Raspberry Pi Projekt Hygrometer

Autor:
Martin Strohmayer
m.stroh@ymail.com
http://evil.hn.vc

C-Programme:
Manfred Wallner
email@mwallner.net
http://mwallner.net

Version 2.1
23. April 2015

Inhaltsverzeichnis

Vorwort

In dieser Anleitung wird mit wenigen elektrischen Bauteileilen ein Hygrometer mit der Raspberry Pi erstellt. Der Aufbau kann die Lufttemperatur und die Luftfeuchte mithilfe des Sensors DHT22/AM2302 messen und verarbeiten. Die Messwerte werden über eine zweistelligen LED-Siebensegmentanzeige ausgegeben.
Die C-Programmierung des Sensors und der Anzeige bzw. der GPIOs der Raspberry Pi erfolgt mithilfe der wiringPi Bibliothek.

Vorraussetzung für den Aufbau ist eine betriebsbereite Raspberry Pi Model B+ oder Raspberry Pi 2 Model B mit installierter wiringPi Bibliothek. Das System muss über eine bekannte IP-Adresse per SSH erreichbar sein.
Für den Aufbau der elektrischen Schaltung wird ein Steckplatine (Breadboard) sowie Kabel für die Verbindungen benötigt. Weiters werden alle Bauteile entsprechend der Bauteillisten von Kapitel 2 Sensor DHT22/AM2302 und Kapitel 3 7-Segmentanzeige benötigt.

1. Entwicklungsumgebung

1.1. Windows Programme

1.1.1. PuTTY (SSH-Client)

Download und Installation von PuTTY:
`http://the.earth.li/~sgtatham/putty/0.64/x86/putty-0.64-installer.exe`

Zuerst muss bei Terminal/Keyboard „The Function keys and keypad“ auf Linux gesetzt werden. Dann sollte unter Terminal/Bell die akustische Warnfunktion (Action to happen when a bell occurs) abgeschaltet werden, indem man in den Einstellungen „None (bell disabled)“ auswählt. Die Zeichenkodierung muss unter Window/Translation auf UTF-8 gesetzt werden. Bei Connection/SSH/X11 kann „Enable X11 forwarding“ eingeschaltet werden. Dann können auch grafische Programme gestartet werden, wenn ein X-Server am lokalen System verfügbar ist.
Unter Session können dann die Verbindungsdaten eingerichtet werden:
Host Name (or IP address): 192.168.2.1
Connection type: SSH
Port: 22
Bei „Saved Sessions“ kann ein beliebiger Name wie z. B. „192.168.2.1 - Raspberry Pi“ eingetragen werden. Mit der Taste „Save“ werden nun alle Einstellungen unter diesem Namen gespeichert. Mit der Taste „Load“ können die Einstellungen wieder geladen werden. Nach dem Drücken der Taste „Open“ wird die Verbindung aufgebaut und man muss Benutzername und Passwort eingeben.
Username: pi
Passwort: raspberry
Nun können direkt Befehl eingeben und Programme im Terminal gestartet werden. Als Editor kann das Programm „nano“ verwendet werden.

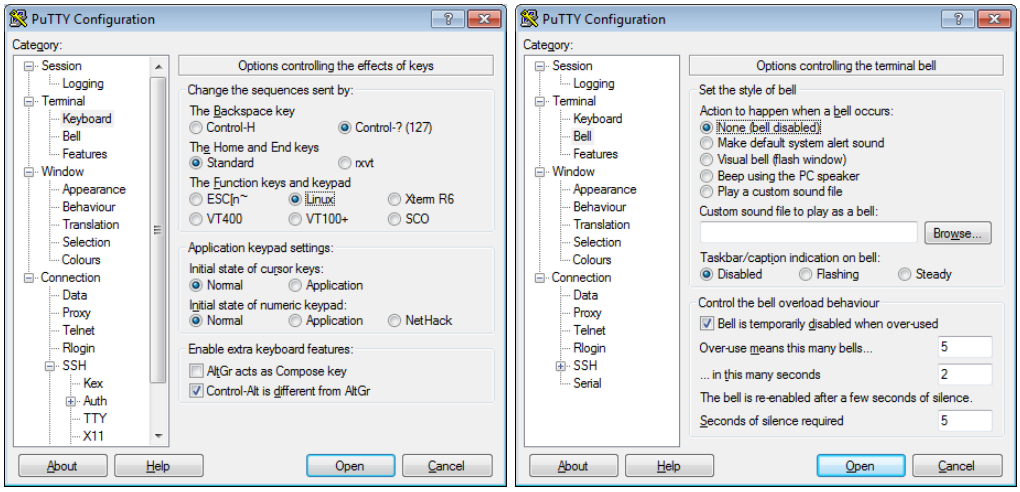


Abbildung 1.1: PuTTY - Keyboard und PuTTY - Bell (Warnsignal)

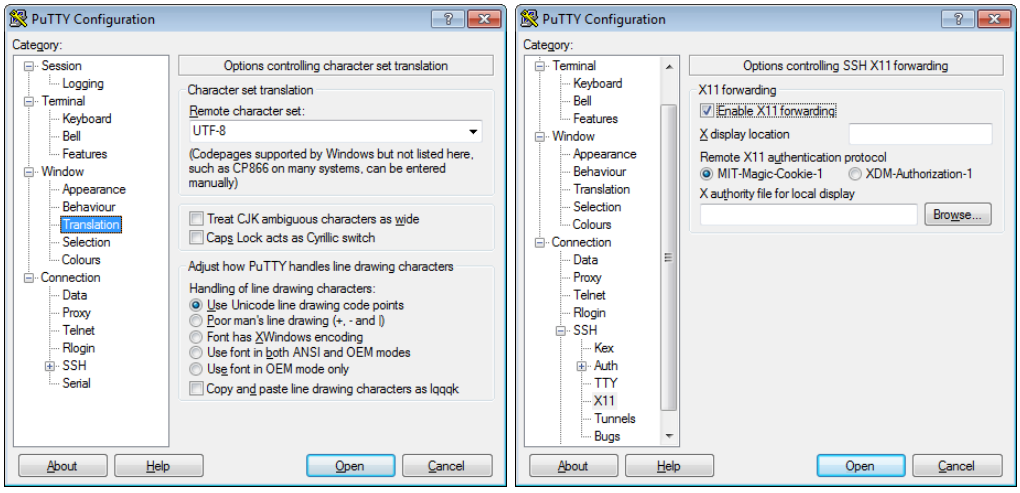


Abbildung 1.2: PuTTY - Translation (Zeichenkodierung) und PuTTY - SSH/X11

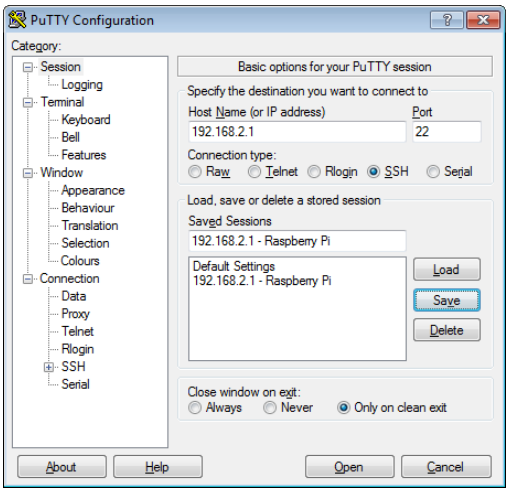


Abbildung 1.3: PuTTY - Session

1.1.2. Notepad++ (C-Editor)

Download und Installation von Notepad++:
`http://download.tuxfamily.org/notepadplus/6.7.4/npp.6.7.4.Installer.exe`

Zuerst muss die Erweiterung nppFTP aktiviert werden. Sie ermöglicht, dass Daten auf anderen System via FTP oder SFTP (SSH) transferieren und bearbeiten werden können. Dann muss ein neues Profil mit dem Namen „Raspberry Pi“ erstellt werden.
Im Profil Einstellungfenster können die Verbindungsdaten eingegeben werden:
Hostname: 192.168.2.1
Connection type: SFTP
Port: 22
Username: pi
Passwort: raspberry
Danach kann mit der Connect-Taste das Profil ausgewählt werden und die Verbindung hergestellt werden. Nun können verschiedene Funktionen ausgeführt werden wenn man das Menü mit der rechten Maustaste in einem selektierten Verzeichnis auswählt. Es kann z. B. eine neu Datei erzeugt werden oder eine bestehende Datei kann transferiert werden.

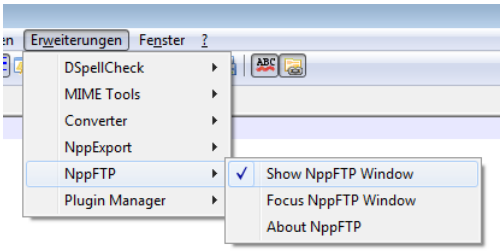


Abbildung 1.4: Erweiterung nppFTP aktivieren

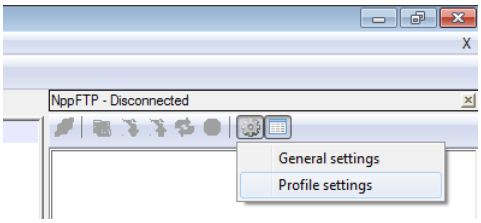


Abbildung 1.5: „Profile settings“ der Erweiterung aufrufen

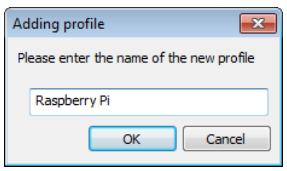


Abbildung 1.6: Profil „Raspberry Pi“ hinzufügen

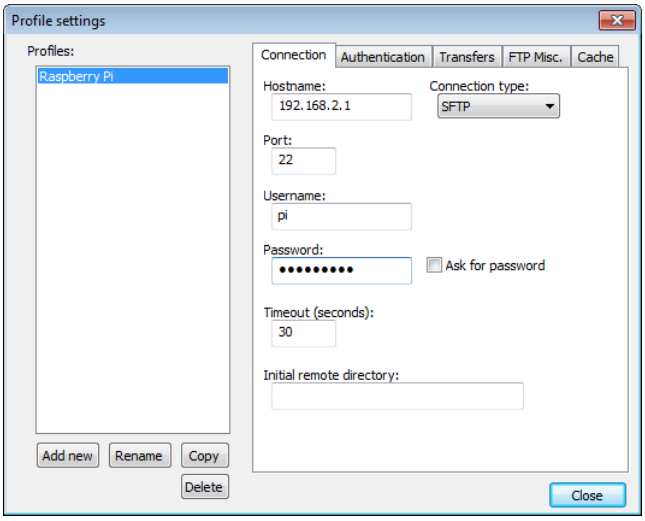


Abbildung 1.7: Profil „Raspberry Pi“ konfigurieren

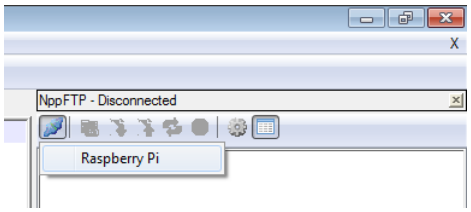


Abbildung 1.8: Verbindung mit Profil „Raspberry Pi“ aufbauen

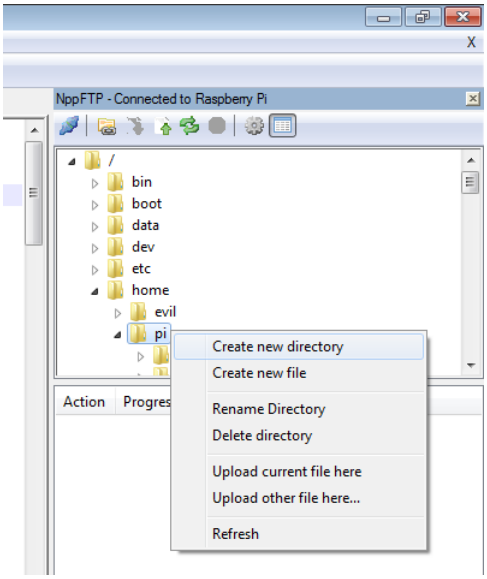


Abbildung 1.9: Funktionen

1.1.3. Xming (X-Server für Windows)

Ein X-Server ermöglicht, dass grafische Programme über das Netzwerk am lokalen PC angezeigt werden, obwohl sie auf einem entfernten System (Raspberry Pi) laufen.
Download und Installation von Xming:
`http://sourceforge.net/projects/xming/files/latest/download`
Installation entsprechend dem Installations-Assistent duchführen und danach das Programm Xming starten.

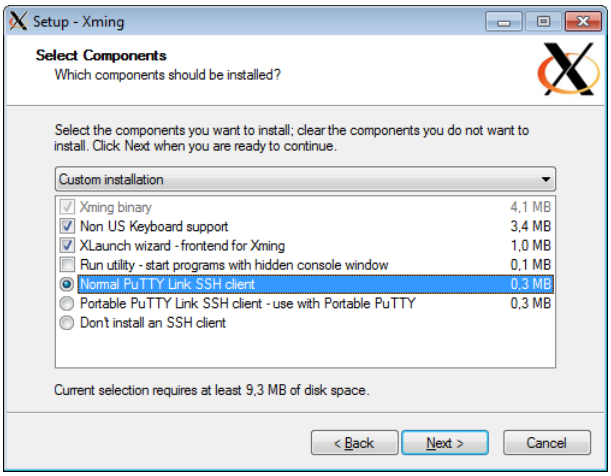


Abbildung 1.10: Xming Installation

1.2. Linux Programme

1.2.1. SSH-Client

```
apt-get install ssh

ssh -X 192.168.2.1
```

1.2.2. X-Server

Damit der X-Server läuft, muss die grafischen Oberfläche gestartet werden. Wenn dies nicht nach dem Bootvorgang automatisch erfolgt, muss der Befehl „startx“ ausgeführt werden.

1.3. Geany - Entwicklungsumgebung

Nach dem Herstellen der Verbindung über SSH, kann die grafische Entwicklungsumgebung Geany installiert und gestartet werden.

```
sudo apt-get install geany
geany &
```

Soll Geany in Englisch ausgeführt werden, so muss man vor dem Start die Variable „LANG“ auf „C“ setzen.

```
LANG=C geany &
```

Nach dem Start kann eine C-Datei geladen werden, z. B. hygrometer.c. Dann kann das Konfigurationsfenster geöffnet werden, indem man im Menü unter `Erstellen` → `Kommandos zum Erstellen konfigurieren` auswählt.
Nun kann bei `Kompilieren` und `Erstellen` die Bibliothek WiringPi mit dem Parameter „-lwiringPi“ hinzugefügt werden. Bei `Ausführen` muss am Beginn „sudo“ hinzugefügt werden.
Danach kann das Programm mit der Taste `Erstellen` auf der Raspberry Pi erzeugt werden und mit der Taste `Ausführen` auch gestartet werden.
Die Tastenkürzel für alle Funktionen können über das Menü `Bearbeiten` → `Einstellungen` → `Tastenkürzel` vorgegeben werden. Dazu wählt man die gewünschte Aktion aus und drückt die Taste „Ändern“. Danach kann man die Taste bzw. Tastenkombination drücken, die dann umgehend im Dialog angezeigt wird. Wenn nun die OK-Taste gedrückt wird, wird die Änderung übernommen und das Tastenkürzel führt in Zukunft die Aktion aus. Im Beispiel wurde der Aktion „Erstellen“ dem Tastenkürzel bzw. der Taste F7 zugewiesen.

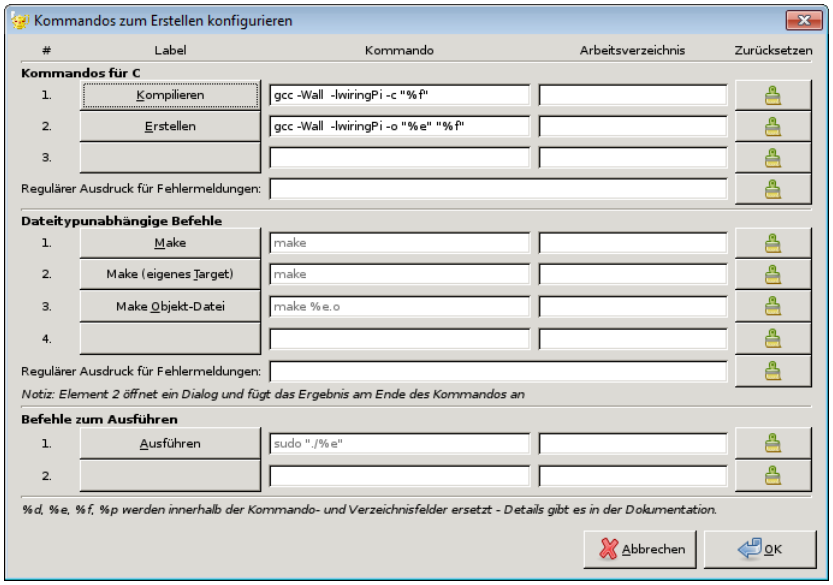


Abbildung 1.11: Kommandos zum Erstellen konfigurieren

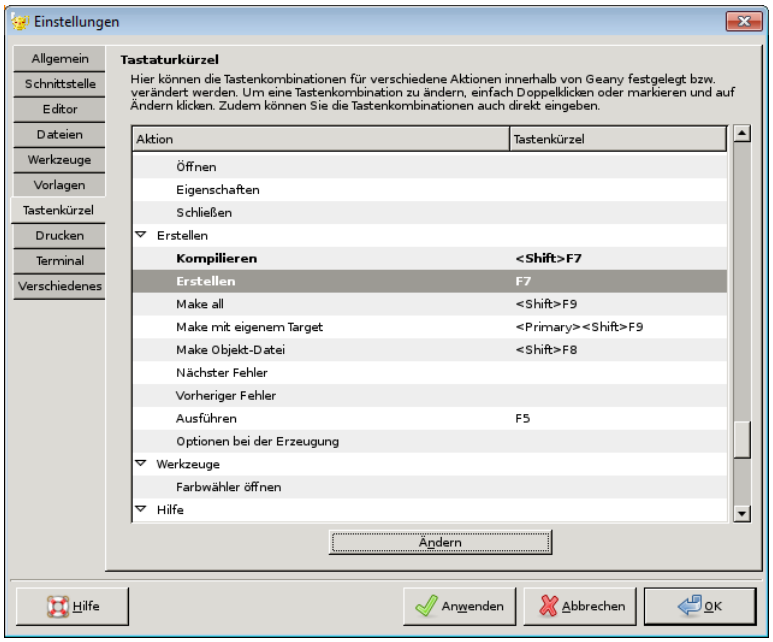


Abbildung 1.12: Tastaturkürzel

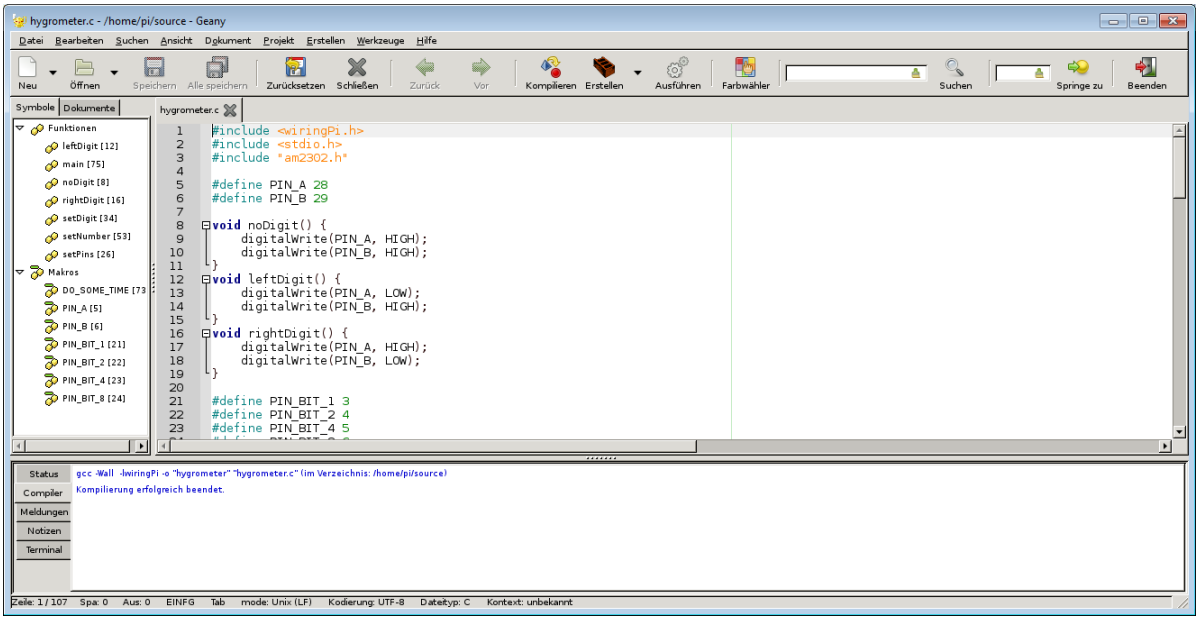


Abbildung 1.13: Geany Oberfläche

2. Sensor DHT22/AM2302

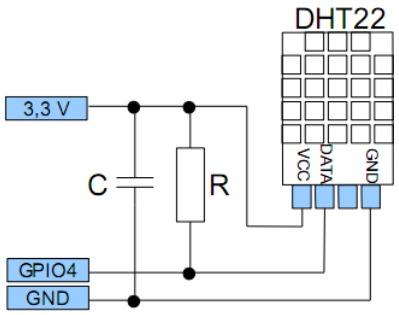


Abbildung 2.1: Luftfeuchte DHT22/AM2302 an GPIO Ein-/Ausgang

Bauteilliste:

- 1 × DHT22 - Temperatur-und Feuchtigkeitssensor
  Datenblatt: `https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf`
  Bezugsquelle: `http://www.ebay.de/sch/i.html?_nkw=DHT22`
- 1 × Pullup-Widerstand (6,8 kΩ oder 10 kΩ)
- 1 × Stützkondensator 100 nF (optional)

Luftfeuchte-/Temperatursensoren DHT22 bzw. AM2302 benutzt ein Kommunikationsprotokoll, das nur eine Signalleitung benötigt. Für diese Signalleitung kann eine GPIO Leitung der Raspberry Pi verwendet werden. Ein Pullup-Widerstand von 5 kΩ bis 10 kΩ muss zwischen der Betriebsspannung und dem DATA-Pin bzw. dem GPIO-Pin geschaltet werden. Optional kann man noch einen 100 nF Kondensator zum Stützen der Betriebsspannung zwischen VCC und GND schalten. Zur Kommunikation bzw. zum Auslesen der Sensordaten benötigt man ein Programm, dass das einfache Protokoll des Sensors implementiert hat.

Listing 2.1: 'am2302.h' C Source Code

```
#include <stdint.h>
#include <sys/types.h>
#include <unistd.h>

static uint8_t sizecvt(const int read)
{
  return (uint8_t)read;
}
```

```
static int read_am2302(const int dhtpin, const int nMaxTimings, float* humidity,
    float* temperature)
{

  static int dht22_dat[5] = {0,0,0,0,0};

  uint8_t laststate = HIGH;
  uint8_t counter = 0;
  uint8_t j = 0, i;

  dht22_dat[0] = dht22_dat[1] = dht22_dat[2] = dht22_dat[3] = dht22_dat[4] = 0;

  // pull pin down for 18 milliseconds
  pinMode(dhtpin, OUTPUT);
  digitalWrite(dhtpin, HIGH);
  delay(10);
  digitalWrite(dhtpin, LOW);
  delay(18);
  // then pull it up for 40 microseconds
  digitalWrite(dhtpin, HIGH);
  delayMicroseconds(40);
  // prepare to read the pin
  pinMode(dhtpin, INPUT);

  // detect change and read data
  for ( i=0; i< nMaxTimings; i++) {
    counter = 0;
    while (sizecvt(digitalRead(dhtpin)) == laststate) {
      counter++;
      delayMicroseconds(1);
      if (counter == 255) {
        break;
      }
    }
    laststate = sizecvt(digitalRead(dhtpin));

    if (counter == 255) break;

    // ignore first 3 transitions
    if ((i >= 4) && (i%2 == 0)) {
      // shove each bit into the storage bytes
      dht22_dat[j/8] <<= 1;
      if (counter > 16)
        dht22_dat[j/8] |= 1;
      j++;
    }
  }

  // check we read 40 bits (8bit x 5 ) + verify checksum in the last byte
  // print it out if data is good
  if ((j >= 40) &&
      (dht22_dat[4] == ((dht22_dat[0] + dht22_dat[1] + dht22_dat[2] + dht22_dat
          [3]) & 0xFF)) ) {
        float t, h;
        h = (float)dht22_dat[0] * 256 + (float)dht22_dat[1];
        h /= 10;
        t = (float)(dht22_dat[2] & 0x7F)* 256 + (float)dht22_dat[3];
        t /= 10.0;
        if ((dht22_dat[2] & 0x80) != 0)  t *= -1;

    *humidity = h;
```

```
    *temperature = t;
    return 1;
  }
  else
  {
    return 0;
  }
}
```

3. 7-Segmentanzeige

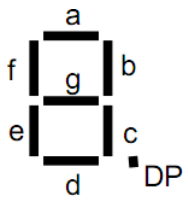


Abbildung 3.1: Elemente einer 7-Segmentanzeige

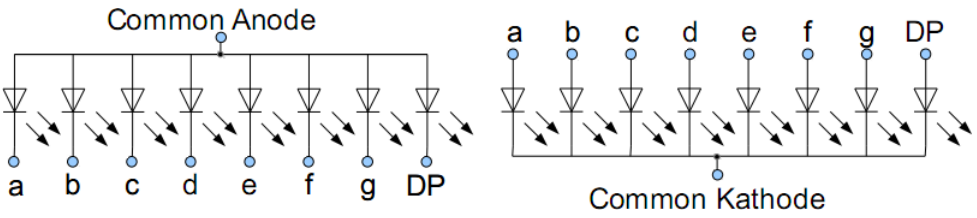


Abbildung 3.2: Mögliche LED Verschaltung der 7-Segmentanzeige

Bei der 7-Segmentanzeige hat jedes Segment bzw. LED einen eigenen und einen gemeinsamen Anschluss. Je nach Ausführung gibt es entweder eine gemeinsame Anode (Anschluss +) oder Kathode (Anschluss -). Bei gemeinsamer Anode (Anschluss +) werden die LEDs dann über ein „Low“-Signal aktiviert, bei gemeinsamer Kathode (Anschluss -) mit einem „High“-Signal.

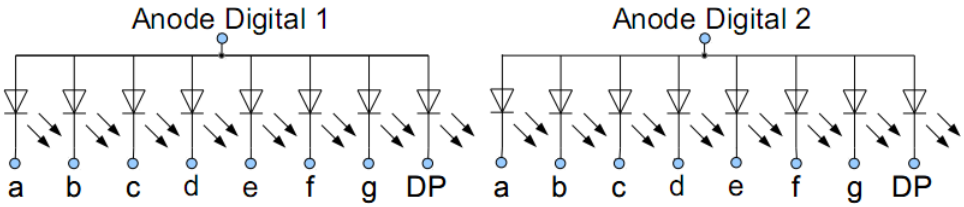


Abbildung 3.3: Zweistellige Anzeige mit mehrfachen Anschluss

Verfügt eine 7-Segmentanzeige über mehrere Stellen, so gibt es zwei Möglichkeiten. Eine Möglichkeit ist es, dass jedes Segment einen eigenen Anschluss hat, allerdings benötigt man dann die doppelte Anzahl an Ausgängen.

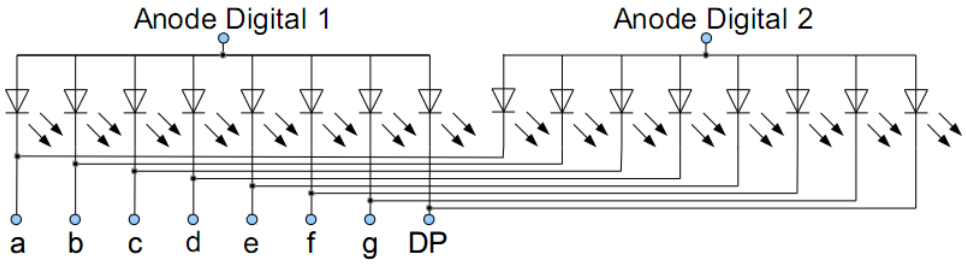


Abbildung 3.4: Zweistellige Anzeige mit gemeinsamen Anschluss

Die andere Möglichkeit ist, dass die Anschlüsse der Segmente verbunden sind. Wenn man also ein Segment aktiviert, so werden beide Segmente der Stellen aktiviert. Es gibt allerdings noch für jede Stelle eine eigene Versorgung (gemeinsame Anode oder gemeinsame Kathode). Will man nun eine Stelle aktivieren, so muss man die Versorgung der anderen Stelle abschalten. Dadurch kann aber immer nur eine Stelle aktiv sein bzw. leuchten. Wenn man nun einen zweistelligen Wert darstellen will, so muss man schnell zwischen den einzelnen Stellen umschalten. Ab 100 Hz ist fürs Auge nur noch eine zweistellige Anzeige zu sehen und die eigentliche Umschaltung ist nicht erkennbar. Dieses Verfahren nennt man Multiplexverfahren. Es ist softwaretechnisch aufwendiger, benötigt aber weniger Ausgänge und Signalleitungen. Ein Nachteil ist, dass die Anzeige nur korrekt ist, wenn das Programm bzw. die Umschaltung läuft. Bei dem direkten Ansprechen aller Segmente kann das Programm beendet werden und die Ausgänge behalten ihre Zustände. Somit bleibt die letzte Anzeige erhalten.
Eine weitere Möglichkeit Signalleitungen zu reduzieren ist die Verwendung eines BCD-to-7Segment-Decoders. Dieser hat als Eingang vier digitale Signale, die binär die dezimalen Werte 0-9 übertragen können. Als Ausgang hat der IC 7 Signale, die direkt mit den Elementen bzw. LEDs einer 7-Segmentanzeige verbunden werden können. So werden statt sieben Signalen nur noch vier benötigt (eigentlich fünf wenn man noch den Punkt ansprechen will).
Nachteil des Verfahrens könnte sein, dass nur die Zahlen 0-9 (IC abhängig auch ein paar Buchstaben) angezeigt werden können. Ein Muster und Buchstaben wie C, E, d, r, o usw. können nicht angezeigt werden.
Der verwendete IC 4543 hat zusätzlich noch weitere Steuereingänge:
Blank: Alle Ausgänge werden abgeschaltet, auch wenn Signale am Eingang anliegen. Die Anzeige bleibt leer.
Phase: Invertierung der Ausgänge (wird benötigt für Umschaltung zwischen Active High und Active Low der Ausgänge)
Latch: Puffer für die Eingangsdaten (nur wenn High anliegt, werden die Eingänge übernommen und die Ausgänge gesetzt)

Vorgegebene Werte:

$I_f = 2{,}0\ mA$
$U_f = 2{,}1\ V$
$U_{CE(sat)} = 0{,}2\ V$
$h_{FE(sat)} \approx 30$

Berechnung und Auswahl Transistor:
$I_C = 7 \times I_f = 7 \times 0{,}002 = 14\ mA$

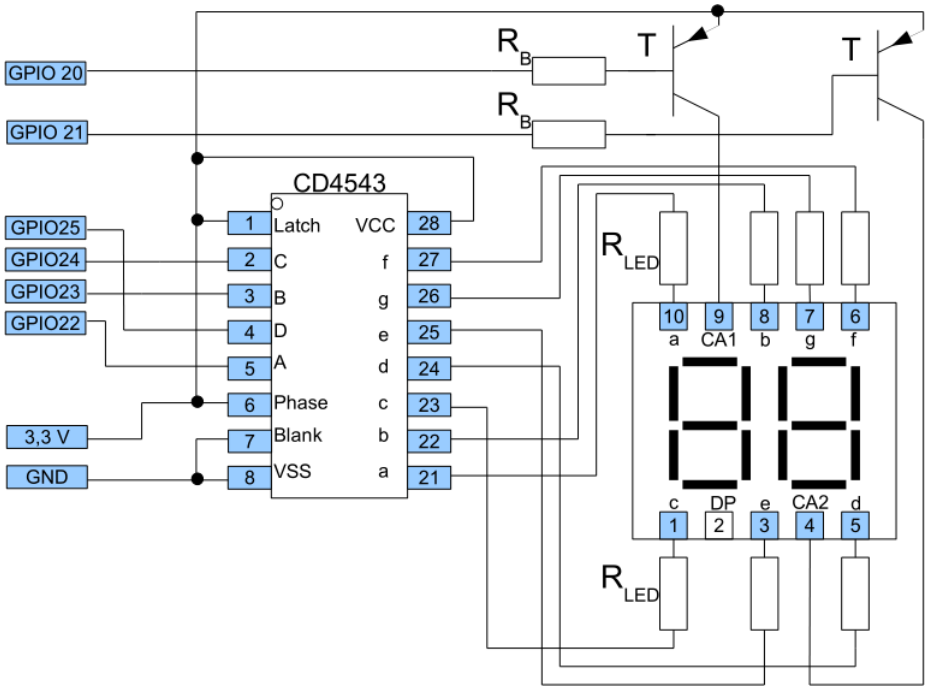


Abbildung 3.5: 7-Segmentanzeige (LTD-4608G) an BCD-to-7Segment-Decoder

$I_B = I_C / h_{FE(sat)} = 0,014/30 = 0,5\ mA$

Der gewählte PNP-Transistor BC307B bzw. BC557B hat mit 100 mA maximalem Kollektorstrom mehr als ausreichend Reserven. Die Verstärkung h_{FE} liegt laut Datenblatt bei über 100. Wird der Transistor in der Sättigung betrieben liegt sie aber niedriger. Es wurde 30 für die Verstärkung angenommen.
Als LED-Vorwiderstand R_{LED} wurde 100 Ω gewählt um nahezu den maximalen Strom des ICs CD4543 erreichen zu können (ca. 2 mA).

Berechnung Basiswiderstand Transistor:
$R_B = (U - U_{BE})/I_B = (3{,}3 - 0{,}7)/0{,}0005 = 5,2\ k\Omega$

Für R_B wurden Widerstände zwischen 5,1 $k\Omega$ und 6,8 $k\Omega$ gewählt, I_B verändert sich dadurch auf 0,4 mA bis 0,5 mA.

Bauteilliste:

- 1 × CD4543 - BCD to 7-segment
  Datenblatt: `http://www.circuitielettronici.it/CD4543.pdf`
  Bezugsquelle: `http://www.conrad.at/ce/de/product/173860/`
- 2 × PNP-Transistor BC307B oder BC557B
  Datenblatt BC307B: `http://www.onsemi.com/pub_link/Collateral/BC307-D.PDF`
  Datenblatt BC557B: `https://www.fairchildsemi.com/datasheets/BC/BC557.pdf`
  Bezugsquelle: `http://www.pollin.de/shop/dt/MTMzOTY4OTk-`

- 1 × Zweistellige 7-Segmentanzeige LITEON LTD-4608G
  Datenblatt: `http://www.datasheets360.com/pdf/-8277776035138596696`
  Bezugsquelle: `http://www.pollin.de/shop/dt/MDMwOTc4OTk-`
- 2 × Widerstand 5,1 oder 6,8 kΩ
- 7 × Widerstand 100 Ω

Listing 3.1: 'numbers.c' C Source Code

```
#include <wiringPi.h>

#define PIN_A 28
#define PIN_B 29

void noDigit() {
  digitalWrite(PIN_A, HIGH);
  digitalWrite(PIN_B, HIGH);
}
void leftDigit() {
  digitalWrite(PIN_A, LOW);
  digitalWrite(PIN_B, HIGH);
}
void rightDigit() {
  digitalWrite(PIN_A, HIGH);
  digitalWrite(PIN_B, LOW);
}

#define PIN_BIT_1 3
#define PIN_BIT_2 4
#define PIN_BIT_4 5
#define PIN_BIT_8 6

void setPins(int n8, int n4, int n2, int n1) {
  digitalWrite(PIN_BIT_1, n1);
  digitalWrite(PIN_BIT_2, n2);
  digitalWrite(PIN_BIT_4, n4);
  digitalWrite(PIN_BIT_8, n8);
}


void setDigit(int n) {
  switch(n) {
    case 0: setPins(0,0,0,0); break;
    case 1: setPins(0,0,0,1); break;
    case 2: setPins(0,0,1,0); break;
    case 3: setPins(0,0,1,1); break;
    case 4: setPins(0,1,0,0); break;
    case 5: setPins(0,1,0,1); break;
    case 6: setPins(0,1,1,0); break;
    case 7: setPins(0,1,1,1); break;
    case 8: setPins(1,0,0,0); break;
    case 9: setPins(1,0,0,1); break;
    default:
      setPins(0,0,0,0);
      break;
  }
}


void setNumber(int n) {
```

```
  int nFirst = n/10;
  int nLast = 0;
  if (n < 9) {
    nLast = n;
  } else {
    nLast = n%10;
  }
  noDigit();
  if (nFirst) {
    leftDigit();
    setDigit(nFirst);
    delay(5);
  }
  rightDigit();
  setDigit(nLast);
  delay(5);
}


int main (void)
{
  wiringPiSetup () ;
  pinMode (PIN_A, OUTPUT) ;
  pinMode (PIN_B, OUTPUT) ;
  pinMode (PIN_BIT_1, OUTPUT) ;
  pinMode (PIN_BIT_2, OUTPUT) ;
  pinMode (PIN_BIT_4, OUTPUT) ;
  pinMode (PIN_BIT_8, OUTPUT) ;

  int a = 99;
  for (;a>0;a--)
  {
  int x = 100;
  for (;x>=0;x--) {
    setNumber(a);
  }
  }
  return 0;
}
```

```
gcc -Wall -o numbers numbers.c -lwiringPi
sudo ./numbers
```

4. Gesamtschaltung

Fügt man Sensor und Anzeige zusammen erhält man ein Hygrometer für den Heimbetrieb.

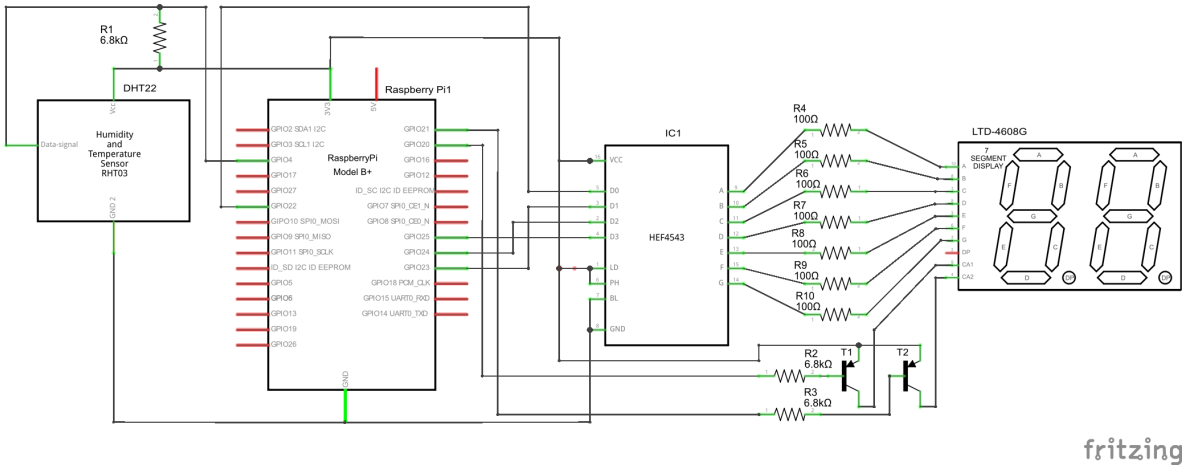


Abbildung 4.1: Schaltplan Hygrometer

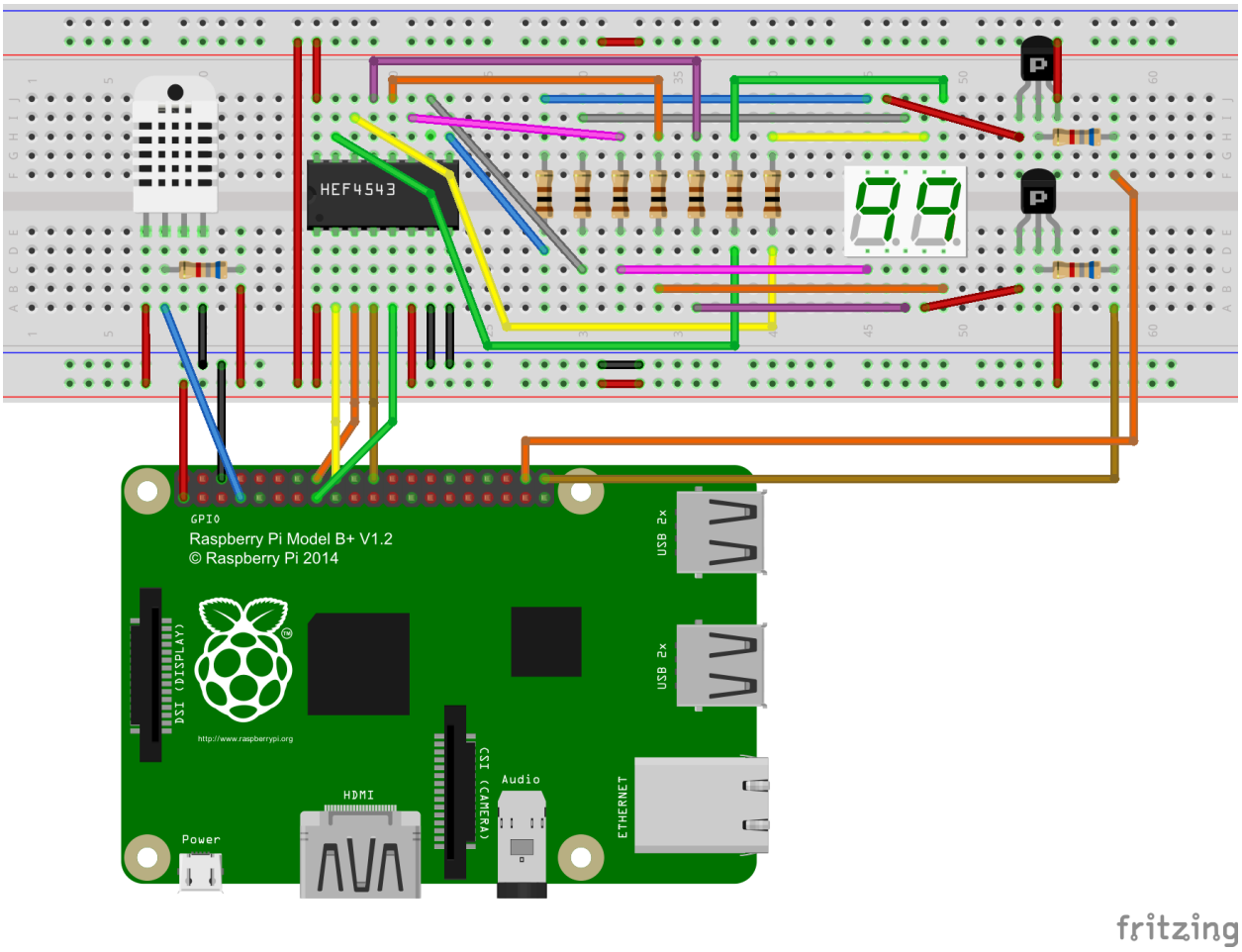


Abbildung 4.2: Steckbrett Hygrometer

Listing 4.1: 'hygrometer.c' C Source Code

```
#include <wiringPi.h>
#include <stdio.h>
#include "am2302.h"

#define PIN_A 28
#define PIN_B 29

void noDigit() {
  digitalWrite(PIN_A, HIGH);
  digitalWrite(PIN_B, HIGH);
}
void leftDigit() {
  digitalWrite(PIN_A, LOW);
  digitalWrite(PIN_B, HIGH);
}
void rightDigit() {
  digitalWrite(PIN_A, HIGH);
  digitalWrite(PIN_B, LOW);
}

#define PIN_BIT_1 3
#define PIN_BIT_2 4
#define PIN_BIT_4 5
#define PIN_BIT_8 6

void setPins(int n8, int n4, int n2, int n1) {
  digitalWrite(PIN_BIT_1, n1);
  digitalWrite(PIN_BIT_2, n2);
  digitalWrite(PIN_BIT_4, n4);
  digitalWrite(PIN_BIT_8, n8);
}


void setDigit(int n) {
  switch(n) {
    case 0: setPins(0,0,0,0); break;
    case 1: setPins(0,0,0,1); break;
    case 2: setPins(0,0,1,0); break;
    case 3: setPins(0,0,1,1); break;
    case 4: setPins(0,1,0,0); break;
    case 5: setPins(0,1,0,1); break;
    case 6: setPins(0,1,1,0); break;
    case 7: setPins(0,1,1,1); break;
    case 8: setPins(1,0,0,0); break;
    case 9: setPins(1,0,0,1); break;
    default:
      setPins(0,0,0,0);
      break;
  }
}


void setNumber(int n) {
  int nFirst = n/10;
  int nLast = 0;
  if (n < 9) {
    nLast = n;
  } else {
    nLast = n%10;
  }
```

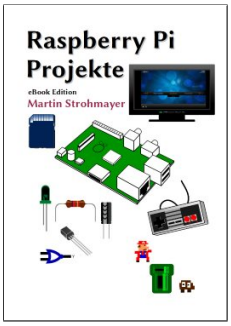
```
  noDigit();
  if (nFirst) {
    leftDigit();
    setDigit(nFirst);
    delay(5);
  }
  rightDigit();
  setDigit(nLast);
  delay(5);
}


#define DO_SOME_TIME for(x = 0;x<250;x++)

int main (void)
{
  wiringPiSetup();
  pinMode (PIN_A, OUTPUT) ;
  pinMode (PIN_B, OUTPUT) ;
  pinMode (PIN_BIT_1, OUTPUT) ;
  pinMode (PIN_BIT_2, OUTPUT) ;
  pinMode (PIN_BIT_4, OUTPUT) ;
  pinMode (PIN_BIT_8, OUTPUT) ;

  int nLastTemp = 0;
  int nLastHumid = 0;

  for (;;) {
  float fTemp = 0;
  float fHumid = 0;
  if (!read_am2302(7, 100, &fHumid, &fTemp)) {
    delay(100);
    continue;
  }
  nLastTemp = (int)fTemp;
  nLastHumid = (int)fHumid;
  int x = 0;
  DO_SOME_TIME { leftDigit(); setDigit(1); delay(5); }
  DO_SOME_TIME { setNumber(nLastTemp); }
  DO_SOME_TIME { rightDigit(); setDigit(2); delay(5); }
  DO_SOME_TIME { setNumber(nLastHumid); }
  noDigit();
  }
  return 0;
}
```

```
gcc -Wall -o hygrometer hygrometer.c -lwiringPi
sudo ./hygrometer
```

Anhang

Lizenz

Dokument verfasst und erstellt von Martin Strohmayer.
C-Programme erstellt von Manfred Wallner.



Die Anleitung enthält Teile aus dem eBook „Raspberry Pi - Projekte: Raspberry Pi als HTPC, Retro-Spielkonsole und für Elektronikprojekte nutzen“ das von
Amazon `http://www.amazon.de/kindle/dp/B00JGEEZOS` oder
Google `https://play.google.com/store/books/details?id=d6KAAwAAQBAJ` bezogen werden kann.
Wenn sie die Arbeit des Autors unterstützen wollen erwerben Sie das eBook, Danke!

Haftungsausschluss

Die Benutzung diesesr Anleitung und die Umsetzung der darin enthaltenen Informationen erfolgt ausdrücklich auf eigenes Risiko. Haftungsansprüche gegen den Autor für Schäden materieller oder ideeller Art, die durch die Nutzung oder Nichtnutzung der Informationen bzw. durch die Nutzung fehlerhafter und/oder unvollständiger Informationen verursacht wurden, sind grundsätzlich ausgeschlossen. Rechts- und Schadensersatzansprüche sind daher ausgeschlossen. Das Werk inklusive aller Inhalte wurde unter größter Sorgfalt erarbeitet. Der Autor übernimmt jedoch keine Gewähr für die Aktualität, Korrektheit, Vollständigkeit und Qualität der bereitgestellten Informationen. Druckfehler und Falschinformationen können nicht vollständig ausgeschlossen werden.