

```

/*****
*   LCD Connection:
*
*   CS                PA.7 chip select
*   C/D(RS)          PA.6 command/data mode
*   WR               PA.5 write strobe
*   RD               PA.4 read strobe
*   RST              PF.4 reset
*   BKLT             PF.3 backlight
*   DATA            PB data line 8-15 on pins B.0-B.7
*
*   XP+              PE.2 touch X+      AIN1
*   YP+              PE.3 touch Y+      AIN0
*   XN-              PA.3 touch X-
*   YN-              PA.2 touch Y-
*
*
*   SHT75 - Temperature and Humidity sensor
*   SDA              PD.7
*   SCL              PD.6
*
*   BMP085 - Pressure and Altitude sensor
*   SDA              PE.5
*   SCL              PE.4
*
*   Anemometer - Wind speed
*   Resistor divider 5k          PD.2
*
*   Wind direction
*   Resistor divider 10k        PE1          AIN2
*
*   if(wvalue <= 552) strcpy(pole, "W_");
*       else if(wvalue <= 940) strcpy(pole, "NW");
*       else if(wvalue <= 1472) strcpy(pole, "N_");
*       else if(wvalue <= 2127) strcpy(pole, "SW");
*       else if(wvalue <= 2748) strcpy(pole, "NE");
*       else if(wvalue <= 3232) strcpy(pole, "S_");
*       else if(wvalue <= 3535) strcpy(pole, "SE");
*       else if(wvalue <= 4096) strcpy(pole, "E_");
*       else strcpy(pole, "W_");
*
*   Rain Gauge          PD.3
*
*   LED display          PC7          Tx3 serial
*
*   Light Sensor        PE.0          AIN3
*
*   Nazvoslovi
*   -----
*   tlak                - Pressure
*   srazky              - WaterLevel
*   rosny bod           - Dewpoint
*   pocitova teplota    - ApparentTemperature
*   vyska                - AltitudeMy

```

```
* intenzita osvetleni   - Light
* vitr                  - Wind      (smer vetru - wind direction, rychlost
vetru - wind speed)
* datum a cas           - Date
* ikona pocasi         - Image
* teplota uvnitr       - TemperatureIn
* teplota venku        - Temperature
* vlhkost              - Humidity
*
*
```

```
*****
**/
```

```
#include "inc/hw_gpio.h"
#include "inc/hw_ints.h"
#include "inc/hw_memmap.h"
#include "inc/hw_nvic.h"
#include "inc/hw_ssi.h"
#include "inc/hw_sysctl.h"
#include "inc/hw_types.h"
```

```
#include "driverlib/debug.h"
#include "driverlib/fpu.h"
#include "driverlib/flash.h"
#include "driverlib/gpio.h"
#include "driverlib/i2c.h"
#include "driverlib/interrupt.h"
#include "driverlib/pin_map.h"
#include "driverlib/rom.h"
#include "driverlib/rom_map.h"
#include "driverlib/ssi.h"
#include "driverlib/sysctl.h"
#include "driverlib/systick.h"
#include "driverlib/timer.h"
#include "driverlib/uart.h"
#include "driverlib/udma.h"
#include "driverlib/adc.h"
```

```
#include "glib/glib.h"
#include "glib/widget.h"
#include "glib/canvas.h"
#include "glib/checkbox.h"
#include "glib/container.h"
#include "glib/pushbutton.h"
#include "glib/radiobutton.h"
#include "glib/slider.h"
```

```
#include "utils/uartstdio.h"
#include "utils/ustdlib.h"
#include "drivers/Adafruit320x240x16TouchTFT_ILI9325.h"
// #include "drivers/sound.h"
```

```

#include "drivers/touch.h"
#include "images.h"
#include "sht75my.h"
#include "math.h"
#include "I2C_Stellaris_API.h"
#include "bmp085my.h"
#include <string.h>
#include <stdlib.h>

#define BMP085_ADDR      0x77
#define PI 3.1415

// odkaz na pole s obrazkem
extern const unsigned char g_pucImage[];

tContext sContext;
tRectangle sRect;
void ClrScreen(void);

/*****
// The error routine that is called if the driver library encounters an error.
*****/
#ifdef DEBUG
void
__error__(char *pcFilename, unsigned long ulLine)
{
}
#endif

/*****
//
// The DMA control structure table.
//
*****/
#ifdef ewarm
#pragma data_alignment=1024
tDMAControlTable sDMAControlTable[64];
#elif defined(ccs)
#pragma DATA_ALIGN(sDMAControlTable, 1024)
tDMAControlTable sDMAControlTable[64];
#else
tDMAControlTable sDMAControlTable[64] __attribute__((aligned(1024)));
#endif

// -----
// -----
PROMENNE

unsigned int humidity = 0;
unsigned int temperature = 0;

```

```

float temperature_f = 0.0;
float temperature_f_max = 0.0;
float temperature_f_min = 100.0;
int temperature_i = 0;
float temperature_app = 0.0;
float humidity_f = 0.0;
float humidity_real = 0.0;
float humidity_real_max = 0.0;
float humidity_real_min = 100.0;
float wvp_exp = 0.0;
float wvp = 0.0;
float dewpoint = 0.0;
float dewa = 0.0;
float dewb = 0.0;
char string_t[5]; // text temperature
char string_t_max[5]; // text temperature_max
char string_t_min[5]; // text temperature_min
char string_t_a[5]; //text temperature apparent
char string_h[5]; // text humidity
char string_h_i[2]; // text humidity int
char string_h_max[5]; // text humidity_i_max
char string_h_min[5]; // text humidity_i_min
char string_d[5]; // text dew point
char string_tl[5]; // text tlak
char string_v[5]; // text vyska
char string_w[5]; // text water level in ml/h
char string_w_max[5]; // text maximum water level in ml/h
char string_time[8] = "00:00:00"; // text time
char string_time_hour[2] = "00"; // text time hour
unsigned int time_hour = 0; // time + 2 hour for CZ time

unsigned char poleTempSend[] = "1234"; //pole pro odeslani na LED displej

int sendChar = 0;

unsigned int msTimer = 0;

volatile unsigned int pocet = 0;
volatile unsigned int pocet_preklopeni = 0; //Preklopeni pri 0,2794 mm -> 294
mL na m2
volatile unsigned int pocet_preklopeni_hour = 0;
volatile unsigned int overeni = 10;
volatile unsigned int overeni2 = 10;
float otacky_f = 0.0;
float otacky_f_max = 0.0;
char string_r[5];
char string_r_kopie[5];
char string_r_max[5];
char *beafourt_table[] = {"bezvetri", "vanek", "vetrik", "slaby vitr", "mirny
vitr", "cerstvy vitr", "silny vitr", "mirny vichr", "cerstvy vichr", "silny vichr", "plny
vichr", "vichrice", "orkan"}; //Beafourtova stupnice sily vetru
// 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 - 17
// < 0.5, ~ 1.25, ~ 3, ~ 5, ~ 7, ~ 9.5, ~ 12, ~ 14.5, ~ 17.5, ~ 21, ~ 24.5, ~ 29, >
30

```

```

int poleTemp[300];
int poleTempIndex = 0;

int beafourt = 0;

long pressure = 0;
double altitude_my = 0;

int circle_x = 279, circle_y = 40, circle_rad = 35;

unsigned long ulADC0_Value[4];           //hodnota z AIN0
unsigned int wvalue = 0;
char pole[] = "00";
double smer = 0;

unsigned int vteriny = 0;
float water_level = 0.0;
float water_level_max = 0.0;

int obnoveni_displeje = 0;           //obnoveni displeje s textem
int kalibrace = 1;                   //kalibrace displeje
int bmp085_enable = 1;               //BMP085 enable

unsigned long polozkaMenu = 0;

char znakGps;                         // GPS
char poleGpsRx[12] = "-----123456";
unsigned int a = 0;

// -----
// ----- Deklarace funkci -----

void zakladniGrafika();

void CallPressure();
void CallWaterLevel();
void CallDewpoint();
void CallApparentTemperature();
void CallAltitudeMy();
void CallLight();
void CallWind();
void CallDate();
void CallImage();
void CallTemperatureIn();
void CallTemperature();
void CallHumidity();

void BackButtonPressure();
void BackButtonWaterLevel();
void BackButtonDewpoint();
void BackButtonApparentTemperature();
void BackButtonAltitudeMy();

```

```

void BackButtonLight();
void BackButtonWind();
void BackButtonDate();
void BackButtonImage();
void BackButtonTemperatureIn();
void BackButtonTemperature();
void BackButtonHumidity();

extern tCanvasWidget canMain; // hlavni Canvas
extern tCanvasWidget canButtons; // Canvas tlacitka

extern tCanvasWidget canPressure; // Canvas Pressure
extern tCanvasWidget canWaterLevel; // Canvas WaterLevel
extern tCanvasWidget canDewpoint; // Canvas Dewpoint
extern tCanvasWidget canApparentTemperature; // Canvas ApparentTemperature
extern tCanvasWidget canAltitudeMy; // Canvas AltitudeMy
extern tCanvasWidget canLight; // Canvas Light
extern tCanvasWidget canWind; // Canvas Wind
extern tCanvasWidget canDate; // Canvas Date
extern tCanvasWidget canImage; // Canvas Image
extern tCanvasWidget canTemperatureIn; // Canvas TemperatureIn
extern tCanvasWidget canTemperature; // Canvas Temperature
extern tCanvasWidget canHumidity; // Canvas Humidity

extern tPushButtonWidget widBackBtnPressure;
extern tPushButtonWidget widBackBtnWaterLevel;
extern tPushButtonWidget widBackBtnDewpoint;
extern tPushButtonWidget widBackBtnApparentTemperature;
extern tPushButtonWidget widBackBtnAltitudeMy;
extern tPushButtonWidget widBackBtnLight;
extern tPushButtonWidget widBackBtnWind;
extern tPushButtonWidget widBackBtnDate;
extern tPushButtonWidget widBackBtnImage;
extern tPushButtonWidget widBackBtnTemperatureIn;
extern tPushButtonWidget widBackBtnTemperature;
extern tPushButtonWidget widBackBtnHumidity;

extern tPushButtonWidget widPushBtnPressure;
extern tPushButtonWidget widPushBtnWaterLevel;
extern tPushButtonWidget widPushBtnDewpoint;
extern tPushButtonWidget widPushBtnApparentTemperature;
extern tPushButtonWidget widPushBtnAltitudeMy;
extern tPushButtonWidget widPushBtnLight;
extern tPushButtonWidget widPushBtnWind;
extern tPushButtonWidget widPushBtnDate;
extern tPushButtonWidget widPushBtnImage;
extern tPushButtonWidget widPushBtnTemperatureIn;
extern tPushButtonWidget widPushBtnTemperature;
extern tPushButtonWidget widPushBtnHumidity;

```

```

// -----
-----
--- CANVAS

```





```
Canvas(canButtons, &canMain, 0, &widPushBtnPressure, &g_sAdafruit320x240x16_ILI9325,
0, 0, 320, 240, CANVAS_STYLE_FILL, ClrBlack, 0, 0, 0, 0, 0, 0);
```

```
RectangularButton(widPushBtnPressure, &canButtons, &widPushBtnWaterLevel, 0,
&g_sAdafruit320x240x16_ILI9325, 1, 2, 59, 78, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
CallPressure);
RectangularButton(widPushBtnWaterLevel, &canButtons, &widPushBtnDewpoint, 0,
&g_sAdafruit320x240x16_ILI9325, 1, 81, 59, 79, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
CallWaterLevel);
RectangularButton(widPushBtnDewpoint, &canButtons, &widPushBtnApparentTemperature, 0,
&g_sAdafruit320x240x16_ILI9325, 1, 161, 119, 39, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
CallDewpoint);
RectangularButton(widPushBtnApparentTemperature, &canButtons, &widPushBtnAltitudeMy,
0, &g_sAdafruit320x240x16_ILI9325, 1, 201, 119, 38, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
CallApparentTemperature);
RectangularButton(widPushBtnAltitudeMy, &canButtons, &widPushBtnLight, 0,
&g_sAdafruit320x240x16_ILI9325, 121, 161, 118, 39, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
CallAltitudeMy);
RectangularButton(widPushBtnLight, &canButtons, &widPushBtnWind, 0,
&g_sAdafruit320x240x16_ILI9325, 121, 201, 118, 38, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
CallLight);
RectangularButton(widPushBtnWind, &canButtons, &widPushBtnDate, 0,
&g_sAdafruit320x240x16_ILI9325, 240, 2, 79, 138, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
CallWind);
RectangularButton(widPushBtnDate, &canButtons, &widPushBtnImage, 0,
&g_sAdafruit320x240x16_ILI9325, 240, 141, 79, 98, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
CallDate);
RectangularButton(widPushBtnImage, &canButtons, &widPushBtnTemperatureIn, 0,
&g_sAdafruit320x240x16_ILI9325, 181, 2, 58, 78, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
CallImage);
RectangularButton(widPushBtnTemperatureIn, &canButtons, &widPushBtnTemperature, 0,
&g_sAdafruit320x240x16_ILI9325, 61, 2, 119, 78, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
CallTemperatureIn);
RectangularButton(widPushBtnTemperature, &canButtons, &widPushBtnHumidity, 0,
&g_sAdafruit320x240x16_ILI9325, 61, 81, 109, 79, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
CallTemperature);
RectangularButton(widPushBtnHumidity, &canButtons, 0, 0,
&g_sAdafruit320x240x16_ILI9325, 171, 81, 68, 79, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
CallHumidity);
```

```
//*****
// Panel 0 - Hlavni Canvas
```

```
Canvas(canMain, WIDGET_ROOT, 0, 0, &g_sAdafruit320x240x16_ILI9325, 0, 0, 320, 240,
CANVAS_STYLE_FILL, ClrBlack, 0, 0, 0, 0, 0, 0);
```

```
// -----
-----
-- FUNKCE
```

```
//convert float to ascii for serial print
void ftoamy2(long double f, char *buf, int dplaces)
{
```

```

int pos=0,ix,dp,num,loop;

if (f<0)
{
    buf[pos++]='-';
    f = -f;
}
dp=0;

loop = dplaces + 2;

while (f>=10.0)
{
    f=f/10.0;
    dp++;
    loop++;
}
// for (ix=1;ix<8;ix++)
for (ix=1;ix<loop;ix++)
{
    num = f;
    f=f-num;
    if (num>9)
        buf[pos++]='#';
    else
        buf[pos++]='0'+num;
    if (dp==0) buf[pos++]='.';
    f=f*10.0;
    dp--;
}

buf[pos] = '\0'; // null term
}

```

```

void itoa( int n, unsigned char * s )
{
    int j,i = 0;
    unsigned char tmp;

    if( n < 0 ) // if negativ set -
    sign
    { s[0] = '-'; s++;
      n = -n;
    }

    do // calculate next
    digit
    { s[i++] = n%10 + '0';
      n /= 10;
    } while( n > 0 );

    for( j = 0; j < i/2 ; ++j ) // revert string
    { tmp = s[j];
      s[j] = s[i-j-1];
    }
}

```

```

        s[i-j-1] = tmp;
    }
    s[i] = '\0';
}

// Smazani obrazovky LCD
void ClrScreen()
{
    sRect.sXMin = 0;
    sRect.sYMin = 0;
    sRect.sXMax = 319;
    sRect.sYMax = 239;
    GrContextForegroundSet(&sContext, ClrBlack);
    GrRectFill(&sContext, &sRect);
    GrFlush(&sContext);
}

// Delay function in milliseconds
void DelayMs(unsigned long ulSeconds)
{
    // Loop while there are more seconds to wait.
    while(ulSeconds--)
    {
        // Wait until the SysTick value is less than 1000.
        while(SysTickValueGet() > 1000)
        {
        }
        // Wait until the SysTick value is greater than 1000.
        while(SysTickValueGet() < 1000)
        {
        }
    }
}

//Interrupt for anemometer and rain gauge
void IntHandler()
{
    static unsigned int lastTime = 0;

    if(GPIOPinIntStatus(GPIO_PORTD_BASE, true) == GPIO_PIN_2 && (msTimer - lastTime
    >= 10))
    {
        lastTime = msTimer;
        //Delay(10);
        overeni = GPIOPinRead(GPIO_PORTD_BASE, GPIO_PIN_2);
        if(overeni == 0)
        {
            pocet++;
        }
    }
}

```

```

// Rain Gauge
    static unsigned int lastTime2 = 0;

    if(GPIOPinIntStatus(GPIO_PORTD_BASE, true) == GPIO_PIN_3 && (msTimer -
lastTime2 >= 10))
    {
        lastTime2 = msTimer;

        overeni2 = GPIOPinRead(GPIO_PORTD_BASE, GPIO_PIN_3);
        if(overeni2 == 0)
        {
            pocet_preklopeni++;
            UARTprintf("Preklopeni v preruseni: %d \n",
pocet_preklopeni);
        }

    }
    GPIOPinIntClear(GPIO_PORTD_BASE, GPIO_PIN_2);
    GPIOPinIntClear(GPIO_PORTD_BASE, GPIO_PIN_3);
}

// Interrupt Timer
void SysTickIntHandler()
{
    msTimer++;

    if(msTimer % 1000 == 0)
    {
        //UARTprintf("Pocet otacek = %d", pocet/2); //2 pulse per round
        //2.4 km/h = 0.666 m/s, = 1 otacka
        otacky_f = (pocet/2) * 0.6666666;

        if ((otacky_f >= 0.0)&&(otacky_f < 0.5)) beafourt = 0;    //Urovne
beafourta

        if ((otacky_f > 0.5)&&(otacky_f < 1.25)) beafourt = 1;
        if ((otacky_f > 1.25)&&(otacky_f < 3.0)) beafourt = 2;
        if ((otacky_f > 3.0)&&(otacky_f < 5.0)) beafourt = 3;
        if ((otacky_f > 5.0)&&(otacky_f < 7.0)) beafourt = 4;
        if ((otacky_f > 7.0)&&(otacky_f < 9.5)) beafourt = 5;
        if ((otacky_f > 9.5)&&(otacky_f < 12.0)) beafourt = 6;
        if ((otacky_f > 12.0)&&(otacky_f < 14.5)) beafourt = 7;
        if ((otacky_f > 14.5)&&(otacky_f < 17.5)) beafourt = 8;
        if ((otacky_f > 17.5)&&(otacky_f < 21.0)) beafourt = 9;
        if ((otacky_f > 21.0)&&(otacky_f < 24.5)) beafourt = 10;
        if ((otacky_f > 24.5)&&(otacky_f < 29.0)) beafourt = 11;
        if (otacky_f > 30.0) beafourt = 12;

        if (otacky_f_max < otacky_f) otacky_f_max = otacky_f; // ulozeni
maxima otacek

```

```

        ftoamy2(otacky_f_max, string_r_max, 2);

        string_r_kopie[0] = string_r[0];
        string_r_kopie[1] = string_r[1];
        string_r_kopie[2] = string_r[2];
        string_r_kopie[3] = string_r[3];
        string_r_kopie[4] = string_r[4];
        //Conver float do char with 2 decimal point
        ftoamy2(otacky_f, string_r, 2);
        pocet = 0;

        //Water level
        vteriny++;
    //    UARTprintf("Vterina = : %d ", vteriny);

        if(vteriny > 3599)
        {
            pocet_preklopeni_hour = pocet_preklopeni;
            pocet_preklopeni = 0;
            vteriny = 0;
        }
    }
}

```

```

void zakladniGrafika()

```

```

{
    //bily ramecek
    sRect.sXMin = 0;
    sRect.sYMin = 0;
    sRect.sXMax = 319;
    sRect.sYMax = 239;
    GrContextForegroundSet(&sContext, ClrWhite);
    GrRectDraw(&sContext, &sRect);

    //rozdelovaci cary
    GrContextForegroundSet(&sContext, ClrWhite);
    GrLineDraw(&sContext, 60, 0, 60, 160); //1
    GrLineDraw(&sContext, 0, 160, 239, 160); //2
    GrLineDraw(&sContext, 239, 0, 239, 238); //3
    GrLineDraw(&sContext, 0, 200, 239, 200); //4
    GrLineDraw(&sContext, 120, 160, 120, 238); //5
    GrLineDraw(&sContext, 0, 80, 239, 80); //6
    GrLineDraw(&sContext, 239, 140, 318, 140); //7
    GrLineDraw(&sContext, 180, 0, 180, 80); //8
    GrLineDraw(&sContext, 170, 80, 170, 160); //9
}

```

```

void bilyRamecek()

```

```

{
    //ClrScreen();
    sRect.sXMin = 0;
    sRect.sYMin = 0;
    sRect.sXMax = 319;
}

```

```

    sRect.sYMax = 239;
    GrContextForegroundSet(&sContext, ClrWhite);
    GrRectDraw(&sContext, &sRect);
}

void bile0sy()
{
    GrLineDrawH(&sContext, 10, 310, 210); // osa x
    GrLineDrawV(&sContext, 10, 10, 210); // osa y
}

//----- Widgets funkce

void Hlavni(tWidget *pWidget)
{
    zakladniGrafika();
    obnoveni_displeje = 1;
}

void BackButtonPressure()
{
    WidgetRemove((tWidget *)&widBackBtnPressure);
    WidgetRemove((tWidget *)&canPressure);
    WidgetAdd(WIDGET_ROOT, (tWidget *)&canButtons);
    WidgetPaint(WIDGET_ROOT);
    WidgetMessageQueueProcess();

    zakladniGrafika();
    obnoveni_displeje = 1;
}

void BackButtonWaterLevel()
{
    WidgetRemove((tWidget *)&widBackBtnWaterLevel);
    WidgetRemove((tWidget *)&canWaterLevel);
    WidgetAdd(WIDGET_ROOT, (tWidget *)&canButtons);
    WidgetPaint(WIDGET_ROOT);
    WidgetMessageQueueProcess();

    zakladniGrafika();
    obnoveni_displeje = 1;
}

void BackButtonDewpoint()
{
    WidgetRemove((tWidget *)&widBackBtnDewpoint);
    WidgetRemove((tWidget *)&canDewpoint);
    WidgetAdd(WIDGET_ROOT, (tWidget *)&canButtons);
    WidgetPaint(WIDGET_ROOT);
    WidgetMessageQueueProcess();

    zakladniGrafika();
}

```

```

        obnoveni_displeje = 1;
    }

    void BackButtonApparentTemperature()
    {
        WidgetRemove((tWidget *)&widBackBtnApparentTemperature);
        WidgetRemove((tWidget *)&canApparentTemperature);
        WidgetAdd(WIDGET_ROOT, (tWidget *)&canButtons);
        WidgetPaint(WIDGET_ROOT);
        WidgetMessageQueueProcess();

        zakladniGrafika();
        obnoveni_displeje = 1;
    }

    void BackButtonAltitudeMy()
    {
        WidgetRemove((tWidget *)&widBackBtnAltitudeMy);
        WidgetRemove((tWidget *)&canAltitudeMy);
        WidgetAdd(WIDGET_ROOT, (tWidget *)&canButtons);
        WidgetPaint(WIDGET_ROOT);
        WidgetMessageQueueProcess();

        zakladniGrafika();
        obnoveni_displeje = 1;
    }

    void BackButtonLight()
    {
        WidgetRemove((tWidget *)&widBackBtnLight);
        WidgetRemove((tWidget *)&canLight);
        WidgetAdd(WIDGET_ROOT, (tWidget *)&canButtons);
        WidgetPaint(WIDGET_ROOT);
        WidgetMessageQueueProcess();

        zakladniGrafika();
        obnoveni_displeje = 1;
    }

    void BackButtonWind()
    {
        WidgetRemove((tWidget *)&widBackBtnWind);
        WidgetRemove((tWidget *)&canWind);
        WidgetAdd(WIDGET_ROOT, (tWidget *)&canButtons);
        WidgetPaint(WIDGET_ROOT);
        WidgetMessageQueueProcess();

        zakladniGrafika();
        obnoveni_displeje = 1;
    }

    void BackButtonDate()
    {
        WidgetRemove((tWidget *)&widBackBtnDate);
        WidgetRemove((tWidget *)&canDate);
    }

```

```

        WidgetAdd(WIDGET_ROOT, (tWidget *)&canButtons);
        WidgetPaint(WIDGET_ROOT);
        WidgetMessageQueueProcess();

        zakladniGrafika();
        obnoveni_displeje = 1;
    }

    void BackButtonImage()
    {
        WidgetRemove((tWidget *)&widBackBtnImage);
        WidgetRemove((tWidget *)&canImage);
        WidgetAdd(WIDGET_ROOT, (tWidget *)&canButtons);
        WidgetPaint(WIDGET_ROOT);
        WidgetMessageQueueProcess();

        zakladniGrafika();
        obnoveni_displeje = 1;
    }

    void BackButtonTemperatureIn()
    {
        WidgetRemove((tWidget *)&widBackBtnTemperatureIn);
        WidgetRemove((tWidget *)&canTemperatureIn);
        WidgetAdd(WIDGET_ROOT, (tWidget *)&canButtons);
        WidgetPaint(WIDGET_ROOT);
        WidgetMessageQueueProcess();

        zakladniGrafika();
        obnoveni_displeje = 1;
    }

    void BackButtonTemperature()
    {
        WidgetRemove((tWidget *)&widBackBtnTemperature);
        WidgetRemove((tWidget *)&canTemperature);
        WidgetAdd(WIDGET_ROOT, (tWidget *)&canButtons);
        WidgetPaint(WIDGET_ROOT);
        WidgetMessageQueueProcess();

        zakladniGrafika();
        obnoveni_displeje = 1;
    }

    void BackButtonHumidity()
    {
        WidgetRemove((tWidget *)&widBackBtnHumidity);
        WidgetRemove((tWidget *)&canHumidity);
        WidgetAdd(WIDGET_ROOT, (tWidget *)&canButtons);
        WidgetPaint(WIDGET_ROOT);
        WidgetMessageQueueProcess();

        zakladniGrafika();
        obnoveni_displeje = 1;
    }
}

```

```

void CallPressure()
{
    obnoveni_displeje = 0;
    WidgetRemove((tWidget *)&canButtons);
    WidgetAdd(WIDGET_ROOT, (tWidget *)&canPressure);
    WidgetAdd(WIDGET_ROOT, (tWidget *)&widBackBtnPressure);
    WidgetPaint(WIDGET_ROOT);
    WidgetMessageQueueProcess();

    bilyRamecek();

    bileOsy();
    GrContextFontSet(&sContext, &g_sFontCm20);
    GrStringDraw(&sContext, "Pressure hPa/hours", -1, 80, 215, 1);

    GrContextForegroundSet(&sContext, ClrAqua);           //vykresleni nahodnych
hodnot do grafu
    int xx;
    for(xx = 11; xx < 311; xx++)
    {
        GrLineDrawV(&sContext, xx, poleTemp[xx-11], 209);
    }
}

```

```

void CallWaterLevel()    // vodni sloupce
{
    obnoveni_displeje = 0;
    WidgetRemove((tWidget *)&canButtons);
    WidgetAdd(WIDGET_ROOT, (tWidget *)&canWaterLevel);
    WidgetAdd(WIDGET_ROOT, (tWidget *)&widBackBtnWaterLevel);
    WidgetPaint(WIDGET_ROOT);
    WidgetMessageQueueProcess();

    bilyRamecek();

    GrLineDrawH(&sContext, 12, 44, 210);
    GrLineDrawH(&sContext, 56, 88, 210);
    GrLineDrawH(&sContext, 100, 132, 210);
    GrLineDrawH(&sContext, 144, 176, 210);
    GrLineDrawH(&sContext, 188, 220, 210);
    GrLineDrawH(&sContext, 232, 264, 210);
    GrLineDrawH(&sContext, 276, 308, 210);

    GrLineDrawV(&sContext, 12, 20, 210);
    GrLineDrawV(&sContext, 44, 20, 210);
    GrLineDrawV(&sContext, 56, 20, 210);
    GrLineDrawV(&sContext, 88, 20, 210);
    GrLineDrawV(&sContext, 100, 20, 210);
    GrLineDrawV(&sContext, 132, 20, 210);
    GrLineDrawV(&sContext, 144, 20, 210);
}

```

```

GrLineDrawV(&sContext, 176, 20, 210);
GrLineDrawV(&sContext, 188, 20, 210);
GrLineDrawV(&sContext, 220, 20, 210);
GrLineDrawV(&sContext, 232, 20, 210);
GrLineDrawV(&sContext, 264, 20, 210);
GrLineDrawV(&sContext, 276, 20, 210);
GrLineDrawV(&sContext, 308, 20, 210);

sRect.sXMin = 13; // prvni sloupec
sRect.sYMin = 209;
sRect.sXMax = 43;
sRect.sYMax = 100; //vyska = 210 - yMax
GrContextForegroundSet(&sContext, ClrLightBlue);
GrRectFill(&sContext, &sRect);

sRect.sXMin = 57; // druhy sloupec
sRect.sYMin = 209;
sRect.sXMax = 87;
sRect.sYMax = 123; //vyska = 210 - yMax
GrContextForegroundSet(&sContext, ClrLightBlue);
GrRectFill(&sContext, &sRect);

sRect.sXMin = 145; // ctvrty sloupec
sRect.sYMin = 209;
sRect.sXMax = 175;
sRect.sYMax = 60; //vyska = 210 - yMax
GrContextForegroundSet(&sContext, ClrLightBlue);
GrRectFill(&sContext, &sRect);

sRect.sXMin = 233; // sestý sloupec
sRect.sYMin = 209;
sRect.sXMax = 263;
sRect.sYMax = 200; //vyska = 210 - yMax
GrContextForegroundSet(&sContext, ClrLightBlue);
GrRectFill(&sContext, &sRect);

GrContextForegroundSet(&sContext, ClrWhite);
GrContextFontSet(&sContext, &g_sFontCm20);
GrStringDraw(&sContext, "Po", -1, 20, 215, 1);
GrStringDraw(&sContext, "Ut", -1, 64, 215, 1);
GrStringDraw(&sContext, "St", -1, 108, 215, 1);
GrStringDraw(&sContext, "Ct", -1, 152, 215, 1);
GrStringDraw(&sContext, "Pa", -1, 196, 215, 1);
GrStringDraw(&sContext, "So", -1, 240, 215, 1);
GrStringDraw(&sContext, "Ne", -1, 284, 215, 1);

```

```

}

```

```

void CallDewpoint()

```

```

{

```

```

    obnoveni_displeje = 0;
    WidgetRemove((tWidget *)&canButtons);
    WidgetAdd(WIDGET_ROOT, (tWidget *)&canDewpoint);
    WidgetAdd(WIDGET_ROOT, (tWidget *)&widBackBtnDewpoint);
    WidgetPaint(WIDGET_ROOT);

```

```

WidgetMessageQueueProcess();

bilyRamecek();

bileOsy();
GrContextFontSet(&sContext, &g_sFontCm20);
GrStringDraw(&sContext, "Dewpoint C/hours", -1, 80, 215, 1);

GrContextForegroundSet(&sContext, ClrLightGreen);           //vykresleni
nahodnych hodnot do grafu
int xx;
for(xx = 11; xx < 310; xx++)
{
GrLineDrawV(&sContext, xx, (209-(rand() % 100)), 209);
}
}

void CallApparentTemperature()
{
obnoveni_displeje = 0;
WidgetRemove((tWidget *)&canButtons);
WidgetAdd(WIDGET_ROOT, (tWidget *)&canApparentTemperature);
WidgetAdd(WIDGET_ROOT, (tWidget *)&widBackBtnApparentTemperature);
WidgetPaint(WIDGET_ROOT);
WidgetMessageQueueProcess();

bilyRamecek();

bileOsy();
GrContextFontSet(&sContext, &g_sFontCm20);
GrStringDraw(&sContext, "Apparent Temperature C/hours", -1, 20, 215, 1);

GrContextForegroundSet(&sContext, ClrGreenYellow);           //vykresleni
nahodnych hodnot do grafu
int xx;
for(xx = 11; xx < 310; xx++)
{
GrLineDrawV(&sContext, xx, (209-(rand() % 100)), 209);
}
}

void CallAltitudeMy()
{
obnoveni_displeje = 0;
WidgetRemove((tWidget *)&canButtons);
WidgetAdd(WIDGET_ROOT, (tWidget *)&canAltitudeMy);
WidgetAdd(WIDGET_ROOT, (tWidget *)&widBackBtnAltitudeMy);
WidgetPaint(WIDGET_ROOT);
WidgetMessageQueueProcess();

bilyRamecek();

GrContextFontSet(&sContext, &g_sFontCm20);
GrStringDraw(&sContext, "AltitudeMy", -1, 20, 50, 1);
}

```

```

void CallLight()
{
    obnoveni_displeje = 0;
    WidgetRemove((tWidget *)&canButtons);
    WidgetAdd(WIDGET_ROOT, (tWidget *)&canLight);
    WidgetAdd(WIDGET_ROOT, (tWidget *)&widBackBtnLight);
    WidgetPaint(WIDGET_ROOT);
    WidgetMessageQueueProcess();

    bilyRamecek();

    GrContextFontSet(&sContext, &g_sFontCm26b);
    GrStringDraw(&sContext, "Light min = 156 lux", -1, 20, 50, 1);
    GrStringDraw(&sContext, "Light now = 453 lux", -1, 20, 100, 1);
    GrStringDraw(&sContext, "Light max = 968 lux", -1, 20, 150, 1);
}

```

```

void CallWind()
{
    obnoveni_displeje = 0;
    WidgetRemove((tWidget *)&canButtons);
    WidgetAdd(WIDGET_ROOT, (tWidget *)&canWind);
    WidgetAdd(WIDGET_ROOT, (tWidget *)&widBackBtnWind);
    WidgetPaint(WIDGET_ROOT);
    WidgetMessageQueueProcess();

    bilyRamecek();

    //kruh pro smer vetru
    GrContextForegroundSet(&sContext, ClrViolet);
    GrCircleDraw(&sContext, 160, 120, 70);

    // vykresleni cary smeru vetru
    GrContextForegroundSet(&sContext, ClrLightPink);
    double alfa = 0;
    double beta = 0;
    alfa = ((cos(smer)*70)+160);
    beta = ((-1)*sin(smer)*70)+120;
    GrLineDraw(&sContext, 160, 120, alfa, beta);

    GrContextFontSet(&sContext, &g_sFontCm20);
    GrStringDraw(&sContext, "Wind max = 9,7 m/s", -1, 40, 215, 1);
}

```

```

void CallDate()
{
    obnoveni_displeje = 0;
    WidgetRemove((tWidget *)&canButtons);
    WidgetAdd(WIDGET_ROOT, (tWidget *)&canDate);
    WidgetAdd(WIDGET_ROOT, (tWidget *)&widBackBtnDate);
    WidgetPaint(WIDGET_ROOT);
    WidgetMessageQueueProcess();
}

```

```

    bilyRamecek();

    GrContextFontSet(&sContext, &g_sFontCm30);
    GrStringDraw(&sContext, "year 2013", -1, 20, 50, 1);
    GrStringDraw(&sContext, "date 18.8", -1, 20, 80, 1);
    GrStringDraw(&sContext, "time 19:02:54", -1, 20, 110, 1);
    GrStringDraw(&sContext, "gps N 49°12'41.861", -1, 20, 140, 1);
    GrStringDraw(&sContext, "gps E 17°38'45.607", -1, 20, 170, 1);
}

void CallImage()
{
    obnoveni_displeje = 0;
    WidgetRemove((tWidget *)&canButtons);
    WidgetAdd(WIDGET_ROOT, (tWidget *)&canImage);
    WidgetAdd(WIDGET_ROOT, (tWidget *)&widBackBtnImage);
    WidgetPaint(WIDGET_ROOT);
    WidgetMessageQueueProcess();

    bilyRamecek();

    GrContextFontSet(&sContext, &g_sFontCm30i);
    GrStringDraw(&sContext, "Meteostanice", -1, 40, 50, 1);
    GrStringDraw(&sContext, "verze: 2013-08-19", -1, 40, 100, 1);
}

void CallTemperatureIn()
{
    obnoveni_displeje = 0;
    WidgetRemove((tWidget *)&canButtons);
    WidgetAdd(WIDGET_ROOT, (tWidget *)&canTemperatureIn);
    WidgetAdd(WIDGET_ROOT, (tWidget *)&widBackBtnTemperatureIn);
    WidgetPaint(WIDGET_ROOT);
    WidgetMessageQueueProcess();

    bilyRamecek();

    bileOsy();
    GrContextFontSet(&sContext, &g_sFontCm20);
    GrStringDraw(&sContext, "Temperature In C/hours", -1, 80, 215, 1);

    GrContextForegroundColorSet(&sContext, ClrLightGreen); //vykresleni
    nahodnych hodnot do grafu
    int xx;
    for(xx = 11; xx < 310; xx++)
    {
        GrLineDrawV(&sContext, xx, (209-(rand() % 100)), 209);
    }
}

void CallTemperature()
{
    obnoveni_displeje = 0;
    WidgetRemove((tWidget *)&canButtons);
    WidgetAdd(WIDGET_ROOT, (tWidget *)&canTemperature);
}

```

```

WidgetAdd(WIDGET_ROOT, (tWidget *)&widBackBtnTemperature);
WidgetPaint(WIDGET_ROOT);
WidgetMessageQueueProcess();

bilyRamecek();

bileOsy();
GrContextFontSet(&sContext, &g_sFontCm20);
GrStringDraw(&sContext, "Temperature C/hours", -1, 80, 215, 1);

GrContextForegroundSet(&sContext, ClrRed);           //vykresleni nahodnych
hodnot do grafu
int xx;
for(xx = 11; xx < 310; xx++)
{
GrLineDrawV(&sContext, xx, (209-(rand() % 100)), 209);
}
}

void CallHumidity()
{
obnoveni_displeje = 0;
WidgetRemove((tWidget *)&canButtons);
WidgetAdd(WIDGET_ROOT, (tWidget *)&canHumidity);
WidgetAdd(WIDGET_ROOT, (tWidget *)&widBackBtnHumidity);
WidgetPaint(WIDGET_ROOT);
WidgetMessageQueueProcess();

bilyRamecek();

bileOsy();
GrContextFontSet(&sContext, &g_sFontCm20);
GrStringDraw(&sContext, "Humidity %/hours", -1, 80, 215, 1);

GrContextForegroundSet(&sContext, ClrYellow);       //vykresleni nahodnych
hodnot do grafu
int xx;
for(xx = 11; xx < 310; xx++)
{
GrLineDrawV(&sContext, xx, (209-(rand() % 100)), 209);
}
}

// -----
-----
----  MAIN

int main(void)
{
// tContext sContext2;

// Save FPU Registers before Interrupt calling
ROM_FPULazyStackingEnable();
// Configure the system clock to run at 40MHz

```

```

);

SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN

// Set up a SysTick Interrupt 1000 per second
SysTickPeriodSet(SysCtlClockGet() / 1000);
SysTickIntEnable();
SysTickEnable();

// SHT75
// Configure pins PD6 and PD7 as output for SHT75
//First open the lock and select the bits we want to modify in the GPIO commit
register.
SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
HWREG(GPIO_PORTD_BASE + GPIO_O_LOCK) = GPIO_LOCK_KEY_DD;
HWREG(GPIO_PORTD_BASE + GPIO_O_CR) = 0x80;
GPIOPinTypeGPIOOutput(GPIO_PORTD_BASE, GPIO_PIN_6 | GPIO_PIN_7);
GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_7, GPIO_PIN_7);
GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_6, 0);

// Anemometer PD2 and Rain Gauge PD3
// Enable the GPIO port D for interrupt
SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
// Configures pin(s) for use as GPIO inputs PD2 - INPUT for Anemometer
GPIOPinTypeGPIOInput(GPIO_PORTD_BASE, GPIO_PIN_2 | GPIO_PIN_3);
// PadConfigureSet - for PULL-UP (is insert in GPIOPinTypeInput only as PUSH-
PULL)
GPIOPadConfigSet(GPIO_PORTD_BASE, GPIO_PIN_2 | GPIO_PIN_3, GPIO_STRENGTH_2MA,
GPIO_PIN_TYPE_STD_WPU);
// GPIOPadConfigSet(GPIO_PORTD_BASE, GPIO_PIN_3, GPIO_STRENGTH_2MA,
GPIO_PIN_TYPE_STD_WPU);
// Define edge of pin for interrupt
GPIOIntTypeSet(GPIO_PORTD_BASE, GPIO_PIN_2 | GPIO_PIN_3, GPIO_FALLING_EDGE);

// Enable Interrupt for pin
GPIOPinIntEnable(GPIO_PORTD_BASE, GPIO_PIN_2 | GPIO_PIN_3);

// Enable interrupt for PORT D, IRQ number
IntEnable(19);
// Enable Interrupt
IntMasterEnable();

// BMP085
//I2C0 enable and run
I2CSetup(I2C2_MASTER_BASE, 0);
ROM_IntMasterEnable();

// Wind direction PE1
SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0); //povolení ADC0

SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE); //povolení portu E
GPIOPinTypeADC(GPIO_PORTE_BASE, GPIO_PIN_1 ); //povolení pinu 1

```

```

    SysCtlADCSpeedSet(SYSCTL_ADCSPEED_250KSPS);           //rychlost vzorkování
//  ADCHardwareOversampleConfigure(ADC0_BASE, 64);      //přidáno, HW průměrování
    ADCSequenceDisable(ADC0_BASE,1);                    //vypnutí ADC před
konfigurací
    ADCSequenceConfigure(ADC0_BASE, 1, ADC_TRIGGER_PROCESSOR, 0); // konfigurace
ADC: ADC0, SS2, Trigger - CPU, Priority - 0 (high)
    ADCSequenceStepConfigure(ADC0_BASE, 1, 0, ADC_CTL_CH0); //konfigurace
sequenceru 3 pro kanal 0, povolene preruseni
    ADCSequenceStepConfigure(ADC0_BASE, 1, 1, ADC_CTL_CH1); //konfigurace
sequenceru 3 pro kanal 0, povolene preruseni
    ADCSequenceStepConfigure(ADC0_BASE, 1, 2, ADC_CTL_CH2); //konfigurace
sequenceru 3 pro kanal 0, povolene preruseni
    ADCSequenceStepConfigure(ADC0_BASE, 1, 3, ADC_CTL_CH3 | ADC_CTL_IE |
ADC_CTL_END); //konfigurace sequenceru 3 pro kanal 0, povolene preruseni
    ADCSequenceEnable(ADC0_BASE, 1); //povolení ADC0 Sequenceru SS3
    ADCIntClear(ADC0_BASE, 1); //vymazani priznaku preruseni

//port F for LED
SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2, 0x00);
GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_3, 0xff);

// Initialize the UART.
ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
ROM_GPIOPinConfigure(GPIO_PA0_U0RX);
ROM_GPIOPinConfigure(GPIO_PA1_U0TX);
ROM_GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);
UARTStdioInit(0);

// START
UARTprintf("START\n");

// Initialize UART 3

//
// Enable Peripheral Clocks
//
SysCtlPeripheralEnable(SYSCTL_PERIPH_UART3);
SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOC);

//
// Enable port PC6 for UART3 U3RX
//
GPIOPinConfigure(GPIO_PC6_U3RX);
GPIOPinTypeUART(GPIO_PORTC_BASE, GPIO_PIN_6);

```

```

//
// Enable port PC7 for UART3 U3TX
//
GPIOPinConfigure(GPIO_PC7_U3TX);
GPIOPinTypeUART(GPIO_PORTC_BASE, GPIO_PIN_7);

//
// Initialize the UART. Set the baud rate, number of data bits, turn off
// parity, number of stop bits, and stick mode.
//

UARTConfigSetExpClk(UART3_BASE, SysCtlClockGet(), 115200,
                    (UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE |
                     UART_CONFIG_PAR_NONE));

//
// Enable the UART.
//
UARTEnable(UART3_BASE);

//get calibration data from BMP085
if(bmp085_enable == 1) bmp_get_cal();

// Inicializace LCD
Adafruit320x240x16_ILI9325Init();
GrContextInit(&sContext, &g_sAdafruit320x240x16_ILI9325);
ClrScreen();

// Initialize the graphics context.
//GrContextInit(&sContext, &g_sAdafruit320x240x16_ILI9325);

SysCtlPeripheralEnable(SYSCTL_PERIPH_UDMA);
SysCtlDelay(10);
uDMAControlBaseSet(&sDMAControlTable[0]);
uDMAEnable();

if(kalibrace == 1)
{
    // Initialize the touch screen driver and have it route its messages to the
    widget tree.
    TouchScreenInit();

    // Paint touch calibration targets and collect calibration data
    GrContextForegroundSet(&sContext, ClrWhite);
    GrContextBackgroundSet(&sContext, ClrBlack);
    GrContextFontSet(&sContext, &g_sFontCm20);
    GrStringDraw(&sContext, "Tukni na kolecka pro kalibraci", -1, 0, 0, 1);

    GrCircleDraw(&sContext, 32, 34, 10);
    GrFlush(&sContext);
}

```

```

    TouchScreenCalibrationPoint(32, 34, 0);

    GrCircleDraw(&sContext, 280, 200, 10);
    GrFlush(&sContext);
    TouchScreenCalibrationPoint(280, 200, 1);

    GrCircleDraw(&sContext, 200, 40, 10);
    GrFlush(&sContext);
    TouchScreenCalibrationPoint(200, 40, 2);

    // Calculate and set calibration matrix
    long* plCalibrationMatrix = TouchScreenCalibrate();
    // Enable touch screen event handler for grlib widgets
    TouchScreenCallbackSet(WidgetPointerMessage);
}

ClrScreen();

    TouchScreenCallbackSet(WidgetPointerMessage);

    WidgetAdd(WIDGET_ROOT, (tWidget *)&canMain);
    WidgetRemove((tWidget *)&widBackBtnPressure);
    WidgetRemove((tWidget *)&widBackBtnWaterLevel);
    WidgetRemove((tWidget *)&widBackBtnDewpoint);
    WidgetRemove((tWidget *)&widBackBtnApparentTemperature);
    WidgetRemove((tWidget *)&widBackBtnAltitudeMy);
    WidgetRemove((tWidget *)&widBackBtnLight);
    WidgetRemove((tWidget *)&widBackBtnWind);
    WidgetRemove((tWidget *)&widBackBtnDate);
    WidgetRemove((tWidget *)&widBackBtnImage);
    WidgetRemove((tWidget *)&widBackBtnTemperatureIn);
    WidgetRemove((tWidget *)&widBackBtnTemperature);
    WidgetRemove((tWidget *)&widBackBtnHumidity);

    WidgetPaint(WIDGET_ROOT);
    WidgetMessageQueueProcess();

    WidgetAdd(WIDGET_ROOT, (tWidget *)&canButtons);
    WidgetPaint(WIDGET_ROOT);
    WidgetMessageQueueProcess();

    zakladniGrafika();
    obnoveni_displeje = 1;

    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0x00);
    //zhasnutí všech LED diod

    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_3, 0xff);    //zapne LCD podsvit

```

```

// -----
// -----
WHILE CYCLE

    while(1)
    {
//      nahodne = rand() % 100;
      WidgetMessageQueueProcess();    //vykresleni widgetu

      //Pressure
      pressure = bmp_get_value_pres();
      pressure = pressure / 100;
      altitude_my = bmp_get_value_alt();

      ftoamy2(pressure, string_t1, 2);
      ftoamy2(altitude_my, string_v, 2);

      // Temperature
      resetSHT(); //reset SHT75
      startSHT(); //start sequence from sensor datasheet
      writeByteSHT(0x03);
      temperature = readByteSHT();
      temperature_f = -40.1 + (0.01 * temperature);

      if (temperature_f_max < temperature_f) temperature_f_max =
temperature_f; // ulozeni maxima teploty
      ftoamy2(temperature_f_max, string_t_max, 2);

      if (temperature_f_min > temperature_f) temperature_f_min =
temperature_f; // ulozeni minima teploty
      ftoamy2(temperature_f_min, string_t_min, 2);

      //Convert float do char with 2 decimal point
      ftoamy2(temperature_f, string_t, 2);
      //      UARTprintf("Temperature = %s C  ", string_t);

      temperature_i = (temperature_f*100); //temperature_i pro LED displej
      if((temperature_i < 1000)&&(temperature_i > 100))
      {
          itoa(temperature_i, poleTempSend+1);
          poleTempSend[0] = '0';
      }
      if(temperature_i < 100)
      {
          itoa(temperature_i, poleTempSend+2);
          poleTempSend[0] = '0';
          poleTempSend[1] = '0';
      }

      if(temperature_i > 1000) itoa(temperature_i, poleTempSend);
//naplni pole hodnotou temperature_i

```

```

LED displej    for(sendChar = 0; sendChar<4;sendChar++)    //odesila znaky z pole pro
{
    UARTCharPut(UART3_BASE, poleTempSend[sendChar]);
}

    if(poleTempIndex > 300) poleTempIndex = 0;
    poleTemp[poleTempIndex] = temperature_i/100;
    poleTempIndex++;

//    DelayMs(100);

    // Humidity
    resetSHT(); //reset SHT75
    startSHT(); //start sequence from sensor datasheet
    writeByteSHT(0x05);
    humidity = readByteSHT();

    humidity_f = -2.0468 + (0.0367 * humidity) + ((-1.5955*1e-
6)*humidity*humidity); //vypocet linearni vlhkosti
    humidity_real = (temperature_f - 25) * (0.01 + (0.00008 * humidity)) +
humidity_f;

    if (humidity_real_max < humidity_real) humidity_real_max =
humidity_real; // ulozeni maxima humidity
    ftoamy2(humidity_real_max, string_h_max, 2);

    if (humidity_real_min > humidity_real) humidity_real_min =
humidity_real; // ulozeni minima humidity
    ftoamy2(humidity_real_min, string_h_min, 2);

    //Conver float do char with 2 decimal point
    ftoamy2(humidity_real, string_h, 2);
    string_h_i[0] = string_h[0];
    string_h_i[1] = string_h[1];

    //    UARTprintf("Humidity = %s %% ", string_h);

    // Dew point
    dewa = log(humidity_real/100);
    dewb = (17.62 * temperature_f) / (243.12 + temperature_f);
//vypocet rosneho bodu

    dewpoint = 243.12 * ((dewa + dewb)/(17.62 - dewa - dewb));

    ftoamy2(dewpoint, string_d, 2);
    //    UARTprintf("Dew point = %s ", string_d);

```

```

//Apparent temperature
//Water vapour pressure wvp
wvp_exp = (temperature_f/(237.7+temperature_f))*17.27;
wvp_exp = exp(wvp_exp);
wvp = (humidity_real/100)*wvp_exp;

temperature_app = temperature_f+(0.33*wvp)-(0.7*otacky_f)-4;
ftoamy2(temperature_app, string_t_a, 2);

// Wind direction
ADCIntClear(ADC0_BASE, 1); //clear interrupt for ADC0
ADCProcessorTrigger(ADC0_BASE, 1); // Trigger od CPU
while(!ADCIntStatus(ADC0_BASE, 1, false)) //cekani na konverzi dat od
ADC0
{
}

ADCSequenceDataGet(ADC0_BASE, 1, u1ADC0_Value); //precteni dat
// UARTprintf(" AIN2(PE1) = %4d ", u1ADC0_Value[2]);

wvalue = u1ADC0_Value[2];

if(wvalue <= 552) {strcpy(pole, "W_"); smer = 0;}
else if(wvalue <= 940) {strcpy(pole, "NW"); smer = (7*PI/4);}
else if(wvalue <= 1472) {strcpy(pole, "N_"); smer = (6*PI/4);}
else if(wvalue <= 2127) {strcpy(pole, "SW"); smer = (PI/4);}
else if(wvalue <= 2748) {strcpy(pole, "NE"); smer = (5*PI/4);}
else if(wvalue <= 3232) {strcpy(pole, "S_"); smer = (2*PI/4);}
else if(wvalue <= 3535) {strcpy(pole, "SE"); smer = (3*PI/4);}
else if(wvalue <= 4096) {strcpy(pole, "E_"); smer = PI;}
else {strcpy(pole, "W_"); smer = 0;}

// UARTprintf("%s\n", pole);

// Water Level
// UARTprintf(" pocet preklopeni hour: %d", pocet_preklopeni_hour);
water_level = pocet_preklopeni_hour * 0.2794;
ftoamy2(water_level, string_w, 4);
// UARTprintf(" Water level: = %s \n", string_w);

if (water_level_max < water_level) water_level_max = water_level; //
ulozeni maxima vody
ftoamy2(water_level_max, string_w_max, 4);

// ----- GRAFIKA

if(obnoveni_displeje == 1)
{

```

```

//Smazani mista pro smerovku vetru
sRect.sXMin = 240;
sRect.sYMin = 2;
sRect.sXMax = 317;
sRect.sYMax = 118;
GrContextForegroundSet(&sContext, ClrBlack);
GrRectFill(&sContext, &sRect);

//kruh pro smer vetru
GrContextForegroundSet(&sContext, ClrPink);
GrCircleDraw(&sContext, 279, 40, 35);

// vykresleni cary smeru vetru
GrContextForegroundSet(&sContext, ClrLightPink);
double alfa = 0;
double beta = 0;

alfa = ((cos(smer) * circle_rad)+circle_x);
beta = ( (-1) *sin(smer) *circle_rad) +circle_y ;

GrLineDraw(&sContext, circle_x, circle_y, alfa, beta );

//Smazani mista pro cisla rychlosti vetru
sRect.sXMin = 240;
sRect.sYMin = 85;
sRect.sXMax = 294;
sRect.sYMax = 105;
GrContextForegroundSet(&sContext, ClrBlack);
GrRectFill(&sContext, &sRect);

//Text rychlost vetru
GrContextForegroundSet(&sContext, ClrPink);
GrContextFontSet(&sContext, &g_sFontCmss20b);
GrStringDraw(&sContext, string_r, -1, 243, 85, 0);
GrContextFontSet(&sContext, &g_sFontCmss12b);
GrStringDraw(&sContext, "m/s", -1, 291, 92, 0);

//Smazani mista pro cisla maximalni rychlosti vetru
sRect.sXMin = 240;
sRect.sYMin = 105;
sRect.sXMax = 270;
sRect.sYMax = 119;
GrContextForegroundSet(&sContext, ClrBlack);
GrRectFill(&sContext, &sRect);

//Text rychlost vetru
GrContextForegroundSet(&sContext, ClrPink);
GrContextFontSet(&sContext, &g_sFontCmss16b);
GrStringDraw(&sContext, string_r_max, -1, 243, 105, 0);
GrContextFontSet(&sContext, &g_sFontCmss12b);
GrStringDraw(&sContext, "m/s", -1, 281, 107, 0);

//Smazani mista pro Beafourtova stupnici

```

```

sRect.sXMin = 240;
sRect.sYMin = 120;
sRect.sXMax = 315;
sRect.sYMax = 139;
GrContextForegroundSet(&sContext, ClrBlack);
GrRectFill(&sContext, &sRect);

//Text rychlost vetru
GrContextForegroundSet(&sContext, ClrPink);
GrContextFontSet(&sContext, &g_sFontCmss16b);
GrStringDraw(&sContext, beafourt_table[beafourt], -1, 243, 125, 0);

//Smazani mista pro cisla teploty
sRect.sXMin = 65;
sRect.sYMin = 90;
sRect.sXMax = 140;
sRect.sYMax = 115;
GrContextForegroundSet(&sContext, ClrBlack);
GrRectFill(&sContext, &sRect);

//Text s teplotou
GrContextForegroundSet(&sContext, ClrRed);
GrContextFontSet(&sContext, &g_sFontCmss30b);
GrStringDraw(&sContext, string_t, -1, 65, 90, 0);
GrStringDraw(&sContext, "C", -1, 140, 90, 0);

//Smazani mista pro cisla teploty_max
sRect.sXMin = 65;
sRect.sYMin = 120;
sRect.sXMax = 110;
sRect.sYMax = 135;
GrContextForegroundSet(&sContext, ClrBlack);
GrRectFill(&sContext, &sRect);

//Text s teplotou_max
GrContextForegroundSet(&sContext, ClrRed);
GrContextFontSet(&sContext, &g_sFontCmss12b);
GrStringDraw(&sContext, string_t_max, -1, 65, 120, 0);
GrStringDraw(&sContext, "C", -1, 100, 120, 0);

//Smazani mista pro cisla teploty_min
sRect.sXMin = 65;
sRect.sYMin = 135;
sRect.sXMax = 110;
sRect.sYMax = 150;
GrContextForegroundSet(&sContext, ClrBlack);
GrRectFill(&sContext, &sRect);

//Text s teplotou_min
GrContextForegroundSet(&sContext, ClrRed);
GrContextFontSet(&sContext, &g_sFontCmss12b);
GrStringDraw(&sContext, string_t_min, -1, 65, 135, 0);
GrStringDraw(&sContext, "C", -1, 100, 135, 0);

```

```

//Smazani mista pro cisla vlhkosti
sRect.sXMin = 175;
sRect.sYMin = 90;
sRect.sXMax = 210;
sRect.sYMax = 115;
GrContextForegroundSet(&sContext, ClrBlack);
GrRectFill(&sContext, &sRect);

//Text s vlhkosti
GrContextForegroundSet(&sContext, ClrYellow);
GrContextFontSet(&sContext, &g_sFontCmss30b);
GrStringDraw(&sContext, string_h_i, -1, 175, 90, 0);
GrStringDraw(&sContext, "%", -1, 210, 90, 0);

//Smazani mista pro cisla vlhkosti_max
sRect.sXMin = 175;
sRect.sYMin = 120;
sRect.sXMax = 230;
sRect.sYMax = 135;
GrContextForegroundSet(&sContext, ClrBlack);
GrRectFill(&sContext, &sRect);

//Text s vlhkosti_max
GrContextForegroundSet(&sContext, ClrYellow);
GrContextFontSet(&sContext, &g_sFontCmss12b);
GrStringDraw(&sContext, string_h_max, -1, 175, 120, 0);
GrStringDraw(&sContext, "%", -1, 210, 120, 0);

//Smazani mista pro cisla vlhkosti_min
sRect.sXMin = 175;
sRect.sYMin = 135;
sRect.sXMax = 230;
sRect.sYMax = 150;
GrContextForegroundSet(&sContext, ClrBlack);
GrRectFill(&sContext, &sRect);

//Text s vlhkosti_min
GrContextForegroundSet(&sContext, ClrYellow);
GrContextFontSet(&sContext, &g_sFontCmss12b);
GrStringDraw(&sContext, string_h_min, -1, 175, 135, 0);
GrStringDraw(&sContext, "%", -1, 210, 135, 0);

//Smazani mista pro rosny bod
sRect.sXMin = 10;
sRect.sYMin = 170;
sRect.sXMax = 115;
sRect.sYMax = 195;
GrContextForegroundSet(&sContext, ClrBlack);
GrRectFill(&sContext, &sRect);

//Text s rosny bodem
GrContextForegroundSet(&sContext, ClrLightGreen);
GrContextFontSet(&sContext, &g_sFontCmss30b);
GrStringDraw(&sContext, string_d, -1, 10, 170, 0);
GrStringDraw(&sContext, "C", -1, 90, 170, 0);

```

```

//Smazani mista pro tlak
sRect.sXMin = 5;
sRect.sYMin = 5;
sRect.sXMax = 55;
sRect.sYMax = 25;
GrContextForegroundSet(&sContext, ClrBlack);
GrRectFill(&sContext, &sRect);

//Text s tlakem
GrContextForegroundSet(&sContext, ClrAliceBlue);
GrContextFontSet(&sContext, &g_sFontCmss12b);
GrStringDraw(&sContext, string_tl, -1, 5, 5, 0);
GrStringDraw(&sContext, "hPa", -1, 15, 15, 0);

//Smazani mista pro nadmorskou vysku
sRect.sXMin = 130;
sRect.sYMin = 170;
sRect.sXMax = 235;
sRect.sYMax = 195;
GrContextForegroundSet(&sContext, ClrBlack);
GrRectFill(&sContext, &sRect);

//Text s nadmorskou vyskou
GrContextForegroundSet(&sContext, ClrMoccasin);
GrContextFontSet(&sContext, &g_sFontCmss24b);
GrStringDraw(&sContext, string_v, -1, 130, 170, 0);
GrContextForegroundSet(&sContext, ClrMoccasin);
GrContextFontSet(&sContext, &g_sFontCmss12b);
GrStringDraw(&sContext, "m.n.m.", -1, 200, 178, 0);

//Smazani mista pro zdanlivou teplotu
sRect.sXMin = 10;
sRect.sYMin = 205;
sRect.sXMax = 115;
sRect.sYMax = 235;
GrContextForegroundSet(&sContext, ClrBlack);
GrRectFill(&sContext, &sRect);

//Text se zdanlivou teplotou
GrContextForegroundSet(&sContext, ClrGreenYellow);
GrContextFontSet(&sContext, &g_sFontCmss30b);
GrStringDraw(&sContext, string_t_a, -1, 10, 205, 0);
GrStringDraw(&sContext, "C", -1, 90, 205, 0);

//Smazani mista pro mnozstvi srazek
sRect.sXMin = 5;
sRect.sYMin = 85;
sRect.sXMax = 55;
sRect.sYMax = 105;
GrContextForegroundSet(&sContext, ClrBlack);
GrRectFill(&sContext, &sRect);

//Text s mnozstvimi srazek
GrContextForegroundSet(&sContext, ClrAqua);

```

```
GrContextFontSet(&sContext, &g_sFontCmss12b);
GrStringDraw(&sContext, string_w, -1, 5, 85, 0);
GrStringDraw(&sContext, "mm/h", -1, 15, 95, 0);

//Smazani mista pro maximum mnozstvi srazek
sRect.sXMin = 5;
sRect.sYMin = 115;
sRect.sXMax = 55;
sRect.sYMax = 135;
GrContextForegroundSet(&sContext, ClrBlack);
GrRectFill(&sContext, &sRect);

//Text s maximem mnozstvi srazek
GrContextForegroundSet(&sContext, ClrAqua);
GrContextFontSet(&sContext, &g_sFontCmss12b);
GrStringDraw(&sContext, string_w_max, -1, 5, 115, 0);
GrStringDraw(&sContext, "mm/h", -1, 15, 125, 0);

//Smazani mista pro hodiny
sRect.sXMin = 240;
sRect.sYMin = 142;
sRect.sXMax = 318;
sRect.sYMax = 238;
GrContextForegroundSet(&sContext, ClrBlack);
GrRectFill(&sContext, &sRect);

//Text s hodinami
GrContextForegroundSet(&sContext, ClrGainsboro);
GrContextFontSet(&sContext, &g_sFontCmss20b);
GrStringDraw(&sContext, string_time, 8, 242, 142, 0);
```

```
}
}
}
```