HEART BEAT MONITOR REPORT

Aya Farag

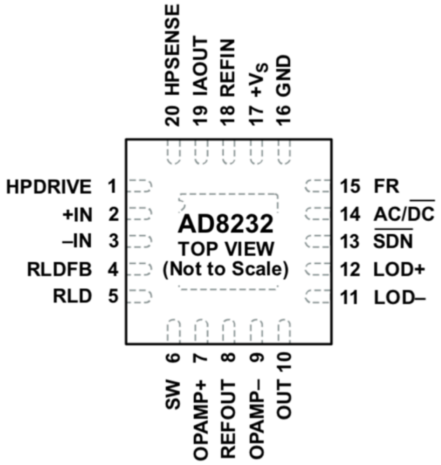
900160580

Overview:

For my project I’ve chosen to implement a simple heart monitor, using the ECG Sensor AD8232, and STM32 Microcontroller. The Electrodes are placed on someone for the ECG to detect the heart beat, and then the output from the ECG Sensor is sent as an analog signal to the STM32 Microcontroller. There are two ADC’s located on the microcontroller, so I used the first ADC , and I connected the output from the sensor to it through pin A0. Then I perform the computation on the microcontroller and then the result is send to the PC using UART, where I display the heart beat live and then view the computed Beats Per Minute.

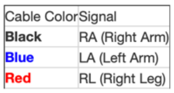
Hardware/Pins

The ECG sensor outputs an analog signal, so I will use an ADC 3202/On Chip ADC, to convert the signal coming out of the ECG sensor before it is sent to the Microcontroller.



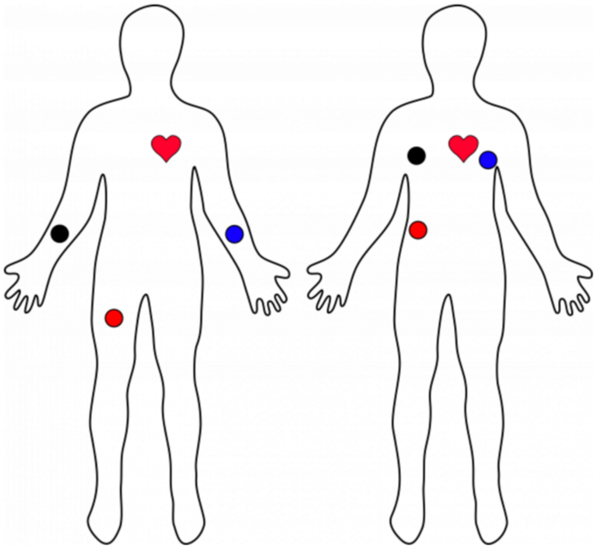
The ECG sensor has 15 pins. I will only be using 3-pins, GND, 3.3v, output. LO-, LO+, and SDN. LO-, and LO+ are leads -off comparators, LO- is always low, and LOD+ will be high. To detect leadoff, the ECG monitors the impedance between each differential-sensing electrode and the lead-off electrode. The impendence measurement provides an input for measuring the respiration rate. The pint OUT outputs the fully conditioned heart. Rate signal which will be connected to an ADC. The also wont be using the SDN pin, because its useful only for low-power applications. So the LO-, LO+ and OUT pins will be inputs to the PC.

Sensor Pad Placement



Cable Color	Signal
Black	RA (Right Arm)
Blue	LA (Left Arm)
Red	RL (Right Leg)

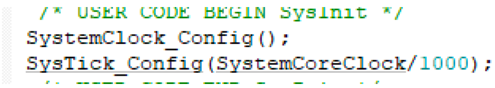
The closer the heart pads are to the heart, the better the measurements, so the diagram below shows how I should place the electrodes in order for it to get the best measurements.



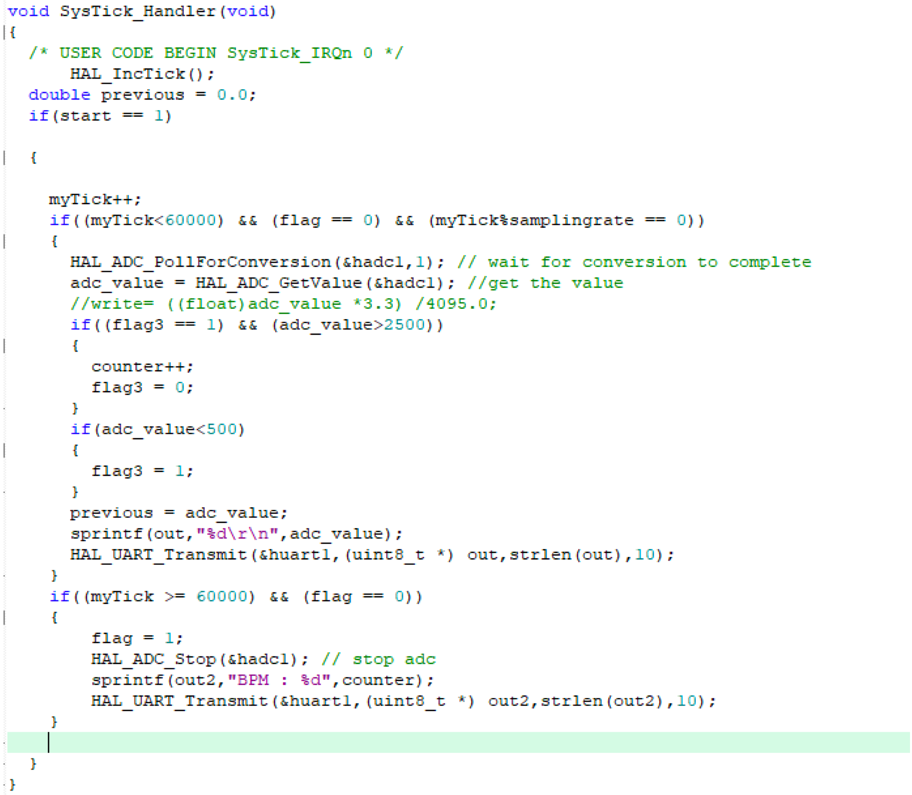
Functions:

- **C-Code:**
  - Syst-tick Handler()
  - UART-Handler()
  - CallBackFunction()
- **Python**
  - Animate()
  - Graph()

Code Walkthrough:

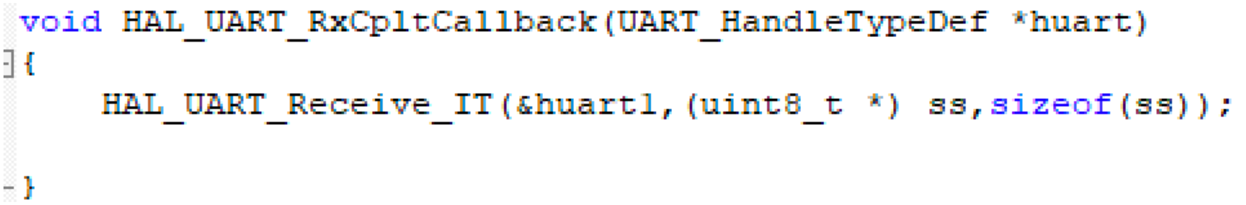


```
  /* USER CODE BEGIN SysInit */
 SystemClock_Config();
 SysTick_Config(SystemCoreClock/1000);
```

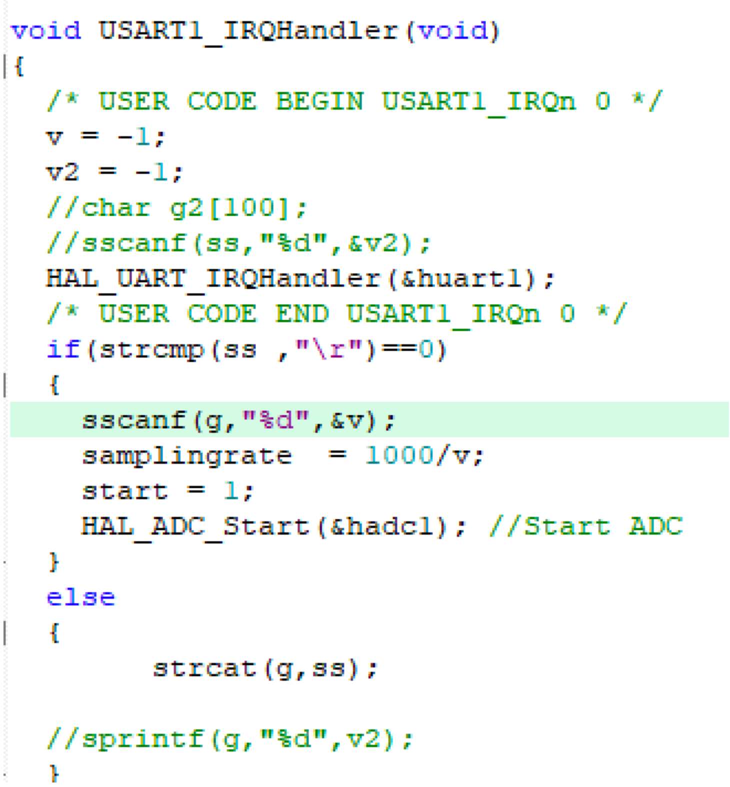


```
void SysTick_Handler(void)
{
  /* USER CODE BEGIN SysTick_IRQn 0 */
      HAL_IncTick();
  double previous = 0.0;
  if(start == 1)

  {

    myTick++;
    if((myTick<60000) && (flag == 0) && (myTick%samplingrate == 0))
    {
      HAL_ADC_PollForConversion(&hadc1,1); // wait for conversion to complete
      adc_value = HAL_ADC_GetValue(&hadc1); //get the value
      //write= ((float)adc_value *3.3) /4095.0;
      if((flag3 == 1) && (adc_value>2500))
      {
        counter++;
        flag3 = 0;
      }
      if(adc_value<500)
      {
        flag3 = 1;
      }
      previous = adc_value;
      sprintf(out,"%d\r\n",adc_value);
      HAL_UART_Transmit(&huart1,(uint8_t *) out,strlen(out),10);
    }
    if((myTick >= 60000) && (flag == 0))
    {
        flag = 1;
        HAL_ADC_Stop(&hadc1); // stop adc
        sprintf(out2,"BPM : %d",counter);
        HAL_UART_Transmit(&huart1,(uint8_t *) out2,strlen(out2),10);
    }

  }
}
```

In order for us to be able to time the data coming in and sample it for one minute, I used systick timer. So first of all I configure the Systick clock, by dividing the SystemCoreClocck by 1000, to transform it a millisecond delay between each consecutive ticks. Then in the SysTick Handler, I start incrementing a counter variable called myTick, that increments every millisecond. So to achieve a minute I need myTick to keep ticking until 60000, which means that I have read enough data for 1 minute. So what I do is poll for conversion using the ADC located on the micontroller and read the value from it. It reads the first sample and I have a flag which is set to one initially. Every time it detects a falling edge it is set to 1 and everytime theres a high edge higher than 2500 it is set to 0, and I do that so that I'd be able to get the Beats Per minute, wbich I will discuss more in details in the following sections. After that I transmit the value I received from the ADC to the terminal using UART. After I sample for one minute I transmit the beats per minute.



```
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
    HAL_UART_Receive_IT(&huart1,(uint8_t *) ss,sizeof(ss));

}
```

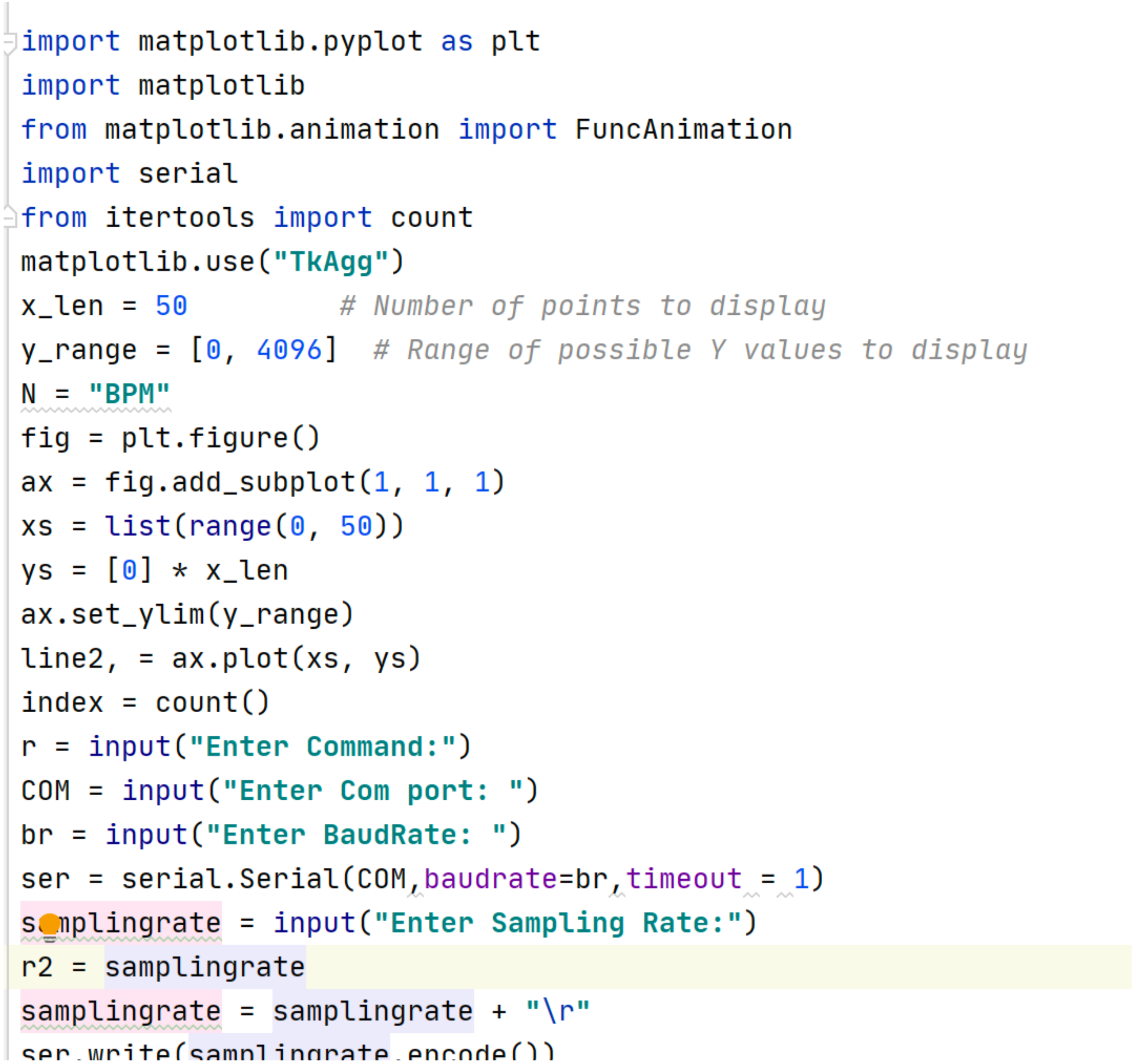


```
void USART1_IRQHandler(void)
{
  /* USER CODE BEGIN USART1_IRQn 0 */
  v = -1;
  v2 = -1;
  //char g2[100];
  //sscanf(ss,"%d",&v2);
  HAL_UART_IRQHandler(&huart1);
  /* USER CODE END USART1_IRQn 0 */
  if(strcmp(ss ,"\r")==0)
  {
    sscanf(g,"%d",&v);
    samplingrate  = 1000/v;
    start = 1;
    HAL_ADC_Start(&hadc1); //Start ADC
  }
  else
  {
        strcat(g,ss);

  //sprintf(g,"%d",v2);
  }
```

To be able to receive the user input for samples per second and I press enter then I receive using UART interrupts in the call back function, and after that in the UART Handler, I parse , in order to be able to get the sampling rate and I send it to the systick function. If we look at the code above you can. See to sample there's an if statement , and I check if myTick%samplerate ==0 then I should sample. In the UART Handler, after I receive the read sample from the call back function, I check if there is an enter sign , once there is an enter that means I've received completely the number needed for sampling rate. Then I divide a 1000 divided by the sampling rate received as an input , I set a start flag to one which starts the ticking in the systick function and after that I start the ADC. Here I start receiving the values from the

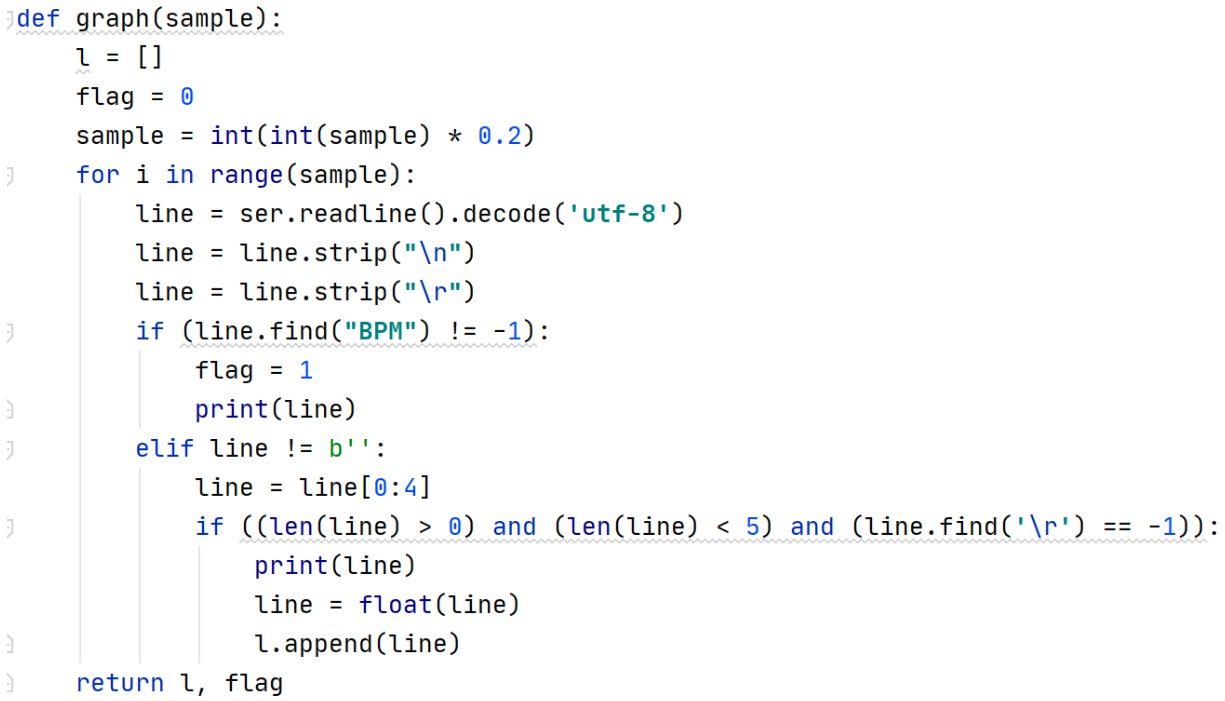
ECG. I send the flag to the systick handler, where it starts ticking, where then the reading and conversion is done, and the computations are done.

Python Code:



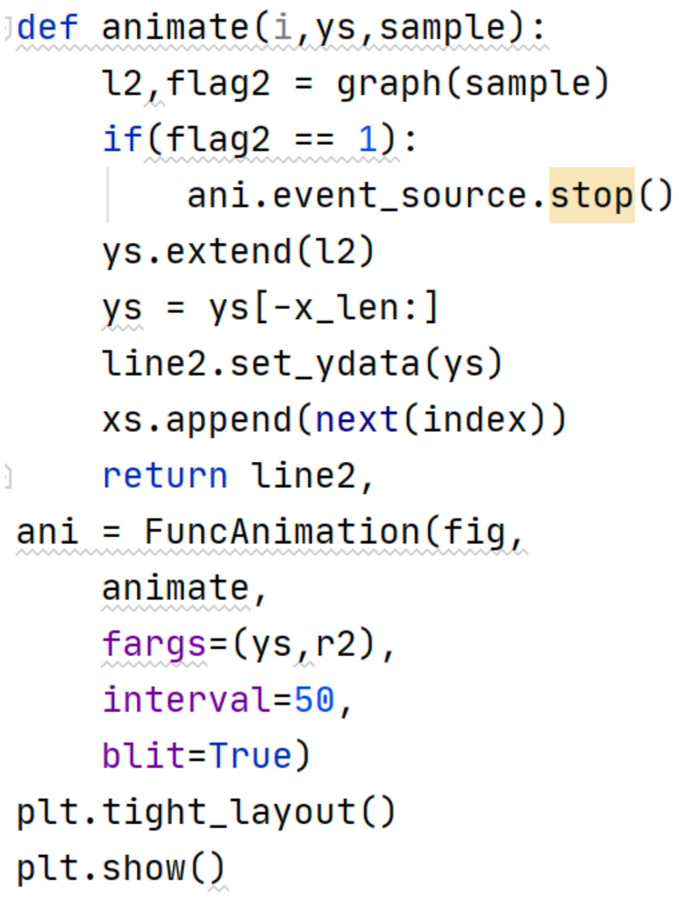
```
import matplotlib.pyplot as plt
import matplotlib
from matplotlib.animation import FuncAnimation
import serial
from itertools import count
matplotlib.use("TkAgg")
x_len = 50         # Number of points to display
y_range = [0, 4096]  # Range of possible Y values to display
N = "BPM"
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
xs = list(range(0, 50))
ys = [0] * x_len
ax.set_ylim(y_range)
line2, = ax.plot(xs, ys)
index = count()
r = input("Enter Command:")
COM = input("Enter Com port: ")
br = input("Enter BaudRate: ")
ser = serial.Serial(COM,baudrate=br,timeout = 1)
samplingrate = input("Enter Sampling Rate:")
r2 = samplingrate
samplingrate = samplingrate + "\r"
ser.write(samplingrate.encode())
```

Here in the python code , I use it in order to allow users to input com port , baud rate and sampling rate. After I receive the sampling rate from the user I write it using pyserial to the microcontroller in order for me to start receiving data. So that’s what I do first, I ask for user input and then open the port. In the below code I also start setting the lists that I will use to graph. So to graph, I graph using a live plot, so as I'm receiving the data, I view it live, and since there are so many samples I limit the samples viewed by viewing 50 samples at a time by sliding the graph. Otherwise all the results are compressed and stuck to each other.



```
def graph(sample):
    l = []
    flag = 0
    sample = int(int(sample) * 0.2)
    for i in range(sample):
        line = ser.readline().decode('utf-8')
        line = line.strip("\n")
        line = line.strip("\r")
        if (line.find("BPM") != -1):
            flag = 1
            print(line)
        elif line != b'':
            line = line[0:4]
            if ((len(line) > 0) and (len(line) < 5) and (line.find('\r') == -1)):
                print(line)
                line = float(line)
                l.append(line)
    return l, flag
```

In the function above, I send to it the sample rate, and I multiply it by 20% and I append the values to a list with that size. So I'm doing this because when the sampling rate increased, the graph stopped updating as fast the sampling rate causing a delay and that was due to the fact that.I'm updating after every frame, so to solve this, instead of updating after every single sample, I append a certain number of values to the list and update them at one time.



```
def animate(i,ys,sample):
    l2,flag2 = graph(sample)
    if(flag2 == 1):
        ani.event_source.stop()
    ys.extend(l2)
    ys = ys[-x_len:]
    line2.set_ydata(ys)
    xs.append(next(index))
    return line2,
ani = FuncAnimation(fig,
    animate,
    fargs=(ys,r2),
    interval=50,
    blit=True)
plt.tight_layout()
plt.show()
```

I call the graph function and I return it to a function called animate, and here is where I graph, by appending the y values. So what I do is receive the flag and list from the graph function, and if flag == 1 that means I should stop animating and graphing and flag is set to 1 when the word BPM is seen as an input, and I use it as my stopping condition, that if its in any of the inputs. that means I finished reading for one minute. Then I join both lists together, y2 and l2, and I set a certain length for the. size of the second list to not view all samples at the same time, and I set it as the y_axis by doing set_ydata, and I return it to the main. Then in the main I call the FuncAnimation function, which allows me to plot an animated graph. It takes my plot figure as an input the animate function, and the number of intervals. Animate does not need to be in a while loop as it keeps on calling itself until the animate function is stopped using event_source.stop(). After that I plot the. Values that I have received and I output the BPM and I’m done.

Computation:

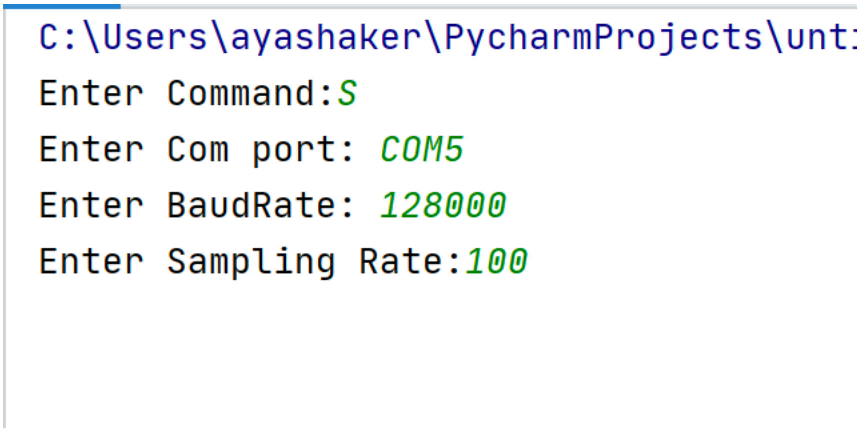
As I’m receiving the values from the ECG Sensor, I always check it meets a certain threshold and that their was a falling edge in the samples before it, and this way it would mean it detected a heart beat. I did this because there is a lot of noise in the middle and in order to detects a correct or almost close heart beat I always check for the rising and falling edge that meet a certain threshold, and once it is met I increment my counter until one minute is done, and then I transmit this value as my BPM.

Graphing:

In order to graph I transmit the values from the microcontroller to my python code, and then as I read each value, I append it to my list, and then I graph them. My range is between 0-4096, because the output from the ecg is ranged between those values.

Output:

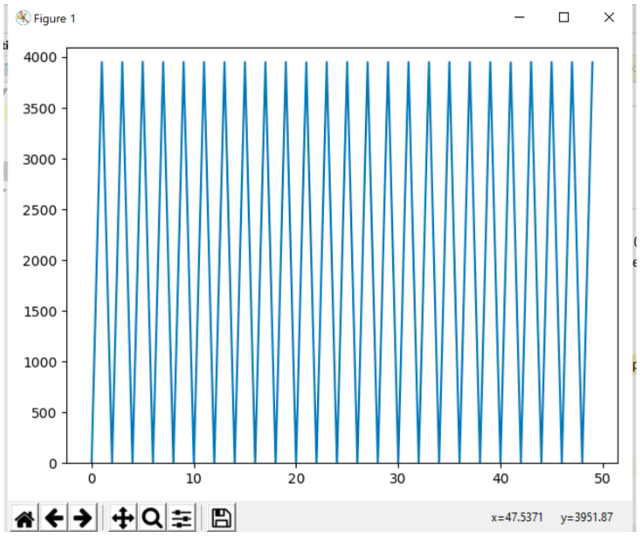
User Input:

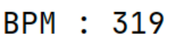


```
C:\Users\ayashaker\PycharmProjects\unt:
Enter Command:S
Enter Com port: COM5
Enter BaudRate: 128000
Enter Sampling Rate:100
```

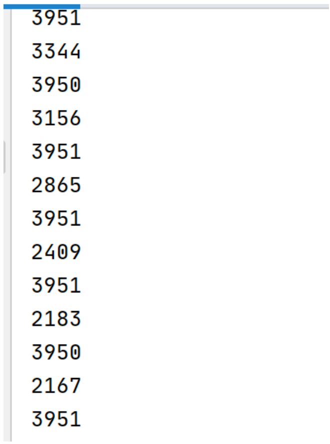
So first of all once you run the code, it will ask you to enter a command and in order to start, the user should enter the letter S, to indicate its time to start, after that the user should enter the COM port baud rate and the sampling rate, Once this is done the output starts being show to the user.

Results:



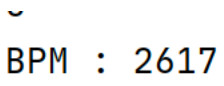


```
BPM : 319
```

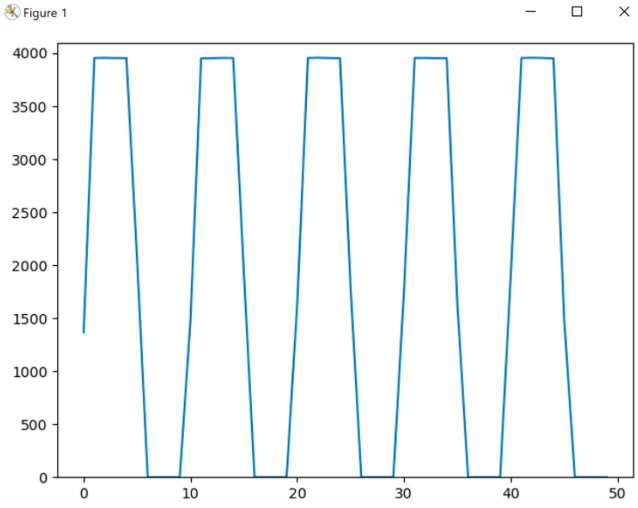


```
3951
3344
3950
3156
3951
2865
3951
2409
3951
2183
3950
2167
3951
```

The picture on the right shows, the output from the ECG and then it graphs them right and it outputs the beats per minute. The above code is a sampling rate of 100, and it outputs, 319 Beats Per Minute.



```
BPM : 2617
```



The Picture above shows the ECG graph for the sampling rate of 500 , and. The beats per minute is 2617.

GitHub Link:

https://github.com/ayashaker98/Embbeded-ProjectREP/tree/master/FinalProjectEmbedded