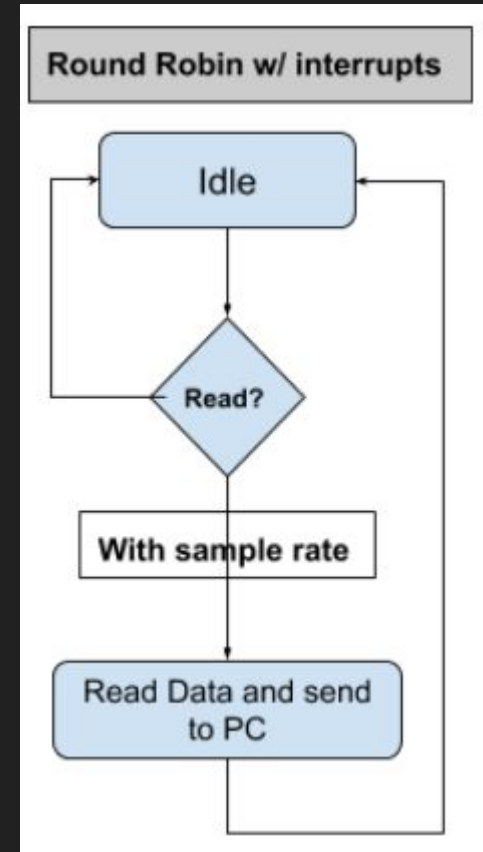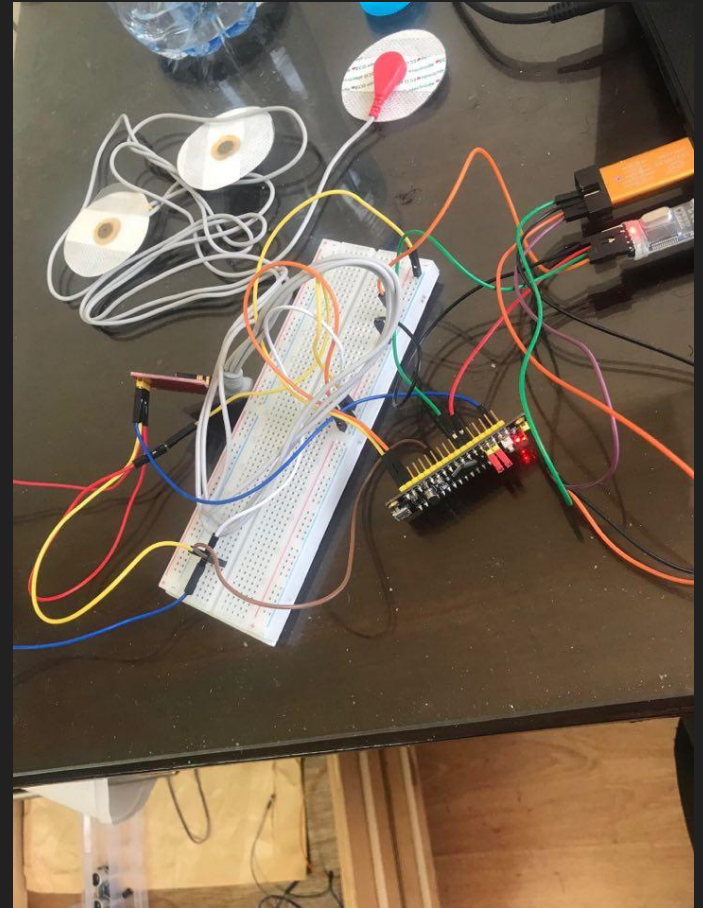# Heart Monitor

MileStone 1

Design

1. Round Robin with interrupts

2. Using ADC_Interrupt Handler

3. Using Systick Timer

4. Using UART_Interrupt Handler

5. Using main While(1) loop

6. Python GUI

Connections Schema

- Ground - Ground

- VDD - VDD

- Output - ADC Input GPIO

- SDN - Output GPIO

Communication

1. The microcontroller is initializing the heart rate monitor chip to being on the low power mode and waiting for a sampling request.

2. As the micro controllers receives the start bits (ss) from the python serail communication, it wakes up the heart rate monitor and starts the ADC conversion according to the requested sampling rate.

3. During an interval of 60 seconds, the python GUI plots the sampled data until the conversion stops and the heart rate monitor chip returns to low power mode.

1. UART Handler for non-poll receiving to receive the start bit at any time. This allows me to receive a new sampling command each it is issued by the user.

2. UART Handler is only responsible for receiving any coming commands from the user over UART.

```c
void USART1_IRQHandler(void)
{
  /* USER CODE BEGIN USART1_IRQn 0 */

  /* USER CODE END USART1_IRQn 0 */
  HAL_UART_IRQHandler(&huart1);
  /* USER CODE BEGIN USART1_IRQn 1 */
  HAL_UART_Receive_IT(&huart1,(uint8_t*)&s,sizeof(s));
}
```

Handlers (2)

1. Systick Handler: using the inner clock and configuring the inner timer systick with the sampling rate and therefore enters the handler every (8M/SampleRate).

2. Each tick generated will increment a counter (ctt) which indicates the total number of samples taken; responsible for ending the conversion after 60 seconds.

```c
void SysTick_Handler(void)
{
  /* USER CODE BEGIN SysTick_IRQn 0 */
if(flg == 1)
  {
  ctt++;
  HAL_ADC_Start_IT(&hadc1);
    //s HAL_UART_Transmit_IT(&huart1,(uint8_t *)
  }
```

1. The processor enters the ADC Handler only when the systick handler issues the ADC_Start command to maintain the sampling rate.

2. The adc handler is responsible for getting the data from the ADC and transmitting the it back through the UART.

```c
void ADC1_2_IRQHandler(void)
{
  /* USER CODE BEGIN ADC1_2_IRQn 0 */

  res = HAL_ADC_GetValue(&hadc1);
  sprintf(out, "%d", res);
  HAL_UART_Transmit_IT(&huart1,(uint8_t*)&out,sizeof(out));
  /* USER CODE END ADC1_2_IRQn 0 */
  HAL_ADC_IRQHandler(&hadc1);
  /* USER CODE BEGIN ADC1_2_IRQn 1 */

  /* USER CODE END ADC1_2_IRQn 1 */
}
```

Main While(1) Loop

1. The core of program is done in the main loop here.

2. It checks if a proper form of the sampling command has been issued.

3. It check if the number of samples has exceeded Samples*60 second; to terminate the sampling.

4. It wakes up the heart monitor when a sampling command has been issued.

5. It puts the heart monitor to sleep when after the 60 seconds.

```c
while (1)
{

if(s[7]=='s' && flg3 == 0)
{
if(flg ==0)
{

  strncpy(kk,s+2,4);
  //sscanf(kk, "%d", &m);

  sscanf(kk, "%d", &mk);
  flg = 1;
  HAL_GPIO_WritePin(GPIOB,GPIO_PIN_11,GPIO_PIN_SET);
  HAL_SYSTICK_Config((1.0/mk) * 8000000 - 1);
}

if(ctt > mk*60)
  flg3 = 1;
}else if(s[0]=='s' && flg3==0)
{
}else
{
ctt = 1;
mk = 1;
flg2 =0;
memset(s, 0, sizeof(s));
memset(kk, 0, sizeof(kk));

flg3 = 0;
flg = 0;
HAL_GPIO_WritePin(GPIOB,GPIO_PIN_11,GPIO_PIN_RESET);
}
}
/* USER CODE END 3 */
}
```
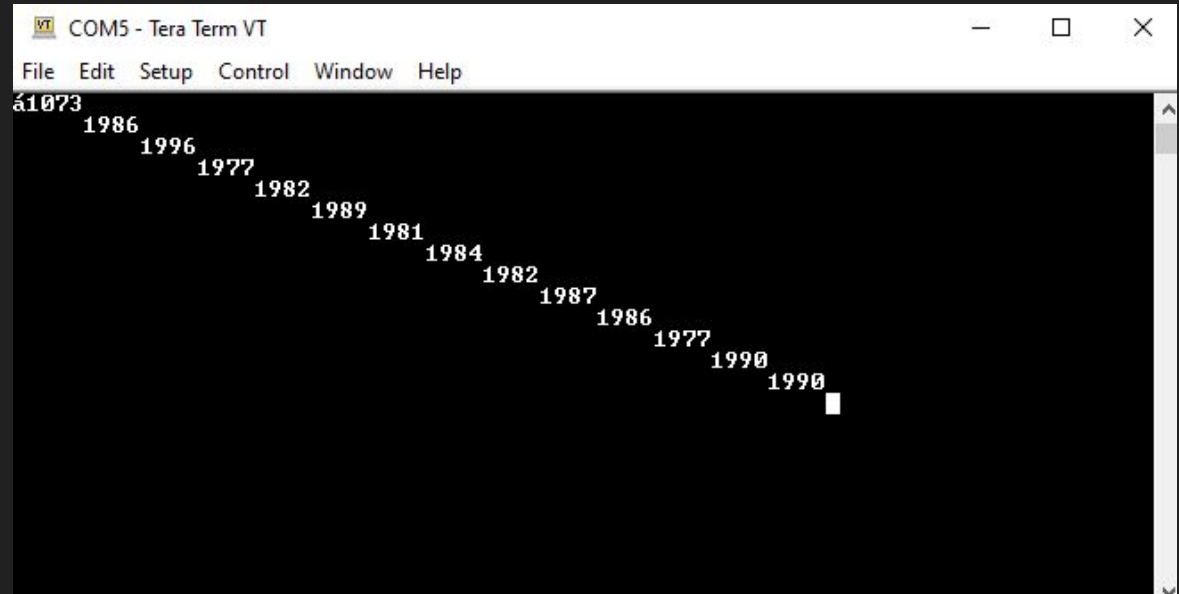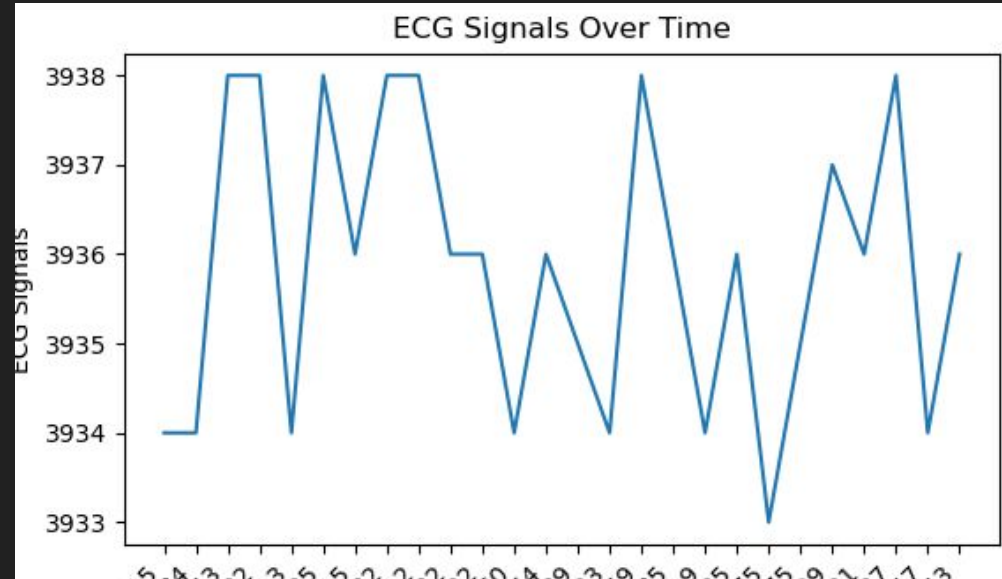
Tera Term Output

- The output of the tera term starts to appear as the start bit is sent which initializes the chip, adc conversion, and transmission.

Using python

1. I used python to utilize a simple GUI to establish a serial connection with MC.

2. It also gets the data from the UART and load the upcoming data over a graph using Matplotlib

3. Outcome:

ECG Signals Over Time

Using Tkinter libraries in python:

1. As a simple GUI was required, I found the simple tkinter library easy to utilize.

2. Each entry or button is responsible for collecting and checking data. The check connection button is responsible for establishing the connection with the UART and therefore must be pressed before sampling

GUI Parameters

1. The user will enter the communication port and baud rate to estable the right connection with the UART port.

2. The sampling rate will also be determined around 150 - 1500 samples per second to ensure best result

3. After pressing start, the sample rate with the start bit will be sent across the UART in such form: ss<SAMPLERATE>ss ,i.e. ss0150ss

Git Hub

https://github.com/YoussefAmir/Embedded_Project.git