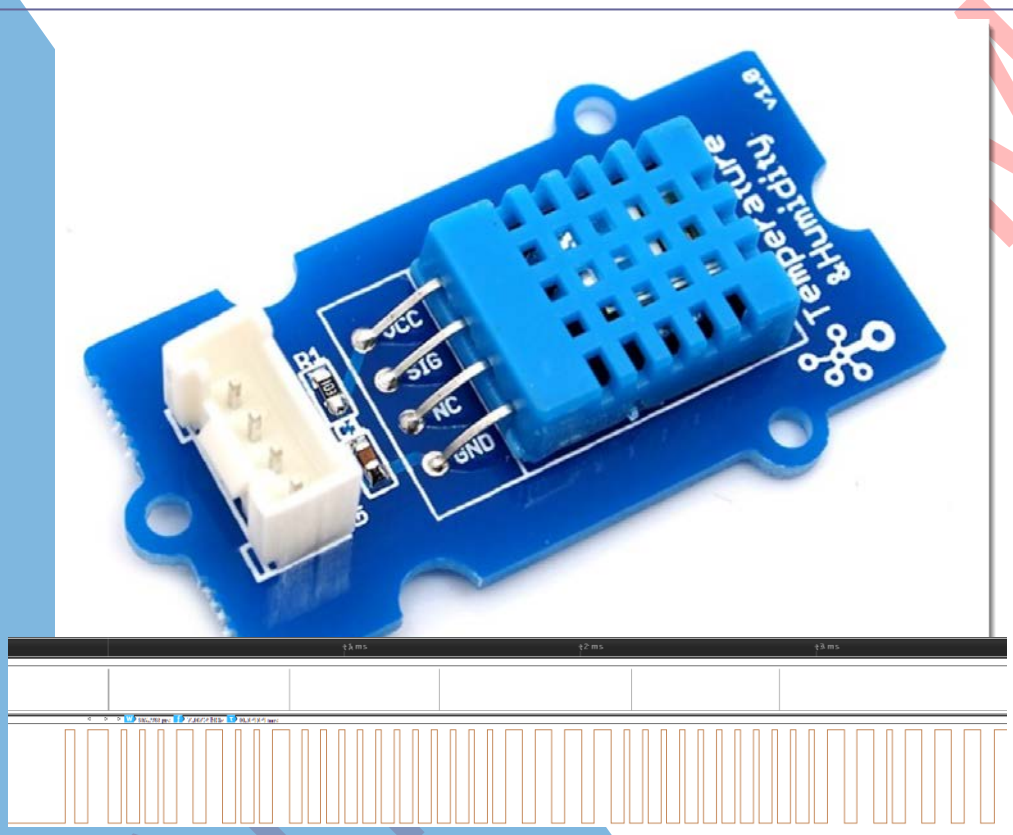
Fach HST

Externe Hardware

Sensor DHT11



Version: 1909.00A

1 Inhaltsverzeichnis

2 Der Sensor DHT11

2.1 Beschreibung

Ein Vorteil dieses Sensors ist die Kombination von Temperaturmessung und Luftfeuchtigkeitsmessung in einer kompakten Bauform - der Nachteil ist jedoch die geringe Abtastrate der Messung, so dass nur jede 2 Sekunden ein neues Messergebnis zur Verfügung steht - dieser Sensor ist somit sehr gut für Langzeit-Messungen geeignet. Sein Einsatzgebiet ist das Messen in unkritischen Umgebungen wo die Genauigkeit keine Rolle spielt.

Interessant ist auch die Kommunikation des Sensors gegen aussen. Ein „1-Wire Protokoll“, kein Standard, wird benutzt um die Daten zu übertragen. Dadurch eignet sich der Sensor gut für den Unterricht um zu lernen, wie z.Bsp ein Treiber für einen beliebigen Sensor gebaut wird.

Die Software beruht auf der Implementation von externen Treibern [1] sowie den Befehlen aus dem MCB32 Befehlsmanual [2] .

2.2 Stecker / Anschlussbelegung

Je nach Ausführung des Sensors, auf Print oder nur der Sensor sieht die Anschlussbelegung anderst aus. Der Sensor benötigt aber nur 3 Anschlüsse (siehe Bild rechts). Er kann direkt mit dem MCB32 verbunden werden.

GND =	Pin GND
+V =	Pin VCC 3.3 .. 5V
Signal =	Pin Signal

2.3 Vergleich von einigen gebräuchlichen Sensoren

Typ	LM35	DHT11	DHT22
Temperaturbereich:	-40 bis +110 °C	0 bis 50 °C, ±2 °C	-40 bis +80 °C, ±0.5
Messbereich Feuchte:	keine	20 - 90 RH	0 - 100 RH
Versorgungsspannung:	-0,2 bis +35 VDC	3,3 bis 5,5 VDC	3,3 bis 5,5 VDC

2.4 Anschluss an MCB32

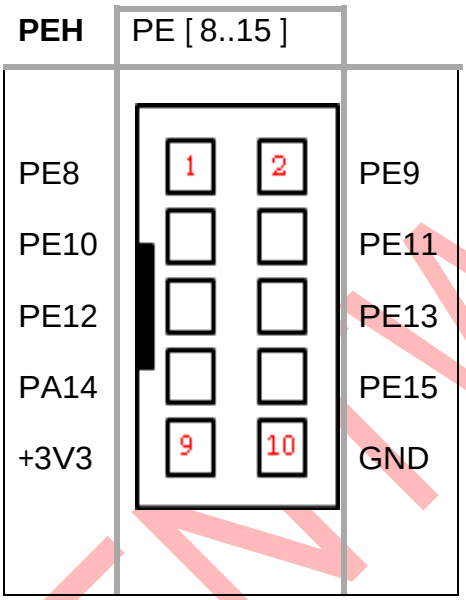
Die folgende Tabelle zeigt die Anschlussbeschreibung.

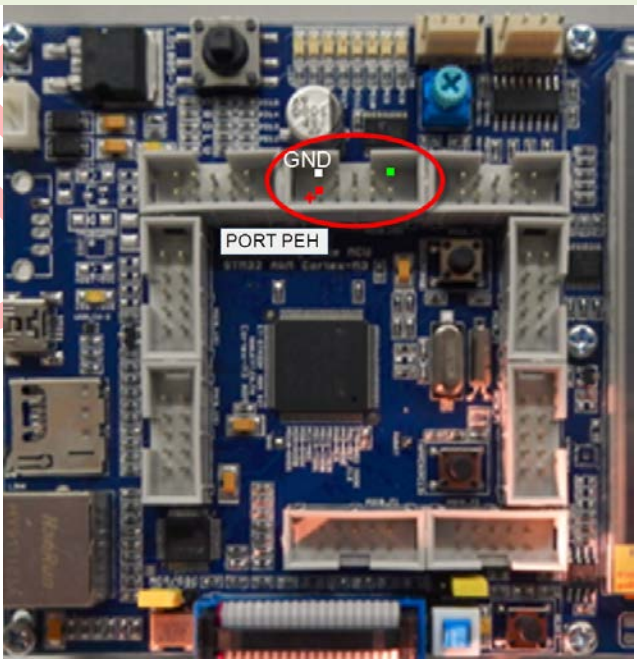
Pin	Spannung	MCB32 Stecker	Beschreibung
VCC	3..5V	PEH9	Speisespannung 3 .. 5V
GND	0V	PEH10	Ground
SIG	0 …3.3V	PEH1 ➔ PE8	Signal 1-Wire mit Pull-Widerstand auf DHT11. Von DHT11 zu MCB32. Pin PEH1= PE8 wechselt zwischen Ausgang und Eingang (siehe Code)

Die Verbindungen können mit handelsüblichen DuPont-Steckerkabeln ausgeführt werden.

In unserem Fall muss der Sensor an PE8 (Pin1) sowie an + und Gnd angeschlossen werden.

2.4.1 Steckerbelegung für 10pol. Stecker PE[8..15]

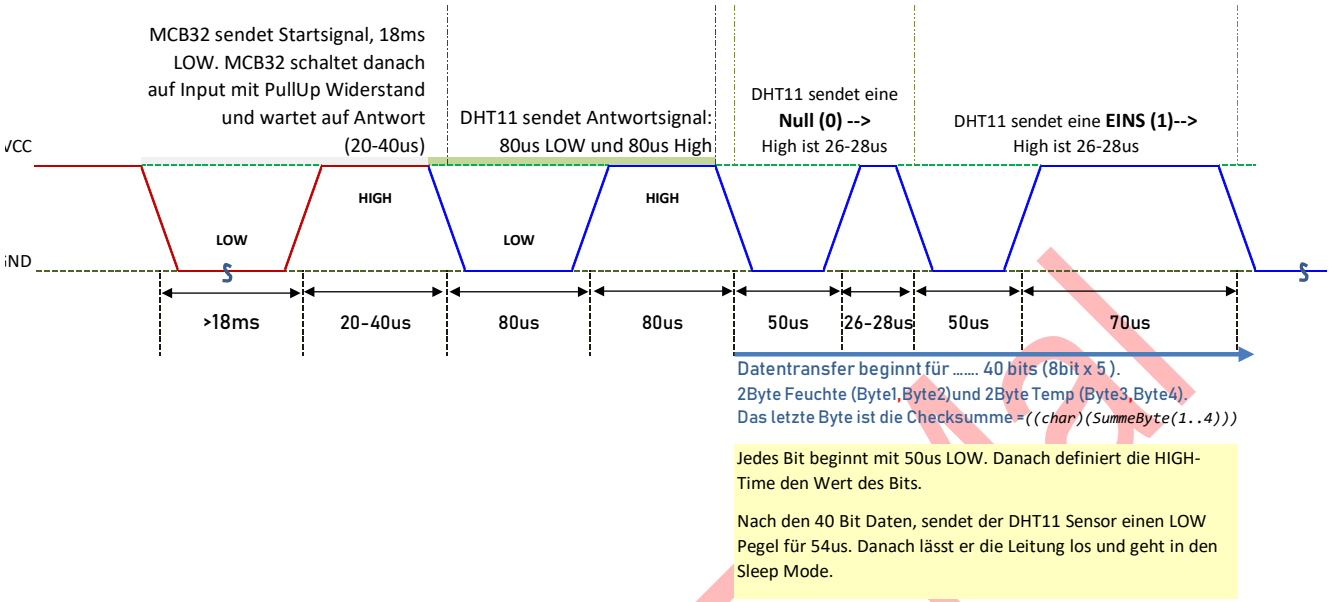




Die Belegung des 10poligen Steckers sieht wie oben am Beispiel des Steckers PEH abgebildet aus. Die roten Zahlen definieren die Adern des Flachbandkabels. Rote Ader = Pin1, daneben Ader 2 = Pin2 usw.

2.5 Datenübertrag und ein Beispiel

Gemäss Unterlagen des Herstellers sendet der DHT11 Sensor die Daten wie folgt aus [3]:

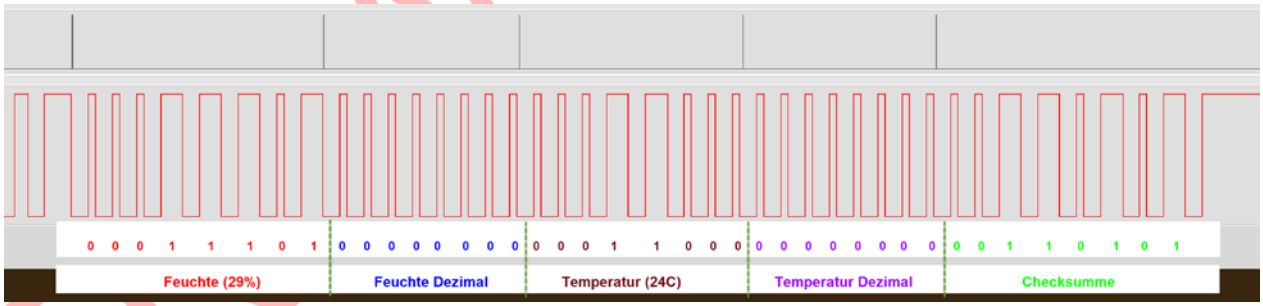


Jedes Bit beginnt mit 50us LOW. Danach definiert die HIGH-Time den Wert des Bits.

Nach den 40 Bit Daten, sendet der DHT11 Sensor einen LOW Pegel für 54us. Danach lässt er die Leitung los und geht in den Sleep Mode.

Das MCB32 sendet zuerst eine Startsequenz (18ms LOW) welche vom DHT mit 2 80us Pulsen beantwortet wird. Wichtig ist, dass das MCB32 nach den 18ms den Ausgang auf Eingang mit PullUp (20-40us HIGH) schaltet um dem DHT11 die Möglichkeit zu geben, das Signal zu manipulieren.

Das folgende Beispiel zeigt die Datenübertragung von 29% und 24°C. Der Sensor startet mit dem MSB des ersten Bytes und endet mit dem LSB des letzten Bytes (Dezimal Temperatur). Die Werte sind Dezimalcodiert. Das heisst, der Wert kann direkt gelesen werden. Die obere Linie markiert jeweils mit einem Impuls den Start einer Byteübertragung.



Aufgabe: Berechne die Checksumme im obigen Bild und prüfe ob sie stimmt. Lösung:

Also	Feuchte	29%
	Temperatur	24C
	Total	53
Minus	Checksumme	**53**
	TEST	**0 = OK**

Wenn das Resultat 0 gibt, ist die Übertragung OK.

3 Muster Code

```
/** @file  DHT11_MAIN.c
*  @brief  Prototyp for DHT11   based on http://tath.eu/projects/dht11-driver/
*  @author  mal
*  @date    12.12.2018
*  @version 1.0
*  @bug     No known bugs. ...
*/
//================================Includes=====================================
// #include <stm32f10x.h>              // Mikrocontrollertyp
#include "TouchP0P1.h"         // P0/P1,8Bit,Touchscreen und Grafik
#include ".\gpio\gpio.h"
#include ".\dht11\dht11.h"
#include "Trigger.h"           // Trigger Driver for generating Triggerpoints to syn. Logicanalayzer

// Prototypes of local Functions
void delay_us(unsigned int uius);
void ldelay_ms(unsigned int uims);
void plotKeepalive(unsigned int pX, unsigned int pY, unsigned char pR, unsigned char toggleON);
//================================main()=======================================
int main(void)                         // Hauptprogramm
{
   long lt = 0;                        // Delayvar.
   uint8_t dht11Data[5] = { 0 };       // dht11 data container
   InitTouchScreen();                  // Init. der Display Hardware
   //InitTouchP0P1("0");               // P0,P1 auf Touchscreen ON. Bei 0 wird auch der Systick Handler abgeschaltet
   setScreenDir(HOR);                  // HOR= Direction of wrting on screen starts at 0,0 (near Reset Switch)
   if (getScreenDir() == VER) {        // IF ScreenDIR = VER the draw some blue background
      clearScreen(BLUE);
      for (lt = 12000000;lt<0;lt--) {};         // Wait
   }

   Trigger_Init();                              // Triggerport init   (for debugging)

   /* setup DHT11 */
   RCC->APB2ENR |= RCC_APB2ENR_IOPEEN;      /* GPIOE clock enable (DHT11) */
   Gpio_SetPinAsOutput(GPIOE, GPIO_PIN8, GPIO_SPEED_50MHZ, GPIO_AF_PUSH_PULL);
   DHT11_Init();

   textxy(" All initialization done", 2, (32 + 32), WHITE, BLACK);
   textxy("DHT 11 Sensor Training Environment", 2, 16, WHITE, BLUE);
   ldelay_ms(200);
   textxy(" Malacarne 12.12.2018", 2, 32, BLACK, YELLOW);
   textxy(" DHT11 Init Done", 2, (32 + 16), WHITE, BLACK);
   ldelay_ms(200);

   // Now lets start
   while (1)
   {
      if (DHT11_OK == DHT11_Read(dht11Data))     // Read Data if there is a DHT11 on Porte Pin 8
      {
      // Data are stored in  dht11Data[ ]: [0] = Humidity Integral Byte, [1] Humidity decimal Byte,
      //                                   [2]=Temp integral Byte, [3] = temp decimal Byte, [4] = Checksum
         textxy("DHT11 data  OK:", 2, (32 + 32 + 32), WHITE, BLACK);
         printAt(6, "DHT11 data: ");
         printDec(3, dht11Data[0]);print("%; Temp:"); printDec(3, dht11Data[2]);
         print("C"); plotDot(220, 120, BRIGHT_RED); // Dec. not used becaus 0
      }
      else /* wrong checksum or timeout */
      {
         textxy("DHT11: ERROR on read", 2, (32 + 32 + 32), WHITE, BLACK);
      }
      ldelay_ms(2000);                 // is ok for prototyp. solution with rtc or timer is better
      plotKeepalive(312, 12, 4, 1);    // show keep alive point on screen with changing colors.
   }
}
// see Codetemplate for all the details
```

4 Programmbeispiele

4.1 Beispiel 1: Blinkender Punkt oben rechts

```
// show keepalive on screen
void plotKeepalive(unsigned int pX, unsigned int pY, unsigned char pR, unsigned char toggleON)
{
   const unsigned int ldelval = 4;   // how many delay shoud be waited (external or local)
   static char toggle = 0;                     // ??
   static unsigned int delaycounter = 0;       // ??
   if (toggleON == 1)
   {
      if (delaycounter< (ldelval / 2))         // see if between 0 and ldelval/2 or
      {
         toggle = 0; delaycounter++;
         //  ldelay_us(1);                    // not used if external delay is active
      }
      else
      {
         if (delaycounter <= ldelval)          // see if between ldelval/2 and ldelval
         {
            toggle = 1; delaycounter++;
            //  ldelay_us(1);                  // not used if external delay is active
         }
         else {                                // if more than ldelval then reset to 0
            toggle = 0; delaycounter = 0;      // reset and start again
         }
      }
      if (toggle == 0) {
         circle(pX, pY, pR, 2, GRAY1, 1);
      }
      else {
         circle(pX, pY, pR, 2, YELLOW, 1);
      }
   }
   else    // plotkeepAlive should not toggle the color
   {
      circle(pX, pY, pR, 2, BRIGHT_GREEN, 1);
   }

}
```

Aufgabe: Das obige Beispiel aus dem Code von `DHT11_Main.c` ist zu studieren und zu debuggen.

- Schreibe einen Header in welchem die Funktion eindeutig beschrieben wird.
- Beschreibe die Parameter welcher der Funktion übergeben werden im Header
- Erarbeite die Funktion der const und static Variablen im Code.
- Woher kommen die Begriffe für die Farben

5 Aufgaben DHT11 und RS232 Programmierung

DHT11_1. Schreibe das Programm um, so dass der Code an jedem beliebigen Pin von PortE-High gelesen werden kann.

DHT11_2. Das MCB32 soll den Feuchtwert und den Temperaturwert via RS232 (USART2) an den PC senden und in einem Terminal darstellen. Definiere das Protokoll, die Darstellung und die Übertragsmodularitäten (8N1 …)

DHT11_3. Schreibe mit C# ein Gui, welches die Daten in einem geeigneten Fenster darstellt.!

DHT11_4. Erweitere das Gui, so dass die RS232 Parameter ausgewählt werden können.

9 Anhang Referenzen

[1] «tath.eu,» tath@o2.pl, www.tath.eu, 25 12 2018. [Online]. Available: http://tath.eu/projects/dht11-driver/. [Zugriff am 25 12 2018].

[2] Malacarne, «MCB32_Befehlsliste_LIB_1802,» Malacarne Enrico, Rüti, 2018.

[3] Malacarne, «Timing DHT11,» Enrico Malacarne, Rüti, 2018.