IMST – Innovationen machen Schulen Top

Informatik kreativ unterrichten

ANHANG ZUM SCHLUSSBERICHT

ID 636 - "LERN-PROZESS(OR)" - Individualisiertes Lernen

Projektnehmer:

Dipl.-Ing. Daniel **Esterl**

HTL Mössingerstraße 25, 9020 Klagenfurt

Klagenfurt, Juli 2012

INHALTSVERZEICHNIS DER ANHÄNGE

1 VORSTELLUNG DES IMST-PROJEKTES

- IMST-Workshop September 2012
- Villach IGIP Veranstaltung September 2012
- Leonardo-Projekt „Microcontroller applications in vocational application“ 2011-13

  Teilnehmer: Türkei, Sizilien, Belgien, Rumänien, Bulgarien, Slowenien, Spanien

2 ANHÄNGE

2.1 Anhang BSP1

2.1.1 Line-Runner

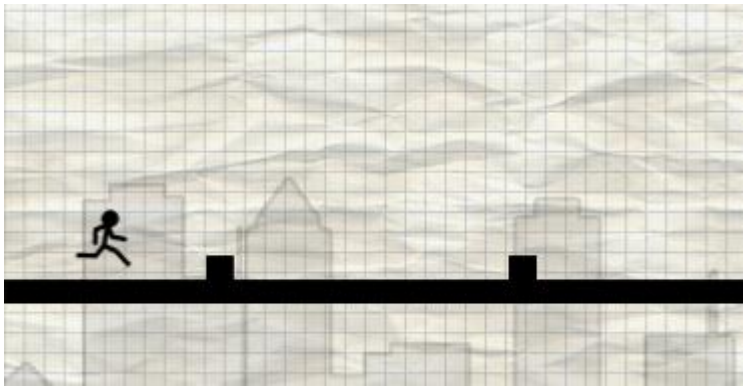
2.1.2 Idee, Ziel (Aufgabenstellung):

Unser Ziel war es, ein beliebiges Spiel auf unserem MyPic – Board, mit geringstmöglichem Hardwareaufwand zu realisieren.
Da es auf dem MyPic – Board nur LEDs als Ausgabemethode gibt, kamen wir bald auf die Idee als Peripherie ein LCD zu verwenden.

Da nun sowohl Ausgabe als auch eigentliche Steuerung des Spiels festgelegt worden waren, kamen wir zum Schluss, das Spiel Line Runner programmieren zu wollen, da dies sowohl mit den gestellten Forderungen von Seiten des Lehrers, als auch dem vorhandenen Know-How perfekt übereinstimmte.

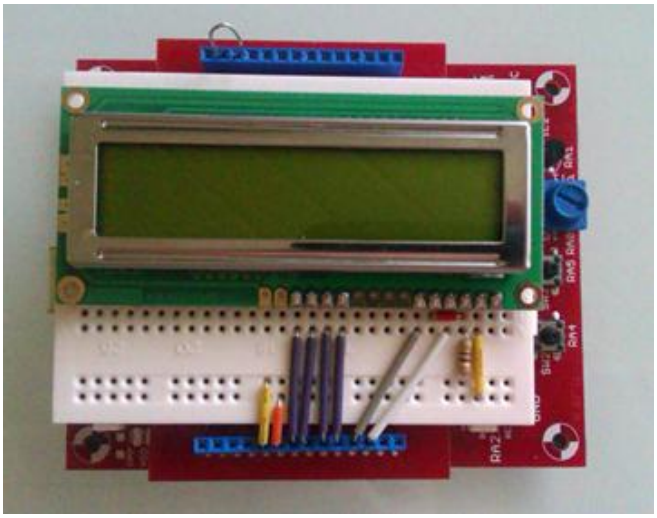
2.1.3 Planung, Blockschaltbild:

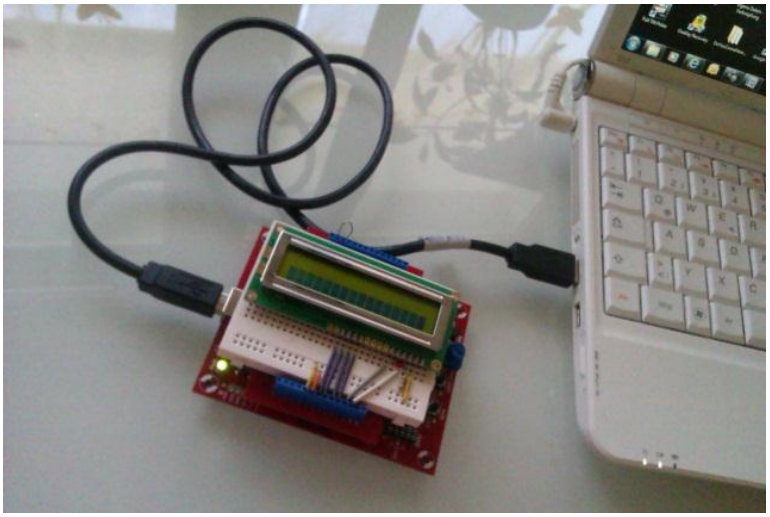
Eine kurze Erklärung des Spiels Line Runner:
Ziel ist es, mit einer Figur springend, immer schneller wiederkehrende Hindernisse zu überwinden.
In unserem Fall findet das Spiel auf einer zweizeiligen LCD statt. In periodischen Abständen bewegen sich in der unteren Zeile Hindernisse vom rechten Rand der LCD zum linken Rand.
Die Figur - ein Strichmännchen - wird über einen Taster zum Springen gebracht.
Wenn der Taster gedrückt ist, springt die Figur in die obere Zeile der LCD und weicht somit dem Hindernis aus. Um die Schwierigkeit des Spiels zu erhöhen, ist die Sprungzeit begrenzt und nach Ablaufen der Zeit wird die Figur zurück auf die untere Zeile gesetzt.
Wenn man mit der Figur ein Hindernis berührt, oder darauf springt, ist das Spiel vorbei und man muss von Neuem beginnen.
Weiters gibt es nach jeweils zwanzig Hindernissen ein „Level Up“ , bei dem sich die Geschwindigkeit der Hindernisse erhöht.
Nach Durchlaufen von zehn Levels oder dem Berühren eines Hindernisses sieht man seine erreichte Punktezahl, die sich aus insgesamt erreichter Weite, übersprungenen Hindernissen und Anzahl der „Level Ups“ errechnet.

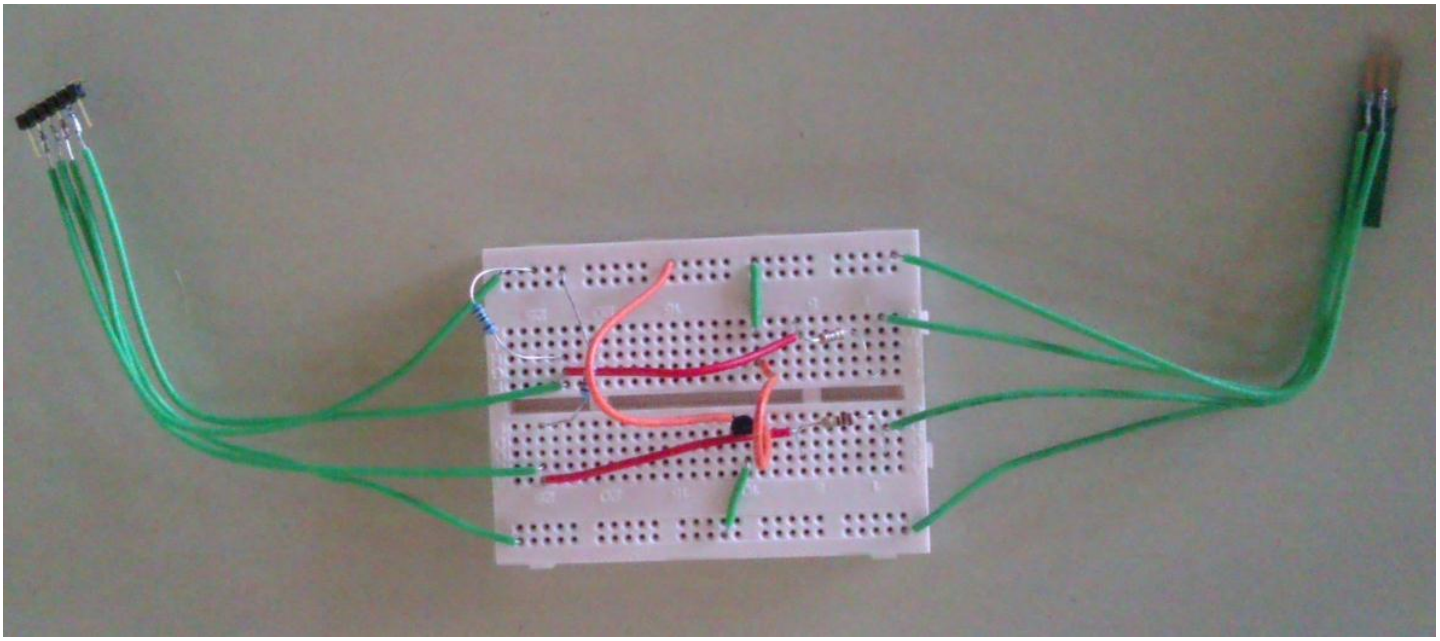


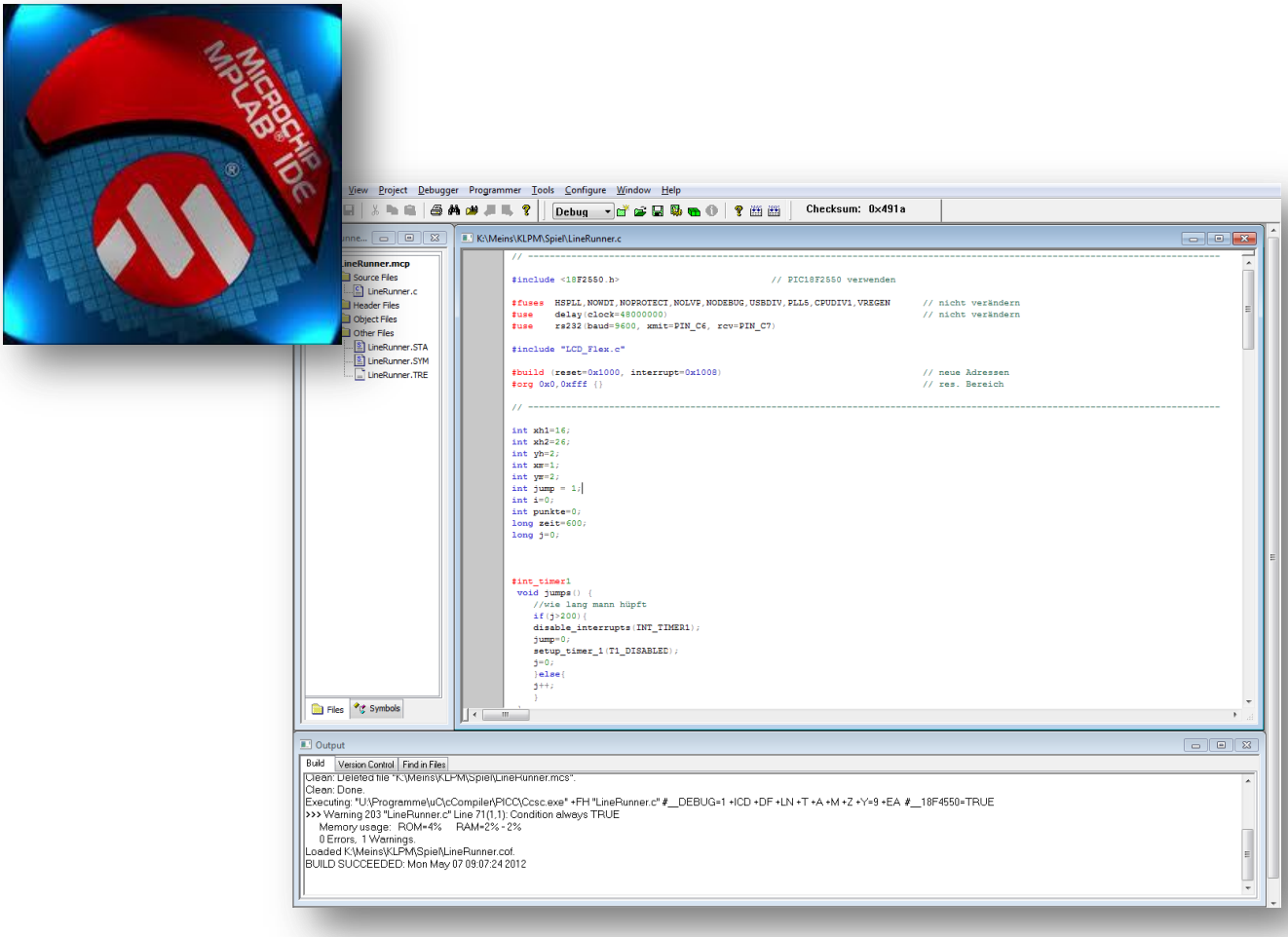
2.1.4 Verwendete Hard- und Software:

Die verwendeten Hardwarekomponenten :	MyPIC Board LCD Display Steckboard Nunchuck
Die verwendeten Softwarekomponenten :	MPLAB








MICROCHIP
MPLAB® IDE
View Project Debugger Programmer Tools Configure Window Help
Debug
Checksum: 0x491a
K:\Meins\KLPM\Spiel\LineRunner.c
LineRunner.mcp
Source Files
LineRunner.c
Header Files
Object Files
Other Files
LineRunner.STA
LineRunner.SYM
LineRunner.TRE
Files
Symbols
// --
#include <18F2550.h> // PIC18F2550 verwenden
#fuses HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL5,CPUDIV1,VREGEN // nicht verändern
#use delay(clock=48000000) // nicht verändern
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)
#include "LCD_Flex.c"
#build (reset=0x1000, interrupt=0x1008) // neue Adressen
#org 0x0,0xfff {} // res. Bereich
// --
int xh1=16;
int xh2=26;
int yh=2;
int xm=1;
int ym=2;
int jump = 1;
int i=0;
int punkte=0;
long zeit=600;
long j=0;
#int_timer1
void jumps() {
//wie lang mann hüpft
if(j>200){
disable_interrupts(INT_TIMER1);
jump=0;
setup_timer_1(T1_DISABLED);
j=0;
}else{
j++;
}
Output
Build
Version Control
Find in Files
Clean: Deleted file "K:\Meins\KLPM\Spiel\LineRunner.mcs".
Clean: Done.
Executing: "U:\Programme\uC\cCompiler\PICC\Ccsc.exe" +FH "LineRunner.c" #__DEBUG=1 +ICD +DF +LN +T +A +M +Z +Y=9 +EA #__18F4550=TRUE
>>> Warning 203 "LineRunner.c" Line 71 (1,1): Condition always TRUE
Memory usage: ROM=4% RAM=2% - 2%
0 Errors, 1 Warnings.
Loaded K:\Meins\KLPM\Spiel\LineRunner.cof.
BUILD SUCCEEDED: Mon May 07 09:07:24 2012

2.1.5 Schaltung und/oder Programm:

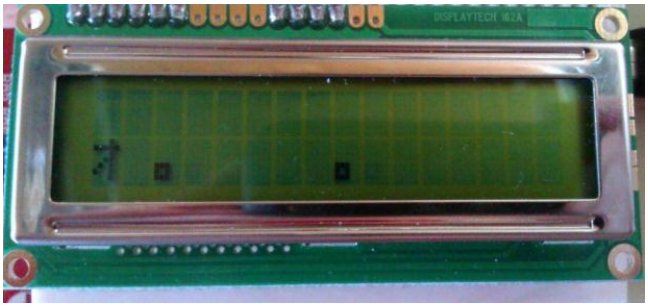
```
// ------------------------------------------------------------------------------------------------------------------------
#include <18F2550.h>                    // PIC18F2550 verwenden
#fuses  HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL5,CPUDIV1,VREGEN
#use    delay(clock=48000000)

#use    rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)
#use    I2C(master, sda=PIN_B7, scl=PIN_B6 )

#include <math.h>
#include <string.h>
#include <stdlib.h>
#include "LCD_Flex.c"

#build (reset=0x1000, interrupt=0x1008)
#org 0x0,0xfff {}
// ------------------------------------------------------------------------------------------------------------------------

// Variablendeklaration
int xh1=10;
int xh2=20;
int yh=2;
int xm=1;
int ym=2;

int jump = 1;                   //ob man hüpfen darf
int i=0;                        //Schleifenvariable für Level
int punkte=0;                   //Punkte
long zeit=600;                  //Geschwindigkeit
long j=0;                       //Schleifenvariable für Timer1
int level = 1;

int8 x, y;
int16 acc_x, acc_y, acc_z;
int8 button;
#bit z = button.0

//wie lang mann hüpfen darf
#int_timer1
 void jumps() {
          if(j>(zeit/3)){
                    disable_interrupts(INT_TIMER1);
                    jump=0;
                    setup_timer_1(T1_DISABLED);
                    j=0;
          }else{
                    j++;
          }
 }

void nunchuk()
{
    i2c_start();                        // Starten des I2C Bussystems
    i2c_write(0xA4);        // Adresse des Nunchuk
    i2c_write(0x40);
    i2c_write(0x00);
```

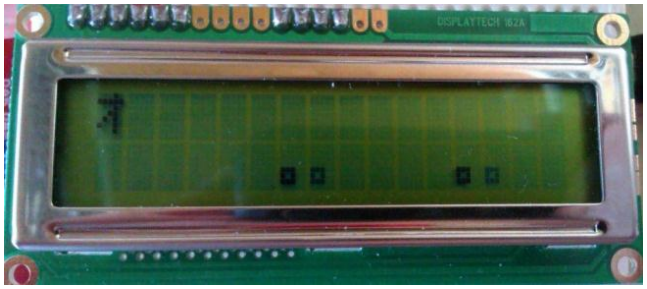
```
    i2c_stop();                         // Stoppen des Bussystems
    delay_ms(10);

    i2c_start();                        // Starten des I2C Bussystems
    i2c_write(0xA4);        // Adresse des Nunchuk
    i2c_write(0x00);
    i2c_stop();
    delay_ms(50);

    i2c_start();
    i2c_write(0xA5);        // Adressierung des Bausteins mit letztem Bit auf 1 („Lesen“)
    x      = i2c_read();
    y      = i2c_read();
    acc_x    = i2c_read();
    acc_y    = i2c_read();
    acc_z    = i2c_read();
    button    = i2c_read(0);
    i2c_stop();

    x =     (x     ^ 0x17) + 0x17;
    y =     (y     ^ 0x17) + 0x17;
    acc_x = ((acc_x  ^ 0x17) + 0x17)*4;
    acc_y = ((acc_y  ^ 0x17) + 0x17)*4;
    acc_z = ((acc_z  ^ 0x17) + 0x17)*4;
    button = (button ^ 0x17) + 0x17;

void main()                         // Begin main program
{

lcd_init();
setup_timer_1 (T1_INTERNAL);
delay_ms(10);

printf(lcd_putc,"  Line Runner   ");
delay_ms(2000);

enable_interrupts(global);
enable_interrupts(INT_TIMER1);

        while(1){

                // Hindernisse
                xh1=xh1-1;
                xh2=xh2-1;

                if(xh1<1){
                        xh1=16;
                }

                if(xh2<1){
                        xh2=16;
                }

                if(ym==2){
                        jump=1;
                }
```

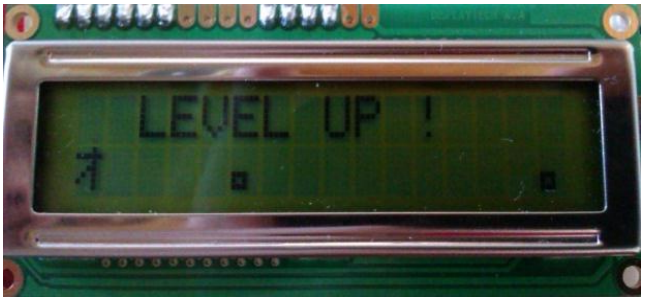
```
nunchuk();

if(z==0 && jump==1){
        enable_interrupts(INT_TIMER1);
        setup_timer_1 (T1_INTERNAL);
        set_timer1(0);
        ym=1;
}else{
        ym=2;
}

//Loser
if(xh1==xm||xh2==xm){
        if(ym==2){
                disable_interrupts(INT_TIMER1);
                        printf(lcd_putc,"\f");
                        lcd_gotoxy(1,1);
                        printf(lcd_putc,"     LOSER ");
                        lcd_gotoxy(1,2);
                        printf(lcd_putc,"L: %u / P: %u",level, punkte);
                enable_interrupts(INT_TIMER1);
                i = 0;
                punkte = 0;
                level = 1;
                zeit=600;
                delay_ms(1000);
        }else{
        punkte = punkte+10;

        }
}


//Level Up
if(i>(12000/zeit)){
        level++;
        disable_interrupts(INT_TIMER1);
                lcd_gotoxy(3,1);
                printf(lcd_putc,"Level %u",level);
        enable_interrupts(INT_TIMER1);
        i = 0;
        punkte = punkte+5;
        zeit = zeit-100;
        delay_ms(500);
}

//Winner
if(zeit<100){
        disable_interrupts(INT_TIMER1);
                printf(lcd_putc,"\f");
                lcd_gotoxy(1,1);
                printf(lcd_putc,"! WINNER !");
                lcd_gotoxy(1,2);
                printf(lcd_putc,"L: %u / P: %u",level, punkte);
        enable_interrupts(INT_TIMER1);
        i=0;
```

```
                punkte=0;
                level = 1;
                zeit=600;
                delay_ms(1000);
        }



        //LCD schreiben
        disable_interrupts(INT_TIMER1);

                printf(lcd_putc,"\f");

                lcd_gotoxy(xm,ym);
                printf(lcd_putc,"µ");

                lcd_gotoxy(xh1,yh);
                printf(lcd_putc,"¡");

                lcd_gotoxy(xh2,yh);
                printf(lcd_putc,"¡");

                lcd_gotoxy(13,1);
                printf(lcd_putc,"L: %u",level);

        enable_interrupts(INT_TIMER1);

        i++;
        delay_ms(zeit);

    }

}                       // End of main
```

2.1.6 Bilder:







2.1.7 Probleme, Lösungen:

Während der Dauer des Projekts traten einige Probleme auf.

Zu nennen wären ein Problem mit dem Prescaler, für die Begrenzung der Sprungzeit, das mit Hilfe einer Schleifenvariable in einer If – Bedingung behoben wurde. Ein weiterer Fehler trat auf, als wir eine Projektverbesserung , die Steuerung der Figur mittels Nunchuck ins Programm implementierten.

Der Fehler bestand in einer Verzögerung nach dem Tastendruck des Nunchuck, der ausgelöst wurde durch eine zu seltenen Abfrage des Timers , den die Methode, die kontrolliert ob der Taster am Nunchuck gedrückt ist, verwendet. Dies erschwerte erheblich ein kontrolliertes Abspringen.

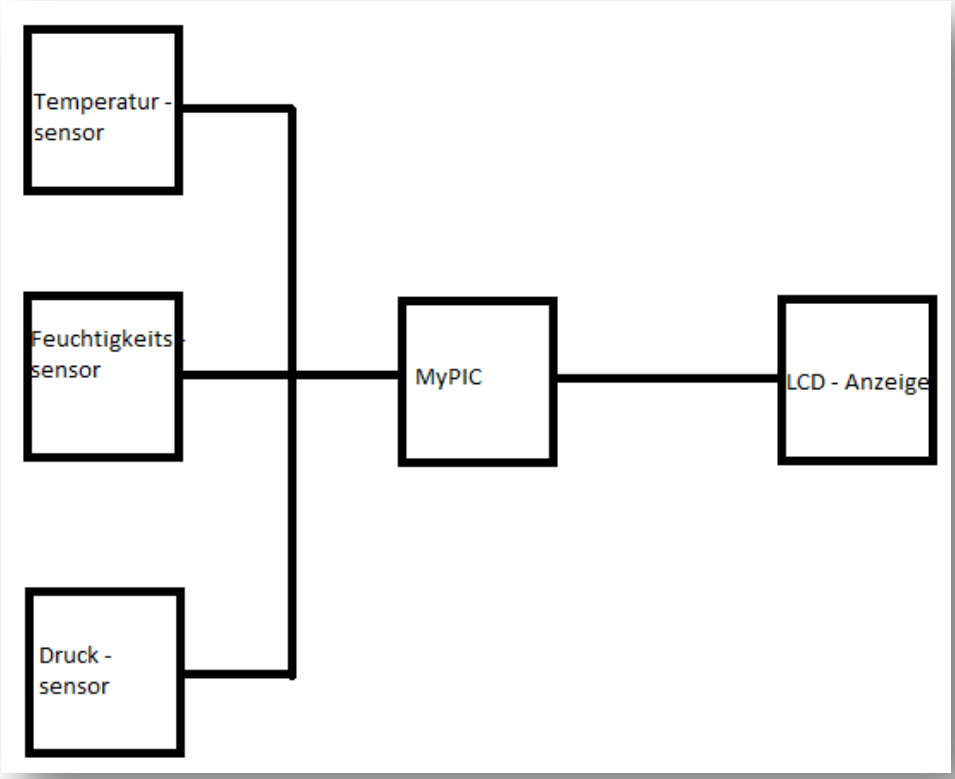
2.2 Anhang BSP2

2.2.1 Mini-Wetterstation

2.2.2 Idee, Ziel (Aufgabenstellung):

Ziel des Projektes war eine Mini-Wetterstation zu kreieren, die die Temperatur, die Luftfeuchtigkeit und den Luftdruck angibt.

2.2.3 Planung, Blockschaltbild:



2.2.4 Verwendete Hard- und Software:

Temperatursensor TMP37

Luftfeuchtigkeitssensor HIH-4000

Luftdrucksensor 24PCCFA6G

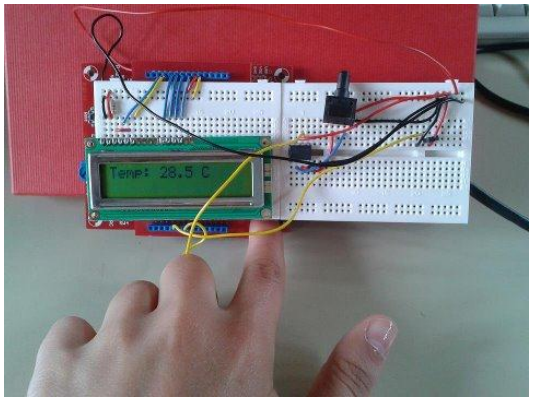
MyPIC

2.2.5 Schaltung und/oder Programm:

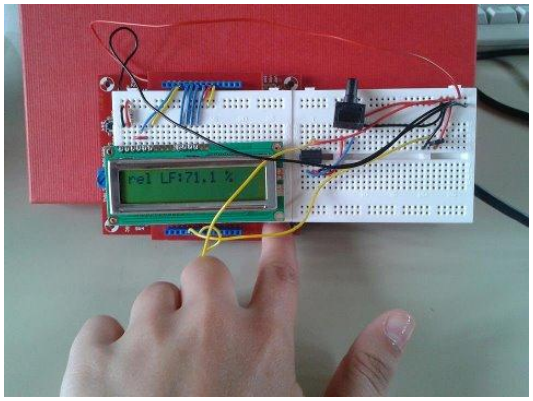
```
void main()  {

                    lcd_init(); // LCD Anzeige wird angegeben
                    delay_ms(10); //warten

                    setup_adc_ports(an0_to_an4); //welche ADC Ports ver-
                    wendet werden
                    setup_adc(adc_clock_div_32); // ADC Zeit angeben
                    delay_ms(1);
                    float temp; //definieren der Variablen
                    float tempaus;
                    float feu;
                    float dru;
                    float feuaus;
                    float feuaus_2;
                    float druaus;

                        while(TRUE) {
                         set_adc_channel(1);  // channel auswählen
                        delay_ms(1);
                    temp = (float)read_adc()*(float)(5.0/1024.0*25.0/0.5);
                    printf(lcd_putc,"\fTemp: %4.1f C",temp);
                    // Berechnung und ausgeben der Temperatur
                    delay_ms(3000);

                set_adc_channel(3);
                delay_ms(1);
                feu =read_adc();
                feuaus = (float)(feu*(float)(5.0/1024.0));
                feuaus_2 = ((feuaus/(5.0*0.0062))-0.16);

                printf(lcd_putc,"\frel LF:%4.1f %% ",feuaus_2);
                delay_ms(3000);

                    set_adc_channel(4);
                    delay_ms(1);
                    dru = read_adc();
                    druaus   =   (float)(dru*(5.0/1024.0*1.034/0.255)*0.07);

                    printf(lcd_putc,"\fDruck:%4.1f bar",druaus);
                    delay_ms(3000);
```

2.2.6 Bilder:

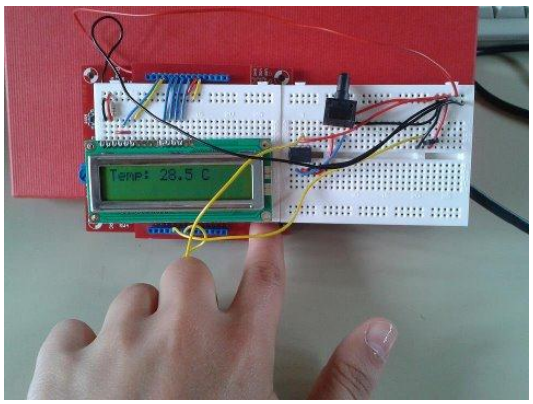
Temperatur:



Luftfeuchtigkeit:



Luftdruck:



2.2.7 Probleme, Lösungen:

.) Fehler in der Schaltung

.) Mehrfaches Austauschen des ICs

.) Neuladen des Bootloaders

.) Umrechnung der digitalen Info in Temperatur, Luftfeuchtigkeit und Luftdruck

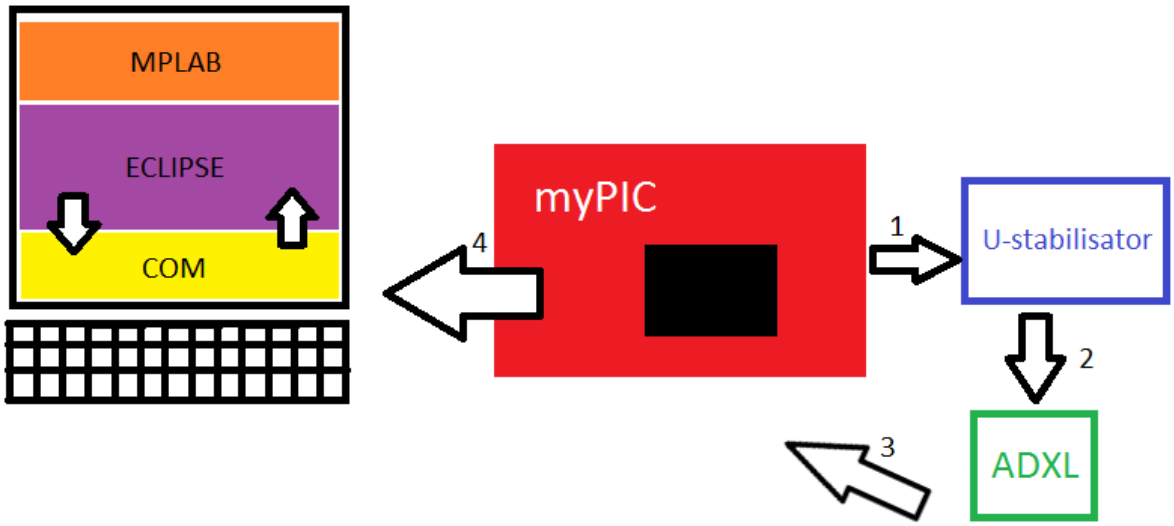
2.3 Anhang BSP3

2.3.1 Tür-Auf-Statistik

2.3.2 Idee, Ziel (Aufgabenstellung):

Mit Hilfe unseres MyPic (18F2550) und eines externen Beschleunigungssensors mitzählen wie oft eine Tür geöffnet wird und die aktuelle Zeit ausgeben.

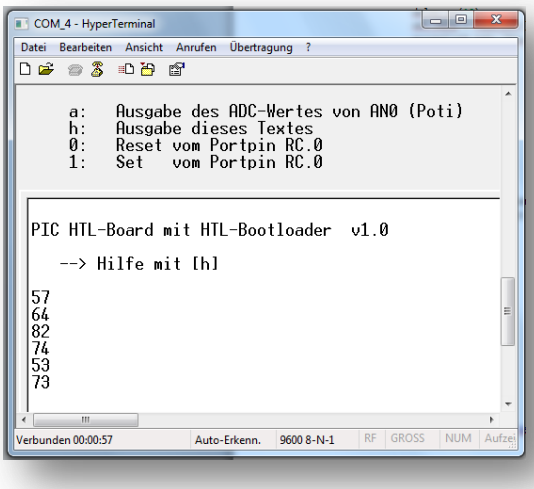
2.3.3 Planung, Blockschaltbild:



2.3.4 Verwendete Hard- und Software:

- MyPic (18F2550)
- 2 dimensionaler Beschleunigungssensor (mit dazugehöriger Schaltung)
  x-Achse wird verwendet
- Spannungsstabilisator

2.3.5 Schaltung und/oder Programm:



```
#include <usb_cdc.h>

// globale Variable
int8      In_Char;
int16     Adc_Data;

//ADC Einstellungen
void HW_Init()
{
  // Init des ADCs
  setup_adc_ports( AN0_TO_AN4 );                    // Ref=Vdd
  setup_adc(adc_clock_div_32);
}

//Verbindung
void usb_putc(char cc)
{
        while(!usb_cdc_putready());
        delay_ms(1);
        usb_cdc_putc(cc);
}

//Ein/Ausgabe USB
void rtUsb()
{
   if (usb_cdc_kbhit())                                 // Zeichen empfangen?
   {
        In_Char=usb_cdc_getc();                         // Zeichen einlesen
        If (In_Char == 'a')
                {
                set_adc_channel(3);                     // Adc starten
                delay_us(10);                           // Warten
                Adc_Data = read_adc();                  // Auslesen
                printf(usb_putc,"%lu \r\n",Adc_Data);   // Ausgabe
                }
   }
}

void main()
{
HW_Init();                                              // Init Hardware
usb_init();
        while(!usb_cdc_connected());                    // Warte auf PC-USB-Verbindung
        // Begruessung
        printf(usb_putc,"\r\n\PIC HTL-Board mit HTL-Bootloader  v1.0 \r\n\n");
        printf(usb_putc,"   --> Hilfe mit [h] \n\n\r");

         while (1)
         {
                rtUsb();                                // Usb-Verbindung
         }
}
```

Zu 5.

→ include HTL Libery „SerielleSchnittstelle“

```
StartRequest sR = new StartRequest(serielleSchnittstelle, 20, "a");
sR.start();

serielleSchnittstelle.addStringListener(new StringListener() {
	public void stringReceived(StringEvent arg0) {
		try {
			counter(Integer.parseInt(arg0.getStringReceived().trim()));
tArea.setText(arg0.getStringReceived());
el.addValue(Integer.parseInt(arg0.getStringReceived()
					.trim()));
		}
		catch (Exception e) {
		}
	}
});
```

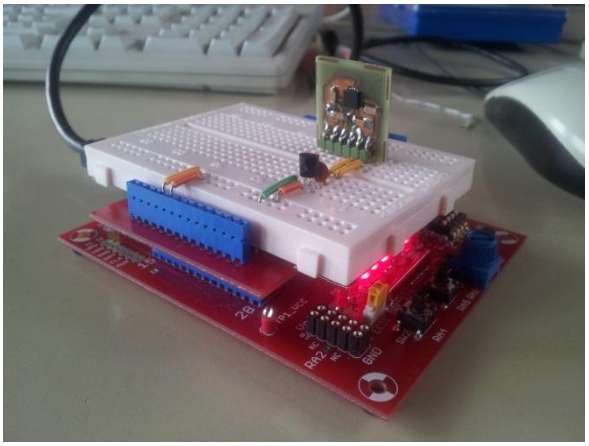
```
// y-Achse
g2d.drawLine(28, 30, 28, this.getHeight() - 30);
// x-Achse
g2d.drawLine(28, this.getHeight()-30, this.getWidth()-30, this.getHeight()-30);

// Nullpunkt nach Koordinatennullpunkt verschieben
g2d.translate(30, this.getHeight() - 30);

for (int i = 1; i < aL.size(); i++) {
	int vor = aL.get(i - 1);
	int nach = aL.get(i);
	int weiterrück = this.getWidth() - 60;

	g2d.setColor(Color.black);
	g2d.fillRect((weiterrück-5) / 500 * (i) - (i / 500) * 500, -150,
		(weiterrück-5) / 500 * (i+1) - (i / 500) * 500,this.getHeight()-100);

	g2d.setColor(Color.red);
	g2d.drawLine(weiterrück / 500 * (i - 1) - (i / 500) * 500, -vor,
		weiterrück / 500 * (i) - (i / 500) * 500, -nach);
}
```

```
private void counter(int wert) {
	if (wert > 95) {
		peak = 1;
	}
	if (peak == 1 && wert < 80) {
		peak = 0;
		counter++;
		Calendar c = new GregorianCalendar();

		timeArea.append(counter + " " + c.getTime() + "\n");
	}
	textAreaZähler.setText("Die Tür wurde: " + counter + " geöffnet.\r\n");
```

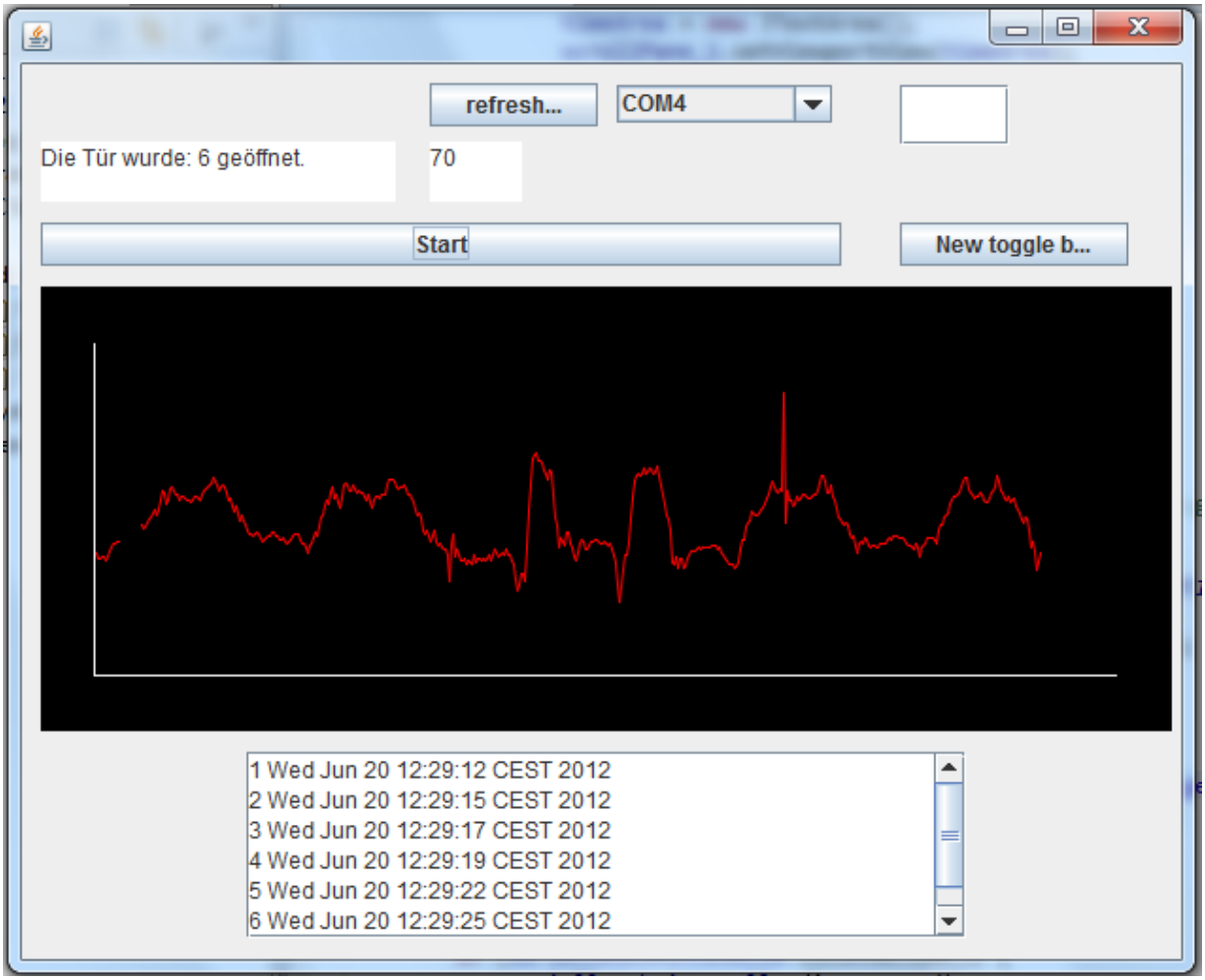
}

2.3.6 Bilder:

Beschleunigungssensor:



Auswertung der Bewegungen:



2.3.7 Probleme, Lösungen:

Beschleunigungssensor 3V, Pic liefert 5V

➔ Spannungsstabilisator

Eclipse Verbindung zu allen ADC-Channels

➔ Java Klasse StartRequest erhalten

Werte grafisch ausgeben:

Koordinatensystem für die Erleichterung der Berechnung verschoben

➔ Java Befehl: translate

Wenn die Werte das Ende des Panels erreicht haben, Panel neu zeichnen

➔ Formel entwickelt, die genau besagt in welchem Bereich die Werte gezeichnet werden sollen

➔ Schwarzer Balken wird vor jeweiligen Wert gezeichnet

Durchgehend die auf Sekunden genaue Zeit ausgeben, bei der sich die Tür öffnet

➔ Gregorian Kalender nicht bei den Standardvereinbarungen definieren, sondern immer neu generieren

Ungelöstes Problem:

Mittels Toggle Buttons den aktuellen Graph anhalten.
Auf verschiedene Arten versucht:
mittels boolean Variable Zustand setzen