel matrix, the gamma value, the support vectors, the intercepts and the coefficients. This implementation required a *one vs one* approach, resulting in a total of 28 binary learners.

  However, the resulting number of support vectors (i.e. 17561) derived from the training does not satisfy the memory constraints of the MCU. In order to reduce the memory footprint of the SVM model, different implementations of the classifier were attempted. SVM is a kernel method that projects data points in a higher dimensional feature space where the original data are represented in a kernel matrix. This type of algorithm, requiring kernel matrices computation, presents high computational costs, at least quadratic in the number of training data. A possible solution is the use of low-rank matrix approximation methods. So, the first trial entailed using the Nystroem method, implemented in both Python and MATLAB. The Nystroem method is a general low-rank kernel approximation method that subsamples the data on which the kernel is computed. It can be applied to any kernel. Here it is used with the RBF kernel. The algorithm decreased the number of support vectors, bringing their number to 5514 and its performance, as shown in Table 3.7. This method was discarded due to the still high number of support vectors and the lowered performance achieved.

  The second implementation used the reduced dataset created for the kNN algorithm. Starting from a minor number of data points, the number of generated support vectors has also decreased, making the model suitable for the memory constraints of the MCU.

Table 3.7: Statistical results obtained with the Nystroem approximation method.

Accuracy(%)	Precision(%)	Recall(%)	F1-Score(%)
76,53	74,35	73,21	75,18

However, in order to use the entire original dataset, a sparse implementation was also realized, following the Newton Sparse SVM (NSSVM) algorithm presented in [75]. This method solves a sparsity constrained kernel-based SVM optimization problem:

$$min_{\alpha\in\mathbb{R}^m} \qquad D(\alpha) := d(\alpha) + \sum_{i=1}^{m} h_c c(\alpha_i), \qquad s.t. \sum_{i=1}^{m} \alpha_i y_i = 0, \qquad ||\alpha||_0 \leq s \tag{3.2}$$

Where the primal d(α) and the dual D(α), are respectively:

$$min_{\alpha\in\mathbb{R}^m} \quad d(\alpha) := \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i\alpha_j y_i y_j \langle x_i, x_j\rangle - \sum_{i=1}^{m}\alpha_i \tag{3.3}$$

$$s.t. \sum_{i=1}^{m} \alpha_i y_i = 0, \qquad 0 \leq \alpha_i \leq C, i \in [m] \tag{3.4}$$

$$min_{\alpha\in\mathbb{R}^m} \quad d(\alpha) := \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i\alpha_j y_i y_j \langle x_i, x_j\rangle - \sum_{i=1}^{m}\alpha_i \tag{3.5}$$

$$min_{\alpha\in\mathbb{R}^m} \qquad D(\alpha) := d(\alpha) + \sum_{i=1}^{m} h_c c(\alpha_i), \qquad s.t. \sum_{i=1}^{m} \alpha_i y_i = 0 \tag{3.6}$$

Here, $s \in [m]$ is an integer number $s \ll m$ called sparsity level which counts the number of non-zero elements of α ,and $||w||_0$ is the zero norm of α. For each binary classifier the value of the sparsity level has been tuned iteratively, in order to select its optimal value.

The Table below 3.8 shows the global accuracy and the number of support vectors obtained for the different implementations:

Table 3.8: Comparison among the method used for the reduction in the number of SV.

Method	Accuracy(%)	N° SV
Nystroem	76.53	5514
Reduced dataset	87.15	2357
NSSM	85.2	681

- **Random Forest** The Random Forest algorithm has been brought to the MCU thanks to the use of the *emleran* library [76]. This library allows converting a model trained model to a header file, used on the MCU. In order to reduce the occupied space in memory, the classifier hyperparameters have been changed diminishing the number of trees to 10. The maximum depth to which each tree can grow was limited to 10 and the number of split to 2. The performance of the classifier did not change significantly. In this way the memory constraints are satisfied and no additional changes had to be done.

Chapter 4

System description

The aim of the developed wearable armband is the real-time classification of hand gestures, using as input the ATC data extracted from the ElectroMyoGraphic surface (sEMG) signal. The embedded low-power system is based on an event-driven approach which reduces the complexity of the classification algorithms by transmitting a lower amount of data, hence diminishing the power consumption. This feature matches the requirements for a wearable battery-powered device. The proposed system comprises seven bipolar acquisition channels consisting of dry electrodes, each acquiring the sEMG signal and providing the extracted TC signal. The research group has developed a PCB with an Apollo3 Blue ultra-low power MicroController Unit (MCU) with an ARM Cortex M4F µP onboard, performing signal conditioning and the computation of the ATC data, fed to the embedded machine learning algorithms for the classification, and containing a Bluetooth Low Energy (BLE) module for data transfer. Figure 4.1 shows an overview of the system.

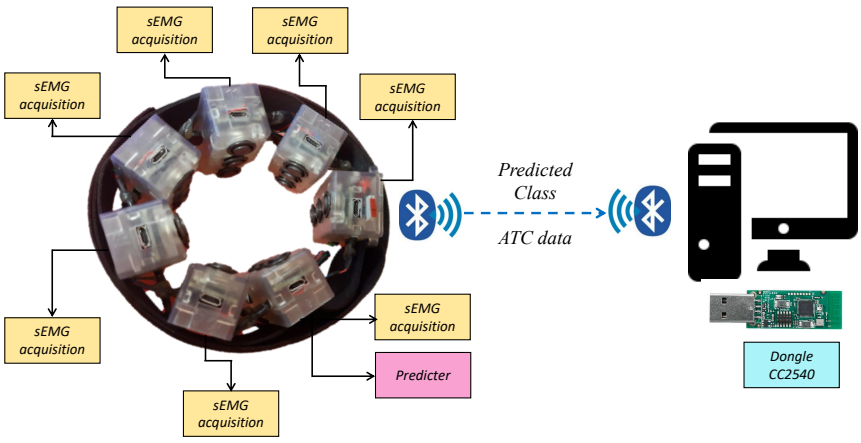
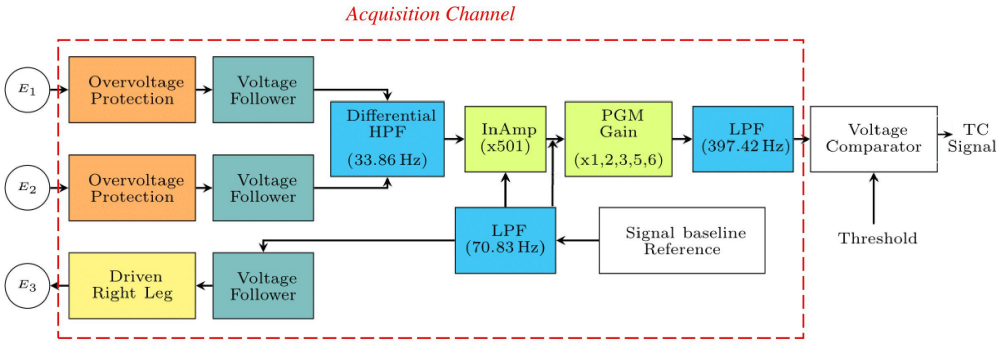


Figure 4.1: System overview.

4.1 Acquisition system

The superficial ElectroMyoGraphic acquisition system consists of the Analog Front End (AFE) channels and their digital control unit. The AFEs are coupled with the Apollo3 Blue MicroController Unit (MCU) equipped with a Bluetooth Low Energy antenna. The main steps of the EMG signal conditioning and the ATC computations are reported in the schematic below.



The electrodes, used in a single-differential sampling configuration, are connected to an overvoltage protection circuit, protecting voltage-sensitive PCB components from the accumulation of electrostatic discharges and transient voltage events. Each of them is linked to a Voltage Follower, introduced for the decoupling of the electrode-skin impedance. Their positioning in the system ensures the collection of the same signals from the electrodes. The reference electrode is linked to the Driven Right Leg (DRL) circuit for reducing the common mode voltage. A passive differential high pass filter with a cut-off frequency of 33.86 Hz is applied to the signal exiting the voltage follower to reduce motion artefacts. The instrumentation amplifier reduces the electromagnetic noise, providing a gain of 501 V/V, thus improving the input impedance. A second-order multiple feedback low pass active filter (cut-off frequency of 70.83 Hz) efficiently rejects the common mode, and an additional adjustable gain allow for further amplification of the signal. Another filter, a second-order Sallen-Key low pass active filter (cut-off frequency of 397.42 Hz), limits the upper bandwidth of the signal. Finally, a voltage comparator is set to a high logical level when the signal is higher than a set threshold, low otherwise, outputting the TC signal, transmitted to the Microcontroller. The ATC parameter is extracted onboard and directly sent through the Bluetooth antenna to the computer.

Each PCB presents onboard four different peripherals: SPI, I^2C slave, I^2C master and USB. The I^2C communication has been chosen for the information exchange among the various units composing the armband. I^2C (Inter integrated circuit) is a bus interface used for communication between a master (or multiple masters) and one or more slaves. This connection utilizes only two wires to establish connections, reducing the number of connectors used in the armband assembly. Each device on the I^2C bus has a specific address differentiating it from other devices present on the same bus. A slave cannot transmit data unless the master has addressed it. I^2C protocol uses two bidirectional open-drain pins, the serial data (SDA) and the serial clock (SCK), for data communication [77]. Both lines are connected to the

V_{CC} through a pull-up resistor present onboard, and which resistance value can be selected depending on the amount of capacitance on the I^2C lines. The value of this resistance will affect the velocity at which the lines are brought to a high level. Data transfer starts when the bus is in idle condition, having both the SDA and the SCK lines high. One data bit is transmitted during each clock pulse of the SCL. A byte representing the device address first and a byte of data written to or read from a slave after is sent during the communication.

The classification and the Low Energy Bluetooth routines have been loaded over different PCBs, the former on the slave unit preceding the master and the latter on the master. Each unit collects the sEMG signal and performs the signal conditioning and the TC signal extraction onboard. After that, the ATC parameter is computed for each unit and transmitted to the next sequentially until all the ATC data reaches the Master. After that, if classification is requested, the Master passes the ATC data to the Slave designated for the prediction. This unit outputs the class label of the executed gesture, passing it back to the Master responsible for the Bluetooth Low-Energy transmission to a computer. This transmission method, employing the I^2C, takes the name of *Daisy chain*, being the devices wired together forming a ring. An additional advantage of this system is the possibility to use different units as masters or changing the predicter designated unit.

The use of dry electrodes is essential for the usability of a wearable device and has been thoroughly investigated in a previous work [78]. The comparison between dry and wet electrodes has been conducted using a stand-alone channel similar to the ones exploited for the armband deployment. The metric used for the comparison has been the Signal to Noise ratio (SNR), evaluated on the sEMG signals extracted from the upper and lower limbs. The results obtained showed comparable values of the SNR for the two electrodes used, and in some cases, higher ratios for the dry ones, as shown in Table 4.1.

Table 4.1: SNR_{dB} comparison between wet and dry electrodes. Wrist extension is referred to the signal acquired from the extensor carpi radialis, while forearm flexion to the signal collected from the brachial bicep

Movement	Intensity	Wet electrodes SNR (dB)	Dry electrodes SNR (dB)
Wrist extension	minimal	13.46	10.74
	normal	20.38	21.85
	maximal	25.28	27.16
Forearm flexion	minimal	13.73	15.08
	normal	21.17	21.70
	maximal	27.41	26.42

These outcomes made them suitable for implementation in the current system. Each channel contains three dry electrodes. The two lateral ones are the active electrodes, acquiring the sEMG signal, and the central one is the reference electrode, used in a bipolar configuration. The system is pressed to the subjects skins to

meliorate the quality of the acquired signal, reducing the otherwise high electrode-skin impedance characterizing dry electrodes without creating discomfort. The signal conditioning performed onboard further improve the quality of the acquired sEMG signal.

4.2 Armband model

As a consequent step in the realization of the armband, a 3D model was developed, taking inspiration from the model for the previous PCB. The model was designed in the CAD environment of Fusion 360® and 3D printed with Clear Resin using the FormLab Form 3D printer, offering a resolution of 1 µm. The armband has seen the realization of two different cases: one model for the master unit and the other for the six slave channels due to the difference in the components contained in the two units. The models have been designed to maximally reduce the dimensions for the components that the channels have to hold, improving the wearability of the device and accommodating different forearm sizes. The measurement of the various constituents are reported in Figure 4.2.

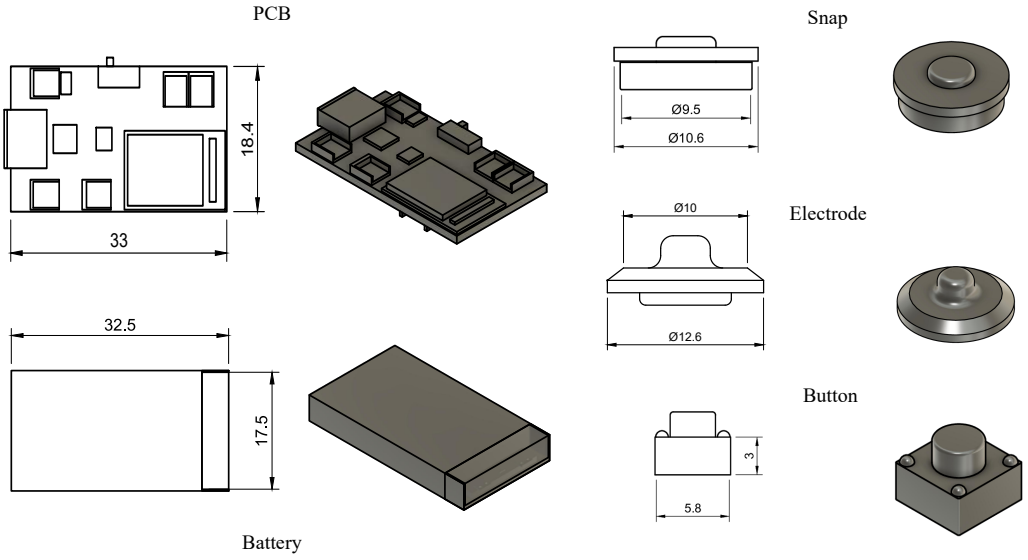


Figure 4.2: Measures of the Armband channels components.

Concerning the Slave modules, the case contains the three dry electrodes used in a bipolar configuration to collect the sEMG signal of the forearm. Together with their sockets, the electrodes are maintained in position thanks to three stoppers that rotate clockwise to lock them in place. The PCB is located above them, and kept in place thanks to two lateral protrusions over it. The cover, closing the unit, permits the passage of an elastic band used to adjust the armband at the desired measure with Velcro, to fit different forearm sizes. The cover closes the box, constituting a single channel of the system, with a sliding mechanism. A reset button is also inserted for each unit, making it possible to reset the MCU. The thickness of the walls is equal to 2 mm, and the inter-electrode distance is 16 mm. The lateral holes

in the module serve for the passage of the connectors that bring the four wires for the common ground, common power supply and two for the I^2C communication.

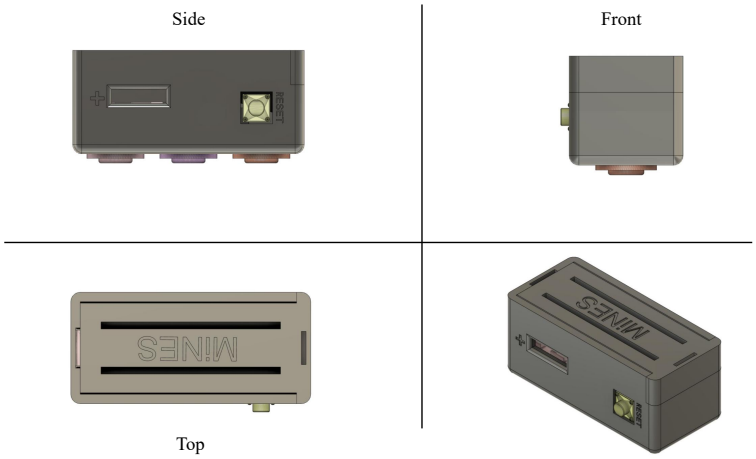


Figure 4.3: Slave module.

The Master module is similar to the slave ones, with few differences. The case is larger for the accommodation of the rechargeable Li-Po battery, that alone powers the entire device, characterized by a rated voltage of 3.7 V and capacity of 175 mA h. It also has two lateral supports used for the passage of the elastic band, which is sewn on one side and passed through the other to consent the adjustment of the armband to the forearm circumference of the subject. The difference in the stopping mechanism for the elastic band permits to have the same height for all the units. In this case, in addition to the reset button, a custom button is added to control the master functionality. The two additional openings in the module are used for the passage of two switches, one that control the powering of the master module, the other the slave ones. Also in this case, the wall thickness is 2 mm, and the inter-electrode distance is 16 mm.

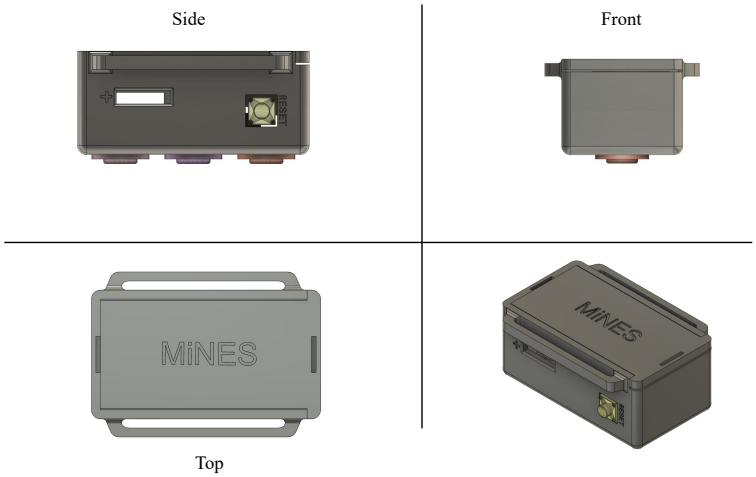


Figure 4.4: Master module.

The components constituting a unit therefore are:

- **Case**: recipient for the electronic components.
- **Cover**: closing part, that accommodate the elastic band for the armband fixing.
- **Socket and electrodes**: three dry electrodes making a clip connection with the sockets.
- **Stopper**: 3 different stoppers containing the sockets for the electrodes. The opening present on the top of the stoppers is added to facilitate their positioning using a screwdriver. The third stopper has also a protrusion that keep the PCB in place when the USB connector is inserted.
- **PCB**: fixed in place thanks to the stopper support and the lateral hooks.
- **Reset button**: used to reset the MCU present on the PCB.
- **Custom button**: available for user interaction (master only).
- **Battery**: rechargeable Li-Po battery with voltage of (master only) of 3.7 V and capacity of 175 mA h (Cellevia Batteries: model LP421730).

Wires are used to connect the sockets, the battery and the buttons to the PCB.

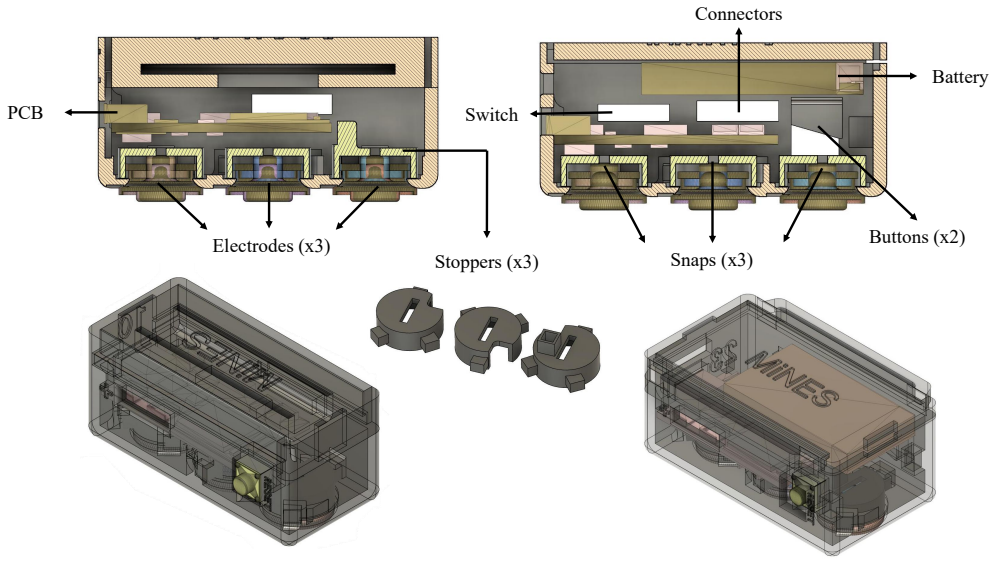


Figure 4.5: Components inside the Armband units. On the right the master module, on the left the slave one.

The width of a single slave module is equal to 23.5 mm, while the width of the master unit is 28.5 mm, bringing the total length of the armband to 17.75 cm, adjustable using the elastic band to higher values. In accord with the average forearm circumferences (Table 4.2), the armband is appropriate to suit a wide range of subjects, without problems of anorexia or obesity.

Table 4.2: Forearm circumferences.

Size	Circumference (cm)
Small	18-22
Medium	22.28
Large	28-36

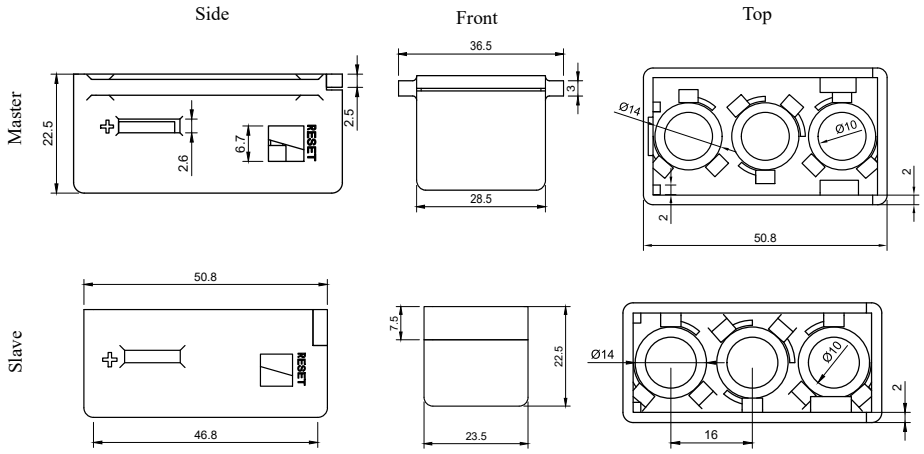


Figure 4.6: Measures of the Master and Slave modules.

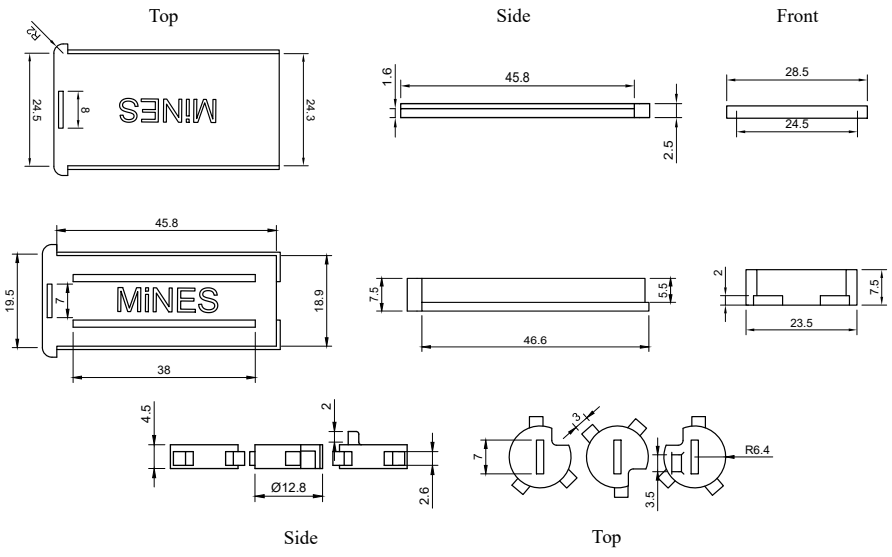


Figure 4.7: Measures of the covers of the Master and Slave modules and of the stoppers.

Chapter 5

System Validation: Data Acquisition and Classification Algorithms Deployment

5.1 Armband Positioning and Performed Gestures

Correct positioning of the channels is essential to obtain good quality of the recorded signal, avoiding noise and maintaining morphological identity. The armband positioning follows the standardization studied in [72], with the first channel, coinciding with the master, placed on the *Extensor Digitorum* muscle and the armband positioned at around 1/3 of the total forearm length, starting from the elbow. The other channels follow the undermentioned disposition and are equally spaced, starting from the medial section.

- **CH1** – Extensor digitorum/Extensor digiti minimi
- **CH2** – Extensor carpi radialis brevis
- **CH3** – Brachioradialis
- **CH4** – Flexor carpi radialis
- **CH5** – Flexor carpi ulnaris
- **CH6** – Extensor carpi ulnaris
- **CH7** – Flexor digitorum profondus

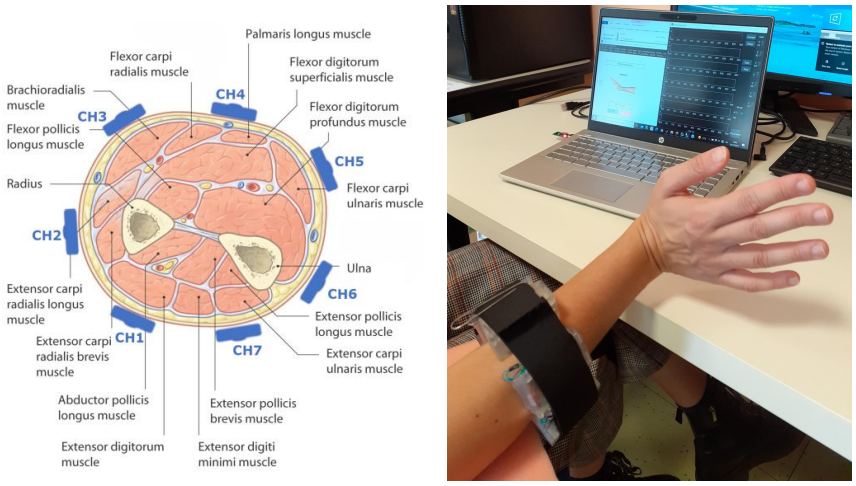


Figure 5.1: Armband's electrode disposition referenced to the forearm muscles. On the left, the forearm section from distal to proximal.On the right, armband positioning during an acquisition session.

With this disposition, each channel approximately acquire the signal deriving from the muscle below it, giving information about their activation during the execution of the gestures.

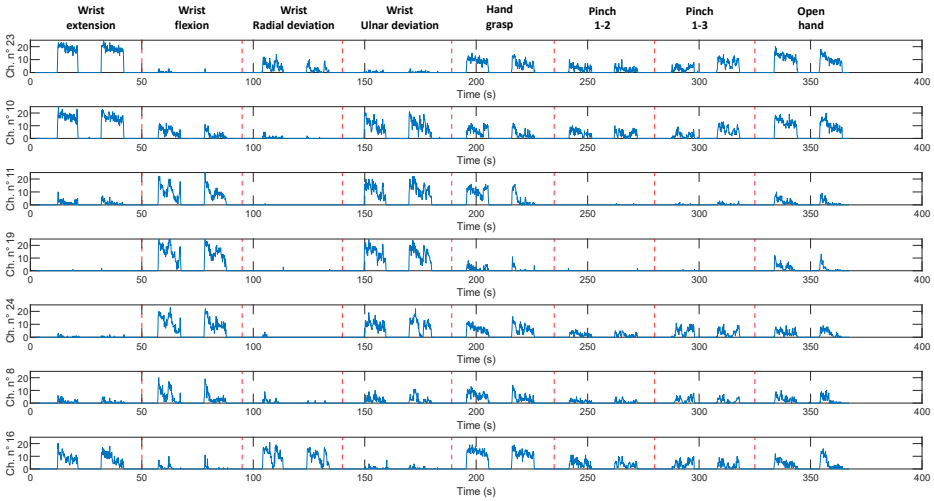


Figure 5.2: ATC parameter over the 7 channels of the Armband during an acquisition.Muscles activation investigated by each channel: **CH1**: Extensor digitorum/Extensor digiti minimi, **CH2**: Extensor carpi radialis brevis, **CH3**: Brachioradialis, **CH4**: Flexor carpi radialis, **CH5**: Flexor carpi ulnaris, **CH6**: Extensor carpi ulnaris, **CH7**: Flexor digitorum profondus.

Figure 5.2 shows the ATC data acquired with the armband during a training session, showing little to no noise over all the channels, even with the use of dry electrodes.

The acquisition phase considered eight active movements and the idle position. The additional gesture introduced, is the pinch 1-3, which has been difficult to recognize and to classify with respect to the pinch 1-2, due to similarity in the activation pattern. This work aims to discern the two gestures.

- *Wrist extension*: the hand back is moved closer to the distal forearm causing the activation of the *extensor carpi radialis longus*, the *extensor carpi radialis brevis* and the *extensor carpi ulnaris* and of other deep muscles.

Figure 5.3: Wrist extension.

- *Wrist flexion*: is the movement of the hand palm towards the inner part of the forearm. The mainly used muscles are *palmaris longus*, *flexor carpi radialis*, and *flexor carpi ulnaris*, and also *digitorum superficialis* and *profundus*.

Figure 5.4: Wrist flexion.

- *Wrist radial deviation*: the hand is moved upwards in the thumb direction. *Abductor pollicis longus*, *flexor carpi radialis*, *extensor carpi radialis longus* and

Figure 5.5: Wrist radial deviation.

brevis are activated for this movement.

- *Wrist ulnar deviation*: the hand is brought down, in the derection of the little finger involving the *extensor carpi ulnaris* and the *flexor carpi ulnaris*.

Figure 5.6: Wrist ulnar deviation.

- *Hand grasp*: the fingers are broight into the palm in a fist. The muscles activated include the *flexor digitorum* and *palmaris longus* primarly, alongside other muscles of the hand.

Figure 5.7: Hand grasp.

- *Pinch 1-2*: precision grip in which the index finger touches the palmar surface of the thumb. The muscles used in the movement are the *flexor digitorum superficialis*, *flexor digitorum profundus*, *palmaris longus*, and *flexor pollicis brevis* coupled with the adducting force of the *adductor pollicis*.

Figure 5.8: Pinch 1-2.

- *Pinch 1-3*: another precision grip in which the thumb and the middle finger touches. The activated muscles are *flexor carpi radialis*, *flexor digitorum profundus* ans *superficialis*, *palmaris longus* and the *adductor pollicis*.

Figure 5.9: Pinch 1-3.

- *Open hand*: the fingers are fully extended and the palm is open. Involve the activation of all the forearm muscles, mainly the *flexor carpi radialis*.

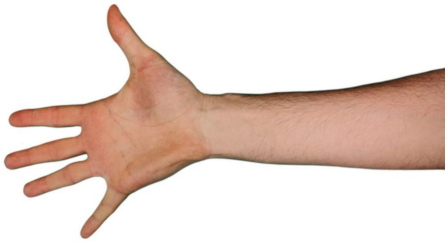


Figure 5.10: Open hand.

- *Idle*: rest position in which all the muscle are at rest maintaining the hand in a steady position.

Figure 5.11: Idle.

5.2 Acquisition protocol

The validation phase entailed the launch of an in vivo experimentation for the acquisition of a new dataset, on which the classifiers will be trained. For the collection of the data 25 people has been enrolled, 10 females and 15 males, with an age ranging between 23 and 29. Before the start of the acquisition sessions, each volunteer received exhaustive information about the experimental protocol and signed the informed consent describing the details of the study, drafted in accord to the bioethical committee regulations.

The acquisition phase comprised two stages with the participation of different subjects. For the first phase, 20 people taken part in the collection of the new

dataset for the training of the algorithms. The second phase consisted in the online validation of the system and 5 people participated in it. The procedure followed in the two sessions were slightly different, as described below. The two sessions have been performed in different days, without trying to replicate environmental conditions, ensuring the independence of the training and testing phase.

The participants sat in a comfortable position with their arm at rest on a plain surface. After accommodating the armband on the volunteer arm, regulating its diameter and finding the right position as indicated in the section above, the session began. The acquisition started with an initial calibration phase for the ATC threshold setting, while the subjects maintains a rest condition. The quality of the signal is checked in order to assess the correctness of the armband positioning and the absence of noise from the channels. If these conditions are not satisfied, the calibration phase is repeated or the gain for the single channels is regulated, through the acquisition GUI (Figure 5.12), in order to improve the signal quality. After that, the subject is asked to perform sequentially a set of eight active gestures, each repeated twice, spaced out by a rest interval. The order in which the movements are performed is kept the same throughout the acquisition for all the subjects. The subject is instructed on the movements to be performed visually.

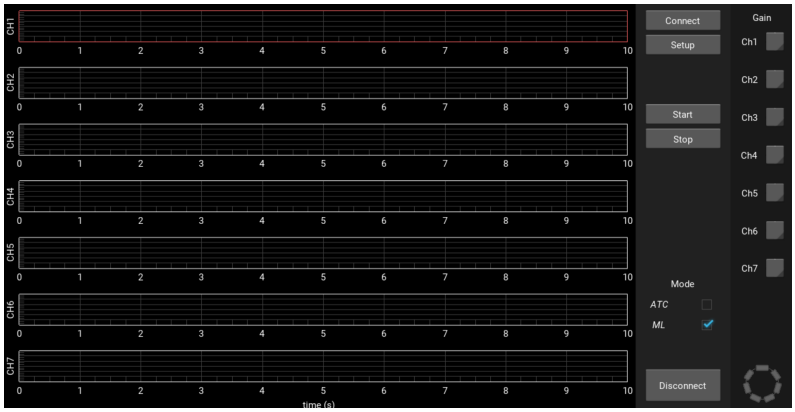


Figure 5.12: GUI developed in Python by the research group used for the training and testing phase for the acquisition of the signal.

Training acquisition protocol

The acquisition protocol steps followed in the training phase are:

1. The subject execute one movement at a time starting from the rest condition, kept for 10 s. During the acquisition period, a total number of 77 ATC values are acquired, being the time window of 130 ms.
2. Each gesture is executed twice and maintained for 10 s, and repeated after a rest time of 10 s, introduced in order to avoid muscle fatigue onset.
3. A rest interval of 15 s is observed and after that the following movement is performed.
4. Step 2-3 are repeated until all the eight active gestures are completed. At the end of the first repetition, a rest period of 60 s is maintained.

5. Three repetitions are performed, in order to acquire a sufficient amount of data.

The obtained data are saved in a .txt file and process in MATLAB environment for the labelling required by the supervised classifiers. Also for this dataset, a threshold for the idle gesture was set. Every ATC set, considered over the seven channels, below the threshold is defined as idle.

Testing acquisition protocol

The testing phase was performed after the training of the classifiers over the training set previously acquired. It consisted in executing the eight gestures, each performed three consecutive times and maintained for 10 s. The movements were spaced out by a rest period of 10 s. The acquisition was performed only once for each volunteer. The supervisor reminded the subject of the movement to be performed.

1. The session starts with the setting of a timer while the subject is in a rest condition.
2. The current gesture is executed and maintained for a period of 10 s.
3. The execution of an active movement is followed by a rest of equal time (i.e. 10 s).
4. The same gesture is repeated twice.
5. The next movement is initiated after observing a rest of 15 s.
6. The acquisition ends with the completion of all the active gestures.

The data are saved through the GUI in a .txt file and used to evaluate the performance of the classifiers.

5.3 Offline Training

The obtained dataset was used for the training of the previously presented machine learning algorithms, with the same training settings. Hyperparameter optimization was conducted again for the parametric classifiers using custom routines.

In this training phase, the classifiers were fed with two different types of inputs. As previously implemented, the first models were trained over the single ATC values from the seven channels, constituting an acquisition set. The second ones were obtained using input windows of different lengths, containing a variable number of ATC values ranging from 2 to 10, from each acquisition channel. This implementation was tested to evaluate if the ML algorithms show improvements by using profiles that give information about the evolution of the gestures. It is expected an improvement in classification rate, especially during a gesture's starting and ending phase. This second implementation was tested only in the offline phase, while the hyperparameter tuning was performed in both cases. The first models were then implemented for the online testing.

5.3.1 Neural Network

The NN architecture was defined with the same custom routine over 1000 epochs of training. The best architecture for the single ATC set is the following:

- **Input layer**: the input layer has 7 nodes, one for each acquisition channel.
- **Hidden layers**: 2 hidden layers, each one comprising 38 nodes. The used activation function is the ReLu.
- **Output layer**: in this case, the number of nodes is equal to 9, as the number of gestures to be recognized. SoftMax activation function is applied to each node.

The net was trained using Adam optimizer and the selected learning rate is 0.01.

Table 5.1: Statistical analysis for the Neural Network over an acquisition set.

	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
EX	97.93	81.95	85.57	83.72
FL	99.06	92.30	91.76	92.03
RD	98.15	86.87	82.40	84.58
UD	98.73	90.64	86.65	88.60
GR	98.73	90.64	86.65	88.60
P1-2	95.54	55.55	51.54	53.44
P1-3	96.57	68.05	65.58	66.79
OH	97.38	76.55	76.97	76.76
ID	100	100	100	100
Avg.	97.74	79.28	79.74	79.29

The ATC profiles are the input for the second model. Window length was set equal to 6 for the Neural Network. Also, in this case, the model architecture was selected for each window, as the one with the highest performances.

- **Input layer**: the input layer has 42 nodes, one for each acquisition channel multiplied for the number of ATC features considered, 6 in this case.
- **Hidden layers**: 2 hidden layers, comprising 40 nodes with ReLu activation function.
- **Output layer**: 9 output nodes, one for each output with a SoftMax activation function.

The learning rate is equal to 0.03.

The classifier's performance tested over the different windows length (i.e. 2-10) is reported in Figure 5.14. The accuracy increased by increasing the window length up to the maximum length of the examined profile, showing an exponential trend. Considering the MCU code implementation constraints, a window containing 6 ATC values was chosen for the Neural Network, reaching an average accuracy of 98.91%.

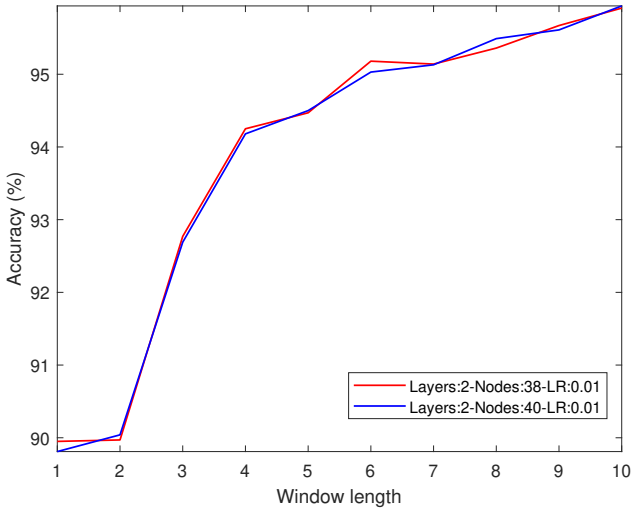


Figure 5.13: NN performance over ATC profiles of different length.

Table 5.2 shows how the error rate decreased for all the movements, improving the overall performances of the classifier.

Table 5.2: Statistical analysis for the Neural Network over a profile of 6 ATC values.

	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
EX	99.07	90.80	94.59	92.66
FL	99.52	96.66	95.11	95.88
RD	97.98	80.21	82.56	81.37
UD	99.16	94.06	92.26	93.15
GR	99.52	96.47	94.99	95.73
P1-2	97.77	79.22	74.65	76.87
P1-3	98.23	82.62	83.88	83.24
OH	98.95	89.91	91.50	90.70
ID	98.91	89.95	89.99	89.95
Avg.	98.91	89.95	89.99	89.95

Wrist extension, *Wrist flexion*, *Ulnar deviation* and *Hand Grasp* are the gestures that have the highest accuracy, while Radial deviation and the *Pinches 1-2* and *1-3* have lesser performances.

5.3.2 Support Vector Machine

Support Vector Machine model was obtained performing hyperparameter optimization. For the single ATC set, the best classifier parameters were an *rbf Kernel*, a regularization parameter C equal to 100 and *Gamma* set as 'scale'. The obtained statistical results are presented in Table 5.3.

Table 5.3: Statistical analysis for the Support Vector Machine over an acquisition set.

	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
EX	97.99	82.23	86.37	84.25
FL	99.04	93.51	91.83	92.66
RD	96.46	65.15	72.55	68.65
UD	98.06	85.58	82.34	83.93
GR	98.71	89.02	88.25	88.64
P1-2	95.62	56.83	49.54	52.93
P1-3	96.44	65.97	66.45	66.21
OH	97.52	78.37	77.04	77.70
ID	99.84	99.70	100	99.85
Avg.	97.75	79.60	79.37	79.42

For the ATC profile implementation, the selected window length was 4, offering a good compromise between the number of consecutive values considered and the obtained performances. Also, in this case, the kernel used was the *rbf*, while the regularization parameter C was set equal to 10.

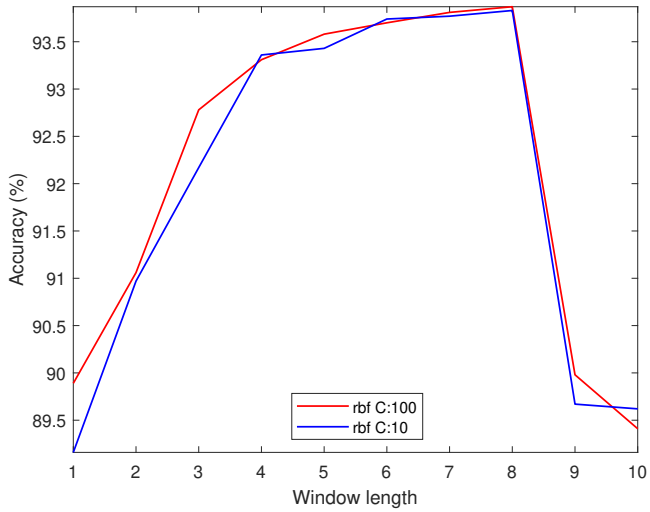


Figure 5.14: SVM performance over ATC profiles of different length.

The performance of the SVM shows a different behaviour compared to that of the NN. The accuracy of the classifier starts to significantly increase only at a profile of 4 values and up to a window length of 8 ATC features and, after it, considerably decreases.

Table 5.4: Statistical analysis for the Support Vector Machine over a profile of 4 ATC values.

	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
EX	98.72	89.16	90.36	89.76
FL	99.33	94.51	94.14	94.32
RD	97.47	74.40	80.14	77.17
UD	98.79	91.69	88.28	89.96
GR	99.29	93.67	93.86	93.76
P1-2	97.20	72.87	69.69	71.24
P1-3	97.83	80.56	77.40	78.95
OH	98.45	84.80	88.32	86.52
ID	99.87	99.94	99.82	99.88
Avg.	98.55	86.84	86.89	86.64

5.3.3 Random Forest

Random Forest classifier required hyperparameter optimization. The model selected in the first training settings (i.e. single ATC acquisition set) was:

- **Number of Trees**: '50', number of trees generated in the forest.
- **Parent Size**: '15', maximum number of branch nodes observations.

Performance are shown in Table 5.5

Table 5.5: Statistical analysis for the Random Forest classifier over an acquisition set.

	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
EX	98.06	83.90	85.08	84.49
FL	98.97	91.16	91.57	91.36
RD	96.47	65.02	73.43	68.97
UD	98.01	85.83	81.09	83.39
GR	98.73	90.94	86.45	88.64
P1-2	95.43	54.07	53.53	53.80
P1-3	96.58	67.57	67.03	67.30
OH	97.31	76.49	75.14	75.81
ID	100	99.99	100	100
Avg.	97.72	79.44	79.25	79.03

For the profiles, a window length of 4 consecutive ATC values was selected. With this input, the best parameters for the algorithm are:

- **Number of Trees**: '90', trees.
- **Parent Size**: '10', branches.

Table 5.6: Statistical analysis for the Random Forest classifier over a profile of 4 ATC values.

	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
EX	98.73	89.66	89.93	89.79
FL	99.28	93.39	94.59	93.99
RD	97.28	72.48	79.00	75.60
UD	98.59	90.59	85.99	88.23
GR	99.29	96.00	91.39	93.64
P1-2	96.91	70.98	64.01	67.31
P1-3	97.56	74.97	80.31	77.54
OH	98.43	85.29	87.02	86.15
ID	99.99	99.98	100	99.99
Avg.	98.45	85.93	85.80	85.80

The highest results are obtained for profiles between 3 and 7 consecutive ATC data. After, the accuracy decreases, becoming lower than the initial one. The length was selected equal to 4, considering the computational costs of a high dimensional dataset and the slight increase in classification performance.

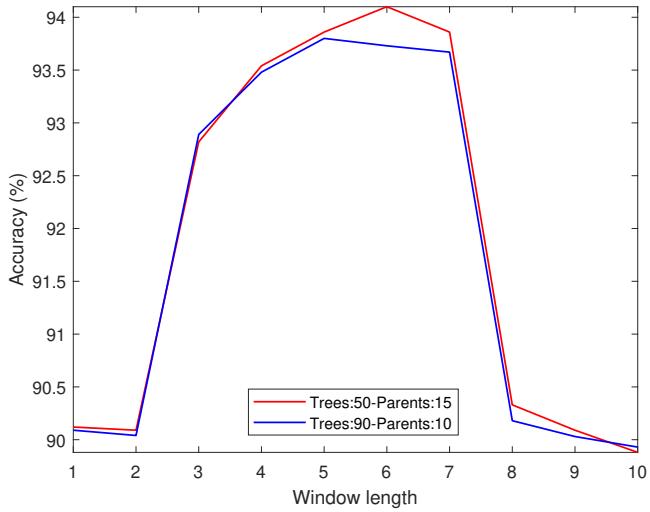


Figure 5.15: RF accuracy over profiles of different length.

5.3.4 K-Nearest Neighbour

The value K for the KNN classifier, was selected using cross validation. For the single acquisition set, K was equal to 17 while the similarity metric used was the *Euclidean* distance.

Table 5.7: Statistical analysis for the K-Nearest Neighbour classifier over an acquisition set.

	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
EX	97.61	77.81	86.07	81.74
FL	98.92	91.66	89.89	90.76
RD	95.88	59.13	74.50	65.93
UD	97.87	82.83	82.51	82.67
GR	98.71	90.17	86,98	88.55
P1-2	95.25	52.52	47.12	49.68
P1-3	96.62	69.66	63.08	66.21
OH	97.24	79.11	69.23	73.84
ID	100	100	100	100
Avg.	97.57	78.10	77.71	78.10

In the Figure 5.16, it can be seen that also for the KNN the performances increased with the number of ATC values considered inside the window, reaching its maximum value at 8. After that, the performance of the classifier started to decrease. In this case, an ATC profile of 5 data was considered.

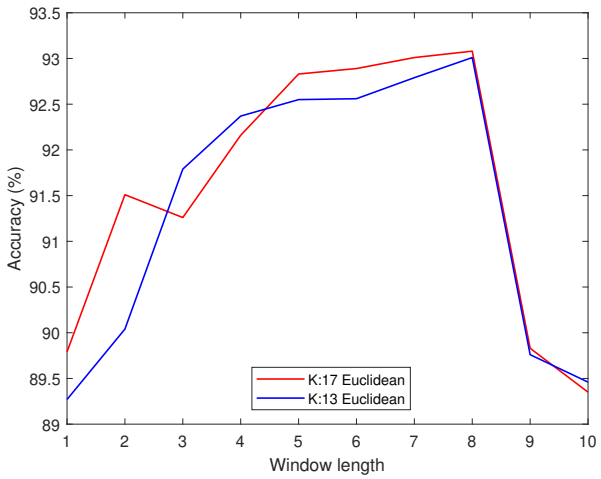


Figure 5.16: KNN accuracy over profiles of different length.

The number of neighbour was set equal to 13 and the distance used was the *Euclidean*. Statistical results are reported in Table 5.8.

Table 5.8: Statistical analysis for the K-Nearest Neighbour classifier over a profile of 5 ATC values.

	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
EX	98.59	87.85	89.68	88.75
FL	99.35	95.17	93.76	94.46
RD	96.95	68.29	80.14	73.74
UD	98.74	90.99	88.28	89.62
GR	99.29	96.46	90.92	93.61
P1-2	96.63	65.74	67.20	66.46
P1-3	97.49	77.59	73.42	75.45
OH	98.07	85.50	78.94	82.09
ID	99.72	99.61	99.87	99.74
Avg.	98.31	85.24	84.69	84.88

5.4 Offline Performance Comparison

Table 5.9: Comparison among all the tested classifier in the offline training phase.

	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
NN	97.74	79.28	79.74	79.29
RF	97.72	79.44	79.25	79.03
SVM	97.75	79.60	79.37	79.42
KNN	97.57	78.10	77.71	78.10
NN (6)	98.91	89.95	89.99	89.95
RF (4)	98.45	85.93	85.80	85.80
SVM (4)	98.55	86.84	86.89	86.64
KNN (5)	98.31	85.24	84.69	84.88

Table 5.9 reports the obtained statistical results of the offline training phase, considering both the single acquisition set and the profile set implementations.

- When using as input a single ATC value, the results for all the algorithms are comparable, with the SVM having the highest accuracy, with a difference of only 0.01% from the NN and of 0.02% from the RF. KNN offers slightly lower results.

- The use of a window containing multiple consecutive ATC values caused an increase in the performance of all the tested classifiers, particularly in terms of *Precision*, *Recall* and *F1-Score*. This solution offers a more robust gesture recognition ability for the classifier that presents lower misclassification rates for all the classes. By using several consecutive ATC values as input, the algorithms can also consider the evolution of each gesture over time which can lead to an increase in the final classification rates.

- In this implementation, the NN classifier offers the highest accuracy, while the KNN the lowest.

Chapter 6

Experimental results

To complete the validation of the system, the classifiers online accuracy, system latency and power consumption were analysed and compared with existent sEMG-based armbands.

6.1 Online classifiers performance

The aforementioned classifiers were deployed on the MCU, using the libraries and the algorithms presented in 3.1.6. In this work, only the classifiers trained with the single ATC acquisition set were tested online. The classifiers accuracy was measured after the testing phase conducted over five volunteers. The acquisition protocol was followed and repeated for each classifier.

Table 6.1: Statistical Results for the Neural Network.

	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
EX	98.55	93.31	93.81	93.56
FL	98.97	92.25	99.19	95.59
RD	87.53	46.12	69.16	55.34
UD	96.45	78.86	93.27	85.46
GR	98.16	90.88	92.22	91.74
P1-2	89.27	58.49	33.42	42.53
P1-3	91.64	59.46	73.55	65.76
OH	90.17	71.88	51.91	60.28
ID	100	100	100	100
Avg.	94.60	76.81	78.55	76.69

The Neural Network presents the highest recognition rate for the *Wrist Extension*, *Wrist Flexion*, *Ulnar Deviation* and *Power Grip*. *Radial Deviation* and *Pinch 1-2* have low *Precision* and *Recall* because the algorithm tends to classify one for the other due to a similar muscle activation pattern. *Pinch 1-3* offers better accuracy for this classifier.

Table 6.2: Statistical Results for the Random Forest.

	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
EX	96.93	82.25	92.47	87.19
FL	97.09	81.65	95.69	88.12
RD	90.43	68.92	67.74	68.32
UD	97.63	91.29	87.36	89.28
GR	96.69	94.27	75.26	83.70
P1-2	89.06	61.00	47.62	53.48
P1-3	90.16	59.02	58.87	58.95
OH	92.84	74.28	55.91	63.38
ID	100	100	100	100
Avg.	94.54	79.22	75.66	76.98

From Table 6.2 it is possible to observe that the worst-performing movements for the Random Forest are *Pinch 1-2* and *Pinch 1-3*. *Radial Deviation*, instead, shows higher accuracy compared to the Neural Network.

Table 6.3: Statistical Results for the Support Vector Machine.

	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
EX	95.02	69.93	96.91	81.24
FL	96.32	75.53	99.07	85.71
RD	86.15	37.29	35.80	36.53
UD	97.28	89.39	85.80	87.56
GR	92.95	80.83	48.14	60.35
P1-2	91.48	80.50	29.68	43.37
P1-3	88.45	48.61	70.27	57.46
OH	93.23	70.29	67.90	69.07
ID	99.99	100	99.99	99.99
Avg.	93.42	70.40	75.66	69.03

Support Vector Machine algorithm presents overall lower performances, also misclassifying the Idle state. In the online training phase, its accuracy was comparable to that of the other algorithms. The lower performances can be related to the sparse implementation adopted to reduce the number of support vectors, which slightly compromises the classifier's ability to classify the gestures correctly.

K-Nearest Neighbour shows the lowest statistical results among the tested classifiers. In particular, the algorithm adopted to perform instance reduction radically decreased the number of examples representing the seventh class, causing the almost absent classification of this gesture, as can be seen from the *Recall* score.

Table 6.4: Statistical Results for the K-Nearest Neighbour.

	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
EX	94.71	68.70	96.77	80.35
FL	97.17	83.57	93.01	88.04
RD	86.23	41.46	56.18	47.71
UD	96.54	80.96	90.32	85.38
GR	94.98	84.05	68.01	75.18
P1-2	84.53	42.21	39.68	40.91
P1-3	87.56	46.78	39.89	43.06
OH	89.18	92.30	0.03	0.07
ID	100	100	99.99	99.99
Avg.	92.31	71.12	65.23	62.99

Table 6.5: Online performance comparison.

	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
NN	94.60	76.81	78.55	76.69
RF	94.54	79.22	75.66	76.98
SVM	93.42	70.40	75.66	69.03
KNN	92.31	71.12	65.23	62.99

Table 6.5 compares the statistical results obtained during the online testing phase for all the classifiers.

6.2 System Latency

System latency is a fundamental parameter in real-time systems when deciding the best classification algorithm. It was measured exploiting the Apollo 3 Blue MCU of the predicter module of the armband, using a timer with a set frequency of 6 MHz. The MCU clock frequency is set at 24 MHz. The timer was initialized and started in correspondence to the arrival of an ATC acquisition set and stopped after the definition of the output class. The averaging values are reported in Table 6.6.

The total latency is obtained by adding the ATC window length and computation time, equal to 130 ms, to the values mentioned above. All tested algorithms have a latency lower than the 300 ms required for the online application, making the system suitable for wearable real-time application. However, NN and RF system latencies are way below the ones for SVM and KNN making them preferable for the developed system. Moreover, in the case of the SVM classifier, the total latency (i.e. prediction + ATC window and computation) is equal to 297.56 ms, being at the upper limit for the transmission of the data. In this case, the system cannot send all the data but lose 24.53% of the ATC values. This delay is not cumulative because the received data is always the newest sent, not constituting a problem for the system during the classification process.

Table 6.6: System Latency for the Tested Classifiers.

	NN	RF	SVM	KNN
System Latency (ms)	1.036	0.269	167.561	70.326

6.3 Power consumption

Power consumption analysis was performed using a DMM7510 7 1/2 digit graphical sampling multimeter and a digital oscilloscope. For the measurements, an INA240EVM with a gain of 200 V/V was also used. The device is a voltage output current shunt that senses drops across shunts at specific common-mode voltages. The setup is shown in Figure 6.1.

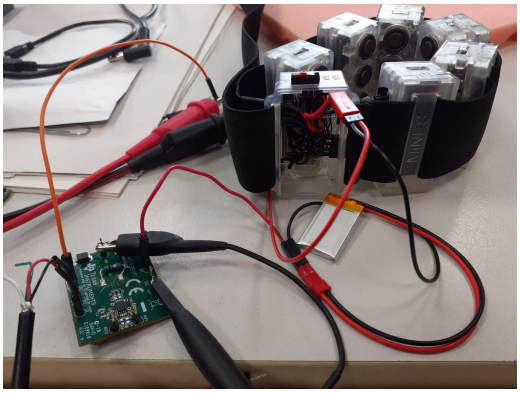


Figure 6.1: Setup used for measuring the power consumption.

In active mode, the system power consumption is low, at 0.58 mW, and shows only peaks at intervals of 500 ms corresponding to the advertising (Figure 6.2). When sending the ATC data, the system power consumption is instead 1.13 mW, which sligthly increase while calibrating the threshold 1.63 mW. During the acquisition window the current is maintained low but it presents an higher step due to the calculation required by the prediction, with different length, depending on the used algorithm. As expected, the classification occurs every 130 ms, except for the SVM classifier, due to the introduced delay. Figures 6.4, 6.5, 6.6, 6.7 show the measured current for each classifier.

Table 6.7: System Latency for the Tested Classifiers.

	NN	RF	SVM	KNN
Power Consumption (mW)	1.230	1.179	4.318	2.606

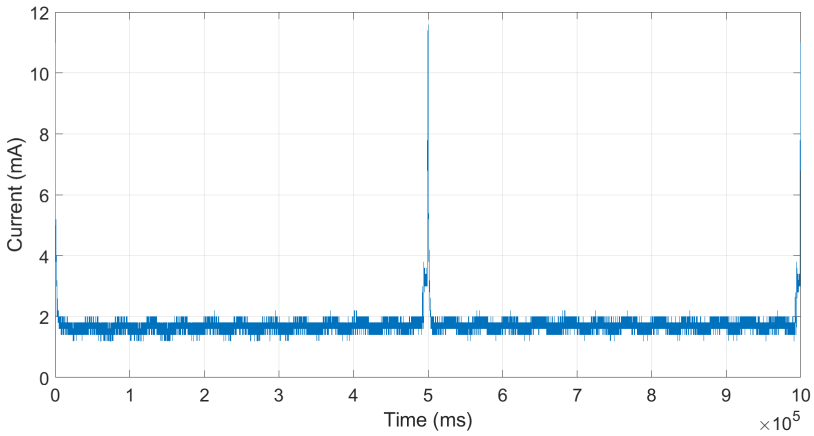


Figure 6.2: Current absorption when the armband is turned on. In this condition, only the advertising is visible.

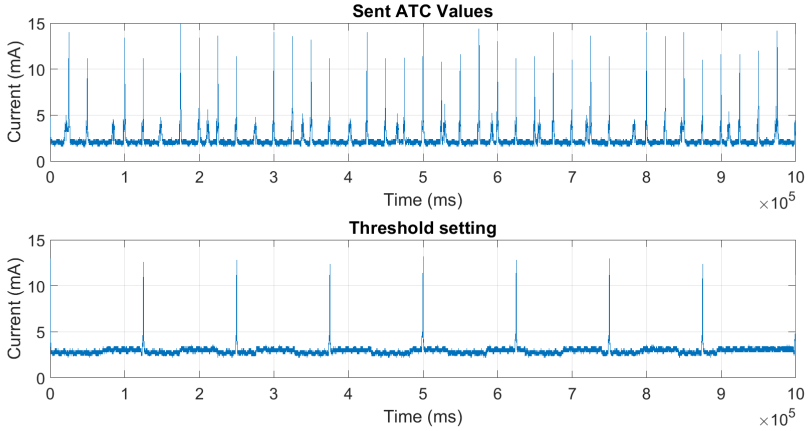


Figure 6.3: Current absorption during ATC data sending and during threshold setting.

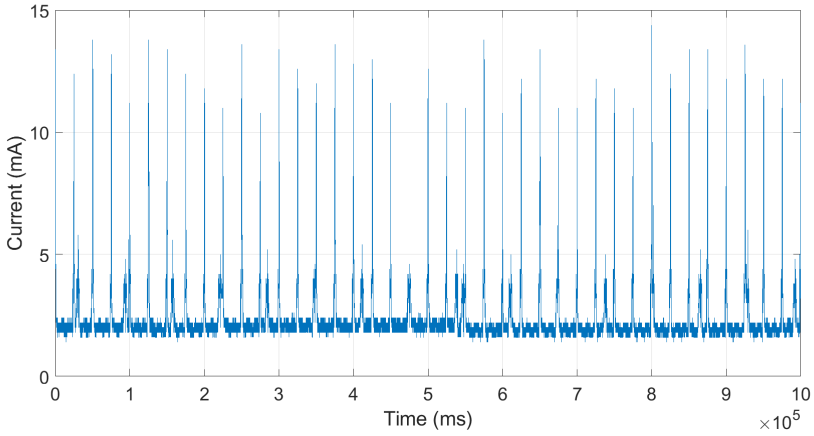


Figure 6.4: Current absorption graph for Neural Network.

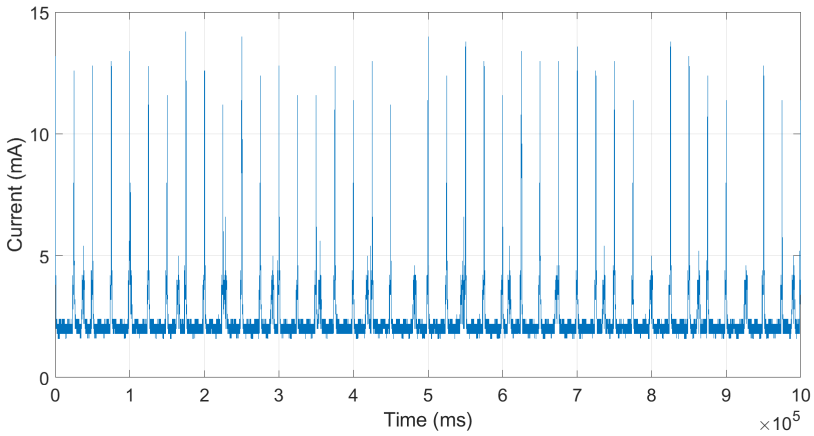


Figure 6.5: Current absorption graph for Random Forest.

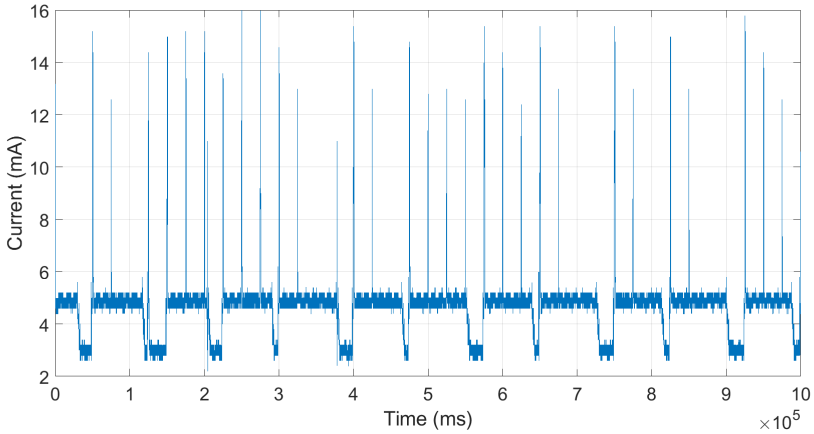


Figure 6.6: Current absorption graph for Support Vector Machine.

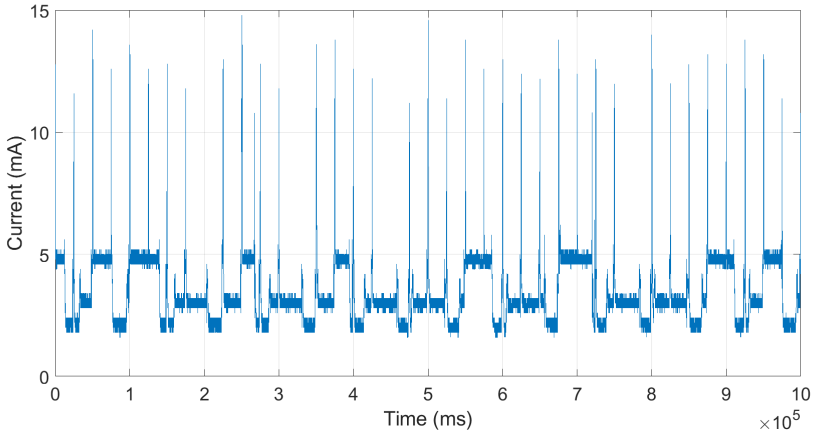


Figure 6.7: Current absorption graph for K-Nearest Neighbour.

6.4 Online results comparison

The results obtained in the analysis of the tested ML classifiers are reported in Table 6.8.

- NN and RF are the overall best classifiers, offering low computational time, nearly equivalent classification accuracy and low power consumption during computation. RF offers a better trade-off in terms of computational time and power consumption.
- KNN and SVM returned lower accuracies, higher computational times, especially for the SVM, which has introduced a delay in data sending and also power consumption. Additionally, their implementation on the MCU is challenging in terms of memory constraints, leading to the requirement of additional algorithms for the reduction of the dataset or of the support vectors. These alterations lead to the lowering of the online performances and do not completely resolve the higher computational weight that these algorithms requires.

Table 6.8: Online Classifiers Comparison.

	Global Accuracy (%)	Computational Time (ms)	Average Power Consumption (mW)
NN	94.60	1.036	1.230
RF	94.54	0.269	1.179
SVM	93.42	167.561	4.318
KNN	92.31	70.326	2.606

6.5 Comparison with Existing sEMG-based Armbands

As presented in Section 2.3, in the research field, several sEMG armbands have been proposed and developed. The presented system shows similar technical parameters and competitive performances, with fewer sEMG channels and a medium to high number of recognized gestures, offering also low power consumption and latencies.

Table 6.9: Comparison among sEMG based Armbands.

	gForce-Pro [65]	Myo [67]	3DC [68]	This work
sEMG channels	8	8	10	7
N° of Gestures	8	6	11	9
Transmitter	BLE 4.2	BLE 4.0	Enhanced Shockburst	BLE 4.2
Autonomy (h)	N.A.	16	6	5
Battery	200 mA h Li-On	100 mA h Li-ion	100 mA h LiPo	110 mA h LiPo
Accuracy(%)	N.A.	95.62	89.47	94.60

Chapter 7

Conclusions and Future Works

In this thesis project, an ATC-based armband for hand gesture recognition is proposed. The embedded low power system acquires the surface ElectroMyoGraphic signal from the forearm employing dry electrodes and extracts the ATC parameter, fed as input for the Machine Learning algorithms used for the classification. Exploiting the ATC event-driven technique, the computational complexity of the ML classifiers is decreased, leading to low power consumption and latency, which make the final implementation suitable for real-time wearable applications.

A preliminary analysis was conducted on an existing dataset created to study the feasibility of a seven-channel armband. In this phase, Neural Network, Support Vector Machine, Random Forest and K-Nearest Neighbour were tested and successively implemented online. The offline implementation was conducted in the MATLAB® environment taking advantage of its high computational efficiency and specific toolboxes to recognize seven active gestures. Firmware deployment has been carried out to obtain low power consumption, matching the requirements of a wearable battery-powered device, and a latency below 300 ms, suitable for real-time applications.

After that, a model of the armband was designed, and 3D printed. The proposed device comprises seven bipolar acquisition channels consisting of dry electrodes, each acquiring the sEMG signal. The PCB with an Apollo 3 Blue MCU with an ARM Cortex M4F µP performs signal conditioning and the ATC parameter extraction onboard. The data is then used as input for the trained classifiers. Data transfer is achieved using a Bluetooth Low Energy module (BLE 4.2).

System validation entailed the creation of a new dataset involving 20 people, each executing nine movements, repeated twice, within three sessions. The previously evaluated algorithms were trained on the newly acquired data and deployed on the MCU for testing, conducted over five volunteers. The performances reached by the classifiers are similar, with the NN and the RF being the best algorithms. In terms of global accuracy, NN reaches 94.60 %, and RF 94.54%. Power consumption is lower for the RF with 1.179 mW, against the 1.230 mW of the NN. Also, for the computational time, the RF presents better performance with a latency of 0.269 ms

while the NN is at 1.036 ms. The obtained results are comparable with the state of art sEMG-based armband present in literature, making the wearable system suitable for real-time use.

Several future improvements can be performed on the system.

Firmware optimization can reduce power consumption and memory constraints for high-cost computational algorithms such as the SVM, leading to faster computation.

The use of the window in the online phase can be investigated by deploying the trained algorithms on the MCU to see if the use of ATC profiles as the classifier input can lead to performance improvements.

Another improvement is related to the positioning of the armband. The recognition of the gestures is strictly related to a good placement of the electrodes over the correct muscle. Adding a routine that identifies the first channel during the calibration phase can lead to an increase in the system performance and usability, making it more user-friendly.

Regarding the hardware, the design of the armband modules can be improved to enhance wearability and ease of use. The closure system of the armband can be designed to make the system more stable on the forearm and easier to wear.

Bibliography

[1] N. Muralitharan I. Peate. *Fundamentals of Anatomy and Physiology for Nursing and Healthcare Students*. pp. 154-168: Wiley Blackwell, 2016 (cit. on p. 1).

[2] D. W. Onyango S. M. Kisia. *Muscular System of Vertebrates*. pp. 39-50: Enfield, 2005 (cit. on p. 1).

[3] L. M. Biga, S. Dawson, A. Harwell, R. Hopkins, J. Kaufmann, M. LeMaster, P. Matern, K. Morrison-Graham, and J. Runyeon D. Quick. *Anatomy and Physiology*. OpenStax/Oregon, State University. 2013. URL: `https://open.oregonstate.education/aandp/chapter/10-1-overview-of-muscle-tissues/` (cit. on pp. 2–4).

[4] Gerard J.Tortora and Bryan H.Derrickson. *Principles of Anatomy and Physiology (11th edition)*. Hoboken, NJ.: Jhon Wiley and Sons, 2009, Chap. 10-11, ISBN:9780470084717 (cit. on p. 5).

[5] C. L. Stanfield. *Fisiologia del muscolo*. Napoli: EdiSES s.r.l., 2012. Chap. 12, pp.330-359. ISBN: 9788879597142 (cit. on p. 5).

[6] J.E.Hall. *Guyton and hall textbook of medical physiology (12th edition)*. Saunders, 2010. pp.71-90. ISBN: 9780323389303 (cit. on p. 5).

[7] B. Mitchell and L. Whited. *Anatomy, Shoulder and Upper Limb, Forearm Muscles*. Treasure Island (FL): StatPearls Publishing, Aug 15, 2020. https://www.ncbi.nlm.nih.gov/books/NBK536975/ (cit. on p. 6).

[8] M. Chaudhry, H. Aminullah, M. Sinkler, and et al. *Anatomy, Shoulder and Upper Limb, Forearm Compartments*. Treasure Island (FL): StatPearls Publishing, Aug 8, 2021. https://www.ncbi.nlm.nih.gov/books/NBK539784/ (cit. on p. 6).

[9] *Anterior Forearm*. url: https://basicmedicalkey.com/anteri or-forearm/ (cit. on pp. 6, 7).

[10] R. Merletti and D. Farina. *SURFACE ELECTROMYOGRAPHY Physiology, Engineering, and Applications*. Hoboken, New Jersey: John Wiley and Sons, Inc., 2016. ISBN: 978-1-118-98702-5 (cit. on pp. 7, 9).

[11] J. Vasković. *Action Potential*. url: https://www.kenhub.com/en/library/ anatomy/action‡ potential (cit. on p. 8).

[12] İmran Göker. «Detection and Conditioning of EMG». In: (Jan. 2014), pp. 58–94. DOI: `10.4018/978-1-4666-6090-8.ch003` (cit. on pp. 8, 15).

[13] T. Vieira, A. Botter, S. Mucelia, and D. Farina. «Specificity of surface EMG recordings for gastrocnemius during upright standing». In: *Scientific Reports* 7 (Oct. 2017). DOI: 10.1038/s41598-017-13369-1 (cit. on p. 9).

[14] Runer Augusto Marson and César Ferreira Amorim. «Application of Surface Electromyography in the Dynamics of Human Movement». In: Chap.16 (Oct. 2012). DOI: http://dx.doi.org/10.5772/52463 (cit. on p. 9).

[15] C. J. De Luca, A. Adam, R. Wotiz, L. D. Gilmore, and S. H. Nawab. «Decomposition of surface emg signals». In: *Journal of neurophysiology* 96, no.3 (2006). DOI: https://doi.org/10.1152/jn.00009.2006 (cit. on p. 9).

[16] R. Merletti and D. Farina. *SURFACE ELECTROMYOGRAPHY Physiology, Engineering, and Applications*. Hoboken, New Jersey: IEEE Press, Wiley Publishing, 2016. ISBN: 978-1-118-98702-5 (cit. on p. 10).

[17] *Pre-Gelled Ag/AgCl Electrodes*. url: https://www.biopac.com/product-category/research/electrodes/ (cit. on pp. 10, 11).

[18] *Non-gelled Reusabe Ag/AgCl*. url: https://plux.info/electrodes/60- non-gelled-reusable-agagcl-electrodes.html. (cit. on p. 11).

[19] A. Phynyomark, P. Phukpattaranont, and C. Limsakul. «Feature reduction and selection for EMG signal classification». In: *Science Direct, Expert System with Applications 39* (2012). DOI: https://doi.org/10.1152/jn.00009.2006 (cit. on p. 13).

[20] Marco Crepaldi, Marco Paleari, Alberto Bonanno, Alessandro Sanginario, Paolo Ariano, Duc Hoa Tran, and Danilo 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)*. 2012, pp. 116–119. DOI: 10.1109/BioCAS.2012.6418406 (cit. on pp. 15, 35).

[21] P. Motto Ros, N. Celadon M.Paleari, A. Sanginario, A. Bonanno, and M. Crepaldi. «A Wireless Address-Event Representation System for ATC-Based Multi-Channel Force Wireless Transmission». In: *5th IEE Int. Workshop Advances in Sensors and Interfaces* (June 2013), pp. 51–56 (cit. on pp. 15, 35).

[22] Shahshahani, P. Motto Ros, A. Bonanno, M. Crepaldi, M. Martina, D. Demarchi, and G. Masera. «An All-Digital Spike-based Utra-Low-Power IR-UWB Dynamic Average Threshold Crossing Scheme for Muscle Force Wireless Transmission». In: *Design, Automation and Test in Europe Conference and Exhibition (DATE)* (2015) (cit. on p. 15).

[23] 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 IEE Biomedical Circuits and Systems». In: *5th IEE Int. Workshop Advances in Sensors and Interfaces* () (cit. on p. 16).

[24] A. Ng. *Machine Learning by Stanford University*. url: https://www.coursera.org/learn/machine-learning/home/welcome, 2018 (cit. on p. 17).

[25] W.G Wolpert D.H.; Macready. «No Free Lunch Theorems for Optimization.» In: *IEEE Trans. Evol. Comput.* (1997), pp. 67–82 (cit. on p. 17).

[26] A. Halevy, P. Norvig, and Pereira F. «The Unreasonable Effectiveness of Data». In: *IEEE Intell. Syst.* 24 (2009), pp. 8–12 (cit. on p. 17).

[27] Artem Oppermann. *Artificial Intelligence vs. Machine Learning vs. Deep Learning*. Oct. 2019. URL: https://towardsdatascience.com/artificial-intelligence-vs-machine-learning-vs-deep-learning-2210ba8cc4ac (cit. on p. 17).

[28] *Machine Learning*. url: https://www.ibm.com/cloud/learn/machine-learning (cit. on p. 18).

[29] Fateh Boutekkouk. «Embedded systems codesign under artificial intelligence perspective: a review». In: *International Journal of Ad Hoc and Ubiquitous Computing* 32 (Jan. 2019), pp. 257–269 (cit. on pp. 18, 25).

[30] Amit D.Kachare A.D.Dongare R.R.Kharde. «Introduction to Artificial Neural Network». In: *International Journal of Engineering and Innovative Technology (IJEIT)* 2 (July 2012), pp. 189–194 (cit. on p. 18).

[31] Stacey Ronaghan. *Deep Learning: Overview of Neurons and Activation Functions*. url: https ://srnghn.medium.com/deep-learning-overview-of- neurons-and-activation-functions-1d98286cf1e4, July 2018 (cit. on p. 19).

[32] E. Ficarra. *Bioinformatics - Overview of Machine Learning and Pattern Recognition*. Mar. 2019, [Bioinformatics lecture] (cit. on p. 19).

[33] Esperanza García-Gonzalo, Zulima Fernández-Muñiz, Paulino Jose Garcia Nieto, Antonio Sánchez, and Marta Menéndez. «Hard-Rock Stability Analysis for Span Design in Entry-Type Excavations with Learning Classifiers». In: *Materials* 9 (June 2016), p. 531. DOI: 10.3390/ma9070531 (cit. on p. 20).

[34] Zhiliang Liu and Hongbing Xu. «Kernel Parameter Selection for Support Vector Machine Classification». In: *Journal of Algorithms and Computational Technology 8.2* (2014), pp. 163–177. DOI: 10.1260/1748-3018.8.2 (cit. on p. 21).

[35] Avinash Navlani. *Decision Tree Classification in Python*. 2018. URL: https://devopedia.org/decision-trees-for-machine-learning#Navlani-2018 (cit. on p. 22).

[36] Onel Harrison. *Machine Learning Basics with the K-Nearest Neighbors Algorithm*. Aug. 2018. URL: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761 (cit. on p. 23).

[37] MathWorks. *Classification Using Nearest Neighbors*. URL: https://it.mathworks.com/help/stats/classification-using-nearest-neighbors.html (cit. on pp. 23, 24).

[38] Paul Rosero, Vivian Batista, Edwin Rosero, Edgar Jaramillo, Jorge Caraguay-Procel, José Pijal-Rojas, and Diego Peluffo. «Intelligence in Embedded Systems: Overview and Applications: Volume 1». In: Jan. 2019, pp. 874–883. ISBN: 978-3-030-02685-1. DOI: 10.1007/978-3-030-02686-8_65 (cit. on p. 25).

[39] Massimo Ravasi, Marco Mattavelli, Paul Schumacher, and Robert Turney. «High-Level Algorithmic Complexity Analysis for the Implementation of a Motion-JPEG2000 Encoder». In: vol. 2799. Sept. 2003, pp. 440–450. ISBN: 978-3-540-20074-1. DOI: 10.1007/978-3-540-39762-5_50 (cit. on p. 26).

[40] André G. Ferreira Sérgio Branco and Jorge Cabral. «Machine Learning in Resource-Scarce Embedded Systems, FPGAs, and End-Devices: A Survey». In: *Electronics* 8 (Nov. 2019). DOI: 10.3390/electronics8111289 (cit. on p. 26).

[41] Darius Morawiec. «sklearn-porter». Transpile trained scikit-learn estimators to C, Java, JavaScript and others. URL: https://github.com/nok/sklearn-porter (cit. on p. 27).

[42] A. Chandler. *Microchip Introduces the Industry's First MCU with Integrated 2D GPU and Integrated DDR2 Memory for Groundbreaking Graphics Capabilities*. Sept. 2019. URL: https://www.microchip.com/pressreleasepage/microchip-introduces-the-industry-s-first-mcu-wit%20h-integrated-2d-gpu-and-integrated-ddr2-memory-for-groundbreaking-graphics-capabilities (cit. on p. 28).

[43] R. Dirvin. *Next-generation Armv8.1-M aRchitecture: Delivering Enhanced Machine Learning and Signal Processing for the Smallest Embedded Devices*. Oct. 2019. URL: https://www.arm.com/company/news%20/2019/02/next-generation-armv8-1-m-architecture (cit. on p. 28).

[44] ETA Compute. *ASICs for Machine Intelligence in Mobile and Edge Devices*. 2019. URL: https:%20//etacompute.com/ (cit. on p. 28).

[45] Chirag Gupta et al. «ProtoNN: Compressed and Accurate kNN for Resource-scarce Devices». In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 1331–1340. URL: https://proceedings.mlr.press/v70/gupta17a.html (cit. on p. 28).

[46] Ashish Kumar, Saurabh Goyal, and Manik Varma. «Resource-efficient Machine Learning in 2 KB RAM for the Internet of Things». In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 1935–1944. URL: https://proceedings.mlr.press/v70/kumar17a.html (cit. on p. 28).

[47] Liangzhen Lai, Naveen Suda, and Vikas Chandra. *CMSIS-NN: Efficient Neural Network Kernels for Arm Cortex-M CPUs*. 2018. arXiv: 1801.06601 [cs.NE] (cit. on p. 28).

[48] Aditya Kusupati, Manish Singh, Kush Bhatia, Ashish Kumar, Prateek Jain, and Manik Varma. *FastGRNN: A Fast, Accurate, Stable and Tiny Kilobyte Sized Gated Recurrent Neural Network*. 2019. arXiv: 1901.02358 [cs.LG] (cit. on p. 28).

[49] Tomasz Szydlo, Krzysztof Zielinski, and Marcin Jarzab. «Resource-aware log monitoring data transmission for Smart and IoT devices». In: *MobiQuitous 2020, in print*. ACM, 2020, pp. 1–10 (cit. on p. 28).

[50] Charles Leech, Yordan P. Raykov, Emre Ozer, and Geoff V. Merrett. «Real-time room occupancy estimation with Bayesian machine learning using a single PIR sensor and microcontroller». English. In: *2017 IEEE Sensors Applications Symposium (SAS 2017) Proceedings*. 12th IEEE Sensors Applications Symposium, SAS 2017 ; Conference date: 13-03-2017 Through 15-03-2017. United States: IEEE, Apr. 2017. DOI: 10.1109/SAS.2017.7894091 (cit. on p. 28).

[51] Karen Zita Haigh, Allan M. Mackay, Michael R. Cook, and Li G. Lin. «Machine Learning for Embedded Systems : A Case Study». In: 2015 (cit. on p. 28).

[52] Gary B. Parker and Mohammad O. Khan. «Distributed neural network: Dynamic learning via backpropagation with hardware neurons using arduino chips». In: *2016 International Joint Conference on Neural Networks (IJCNN)* (2016), pp. 206–212 (cit. on p. 29).

[53] Jusoh S. Yasen M. «MA systematic review on hand gesture recognition techniques, challenges and applications.» In: *PeerJ Comput Sci.* (Sept. 2019). DOI: 10.7717/peerj-cs.218 (cit. on p. 30).

[54] Tushar Chouhan, Ankit Panse, Anvesh Voona, and Sameer M. «Smart Glove With Gesture Recognition Ability For The Hearing And Speech Impaired». In: Sept. 2014. DOI: 10.1109/GHTC-SAS.2014.6967567 (cit. on p. 30).

[55] Nicolò Bargellesi, Mattia Carletti, Angelo Cenedese, Gian Antonio Susto, and Matteo Terzi. «A Random Forest-based Approach for Hand Gesture Recognition with Wireless Wearable Motion Capture Sensors». In: *IFAC-PapersOnLine* 52 (Jan. 2019), pp. 128–133. DOI: 10.1016/j.ifacol.2019.09.129 (cit. on p. 30).

[56] Linchu Yang, Ji'an Chen, and Weihang Zhu. «Dynamic Hand Gesture Recognition Based on a Leap Motion Controller and Two-Layer Bidirectional Recurrent Neural Network». In: *Sensors* 20.7 (2020). ISSN: 1424-8220. DOI: 10.3390/s20072106. URL: https://www.mdpi.com/1424-8220/20/7/2106 (cit. on p. 30).

[57] Mohidul Alam Laskar, Amlan Jyoti Das, Anjan Kumar Talukdar, and Kandarpa Kumar Sarma. «Stereo Vision-based Hand Gesture Recognition under 3D Environment». In: *Procedia Computer Science* 58.Complete (2015), pp. 194–201. DOI: 10.1016/j.procs.2015.08.053 (cit. on p. 31).

[58] Zengshan Tian, Jiacheng Wang, Xiaolong Yang, and Mu Zhou. «WiCatch: A Wi-Fi Based Hand Gesture Recognition System». In: *IEEE Access* 6 (2018), pp. 16911–16923. DOI: 10.1109/ACCESS.2018.2814575 (cit. on p. 31).

[59] Benatti S, Casamassima F, Milosevic B, Farella E, Schönle P, Fateh S, Burger T, Huang Q, and Benini L. «A Versatile Embedded Platform for EMG Acquisition and Gesture Recognition». In: *IEEE Trans Biomed Circuits Syst.* (Oct. 2015), pp. 620–630. DOI: 10.1109/TBCAS.2015.2476555 (cit. on p. 31).

[60] Hang Zhao, Jiangping Hu, Yuping Zhang, and Hong Cheng. «Hand gesture based control strategy for mobile robots». In: *2017 29th Chinese Control And Decision Conference (CCDC)*. 2017, pp. 5868–5872. DOI: 10.1109/CCDC.2017.7978217 (cit. on p. 31).

[61] Siddharth S. Rautaray and Anupam Agrawal. «Interaction with virtual game through hand gesture recognition». In: *2011 International Conference on Multimedia, Signal Processing and Communication Technologies*. 2011, pp. 244–247. DOI: 10.1109/MSPCT.2011.6150485 (cit. on p. 31).

[62] Trong Khanh Nguyen, Bui Ha, and Cuong Pham. «Recognizing Hand Gestures for Controlling Home Appliances with Mobile Sensors». In: Sept. 2019. DOI: 10.1109/KSE.2019.8919419 (cit. on p. 32).

[63] Shaun Canavan, Walter Keyes, Ryan Mccormick, Julie Kunnumpurath, Tanner Hoelzel, and Lijun Yin. «Hand gesture recognition using a skeleton-based feature representation with a random regression forest». In: *2017 IEEE International Conference on Image Processing (ICIP)*. 2017, pp. 2364–2368. DOI: 10.1109/ICIP.2017.8296705 (cit. on p. 32).

[64] Helman Stern, Yael Edan, Michael Gillam, Jon Handler, Craig Feied, and Mark Smith. «A Gesture-based Tool for Sterile Browsing of Radiology Images». In: *Journal of the American Medical Informatics Association : JAMIA* 15 (May 2008), pp. 321–3. DOI: 10.1197/jamia.M241 (cit. on p. 32).

[65] Paolo Visconti, Federico Gaetani, Giovanni Zappatore, and Patrizio Primiceri. «Technical Features and Functionalities of Myo Armband: An Overview on Related Literature and Advanced Applications of Myoelectric Armbands Mainly Focused on Arm Prostheses». In: *International Journal on Smart Sensing and Intelligent Systems* 11 (June 2018), pp. 1–25. DOI: 10.21307/ijssis-2018-005 (cit. on pp. 33, 76).

[66] Paolo Visconti, Federico Gaetani, Giovanni Zappatore, and Patrizio Primiceri. «Technical Features and Functionalities of Myo Armband: An Overview on Related Literature and Advanced Applications of Myoelectric Armbands Mainly Focused on Arm Prostheses». In: *International Journal on Smart Sensing and Intelligent Systems* 11 (June 2018), pp. 1–25. DOI: 10.21307/ijssis-2018-005 (cit. on p. 33).

[67] *gForcePro EMG Armband*. URL: https%20:%20/%20/%20www%20.%20teachpe%20.%20com%20/%20anatomy%20-%20physiology/types-of-muscle (cit. on pp. 34, 76).

[68] Ulysse Côté Allard, Gabriel Gagnon-Turcotte, Francois Laviolette, and Benoit Gosselin. «A Low-Cost, Wireless, 3-D-Printed Custom Armband for sEMG Hand Gesture Recognition». In: *Sensors* 19 (June 2019), p. 2811. DOI: 10.3390/s19122811 (cit. on pp. 34, 76).

[69] Stefano Sapienza, Paolo Motto Ros, David Alejandro Fernandez Guzman, Fabio Rossi, Rossana Terracciano, Elisa Cordedda, and Danilo Demarchi. «On-Line Event-Driven Hand Gesture Recognition Based on Surface Electromyographic Signals». In: *2018 IEEE International Symposium on Circuits and Systems (ISCAS)* (2018), pp. 1–5 (cit. on p. 35).

[70] Andrea Mongardi, Paolo Motto Ros, Fabio Rossi, Massimo Ruo Roch, Maurizio Martina, and Danilo Demarchi. «A Low-Power Embedded System for Real-Time sEMG based Event-Driven Gesture Recognition». In: *2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)* (2019), pp. 65–68 (cit. on p. 35).

[71] Vincenzo Barresi. «Machine Learning Approaches for Embedded Real-Time Gesture Recognition». Tesi di laurea. Politecnico di Torino, Jul. 2019 (cit. on p. 35).

[72] Matteo Tolomei. «Towards an Electromyographic Armband: an Embedded Machine Learning Algorithms Comparison». Tesi di laurea. Politecnico di Torino, Dec. 2020 (cit. on pp. 37, 39, 54).

[73] Álvar Arnaiz Gonzales. «Estudio de Métodos de Selección de Instancias». Tesi di dottorato. UNIVERSIDAD DE BURGOS, Jen. 2018 (cit. on p. 43).

[74] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM : a library for support vector machines". In: ed. by ACM Transactions on Intelligent Systems and Technolgy.* 2011. URL: `http://www.csie.ntu.edu.tw/%20cjlin/libsvm` (cit. on p. 44).

[75] Shenglong Zhou. «Sparse SVM for Sufficient Data Reduction». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (Apr. 2021), pp. 1–11. DOI: `10.1109/TPAMI.2021.3075339` (cit. on p. 45).

[76] Jon Nordby. *emlearn: Machine Learning inference engine for Microcontrollers and Embedded Devices*. Mar. 2019. DOI: `10.5281/zenodo.2589394`. URL: `https://doi.org/10.5281/zenodo.2589394` (cit. on p. 46).

[77] Jared Becker Jonathan Valdez. *Understanding the I 2C Bus*. June 2015. URL: `https://www.ti.com/interface/i2c/overview.html` (cit. on p. 48).

[78] Andrea Zimarra. «Towards an Electromyographic Armband with dry electrodes for Hand Gesture Recognition». Tesi di laurea. Politecnico di Torino, Mar. 2020 (cit. on p. 49).