

HÖHERE TECHNISCHE BUNDESLEHRANSTALT HOLLABRUNN

Höhere Abteilung für Elektronik – Technische Informatik

Klasse/ Jahrgang: 5BHEL	Gruppe: 3	Übungsleiter: Dipl. Ing. Mag. Wihsböck
Übungsnummer: I/6	Übungstitel: Raspberry Pi	
Datum der Übung: 22.09.2015	Teilnehmer: Holzmann, Nemecek, Weixelbraun, Hartmann	
Datum der Abgabe: 06.10.2015	Schritfführer: Hartmann	Unterschrift:

	Beurteilung
Aufgabenstellung	
Dokumentation	
Messschaltungen	
Messtabellen	
Berechnungen	
Programmlistings	
Auswertung	
Diagramme	
Berechnungen	
Simulationen	
Schlußfolgerungen	
Kommentare	
Inventarliste	
Messprotokoll	
Form	
Summe	

1. Allgemeines

Titel der Übung: Raspberry Pi
Übungsnummer: I/6
Übungsplatz:
Datum der Übung: 22.09.2015
Klasse: 5BHEL
Schriftführer: Hartmann Lorenz
Übungsteilnehmer: Holzmann Philip
Nemecek Thomas
Weixelbraun Patrick
Hartmann Lorenz

Inhaltsverzeichnis

1. Allgemeines	2
2. Raspberry Pi Typ B	3
2.1. Allgemeines	3
2.2. Programmiersprachen	3
2.3. Aufbau	3
3. Berryclip IO-Board	3
4. Ampelsteuerung	4
4.1 Aufgabenstellung	4
4.2 Programm in C	4
4.3 Programm in Bash	5
4.4. Makefile	7
5. Temperaturmessung	8
5.1. Aufgabenstellung	8
5.2. Programm in C	8
5.3. Programm in Bash	10
6. Benchmark „Toggle GPIO-Pin“	12
6.1. Aufgabenstellung	12
6.2. Programm in C	12
7. Erkenntnisse	13

2. Raspberry Pi Typ B

2.1. Allgemeines

Für diese Übung wurde ein Raspberry Pi Typ B benutzt der einige Schnittstellen wie zum Beispiel HDMI, USB, Ethernet, GPIOs, SPI, I2C und UART besitzt. Außerdem ist er sehr preiswert und hat ein reichhaltiges Angebot was Zubehör betrifft. Ein Raspberry Pi wird vor allem für einfache Aufgaben eingesetzt wie zum Beispiel für Steueraufgaben oder als E-mail Server.

2.2. Programmiersprachen

Um Programme für den Raspberry Pi zu schreiben werden normalerweise Scriptsprachen wie Bash oder Python aber auch Programmiersprachen wie C, C++ und Java benutzt. Die Programme bei der Übung wurden in C und in Bash geschrieben um die Vor- und Nachteile dieser Sprachen zu erkennen.

Um eine einfache Konfigurierung der GPIO Ports in C zu ermöglichen wurde die WiringPi Library benutzt.

2.3. Aufbau

Der Raspberry Pi wurde mit einem DVI Kabel an den vorhandenen Monitor, der sich am Übungsplatz befindet und an eine Tastatur angeschlossen. Außerdem wurden noch eine zweite Arbeitsstation mitsamt Computer und Monitor aufgestellt da immer zwei Leute gemeinsam mit einem Raspberry Pi an einer Aufgabe arbeiteten. Desweiteren wurde der Raspberry Pi mit einem Laptop über Putty verbunden um einen Zugriff zu gewähren. WinSCP wurde auch noch auf den Laptops gestartet um eine einfache Datenübertragung zu ermöglichen. Zuletzt wurde noch Linux Debian von einer SD-Karte, die uns der Übungsleiter gab, gebootet.

3. Berryclip IO-Board

Das Berryclip IO-Board ist eine einfache Aufsteckkarte auf der sich 6 LEDs, 1 Summer und 1 Schalter befinden. Es wird mit den GPIO-Schnittstellen an den Raspberry Pi angeschlossen

Berryclip IO-Board GPIO-Zuordnung:

IO	Hardware-PIN	GPIO ID (für „gpio“ utility)	WiringPi ID (für C-Programm)
LED 1	Pin 7	4	7
LED 2	Pin 11	17	0
LED 3	Pin 15	22	3
LED 4	Pin 19	10	12
LED 5	Pin 21	9	13
LED 6	Pin 23	11	14
Buzzer	Pin 24	8	10
Button	Pin 26	7	11

4. Ampelsteuerung

4.1 Aufgabenstellung

Es soll eine Ampelsteuerung realisiert werden, welche eine Verkehrsampel für KFZ in einer definierten Sequenz auf ROT schaltet, wenn ein Fußgänger eine Taste bestätigt um den Straßenübertritt gefahrlos zu ermöglichen.

Der Ablauf sieht im Detail folgendermaßen aus:

1. Ausgangszustand: Die Ampel ist GRÜN
2. Wird die Taste gedrückt, soll die Ampel zunächst GRÜN BLINKEN (3x, EIN/AUS-Zeit je 0,5 sec)
3. Danach soll die Ampel zwei Sekunden GELB leuchten.
4. Anschließend soll die Ampel für einen Zeitraum von sechs Sekunden ROT sein. Falls die Taste beim Bestätigen mindestens drei Sekunden gehalten wurde, soll zunächst der Buzzer während der Rotphase aktiv sein.
5. Nach der Rotphase ist die Ampel für drei Sekunden GELB.
6. Zurück zum Ausgangszustand

Zuerst soll das Programm in C, danach in Bash geschrieben werden.

4.2 Programm in C

```
//  
// Example C program for a Raspberry PI equipped with a Berryclip IO board  
//  
// Author: Hartmann Lorenz  
// Date: 22.9.2015  
//  
  
#include <wiringPi.h>  
#include <stdlib.h>  
#include <stdio.h>  
  
//  
// IO definitions with WiringPI identifiers  
//  
int LED1=7, LED2=0, LED3=3, LED4=12, LED5=13, LED6=14;  
int BUZZER = 10;  
int BUTTON = 11;  
int flag=0;  
  
//  
// Main program  
//  
int main (void)  
{  
    pinMode(BUTTON, INPUT);  
    pinMode(LED1, OUTPUT);  
    pinMode(LED3, OUTPUT);  
    pinMode(LED5, OUTPUT);
```

```

//Check we have wiringPi
if (wiringPiSetup () == -1 )
{
    exit (1);
}

for (;;)
{
    digitalWrite (LED6, 1);    //grüne LED wird eingeschalten
    if( digitalRead(BUTTON)==1){    //wenn der Button von einem Fußgänger
                                    //gedrückt wird startet die Ampelsteuerung

        delay(3000);
        flag=0;

        if(digitalRead(BUTTON)==1){ //wenn der Button drei Sekunden gedrückt
                                    //wird, wird ein flag auf 1 gesetzt
            flag=1;
        }
        digitalWrite(LED6, 1); //grüne LED blinkt dreimal
        delay(500);
        digitalWrite(LED6, 0);
        delay(500);
        digitalWrite(LED6, 1);
        delay(500);
        digitalWrite(LED6, 0);
        delay(500);
        digitalWrite(LED6, 1);
        delay(500);
        digitalWrite(LED6, 0);
        delay(500);
        digitalWrite(LED4, 1); //gelbe LED leuchtet für 2 Sekunden
        delay(2000);
        digitalWrite(LED4, 0); //gelbe LED wird ausgeschalten
        digitalWrite(LED1, 1); //rote LED wird für 6 Sekunden eingeschalten
        if(flag==1){ //der Buzzer wird aktiviert falls flag gesetzt ist
                    //(statt dem Buzzer wurde eine LED verwendet wegen Lärmgründen)
            digitalWrite(LED2, 1); //Buzzer ist aktiv (in diesem Fall die LED)
        }
        delay(6000);
        digitalWrite(LED2, 0);
        digitalWrite(LED1, 0);
        digitalWrite(LED4, 1); //nach der Rotphase wird die gelbe LED für drei
                                //Sekunden eingeschalten

        delay(3000);
        digitalWrite(LED4, 0);
        digitalWrite (LED6, 1); //Ausgangszustand
    }
}
exit(0);
}

```

4.3 Programm in Bash

```

#!/bin/bash

#
# Example bash script for a Raspberry PI equipped with a Berryclip IO board
#
# Author: Hartmann Lorenz
# Date: 22.9.2015
#

#####
# Variables
#####

```

```

# LED numbers given as GPIO identifiers
LED1=4
LED2=17
LED3=22
LED4=10
LED5=9
LED6=11

ALLEDS=( $LED1 $LED2 $LED3 $LED3 $LED4 $LED5 $LED6 )

# Input button as GPIO identifier
BUTTON=7

# Buzzer as GPIO identifier
BUZZER=8

#####
# Wait until the button is pressed
#####
waitForButton ()
{
    echo "Waiting for button ..."
    while [ `gpio -g read $BUTTON` = 0 ]; do
        sleep 0.1
    done
}

#####
# The main program
#####
    gpio -g mode $LED1 output
    gpio -g mode $LED4 output
    gpio -g mode $LED6 output
    gpio -g mode $BUTTON input
    gpio -g mode $LED2 output
while true; do

    gpio -g write $LED6 1 #grüne LED wird eingeschalten

    if [ `gpio -g read $BUTTON` = 1 ]; then #wenn der Button von einem
        #Fußgänger gedrückt wird startet die Ampelsteuerung
    sleep 3
    flag=0
        if [ `gpio -g read $BUTTON` = 1 ]; then #wenn der Button drei Sekunden
            #gedrückt wird, wird ein flag auf 1 gesetzt
        flag=1
        fi

        gpio -g write $LED6 1 #grüne LED blinkt dreimal
        sleep 0.5
        gpio -g write $LED6 0
        sleep 0.5
        gpio -g write $LED6 1
        sleep 0.5
        gpio -g write $LED6 0
        sleep 0.5
        gpio -g write $LED6 1
        sleep 0.5
        gpio -g write $LED6 0
        sleep 0.5
        gpio -g write $LED4 1 #gelbe LED leuchtet für 2 Sekunden
        sleep 2

```

```

gpio -g write $LED4 0    #gelbe LED wird ausgeschalten
gpio -g write $LED1 1    #rote LED wird für 6 Sekunden eingeschalten

if [ $flag = 1 ]; then #der Buzzer wird aktiviert falls flag gesetzt
    #ist (statt dem Buzzer wurde eine LED verwendet wegen Lärmgründen)
    gpio -g write $LED2 1
fi
sleep 6
gpio -g write $LED2 0
gpio -g write $LED1 0
gpio -g write $LED4 1    #nach der Rotphase wird die gelbe LED für drei
                          #Sekunden eingeschalten

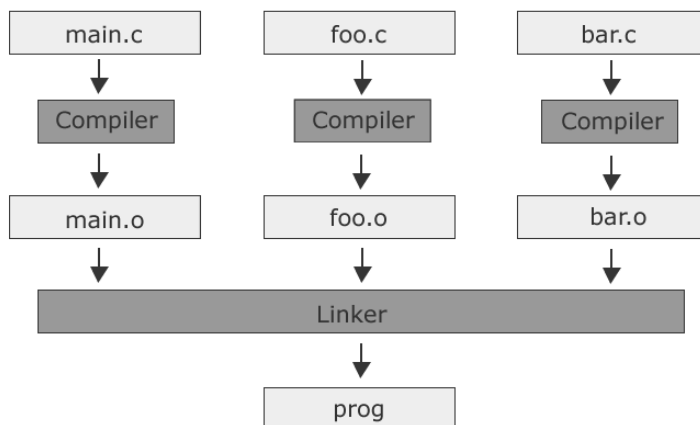
sleep 3
gpio -g write $LED4 0    #Ausgangszustand
fi

done

```

4.4. Makefile

Das Kompilieren von Quelltext-Dateien kann mit dem Programm make gesteuert werden. Dies ist vor allem dann sinnvoll, wenn es sich um ein großes Projekt handelt bei dem die Kompilierung aus vielen einzelnen Schritten und Abhängigkeiten besteht. Die einzelnen Schritte können zum Beispiel separate Kompilierungen oder andere Dateioperationen wie Kopieren und Löschen sein. Abhängigkeiten können zum Beispiel bedingte Kompilierungen sein. Diese Schritte und Abhängigkeiten werden in ein Makefile zusammengefasst, welches dann vom Programm make verarbeitet wird.



Hier sieht man die Verwendung von Makefiles, mehrere Dateien tragen zum kompilierten Programm bei. Quelltext-Dateien werden vom Compiler in Objektdateien bzw. Programmmodule übersetzt. Der Linker verbindet die Programmmodule zu einem Programm.

Makefile für die Ampelsteuerung:

```

DEBUG = -O3
CC     = gcc
INCLUDE = -I/usr/local/include
CFLAGS = $(DEBUG) -Wall -std=c99 $(INCLUDE) -Winline -pipe

LDFLAGS = -L/usr/local/lib
LIBS    = -lwiringPi

```

```
OBJ = Ampelsteuerung.o

BINS = Ampelsteuerung
Ampelsteuerung: Ampelsteuerung.o
    @echo [link]
    $(CC) -o $@ Ampelsteuerung.o $(LDFLAGS) $(LIBS)

.c.o:
    @echo [CC] $<
    @$ (CC) -c $(CFLAGS) $< -o $@

clean:
    rm -f $(OBJ) *~ Ampelsteuerung
```

5. Temperaturmessung

5.1. Aufgabenstellung

Die LEDs des BerryClip-IO-Board sollen als Temperaturskala für die Prozessortemperatur verwendet werden. Pro 10 °C soll eine LED leuchte, also z.B. bei 43°C sind die beiden grünen und die beiden gelben LEDs aktiv.

Das Programm soll wieder in C und Bash geschrieben werden.

5.2. Programm in C

```
//
// Example C program for a Raspberry PI equipped with a Berryclip IO board
//
// Author: Hartmann Lorenz
// Date: 22.9.2015
//

#include <wiringPi.h>
#include <stdlib.h>
#include <stdio.h>

//
// IO definitions with WiringPI identifiers
//
int LED1=7, LED2=0, LED3=3, LED4=12, LED5=13, LED6=14;
int BUZZER = 10;
int BUTTON = 11;
int flag=0;

//
// Main program
//
int main (void)
{
    FILE * fp;          //Filepointer
    char temp[60];       //string in den die Temperatur geschrieben wird
    //Check we have wiringPi
```

```
if (wiringPiSetup () == -1 )
{
    exit (1);
}

//LED werden als OUTPUT definiert
pinMode(LED1, OUTPUT);
pinMode(LED2, OUTPUT);
pinMode(LED3, OUTPUT);
pinMode(LED4, OUTPUT);
pinMode(LED5, OUTPUT);
pinMode(LED6, OUTPUT);

for (;;)
{
    fp = fopen ("/sys/class/thermal/thermal_zone0/temp", "r"); //File öffnen
    if(fp == NULL)
    {
        perror("Error opening file");
        return(-1);
    }

    fgets (temp, 60, fp); //Temperaturwert in temp schreiben
    puts(temp);           //Ausgabe der Prozessortemperatur
    delay (1000);

    //Temperaturanzeige mit LEDs
    if(temp[0]=='1')
    {
        digitalWrite (LED6, 1);
        digitalWrite (LED5, 0);
        digitalWrite (LED4, 0);
        digitalWrite (LED3, 0);
        digitalWrite (LED2, 0);
        digitalWrite (LED1, 0);
    }
    if(temp[0]=='2')
    {
        digitalWrite (LED6, 1);
        digitalWrite (LED5, 1);
        digitalWrite (LED4, 0);
        digitalWrite (LED3, 0);
        digitalWrite (LED2, 0);
        digitalWrite (LED1, 0);
    }
    if(temp[0]=='3')
    {
        digitalWrite (LED6, 1);
        digitalWrite (LED5, 1);
        digitalWrite (LED4, 1);
        digitalWrite (LED3, 0);
        digitalWrite (LED2, 0);
        digitalWrite (LED1, 0);
    }
    if(temp[0]=='4')
    {
        digitalWrite (LED6, 1);
        digitalWrite (LED5, 1);
        digitalWrite (LED4, 1);
        digitalWrite (LED3, 1);
        digitalWrite (LED2, 0);
        digitalWrite (LED1, 0);
    }
    if(temp[0]=='5')
    {
        digitalWrite (LED6, 1);
        digitalWrite (LED5, 1);
        digitalWrite (LED4, 1);
        digitalWrite (LED3, 1);
    }
}
```

```

        digitalWrite (LED2, 1);
        digitalWrite (LED1, 0);
    }
    if(temp[0]=='6')
    {
        digitalWrite (LED6, 1);
        digitalWrite (LED5, 1);
        digitalWrite (LED4, 1);
        digitalWrite (LED3, 1);
        digitalWrite (LED2, 1);
        digitalWrite (LED1, 1);
    }
    fclose(fp);

}
exit(0);
}

```

Die Prozessortemperatur findet man in der virtuellen Datei: „sys/class/thermal/thermal_zone0/temp“

Man kann sich den Inhalt dieser Datei auch mit dem Befehl cat im Consolenfenster anzeigen lassen
z.B. cat /sys/class/thermal/thermal_zone0/temp

Ausgabe: 44654

Bei der Übung funktionierte das Programm nicht beim ersten Versuch da die wiringPi Bibliothek nicht am Programmfang stand.

5.3. Programm in Bash

```

#!/bin/bash
# LED numbers given as GPIO identifiers
LED1=4
LED2=17
LED3=22
LED4=10
LED5=9
LED6=11
ALLLEDS=( $LED1 $LED2 $LED3 $LED3 $LED4 $LED5 $LED6 )
# Input button as GPIO identifier
BUTTON=7
# Buzzer as GPIO identifier
BUZZER=8
#####
# The main program
#####
    gpio -g mode $LED1 output
    gpio -g mode $LED2 output
    gpio -g mode $LED3 output
    gpio -g mode $LED4 output
    gpio -g mode $LED5 output
    gpio -g mode $LED6 output
while true; do
    temp=`cat /sys/class/thermal/thermal_zone0/temp` #der Rückgabewert von cat
                                                    #wird in temp gespeichert
    echo $temp          #Ausgabe der Prozessortemperatur
    sleep 1

    #Temperaturanzeige mit LEDs

```

```
if(temp[0]=='1')
{
    gpio -g write $LED6 1
    gpio -g write $LED5 0
    gpio -g write $LED4 0
    gpio -g write $LED3 0
    gpio -g write $LED2 0
    gpio -g write $LED1 0
}
if(temp[0]=='2')
{
    gpio -g write $LED6 1
    gpio -g write $LED5 1
    gpio -g write $LED4 0
    gpio -g write $LED3 0
    gpio -g write $LED2 0
    gpio -g write $LED1 0
}
if(temp[0]=='3')
{
    gpio -g write $LED6 1
    gpio -g write $LED5 1
    gpio -g write $LED4 1
    gpio -g write $LED3 0
    gpio -g write $LED2 0
    gpio -g write $LED1 0
}
if(temp[0]=='4')
{
    gpio -g write $LED6 1
    gpio -g write $LED5 1
    gpio -g write $LED4 1
    gpio -g write $LED3 1
    gpio -g write $LED2 0
    gpio -g write $LED1 0
}
if(temp[0]=='5')
{
    gpio -g write $LED6 1
    gpio -g write $LED5 1
    gpio -g write $LED4 1
    gpio -g write $LED3 1
    gpio -g write $LED2 1
    gpio -g write $LED1 0
}
if [temp[0]=='6']
{
    gpio -g write $LED6 1
    gpio -g write $LED5 1
    gpio -g write $LED4 1
    gpio -g write $LED3 1
    gpio -g write $LED2 1
    gpio -g write $LED1 1
}
done
```

Screenshot der Ausgabe:

```

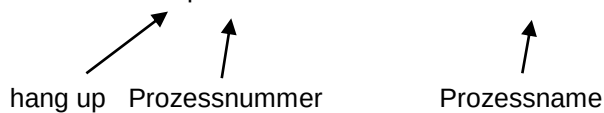
root@raspberrypi:/home/pi/Laboruebung/C# sudo ./Temp
45464
44925
44925
44925
44925
44388
45464
44388

```

Die Prozessortemperatur ist in milligrad Celsius angegeben.

Um die Temperatur zu erhöhen kann man Endlosschleifen in weiteren Consolenfenstern öffnen. Wenn man die Prozesse wieder beenden möchte muss man den Befehl kill benutzen.

z.B. `kill -hup 4254` oder `sudo killall a.out`



Um die Prozessnummer zu finden wurde der Befehl ps verwendet, dieser listet alle aktuellen Prozesse auf.

6. Benchmark „Toggle GPIO-Pin“

6.1. Aufgabenstellung

Es sollte ein Programm geschrieben werden, welches einen Pin toggeln lässt. Danach soll an diesem Pin die Frequenz gemessen werden. Anschließend sollten die Unterschiede zwischen C und Bash erkannt werden. Wegen Zeitmangels konnte diese Aufgabe nicht mehr vollständig erfüllt werden.

6.2. Programm in C

```

#include <wiringPi.h>
#include <stdlib.h>
#include <stdio.h>

int LED1 = 7;

int main (void)
{
    if (wiringPiSetup () == -1 )
    {
        exit (1);
    }
    pinMode(LED1, OUTPUT);
}

```

```
for (;;)
{
    digitalWrite(LED1,0); //LED1 für 0,1 Sekunden einschalten
    delay(100);
    digitalWrite(LED1,1); //LED1 für 0,1 Sekunden ausschalten
    delay(100);
}
exit(0);
}
```

Das Programm wurde nur mehr in C geschrieben und der Pin mit dem Oszilloskop gemessen, jedoch wurde kein Screenshot von der Messung gemacht da keine Zeit mehr vorhanden war.

7. Erkenntnisse

Bei dieser Übung wurde uns gezeigt wie man einfache Aufgaben mit einem Raspberry Pi lösen kann. Außerdem wurden uns die Unterschiede zwischen einer Scriptsprache und Programmiersprache gezeigt.