

**Embedded Systems**

**Final Project**

**Live Heart Monitor Using AD8232**

Fadi Dawoud

900163212

Dr Mohamed Shalan

**Problem Definition**

Electrocardiograms (ECG) monitors are analog sensors that produce an analog electrical signal corresponding to electric signals generated through a patient’s heartbeat.

Biological signals are recorded through special electrodes and are passed to a signal conditioning circuit to amplify and enhance the signals for further analysis.

For this project, the signal conditioning circuit is within the AD8232 component. It accepts three (or two) electrodes for input that must be placed at one of a few defined placement positions on the patient’s body (figure 1).

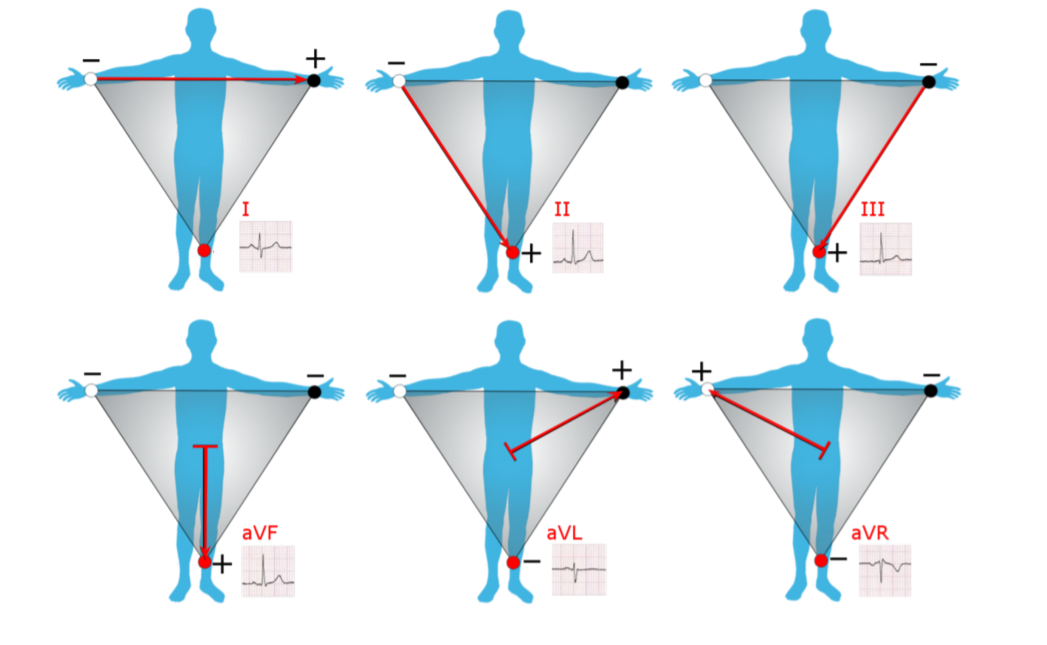


Figure 1: Different placements for electrodes and the expected waveform. Courtesy of Dr Shalan's slides

**Project Overview**

Metallic electrodes attached to the AD8232 were utilized to generate an analog electrical signal that maps electric pulses from the heart. If plotted over time, this signal will produce the family heartbeat waveform, examples of which are shown in figure one depending on the positioning of the electrodes.

The analog signal is consumed by the STM Black Pill (STM32F103C8) microcontroller. The microcontroller is connected to a PC application over serial link and offers a serial interface of one of three commands:

1. Setting sampling rate
2. Returning live values of the input analog signal in real time for 60 seconds
3. Computing and returning the instantaneous heart rate in beats per minute (bpm)

The PC application controlling the microcontroller offers a customizable serial interface and a graphical user interface (GUI) to facilitate connecting serially to the microcontroller, sending commands and displaying the outputs.

**System Architecture**

A simplified system architecture diagram is shown in figure 2:

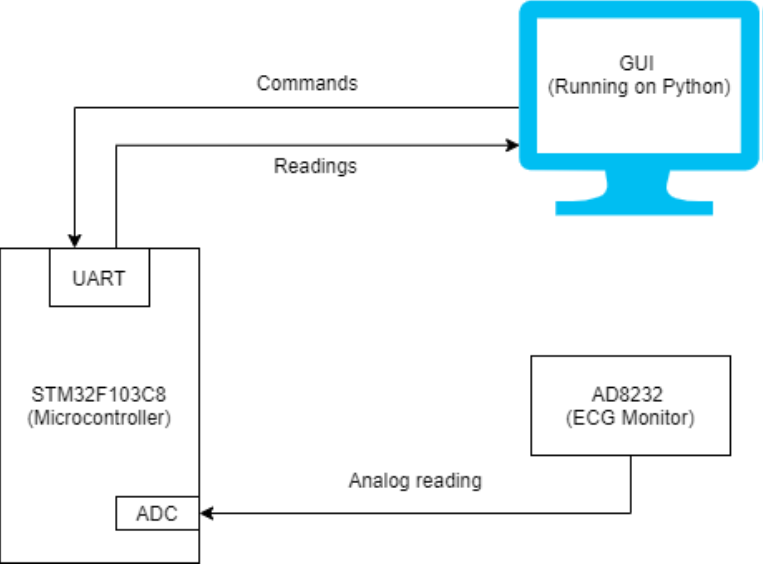


Figure 2: The microcontroller receives analog readings from the ECG monitor and communicates serially with the PC

There are three main components to my system:

1. ECG Monitor

The monitor consists of a three-way electrode and a signal conditioning circuit. Usually, they are purchased together as one package. The electrodes are placed on a patient’s body according to one of the placements in figure 1. With little to no configuration, the circuit conditioning circuit offers an amplified analog output on its output pin corresponding to the patient’s heartbeat.

1. Microcontroller (Black Pill / STM32F103C8)

The microcontroller acts as intermediary between the user and the output signal of the ECG monitor. The following sub-components of the microcontroller are used:

1. UART

The UART module is used to facilitate serial communication between the microcontroller and the PC app.

1. ADC

The on-board analog-to-digital converter (ADC) module of the microcontroller converts the input analog signal into a series of values. The frequency with which the ADC is triggered is controlled by the general-purpose timers, below.

1. General purpose timers

Three general-purpose timers were utilized:

* + Timer 1 to regulate readings according to the sampling frequency
  + Timer 2 to compute 60 seconds of data
  + Timer 3 to regulate readings of 100 samples/s for heart rate calculation

1. PC Application

A GUI application running on top of python provides a user interface to the microcontroller. The application provides the following services to the end user:

1. Connecting to a serial COM port, using a specified baud rate
2. Specifying the microcontroller sampling rate, which configures ADC readings for the live data
3. Fetching and displaying live data from the microcontroller ADC
4. Fetching and displaying the instantaneous heart rate in bpm.

**System Operation**

The microcontroller supports the following serial commands:

1. Calculating heart rate:
   * App to microcontroller: “a”.
   * Microcontroller to app: ascii representation of the patient’s heart rate as computer by the microcontroller by examining the patient’s readings at 100 sps for two seconds.
   * Behavior: The microcontroller starts sampling and storing values at the rate of 100 samples/s, for two seconds. The array of 200 values are analyzed for the QRS Complex peaks (see figure 3). To calculate the frequency of this cyclical waveform, we locate two consecutive QRS Complex peaks, calculate the spatial distance between them, and invert it to get the frequency (heart rate).

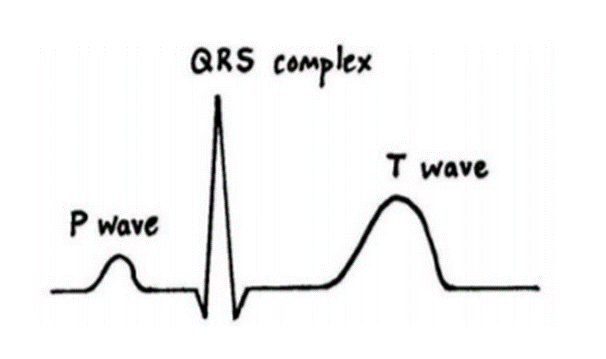


Figure 3: Expected heart beat waveform, courtesy of Dr Shalan’s slides

1. Returning live readings for sixty seconds:
   * App to microcontroller: “b”.
   * Microcontroller to app: ascii representation of the ADC output of the analog signal according to the specified sampling rate. Values range from 0 to 4095 as the on-board ADC has a 12 bits precision.
   * Behavior: The microcontroller enables both Timer 1 and Timer 2. Timer 1 overflows with a frequency equal to the specified sampling rate. Timer 2 overflows after 60 seconds. On Timer 1 overflow, the ADC is triggered to convert the analog value to digital, which is consequently sent over the serial link to the expecting PC application. On Timer 2 overflow, both timers are disabled. This way, Timer 1 is enabled only for 60 seconds from receiving the command, where, at the specified sampling rate, it triggers the ADC conversion.
2. Setting sampling rate
   * App to microcontroller: “c<rate>c” (for example: “c100c”)
   * Microcontroller to app: no response.
   * Behavior: The frequency of Timer 1 overflow event is modified to match the selected sampling rate. The base clock frequency for the timer is 8 MHz, and the counter period (maximum value for the timer counter) is 500. Upon receiving a new sampling rate, the pre-scaler of Timer 1 is set to 16000 / sampling rate. This way, the overflow frequency of Timer 1 will be equal to , as expected. Timer 1 behavior on overflow is outlined in the *behavior* section of part b of this section.

**Build and Run Instructions and Walkthrough**

1. Connect the physical circuit. The microcontroller must be connected to the PC via an FTDI cable or a similar circuit that would interface the UART module to the USB port of the computer.

The electrodes are connected to the conditioning chip (AD8232). The chip’s VDD and gnd must be connected to the microcontroller’s VDD and ground. Finally, the output pin of the chip must be connected to the microcontroller’s ADC1 input pin. The completed circuit is found in figure 4.

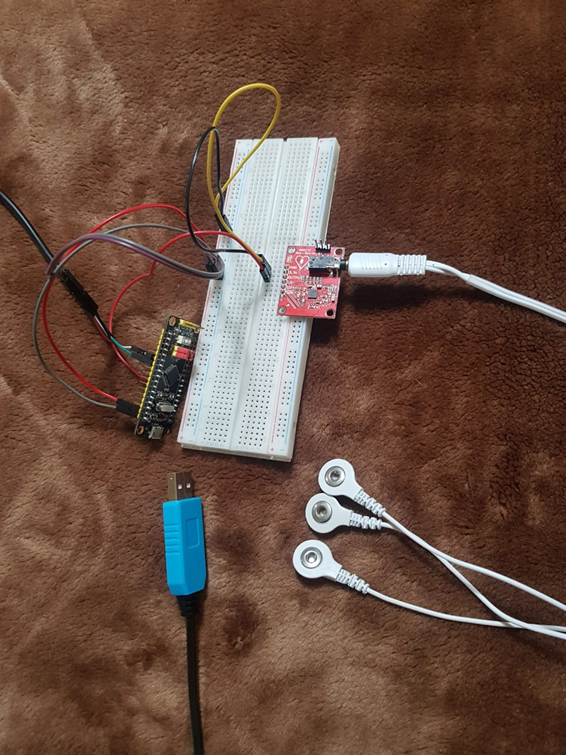
****

Figure 4: Complete phsyical circuit

1. Flash the microcontroller with the embedded hex code executing the interface. If you are using the same Black Pill, you can use the Hex file on the repo directly. If not, you will need to re-configure the project to match your hardware and generate a new hex code file.
2. Run the PC application. It is built over python, which does not require compilation. To run the application, in the main directory of the repo, run the command “python monitor.py”

Note: You need to have python installed as well as the following required packages:

* + - Tkinter, for GUI
    - Numpy
    - pySerial
    - Matplotlib, for graphing ECG data in real time

1. The first screen (figure 5) allows the user to select the COM port and the baud rate to connect to the microcontroller. If one device is connected to the COM ports, it is selected by default for the user to confirm.

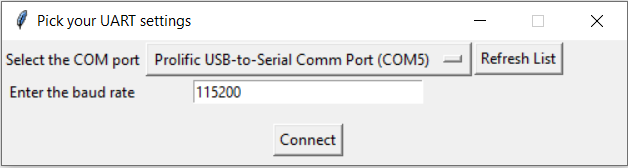


Figure 5: Connection screen list

1. Next, the main dashboard (figure 6) is displayed. The user can set a new sampling rate, start live data reporting, and calculate the heart rate.

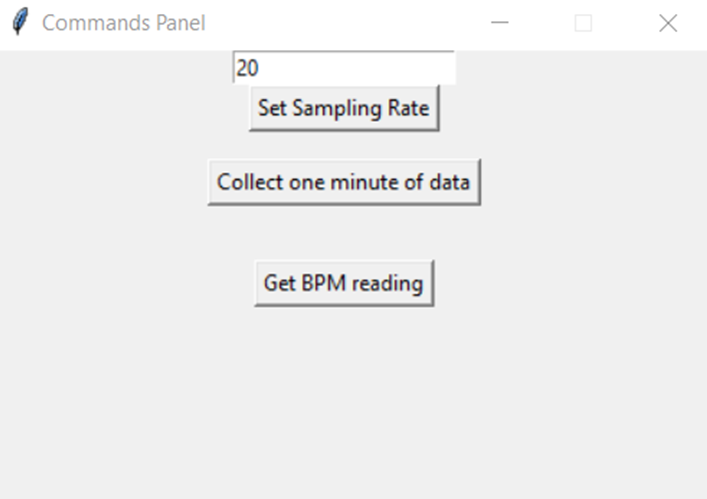


Figure 6: Main dashboard

* 1. Setting the sampling rate is as simple as modifying the field in figure 6 and clicking on the appropriate button.
  2. To view live data for one minute, the second button is pressed. It opens a new screen (figure 7) for the live data.

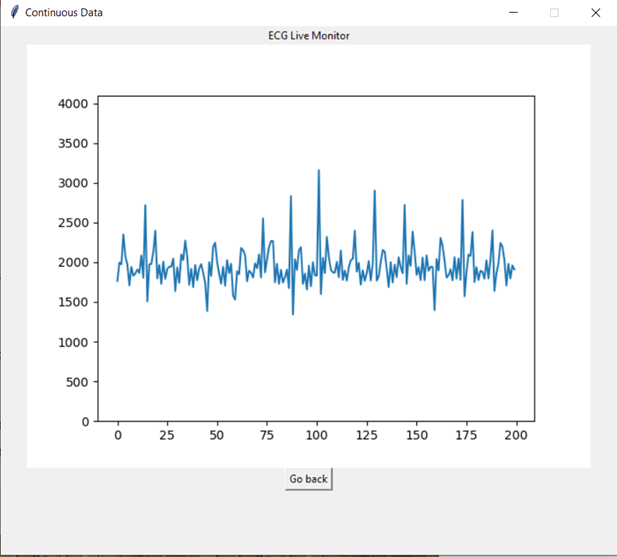


Figure 7: Live value monitor

* 1. To calculate the heart rate, the third button is clicked. A new screen is opened reporting the required value (figure 8).

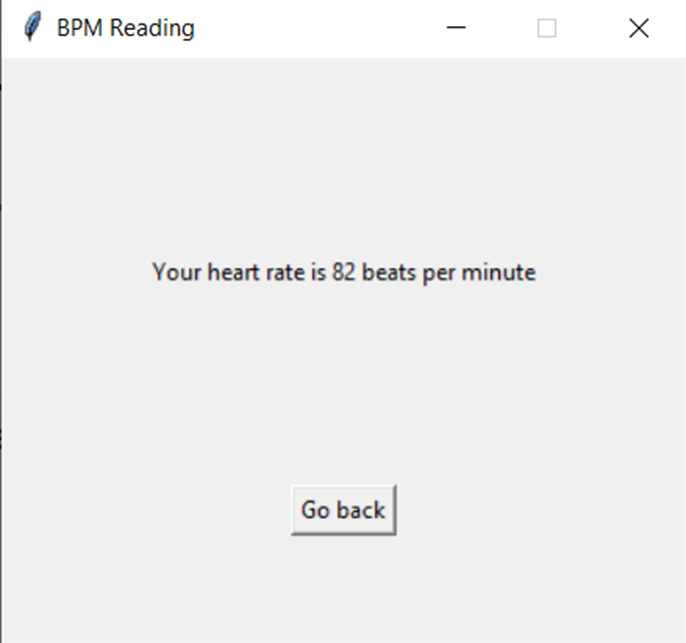


Figure 8: Heart rate reporting screen

**Design Decisions**

1. I have configured the interrupt handler of Timer 1 to trigger the ADC conversion. However, Dr Shalan recommended directly configuring the ADC to conversion on Timer interrupt. I have tried to do that, but there were the following problems:
   1. The callback was quite short that no significant degradation was reported at high sampling rates
   2. CubeMX would only allow the subscription to Timer 3, which would require re-working Timers 1 and 3 from the beginning.

Therefore, I opted to keep things as is, since no performance is degraded there is little to win from the migration

1. The python live monitor was updating with every new value, which caused degraded performance for sampling rates > 20 samples per second (as python rendered a new frame for each new data point). Per Dr Shalan’s advice, I chose a constant frame rate of 5 frames per second, so the performance for high sampling rates became much better.