

**Embedded Systems**

**Project Report**

**Mohamed A. Abdel Hamed**

**900163202**

**Dr. Mohamed Shalan**

**Spring 2020**

**Overview & Description**

Presented is a real-time embedded ECG heart monitor system application. The embedded side is to operate on an STM32F1 microcontroller (e.g. Black Pill) while the desktop-side is to run on any Python3- supporting environment. The ECG sensor being used is the SparkFun AD8232. The application provides the two basic functionalities of a heart monitor, namely, heart activity plotting, and heart beat rate reporting. Furthermore, the app enables the user to control some side features such as sampling and baud rates in addition to the COM port used. The application also provides a graphical control panel for these features.

**This report is dedicated to discussing the following:**

1. **Development Environment & Technology**
2. **System Features**
3. **Design & Implementation**
4. **User Guide & Sample Runs**
5. **Future Development**
6. **Development Environment & Technology**

**Software**

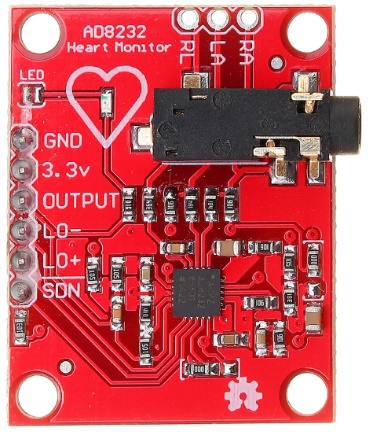
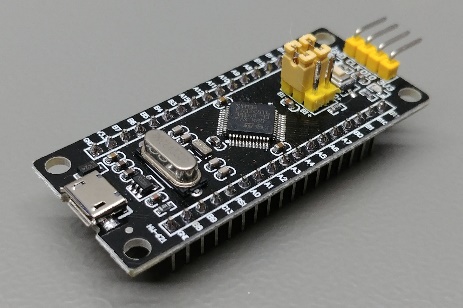
**Programming languages used**:

* **Black Pill app:** developed in C
* **PC app:** developed using Python 3.8.2

**Development software used:**

* **Black Pill app: Keil uVision IDE + CubeMX**
* **PC app:** 
  + **pySerial:** used for microcontroller-to-PC communication over a serial USB COM port
  + **matplotlib:** used for real-time plotting
  + **GUI:** used for graphical interface

**Hardware**

* **Microcontroller:** STM32F1 Black Pill
* **ECG Sensor:** SparkFun AD8232
* **Serial Cable**: prolific USB-UART cable

1. **System Features**

**Functional Features:**

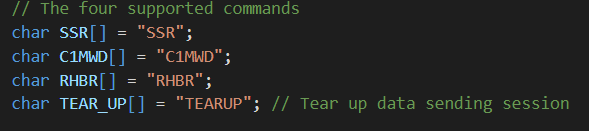
1. **Serial Port Selection:** in the beginning, the user is prompted with the available COM ports for them to select the one, to which the microcontroller is hooked.
2. **Baud Rate Control:** along with the port, the user is also prompted to set the desired baud rate (115200 is recommended).
3. **Sampling Rate Control:** at any point during operation (except for when ECG data collection is ongoing), a user can set the desired sampling rate to be used for collecting heart activity data.
4. **Collect & Plot ECG Data for 1 Minute:** a user can start the one-minute ECG data collection mode where the plot window is launched and updated real-time. Meanwhile, any other interaction is frozen until the plot window is closed.
5. **Report Heart Beat Rate:** after at least one data collection run, the application can report the average HBR based on the collected data.

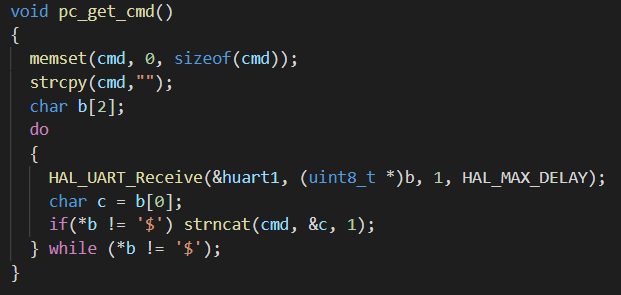
**Non-functional Features:**

1. **Leads-off Detection:** the system detects if the sensor pads were not attached properly at some point and produces a flat middle line in the plot.
2. **Premature Data Collection Cancelling:** in case the user closes the plot window midway through an ongoing data transmission, the desktop app signals the microcontroller to halt the data transmission from its side.
3. **Design & Implementation**

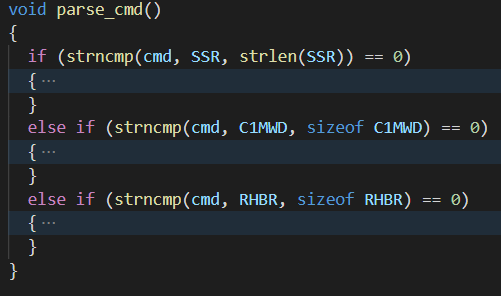
**Low-level utilities:**

1. **Command Extraction:** all commands read from the serial port byte by byte, and are delimited by ‘$’ as shown below:





1. **Command Parsing:** once a command has been extracted, it is then passed for parsing it by its predefined handle (e.g. SSR), and then the corresponding procedure is selected based on that:



**In the following two sub-section, the implementation of each of the aforementioned System Features are to be discussed.**

**Functional Features:**

1. **Serial Port Selection:**

This is readily achieved through the pySerial utility tools as shown below:



Note that it takes some time initially to detect the available ports (~5 seconds)

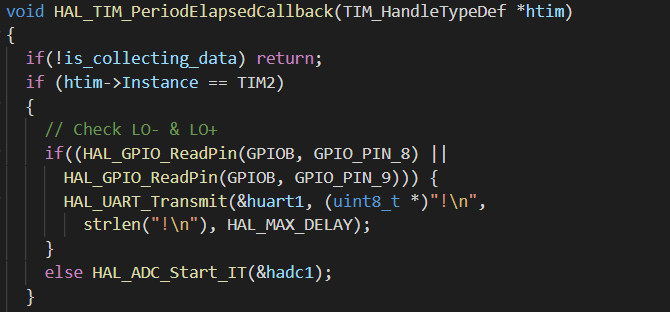
1. **Baud Rate Control:**

This is specified when the serial port is opened initially:



1. **Sampling Rate Control:**

Sampling rate controlled through the period of GPTIM2 which triggers the ADC1 conversions with each elapsed period when uC is in data collection mode.



When a user sets the heart beat rate through the control panel, a formatted command is sent to the microcontroller (via UART), which gets parsed and the auto-reload register (ARR) of the GPTIM2 is adjusted according to the following formula:

* ARR =

Given that the GPTIM2 source clock is 8 MHz, and the prescaler is fixed at (8K – 1), so the timer trigger rate becomes equivalent to the sampling rate. The allowed range for the sampling rate is between 1 and 1000.

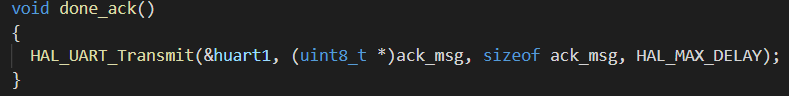
1. **Collect & Plot ECG Data for 1 Minute:**

When this mode is triggered by the user, the microcontroller stops waiting for new commands and keeps sending numerical data representing the pulse magnitude relative to the ADC range (12 bits = 0 to 4095). Every data reading is sent in a new line as shown:



When the uC is done sending data, it sends a pre-set ACK msg to signal the desktop app to stop updating the plot.





The real-time plot update is done by periodically redrawing the plot to reflect any new changes. In this app, the plot is refreshed every 10 ms:



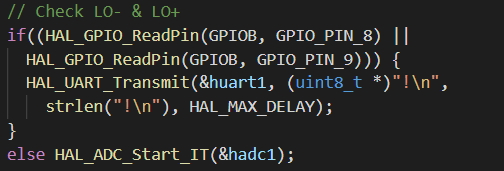
So that achieves a refresh rate of around 100 fps. It is advised to reduce that rate for devices (PCs) that cannot support that refresh rate.

1. **Report Heart Beat Rate:**

This is done by counting the beats detected during a 1-minute data collection. A beat is defined as a pulse that’s at least ~60% of the ADC scale in magnitude. Due to limitations on the operation guarantees at the user side, the heart beats counter is reset before each data collection to prevent previous poorly operated (e.g. disconnected pads) runs from corrupting the HBR.

**Non-functional Features:**

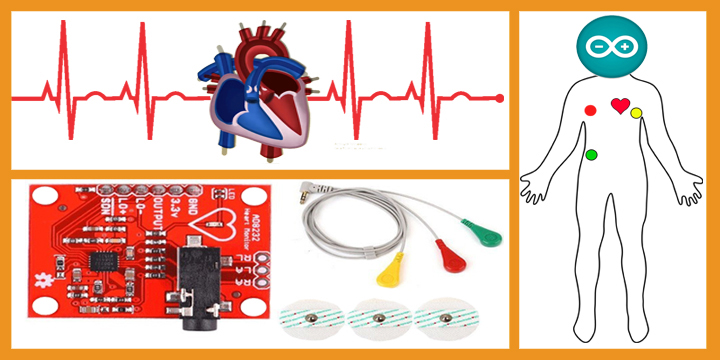
1. **Leads-off Detection:** this is done by digitally reading the GPIO pin value of the LO+ & LO- sensor signals, and if they’re high, the microcontroller sends a special pre-set character ‘!’ to signal the PC app to flatten the plot at that point.

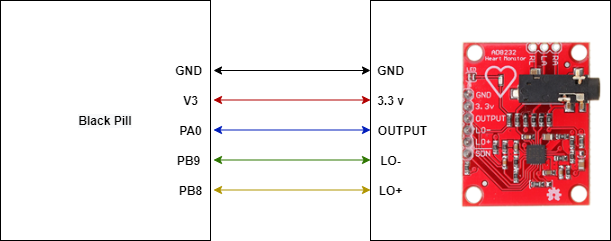


1. **Premature Data Collection Cancelling:** this is achieved by sending a pre-agreed character ‘#’ to the uC to signal it to stop.
2. **User Guide & Sample Runs**

This guide assumes the user has the hardware described before (along with a means to download the hex code to the uC) in addition to the required software (including the Python packages).

* Below are the steps to use the system:
  1. Make sure all connections are as shown below.
  2. Download the hex code to the Black Pill uC (e.g. though serial programming)
  3. Put the uC into operation mode (if you’re using SP) and click the Reset button.
  4. Go to the “**main\_app**” directory and run the following command “**python** **main.py**”





**Sample Run:**

1. **Future Development & Improvement**

The following could be a potential improvement on the presented system:

1. Using DMA for managing ADC conversions instead of interrupts
2. Utilizing the SDN (shut down) signal of the AD8232 sensor to save power when we’re not sampling its output.
3. Using a smoother technique for updating graph data in real-time
4. Adding a layer of fault tolerance around port binding in case of failures to connect to UART.

**References**

* **AD8232 Sensor Datasheet**
* **STM32F1 Datasheet**
* [**https://learn.sparkfun.com/tutorials/graph-sensor-data-with-python-and-matplotlib/update-a-graph-in-real-time**](https://learn.sparkfun.com/tutorials/graph-sensor-data-with-python-and-matplotlib/update-a-graph-in-real-time)
* [**https://pythonhosted.org/pyserial/index.html**](https://pythonhosted.org/pyserial/index.html)