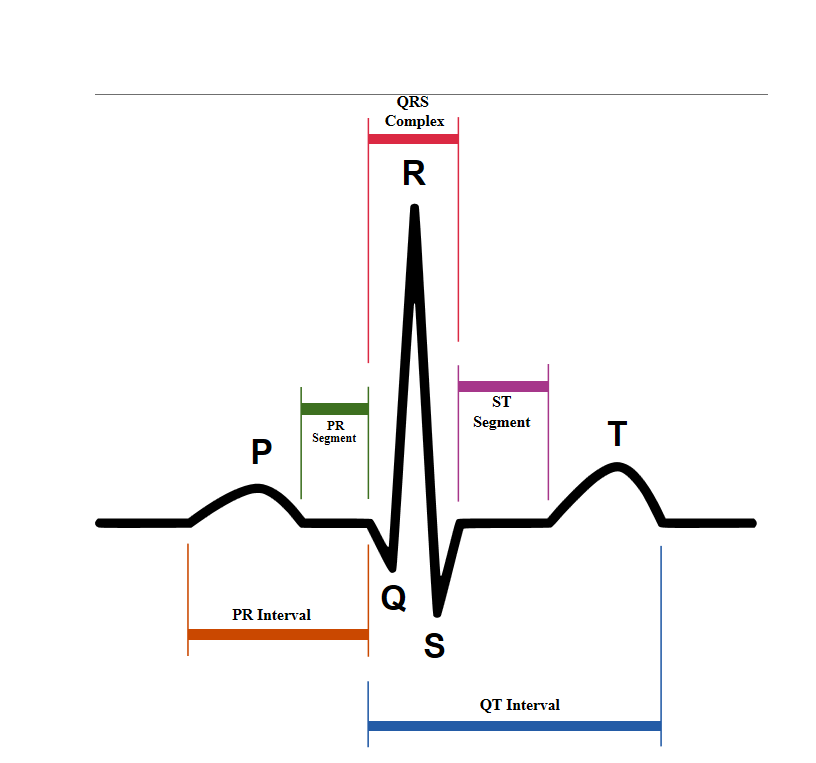
**Code Explanation**

**Introduction:**

The Canadian Space Agency (CSA) is working on developing a continuous health monitoring and decision support system for deep-space missions. To achieve that, the CSA has contracted Carré Technologies to develop the AstroSkin. The AstroSkin is a shirt worn by the user that measures various health sign such as breathing rete, heart rate, electrocardiogram (ECG), etc. For this document our focus will be mainly on the ECG.

**Defining ECG**

The heart, while pumping blood generates an electric signal that can be measured by placing electrodes at specific locations on the body; this signal is called an ECG signal. The main features of an ECG are the PQRST waves and the intervals and segments caused by them.



<https://en.wikipedia.org/wiki/Electrocardiography#/media/File:SinusRhythmLabels.svg>

Due to the importance of these points an algorithm was developed to find them and do certain calculations.

**Algorithm**

The algorithm was written in Matlab and can be run by typing “MainScript” in Matlab’s command window or by opening the script and pressing the run button.

**MainScript**

This is the main body of the code; it starts by selecting the name of the ECG folder that will be run along with a quality folder. Then it selects the sampling frequency which is 256Hz and a Boolean (IsAstro) that specifies if the data we have is from the AstroSkin or from another location. It was noticed that while using the AstroSkin data a less sensitive measurement is needed to prevent unnecessary warnings.

The MainScript runs an endless loop that reads a part of the data of approximately two seconds and then runs the analysis.

**ReadEcg**

This is the function responsible for reading the ECG data. It starts by reading two seconds of data and the peaks found (by Carré’s detection) during these two seconds. The peaks are read from the “RR\_interval\_quality.csv” and from this folder we can get the quality of the peaks. After that the last peak detected is checked that it has at least RR\*0.5 + RR\*0.14 (RR is difference between consecutive R waves) of data after it (if RR was not found then 0.35 + 0.1 = 0.45 sec). This time is selected because it is enough time to detect a T wave and its end. If the data read ends before that, then more data has to be acquired to satisfy this condition.

Initially the start time and end time are two seconds apart, but in the case mentioned above the end time changes to accommodate more data. When this occurs, the end time is saved under NewEnd to keep track of the end of the collected ECG.

In some cases the additional data read might contain a peak; we have to recheck for peaks in that range if we found one we recheck is it has enough data after it. Then we keep repeating this process until we have enough data after last peak we found

Since the peaks read are a function of seconds transforming them into points might cause rounding errors that is why the peaks are verified to ensure that they are true maximum rather than a shifted maximum. This step is done after the next step (Filtering) to ensure better results.

**FilterEcg**

Filtering is more accurate when more data points are taken, but at the same time the more the points the more time it takes, so a number of points has to be selected without having a big effect on the time. That’s why 10 seconds was selected, if the data is more than 10 seconds we apply a filter on the last 10 seconds and then save the period that was not already saved.

For filtering, a high pass filter was applied followed by reducing the signals amplitude to mV. Then the signal is sent to PanTompkinsFilter which applies a band stop filter along with a high pass and band pass filter.

**Derivative**

The derivative is then taken by following this formula:

If X[n] = [x1 x2 x3 ... xn]

Derivative of X[n] = [ … ]/dx

This formula is finding the slope between two points that are not consecutive (except first and last) so a better slope can be found.

**NewDetect**

This function detects the PQST waves; it uses the R wave that is already found and then proceeds to find a P wave and its start and end points then a Q wave and where it starts then an S wave and where it ends, finally a T wave and its start and end points.

P Wave Detection

A window of 0.2\*RR is selected, starting at 0.3\*RR before the R wave and ending 0.1\*RR before the R wave. In this window the maximum is found followed by the derivative. The derivative is used to verify that the P wave is not just a plateau.

After that the detection of the start of the P wave (Pon) begins, another window of 0.08\*RR before the P wave is taken and the minimum of that value is taken as the far most Pon. Then to determine the Pon the derivatives are checked to verify that a slope increase occurred, if no slope increase happened then Pon is moved closer to the P wave until the condition is satisfied.

A similar process is followed for the end of the P wave (Poff), but rather than moving forward the detection was moving backward until Poff was found.

Q Wave Detection

A window of 0.12\*RR is taken before the R wave and the minimum value found is the Q wave. After that it is verified that this is the only minimum, if this is true the Q wave is saved else it is missed.

Then the beginning of the Q wave is detected by searching for a maximum in a window of 0.03\*RR and then the maximum is modified until a noticeable slope change occurs.

S Wave Detection

This is similar to the Q wave but instead of searching before the R wave a 0.12\*RR window is selected after the R wave.

T Wave Detection

This is similar to the P wave but it comes after the R wave and the window is 0.2\*RR from 0.25\*RR after R wave to 0.5\*RR after R wave.

Additionally it is important if the T wave is inverted as this might help determine myocardial infraction as well as other health conditions. So if the amplitude of the maximum found was less than 0.05mV (low value for a T wave), we might have an inverted T wave. To check we find the minimum in that range and if it is more than 3\*the amplitude (this range was selected to take care of noise), then we have an inverted T wave. If not then the T wave is not inverted.

**Calculation**

This function uses the points found by NewDetect to find Intervals, Segments, Value, and Durations. It loops through the peaks and verifies if the waves were detected then does the necessary calculation.

Intervals:

* RR: Found by applying the difference between the peaks
* RRIndex: Found by selecting the peak relative to the difference found
* PR: Found by subtracting the start of a Q(Qon) wave and the start of a P wave(Pon)
* PRIndex: Specifies the location of Pon that was used to find PR
* QT: Found by subtracting the end of T(Toff) wave and Qon
* PRIndex: Specifies the location of the T wave that was used to find QT
* QTc: Corrected QT which takes into consideration the Heart Rate. Found using Bazett’s calculation: QTc =

Segments:

* PR: Found by subtracting the end of a P wave(Poff) and Qon
* PRIndex: Specifies the location of Pon that was used to find PR
* ST: Found by subtracting the start of T Wave(Ton) and end of S wave(Soff)
* STIndex: Specifies the location of Soff that was used to find ST
* QRS: Found by subtracting Soff and Qon
* QRSIndex: Specifies the location of the R wave that was used to find QRS

Value:

* AverageST: Mean of the ST segment
* STIndex: Specifies the location of Soff that was used to find AverageST
* Iso: Mean of the points between Toff and the Pon that comes after it
* IsoIndex: Specifies the location of Toff that was used to find Iso
* STMeasurement: Subtracts AverageST and Iso

Duration:

* P: Subtract Poff and Pon
* T: Subtract Toff and Ton

All the above calculations are relatively simple except for the Iso, since it requires two consecutive R waves instead of one. The calculation is being done by checking if there exists a peak after the peak that was selected in the specified range. If yes the calculation is done, else it is skipped and another range is taken, when the analysis on the new range is done then the old Iso is found using the first peak from the new range and the last peak from the old range.

**gui**

This script runs the Graphical User Interface. It plots the filtered ECG with the waves showing and displays any warnings that are generated. It also allows the user to pause the display and then to continue it. Also by clicking on any warning it will display 4 graphs showing where it occurred. In cases were the user doesn’t think a warning is valid, it can be deleted by pressing the “Delete” button.

**NewPlot**

This function plots the graph in a real time view. To do that it plots on top of an old graph with more data, a small delay has to be added between the plots to manage the display.

This function calls PlotPoints after every plot to display the points of interest (PQRST).

**UnselectBad**

This function unselects the points and the calculations that come from a bad signal or that are outside the specific range.

**PlotPoints**

This function plots the points that result from a good signal and that are between a specified range.

**CheckHealth**

This script uses the calculated data to generate warnings. It loops through the peaks in a range and checks if the calculation done resulted in a healthy condition or an unhealthy condition. In the case of an unhealthy condition it saves the warning and its location after making sure it is not already saved.

The warnings that are generated by this script might be caused by a noisy signal or imperfect detection. That’s why the results generated cannot be taken as an immediate risk. But if a warning occurs 4 or more times in the past six seconds then it is considered as a warning that will be displayed. This is done by SelectImportantWarning script.

Knowing that the RR quality we are reading from the folder only checks for the peak quality, there might be a noisy signal, between two peaks, that was not detected. And since noisy signal might increase the value of the peaks, we might get a warning of a high P wave. This warning is not valid, to solve this a test is done to check the number of peaks 100ms to the left of the P wave that was detected detected, and if there are a more than three noticeable (amplitude>0.15mV) peaks then we know that the signal is noisy and we should not generate a warning.

**SelectImportantWarning**

This script selects the warnings that need to be displayed. As mentioned above if the warning occurs four or more times it is considered important. And then it is saved under ImportantWarning which gets displayed on the GUI.

If an important warning occurred in the past six seconds another warning won’t get saved. This tends to help show warnings that need attention and are not repetitive.

Based on these warnings an alert can be generated that needs immediate attention.

**WarningWorthAlert**

Knowing that not all warnings cause a serious risk and need immediate attention this function returns a Boolean to specify if the warning that is being generated needed instant care.

**CheckAlert**

This script loops through the important warnings and in case an important warning occurs more than two times during 40 seconds and the warning is worth sending an alert, then an alert is generated.

Also an alert cannot be generated if a previous alert was generated in the previous four hours. This is done using CheckSimilarAlert function.

**AlertFound**

In the case an alert was found, a warning message will be displayed and the three important warnings that where causing the alert will pop up and display where the problem occurred. Also an email will be sent using SendEmailWithOutlook function.

**Specific Warnings**

Along with the warnings mentioned on top there are specific warnings that can lead to certain health condition. An example to that can be a high heart rate after waking up. In general a resting heart rate is considered normal if it lies between 60 to 100 beats per minute (BPM). But when waking up the resting heart rate has to be between 60 and 80 BPM. So in case we had a high heart rate in the morning this can indicate that the user is under stress or not getting enough rest.

To find this a warning was added in CheckHealth that gets the time and if the time is between 5:30 am and 7:00 am and if the heart rate is high then a warning is generated. Knowing that the final goal is to monitor the astronaut’s health, the time has to be selected based on their sleeping pattern. Astronauts wake up at 6:00 am GMT so a range around that was selected.

To get the time read it from the ECG folder and then we add the start time found in statistics.csv folder. The time in this folder is in GMT so no need to convert it to another time zone.

**MyocardialInfractionTest**

This script tests for myocardial infraction. Myocardial Infraction causes significant Q waves, ST elevation, and Inverted T waves. So when these three warnings are detected at the same time myocardial Infraction warning is generated.

**Additional Comments:**

Along with the Real time script, a non-real time script was developed with the purpose of running faster through old data and saving the final results. Since the non-real time algorithm has more data to start with it will have slightly better results. This is caused by the filtering process, since the non-real time filters the whole signal while the real time filters two or 10 seconds of the signal.

Also for QTc calculation as the formula shows: QTc = we need to find RR (Difference between consecutive heart beats). But since for the real time we are working with chucks of data, for the last QTc calculation the RR interval cannot be directly calculated since we need to wait for another chunk to measure the difference. But to get faster results the RR of the old QT is being used. For the non-real time this is not a problem.

**Running the Code**

To be able to run the code, open Matlab-> wfdb-app-toolbox-0-9-9 -> ECG\_testing ->OrganizedDetection.

Then add the ECG (ECG\_I.csv or ECG\_II.csv or ECG\_III.csv) along with the RR\_interval\_quality.csv. Open “MainScript” and change “FileName” to the name of the ECG folder used. Then run “MainScript” for real time analysis. OR open “NonRealTime” script and change the “FileName” to the name of the ECG folder used. Then run “NonRealTime” for non-real time analysis.