Gömülü Sistem Uygulamaları

Final Ödev Sunumu

Öğrenci Adı: Selçuk Altınay

Öğrenci Numarası: 523120003

Ödev Konusu: FreeRTOS ile STM32F429 kullanarak ölçülen sensör verilerinin XML formatta UART protokolü ile bilgisayara aktarmak ve Visual Studio üzerinden C# dilini kullanarak loglama arayüzü oluşturmak.

Sunum Video Linki: https://1drv.ms/v/s!AoJmP15gL0HOtmIpFh2kz270ba3G?e=eFV3gL

İlgili işlem için STM32F429 Discovery geliştirme kartı kullanıldı. FreeRTOS ile 3 adet task kullanılarak ilgili veriler elde edildi. Bu veri, XML formatta UART üzerinden USB-TTL dönüştürücü yardımı ile kullanıcıya aktarıldı. İlgil tasklara ilişkin bilgileri Görsel 1 üzerinde görebilirsiniz.

Task Name	Priority	Stack Size (Wo...	Entry Function	Code Generatio...	Parameter	Allocation	Buffer Name	Control Block N...
defaultTask	osPriorityN...	128	StartDefaultTask	Default	NULL	Dynamic	NULL	NULL
taskAnalog	osPriorityN...	512	taskFuncAnalog	Default	NULL	Dynamic	NULL	NULL
taskBMP	osPriorityN...	512	taskFuncBMP	Default	NULL	Dynamic	NULL	NULL

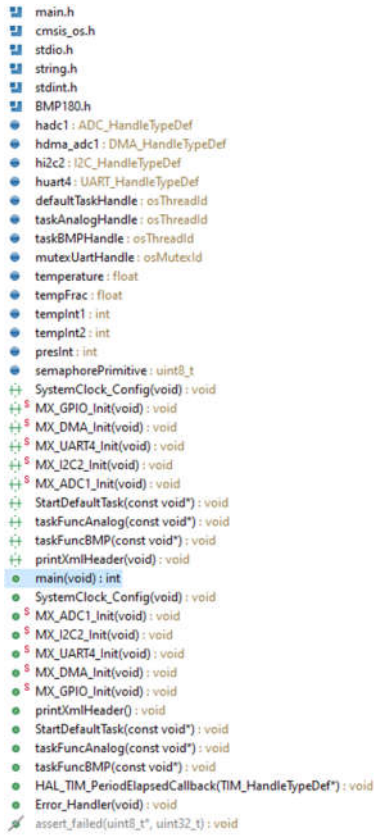
Görsel 1

Uart üzerinden veri transferi yaparken ortak kaynak kullanımı sırasında oluşabilecek girişimi engellemek için bir adet MUTEX kullanılmıştır. Aynı zamanda veri transferi sırasındaki akışı koruyabilmek adına bir adet primitive türden semaphore kullanılmıştır.

İlgili program outline'ı Görsel 2'deki gibidir.

BMP180'den sıcaklık ve basınç verilerini almak için bir kütüphane yazılmıştır. Kütüphaneye ilişkin kodlar, kodlama örneği olması açısından aşağıda belirtilmiştir.

```
/*
 * BMP180.c
 *      Author: selcukaltinay
 */
#include "BMP180.h"



void BMP180_Init()
{
       if(HAL_I2C_IsDeviceReady(&hi2c2, BMP180_WRITE_ADD, 1,
HAL_MAX_DELAY) == HAL_OK){
              AC1=1; // 8 bit msb 8 bit lsb
              AC2=1;
              AC3=1;
              AC4=1;
```



Görsel 2

```
                AC5=1;
                AC6=1;
                B1=1;
                B2=1;
                MB=1;
                MC=1;
                MD=1;
                BMP180_GetCalibrationValue();

        }
}

void BMP180_GetCalibrationValue()
{
        uint8_t counterCalibration = 0;

        uint8_t bmp180Buffer[COUNT_OF_CAL_VAL];
        HAL_I2C_Mem_Read(&hi2c2, BMP180_READ_ADD, 0xAA, 1, bmp180Buffer, COUNT_OF_CAL_VAL,
HAL_MAX_DELAY);
        HAL_Delay(15);
        AC1 = bmp180Buffer[counterCalibration] << 8 | bmp180Buffer[counterCalibration+1];
        counterCalibration+=2;
        AC2 = bmp180Buffer[counterCalibration] << 8 | bmp180Buffer[counterCalibration+1];
        counterCalibration+=2;
        AC3 = bmp180Buffer[counterCalibration] << 8 | bmp180Buffer[counterCalibration+1];
        counterCalibration+=2;
        AC4 = bmp180Buffer[counterCalibration] << 8 | bmp180Buffer[counterCalibration+1];
        counterCalibration+=2;
        AC5 = bmp180Buffer[counterCalibration] << 8 | bmp180Buffer[counterCalibration+1];
        counterCalibration+=2;
        AC6 = bmp180Buffer[counterCalibration] << 8 | bmp180Buffer[counterCalibration+1];
        counterCalibration+=2;
        B1 = bmp180Buffer[counterCalibration] << 8 | bmp180Buffer[counterCalibration+1];
        counterCalibration+=2;
        B2 = bmp180Buffer[counterCalibration] << 8 | bmp180Buffer[counterCalibration+1];
        counterCalibration+=2;
        MB = bmp180Buffer[counterCalibration] << 8 | bmp180Buffer[counterCalibration+1];
        counterCalibration+=2;
        MC = bmp180Buffer[counterCalibration] << 8 | bmp180Buffer[counterCalibration+1];
        counterCalibration+=2;
        MD = bmp180Buffer[counterCalibration] << 8 | bmp180Buffer[counterCalibration+1];
        counterCalibration+=2;

        if(AC1 == 0x00 || AC1 == 0xFFFF)
        {
                BMP180_GetCalibrationValue();
        }
        if(AC2 == 0x00 || AC2 == 0xFFFF)
        {
                BMP180_GetCalibrationValue();
        }
        if(AC3 == 0x00 || AC3 == 0xFFFF)
                {
                        BMP180_GetCalibrationValue();
                }
        if(AC4 == 0x00 || AC4 == 0xFFFF)
                {
                        BMP180_GetCalibrationValue();
                }
        if(AC5 == 0x00 || AC5 == 0xFFFF)
```

```
		{
			BMP180_GetCalibrationValue();
		}
	if(AC6 == 0x00 || AC6 == 0xFFFF)
		{
			BMP180_GetCalibrationValue();
		}
	if(B1 == 0x00 || B1 == 0xFFFF)
		{
			BMP180_GetCalibrationValue();
		}
	if(B2 == 0x00 || B2 == 0xFFFF)
		{
			BMP180_GetCalibrationValue();
		}
	if(MB == 0x00 || MB == 0xFFFF)
		{
			BMP180_GetCalibrationValue();
		}
	if(MC == 0x00 || MC == 0xFFFF)
		{
			BMP180_GetCalibrationValue();
		}
	if(MD == 0x00 || MD == 0xFFFF)
		{
			BMP180_GetCalibrationValue();
		}

}

float BMP180_GetTemperature()
{
	uint8_t wData[1];
	wData[0]=0x2E;

	HAL_I2C_Mem_Write(&hi2c2, BMP180_WRITE_ADD, 0xF4, 1, wData, 1, HAL_MAX_DELAY);
	HAL_Delay(5);
	HAL_I2C_Mem_Read(&hi2c2, BMP180_READ_ADD, 0xF6, 1, dataRead, 2, HAL_MAX_DELAY);
	HAL_Delay(5);

	uncTemp = ((dataRead[0] << 8) | dataRead[1]);


	tX1 = ((uncTemp - AC6)*AC5/32768);
	while(MD==0 && tX1==0)
	{
		BMP180_Init();
	}
	tX2 = (MC*2048)/(tX1+MD);

	tB5 = tX1 + tX2;
	comTemp = (float)(tB5 + 8) / 16;
	return comTemp;
}

uint16_t BMP180_GetPressure()
{
	uint8_t rData[3]={0};
	uint8_t wData[1];
	wData[0] = 0x34 | (0x03 << 6);
	HAL_I2C_Mem_Write(&hi2c2, BMP180_WRITE_ADD, 0xF4, 1, wData, 1, HAL_MAX_DELAY);
```

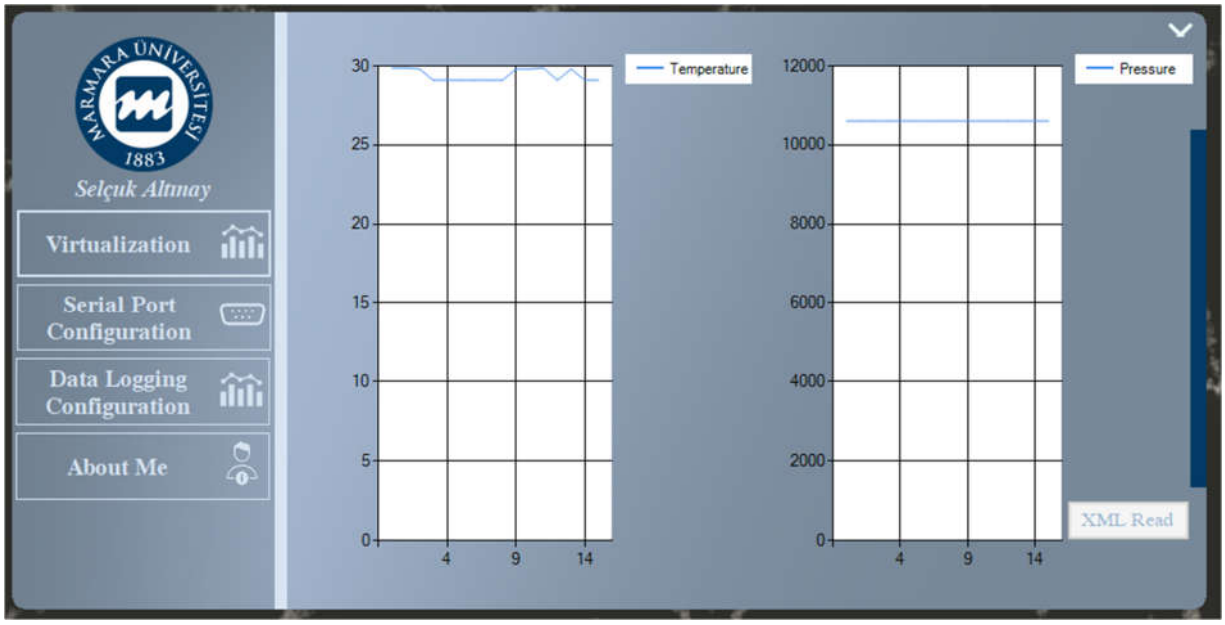
```
        HAL_Delay(26);
        HAL_I2C_Mem_Read(&hi2c2,BMP180_READ_ADD,0xF6,1,rData,3,HAL_MAX_DELAY);
        uncPressure = (rData[0] << 16 | rData[1] << 8 | rData[2]) >> (8 - (uint8_t)(0x03));

        pB6 = tB5 - 4000;
        pX1 = ( B2 * (pB6*pB6/4096))/2048;
        pX2 = AC2 * pB6 / 2048;
        pX3 = pX1 + pX2;
        pB3 = (((AC1*4 + pX3) << 0x03)+2)/4;
        pX1 = AC3 * pB6 / 8192;
        pX2 = (B1 * (pB6*pB6/4096))/65536;
        pX3 = ((pX1 + pX2)+2)/4;
        pB4 = AC4 * (signed long)(pX3+32768)/32768;
        pB7 = ((signed long)uncPressure - pB3) * (50000>>0x03);
        if(pB7 < 0x80000000 && pB4 != 0)
        {
                comPressure = (pB7 * 2) / pB4;
        }
        else{
                comPressure = (pB7/pB4)*2;
        }

        pX1 = (comPressure/256)*(comPressure/256);
        pX1 = (pX1*3038)/65536;
        pX2 = (-7357 * comPressure)/65536;
        comPressure = (comPressure + (pX1 + pX2 +3791)/16)/21.879;//hectopascal
        comPressure = comPressure/1013.2501;//atm
        return comPressure;
}
```

XML datasında, root olarak uartXmlData etiketi kullanılmıştır. Aynı zamanda element olarak ‘Analog’, ‘Temperature’ ve ‘Pressure’ etiketleri kullanışmıştır. Her etiket kendi altında sensor adı, değer ve örnekleme numarası adı verilen değişkenler barındırmaktadır. Örnek bir XML verisi Görsel 3 üzerinden gözlemlenebilmektedir.

```
1  <?xml version="1.0" encoding="utf - 8" ?>
2  <uartXmlData>
3      <context id="Analog">
4          <sensorName>AnalogPin</sensorName>
5          <value>0</value>
6          <sampleNumber>1948</sampleNumber>
7      </context>
8      <context id="Temperature">
9          <sensorName>BMP180</sensorName>
10         <value>29.6187</value>
11         <sampleNumber>1948</sampleNumber>
12     </context>
13     <context id="Pressure">
14         <sensorName>BMP180</sensorName>
15         <value>10656</value>
16         <sampleNumber>1948</sampleNumber>
17     </uartXmlData>
```

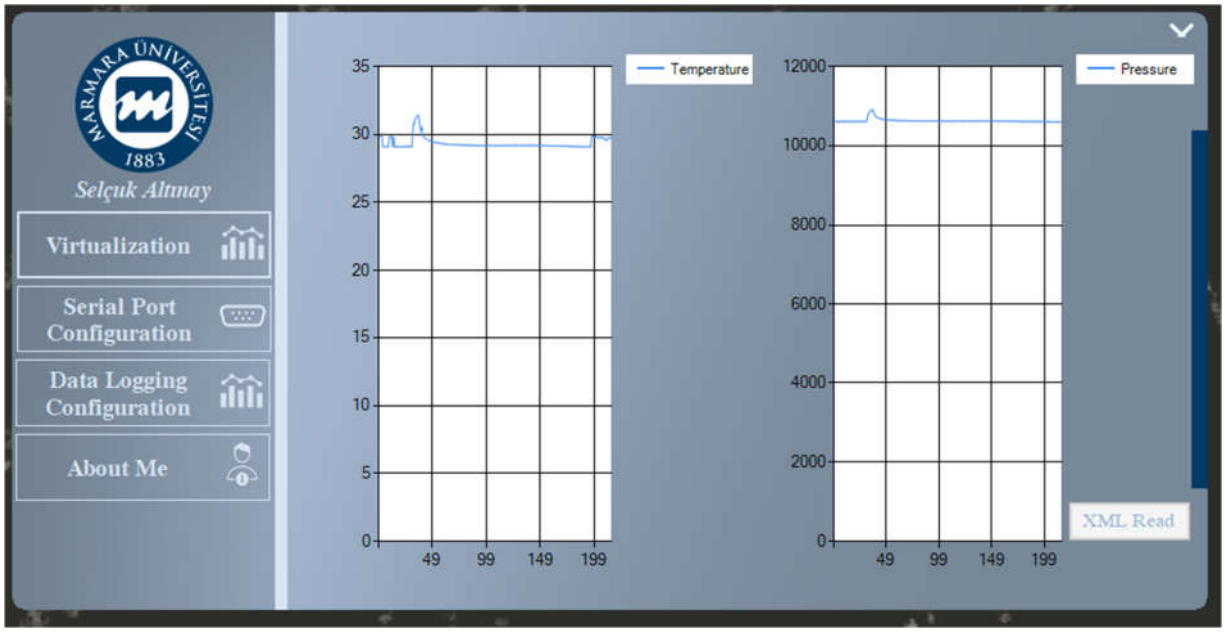
Görsel 3

Program arayüzünde bulunan Virtualization sekmesi üzerinden, verilere ilişkin grafikler Görsel 3’te gösterildiği gibi gözlemlenebilmektedir.



Görsel 4

200 örneklemlik çalışma sonrasındaki grafik aşağıda gözükmektedir. 6 saatlik test işlemi sonucunda program akışında over-flow dahil hiç bir aksama gözlemlenmemiştir. XML Read butonu, önceden loglanmış xml verilerini grafiğe dönüştürmek amacıyla konulmuştur. İmplementasyonu tamamlanmamıştır.



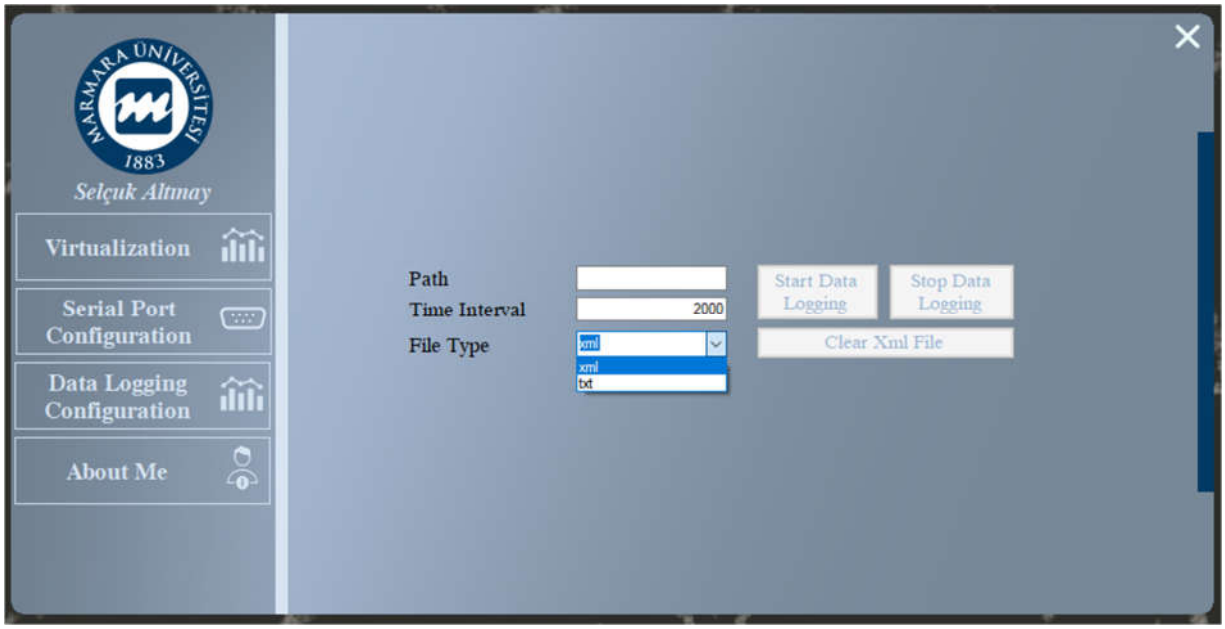
Görsel 5

Serial Port Configuration sekmesi, ilgili konfigürasyonu yapmamıza olanak sağlamaktadır. Buradan seri haberleşmeyi başlatıp durdurabilmekteyiz.

Görsel 6

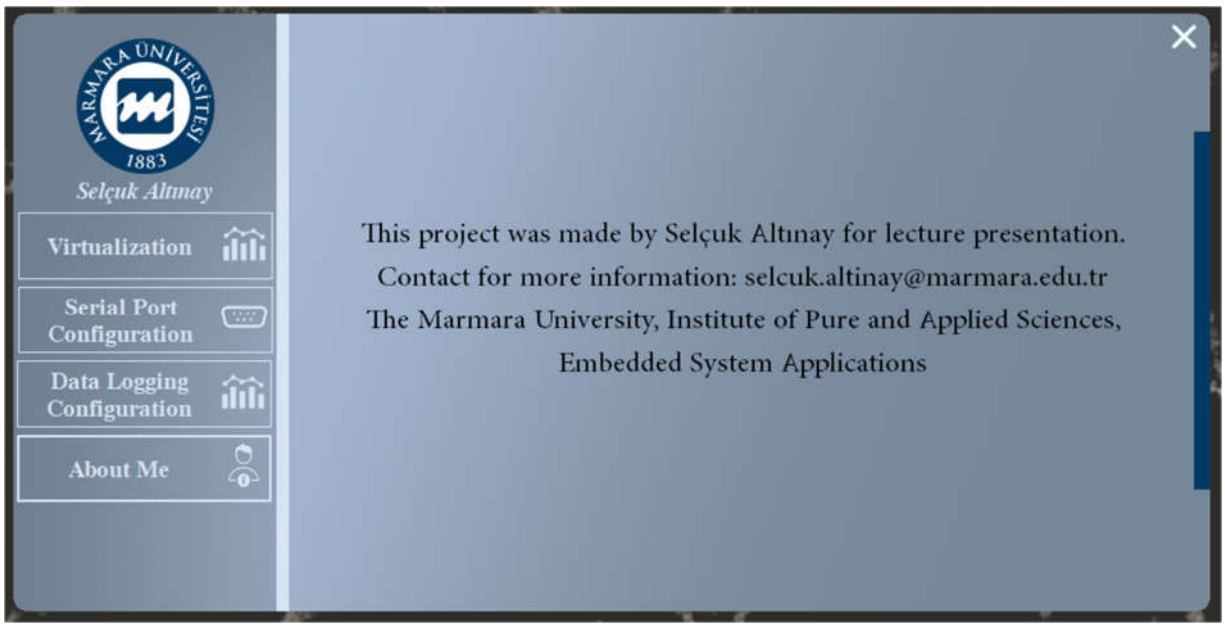
Baud Rate, Parity, Data Bits ve Com Port combobox'ları üzerinden veri seçim işlemi gerçekleştirdikten sonra Connect butonuna bastığımızda seri haberleşme başlatılacaktır. Port görüntülemede oluşabilecek sonunlar için R harfi ile gösterilen Refresh butonu konulmuştur. Disconnect butonu ile haberleşme sonlandırılabilmektedir. Anlık veri akışı sağ tarafta bulunan multiline textbox yardımı ile gözlemlenebilmektedir. İhtiyaç halinde veri transferi gerçekleştirmek adına bir adet buton ve textbox konumlandırılmıştır.

Data Logging Configuration sekmesi üzerinden, verilerimizi loglayabilmekteyiz. Buradan loglama yapmak istediğimiz dosya türünü combobox yardımı ile seçebilmekteyiz. Aynı zamanda time interval ayarı yardımı ile verilerimizi ne kadarlık süreler ile loglamak istediğimizi parametre olarak girebilmekteyiz. Start Data Logging denildiğinde loglama işlemi başlatılır. Grafik çizdirme işleminin başlaması için loglama işleminin başlaması gerekmektedir. Stop Data Logging denildiğinde loglama işlemi durdurulur. Verilerin kaydedilmesi için Stop Data Logging denilmesine gerek yoktur. File Close işlemi her yazdırma işleminin ardından veri güvenliği için yapılmaktadır.



Görsel 7

About Me sekmesi üzerinden, iletişim için gerekli mail adresimi görebilmektesiniz.



Görsel 8

Program çerçevesi geliştirmeye açık olması açısından geniş bırakılmıştır.