Contents

1 Introduction

Electrophysiology TODO: what is

The most prevalentTODO: cite modes of biopotential measurements are electrocardiography (ECG) and electromyography (EMG), measuring cardiac muscle potentials and skeletal muscle potentials respectively. While methods that use intracellular electrodes exist and are much more precise TODO: cite, using surface electrodes is very common thanks to their non invasive nature, albeit at a loss of specificity due to measuring a superposition of nearby potentials. The field of ECG is by and large applied to identifying and diagnosing cardiac abnormalities in clinical and first responder settings, focusing on disturbances of the cardiac rhythm, artery blood flow, or electrolyte availability for diagnosis of heart conditions.TODO: cite On the other hand, Surface EMG (henceforth sEMG) has broader applications, ranging from the clinical, such as diagnosing nerve and muscle damage, to prosthetics, rehabilitation, biofeedback therapy, ergonomics, sports and movement science [1], and it also opens up the possibility of novel Human-Computer Interaction methods.

Although the electrical design of biopotential measurement devices has been studied for hundreds of years, tracing back to Luigi Galvani's work in the 18th century, and there exist a number of current-day commercial, academic, and DIY solutions, only a small part aspire to the ideals of Open Source Hardware and/or Open Source Software (ASPEN[2], openBCI[3], BioAmp-EXG-Pill[4], MyoWare EMG[5]), and there exist no systems that integrate with ROS2 or the Linux IIO subsystem, limiting their flexibility in terms of building software that seamlessly interfaces with them.

We aim to fill this gap and provide an affordable and easily maintainable hardware and software solution for measuring multi-channel biopotential signals and interfacing them with ROS2 and LibIIO. Adapting this system to work with other software suites should be made as simple as possible thanks to the versatility of both ROS2 and LibIIO, and adapting it to other hardware platforms should also be facilitated by the decoupled nature of the hardware, firmware and software parts.

TODO: What is ROS2

The Linux Industrial I/O (IIO) subsystem was created in 2008 and mainlined in 2012 (commit `06458e2`) to provide a simple and unified way to interact with devices with ADC or DAC capabilities, broadly ranging from plain ADC/DAC chips to Inertial Measurement Units (Accelerometers, Gyroscopes, Magnetometers), Software Defined Radio, robotic actuators and sensors, and more. Within the Linux kernel, it fills in a gap between the *input* subsystem which can stream user input data and the *hwmon* subsystem used for monitoring and controlling hardware state such as CPU temperature and fan settings, albeit with low refresh rate. The IIO subsystem provides a modular way to define high bandwidth communication with input/output devices directly connected to the host via SPI or I2C and specified as part of the device tree. Despite the subsystem's name, it is not limited to industrial production applications: IIO usage is ubiquitous in today's linux-based devices for integrating features such as accelerometers, gyroscopes or ambient light sensors in a uniform way, besides other hardware abstraction layers.

Libiio is a library which provices cross-platform functionality for interacting with both local and remote IIO devices within the same framework. To accomplish this, it implements the IIO Daemon (IIOD) for which it defines the *IIOD protocol*. The current version of the IIOD protocol (v0.1) is based on a set of human readable commands sent via a single data stream of alternating client requests and server responses. This allows the same protocol to be used over a variety of transport layers, officially Serial (most relevant for this application), USB and TCP.

We present overview of the information flow, from the biological signals to the software, in figure 1.1.

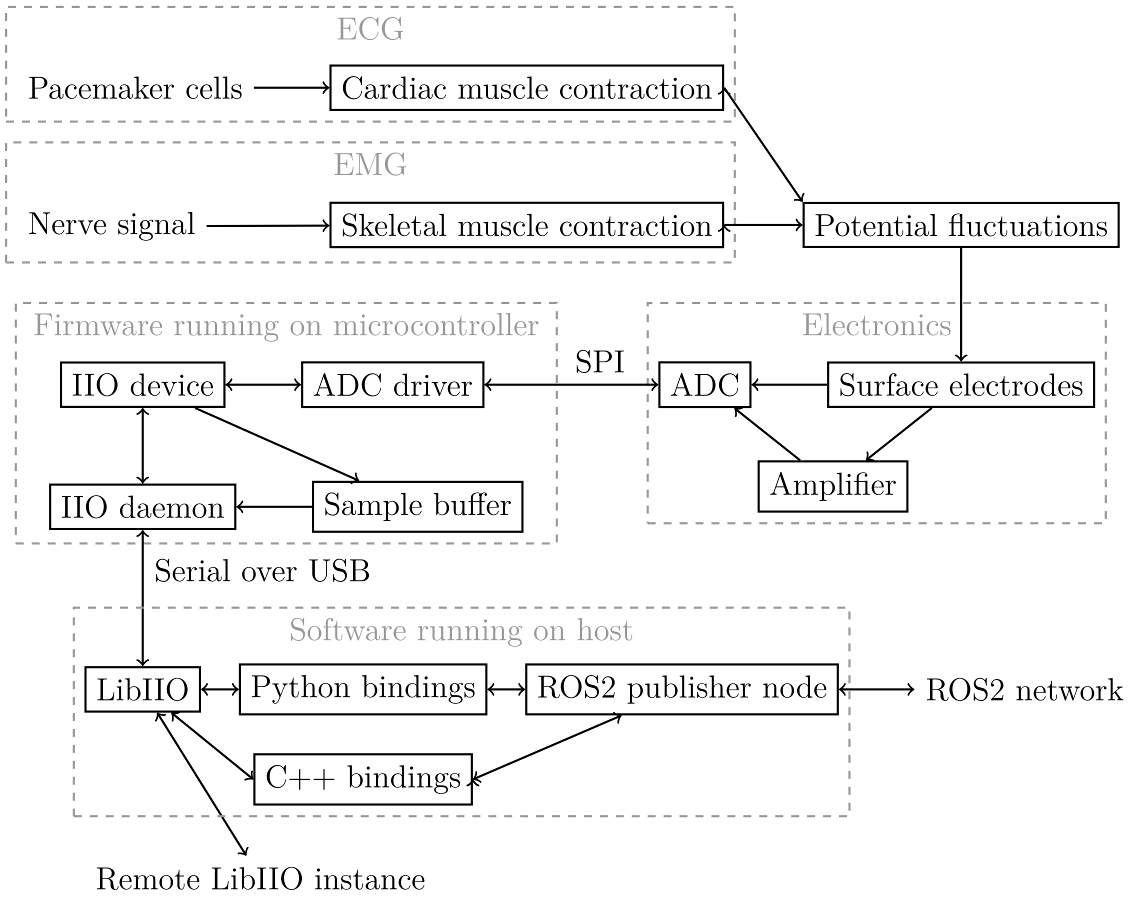

Figure 1.1. Overview of information flow

2 Bibliographic research

2.1 Anatomical basis

As part of normal function, all biological cells have a difference of electrical potential between their interior and their exterior, called the *membrane potential*. In the steady state, the *resting membrane potential* of both skeletal muscle cells and that of cardiomyocytes (cardiac muscle cells) is in the -70 to $-90\mathrm{mV}$ range [6], [7].

TODO: WIP: skeletal muscles, Sarcomere, sliding filament model, essential parts from [1]

TODO: ECG: find a reputable source to cite for cardio physiology

2.2 Market research and literature review

Biopotential measurements are no new subject and many products already exist both in the academic and commercial fields. All are subject to engineering tradeoffs between parameters such as number of channels, sample rate, precision, noise, electrode types, etc., making each better suited to a certain use case. To the best of our research, there is no universal, modular and extensible solution to the problem of biopotential measurements. Further on, we overview some representative options for EMG / ECG DAQ systems:

OpenBCI

Expensive, low data rate. TODO: more :)

ASPEN

[2]

What we're doing, but better, but a PhD thesis.

TODO: more :)

Quanser QNET Myoelectric board

Depends on NI Elvis; one channel; not open TODO: more :)

BioWare EXG Pills

Just the amplification part, no DAQ, rely on an arduino's ADC or some other. TODO: more :)

Cometa Pico EMG

Cometa Pico EMG

not open, expensiveTODO: more :)

MyoWare

TODO: more :)

the green one I forgot the name of

TODO: more :)

2.3 Noise sources

The superposition of electrical signals measurable by biopotential DAQs contains a complicated mix of noise coming from various physiological and other external factors, besides the signal of interest. Additionally, the measurement circuitry compounds noise at every step. Modeling some of the main noise components of the entire signal chain is essential for its design.

Some of the studied noise sources are *intrinsic* to the process and impossible to remove, such as thermal noise and the amplifier and ADC noise profiles. Others are not intrinsic, depend on the setup, and can be compensated for, such as mains hum.

2.3.1 Thermal noise

Even with ideal conductors and measurements, any resistive circuit will be affected by thermal noise, as a consequence of Planck's law of blackbody radiation [8, chapter 3.2]. An ideal resistor R at absolute temperature T will manifest thermal noise V_n over a frequency window Δf having mean squared value $\overline{V_n^2} = 4k_B TR\Delta f$, where $k_B = 1.380649 \cdot 10^{-23}\mathrm{J/K}$ is the Boltzmann constant.

For a rough estimate of the order of magnitude of thermal noise in our biopotential measurements, we study the following hypothetical setup: Assume the electrodes attached to the patient are in thermal equilibrium at a typical skin temperature of $T \approx 308\mathrm{K}$ [9], we are limiting the frequency band to $\Delta f = 500\mathrm{Hz}$ by filtering, and experiencing a typical resistance $R \approx 3500\Omega$ (electrode + contact + internal resistance) [10]. This is only useful as a rough order estimation, because the resistance will vary depending on the electrode location, surface preparation, measured individual, electrode condition, etc. The estimated thermal noise will is:

$$\overline{V_n^2} = 4k_B TR\Delta f \approx 0.02977\mathrm{\mu V}^2 \tag{2.1}$$

Which can alternatively be interpreted as having an RMS noise of $\sqrt{0.02977\mathrm{\mu V}^2} \approx 172.5\mathrm{nV}$ or a noise density of $\sqrt{4k_B TR} = 7.716\mathrm{nV}/\sqrt{\mathrm{Hz}}$.

2.3.2 Amplifier and ADC noise

Both ADCs and amplifier circuits have specified noise figures which pose design tradeoffs. In ADCs, the noise is specified as additive noise at the input V_{na}, such that a measurement V_{meas} of a "real" voltage V will be $V_{meas} = V + V_{na}$. Amplifiers are characterized by both input noise V_{ni} created by the input stage, as well as output noise V_{no} created by internal circuirty and the output stage. The output of an amplifier with gain G is $V_{amp} = (V + V_{ni}) \cdot G + V_{no}$. As an easier to use figure, the *Input Referred Noise* is the equivalent disturbance at the input that would create both the effects of input and output noise componded. $\mathrm{IRN} = V_{ni} + V_{no}/G$ such that we can simplify to $V_{amp} = (V + \mathrm{IRN}) \cdot G$.

This raises one of the most important design decisions of the signal chain, namely deciding on the amplification structure before the ADC. Using no amplifier at all is the most economic and also avoids V_{ni} and V_{no}, but in the case of very small input signals, they are left to be dominated by ADC noise V_{na}. Adding an amplifier with unity gain only compounds amplifier noise and is as such detrimental, but further increasing the gain increases the signal-to-noise ratio (SNR):

$$\text{SNR} = \frac{\text{"signal"}}{\text{"noise"}} = \frac{G \cdot V}{G \cdot V_{ni} + V_{no} + V_{na}} = \frac{V}{V_{ni} + (V_{no} + V_{na})/G} \longrightarrow \frac{V}{V_{ni}} \tag{2.2}$$

Thus, given an amplifier with known V_{ni} we can compute the best-case noise figure assuming "infinite" gain. The rationally convergent nature of the above expression inevitably creates a situation of diminishing returns, wherein increasing the gain will always increase the signal quality, but even relatively small gains achieve practically ideal performance.

The drawback of having very large gain is the limit it incurs on the measurable range of input voltages. Any analog circuit can only handle a limited interval of voltages, which when exceeded leads to unreliable measurements, with the possibility of permanent damage to the equipment. If the ADC is limited to measuring $\pm V_{max}$, the measurable signals are implicitly limited to $\pm V_{max}/G$.

2.3.3 ECG noise models

Besides the unavoidable thermal and electronics noise, further disturbances in ECG measurements can be classified, as per [11], into: low frequency ($< 0.5\text{Hz}$) baseline wander caused by body motion and electrode contact issues, power-line artifacts ($50/60\text{Hz}$ and their harmonics) caused by insufficient power supply decoupling when the measurement device is connected to mains power, and muscle artifacts (EMG signals). Of the three, the first two are relevant and undesireable in any biopotential measurement, while the latter is obviously the signal of interest in EMG applications.

From a signal processing standpoint, a simple solution for low frequency wander and power line artifacts is using a DC blocking comb filter.

2.4 Signal chain design

TODO: figure out which parts go in the background section and which go to the design section. Currently put everything in the design section.

2.4.1 Instrumentation amplifiers

2.4.2 Multiplexed ADCs

2.4.3 ?

3 Analysis, design and implementation

3.1 Objectives

Additionally, we

TODO: Justify objectives, format these boring bullet lists in a nicer way

MUST:

- Open Hardware and So20ftware
- 8 channels
- Per-channel configuration
- 1000Hz data rate (500Hz bandwidth) TODO: NNNNNOOOOOOOOO
- High CMRR, PSRR
- High SNR (>40dB EMG; >20dB for ECG)
- Open and extensible interfacing: ROS2, IIO

SHOULD (unchecked go to future work):

- 16 channels
- Differential / single-ended configurability
- Right Leg Drive
- 1024Hz/2048Hz /channel data rate (1 Hz per FFT bin)
- <1uV noise after filtering
- 1uV precision

Guiding principles:

- Cost
- Usability as an educational tool
- Use Analog Devices parts
- Modularity (decoupled hardware-firmware-software)

TODO: After finishing the prototype, make a comparison table between the already existing products and mine, especially on these objectives

TODO: Signal chain figure: 1. Patient, anatomy 2. Electrodes 3. Buffer and amplification 4. ADC 5. Microcontroller, using the no-OS framework. Contains customizable digital filters 6. IIO protocol 7. Linux IIO subsystem 8. libiio, pyadi-iio variant A, with a network-enabled microcontroller: from 5, ROS2 protocol find the name! to ROS2 network running on the microcontroller variant B, with a microcontrollwe without networking capabilities: from 8, ROS2 protocol to ROS2 network running on the carrier board / host

3.2 Electical part

3.2.1 Interfacing with the patient

The first step we control in the signal chain is the point of contact with the patient. The electrode type and quality have a great influence on the overall design, performance, and subjective user experience.

The state of the art are conductive polymer PEDOT:PSS electrodes, thanks to their great mechanical, electrical, and chemical properties: They adhere and conform to skin, are flexible, self-healing, and have good biocompatibility, only falling behind the classic Ag/AgCl electrodes in terms of contact impedance, yet still being conductive enough for biopotential readings. [12], [13] We do not consider them viable for this project due to them not being commercially available in standard and mass produced forms, being mostly a custom product experimented with in today's laboratories, even half a century after their discovery in the 1980s.

DIY cheap alternative TODO: cite that paper with DIY AgCl electrodes

This project settles for snap-on adhesive ECG gel electrodes thanks to being cheap, readily avaialble, standardized, and they can be connected to with inexpensive aligator clips, besides special (but more expensive) snap clips. The downsides are their low shelf life TODO: compare expired electrode signal quality with fresh electrodes once I get some fresh ones, comparatively larger contact area (cca 200mm^2 vs [1]'s 2mm electrode diameters), and are lacking in subjective user experience, sometimes being very painful to detach.

3.2.2 ADC

TODO: preface on how to choose an ADC

We initially chose the AD7124-8[14] thanks to its very diverse feature set, including: programmable gain amplifiers (PGAs) which had the potential to remove the need for the external amplification stage, high precision, high number of channels, and high sample rate. Unfortunately, it is not a single cycle ADC. This means that although it has very good throughput on a single channel, when switching between two channels there is a dead time for the integrated filter to settle, which effectively reduces the per-channel sample rate to less than 100Hz when sampling all 16, which is virtually useless for ECG ($f_{\text{max}} = 150\text{Hz}$) and EMG ($f_{\text{max}} = 500\text{Hz}$).

Our next choice was the AD4114[15]. It doesn't include PGAs, but otherwise has very good specs and has a much higher sample rate and a single cycle conversion mode.

TODO: spec comparison between AD7124-8 and AD4114?

TODO: describe AD4114 function, especially: channels vs setups, internal channel sequencer, overall communication model, **how to check for new samples**, noise versus sample rate. All of these will more or less paraphrase the datasheet.

TODO: 1007Hz and 6211Hz are the sample rates we're most interested in

3.2.3 Amplification stage

The chosen AD4114 ADC has $N = 24$ bits of resolution with one LSB equating to $\Delta v \approx 1.49\mu\text{V}$ if using the internal 2.5V reference. At $f_s = 1007\text{Hz}$ and $f_s = 6211\text{Hz}$, the ADC noise is specified to be around $\overline{V^2_{na,1007}} = 20\mu\text{V}$ and $\overline{V^2_{na,6211}} = 74\mu\text{V}$ RMS respectively. Without an

Table 3.1. a

Gain	Full scale	$\text{SNR}_{(\text{dB})}$
1	5V	34dB
2	2.5V	40dB
5	1V	48dB
10	500mV	54dB
20	250mV	59dB
50	100mV	64dB
100	50mV	65dB
1000	5mV	66dB

amplification stage, this would result in the following SNRs:

$$\begin{aligned}\text{SNR}_{(\text{dB})} &= 10\log_{10}\left(\frac{(1\text{mV})^2}{\overline{V_{na}^2}}\right)\\ \text{SNR}_{1007} &\approx 34\text{dB}\\ \text{SNR}_{6211} &\approx 23\text{dB}\end{aligned} \tag{3.1}$$

We choose the AD8226 **AD8226** instrumentation amplifier for the preamp stage. It is inexpensive, readily available, has more than sufficient CMRR ($> 80\text{dB}$) and can be configured for a gain of $G = 1$ to 1000 using a resistor $R_G = 49.4\text{k}\Omega/(G-1)$. Its RMS voltage noise for our desired bandwidth $\Delta f = 500\text{Hz}$ is:

$$\begin{aligned}V_{ni,RMS} &= \sqrt{500\text{Hz}\cdot(22\text{nV}/\sqrt{\text{Hz}})^2} = 0.492\mu\text{V}\\ V_{no,RMS} &= \sqrt{500\text{Hz}\cdot(120\text{nV}/\sqrt{\text{Hz}})^2} = 2.683\mu\text{V}\end{aligned} \tag{3.2}$$

With the preamp stage added before the ADC, the total noise of a sample and the theoretical SNR (1mV input) are:

$$\begin{aligned}\overline{V_n^2} &= \overline{V_{ni}^2} + \frac{\overline{V_{no}^2} + \overline{V_{na}^2}}{G^2}\\ \text{SNR}_{(\text{dB})} &= 10\log_{10}\left(\frac{(1\text{mV})^2}{\overline{V_n^2}}\right)\end{aligned} \tag{3.3}$$

An illustrative handful of values for various gain settings: TODO: rephrase :)

- adc has 24 bits of resolution => ? uV LSB => would seem like it's already suitable for our measurement ranges. Compute number of bits and SNR given a known ECG, EMG amplitude. - but we can do better! A pre-amp stage will amplify the signal, add some noise, but ultimately reduce the IRN. - can't amplify too much, though, because we don't want to exceed the full scale (or really the 0-5V supply for that matter) - set a hard limit for the gain so we don't clip

- compute bits / SNR for some gain values (as well as for no amplifier) - plot or table showing diminishing returns. Just stop when we're <0.5-1 bits from the best we can do (gain=1000)

TODO: tikz figure of signal chain

Assuming a single , followed by an ADC input noise V_{na}, are measuring the "true" voltage x, the resulting digitized sample y can be decomposed into:

$$y = Gx + GV_{ni} + V_{no} + V_{na} \tag{3.4}$$

Which can be rearranged to obtain an expression for RMS total noise V_n^2:

$$\overline{x^2} = \frac{\overline{y^2}}{G^2} + \underbrace{\left(\overline{V_{ni}^2} + \frac{\overline{V_{no}^2} + \overline{V_{na}^2}}{G^2} \right)}_{\overline{V_n^2}} \tag{3.5}$$

3.2.4 Microcontroller

The chosen microcontroller platform is the MAX78000FTHR.

Main requirements for the microcontroller: - SPI rate high enough to read the ADC at the wanted sample rate - Enough CPU freq and memory to run the IIO Daemon.

Nice-to-haves: - Networking capabilities so it can run a ROS2 node as a standalone device, not needing another "carrier" to forward the data to the ROS network.

Suitable very cheap microcontrollers: - ESP8266, ESP32 - RP2040 based boards (Raspberry Pi Pico, GroundStudio Pico, other clones)

The popular (yet more and more obsolete) ATMEGA328p of Arduino fame isn't fast enough for recording and further transferring all 16 channels. Especially not with a non-custom IIOD / ros-micro implementation.

3.3 Firmware part

TODO: Short 100-200 word intro about firmware

TODO: The no-OS framework

TODO: What is No-OS and why do weuse it? Main points: more than a Hardware Abstraction Layer; decouples hardware platforms, peripherals, libraries. While it adds an initial development overhead, it helps a lot with porting the firmware to other boards, sensors, or even software communication schemes.

3.3.1 Serial communication with host

In order to send data to the attached computer or carrier board, the de facto solution is using UART-over-USB, which exposes an USB interface to the attached host and UART to the microcontroller. Most commonly, microcontrollers have dedicated circuitry for UART transmit and receive, which allows the rest of the program to run concurrently to serial communication. This is implemented using two FIFO buffers, one for data pending to be sent from the microcontroller (TX FIFO) and one for received data ready to be processed by the program (RX FIFO).

De facto, most microcontroller development boards, including the MAX78000FTHR, MAX-Arduino, and the more common ATMEGA368P "Arduino-like" boards provide UART-USB translation via a dedicated IC, abstracting away the complexities of the USB communication stack.

UART modules have varying degrees of configurability of the baud rate, flow control, and framing. For most practical uses, though, including this project, one may just specify the baud rate, defaulting to 8N1 (8 data bits, no parity bit, 1 start bit, 1 stop bit) framing and no flow control. Typical baud rates include 9600 baud, a legacy of 20th century POTS modems (introduced by the V.32 ITU-T standard <**empty citation**>) and 115200 baud, common both in last-century modem systems, but also recently popularized by its default use in the Arduino development environment.

To achieve the required bandwidth for this project, a suitable baud rate must be chosen. Given the sample rate f_s, number of channels N, and number of bytes per channel sample d, and 8N1 framing (giving 10 baud/byte), the baud rate f_{baud} must be:

$$f_{baud} \geq f_s \cdot N \cdot d \cdot 10 = 1024 \cdot 16 \cdot 3 \cdot 10 = 491520 \text{ baud} \tag{3.6}$$

Additionally, choosing a baud rate larger than this minimum will allow for less timing concerns, especially when it comes to not overflowing the receive and transmit buffers.

Baud rate selection is not universal across devices and the firmware for each hardware platform must choose a rate that is both compatible with the hardware and that satisfies relation 3.6. In the case of the MAX78000FTHR, a rate of exactly 491520 baud can be achievend by using the IBRO (Internal Baud Rate Oscillator) of 7.3728MHz with a clock divider of 15, but [we] will use a divider of 9 for a "round" value of 819200 baud to have a good margin of error and not need such strict synchronization between the UART module and the rest of the program.

3.3.2 SPI Communication basic proof of concept (2024-05 single channel demo)

For initial validation of the concept on the AD4114, TODO: write about the PoC: 1 channel, no amplification, full sample rate, can see both EMG and ECG on it. can do nice graphs, can diagnose my RBBB

3.3.3 No-OS Driver

For easy portability to different microcontroller platforms and even other ADCs, I'm using the No-OS framework, which facilitates separating the general logic from product-specific code and settings.

3.3.4 IIOd integration

Defining the structure of the IIO context is a very important step as it solidifies a mapping between the device-specific functionality on the hardware side and the application-specific functionality on the software side. Choosing a versatile collection of attributes to configure the device and channels to structure the data is no trivial task with no unique solution. The exposed attributes may encode very low level details, as in find an AD part with a very bare-metal attribute mapping, or more high level ones. Choosing which configuration is to be specified on the software side and which configuration is to be derived from that on the firmware side of pula mea ce scriu aici... vreau sa zic ca nu o sa expun direct registri, ci o sa expun chestii mai high level: frecventa de mains hum (50/60), parametri de filtrare, poate sample rate daca o sa imi dau variante (posibil tho sa fie fix la 1k), si pe baza lor se ocupa firmware sa puna registrii cum trebuie.

For lower-level functionality, an IIO debug interface is also specified, which allows for direct manipulation of the ADC's registers. This shouldn't be normally used, but allows for total control of the device in case the user needs it.

1. Defining the desired iio context structure: what do we expose, in which formats, etc 2. Handler functions 3. Boom done, connect to it from software

3.3.5 Continuous and asynchronous data acquisition using IIO triggers

By default, the expected behaviour of IIO devices is to start acquisition on selected channels when requested (via an `OPEN` command), and stop and disable the channels once the requested number of samples is reached. In low sample rate scenarios, the host may request the next acquisition window very soon after the current one finishes, and assuming the time delay between windows can be minimized with respect to the overall window size, this could prevent missing any samples and only introduce a small amount of phase drift. Additionally, there are high-speed applications in which continuity between acquisition windows is not strictly required. On the other hand, with ourroughly 16kHz sample rate (cca. $60 - 62\mu\text{s}/\text{sample}$) and requirement to not miss any samples, this application has very little room for error in timing. As per [16], modern real-time linux setups have scheduler latencies in the tens and hundreds of microseconds, which realistically leaves no time to actually receive the current window's buffer and send a new acquisition request without missing any samples, even if the acquisition channels were left enabled.

TODO: figure, vertical stack of sequence diagrams of: normal iio polling with phase drift in slow ADC, fast ADC - loss of samples, trigger and buffer

With this general issue in mind, IIO defines triggers as software or hardware events that initiate data acquisition without requiring regular polling from the host. Data captured as a result of triggers is stored in circular buffers and provided with minimal latency whenever the host polls next.

With the AD4114 in continuous conversion mode, two trigger sources are apparent: using the $\mathtt{DOUT}/\overline{\mathtt{RDY}}$ behaviour in continuous conversion mode (as described in ??) or using a timer to poll the AD4114 at least as fast as the conversion rate. While the first option is more elegant and wouldn't do any unnecessary polling, it requires nonstandard operation of the SPI protocol by keeping CS active and monitoring the MISO ($\mathtt{DOUT}/\overline{\mathtt{RDY}}$) line, which isn't generally possible unless webitbang the SPI protocol or have access to more configurable IO options like on FPGA platforms or the Programmable Input and Output subsystem of RP2040 fame ([17]). The second option of setting up a timer interrupt which polls the AD4114 status register to check if a new conversion result is available

TODO: more :)

3.3.6 ROS2 on network-capable microcontrollers

In recent years, there has been a wave of developments in fast, network-capable, cheap microcontroller platforms, leading to products such as the Espressif ESP8266 and ESP32, or the Raspberry Pi Pico-W, which are capable of running ROS2 nodes all by themselves.

TODO: more :)

3.3.7 Digital filtering of the signal

Although the digital signal processing part of such biopotential measurement systems is very important for a full product, it is not the main topic of this thesis, and yet the chosen system architecture offloads most of the signal conditioning to the digital domain. The main source of noise is mains hum, with a fundamental frequency of 50Hz and various harmonics, which can all be removed using a notching filter, resulting in minimal loss of information in other portions of the frequency spectrum. The Qaulity factor Q of the filter controls how narrow the attenuated frequency bands are, and it can be adjusted. It poses a tradeoff between good noise rejection and signal quality, and is designed to be adjustable. Experimentally, we found values of around 5-10 to give the best waveforms for visual inspection. Figure X illustrates the tradeoff between small Q doing ??? and high Q doing ???.

Taking from scipy's implementation of an IIR Notching filter ('scipy.signal.iircomb'), I also use the structure described in cite: Sophocles J Orfanidis, "Introduction to Signal Processing", Prentice-Hall, 1996, ch. 11, "Digital Filter Design" thanks to its adjustable Q factor, numerical stability even at higher orders, and default rejection of the DC component of the signal, which are all very useful for the required signal conditioning. Parametrized by the the sampling frequency f_s, the cutoff freqiency f_0 (which must be an integer divisor of f_s), quality factor Q, the IIR filter $H(z)$ is computed as:

$$N = f_s/\omega_0 \overset{!}{\in} \mathbb{N}$$
$$\omega_0' = \frac{2\pi\omega_0}{f_s}$$
$$\omega_\Delta = \frac{\omega_0'}{Q}$$
$$\beta = \tan(N * \omega_\Delta/4)$$
$$a = \frac{1-\beta}{1+\beta}$$

$$H(z) = \frac{1}{1+\beta} \cdot \frac{1-z^{-N}}{1-az^{-N}} \tag{3.7}$$

TODO: check I transcribed these correctly and actually implement them myself! I only blindly copied these for now

3.4 Software part

3.4.1 ROS2 agent for non network-capable microcontrollers

In the case of microcontroller platforms that aren't network capable, including the MAX78000FTHR, the device itself won't have a means of directly connecting to the ROS2 network as a node. As such, it needs to pass along the data to another device that can. Two implementation variants are immediately evident: Using the micro-ROS framework, letting ros logic take over the serial device instead of having it communicate using the IIO protocol, and running a ROS-micro agent on the host - Using the ros-control framework and an IIO-ROS2 adapter on the host / carrier board.

Of which I chose the second because it doesn't require reconfiguring the serial port and allows for simultaneous use of ROS and IIO.

TODO: implement :)

3.4.2 libiio, pyadi-iio

ROS is for robotics, what about just doing some basic signal processing in python without setting it up as a ros node? Libiio has got you covered! Pyadi-iio adds a cherry on top, exposing the entire system as a very simple to initialize and query python object.

TODO: implement :)

4 Results and validation

To test the performance of the system:

- ECG is more difficult and as such a greater achievement. It has smaller amplitudes and very well-defined features, requiring more care all across the signal chain. - EMG is the main goal of the project, but it is more forgiving, because: - The signal has greater amplitude (about 10x? check both sources and experimental data!!!) - The signal features are not as well defined and are usually modeled like noise for signal processing.

4.1 Electrical validation

Using a waveform generator, we can compare this device's measurements with both: - Precision oscilloscope readings - Another biopotential DAQ system (Tassos')

TODO: more :) :)

4.2 Usage for EMG

TODO: more :) :)

4.3 Usage for ECG

TODO: more :) :)

Usage for EEG???

TODO: This would just be a cherry on top if I really don't have anything left to do in the last few days. A cool experiment would be trying out a quick and dirty N100 demo. Could have a handful of visible targets on the screen, have some orthogonal and time-uncorrelated activations for each, correlate EEG (single ended frontal-occipital) with each of the PRBS to determine which target the user is focusing on

5 Conclusion

We present a hardware + software solution for biopotential measurements of up to 16 single-ended or 8 differential channels, with TODO: specs, intended for easy and affordable maintainability, using off-the-shelf components and open source hardware and software, and being fit for

5.1 Future work

Among initial ambitions which unfortunately

6 Acknowledgements

- Mark Thoren

7 Bibliography

[1] M. Barbero, R. Merletti, and A. Rainoldi, *Atlas of Muscle Innervation Zones.* Springer Milan, 2012, ISBN: 9788847024632. DOI: 10.1007/978-88-470-2463-2. [Online]. Available: http://dx.doi.org/10.1007/978-88-470-2463-2.

[2] W. J. Esposito, "The ASPEN platform : Tools for mixed signal electronics, digital signal processing, and biomedical electronics," PhD dissertation, Department of Electrical Engineering, Stanford University, CA, USA, 2018. [Online]. Available: https://purl.stanford.edu/nq977kw3386.

[3] *OpenBCI - Open Source Biosensing Tools (EEG, EMG, EKG, and more)*, https://openbci.com/, Accessed: 2024-06.

[4] *BioAmp EXG pill*, https://github.com/upsidedownlabs/BioAmp-EXG-Pill, Accessed: 2024-06.

[5] *Myoware muscle sensors*, https://www.sparkfun.com/myoware, Accessed: 2024-06.

[6] P. M. Hopkins, "Skeletal muscle physiology," *Continuing Education in Anaesthesia Critical Care & Pain*, vol. 6, no. 1, pp. 1–6, Feb. 2006. DOI: 10.1093/bjaceaccp/mki062. [Online]. Available: https://doi.org/10.1093/bjaceaccp/mki062.

[7] R. E. Klabunde, *Cardiovascular Physiology Concepts*, 2nd ed. Philadelphia, PA: Lippincott Williams and Wilkins, Sep. 2011.

[8] V. J. Urick, K. J. Williams, and J. D. McKinney, *Fundamentals of microwave photonics* (Wiley Series in Microwave and Optical Engineering), en. Nashville, TN: John Wiley & Sons, Feb. 2015.

[9] F. Suarez, A. Nozariasbmarz, D. Vashaee, and M. C. Öztürk, "Designing thermoelectric generators for self-powered wearable electronics," *Energy & Environmental Science*, vol. 9, no. 6, pp. 2099–2113, 2016, ISSN: 1754-5706. DOI: 10.1039/c6ee00456c. [Online]. Available: http://dx.doi.org/10.1039/C6EE00456C.

[10] J. F. Kurniawan, A. B. Allegra, T. Pham, *et al.*, "Electrochemical performance study of ag/agcl and au flexible electrodes for unobtrusive monitoring of human biopotentials," *Nano Select*, vol. 3, no. 8, pp. 1277–1287, May 2022, ISSN: 2688-4011. DOI: 10.1002/nano.202100345. [Online]. Available: http://dx.doi.org/10.1002/nano.202100345.

[11] S. Chatterjee, R. S. Thakur, R. N. Yadav, L. Gupta, and D. K. Raghuvanshi, "Review of noise removal techniques in ecg signals," *IET Signal Processing*, vol. 14, no. 9, pp. 569–590, Dec. 2020, ISSN: 1751-9683. DOI: 10.1049/iet-spr.2020.0104. [Online]. Available: http://dx.doi.org/10.1049/iet-spr.2020.0104.

[12] M. Seiti, A. Giuri, C. E. Corcione, and E. Ferraris, "Advancements in tailoring pedot: Pss properties for bioelectronic applications: A comprehensive review," *Biomaterials Advances*, vol. 154, p. 213 655, Nov. 2023, ISSN: 2772-9508. DOI: 10.1016/

j.bioadv.2023.213655. [Online]. Available: http://dx.doi.org/10.1016/j.bioadv.2023.213655.

[13] S. Hou, D. Lv, Y. Li, *et al.*, "Highly stretchable self-adhesive pedot:pss dry electrodes for biopotential monitoring," *ACS Applied Polymer Materials*, Jun. 2024, ISSN: 2637-6105. DOI: 10.1021/acsapm.4c01233. [Online]. Available: http://dx.doi.org/10.1021/acsapm.4c01233.

[14] *8-channel, low noise, low power, 24-bit, sigma-delta ADC with PGA and reference*, AD7124-8, Rev. F, Analog Devices, 2023. [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/ad7124-8.pdf.

[15] *Single supply, multichannel, 31.25 kSPS, 24-bit, sigma-delta ADC with ±10 V inputs*, AD4114, Rev. A, Analog Devices, 2020. [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/ad4114.pdf.

[16] D. B. de Oliveira, D. Casini, R. S. de Oliveira, and T. Cucinotta, "Demystifying the real-time linux scheduling latency," en, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. DOI: 10.4230/LIPICS.ECRTS.2020.9. [Online]. Available: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ECRTS.2020.9.

[17] S. Smith, "How to initialize and interact with programmable i/o," in *RP2040 Assembly Language Programming*. Apress, Oct. 2021, pp. 177–200, ISBN: 9781484277539. DOI: 10.1007/978-1-4842-7753-9_10. [Online]. Available: http://dx.doi.org/10.1007/978-1-4842-7753-9_10.