UNIVERSITY OF THESSALY

DEPARTMENT OF COMPUTER SCIENCE

AND BIOMEDICAL INFORMATICS

Design and implementation of an electrocardiograph using the Arduino embedded platform

George Kantasis

THESIS

Supervisor
Anthony Aletras
Professor

Lamia 2013

Contents

Abstract

Electrocardiography is a common tool in modern medical practice. It monitors heart rate and rhythm and is used to diagnose a number of ailments. This thesis focuses on implementing the digital and analog components of such a device as well as to design the software necessary for sampling, interpreting, visualizing and storing the signal. In Section 1 we present the background information regarding the anatomy and the physiology of the heart as well as basic principles in amplifier circuits, filters, microcontrollers and Kohonen networks. Section 2 describes the design and implementation of the system following a block diagram of discrete modules that are then analyzed. Then we discuss the results of the system and the future perspectives it has. Finally in the appendices we present the Arduino, Java and Matlab code used to process the signal.

Περίληψη

Ηλεκτροκαρδιογραφία είναι ένα σύνηθες εργαλείο στην ιατρική πρακτική. Παρακολουθεί τον καρδιακό κύκλο και ρυθμό και χρησιμοποιείται για τη διάγνωση διάφορων παθήσεων. Αυτή η πτυχιακή εργασία επικεντρώνεται στην υλοποίηση των ψηφιακών και αναλογικών στοιχείων μίας τέτοιας συσκευής καθώς και στη σχεδίαση του απαραίτητου κώδικα για τη δειγματοληψία, την ερμηνεία, την παρουσίαση και την αποθήκευση του σήματος. Στο Τμήμα 1 παραθέτουμε τις απαραίτητες γνώσεις για την ανατομία και τη φυσιολογία της καρδιάς καθώς και τις βασικές αρχές κυκλωμάτων ενισχυτών, φίλτρων, μικροελεγκτών και δικτύων Kohonen. Το Τμήμα 2 περιγράφει τη σχεδίαση και υλοποίηση του συστήματος ακολουθώντας την ανάλυση ενός μπλοκ διαγράμματος διακριτών στοιχείων. Ακολουθούν τα αποτελέσματα της υλοποίησης και οι μελλοντικές προοπτικές του συστήματος. Τέλος παραθέτουμε τον κώδικα που χρησιμοποιήθηκε σε Arduino, Java, και Matlab.

Introduction

Electrocardiography is the recording and interpretation of the electrical activity of the heart muscle during the cardiac cycle. The resulting signal, the electrocardiogram (ECG), represents electrical events that are related to cardiac stimulation. It was first used in 1903 by Willem Einthoven, a Dutch physician and inventor, who received the Nobel Prize in Medicine in 1924 for his contributions to the field. Although the ECG does not provide direct information regarding the heart's pumping capability, it provides knowledge about the cardiac rhythm, the function of the cardiac conduction system and may indicate a variety of clinical conditions like ischemia, arrhythmia or AV block. ECG is an ubiquitous tool used for monitoring the heart condition of patients, and for the diagnosis of various cardiac and other related diseases. It is most often seen in intensive care units (ICU) in hospitals that require constant monitoring of the heart rhythm. The recorded ECG waveforms provide basic information about a patient's condition.

The original design of the ECG has received a number of modifications over the years so as to perform different intended functions. The cardiac stress test, performs ECG recording while the patient undergoes gradually increasing levels of physical exercise on a treadmill or a stationary bicycle or pharmacologically. This measures the heart's response to external stress and has a diagnostic value for conditions like myocardial ischemia or artherosclerosis. The ECG Holter monitor is another variant that is attached to the patient and continuously records the ECG signal over

his/her daily routine for a period of 24 to 48 hours. This allows the patient's physician to review the data and diagnose disease that would otherwise require a hospital environment. This continuous recording of the signal also allows the monitor to provide early alerts of emergency conditions like cardiac arrest, warn the user or even notify medical personel. The simplest variant of an ECG is a heart rate monitor that is a relatively simple medical device that simply counts the heart's rate in beats per minute. It is used mainly by athletes to keep track of their heart rythm while undergoing physical exersice.

The ECG also has applications in cardiac imaging techniques. The motion of the heart muscle causes motion artifacts which degrade image quality in CT and MRI. These artifacts can be prevented if the acquisition time is reduced so as not to include any notable motion; however, this usually results in poor image resolution. In order to improve resolution, several acquisitions are performed over the course of many heart beats and the acquisition data are combined in one image. The ECG signal is used to synchronize these acquisitions so as to occur at the same phase of the cardiac cycle. This is called Prospective Cardiac Gating. Retrospective cardiac gating is another method where acquisition is continuous over the course of several heart beats and the resulting images, one corresponding to each cardiac phase, are combined in a video movie of the beating heart. The design and implementation of a prospective ECG gating system is the main objective of this thesis.

Background

Biosignals

Signal is defined as the sequence of values of a depended variable (usually a physical quantity) that varies according to an independent variable (usually time or space). Biosignal is any signal that originates from living beings. It contains important information about the operation of the living system from which it originates. The three main categories of biosignals are mechanical, electrical and chemical signals.

Mechanical biosignals represent quantities such as pressure, flow, displacement etc., which are converted by special sensors. These sensors convert the physical quantity they measure into an electrical signal, which is more convenient to store and process. These sensors are not always simple in design or cheap. Also, in some cases these sensors are highly invasive.

Electrical signals are caused by the polarization and depolarization of neurons or muscle cells. They are characterized by the difference in ion concentration across the two sides of the cell membrane. This ionic membrane potential, when stimulated, can produce an action potential. Because the propagation of the action potential is done through ions, rather than electrons, these signals require the use of electrodes, which are simple and cheap. Measurements at the cellular level can be performed by using microelectrodes. In this case the signal source is the action potential itself. On a larger scale, the signal is the electrical field caused by the interference of all the action potentials from the cells neighboring the electrodes. The electric field can be propagated through a biological channel (the organism itself) and reach the surface of the body so that electrodes placed there can record it. This makes recording the signal fairly simple and non-invasive.

The Heart

The heart is the center organ of the circulatory system. It is responsible for pumping blood by means of repeated and rhythmic contractions through the blood vessels that to the entire body. This provides oxygen to the various tissues and removes carbon dioxide through the process of diffusion. The heart is separated into two pumps, the right side that pushes de-oxygenated blood coming from the body through the pulmonary circuit into the lungs and the left side that pushes oxygenated blood from the lungs into the systemic circuit.

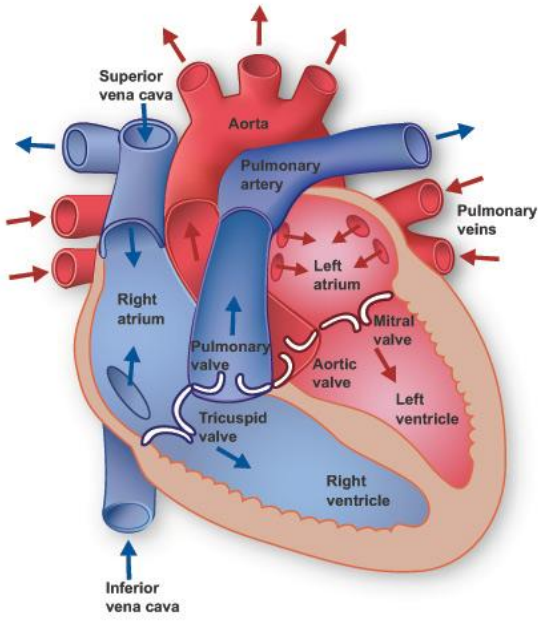


Figure 1: Anatomy of the heart. Source www.texasheartinstitute.org

Anatomy of the heart

The heart consists of four cavities (chambers): two ventricles, which pump blood directly to the circulation via the blood vessels, and two atria, which are responsible for filling the ventricles with blood just before the ventricles contract. The triscupid valve on the right side of the heart and the biscupid valve on the left are located between the atria and ventricles and allow the blood to flow in only one direction i.e. from the atria to the ventricles. Also, the pulmonary valve prevents blood from returning to the right ventricle while the aortic valve prevents blood from flowing into the left ventricle. The right atrium and ventricle are responsible for moving de-oxygenated blood from the body tissues to the lungs. The left atrium and the left ventricle pump the oxygenated blood received from the lungs to the body. The right atrium receives blood from the superior and inferior vena cava as well as the coronary sinus and the right ventricle sends the blood through the left and right pulmonary arteries. The left atrium receives blood from the left and right pulmonary veins and sends it to the left ventricle where it is pumped to the aorta. The ventricles do most of the work in terms of pumping blood and as such are thicker than the atria. Also, the left ventricle is thicker than the right ventricle to compensate for the increased workload of pushing blood into a high pressure system.

Physiology of the heart

The cardiac muscle consists of relatively thin and elongated muscle cells that run through the heart in bundles. These cells are surrounded by a

plasma membrane that connects neighboring cells in a dense formation known as intercalated disc. This causes the heart to act as a functional syncytium where the stimulation of one cell stimulates all the neighboring ones in a manner of chain reaction. This results in the entire syncytium to contract simultaneously.

All cells are surrounded by a membrane that has several mechanisms to control its permeability. One of these is the sodium-potassium ion pump that uses chemical energy from the cell to actively transport three sodium ions out of the cell and two potassium ions into it. By sending out more positive ions than it receives, the exterior of the cell becomes more positively charged than the interior which results in a potential difference on both sides of the membrane called membrane potential. In a stable state (polarized) the membrane potential of a myocardial cell is approximately -90mV. If for some reason (stimulation) this voltage goes over the threshold of -50mV to -45mV, various mechanisms of the membrane are activated that allow potassium ions to cross the cell membrane. As a result, the membrane potential rapidly increases (at a rate around 150V/s). The potential changes from about -110mV to +20mV (de-polarized state). A period of mild depolarization called plateau follows during which the membrane remains depolarized (unable to be stimulated again) and its duration is approximately 150ms for the atria and 300ms for the ventricles after which the repolarization of the cell completes rapidly. This plateau makes the depolarization duration (action potential) last 10-30 times more than any other striated muscle and prolongs accordingly the contraction of the muscle.

In order for the heart to function properly, the contractions of the atria and the ventricles need to occur in a synchronized and rhythmic manner. The atria contract approximately 130ms before the ventricles, something that ensures the optimal filling of the ventricles with blood before pumping it to the circulation. Also, there is an almost simultaneous contraction of the ventricles in order for the pressure to rise inside the chambers. Any deviation from this rhythm could lead to a number of heart ailments, most notably, arrhythmias. The synchronization of the cardiac contraction is done through a specialized conduction system responsible for initiating and propagating the action potentials throughout the heart muscle.

The Cardiac Conduction System

The heart contains three types of myocardial cells, atrial, ventricular and cells of the electrical conduction system. The electrical conduction system is formed by specialized myocytes that although they do possess some contractile filaments, they do not contract robustly but are responsible for the rapid propagation of the action potential. This system contains the

sinoatrial node (SA node), the internodal pathways, the atrioventricular node (AV node), the bundle of His and the Purkinje fibers.

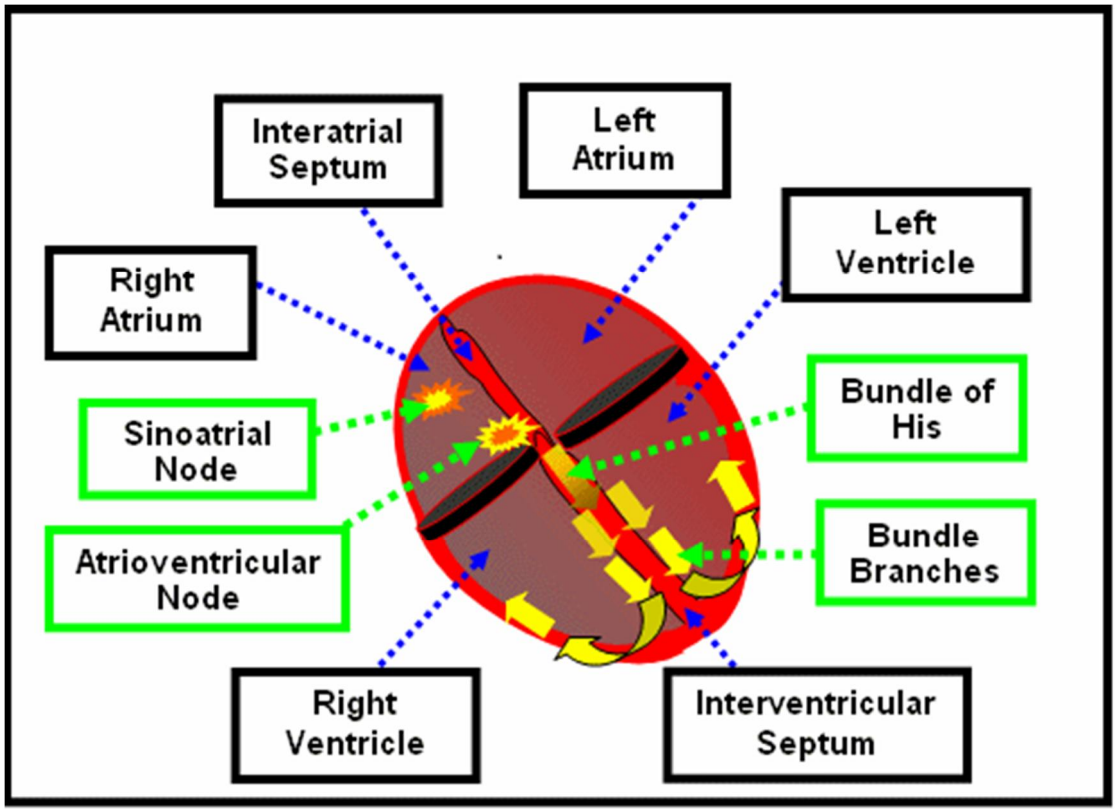


Figure 2: The cardiac conduction system. Source http://www.medic-ce.com

The sinoatrial node functions as the heart's natural pacemaker. This is where the stimulation impulse begins, the sinoatrial node is an impulse generating tissue located below the extrusion of the superior vena cava in the right atrium of the heart. It acts as the natural pacemaker of the heart and as the generator of the sinus rhythm. The internodal pathways are separated into anterior, posterior and middle tracks. These are the paths the impulse takes across the atria and towards the atrioventricular node. The Cachman bundle or interatrial pathway is a branch of the anterior internodal pathway that spreads the impulse from the right atrium to the left atrium. The fibrous ring, that separates the atria from the ventricles, does not allow the stimulation to propagate through it right into the ventricles.

The only way to the ventricles is through the atrioventricular node. It is located between the right atrium and the right ventricle near the interatrial septum and the opening of the coronary sinus. This node consists of small myocytes that conduct the impulse slowly and its function is to delay the propagation of it for approximately 130ms. Then the impulse reaches the His' bundles that are wide muscle cells that propagate rapidly across the interventricular septum and are separated in left and right branches that reach the apex of the respective ventricle.

These branches in turn give rise to thin filaments, the Purkinje fibers that distribute the impulse to the ventricular muscle.

The overall time delays of the action potential propagation are an important aspect for assessing the heart activity. In terms of time, the cardiac impulse needs 30ms to reach the AV node from the SA node where it is delayed by 130ms. Then it needs 40ms to traverse the His' bundle and from there it takes up to 40ms until the most remote parts of the myocardium are reached.

The Electrocardiograph

The electrocardiograph (ECG) is a method of monitoring and interpreting the electrical activity of the heart. As the action potentials traverse through the entire heart, the electrical field can be captured from the surface of the body using skin electrodes on both sides of the heart. The normal ECG consists of a P wave, a QRS complex and a T wave. The P wave is caused by the impulse propagation across the atria and is followed by atrial contraction. The QRS complex is the stimulation of the ventricles and is followed by ventricular contraction. Finally the T wave is caused by the re-polarization of the ventricles and it is followed by ventricular relaxation. The re-polarization of the atria occurs at the same time as the ventricle depolarization and is therefore overlaps with the QRS complex.

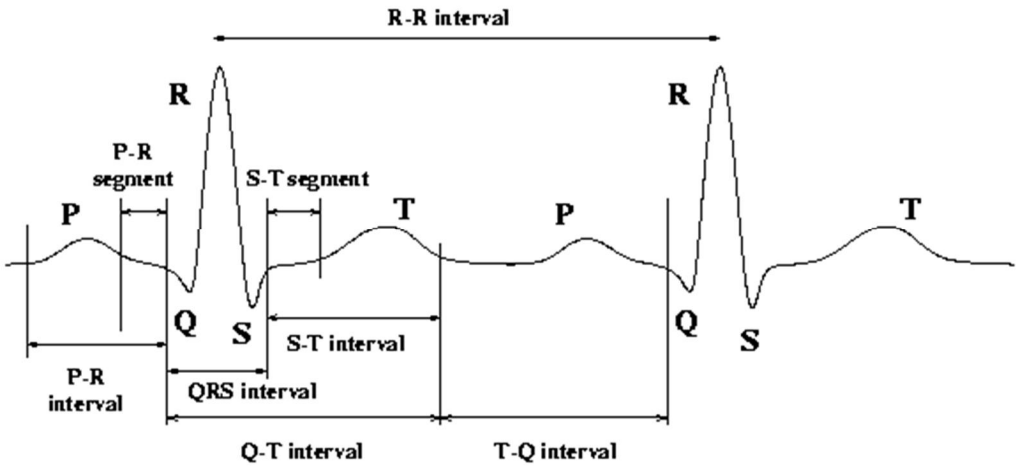


Figure 3: Normal ECG and associated intervals. Source http://bme.elektro.dtu.dk

During normal heart function, the ECG signal can show us a number of useful intervals and segments as well as the amplitudes of the corresponding waves. Normally the heart of an adult is depolarized 60 to 90 times per minute. A depolarization rate lower than this, is called sinus bradycardia, while a higher rate is called sinus tachycardia. The heart rate of the normal newborn is much higher than that of an adult. The heart

rate can be calculated by measuring the intervals between subsequent R waves.

The P wave's duration is measured from the beginning of the P wave to its end. Normally it is less than 120ms while in newborns it is less than 80ms. This is the time required for the depolarization wave to spread through the entire atrial myocardium and to reach the AV node.

The PR interval represents the amount of time required for the depolarization wave to spread from its origin, the SA node, and through to the AV node where it's delayed to the His bundle and the Purkinje fibers. It is measured from the beginning of the P wave to the beginning of the QRS complex (which makes it actually "the PQ interval"). When the Q wave is absent the measurement is made until the beginning of the R wave. The difference between the intervals as measured to the beginning of the Q wave, and those measured to the R wave, is usually 20-40ms. The PR interval is around 130ms in adults and less than that in children. The time interval between atrial depolarization and repolarization is also measurable and it is about 150ms to 200ms.

The duration of the QRS complex represents the amount of time required for the depolarization of the ventricular muscle. It is measured from the beginning of the Q wave to the end of the S wave normally with a duration is 250ms to 350ms.

The ST segment represents the time interval between ventricular depolarization and repolarization. The depolarization process ends with the end of the QRS complex and the repolarization begins with the T wave. The ST segment duration is determined by measuring the time interval between the end of the S wave and the beginning of the T wave and measures around 200ms.

The QT interval represents the time required for depolarization of the ventricles, the amount of time during which the ventricles remain contracted and the amount of time required for their repolarization. This interval represents the duration of electrical systole, which is different from the duration of mechanical systole. The QT interval is measured from the beginning of the Q wave of the QRS complex to the end of the T wave. The duration of the normal QT interval is usually around 250ms.

Finally the T wave lasts about 150ms. The area encompassed be the T wave is usually 60% the area encompassed by the QRS complex and the upstroke of the T wave is generally less steep than the downstroke.

ECG leads

The ECG is sampled by electrodes that are placed in specific locations in the surface of the body. The sign of the signal indicates the orientation of the electrodes and the direction of the current in the heart.

Limb leads

The three basic leads are recorded by limb electrodes. In lead I the electrodes record the potential difference from the right arm to the left arm. In lead II the electrodes are recoding the voltage from the right arm to the left leg. In lead III the left arm and the left leg are used. Einthoven's triangle (Figure 6) shows the layout of the three limb leads on the human body. The corners of the triangle represent the two arms and the left leg while its sides correspond to the leads and its respective connections. Einthoven's triangle represents a closed electrical circuit that surrounds the heart. It is noted here that while the polarities of leads I and III are clockwise (from negative to positive) the polarity of lead II is counter-clockwise. Einthoven's law states that at any time, the sum of leads I and III equals the value of lead II. This is evident if we consider Kirchhoff's law of voltages in the circuit described by the triangle. Therefore if at any time we know the values of two leads we are able to calculate the value of the third.

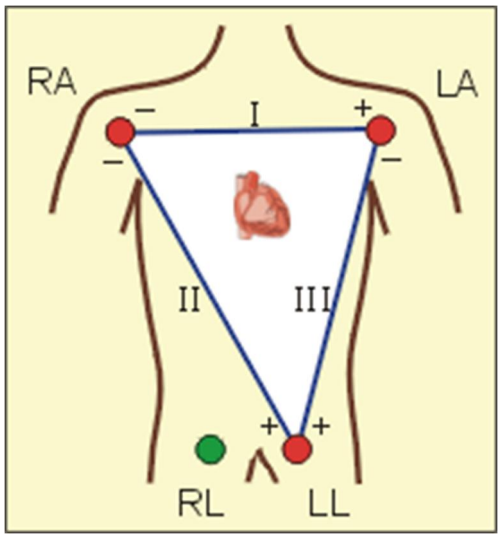


Figure 4: Einthoven's Triangle. Source http://www.cvphysiology.com/

Amplifiers

An amplifier is an instrument that increases the amplitude of a signal by means of a power supply. A basic characteristic of an amplifier is the output to input ratio, which is called gain.

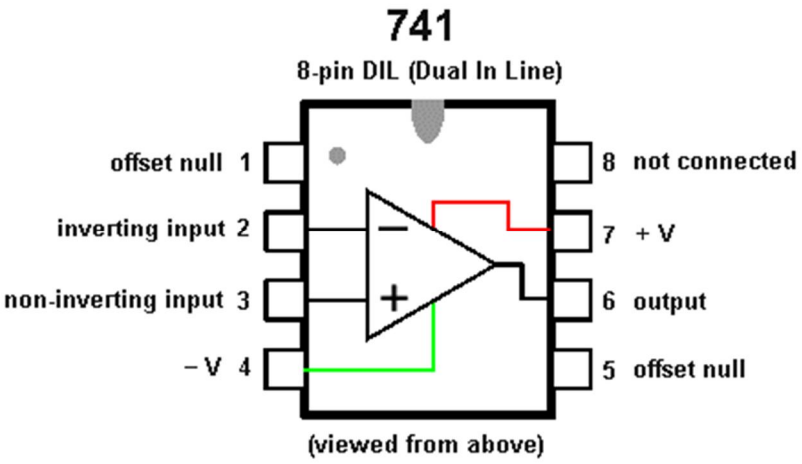


Figure 5: Pinout of the 741 operational amplifier. Source http://www.talkingelectronics.com/

Operational Amplifier

The operational amplifier is an elementary analog electronic device that amplifies voltage signals. It is called “operational” because of its ability to perform various mathematical operations upon analog signals given the proper setup e.g. addition or subtraction. Operational amplifiers were developed as early as 1960 as they performed better than vacuum tubes.

An operational amplifier has two inputs u_+ and u_- and one output u_o. Also it has positive (V_{CC}) and negative (V_{EE}) supply lines that are symmetric (±5V, ±12V etc) which determine the maximum positive and negative cut-off voltages. When the output is between these two values, the amplifier is said be functional and otherwise, the amplifier is said to be in saturation mode. At any time we have $V_{CC} \geq u_o \geq V_{EE}$. For simplicity reasons during design, we may omit the supply lines from the amplifier representation.

The basic characteristics of an operational amplifier are the open-loop gain (A_{OL}), the input resistance (R_{In}) and output resistance (R_{Out}). Other significant characteristics include the common mode rejection ratio (CMMR), max output current, bias voltage and unity gain bandwidth. An ideal operational amplifier has infinite common mode rejection ratio, infinite open loop gain, infinite input resistance and zero output resistance. In practice these values are approximations. The output of the amplifier is given as $u_o = A_{OL}(u_+ - u_-)$.

The operational amplifier does not amplify equally across all frequencies. Its ability to amplify across frequencies is described by the unity gain bandwidth. An operational amplifier can provide similar gain to signals of a limited bandwidth. Frequencies above a specific cut-off point are rejected as the amplifier functions itself as a low-pass filter. This bandwidth is directly related to the amplifier’s gain as the bandwidth-gain

product remains constant. Therefore, the larger the gain, the smaller the bandwidth the device can deliver that gain.

Since not all applications require infinite gain, it is not necessarily a desired characteristic in amplifiers. Infinite gain means that even the smallest voltage differences between the two inputs will be multiplied by infinity and therefore will reach the saturation voltages of the amplifier. This means that the op amp will operate in saturation even for the smallest voltage differences between. In such cases even random noise will be enough to cause the output to swing between the two saturation voltages. This open loop gain is constant and is determined by the circuit's manufacturer. It has a large value and is a limiting factor when it comes to designing a specialized system. Finally because of the unity gain bandwidth considerations, the amplified bandwidth of the open loop will approach zero. In order to control and limit the gain of the amplifier so that it complies with the application's requirements and also so that none the desired signal's frequency components are attenuated because of unwanted filtering, negative feedback is used in the circuit. The two most prominent operational amplifier designs utilizing negative feedback are the inverting and the non-inverting.

Non inverting amplifier

In the non inverting amplifier setup, as seen in the figure 8, the output is connected through a voltage divider to the inverting input while the input signal is supplied to the non-inverting input. This way the amplifier's transfer function becomes $u_o = \left(\frac{R_F}{R_I} + 1\right) u_i$ The factor $\frac{R_F}{R_I} + 1$ is called closed loop gain or A_{CL}. This way the amplifier's gain is adjusted using the proper resistors in the feedback loop.

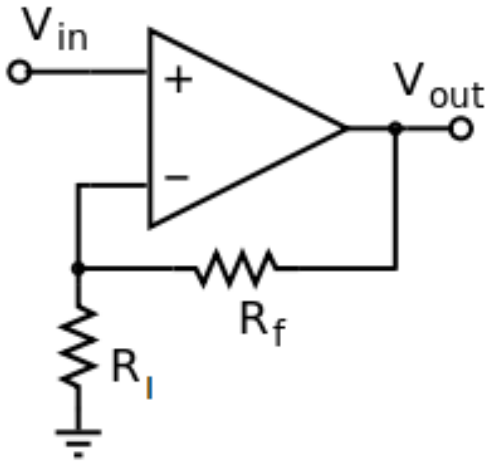
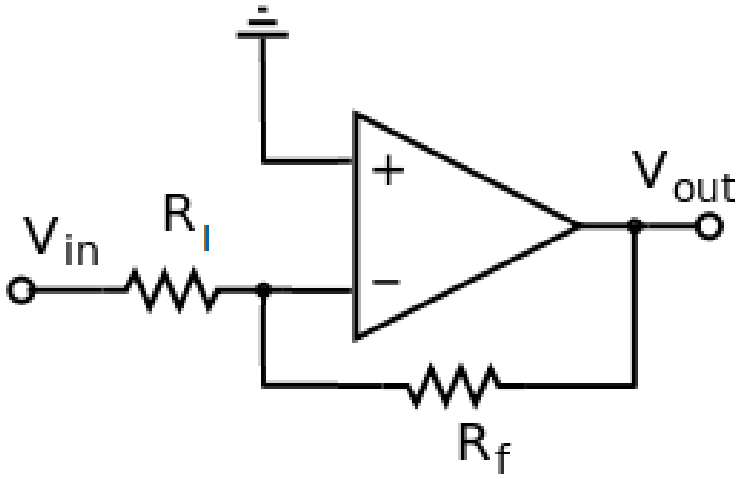


Figure 6: Non-Inverting Amplifier

Inverting amplifier

The inverting setup is similar to the non-inverting one. However, the output is connected with the inverting input through R_F and through R_I with the signal source. The non-inverting input is connected to the ground. Here the transfer function is $u_o = -\frac{R_F}{R_I}u_i$. The closed loop gain of this setup is $-\frac{R_F}{R_I}$ and it is negative because it inverts the voltage that is applied to the input. The gain can be controlled through proper selection of resistors



.

Figure 7: Inverting Amplifier

Voltage follower or Buffer

The voltage follower is a special case of non-inverting amplifier widely used in sensor and data collection applications. The reason for using this design is the buffer's high input resistance combined with a close to zero output resistance. Here the output is directly connected to the inverting input and the signal is introduced to the circuit through the non-inverting input. This can be interpreted as zero R_F and infinite R_I. Therefore, the gain from the type $\frac{R_F}{R_I} + 1$ becomes unity. The goal of this setup is to take a signal and propagate it as it is without loading the previous stages. Any load connected to the output of the voltage follower will draw current from the operational amplifier and not from the signal source. The driving voltage source will be barely encumbered because of the very large input resistance of the operational amplifier that lets no notable current flow through it.

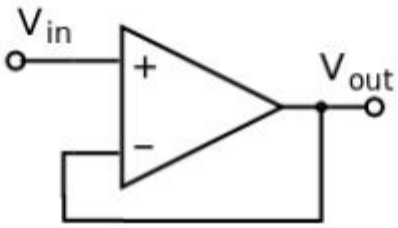


Figure 8: Voltage Follower / Buffer

Differential amplifier

The differential amplifier calculates the difference between two input signals and then multiplies that difference by the gain. This configuration has two inputs, unlike the previously described configurations. Four resistors determine the overall gain. These resistors create a voltage divider before the non-inverting input for the u_1 input signal and a negative feedback loop as in the inverting configuration for the input u_2 in the inverting input. The output is given by the type $u_o = \frac{R_2+R_4}{R_3+R_1}\frac{R_3}{R_2}u_1 - \frac{R_4}{R_2}u_2$. This function is more practical to use when $R_1 = R_2$ and $R_3 = R_4$ so that it becomes $u_o = \frac{R_3}{R_1}(u_2 - u_1)$, where the gain of the differential amplifier setup is $(\frac{R_3}{R_1})$.

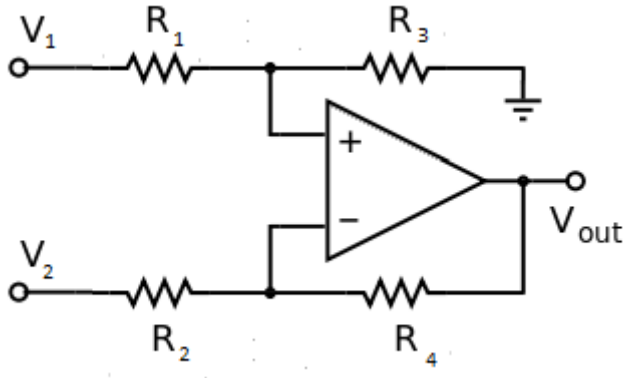


Figure 9: Differential Amplifier

Instrumentation amplifier

The instrumentation amplifier comes to cope with the differential amplifier's low input resistance. For this reason a voltage follower is added before each input of the differential amplifier. The followers can be substituted by non-inverting amplifiers, retaining the same result. This

allows for the addition of another stage of amplification in this configuration. Also two R_I resistors can be connected between them the two op-amps in series omitting the ground connection as a result of the virtual ground. Simply by changing the value of this resistor the gain of this amplification stage can be varied. This configuration is an instrumentation amplifier and is usually used instead of the plain differential amplifier because of the higher input resistance, better common mode rejection ratio and ease of adjusting the gain.

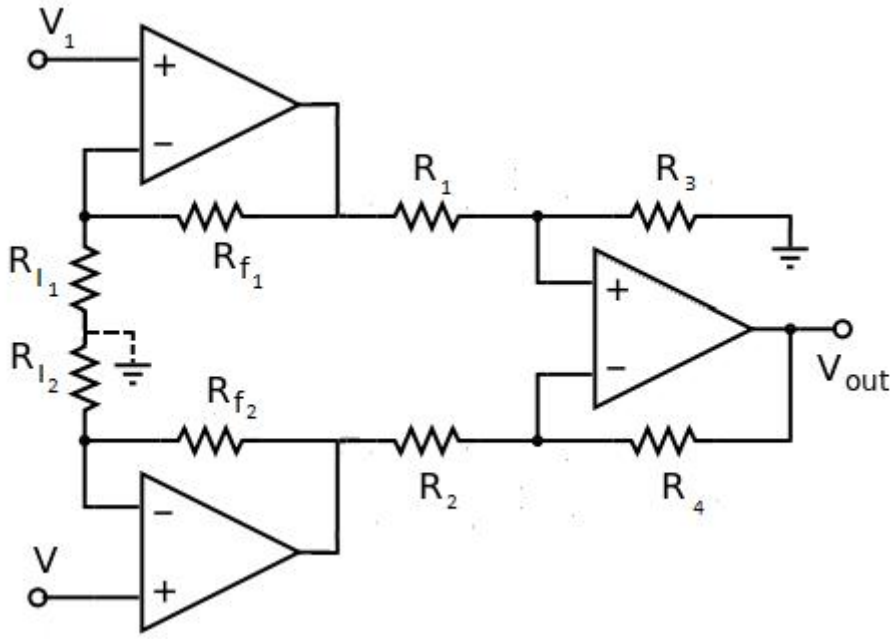


Figure 10: Instrumentation Amplifier

Filters

A filter is an electronic device or a mathematical function that removes unwanted frequency components from a signal and isolates useful parts located in a specific frequency band. The main categories of filters are analog and digital although they may be also categorized as active or passive.

An analog filter uses electronic components (capacitors, inductors and operational amplifiers) to perform its function. These circuits are widely used for noise reduction, video and audio amplifiers and radio communication applications. The design techniques of analog filters are simple and predetermined depending on the requirements of each application. However, their implementation or any kind of alteration must be done at the hardware level.

A digital filter uses a processor to perform the calculations to sampled values of the analog signal. This processor may be a general purpose

computer or even a microcontroller. The program that implements the filter is stored in the processor's memory and that makes it easy to change. A digital filter is able to be reprogrammed as needed without any change to the underlying circuitry. Nevertheless the use of this kind of filter is only usable only after the sampling stage; therefore we cannot rely on it to prevent signal aliasing.

The simplest filters are roughly divided in four categories depending on their frequency response.

- Low-pass filters: They preserve low frequency signal components and attenuate-reject components of higher frequency. The frequency in which the transition from preserve to reject is done is called cut-off frequency and in an RC (resistor-capacitor) filter, is calculated as $f_c = \frac{1}{2\pi RC}$

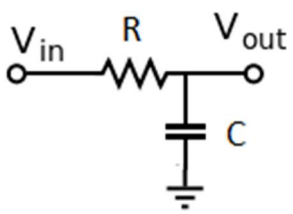


Figure 11: A passive RC low-pass Filter

- High pass filters: These are complementary to low-pass filters. They reject low frequencies and preserve higher ones. The cutoff frequency is calculated by the same equation as above.

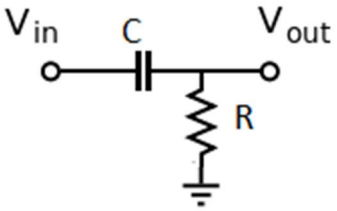


Figure 12: A passive RC high-pass filter

- Band-pass filters: Preserve the signal components that are within a certain frequency band (called the pass-band) and reject all others. This is a composition of a high-pass filter with a cut-off frequency at the low end of the pass-band and a low pass filter with a cut-off frequency at the high end of the band, in series. The difference of the two cut-off frequencies is called the band-pass filter's bandwidth.

- Band-stop filters: These are the complementary filters of band-pass filters. Instead of preserving a certain band, they reject it by letting all other frequencies pass. This is a composition of a high-pass filter with a cut-off frequency at the high end of the pass-band and a low pass filter with a cut-off frequency at the low end of the band, in series.

- Notch filters: These filters are essentially band-stop filters with a very narrow bandwidth. They are designed to block a very specific frequency, e.g. 50Hz interference arising from the mains power supply.

Filters are also categorized as active and passive. Passive filters consist only of passive electronic components (capacitors, inductors, resistors) whereas active filters consist of active components as well (transistors, operational amplifiers). This allows the filter to operate as an amplifier as well, increasing the amplitude of the input signal.

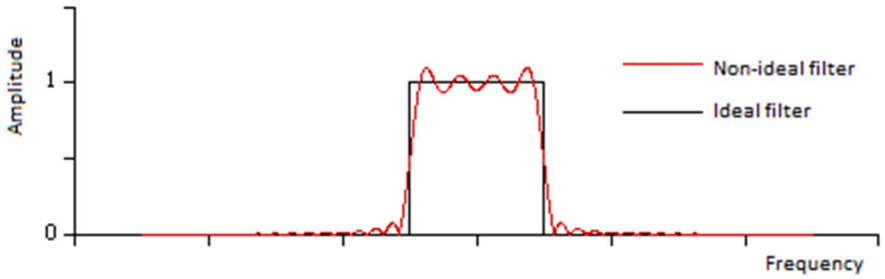


Figure 13: Ideal (black) and non-ideal frequency responses of filters. Source http://paulbourke.net/

Ideal filters do not affect the preserved components of the signal at all while completely rejecting the rest but in practice this is not possible. Figure 17 shows the Bode plot of an ideal filter (black) where gain is constant in the pass-band (equal to unity in the case of passive filters or greater than one in the case of active filters), zero in the stop-band and we see no ripple present in neither the pass-band nor the stop-band, as the amplitude remains the same. Also at $f=f_c$ we see a vertical drop of the curve. The red line represents a more practical approach to a filter's response as ripples in the stop-band and the pass-band, are evident and the transition from one band to another is not vertical. Filters generally transit from the pass-band to the stop-band in a rate of 3dB per octave (or 20dB per decade). Those are called first order filters. If two first order filters would be connected in series, the drop would be steeper i.e. 6dB/oct and the resulting filter would be called a second order filter and so on. This way we define the cut-off frequency as the frequency at which the output-input ratio becomes $\frac{\sqrt{2}}{2} = 0.707$ or $20\log(0.707) = -3dB$.

Analog to Digital Conversion

An analog to digital converter (or ADC) is a device used to convert a continuous analog voltage signal to a series of digital values, thus employed in digitization applications. This process consists of three discreet steps, sampling, quantization and coding.

Sampling is the conversion of a continuous time signal to a discreet time signal. During sampling the signal's amplitude is measured in constant time intervals. The sampling period (i.e. the time between two samples) is called T_s and its inverse f_s is called sampling frequency. Note that a spike in the signal, occurring right between two consequent sampling times, will be completely missed in the recorded signal regardless of amplitude. Therefore important information could be missed if the sampling frequency is not sufficiently high. The Nyquist-Shannon sampling theorem provides us with an answer as to just how high the sampling frequency should be.

Nyquist-Shannon sampling theorem

According to the Nyquist-Shannon sampling theorem, a continuous function (real analog signal) can be completely described by a discrete sequence (digital signal) given that the function does not contain frequencies higher than f_{max} and the sampling frequency is at least 2 f_{max}. Therefore the function can be perfectly reconstructed by the sequence and no information is lost because of the sampling process. Therefore $f_s >= 2f_{max}$ where f_s is the sampling frequency and f_{max} is the maximum frequency of the signal.

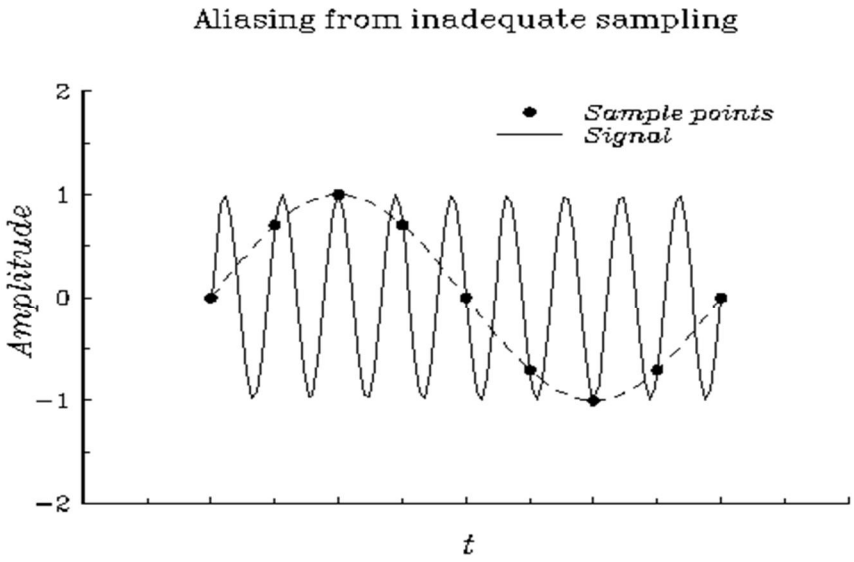


Figure 14: The effect of aliasing. The dashed line represents how the signal is read by the computer when the sampling frequency is low. Source http://www.phys.virginia.edu/

When the sampling frequency is less than double the max frequency of the signal, we see the aliasing effect. This is when the high frequency content, which is higher than half the sampling frequency, appears as low frequencies erroneously. When this happens the signal cannot be reconstructed as the original lower frequencies are irreparably corrupted. The aliasing effect is avoided when the signal has finite frequency content i.e. when there is a maximum frequency with no other frequencies higher than that present. In practice though, finite signals in time domain have infinite frequency domain. Also, thermal noise can extend the frequency content of an otherwise band-limited signal as far as infinity. The solution to this problem is to select a cut-off frequency for the analog signal to be sampled. Using a low-pass filter to attenuate all frequencies above this cut-off threshold should not considerable affect the original signal but will prevent aliasing during the digitization. This is the anti-aliasing filter and should always be present in the design of sampling systems.

Quantization is part of sampling and it refers to the mapping of a continuous range of values of the original signal into a discrete set of values. Usually the method used is rounding or thresholding. This process is inherently irreversible as the difference between the original value and the quantified one is lost. This difference is called quantization error. When the number of quantization levels is large enough, this error essentially forms quantification noise due to its stochastic behavior. The error can be described by $q = \frac{max-min}{l}$ where max and min are the

maximum and minimum values and l is the finite number of stages. The finite range of values is defined by the maximum and minimum values the quantifier can operate with. Most usually the min value is 0V and the max value is referred to as V_{ref} (reference voltage).

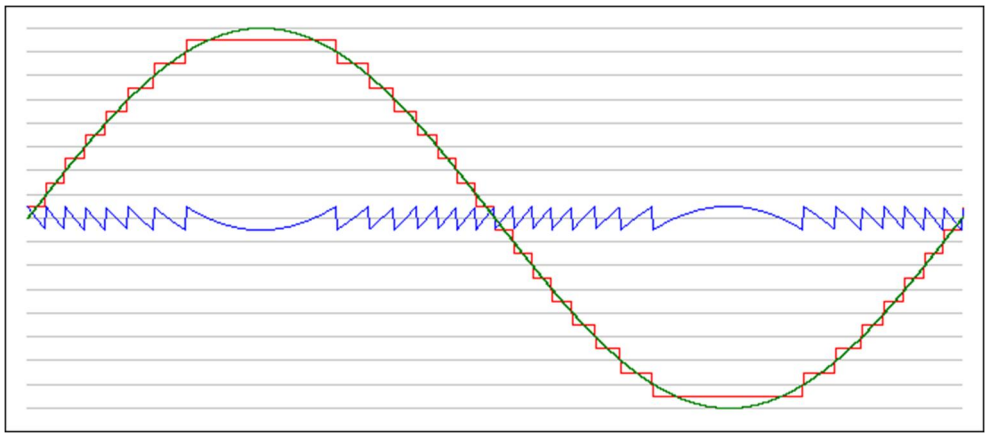


Figure 15: Representation of the analog signal (green) quantified signal (red) and the quantization error i.e. the difference of the two (blue) in the time domain. Source http://www.beis.de

Coding is the representation of the quantified signal into binary data. Usually the number of stages in the quantization step is selected according to the coding bits the ADC supports, as a power of 2. Note that the smallest difference the ADC may record, represented by successive binary values, is the same as the quantization error and for the default range of values is described as $q = \frac{V_{ref}}{2^n - 1}$ where n is the number of coding bits.

There is a variety of ADC types most common of which are the following: Direct-conversion ADCs utilize a large quantity of comparators, sampling the signal in parallel. This allows very fast sampling rates but the circuit cost increases exponentially with each bit of resolution added. Successive approximation ADCs use one comparator to compare the input signal to a successively narrowed reference voltage, each time reducing internally the reference voltage range by half. This allows for an inexpensive circuit but results in slower sampling rates as each successive bit in resolution results in one more comparison. Ramp-compare ADCs utilize a timed saw wave that linearly increases and then drops to zero when it reaches the reference voltage. When the input signal matches the saw wave amplitude, the comparator signals the timer to store its value as the output of the ADC.

Shock Hazard and Safety

A concern when dealing with electronic devices that come into contact with the human body is the danger of electric shock. Electric shock occurs when the body becomes part of a low impedance circuit and current flows through the skin, muscle and tissue. The electric shock describes the harmful effects of electricity to the organism. Its effects are highly depended on the magnitude of the current, its frequency, its duration as well as other factors. It ranges from imperceptible to incapacitating for example to the point where the victim may be rendered unable to let go of the exposed wire because of the muscle contraction caused by the electricity. Electricity can also cause serious burns, interferes with neuromuscular cells altering membrane potential but more importantly, may induce ventricular fibrillation or permanent heart damage. Therefore, reducing the danger of a potential shock is important especially since the device in question, the electrocardiograph, is associated directly with the heart and patients may even have underlying heart disease.

Most electronic devices can be powered by low direct voltage sources (less than 10V) or by low voltage isolating transformers. Even though electric cannot be guaranteed to be prevented in this way, this is a safer practice than using higher voltage devices. Another way of protecting the patient is via electrical isolation. Electrical isolation is achieved through three main methods: transformer isolation, optical isolation and capacitive isolation.

Isolation methods are considered safer since they implement an electrical barrier that prevents current from the other side of it to reach the patient. With the isolation transformer method, the signal is frequency or amplitude modulated through an innate oscillator and with limited bandwidth up to 30 KHz. Then, a transformer is used to transfer the signal across the electric barrier. The optoisolation method uses an LED and a photodiode to create the barrier. A modulator is not required since the signal is transmitted as light. A photodiode at the input side may be used as feedback for improving linearity. The capacitive method modulates the input voltage and frequency so that the signal is sent across a ceramic capacitive barrier. There is no feedback needed but power sources at both ends are required.

Arduino

Arduino is an open-source electronics prototyping platform. It is based on the Atmel 8-bit AVR microcontroller family that features functionalities like general-purpose I/O pins, serial communication, 3 timer/counters and a 10-bit analog to digital converter. Arduino started in 2005 as a project for students at the Design Institute of Ivrea, Italy. It can be programmed using the Arduino IDE, a cross-platform application written in java, with a

language based on C. Instead of void main() the Arduino software implements the void setup() function that is executed once on power-up, and void loop() that runs continuously as long as the board is powered.

Inline assembler

The Arduino IDE is a variant of the C programming language and as such a high level programming language. This high-level language provides many conveniences like abstraction, encapsulation and control flow, but it suffers when it comes to processing speed. Although not directly noticeable, processing speed is crucial in real-time medical applications where the full capabilities of the RISC microprocessor must be utilized.

Programming a microprocessor in assembly language is guaranteed to optimize the performance but the code quickly becomes large and difficult to manage and debug. In this project, an inline assembler feature of the IDE is used, which allows for the injection of assembly instructions directly into the C code wherever high performance is needed. The non time-critical parts may remain in Arduino's high-level C-like language while the real-time processing is coded in assembly.

Interrupts

Interrupts are a control flow mechanism present in most microcontrollers. They are used to interface the microcontroller with asynchronous peripherals like counters, analog-to-digital converters, mice, serial connection etc. An interrupt request (IRQ) refers to an external event that calls for a specific routine to be executed as soon as possible. This routine, called interrupt service routine (ISR), is a high priority function that suspends the execution of the main program at an unpredicted point of execution and returns to that point after the ISR has finished serving the interrupt. This requires that the system stack be initialized so that the program counter and the status register at the time of the interrupt can be saved in the stack. After the ISR finishes executing the program counter and the status register can be retrieved from the stack and the main program can continue execution normally.

The complete sequence of events associated with interrupts is generally described as followed

1. The peripheral notifies the system (asynchronously) and requests a certain routine to be called. The request includes the peripheral's identity in order to execute the proper corresponding ISR
2. The CPU finishes the instruction it was executing while the IRQ was received

3. The program counter and status register are pushed into the stack
4. The program counter is loaded with the address of the ISR of the corresponding IRQ (interrupt vector)
5. The ISR is executed
6. The program counter and status register are popped from the stack and the main program resumes execution

Timer/Counter

A timer is a device that counts time using the internal system clock or a prescaled version of it. A counter is like a timer but instead of counting time, it counts events (rising or falling edge) of an external signal to a specific input pin. Timer/counters are capable of producing interrupt requests in the microcontroller in the events of overflow or compare match. Compare match interrupts occur when the counter register value equals (matches) the value of another register, the compare register. These interrupts may be enabled or disabled through specific bits in the timer/counter control registers.

The ATMega328 timer/counter has four distinct operation modes that define the behavior of the timer/counter and the output pins. In Normal Mode the timer/counter counts upwards without any intermediate clearing. When the count reaches 255, it overruns to 0 setting the output compare pin. Clear Timer on Compare match (CTC) is the mode used where a secondary register is used to store a "TOP" value and when the count reaches it, it is reset. This feature allows greater control over the output frequency. Fast Pulse Width Modulation (fast PWM) mode, the output pin produces a PWM signal with duty cycle defined by a secondary register. Finally Phase Correct PWM mode generates a high resolution waveform using a dual-slope operation unlike the single-slope operation used in fast PWM.

ADC

The ATMega328 features a successive approximation ADC with 8-channels and 10-bit resolution (±2LSB accuracy). This means that the output range is between 00000000 and 11111111 in binary (0 to 1023 in decimal). Therefore the quantization error is $\frac{5}{2^{10}-1}$ = 4.88mV. Also, the ADC has an inherent prescaler that generates a proper ADC clock frequency from the microcontroller's clock signal. In order to maintain 10-bit resolution, the ADC must use a clock signal between 50 and 200 kHz. Higher clock frequencies result in faster samples but lower resolution. A typical conversion lasts 25 ADC clock cycles while subsequent conversions last 13 cycles. This is due to initialization of the ADC hardware needed before the

first conversion. The operation modes of the ATMega328 ADC are single conversion, free running, external interrupt request and timer/counter interrupt request modes.

The RS-232 Standard

In general, information transferred between data processing equipment and peripherals is in the form of digital data which is transmitted in either serial or parallel mode. Serial transmission involves sending of data one bit at a time, over a single communications line. In contrast, parallel communications require at least as many lines as there are bits in a word being transmitted. Serial transmission is beneficial for long distance communications, whereas parallel is designed for short distances or when very high transmission rates are required.

The RS-232 Standard is a data transfer method through a serial connection between data terminal equipment and data communications equipment. It is the most common way of data transmission used in microcontroller design. The RS-232 link is based on bitwise asynchronous serial data transmission and is divided in two parts. The first part describes the way in which the original byte of data is arranged before transmission. The second part describes the way the serial data are propagated across the communication lines.

The way in which the original data are arranged before transmission is not exclusive to the RS-232 protocol. Other protocols like RS-422, RS-425 and RS-485, all utilize the exact same method. All these protocols are only different in the way they propagate the serial data. In the following figure we see the arrangement of data bits in the original byte and the corresponding rearrangement. A start bit is added to the beginning of the sequence and a parity bit and up to two stop bits are added to the end. Finally the order of the data bits is reversed, as it now begins with the least significant bit (LSB) and ends with the most significant bit (MSB).

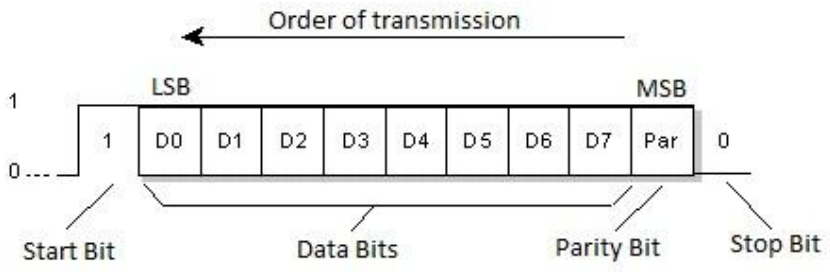


Figure 16: RS-232 Bit order

After the proper preparation of the data, what matters is the way the data are going to be transmitted, the voltage levels of each logical state of the signal, as well as the time window of each transmission. The transfer-receive units work in predetermined baud rates or bitrates the most common of which are 200, 600, 1200, 1800, 200, 2400, 4800, 9600, 19200 bits per second. The voltage levels for each logical state is +3 to +12 Volts for HIGH and -3 to -12 Volts for LOW. It is apparent that this is incompatible with most microcontrollers' TTL voltage levels (0V for LOW and +5V for HIGH) but it is possible to make a functional connection without explicit conversion. However, this practice is not completely compliant with RS-232 protocol and can lead to potential problems. If we full protocol compliance is needed (ex a long-distance wire is needed) then a specialized RS-232 to TTL converter should be used (such as Maxim's MAX232 or MAX212).

The original RS-232 protocol is quite thorough and fairly complicated as it deals with management of multiple signals. In the simplest form of bidirectional serial communication we need at least 3 pins. These are RX, TX and GND. The rest of the signals complementing the RS-232 requirement correspond to handshake signals for a safer and more reliable transmission.

Methods

ECG QRS detection

In order to implement ECG gating to an imaging system, a QRS detection technique must be performed on the sampled electrocardiogram so as to trigger the imaging device. This way the QRS complex becomes a landmark on the signal. This can then be used as a point of reference for the timing of the heart muscle, subsequent detection of P and T waves and computing the PR interval, QT interval etc.

Band-pass filtering

The QRS complex consists of steep slopes and sharp edges and as a result contains high frequency components in contrast with the P and T waves. Therefore a high-pass filter should be able to attenuate the low frequencies of P and T waves effectively isolating the range of frequency components corresponding to the QRS complex.

Thresholding

This method involves setting a certain threshold to be compared with one of the signal's attribute. If the threshold is reached or surpassed, the presence of a QRS complex is detected. A constant amplitude threshold is not recommended because of the signal's baseline drift across multiple applications. This could potentially result in erroneously detecting either P or T waves or not trigger at all at the QRS complex. Thus a dynamic threshold should be implemented that depends on the signal at hand. Also, thresholding the first time derivative of the signal could solve this issue as this method uses the signal's slope as an attribute instead of its amplitude. In this case though, a false positive may occur by spike artifacts in the signal due to external noise. To counter this effect a moving window integrator is used on the signal's derivative to find the area under the curve across 16 points, to provide us with the points of a sustained ramp.

Kohonen Networks

A Kohonen network or self-organizing map is a classification method using neural networks and competitive unsupervised learning. Each neuron of the Kohonen network represents a point in the n-dimensional input space with its weights acting as coordinates. Given an input vector, the Euclidian distance from each neuron is calculated and the input vector is considered to belong to the class of the nearest neuron which is then called the winner neuron. The winner neuron is then trained to better fit the input vector for subsequent classifications. This is done by moving it closer to the input vector by a small factor, called the training factor (usually between 0.2 and 0.7).

The selection of a proper training factor plays an important role on the neural network’s training process. A very large training factor will cause the network to oscillate continuously around the input vectors without it ever settling to a proper centroid. A very small training factor will result in a slow-paced learning cycles that require a great number of training vectors to successfully complete the training process. A widely accepted practice is to use a high learning rate during the initial training cycles and gradually reducing it in order to perform progressively refined updates to the weights.

The training process may also include non-winner neurons. Selected according to their distance from the winner neuron, neighboring neurons may also be trained with a lesser training factor, a process called lateral excitation. At the same time, other neurons further away from the winner are not adjusted or may even be moved further away from the input vector, a process called lateral inhibition. Lateral excitation and inhibition ensure that the resulting classes are well defined. This allows us to find the inherent structure of the input data.

The ECG's one-dimensional nature provides a simple input signal in order to utilize a Kohonen network in an embedded, real-time system. The Euclidean distance is calculated with a simple subtraction without squaring and rooting. Also the training that requires multiplication with a training factor, may be implemented with right shifts for training factors like 0.5, 0.25, 0.126 etc. A one-dimensional, two-neuron Kohonen network is an oversimplification of the model but is ideal for this application and easily coded even in assembly language.

Block Diagram

The overall layout of the ECG system is described by the following figure. This block diagram separates the system into abstract modules and the lines that connect each other represent the relations between them.

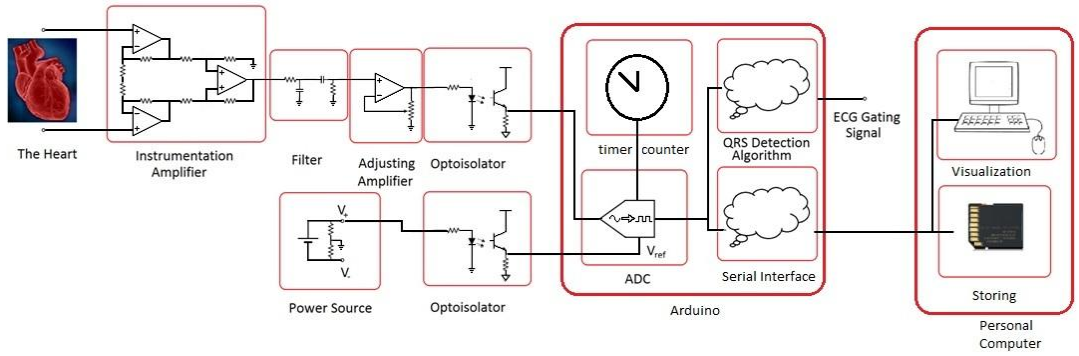


Figure 17: The block diagram for an electrocardiograph

The ECG signal goes through each of the blocks and becomes a digitized stream of binary values. The electrical field formed by the charge of the heart muscle is picked up by the sensors of the ECG system (surface electrodes) and guided into the pre-processing stage. There the weak signal from the heart is first amplified to the voltage range of the ADC's V_{ref} and then the anti-aliasing filter attenuates high frequencies. To avoid misinterpreting the signal it is important not to lose information of the spectral domain. Therefore, the amplifier was designed to have constant gain, linear or zero phase and the filter to not attenuate useful frequencies. After that, another amplifier with gain controlled by a potentiometer was used in order to provide the user with gain control and to compensate for the passive filter's loss in amplitude

The signal was fed through the optoisolator to the Arduino. The entire pre-processing stage is in a completely isolated circuit from the rest of the system for reasons of safety and as a result it has a separate power supply, a 9V battery. Note that a second channel of optoisolation was used to send the positive supply voltage of the op-amps to be used as the reference voltage for the ADC. This was done because this voltage represented the saturation voltage of the amplifiers and therefore the maximum voltage they can output. This voltage had a nominal value of 4.5V (even lower as the battery was drained over time) and using that as reference instead of the default 5V made better use of the full range of the ADC.

After the signal went through the optoisolator's voltage barrier, it was safe to use devices powered by the mains. The ADC converted the analog signal into digital. An internal timer/counter was used to trigger the ADC's sampling. The QRS detection algorithm then received the data and reported whether the signal corresponded to a QRS complex. Last, the

ECG data were transmitted through a serial connection to a personal computer it was stored and presented.

Design

Pre-processing

Supply

The pre-processing stage was powered by a separate power supply than the rest of the system in order to protect the organism from electric shock. Therefore a 9V battery was used in the operational amplifiers' negative and positive supply voltage. The battery was connected to a voltage divider, as shown in the figure below, consisting of two resistors 1MΩ each. That way, the output of the voltage divider served as the ground voltage, the positive end of the battery served as V+ (+4.5V) and the negative end served as V- (-4.5V)

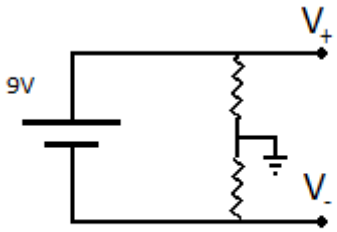


Figure 18: The Power Supply schematic

The Instrumentation Amplifier

The instrumentation amplifier was the first part to be implemented. The initial buffers used 1MΩ resistors for R_{F1} and R_{F2} and 1kΩ resistors in R_{I1} and R_{I2}. Therefore the gain was 1000. The operational amplifier selected was the 741, with a unity gain bandwidth of 1Mhz, which means that with a gain value of 1000, frequencies higher than 1kHz would be attenuated. Since only frequencies lower than 100Hz were to be retained, there was no need to separate the amplification process in multiple steps. Subsequently the differential amplifier did not need to perform any amplification and thus, the resistors R1 R2 R3 and R4 all had the same value, 10kΩ.

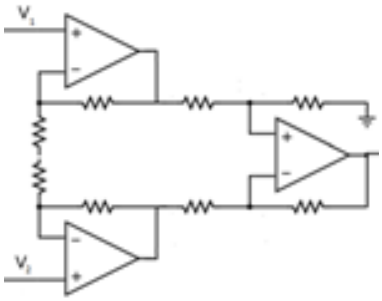


Figure 19: The Instrumentation Amplifier schematic

The Filters

The anti-aliasing filter used a passive RC low-pass filter with a resistor value of 3MΩ (three 1MΩ resistors in series) and a capacitance of 500pF (two 1nF capacitors in series) which yielded a cut-off frequency of 106 Hz.

A passive high-pass filter was placed after the optoisolator in order to reject any interference from nearby muscle movement or baseline deviation. Also this filter served as a first step in the QRS detection method, where the low-frequency components of the P and T waves were attenuated. This made it easier for the algorithm of the Arduino to detect the QRS complex. This filter was implemented with a capacitance of 2nF (two 1nF capacitors in parallel) and a resistance of 3MΩ (three 1MΩ resistors in series) so that it had a cut-off frequency of 26.5 Hz

In the figure below we see the original signal and the high-pass filtered signal. The QRS complex is prominent although the amplitude of the signal is dramatically reduced. The red horizontal lines in the second plot indicate the ADC's quantization steps.

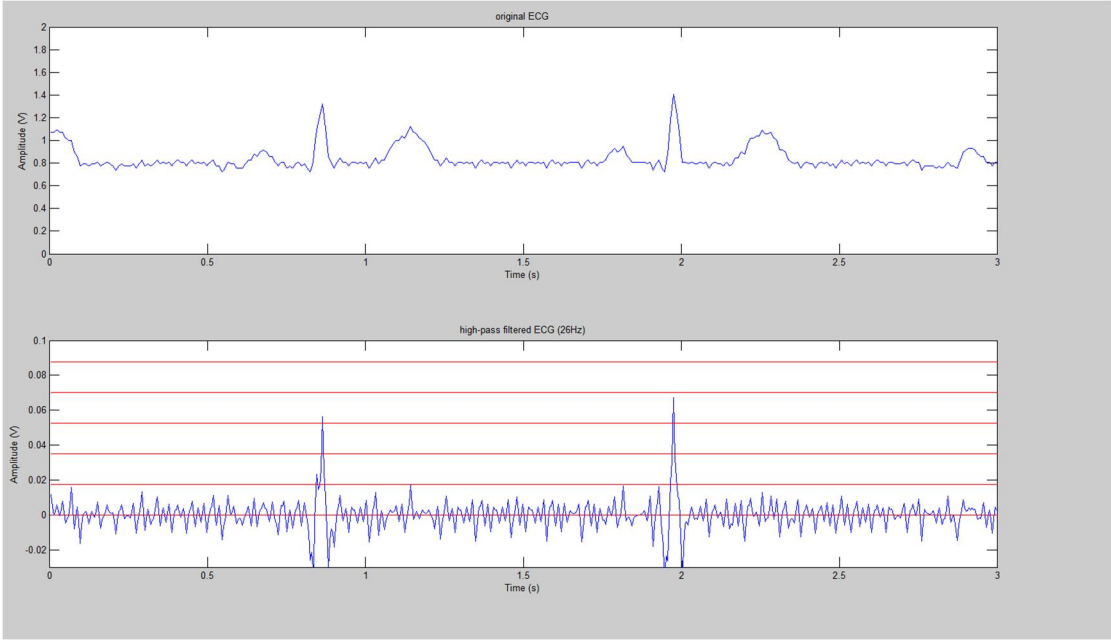


Figure 20: The unfiltered and filtered signals

The Adjusting Amplifier

As a last step in the pre-processing stage, a voltage amplifier was added, in order to adjust the overall gain of the signal using a 10KΩ potentiometer. This compensated for the passive filter's loss in amplitude in the previous stage. The potentiometer acted as a voltage divider controlling the gain of the amplifier. This way the signal could be scaled up or down to better fit the ADC voltage range in order to make use of its full resolution, without requiring hardware changes.

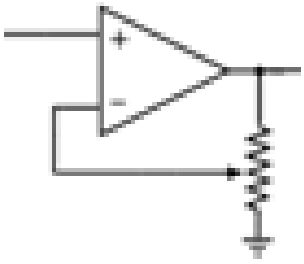


Figure 21: The Adjusting Amplifier schematic

The Optoisolator

The optoisolator chip PS2501 provided an electric barrier between the pre-processing stage of the ECG and the ADC, in order to protect the subject from electric shock caused by malfunctioning components. The isolator consisted of a GaAs light emitting diode, a translucent electric barrier and an NPN silicon phototransistor, encapsulated in a plastic DIP (Dual In-line Package). The input signal was connected to the LED's anode through a 220Ω resistor which ensured the current through the diode did not increase above nominal values. The cathode of the LED was connected to the ground of the power supply. The transistor's collector was powered by the 5V source that the Arduino operated (mains) and the emitter was connected to the Arduino's ground through a 50Ω load resistor. The output signal was taken from the transistor's emitter.

The use of the isolator introduced some unwanted distortion in our signal that should be taken into account. The LED causes a 0.8V drop in the input signal and the output was saturated at about 2V. To deal with the voltage drop, a proper gain should be selected through the adjusting amplifier's potentiometer. This allowed the signal's amplitude to be high enough not to be notably distorted. To deal with the optoisolator's saturation voltage, the pre-processing stage's 4.5V positive supply line, which was also its own saturation voltage, was connected as an input to a

second channel of the optoisolator in order to be used as reference voltage by the ADC.

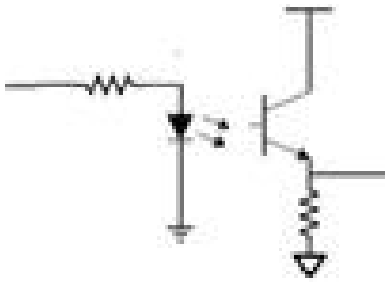


Figure 22: The Optoisolator schematic

The Arduino

ADC

The digitization stage was implemented on the Arduino board. The reference voltage (around 4.5V) was taken from the optoisolator into the A_{ref} pin (pin 21) and the ECG signal was connected to A0 pin (pin 23). As mentioned above, the anti-aliasing filter attenuated frequencies above 106Hz. Therefore, the sampling rate was set at least at 212.2Hz. Thus, the time interval between each sample was 4.7ms. The Arduino's ATMEGA328p has a built-in timer/counter to allow precise triggering of the ADC sampling. The timer/counter's clock was set to be the CPU clock (16MHz), prescaled by a factor of 1024, and by utilising a count up to 73, the sampling frequency would be 214.04. Instead of the 10 bits resolution, only the 8 most significant were used, to simplify the arithmetic operations of the QRS detection algorithm. Therefore the ADC yielded a quantization error of $\frac{4.5}{2^8-1}$ = $17.64mV$

The Program

The Arduino program consisted of three functions. The function void setup() was executed once during power up and was responsible for the initialization of the board. The function void loop() was repeatedly called after the execution of void setup() for as long as the board was powered. This function was mainly responsible for training the Kohonen network by adjusting the weight of the winning neuron. Finally, the interrupt service routine defined by the macro ISR(ADC_vect) responded to the Arduino ADC interrupt request (as declared by the ADC_vect argument) and performed the classification of the signal in real-time.

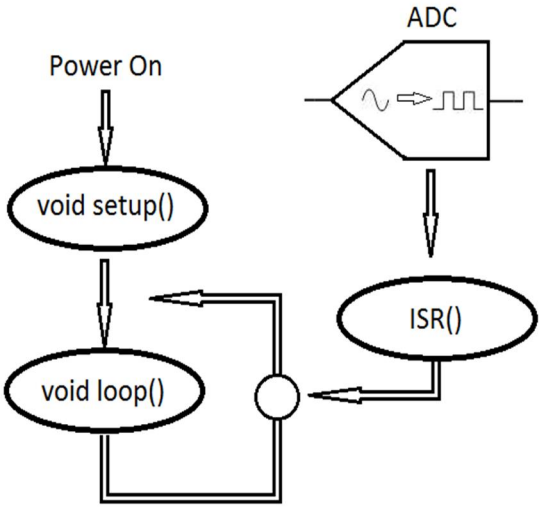


Figure 23: Arduino Code flowchart

void setup()

This function initialized the serial interface of the board with a baud rate of 9600bps with 8 data bits 1 stop bit and no parity check. Also, bit 5 of PORTB was set as output through the Data Direction Register B. This pin served as the output the result of the QRS detection algorithm. Then, the peripherals associated with the sampling of the ECG signal (i.e. the ADC and timer/counter0) were initialized. The ADC was turned on, with a new conversion starting immediately and with enabled interrupts. It was set to start each successive conversion every time the timer/counter0 Compare Match A event triggered. Its reference voltage was set to be external, i.e. by the Vref pin and the output of the ADC was set to be left-adjusted so that reading the 8 most significant bits from the register ADCH would suffice. The timer/counter0 peripheral was set to run in CTC mode (Clear Timer on Compare match) and to use the CPU clock signal, prescaled by 1024 (62.5KHz). Any interrupts associated with timer/counter0 were disabled, the initial counter value was set to zero and the Compare Match register was set to 73 in order to trigger the ADC with a frequency of 214Hz.

ISR(ADC_vect)

The ISR function was called whenever a conversion completed and the ADC interrupt was triggered. This routine was responsible for reading the binary value of the signal from the ADCH register and then calculating the difference of that value from the two neurons. Then by comparing the two differences, the program provided an output of the resulting winner neuron by either setting bit 5 in PORTB high or low. Bit 5 in PORTB corresponded to the Arduino's digital pin 13 which is directly connected to an LED in the Arduino board. This is the time-critical part of the algorithm so it was necessary to write the code needed through AVR assembly

instructions using the asm() command. This allowed for real-time behavior since it was known how long it would take to complete each task by avoiding compiler optimizations that would not be suitable for real-time QRS detection. Also any SBRC instruction was followed by a complementary SBRS instruction with a NOP, in order for the program not to deviate from true periodicity between the ADC sampling and the reporting of a QRS wave.

void loop()

The loop() function acted as the main loop of the program and was responsible for the training of the Kohonen network. A while loop, locked the execution in place until the ISR function was called. Then the winner neuron was trained and the ADC value was sent through serial connection to a computer for further processing and visualization. That was a non time-critical part in the code, therefore C code was used to simplify the arithmetic operations required for the training.

Visualization and Storing

The presentation and visualization of the ECG signal was done through a java program running on a personal computer (or any compatible device that supports serial interface and Java Virtual Machine). Libraries used in this program included java.awt, javax.swing, java.util, java.io and gnu.io. The libraries java.awt and javax.swing provided the user interface objects for the graphical presentation of the signal in a window environment. The java.util class LinkedList was used as a useful implementation of a linked list data type that stored all the data points provided by the serial input stream and provided the functionality of a queue through the poll() and offer() functions. Also gnu.io (or else, the RXTX library) is a native java library that provided serial and parallel communication for the Java Development Kit. Finally, the java.io package was used to provide the BufferedReader and BufferedWriter classes that would encapsulate the streams from the RXTX library for simpler implementation from a programming aspect. Also a new text file was created in order to store the values read for further processing.

The program's operation was to read the data from a serial port and graph them onto the screen. When executed, the program would open a serial connection to a specified port (usually COM3) and create a frame on the screen with a single component, the panel where the signal was shown. Then it continuously read ASCII encoded numbers from 0 to 255 and convert them into integers. Then the values were drawn in the panel in a scale where the value '0' corresponded to the lower edge of the frame, the value '255' corresponded to the higher edge of the frame and the horizontal axis corresponded to time.

Results

The figure below shows a reconstruction of the QRS detection algorithm used in the Arduino using MATLAB. The ECG signal was obtained by the system. The upper plot shows us with red the ECG signal and with blue the thresholds of the two Kohonen neurons. The bottom plot shows the output of the QRS detection algorithm. It can be seen that the weights of the neurons change whenever their corresponding neuron is declared a winner, to better fit any variations of the signal. Also, note that after the first QRS in this case, the algorithm mistakenly reported the following T wave as QRS. This false positive though does not reoccur because the algorithm was better trained in subsequent classifications.

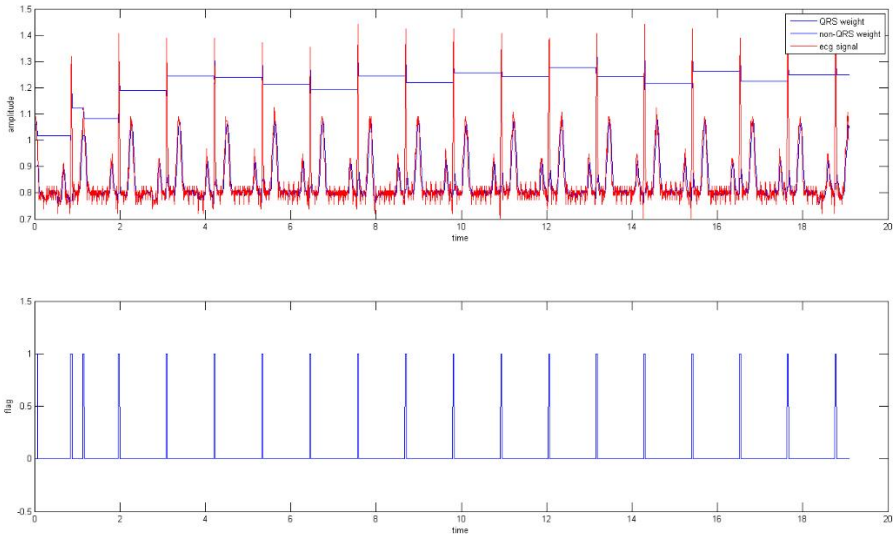


Figure 24: Above, the ECG signal (red) and the Kohonen Thresholds (blue). Below, the output of the QRS Detection algorithm

Also, in the figure below one can see the output of the QRS detection algorithm when the input signal is filtered by the high-pass filter at 26Hz. The QRS complex is evident without thresholding but there is a variation in the heights of the signal's peaks, which makes the use of a static threshold prone to errors. The Kohonen network is trained to the variations of the peaks and as such recognizes them correctly. Using together high-pass filtering and Kohonen networks improved the accuracy of the algorithm. The false positive that occurred in the previous case, did not occur here. The T wave was attenuated and did not have enough amplitude to be mistaken as a QRS complex.

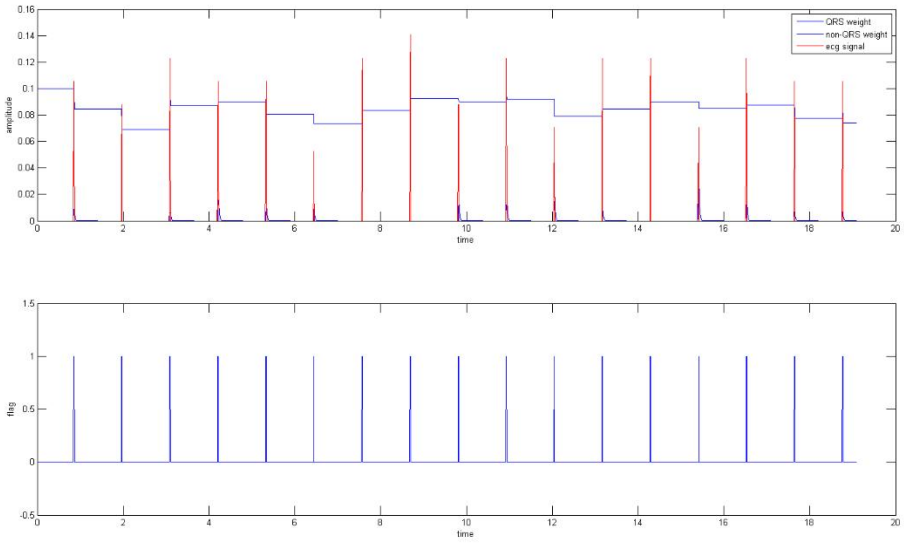
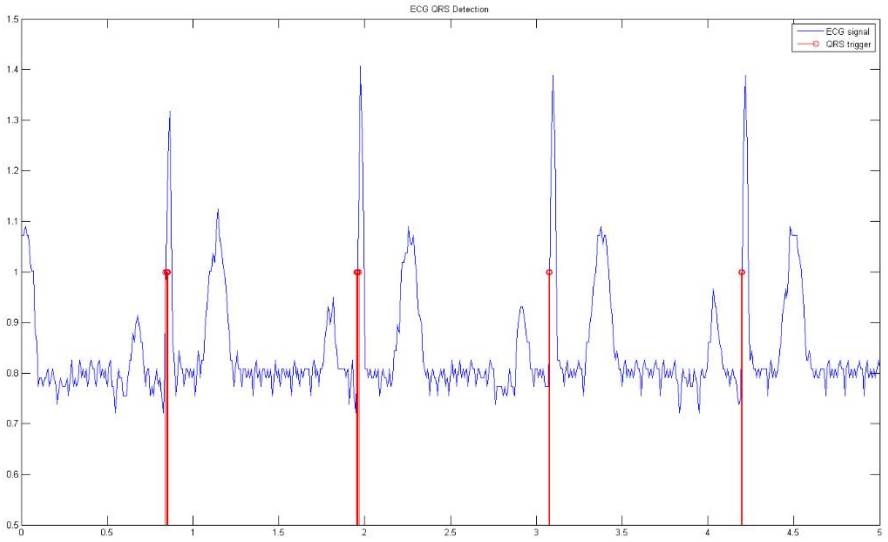


Figure 25: The same algorithm applied in the filtered signal

Last, in the figure below we see the ECG signal and the QRS detection flag as recorded by the Arduino itself. The QRS detection flag recognizes the steep slopes and high amplitudes of the QRS complex adequately to be used as an ECG triggering device.



Future Work

There are a number of directions this design could be improved so as to extend its functionality. Several minor modifications or major reworks can be done to enable it in a number of applications.

- The optoisolator's linearity could be improved if we used a negative feedback loop with an operational amplifier. This would improve the accuracy of the readings
- The device could be implemented as a heart rate monitor. By counting the time between subsequent heartbeats, the heart rate could be measured and provide the user with real-time information about the heart's condition as well as provide alerts for bradycardia or tachycardia
- If only the high-pass filtering would be used as a QRS detection method, one could simply use an operational amplifier as a Schmidt trigger to provide the output flag. This way the microcontroller is not necessary to perform the thresholding or even A/D conversion, so the cost and simplicity of the board would be reduced
- The java program instead of simply visualizing and storing the data could be designed to perform some form of post-processing to the signal and using some pattern recognition techniques and as such could provide diagnoses for cardiac conditions such as ischemia or AV block.
- All three basic limb leads could be recorded or even the precordial leads. This would enable more information for the prognostic application mentioned above.
- The use of Einthoven's law would be useful if recording the three basic limb leads. If the three signals didn't follow the law, it would indicate that a lead was disconnected.
- Given the small scale of the electronics used and the Arduino versions "micro" and "lilypad", an ECG Holter could be implemented that used wireless (Bluetooth) communication with the user's Android phone and the program in the phone could perform monitoring of the user's condition and if necessary contact the patient's physician or emergency services.

Bibliography

1. Igoe, T. (2007). Making things talk. Beijing Cambridge Mass: O'Reilly.
2. Watson, R. (2005). Anatomy and physiology for nurses. Edinburgh New York: Elsevier.
3. Alberts, B. (2004). Essential cell biology. New York, NY: Garland Science Pub.
4. Russell, S., Norvig, P. & Davis, E. (2010). Artificial intelligence : a modern approach. Upper Saddle River, NJ: Prentice Hall

5. Ward, J., Clarke, R. & Linden, R. (2005). Physiology at a glance. Malden, Mass: Blackwell Pub.
6. Faiz, O., Blackburn, S. & Moffat, D. (2011). Anatomy at a glance. Chichester, West Sussex, UK Hoboken, NJ: Wiley-Blackwell.
7. McClellan, J., Schafer, R. & Yoder, M. (2003). Signal processing first. Upper Saddle River, NJ: Pearson/Prentice Hall.
8. Theodoridis, S. & Koutroumbas, K. (2009). Pattern recognition. Burlington, MA London: Academic Press.
9. Malvino, A. (1999). Electronic Principles. New York: Glencoe/McGraw-Hill.
10. Vlahavas I., Kefalas P., Bassiliades N., Kokkoras F., Sakellariou I.(2011). Artificial Intelligence. Thessaloniki: University of Macedonia Press
11. Deitel, P. & Deitel, H. (2010). Java : how to program. Upper Saddle River, N.J: Pearson Prentice Hall.
12. Webster, J. & Clark, J. (1998). Medical instrumentation : application and design. New York: Wiley.
13. Gadre, D. (2001). Programming and customizing the AVR microcontroller. New York: McGraw-Hill.

Appendix I, Arduino Code

```
#define BIT(x) (1<<(x))

void setup(){
  analogReference(EXTERNAL);
  pinMode(A0,INPUT);
  Serial.begin(9600);
  int temp;
  pinMode(13,OUTPUT);

  temp=0b00100000;  //external reference
  //  ADMUX REGISTER
  //  REFERENCE VOLTAGE: AVcc
  //  LEFT ADJUSTMENT: NO
  //  INPUT PIN: A0
```

```
 ADMUX=temp;

 temp=0b11101111;
 //  ADC STATUS REGISTER A
 //  ENABLE ADC: YES
 //  START CONVERSION: YES
 //  AUTO-TRIGGER: YES
 //  INTERRUPTS: ENABLED
 //  PRESCALER: 128    (125KHz ADC clock)
 ADCSRA=temp;

 temp=0b00000011;
 //  ADC STATUS REGISTER B
 //  AUTO-TRIGGER SOURCE: TIMER0 MATCH A
 ADCSRB=0;

 temp=0b00000010;
 //  TIMER 0 CONTROL REGISTER A
 //  COMPARE MATCH A OUTPUT: NONE
 //  COMPARE MATCH B OUTPUT: NONE
 //  MODE: CTC
 TCCR0A=temp;

 temp=0b00000101;
 //  TIMER 0 CONTROL REGISTER B
 //  FORCE MATCH IN COMPARATOR A: NO
 //  FORCE MATCH IN COMPARATOR B: NO
 //  CLOCK SELECT: PRESCALED 1024
 TCCR0B=temp;

 temp=0b00000000;
 //  TIMER 0 INTERRUPT MASK REGISTER
 //  COMPARATOR A INTERRUPT: DISABLED
 //  COMPARATOR B INTERRUPT: DISABLED
 //  OVERFLOW INTERRUPT: DISABLED
 TIMSK0=temp;

 TCNT0=0;
 //  TIMER 0 COUNTER REGISTER INITIALIZED IN 0

 OCR0A=73;
 //  TIMER 0 COMPARATOR A REGISTER
 //  16MHz frequency, 1024 prescaler, 200Hz desired frequency:
16M/1024/212=73.629

}

volatile unsigned char val;
volatile char flag;
volatile char wait;
volatile char d1,d2;
```

```
unsigned char c1=5;              // Arbitrary initial value. It is going to be
adjusted as the program goes on
unsigned char c2=0;
static int i=0;

void loop(){

  while (wait==0) i++;
  //Train the network
  if (flag==32){
    c1=c1+(val-c1)/4;
  }else if (flag==0){
    c2=c2+(val-c2)/4;
  }else
    Serial.print("unknown flag :");

  // Send the results through serial connection
  Serial.print(val,DEC);
  Serial.print(",");
  Serial.println(flag,DEC);

  wait=0;
  i=0;
}

ISR(ADC_vect){
  asm volatile(

     "lds %[val],  0x79  \n"    //Address 0x79 is ADCH

     // Calculate the distance from the first neuron
     "mov r16,    %[val]  \n"
     "sub r16,%[c1]  \n"
     "in __tmp_reg__, __SREG__\n"
     "sbrc __tmp_reg__, 4\n"
     "neg r16  \n"
     "sbrs __tmp_reg__, 4\n"      //anti-jitter buffering
     "nop\n"
     "mov %[d1], r16  \n"

     // Calculate the distance from the second neuron
     "mov r17, %[val]  \n"
     "sub r17, %[c2]  \n"
     "in __tmp_reg__, __SREG__\n"
     "sbrc __tmp_reg__, 4\n"
     "neg r17  \n"
     "sbrs __tmp_reg__, 4\n"      //anti-jitter buffering
     "nop\n"
     "mov %[d2], r17  \n"

     //Compare the two distances
     "cp r16,r17  \n"
```

```
    "in __tmp_reg__, __SREG__\n"

    //Report the result
    "sbrc __tmp_reg__, 4\n"
    "ldi %[flag], 32  \n"
    "sbrs __tmp_reg__, 4\n"     //anti-jitter buffering
    "ldi %[flag], 0  \n"
    "out %[portb],%[flag]\n"

 : //output operants
   [flag]"=&d"(flag)
 , [d1] "=&d" (d1)
 , [d2] "=&d" (d2)
 , [val] "=&d" (val)
 : //input operants
  [c2] "d" (c2)
  ,[c1] "d" (c1)
  ,[portb] "I" (_SFR_IO_ADDR(PORTB))
 : "r16","r17"
 );

 wait++;
}
```

Appendix II, Java Code

SerialSocket.java

```
package arduinointerface;

import java.io.*;
import java.util.*;
import gnu.io.*;

public class SerialSocket {

   private InputStream in;
   private OutputStream out;
   private BufferedReader reader;
   private BufferedWriter writer;
   private SerialPort port;
   public static final int TIMEOUT = 2000;


   public SerialSocket(String name, int baud){
      try{
         CommPortIdentifier id =
CommPortIdentifier.getPortIdentifier(name);
         port = (SerialPort) id.open(this.getClass().getName(),TIMEOUT);
         port.setSerialPortParams(baud, SerialPort.DATABITS_8,
SerialPort.STOPBITS_1, SerialPort.PARITY_NONE);
```

```
            in = port.getInputStream();
            out = port.getOutputStream();
            reader = new BufferedReader(new InputStreamReader(in));
            writer = new BufferedWriter(new OutputStreamWriter(out));
        }
        catch(NoSuchPortException e){
            e.printStackTrace();    //some error handling
        }catch(PortInUseException e){
            e.printStackTrace();    //some error handling
        }catch(IOException e){
            e.printStackTrace();    //some error handling
        }catch(UnsupportedCommOperationException e){
            e.printStackTrace();    //some error handling
        }
    }

    public InputStream getInputStream(){
        return in;
    }

    public OutputStream getOutputStream(){
        return out;
    }

    public BufferedReader getBufferedReader(){
        return reader;
    }

    public BufferedWriter getBufferedWriter(){
        return writer;
    }

    public void addEventListener (SerialPortEventListener listener){
        try{
            port.addEventListener(listener);
        }catch(TooManyListenersException e){
            e.printStackTrace();    //some error handling
        }
        port.notifyOnDataAvailable(true);
    }

    public void close(){
        port.close();
    }

    public static void main(String args[])throws Exception{
        final SerialSocket x = new SerialSocket("COM7",9600);
        javax.swing.JFrame app = new javax.swing.JFrame("Osciloscope
Widow");
        app.setVisible(true);

app.setDefaultCloseOperation(javax.swing.JFrame.EXIT_ON_CLOSE);
```

```
        final Osciloscope osciloscope = new Osciloscope();
        app.add(osciloscope);
        app.setSize(600,300);
        final File file = new File("ECG.txt");
        final FileWriter wr = new FileWriter(file);

        x.addEventListener(new SerialPortEventListener(){
           public synchronized void serialEvent(SerialPortEvent evt){
              if (evt.getEventType()!=SerialPortEvent.DATA_AVAILABLE)
                 return;
              try{
                 String str = x.getBufferedReader().readLine();
                 System.out.printf("--(%s)\n", str);
                 StringTokenizer tkn = new StringTokenizer(str,",");
                 String val1=tkn.nextToken();
                 String val2=tkn.nextToken();
                 int i = Integer.parseInt(val2);

                 wr.write(String.format("%s\r\n",str));
                 osciloscope.addVal(i);
              }catch(Exception e){
                 e.printStackTrace();
              }
           }
        });
     }


  }
```

Osciloscope.java

```
package arduinointerface;

import java.awt.*;
import javax.swing.*;
import java.util.*;

public class Osciloscope extends JPanel{

   public LinkedList<Integer> list;
   public static Color
color[]={Color.BLACK,Color.BLUE,Color.RED,Color.GREEN};
   public double max=255;

   public Osciloscope(){
      super();
      list = new LinkedList<Integer>();
   }

   public final LinkedList<Integer> getChannel(int i){
      return list;
```

```
    }

    private int index=0;
    public void addVal(int x){

        list.offer(x);

        if (list.size()>getWidth())
              list.poll();

    }


    public void paintComponent(Graphics g){
        super.paintComponent(g);
        g.setColor(color[0]);

        for (int i=1;i<list.size();i++){
            double val1 = list.get(i)/max*getHeight();
            double val2 = list.get(i-1)/max*getHeight();
            val1 = getHeight() - val1;
            val2 = getHeight() - val2;
            g.drawLine(i, (int)val1, i,(int) val2);
        }
    }

}
```

Appendix III, Matlab code

```
clc
clear

file=load('ECG.txt');

N=length(file);
ecg=file(:,1)*5/1024;
flag=file(:,2);

cpu=16e6;
ps=1024;
timer=73;

fs=cpu/ps/timer;
DW=1/fs;
AT=N*DW;
t=DW:DW:AT;

BW=fs;
df=1/AT;
k=-(BW-df)/2:df:(BW-df)/2;
```

```
ECG=fft(ecg-mean(ecg))/N;

lag=0;
width=2;
fov=[lag,lag+width,0,100];

figure(1);
plot(t,ecg);
axis([0,AT,0.2,0.45]);
title('ECG');
ylabel('Amplitude (V)');
xlabel('Time (s)');

figure(2);
plot(k,abs(fftshift(ECG)));
axis([-BW/2,BW/2,0,0.01]);

figure(3);
plot(t,ecg,'b')
hold on;
stem(t,flag/32*0.3,'r');
hold off;
axis([0,3,0.2,0.4]);
title('ECG QRS Detection');
legend('ECG signal','QRS trigger');
```

Appendix IV, Circuits

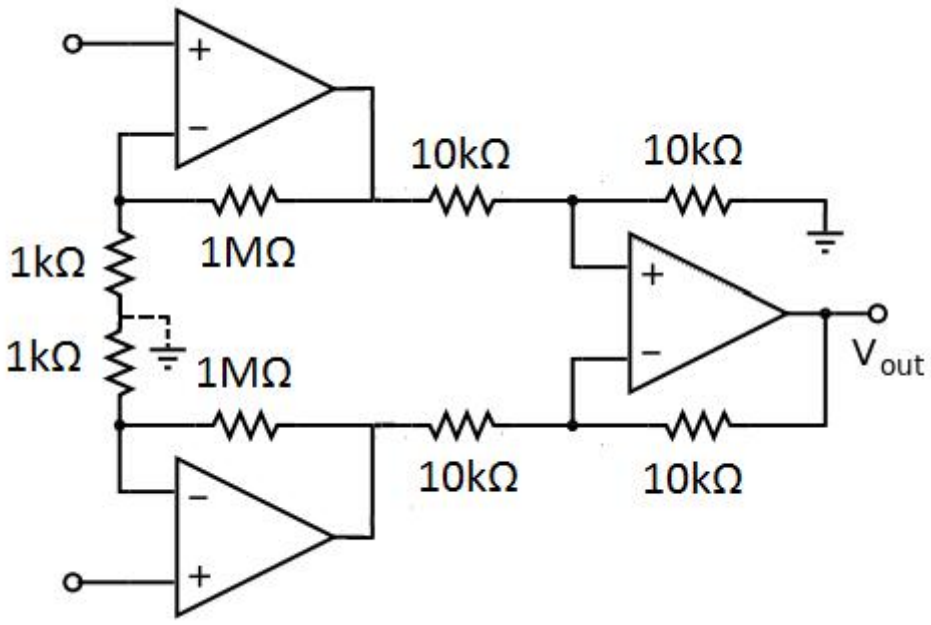


Figure 26: The Instrumentation Amplifier circuit

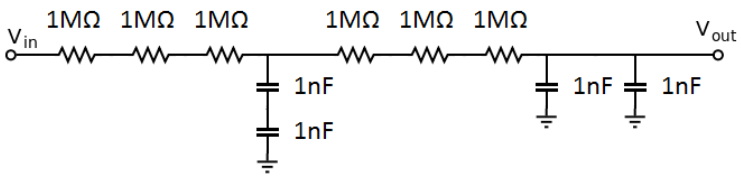


Figure 27: The Filter circuit

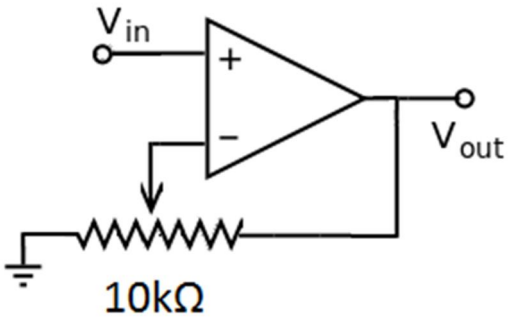


Figure 28: The Adjusting Amplifier circuit

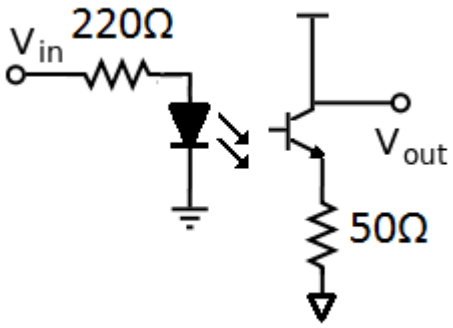


Figure 29: The Optoisolator circuit