13.01.2020

EEM 449

EMBEDDED SYSTEM DESIGN

PROJECT

Name: Yunus Emre

Surname: ESEN

Student No: 22280328940

Instructor: Assist. Prof. Dr. Hakan Güray ŞENEL

1. INTRODUCTION

This project has been created for project of EEM 449. The subject of that is design an application on TM4C1294XL that get temperature and pressure values from SENSORHUB and two different weather forecast sites. SENSORHUB is used to get a value of temperature from TMP006 sensor and get a value of pressure from BMP180 sensor. The other temperature values are taken from OpenWeatherMap and eskisehir.mgm.gov.tr sites. The taken values are stored in global variables and sent to a server on port 5011 in local pc network. These values are read via SocketTest.

Code Composer Studio 9.2 was used to create this project. Also, SocketTest v3.0 was used to read values from port 5011.

2. HOW DOES IT WORK?

This system uses clock, tasks and semaphores in TI-RTOS. Clock creates a new semaphore to trigger all tasks. Every task works in infinite while loop and they pend a semaphore to start and post another semaphore to start another task. All tasks, semaphores and clock function are described in figure 1 below.

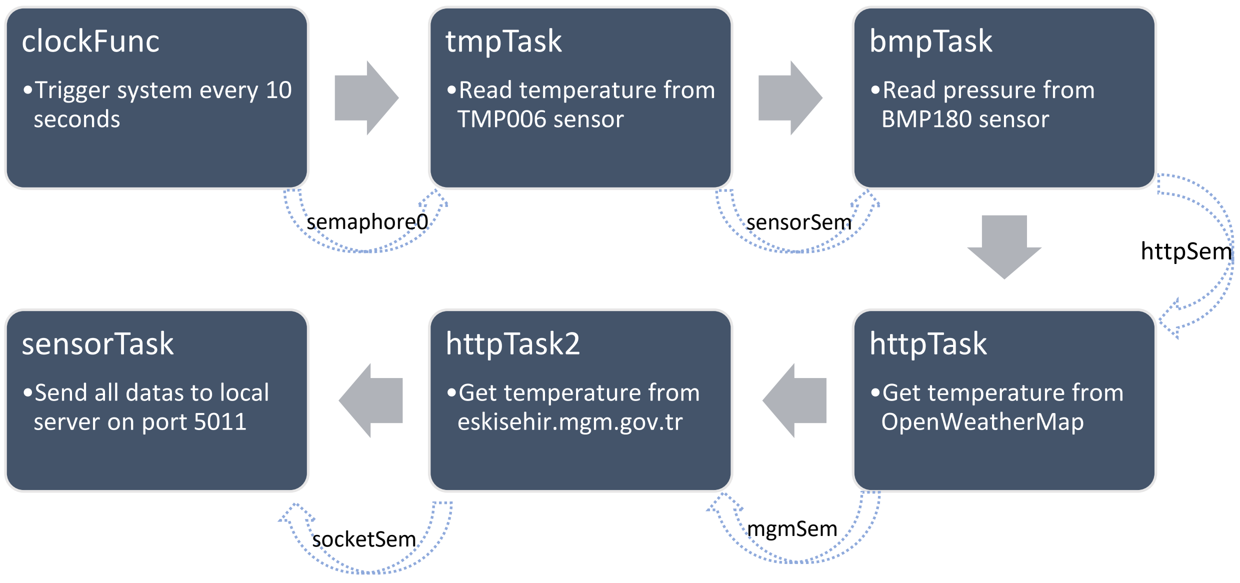

clockFunc
•Trigger system every 10 seconds
tmpTask
•Read temperature from TMP006 sensor
bmpTask
•Read pressure from BMP180 sensor
semaphore0
sensorSem
httpSem
sensorTask
•Send all datas to local server on port 5011
httpTask2
•Get temperature from eskisehir.mgm.gov.tr
httpTask
•Get temperature from OpenWeatherMap
socketSem
mgmSem

Figure 1: Working Principle

The structures used in the program are described below.

A. TASKS

There are 5 tasks in this project. These are tmpTask, bmpTask, httpTask, httpTask2 and socketTask that are described below.

tmpTask measures room temperature and writes this value to tempstr_tmp. It uses I2C communication. This task is created in XGCONF. It has an infinite while loop which pends a semaphore named semaphore0, and posts a semaphore named sensorSem to run bmpTask when finish the measurement process. Also, it prints temperature to the console.

bmpTask measures room pressure and writes this value to tempstr_bmp. It uses I2C communication. This task is created in XGCONF. It has an infinite while loop which pends a semaphore named sensorSem, and posts a semaphore named httpSem to run httpTask when finish the measurement process. Also, it prints pressure to the console.

httpTask makes a http GET request to OpenWeatherMap site. It gets data from api.openweathermap.org that is defined as HOSTNAME in the code and sends a request with the URI address defined as REQUEST_URI. It has an infinite while loop which pends a semaphore named httpSem. When it got the data, it searches for “temp”, after finding that searches “,” and returns last 6 chars. The unit of this temperature value is Kelvin. Temperature data is stored in tempstr as a global variable. It prints temperature to the console and posts a semaphore named mgmSem when all processes are done.

httpTask2 makes a http GET request like httpTask. Its main difference than httpTask creates a connection with eskisehir.mgm.gov.tr and different URI address that are defined as HOSTNAME2 and REQUEST_URI2, respectively. All processes are the same except finding temperature value from the data. To do that, it searches “renkMax”, when it found, it searches for “</” and takes last char to get temperature value and stores to tempstr_http. The end of the code, it prints temperature value to console and posts a semaphore named socketSem.

socketTask has an infinite while loop. This task pends socketSem and gets the whole global variables and put all of them in printString. After that, it sends this to the server with sendData2Server function. The datas are putted on port 5011 of SOCKETTEST_IP that is predefined.

B. CLOCK

Clock module is allowed to create a trigger signal every 10 seconds in this project which can be changed later. clockFunc is created for this purpose in XGCONF with the period of 10000. Its trigger process is provided with a semaphore named semaphore0. The desired period was 10 seconds. It is changed because of the avoid of overlap between httpTask and httpTask2 processes.

C. SEMAPHORES

There are 5 semaphores in this project. These are semaphore0, sensorSem, httpSem, mgmSem, socketSem. Semaphore types are binary (FIFO).

semaphore0 is used for creating a trigger signal from clockFunc. It is posted from clackFunc and pended in tmpTask.

sensorSem is used to run bmpTask when tmpTask is over. It is posted from tmpTask and pended in bmpTask.

httpSem is used to run httpTask when SENSORHUB processes are finished. It is posted from bmpTask and pended in httpTask.

mgmSem is used to run httpTask2 when httpTask is over. It is posted from httpTask and pended in httpTask2.

socketSem is used to run socketTask when the all processes are done. It is posted from httpTask2 and pended in socketTask.

D. EVENTS (ADDITIONAL)

The all processes created also with events instead of semaphores. But using event created some lags in the program and it effected httpTask and httpTask2. In order to avoid this situation, the system established with semaphores was returned.

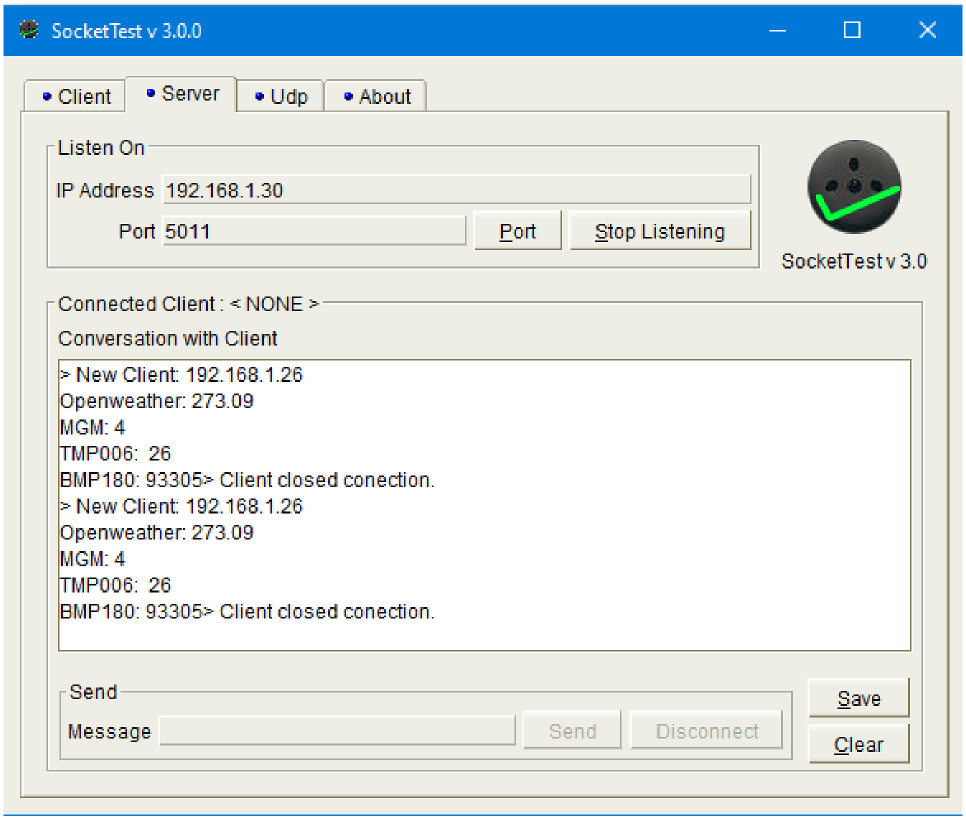
The system run with event was used an event named event0 and it had 5 different event Id instead of semaphores. It was easier to use compared to semaphores, but the lag that is mentioned before was making impossible to use. The project that is created with events is still in my workspace.

3. CONCLUSION

The console will look like this when the program is run.

```
Using MAC address in flash
Starting EEM449 project.
Service Status: DHCPC    : Enabled  :           : 000
Service Status: DHCPC    : Enabled  : Running   : 000
Anlik sicaklik: 26 (C)
Pressure: 93305
Network Added: If-1:192.168.1.26
Service Status: DHCPC    : Enabled  : Running   : 017
Sending a HTTP GET request to 'api.openweathermap.org'
HTTP Response Status Code: 200
Recieved 7437 bytes of payload
Temperature 273.09
Sending a HTTP GET request to 'eskisehir.mgm.gov.tr'
HTTP Response Status Code: 200
Recieved 24489 bytes of payload
Temperature 4 (C)
```

The datas that are send to the local network on port 5011 are showed as Figure 2 below.



IP Address 192.168.1.30
Port 5011
Connected Client : < NONE >
Conversation with Client

```
> New Client: 192.168.1.26
Openweather: 273.09
MGM: 4
TMP006:  26
BMP180: 93305> Client closed conection.
> New Client: 192.168.1.26
Openweather: 273.09
MGM: 4
TMP006:  26
BMP180: 93305> Client closed conection.
```

Figure 2: Datas are in SocketTest program

The IP address shown in Figure 2 is ipv4 address of local network connection which is mentioned before named SOCKETTEST_IP.

ERROR

Unfortunately, there is an error occurring after this program is executed. The reason of the problem is about DNS buffer. Generally, this error occurs when the program is trying to send a request seventh time and the error message appears as follows.

```
Sending a HTTP GET request to 'api.openweathermap.org'
nfo: Error: couldn't resolve host name "api.openweathermap.org"

Error! code = -106, desc = httpTask: address resolution failed
```

Error message

This error might be occurred because of the DNS buffer is got full according to my researches. I was trying to fix that problem. I think my need is create my own DNS and its buffer, and close it every time when my job is done. But, I couldn't what I thought.

As a result, this program takes temperature and pressure values from 2 different internet sites and sensors and sends them to local server on port 5011. But it works in a limited time. (I know it is not a good result for a RTOS system.)

Added Sections in the Code:

******************************** **Head of the Code** ********************************

```
#include <stdlib.h> //To use ltoa function
/* BMP 180den gelenler  */
#include "i2cbmp180.h"

#define Board_BMP180_ADDR 0x77
#define HOSTNAME          "api.openweathermap.org"
#define REQUEST_URI       "/data/2.5/forecast/?id=315202&
                          APPID=bb8f49f514e12ad4e63ed9b05a8e5e1a"
#define HOSTNAME2         "eskisehir.mgm.gov.tr"
#define REQUEST_URI2      "/tahmin-gunluk.aspx"
#define SOCKETTEST_IP     "192.168.1.30"

extern Semaphore_Handle semaphore0;
extern Semaphore_Handle sensorSem;
extern Semaphore_Handle httpSem;
extern Semaphore_Handle socketSem;
extern Semaphore_Handle mgmSem;
//extern Event_Handle event0;

char   tempstr[20]; //openweather temperature
char   tempstr_tmp[20]; //tmp006 temperature
char   tempstr_bmp[20]; //bmp180 pressure
char   tempstr_http[20]; //MGM temperature
char   printString[1024];
```

BMP180 I2C address has been added. (Also, I2C address of TMP006 was changed in Board.h) HOSTNAME2 and REQUEST_URI2 are added to get data from MGM. SOCKETTEST_IP is used with that value for creating object in local network. Semaphore handles are added to use semaphores. Tempstr values are strings to store values. printString is the string that is sended to server.

******************************** **clockFunc**********************************

```
Void clockFunc(void){
    /* Turn on sequence LED */
    GPIO_write(Board_LED1, Board_LED_ON);
    Semaphore_post(semaphore0);
    //Event_post(event0, Event_Id_00);
}
```

Every clock trigger is posted a semaphore. Also, LED D2 is on to see process is going.

******************************** **tmpTask** ********************************

```
Semaphore_pend(semaphore0, BIOS_WAIT_FOREVER); // wait for the semaphore
//Event_pend(event0, Event_Id_00, Event_Id_NONE, BIOS_WAIT_FOREVER);
sprintf(tempstr_tmp, " %d", temperature);
Semaphore_post(sensorSem);
//Event_post(event0, Event_Id_01);
```

Semaphores are pended and posted in the while loop. Temperature value is written in tempstr_tmp.

******************************** **bmpTask** ********************************

I made it a library named i2cbmp180.h to keep codes belongs BMP180.

```
Semaphore_pend(sensorSem, BIOS_WAIT_FOREVER);
//Event_pend(event0, Event_Id_01, Event_Id_NONE, BIOS_WAIT_FOREVER);
uint32_t pressure;
pressure = (int)press;
ltoa(pressure, tempstr_bmp); //int to string (global variable)
System_printf("Pressure: %d\n", pressure);
Semaphore_post(httpSem);
//Event_post(event0, Event_Id_02);
```

Semaphores are pended and posted. pressure is created as a 32-bit unsigned integer. press was a float and, the need is integer to get value. To do that, type casting was performed. Also, this value must be converted to string and sprintf function can not do that because of the stack overflow. ltoa is used for that reason and putted data to tempstr_bmp.

******************************** **httpTask** ********************************

```
Semaphore_pend(httpSem, BIOS_WAIT_FOREVER);
//Event_pend(event0, Event_Id_02, Event_Id_NONE, BIOS_WAIT_FOREVER);
Semaphore_post(mgmSem);                // activate mgm httpTask2
//Event_post(event0, Event_Id_03);
Task_sleep(1000);                      // sleep 1 seconds to avoid construct error
HTTPCli_destruct(&cli); // I took this line inside to while loop
```

Semaphores are pended and posted. Destruct operation is added in while loop to close all connection and avoid to stack overflow error. Sleep operation is made, because http destruct can give an error when it is done immediately.

******************************** **httpTask2** ********************************

```
Semaphore_pend(mgmSem, BIOS_WAIT_FOREVER);
//Event_pend(event0, Event_Id_03, Event_Id_NONE, BIOS_WAIT_FOREVER);
else {
      // string is read correctly
      // find "renkMax" string
      s1=strstr(data, "renkMax");
      if(s1) {
            if(temp_received) continue;      // temperature is retrieved before,
                                             // continue
            //find "<\" to reach temperature value
            s2=strstr(s1, "</");
            if(s2) {
                  *s2=0;                           // put end of string
                  strcpy(tempstr_http, s1+17);     // copy the string
                  temp_received = 1;

                  }
            }
      }
Semaphore_post(socketSem);  // activate socketTask
//Event_post(event0, Event_Id_04);
```

Semaphores are pended and posted. The data coming from eskisehir.mgm.gov.tr hold a value after “renkMax” and before “</”. These operations are done to find that value.

******************************** socketTask *******************************

```
Semaphore_pend(socketSem, BIOS_WAIT_FOREVER);
//Event_pend(event0, Event_Id_04, Event_Id_NONE, BIOS_WAIT_FOREVER);

//put all variables in printString
strcpy(printString, "Openweather: ");
strcat(printString,tempstr);
strcat(printString,"\nMGM: ");
strcat(printString,tempstr_http);
strcat(printString,"\nTMP006: ");
strcat(printString,tempstr_tmp);
strcat(printString,"\nBMP180: ");
strcat(printString,tempstr_bmp);
sendData2Server(SOCKETTEST_IP, 5011, printString, strlen(printString));

/* Turn off sequence LED */
GPIO_write(Board_LED1, Board_LED_OFF);
```

All values that program got are in different strings. printString is the string that is combined all of the other strings together. After the combine process, it sends data to defined IP on port 5011.

I also create 2 tasks, 5 semaphores and the clock in XGCONF (Event is also added.). The post and pend processes that belongs to events are written as a comment. the design was made this way, but it was pointless to use because of the delay.