

EGE ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

GÖMÜLÜ VE GERÇEK ZAMANLI SİSTEMLER PROJE

RAPORU

Ahmet GÜL 05110000002

Alper DOĞAN 05120000277

Büşra YUMUTURUĞ 05110000996

İçindekiler

1) Projenin Amacı……………………………………………………

2) Proje Ekipmanları Tanıtımı……………………………………….

2.1. Projede Kullaılan Donanım……………………………….

2.2. Projede Kullanılan Yazılım……………………………...

3) Projenin Gerçekleştirimi……………….…………………………..

4) Donanım Bağlantı Şeması …………….………………………......

5) Ekran Görüntüleri………………………………………………….

6) Kaynak Kodlar…………………………………………………….

7) Karşılaşılan Zorluklar……………………………………………...

8) Program Kısıtlamaları……………………………………………...

1. **Projenin amacı**

İlk olarak kullanılan donanım ve yazılım ekipmanlarıyla belli bir sıcaklık değerini ölçümünü yapmaktır. Yaptığımız bu ölçüm sonucunda elde ettiğimiz bu değeri kullandığımız donanım olan LCD üzerinde görüntülenmesini sağlamaktır.

Ayrıca ek özellik olarak görüntülediğimiz bu ölçüm değerlerini yazılımsal ve işlevsel olarak daha bir kullanılabilir hale getirebilmek istenmiş ve bunun için sıcaklık değerlerinin anlık olarak grafik arayüzü görüntülenmesini sağlamak ve elde edilen değerlerin kayıt altına alınabilmesi için bir yerde saklanması amaçlanmıştır. Ayrıca ölçülen sıcaklığın değerine göre Stellaris üzerindeki ledlerin rengide değişmektedir.

Tüm bu çalışmalar sonucunda elde ettiğimiz sıcaklık verilerinin değerlerini ve grafiklerini tıbbi olarak kullanılmasını amaçladık. Bu sebeble projemiz üzerine eklemeler ve güncellemeler yaptık.

1. **Proje ekipmanlarının tanıtımı**
2. **Donanım tanıtımı**

*Stellaris Launchpad with LMF4120H5QR* : Projede kullanılan mikrodenetleyicidir. Isı sensörü ve LCD için gerekli olan kodlar bu cihaz için yazılmıştır.

*16x2 led ekran* : Projede kullanılan LCD ekrandır. 2 satır ve her satırda 16 kolon barındırmaktadır. Sıcaklık değerleri buraya yazdırılmıştır. 16 pini bulunmaktadır. 8 pini veri transferi için kullanılabilmektedir. Biz projemizde son 4 pinini veri transferi için kullandık.

*Max31855 termocouple amplifier* : Sıcaklık ölçümü bu cihaz ile yapılmıştır. 2 tanesi topraklama ve güç için kullanılmak üzere 6 pine sahiptir.

*3mm Kırmızı LED* : Sıcaklığın okunup okunamadığını anlamamızı sağlayan LEDdir. Eğer bu led yanıyorsa programın sıcaklık değerini okuduğunu anlamaktayız.

*10K Pot*: 16x2’lik LCD’ye bağlanmıştır. LCD’nin konsantrasyon ayarında kullanılmıştır.

1. **Yazılım tanıtımı**
2. **Energia**: LM4F cihazı için yazılan kodlar bu ortamda yazılmıştır.

SPI: Veri transferi için kullanılan kütüphanedir.LCD’ye veri yazdırmak için bu LiquidCrystal kütüphanesi tarafından bu kütüphane kullanılmıştır. Bu yüzden programımıza eklenmiştir.

LiquidCrystal: 16x2’lik LCD’ye okunan sıcaklık değerleri bu kütüphanedeki methodlar aracılığı ile yazılmıştır.

MAX31855: Sıcaklık sensörünü kullanmak için gerekli olan kütüphanedir. ReadJunction ve ReadThermoCouple methodları sıcaklık okumak için kullanılmıştır.

1. **Visual Studio 2010 :** C# programlama dili sıcaklık grafiği ve sıcaklık değerlerini kaydeden program bu ortamda yazılmıştır.

Microsoft.Office.Interop.Excel: Alınan sıcaklık değerlerini kaydetmek için çağrılan referanstır.Buradaki methodlar aracılığı ile alınan sıcaklık değerleri excel dosyasına kaydedilmiştir.

1. **Projenin Gerçekleştirimi**

**Stellaris LM4F cihazı için program:**

Projede öncelikli olarak LiquidCrystal ve MAX31855 kütüphanelerinden oluşturulan nesneler ilklenmektedir. MAX31855’e sırasıyla Stellaris DO, CS ve CLK bacaklarından çıkan pinlerin gönderildiği için bu pin paramtere olarak gönderilmiştir. LiquidCrystal için ise donanımda bağlandığı şekliyle sırasıyla MISO, MOSI , PB\_3, PB\_2 , PB\_6 ve PB\_7 pinleri parametre olarak gönderilmiştir. Daha sonra Serial.begin ile seri haberleşme başlatılmıştır.

Programımız loop döngüsü içerisinde her seferinde Stellaris LM4F cihazı üzerindeki 2 numaralı butona basılıp basılmadığına bakmaktadır.

* Eğer basılmadıysa normal olarak ReadJunction() methodu ile sıcaklığı okuyarak okunan değeri Seri Port üzerinden yazdırmaktadır.
* Eğer 2 numaralı butona basıldıysa ReadThermoCouple() adlı method ile sıcaklığı okuyarak okunan değeri Seri Port üzerinden yazdırmaktadır.

Program her iki durum içinde butona ikinci defa basılıncaya kadar aynı method ile okuma yapmaktadır. Başka bir deyiş ile 2 numaralı butona her basıldığında öteki method ile okumaya başlamaktadır.

Daha sonra okunan bu sıcaklık değeri kullanıcıya gösterilmek için LCDye print() methodu ile yazdırılmaktadır. Burada yazdırma işleminden önce kullanıcıya sıcaklık hakkında daha iyi bir çıkarsama yapabilmesini sağlamak amacıyla ile sıcaklık değerinin hangi sıcaklık aralığında olduğu kontrol edilmektedir. Eğer sıcaklık değeri;

* 10 Celsiusdan küçükse ya da eşitse Stellaris LM4F cihazı üzerinde cyan renkte ışık yanmaktadır.
* 11 Celsius ile 20 Celsius arasında ise Stellaris LM4F cihazı üzerinde mavi renkte ışık yanmaktadır.
* 21 Celsius ile 30 Celsius arasında ise Stellaris LM4F cihazı üzerinde magenta renkte ışık yanmaktadır.
* 30 Celsius ile 40 Celsius arasında ise Stellaris LM4F cihazı üzerinde kırmızı renkte ışık yanmaktadır.

Bu işlemlerin sonunda sıcaklığın doğru bir şekilde okunduğunu anlamak için kullanıcının daha rahat görebileceği devreye sonradan bağlanmış bir Kırmızı LED de yanmaktadır.

**Microsoft Visual Studio 2010 C# programı:**

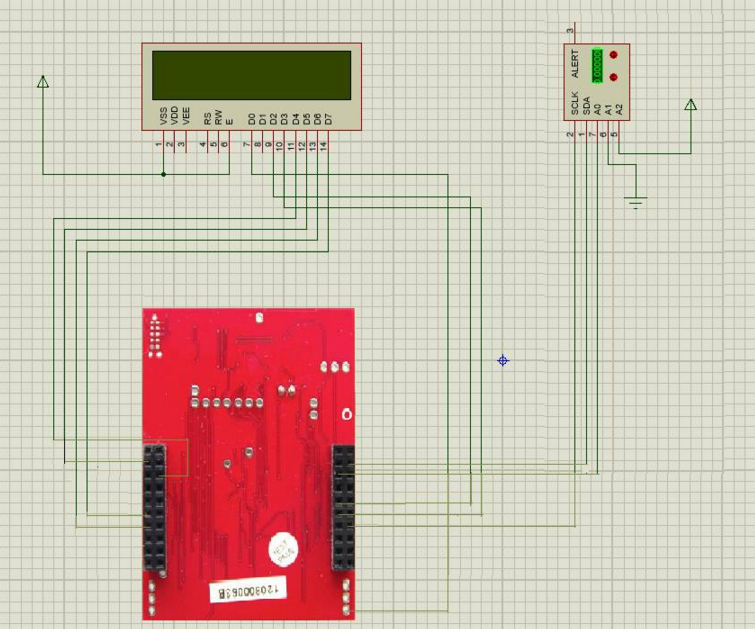
Öncelikle IO port namespace’inden SerialPort classı ile Energia LM4F’den gönderilen bilgileri okuyabilmek için nesne oluşturuluyor. Daha sonra alınan bilgilerin grafiğini çizdirmek için toolbardaki chart objesi üzerine çizim yapılabilmesi için ilklenmeleri ve ayarları yapılmıştır.

Daha sonra start butonu eventi için yeni bir iş parçacığı(Thread) oluşturuluyor. Bunun yapılmasının sebebi grafik GUI arayüzünün iş parçacığı ile çalışacağı için ve bu sürede sıcaklık değerlerini sürekli okumak için sonsuz döngüde kalacak olan programı durdumak için farklı bir threade (iş parçacığına) ihtiyaç duyulmuştur.

Bu işlem sonunda drawGraph() methodu içinde Excel dosyasına yazdırmak için xlApp adlı nesne tanımlanmaktadır.Bu işlemden sonra try – catch bloğu içinde kod içine yazılan yoldaki excel dosyası açılıyor.

* Eğer burada bir dosya yoksa program exception fırlatacak ve program catch bloğuna girecektir. Burada excel dosyası oluşturuluyor ve program excel dosyasının 1.satır 1.kolonunu Zaman , 1.satır 2.kolonuna Sıcaklık Celcius yazarak sütunlara ad veriyor.Daha sonra kullanıcıya excel dosyasını oluşturduğunu belirten bir mesaj fırlatıyor.
* Eğer daha önceden bir excel dosyası oluşturulduysa , dosya güzel bir şekilde açılıyor. Burada dosyanın son yazılan satırı ve sütunu bulunuyor.Daha sonra Portdan sıcaklığı okumak için oluşturulan nesneden ReadLİne() methodu ile porttan gönderilen değerler okunuyor ve sıcaklık değeri string işlemleri ile double hale getirilip alınıyor. Daha sonra GUI threadi ile değer alıp yazan thread birbirlerini kilitlemesin diye bizim oluşturduğumuz thread durduruluyor. Thread durdurulunca, sıcaklık değeri chart objesine ekleniyor ve excel dosyasında en son yazılan satırın 1.kolonuna yıl/ay/gün saat: dakika:saniye formatında yazdırılıyor, 2.kolonuna sıcaklık değeri yazdırılıyor.Bu işlemler stop butonuna basılıncaya kadar devam ettiriliyor.Stop butonuna basınca dosyaya okunan değerlerin kayıt edilip edilmemesi gerektiği soruluyor ve kaydet tuşuna basınca excel dosyasına kaydediliyor.

1. **Donanım Bağlantı Şeması**



Aşağıda cihazlar üzerinde hangi bacakların nereye bağlandığı gösterilmektedir.

LCD STELLARIS

K GND

A VBUS

1 GND

2 VBUS

3 POT ORTA BACAK

4 PD\_2 / MISO(3)

5 GND

6 PD\_3 / MISO(4)

7 GND

8 GND

9 GND

10 GND

11 PB\_3

12 PB\_2

13 PB\_6

14 PB\_7

TERMOSTAT STELLARIS

CLK A10

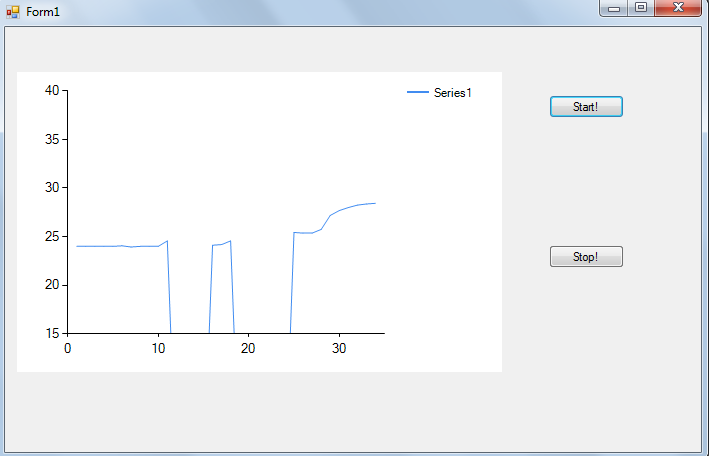
CS A11

DO A7

GND GND

3VO 3.3V

1. **Ekran görüntüleri**



1. **Kaynak Kodlar**

**Energia**

#include <SPI.h>

#include <LiquidCrystal.h>

#include <MAX31855.h>

// sicaklik okumak icin pinler tanimlaniyor

const unsigned char thermocoupleSO = A7;

const unsigned char thermocoupleCS = A11;

const unsigned char thermocoupleCLK = A10;

MAX31855 MAX31855(thermocoupleSO, thermocoupleCS, thermocoupleCLK);

//lcd icin pinleri tanimladik

LiquidCrystal lcd(PD\_2,PD\_3,PB\_3,PB\_2,PB\_6,PB\_7);

const int buttonPin = PUSH2; // buton tanimlaniyor

int buttonState = 0; // buton durumu icin deger tanimlaniyor

int stateControl = 0; // false durum ile ilkledik

//ledler tanimlaniyor

#define RED RED\_LED

#define GREEN GREEN\_LED

#define BLUE BLUE\_LED

void setup()

{

// put your setup code here, to run once:

Serial.begin(9600);

lcd.begin(16,1);

pinMode(buttonPin, INPUT\_PULLUP);

pinMode(RED, OUTPUT);

pinMode(GREEN, OUTPUT);

pinMode(BLUE, OUTPUT);

pinMode(PA\_4,OUTPUT); /// calistigini gosteren pinin ilklenmesi

}

void loop()

{

buttonState = digitalRead(buttonPin);

double heating;

if (buttonState == HIGH){

if(stateControl == 0){

heating = MAX31855.readJunction(CELSIUS);

Serial.print("Junction temperature: ");

Serial.print(heating);

Serial.println(" Degree Celsius");

}

else if(stateControl == 1)

{

heating = MAX31855.readThermocouple(CELSIUS);

Serial.print("Thermocouple temperature: ");

Serial.print(heating);

Serial.println(" Degree Celsius");

}

}

else if(buttonState == LOW){

switch(stateControl)

{

case 0: stateControl = 1; delay(500); break;

case 1: stateControl = 0; delay(500); break;

}

}

if(heating)

{

delay(500);

lcd.setCursor(0,0);

//LCDye burada yazdiriyor

lcd.print("temp");

lcd.print(heating);

if(heating <= 10 )

{

//cyan

digitalWrite(RED,LOW);

digitalWrite(BLUE, HIGH);

digitalWrite(GREEN,HIGH);

}

else if(heating>10 && heating<=20)

{ //blue

digitalWrite(RED,LOW);

digitalWrite(GREEN,LOW);

digitalWrite(BLUE, HIGH);

}

else if( heating>20 && heating<=30)

{

//magenta

digitalWrite(GREEN,LOW);

digitalWrite(RED,HIGH);

digitalWrite(BLUE,HIGH);

}

else if(heating>30 && heating<=40)

{

//RED

digitalWrite(GREEN,LOW);

digitalWrite(RED,HIGH);

digitalWrite(BLUE,LOW);

}

//eger okuyamiyorsa extra bir kirmizi isik yanar ve bize calisamadigini anlatir.

digitalWrite(PA\_4,HIGH);

delay(1000);

}

else

{

//eger sicakligi okuyamazsa extra led yanmaz!

digitalWrite(PA\_4,LOW);

}

}

**Visual Studio**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.IO.Ports;

using System.Threading;

using Excel = Microsoft.Office.Interop.Excel; //excel referans kütüphaneleri ekleniyor!

namespace ReadFromSerial\_DrawGraphic

{

public partial class Form1 : Form

{

public SerialPort myport;

Thread thread1;

private string excelFilePath = string.Empty;

Excel.Workbook xlWorkBook;

Excel.Worksheet xlWorkSheet;

object misValue;

DateTime theTime; // zamanı excel dosyasına yazdırmak icin olusturuldu

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

myport = new SerialPort(); // SerialPort olusturuluyor

myport.BaudRate= 9600; // SerialPort hızı belirleniyor

myport.PortName = "COM11"; // Kullanılacak Port belirleniyor

myport.Open(); // port aciliyor

chart1.Series["Series1"].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.FastLine; // cizdirilecek grafik türü

chart1.ChartAreas[0].AxisX.MajorGrid.Enabled = false; // ekseninde chart taki arka plani temizliyoruz

chart1.ChartAreas[0].AxisY.MajorGrid.Enabled = false;

chart1.ChartAreas[0].AxisY.Maximum = 40;

chart1.ChartAreas[0].AxisY.Minimum = 15;

misValue = System.Reflection.Missing.Value;

}

private void button1\_Click(object sender, EventArgs e)

{

thread1 = new Thread(new ThreadStart(drawGraph));

thread1.Start();

}

private void drawGraph()

{

Excel.Application xlApp = new Microsoft.Office.Interop.Excel.Application();

try

{

xlWorkBook = xlApp.Workbooks.Open("C:\\Users\\DELL\\Desktop\\ReadFromSerial\_DrawGraphic\\csharp-Excel.xls", 0, false, 5, "", "", true, Microsoft.Office.Interop.Excel.XlPlatform.xlWindows, "\t", false, false, 0, true, 1, 0);

xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get\_Item(1); //Get all the sheets in the workbook

while (thread1.IsAlive)

{

//son satır bulunuyor excel dosyasındaki

Excel.Range last = xlWorkSheet.Cells.SpecialCells(Excel.XlCellType.xlCellTypeLastCell, Type.Missing);

Excel.Range range = xlWorkSheet.get\_Range("A1", last);

int lastUsedRow = last.Row;

int lastUsedColumn = last.Column;

string ReceiveData = myport.ReadLine(); // comdan degeri okuyuruz

// alınan degerdeki stringleri temizleyerek sadece double değeri yakalıyor

string[] HeatingData = ReceiveData.Split(':');

string[] HeatingData2 = HeatingData[1].Split('D');

var result = HeatingData2[0];

double heating = Convert.ToDouble(result);

theTime = DateTime.Now; // anlik olarak zamani ogreniyoruz!

string zaman = theTime.ToString("yyyy/MM/dd HH:mm:ss");

Thread.Sleep(1000); // ilk threadi anlik olarak durduruyor ve Invoke ile GUI threadini ulasip cizdiriyor!

this.Invoke((MethodInvoker)delegate

{

chart1.Series["Series1"].Points.AddY(result);

// excel dosyasındaki son yazılan satırdan bir sonraki satıra sıcaklığı yazdırıyor

xlWorkSheet.Cells[lastUsedRow+1, 2] = (heating / 100);

xlWorkSheet.Cells[lastUsedRow + 1, 1] = zaman;

});

}

}

catch

{

// MessageBox.Show("Dosya bulunamadı");

xlWorkBook = xlApp.Workbooks.Add(misValue);

xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get\_Item(1);

xlWorkSheet.Cells[1, 1] = "Zaman";

xlWorkSheet.Cells[1, 2] = "Sıcaklık Celcius";

xlWorkBook.SaveAs("C:\\Users\\DELL\\Desktop\\ReadFromSerial\_DrawGraphic\\csharp-Excel.xls", Excel.XlFileFormat.xlWorkbookNormal, misValue, misValue, misValue, misValue, Excel.XlSaveAsAccessMode.xlExclusive, misValue, misValue, misValue, misValue, misValue);

xlWorkBook.Close(true, misValue, misValue);

xlApp.Quit();

releaseObject(xlWorkSheet);

releaseObject(xlWorkBook);

releaseObject(xlApp);

MessageBox.Show("Dosya oluşturuldu , proje klasörünüzde bulunmaktadır");

}

}

private void releaseObject(object obj) //gecmis datayı , objeleri temizlemek icin

{

try

{

System.Runtime.InteropServices.Marshal.ReleaseComObject(obj);

obj = null;

}

catch (Exception ex)

{

obj = null;

MessageBox.Show("Exception Occured while releasing object " + ex.ToString());

}

finally

{

GC.Collect();

}

}

private void button2\_Click(object sender, EventArgs e)

{

// cizdirmeyi durdurmak icin ilk threadi sonlandiriyor!

thread1.Abort();

xlWorkBook.Close(misValue, misValue, misValue);

//xlApp.Quit();

releaseObject(xlWorkSheet);

releaseObject(xlWorkBook);

//releaseObject(xlApp);

}

}

}

1. **Karşılaşılan Zorluklar**

Proje kapsamında kullanılacak donanımlar için gerekli kütüphanelerin çoğunun Stellaris LM4f cihazı için yazılmamış olmasından dolayı doğru ve uyumlu kütüphaneleri bulma işlemi uzun zaman almıştır.

Projede kullanılan 16x2 LCD’nin pinlerinin sıralaması diğer LCDlerden farklıdır ve bu versiyonunun datasheeti bulunamamıştır. Bu versiyona en yakın datasheete bakılarak pinlerin üzerinde yazan numaralara göre bağlantılar yapılmıştır.

Energia ortamında geliştirilen kodlar HEX hale getirilememiştir. Bu özellik yüzünden LCD’ye yazı yazdırmada yaşanan sorunlar kütüphane kaynaklı mı bağlantı kaynaklı mı yoksa donanım kaynaklı mı olduğunu anlamak için Proteus ortamına simülasyon yaptırılamamıştır.

1. **Program Kısıtlamaları**

* C# programlama dili ile yazılan kodda portun com değeri kod içinde girilmiştir.Eğer farklı bir bilgisayarda kullanılacaksa bu değerin el ile değiştirilmesi gerekmektedir.
* C# programlama dili ile yazılan kodun bir görevi de Excel dosyasına zaman ve sıcaklık değerlerini yazdırmaktı. Bu kısımda Excel dosyasının açılacağı ya da oluşturulacağı path el ile yazılmıştır. Programın farklı bilgisayarlarda istenilen lokasyona dosyayı açabilmesi için kod içinde değişiklik yapılması gerekebilir.