****

**Embedded Systems**

**Project Report**

**Arsani Sinout**

**900143721**

**Project Description**

The project requires developing an application that implements an embedded system that uses the STM32 module (connected with USB link) and connected to the ECG sensor module. The application should be running on the PC and should measure the heartbeat rate for 1-minute after the user sets the sampling rate, and show the output in a graphical way.

**Project Components**

**Hardware Components**

* STM32L432: Nucleo-32

A circuit board

Description automatically generated

* AD8232: ECG Module

A close up of a device

Description automatically generated

**Software Components**

* ARM Kiel (C)
* Python

**Pin Connections**

**A screenshot of a cell phone

Description automatically generated**

**A circuit board

Description automatically generated**

**Data Flow (Hardware)**

**Data Flow (Software)**

The project had several stages regarding the software development of the heart beat monitor using the Nucleo-32 board and the ECG module. First of all, to use the programable chip (Nucleo-32), STM32 CubeMX was used to configure the technologies to be used in the board and for its pin and clock configuration. Only UART2, the ADC and TIM1 were used in this project. The ADC is responsible for data receiving and conversion from the ECG module. The UART was used to communicate with the PC application (scripted in python). And finally the timer was sufficient for setting the sampling rate of the output.

A screenshot of a cell phone

Description automatically generated

As for the timer (TIM1), the prescaler value was set to 7 (so it would be 8) and the ARR (counter period) to 49999 (so it would be 50000) so to have the delay in micro-seconds.

**C Code**

The C code is the code that was used to program the microchip using ARM Kiel software. After configuring the pins for the microchip, a code was generated to insert in it the program and instructions that the microchip is required to do. The role of the nucleo-32 board in this project is to receive the desired sampling rate, reading and converting data from the ECG module and transmitting this data to the PC application in the suitable form to be displayed and processed. Both, receiving the sampling rate and transmitting the data to the PC application are done through the UART. However, capturing data from the ECG module and setting its sample rate (with accordance to the input from the UART) is done using the ADC and the TIM respectively as shown in the code below. The timer was used in order to have sampling in microseconds (not melliseconds as the HAL\_Delay does).

int main(void)

{

uint32\_t res;

char out[6];

out[5] = '\n';

SystemClock\_Config();

MX\_GPIO\_Init();

MX\_ADC1\_Init();

MX\_USART2\_UART\_Init();

MX\_TIM1\_Init();

HAL\_TIM\_Base\_Start(&htim1);

int trans\_rec = 0;

int srate = 0;

while (1)

{

if (trans\_rec == 0)

{

HAL\_UART\_Receive\_IT(&huart2 , (uint8\_t \*) srate, sizeof(srate));

if (srate > 0)

{

trans\_rec = 1;

}

}

else

{

HAL\_ADC\_Start(&hadc1);

res = HAL\_ADC\_GetValue(&hadc1);

sprintf(out, "%d\r\n", res);

HAL\_UART\_Transmit(&huart2, (uint8\_t\*)out, sizeof(out),200);

\_\_HAL\_TIM\_SET\_COUNTER(&htim1,0);

while (\_\_HAL\_TIM\_GET\_COUNTER(&htim1) < 100000 / srate);

}

}

}

**Python Code**

The python part of coding has the role of user interaction/user interface. The python code for the PC application is responsible for taking connection data (to the board through serial port) and establishing the connection. The application is required also to display the data according to the input sampling rate that was provided by the user through the user interface. The application, although, does not validate any input.

fig = plt.figure()

ax = fig.add\_subplot(1, 1, 1)

xs = []

ys = []

arr = []

count = 0

ser = serial.Serial()

def read():

a=''

while True:

if ser is None:

continue

try:

val = ser.readline().decode("utf-8")

return val

except:

print(" ")

def animate(i, xs, ys):

global count

val = read()

arr.append(val)

count = count + 1

print(count)

print(val)

# Add x and y to lists

xs.append(dt.datetime.now().strftime('%H:%M:%S.%f'))

ys.append(val)

# Limit x and y lists to 20 items

xs = xs[-20:]

ys = ys[-20:]

# 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')

def write():

global port

port = port.get()

print(port)

global brate

brate = brate.get()

print(brate)

global srate

srate = srate.get()

print(brate)

ser.port = port

ser.baudrate = int(brate)

ser.bytesize = serial.EIGHTBITS

ser.partiy = serial.PARITY\_NONE

ser.timeout = 2

ser.open()

ser.write(srate.encode())

ani = animation.FuncAnimation(fig, animate, fargs=(xs, ys), interval=50)

plt.show()

window = Tk()

window.title("Project")

window.geometry('250x100')

lbl = Label(window, text="Select Port Number")

lbl.grid(column=0, row=0)

port = Entry(window)

port.grid(column=1, row=0)

lbl = Label(window, text="Select Baudrate")

lbl.grid(column=0, row=1)

brate = Entry(window)

brate.grid(column=1, row=1)

lbl = Label(window, text="Select Sample Rate")

lbl.grid(column=0, row=2)

srate = Entry(window)

srate.grid(column=1, row=2)

btn = Button(window, text="Start", command=write)

btn.grid(column=1, row=3)

window.mainloop()

**User Interface**

A screenshot of a cell phone

Description automatically generated

**Output**

**A screenshot of a social media post

Description automatically generated**