Heart Monitor (Demo 1)

By: Habiba Gamal (900151007)

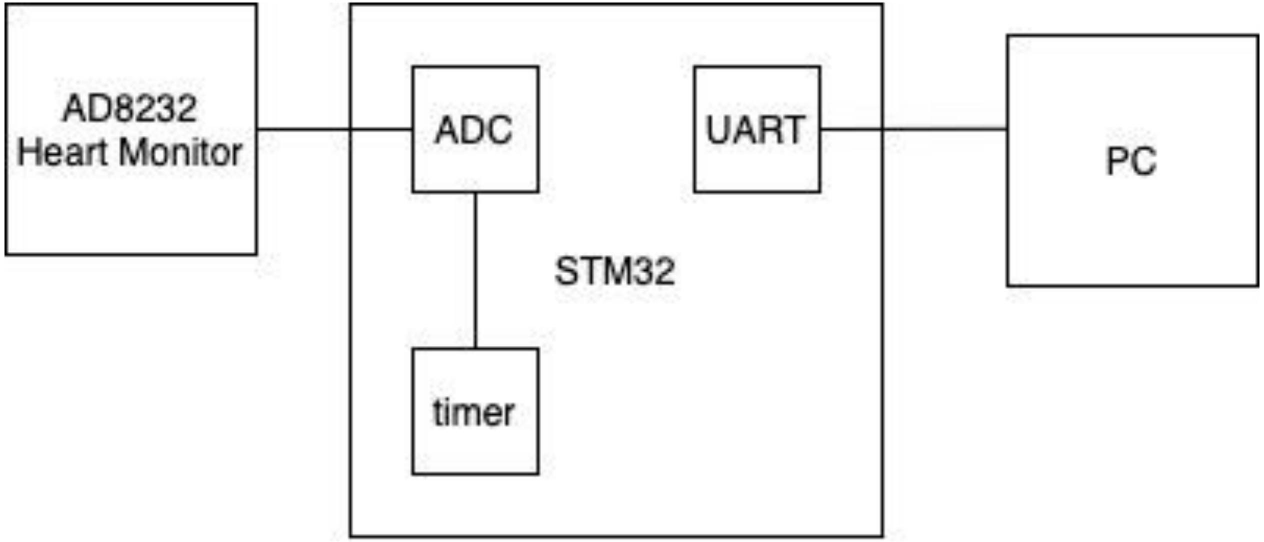
Current Progress

- ADC samples ECG signals
- Sampling frequency of ADC is determined by timer 3 (Timer 3 triggers ADC start of conversion)
- Timer 2 is used to count 1 minute (Used in “Collect 1 minute worth of data” command)
- The UART supports the receipt of the 3 required commands
- Python application is developed

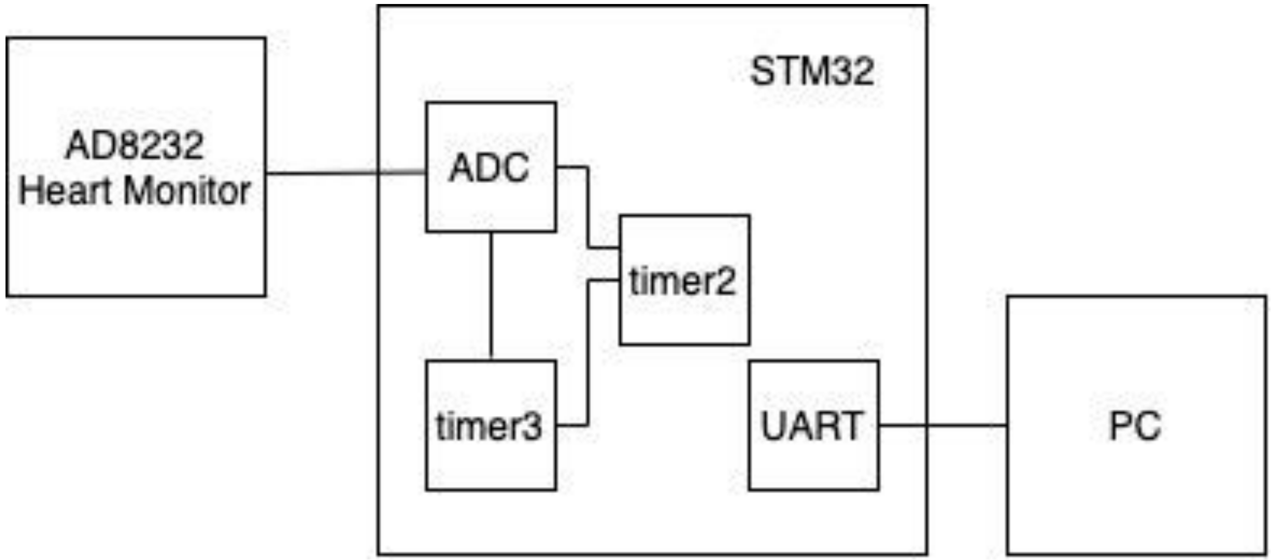
To-do

- Calculate heart rate (bpm)
- Increase the #of digits of sampling frequency to 7 (to support up to the maximum sampling frequency of 1MSPS)
- Auto-concatenate white spaces to the short commands (Now, it is done manually when inputting the command)

Block Diagram In Proposal



Updated Block Diagram



Design Details of Embedded Application

- UART receive generates an interrupt. If one of the three supported commands are sent, its operation starts
- If “start” command is sent, the ADC, timer2 and timer3 is started. At the end of each ADC conversion, the ADC converted value is read and sent through UART.
- The python application uses this value to plot the real-time graph

Design Details of Embedded Application (Cont’d)

- If “f=xxxx” is sent, the program reads “xxxx”, converts it to integer, then computes the period that corresponds to this frequency.
- The period of timer3 (which triggers the start of conversion of the ADC) is changes to match the computed period.
- The next “start” command will use this new frequency

Design Details of Embedded Application (Cont’d)

- If “bpm” command is sent, currently the “report” is sent through UART to indicate that the command is received
- In my to-do list, the “bpm” command will compute the bpm and then report it to the use through transmission over UART
- If none of the 3 commands is received, the UART transmits “no command” to indicate that it received unsupported command
- The python application will prevent this last case from happening, through checking the commands inputted by the user before sending them to the microcontroller

Design Details of Python Application

- Multi-threaded application
- Producer-Consumer architecture
- Thread 1 (Producer Thread)
  - Reads command line input
  - Waits until it is consumed
- Thread 2 (Consumer Thread)
  - Consumes read user input
  - Send this data to the microcontroller

Design Details of Python Application (Cont’d)

- Thread 3 (Printing thread)
  - Reads data from the microcontroller
  - Prints received data to the user
- Main thread
  - Plots the real-time ECG signals