2020

Embedded Systems Project

SPRING 2020

AHMED ABOUZAID

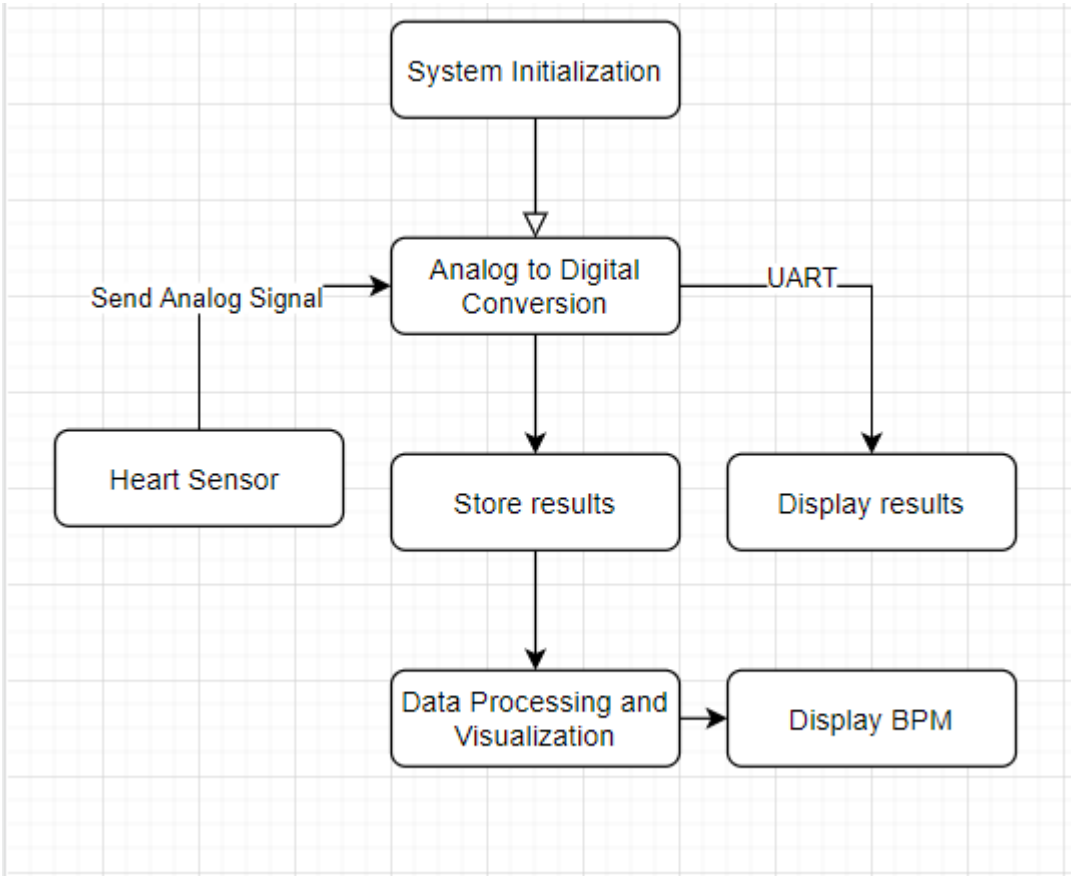
900163141

Table of Contents

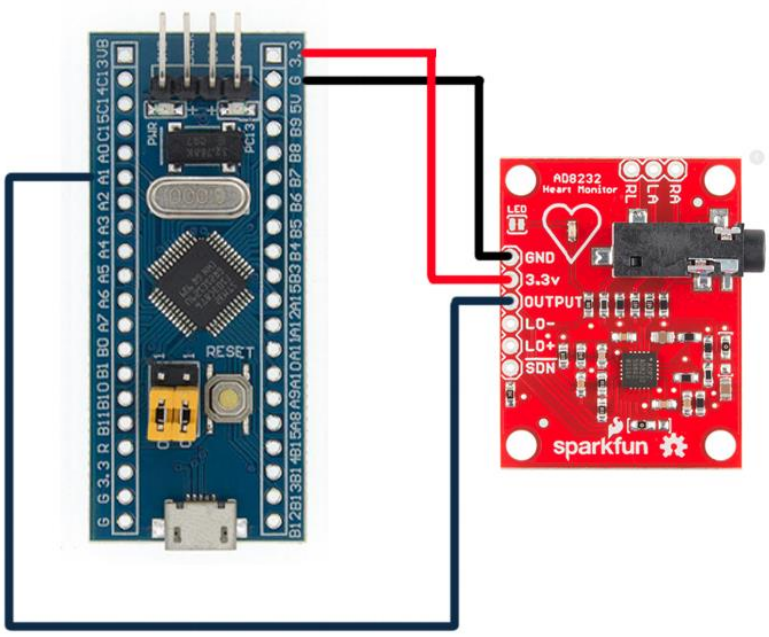
Project Objectives

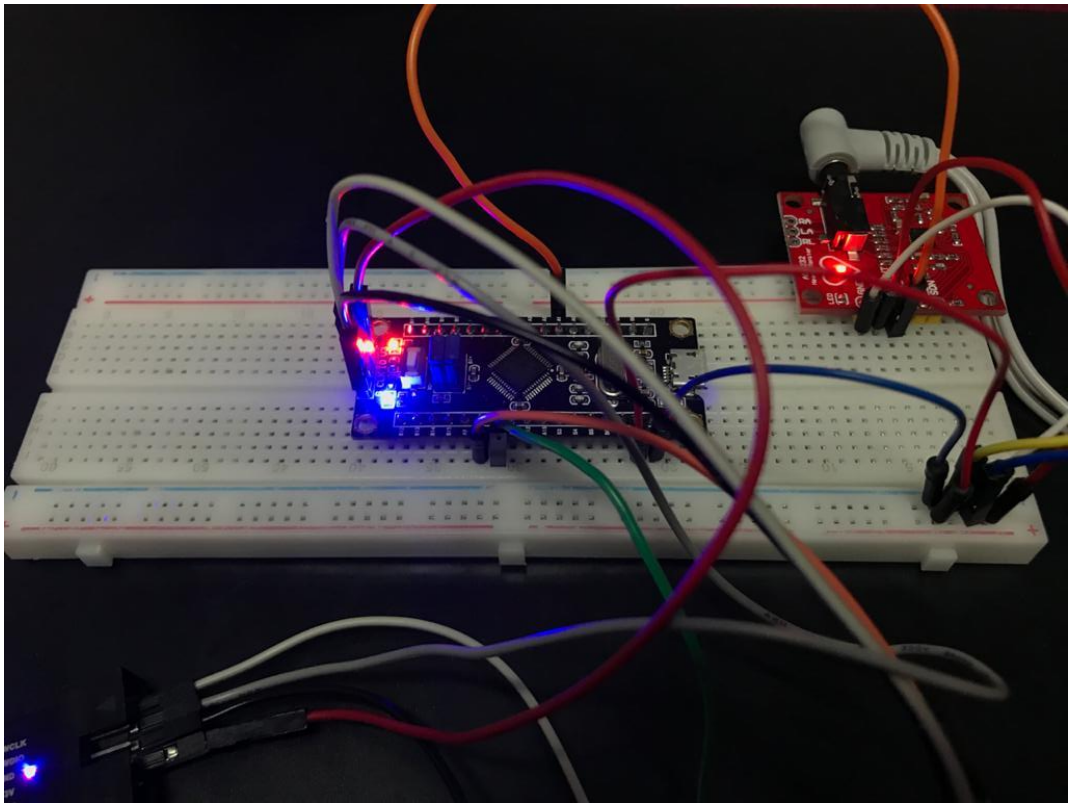
- Develop an embedded application using the STM32 module and the ECG sensor to collect ECG signal and report it to a PC over USB link for displaying and further analysis. The embedded code accepts the following commands from the connected PC over the serial link:
- Set the sampling rate - Collect 1-minute worth of data - Report the heart beat rate (bpm)
- The commands must be sent using an application running on the PC. The sent ECG data must be graphed using the same application. If you develop in Python, use pySerial and matplotlib. The application must provide UI elements to select the COM port and baud rate. Also, it must provide UI element to set the sampling rate.

Program Architecture



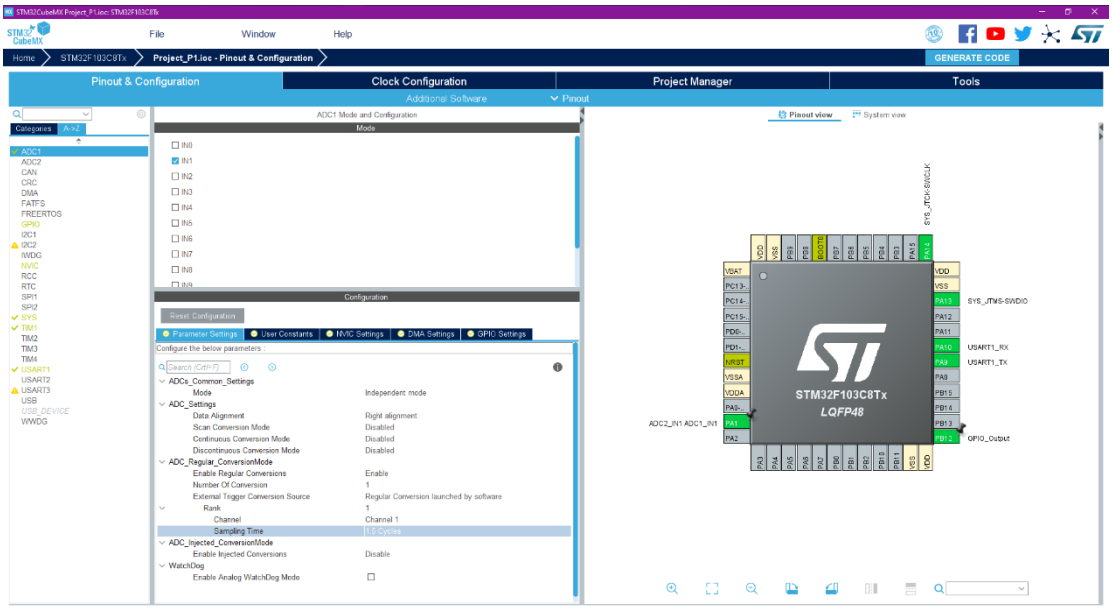
Circuit diagram



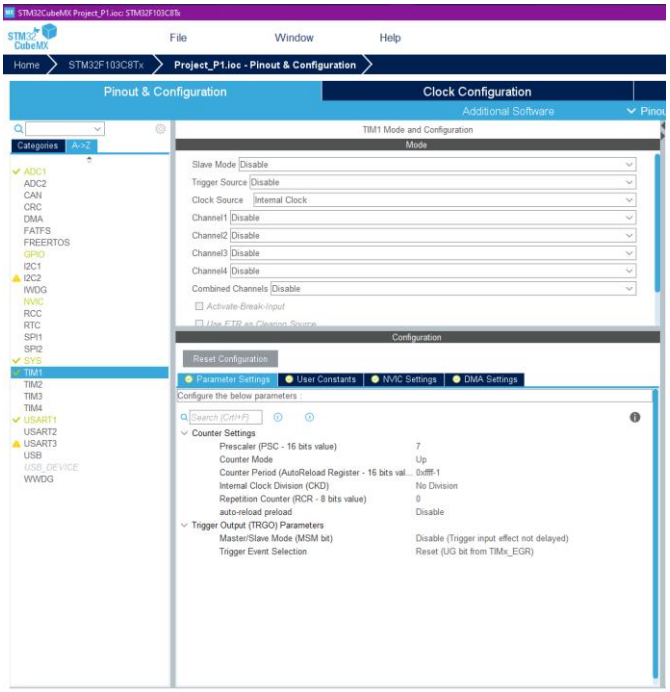


STM32CubeMX configurations

A1 pin is configured to be the analog input port which is connected to the ADC. Then, the signal is converted and sampled to be represented either in Tera Term (Milestone 1) or in a graph.



Timer 1 is configured to handle the sampling process which produces 150 sample per second using 8 MHz clock and Prescaler value equal to 7.



Code implementation

The project main while loop which starts the ADC and force them to convert the received analog values. Then the program output the results and repeat the process every 1MHz/sampling_rate

The sampling rate is set equal to 150.

```
  while (1)
  {
    /* USER CODE END WHILE */
    HAL_ADC_Start(&hadc1);
    if(HAL_ADC_PollForConversion(&hadc1, 1)==HAL_OK){
      res = HAL_ADC_GetValue(&hadc1);
    }
    HAL_ADC_Stop(&hadc1);
    sprintf(out, "%d\r\n", res);
    HAL_UART_Transmit(&huart1, (uint8_t*)out, sizeof(out),200);
    delay_us(1000000/sampling_rate);
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

The animate function calls the get_read function to receive the data from the serial port. Then, the data is stored in “arr” to calculated the bpm and to provide file that has the output data for 1 min

```
def animate(i, xs, ys):
    global count
    read = get_read()
    arr.append(read)
    measures = hp.process_segmentwise(arr, sample_rate=150)

    # Add x and y to lists
    xs.append(dt.datetime.now().strftime('%H:%M:%S.%f'))
    ys.append(read)

    # Limit x and y lists to 20 items
    xs = xs[-50:]
    ys = ys[-50:]

    # Draw x and y lists
    ax.clear()
    ax.plot(xs, ys)

    plt.xticks(rotation=45, ha='right')
    plt.subplots_adjust(bottom=0.30)
    plt.title('Heart Monitor')
    plt.ylabel('Heart sensor data')

ani = animation.FuncAnimation(fig, animate, fargs=(xs, ys), interval=1)
```

The UI receives three values from the user as shown below (port number, baud rate, and sampling rate) to send them as arguments to run the visualize_data.py script

```
def sel_port():
    global port_chosen
    port_chosen = port.get()
    print(port_chosen)

def sel_baudrate():
    global rate_chosen
    rate_chosen = rate.get()
    print(rate_chosen)

def sel_sampling():
    global sampling_chosen
    sampling_chosen = sampling.get()
    print(sampling_chosen)
```

How to run the project

clone/download the files from master branch "https://github.com/AhmedAbouzaid1/Heart-Monitor"

The source code must be loaded on the microcontroller

Run "start_ui.py"

Choose the baud rate and click "select" to save your choice choose the port number and click "select" to save your choice

Click "Start" to open the graph viewer and dispaly the heart sensor data.

Dependencies: (Libraries)

import sys

import os

import serial

import pickle

import heartpy as hp

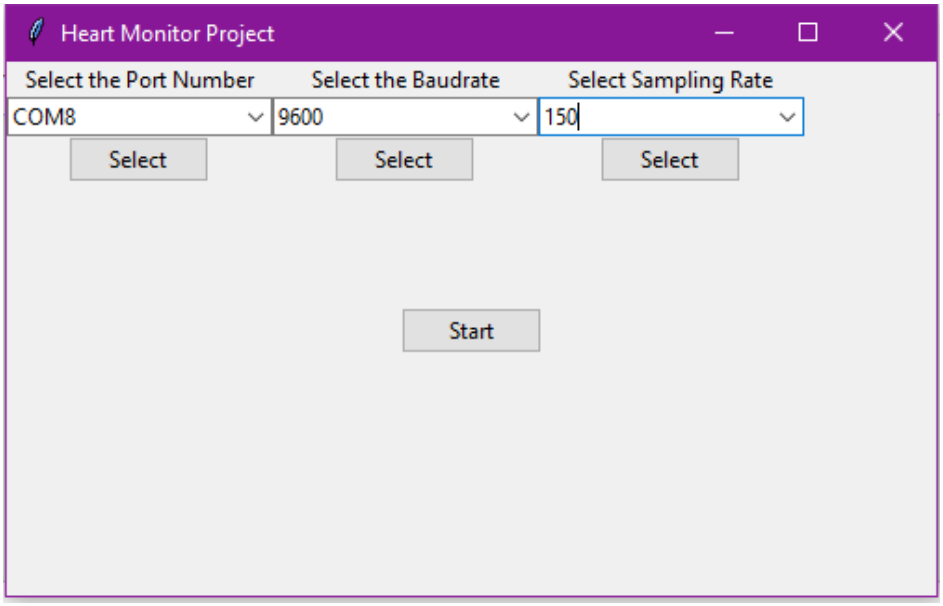
from time import sleep

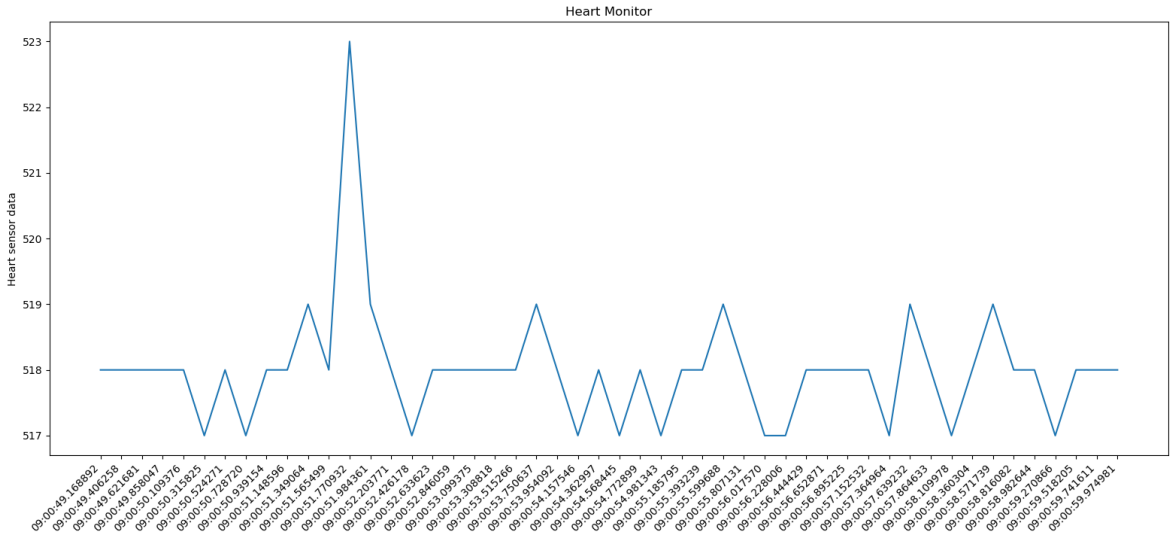
import datetime as dt

import matplotlib.pyplot as plt

import matplotlib.animation as animation

- Start main.py and select the desired values as shown below and click “Start”





Bpm calculation

- To calculate the bpm, “heartpy” lib is used to report the number pulses (spikes) per minute, but basically I calculate it every 15 seconds and multiply it by 4 and repeat the process every 15 seconds. (The below values are inaccurate since I have only 2 electrodes)