EINE 433 MHZ FUNKSTRECKE MIT 2 ARDUINOS AUFBAUEN

[GEBEN SIE IHRE ADRESSE EIN]

Zielsetzungen

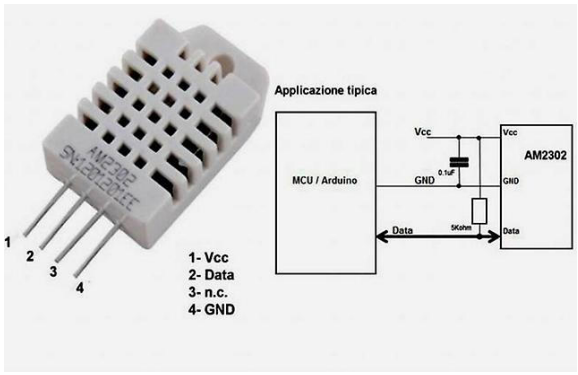
Die zuvor im Projekt ‚Sensorik' gesammelten Meßdaten sollen nun drahtlos übertragen werden.

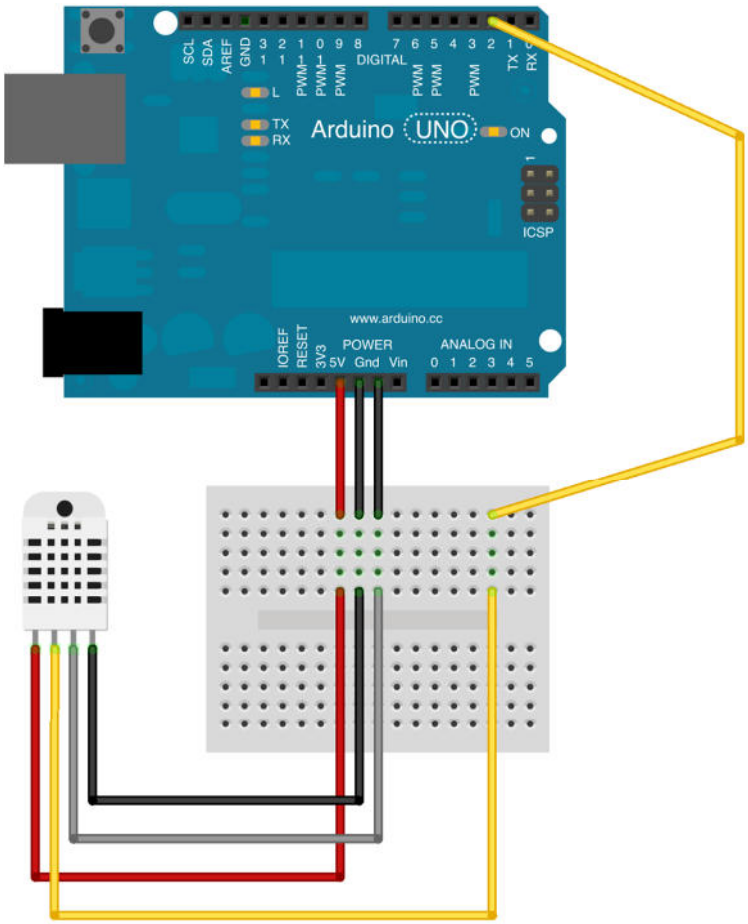
- ➔ http://shelvin.de/433-mhz-sender-und-empfaenger-funkstrecke-aufbauen/
- ➔ http://shelvin.de/433-mhz-datenuebertragung-mit-dem-arduino/
- ➔ http://www.wetterstationschlei.de/28.html

Auch dieses Projekt wird mehrstufig angelegt.
Im ersten Schritt soll zunächst nur der Empfänger und der separate Sender aufgebaut werden. Beides mit einem eigenen ARDUINO UNO R3. Der Sender soll dann mit einem Temperatur-Sensor wie dem AM2302, oder dem DHT22 ausgestattet werden und Luftfeuchte und Temperatur drahtlos an den Empfänger senden.

Benötigt werden dazu 2 ARDUINO UNO, ein Temperatur-Sensor, je eine Sende- und Empfangseinheit für 433 MHz, 2 Breadboards und ein paar Verbindungskabel. Zudem sind zwei Sketche nötig, als auch die Library RCSwitch.h.

Der Aufbau:
Wer den AM2302-Sensor nutzt, sollte unbedingt noch einen 4.7K-Widerstand zwischen den beiden Anschlüssen Vcc und Data vorsehen !





Arduino Uno + AM2302 Sensor
Derek Erb - 31/01/2013

about.me/derekerb
derek.erb@gmail.com

AM2302 Left to right
Sticker / Label facing up

AM2302 +5v power to Arduino 5v
AM2302 Data to Arduino Pin 2 (or 3)
AM2302 Supplementary ground to Arduino GND
AM2302 Principal ground to Arduino GND

Made with Fritzing.org

Das Sender-Modul kann direkt auf den ARDUINO aufgesteckt werden (Bestückungsseite nach innen auf den ARDUINO). Ist hier, in unserem Fall, aber nicht erstrebenswert, da wir ja auch noch den Sensor anschließen müssen. Und beim AM2302 kommt dann auch noch ein Widerstand hinzu. Zudem sollten alle 4 Anschlüsse des AM2302 (2x Masse) angeschlossen werden, da bei mir im Aufbau der Sensor ansonsten extrem heiß wurde. Das alles bedarf dann eben doch noch zusätzlich eines kleinen Steckboards. Die Anschlüsse lauten für den Sender (der trotzdem aufgesteckt werden kann) Pin7 = GND, Pin6 = Data, Pin5 = Vcc. Wobei der Sender mit bis zu 12,5 V betrieben werden könnte, was seine Leistung von 10mW auf 25 mW erhöhen würde – und damit dann auch seine Reichweite vergrößern würde, die aktuell (5V) bei mind. 10mtr. liegt. Die Datenleitung des Sensors wird mit Pin9 verbunden.

Die Pins können die Strom-versorgung übernehmen, Das spart das Steckbrett.
D7 ist GND
D5 ist VCC / 5V
D6 ist Datenausgang

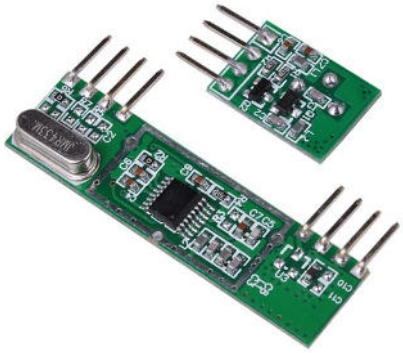
Zudem ist noch an den Sender eine Lambda ¼ -Antenne in Form eines festen Drahtes von 17 cm anzulöten. Das war schon alles, was auf der Sender-Seite zu machen ist. Natürlich ist das Programm noch auf den ARDUINO zu laden und dieser mit Strom zu versorgen.

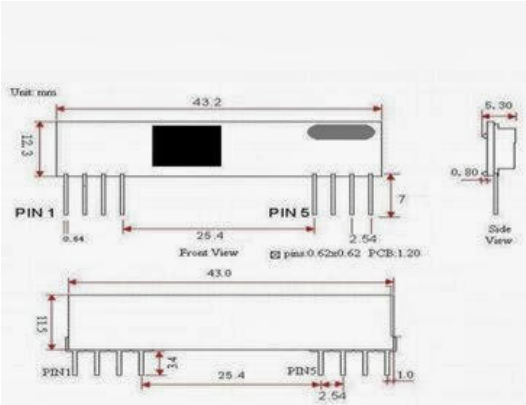
Hier das Script für den Sender, mit dem Sensor

```
#include <RCSwitch.h>
RCSwitch mySwitch = RCSwitch();
int led = 13;

#include <DHT.h>           // DHT-Bibliothek
#define DHTPIN 9
#define DHTTYPE DHT22       // DHT 22  (AM2302), AM2321
// #define DHTTYPE DHT21       // DHT 21 (AM2301)
// #define DHTTYPE DHT11       // DHT 11
DHT dht(DHTPIN, DHTTYPE);   // Temp-Feuchte-Sensor
float h;                    // Luftfeuchte DHT22
float t;                   // Temperatur DHT22


void setup()
{
  Serial.begin(9600);

  pinMode(led, OUTPUT); // LED
  pinMode(6, OUTPUT);   // Data
  pinMode(7, OUTPUT);   // GND
  digitalWrite(7, LOW);
  pinMode(5, OUTPUT);   // Vcc 5V
  digitalWrite(5, HIGH);

  // Transmitter is connected to Arduino Pin #6
  mySwitch.enableTransmit(6);

  // Optional set pulse length.
  // mySwitch.setPulseLength(320);

  // Optional set protocol (default is 1, will work for most outlets)
  // mySwitch.setProtocol(2);

  // Optional set number of transmission repetitions.
  // mySwitch.setRepeatTransmit(15);

  dht.begin();                  // Start des Temp-Sensors
} // ende setup



void loop()
{ // *** Sensorwerte einlesen ***
  h = (dht.readHumidity() + 30);    // Luftfeuchte auslesen, sowie mit einem Korrektur-Wert addieren
  t = dht.readTemperature();        // Temperatur auslesen
  int H = (h) * 10;                 // ich will einen Komma-freien-Wert
  int T = (t) * 10;                 // ich will einen Komma-freien-Wert
  unsigned long Wert;


  // *** Zusammenbau der beiden Werte mit einer trennenden '0' ***
    String x = String (int(T) + String("") +  int(H));
    Wert = x.toInt();

  // Wenn der Temperatur-Wert einstellig ist - setze noch eine '0' davor
  // --- diese Korrektur muss auf der Empfängerseite bei Werten < 100000 passieren

  // **** Es muss noch eine Loesung für Minus-Werte entwickelt werden. ****
  // ***************************************************************************
```

```
  if (isnan(t) || isnan(h))          // Pruefen ob eine gueltige Zahl zurueckgegeben wird. Wenn NaN (not a number) zurueckgegeben wird, dann
Fehler ausgeben.
  {
    // Serial.println("AM23002/DHT22 konnte nicht ausgelesen werden");
  }
  else
  {
    Serial.print("Luftfeuchte  = ");
    Serial.print(H);
    Serial.println(" %\t/10 ");  // die Zeichenfolge \t erzeugt einen Tabulator-Sprung
    Serial.print("Temperatur   = ");
    Serial.print(T);
    Serial.println(" C\t/10 ");

    mySwitch.switchOn("11111", "00010");
    delay(1000);
    mySwitch.switchOn("11111", "00010");
    delay(1000);

    digitalWrite(led, HIGH);       // LED anmachen
    mySwitch.send((Wert), 19);     // Temperatur *10, als 19-Bit-Signal = Werte bis 524.287
    digitalWrite(led, LOW);        // LED ausmachen
    Serial.print("Die gesendete Zahl lautet. "); Serial.println(Wert);
    delay(15000);                  // 15 Sekunden bis zur naechsten Uebertragung warten
  } // Ende else
} // Ende loop
```

Jetzt kommen wir schon zum Empfänger. Es gibt diverse Arten von Empfängern. Wir haben uns hier zu einem Superhet-Empfänger mit Quarz entschlossen. Zusammen mit dem Sender sind das dann Kosten von 10,- €. Beim dem Empfänger ist es so, dass hier von den 8 Anschlüssen nur sieben benötigt werden. Von den sieben Anschlüssen ist einer die Antenne.





Pin	Name	Function
1	ANT	Antena In
2	GND	Ground
3	GND	Ground
4	VDD	Positive Supply Power
5	VDD	Positive Supply Power
6	DE	Option output: 1, DATA output(default) 2. Battery Saveing Mode
7	DATA	Data Output
8	GND	Ground

Das Script **_433_MHz_RX_sensorB.ino** zeichnet sich durch die Besonderheit aus, dass es an sich aus 2 Scripten besteht, die zusammen aufgerufen werden. Zum einen ist es das eben das zuvor genannte Script und zusätzlich auch noch das **output.ino**. Im Augenblick wird dann das empfangene Signal verarbeitet und (nur) im seriellen Monitor ausgegeben.
Alleine bis hierhin kann der interessierte Bastler noch leicht sehr viel verändern und anpassen. So etwa könnte die Anzeige auf einem LCD-Display, oder einem kleinen TFT-Monitor erfolgen. Da aber auch andere Signale aufgefangen werden können (z.B. die fernsteuerbare Steckdosen, Toröffner, etc.) , könnten diese Signal auch weiter verarbeitet werden. So z.B. mit Relais, oder mit zusätzlichen Anzeigen und Ausgaben.

Und hier die aktuellen Scripte (20.04.2016) für den Empfänger

```
// *** Komplett ueberarbeitet und erweitert, April 2016, DL2EBZ ****

#include <RCSwitch.h>
RCSwitch mySwitch = RCSwitch();

void setup() {
  Serial.begin(9600);
  mySwitch.enableReceive(0);  // Receiver on interrupt 0 => das ist pin #2
}

void loop()
{
  if (mySwitch.available())
  {
    output(mySwitch.getReceivedValue());
    mySwitch.resetAvailable();
  }
}
```

Output.ino

```
// void output(unsigned long decimal, unsigned int length, unsigned int delay, unsigned int* raw, unsigned int protocol)
void output(unsigned long decimal)
{
  if (decimal == 0)
  {
    Serial.print("Unbekannte Codierung.");
  }
  else
  {
    if (decimal == 5393)  // dieser Wert laueft mir nach (womoeglich DIP-Switch-Kennung)
    {
      // nichts machen
    }
    else
    {
      unsigned long Wert = (decimal);

      if ((Wert < 100000) && (Wert > 9999)) // fuer Temp-Werte zwischen 1,0 Grad und 9,9 Grad
      {
        Serial.println("Die Temperatur liegt < 10 C");

        // *** Es folgt das Trennen der beider Zahlen-Trios und die Ausgabe der Einzel-Werte ***  --> Der zusammengesetzte Wert lautet
bspw.: 242607
        String TH = String(Wert, DEC);
        Serial.print("Die String-Variable TH hat den Inhalt "); Serial.println(Wert);
        String T = TH.substring(0, 2); // 1.+2.Stelle
        String H = TH.substring(2, 5); // 3.-5.Stelle
        // *** Dann sind die Einzelwerte in float-Variablen umzusetzen und durch '10' zu dividieren um wieder auf den Normalwert zu
kommen
        float Tem = T.toFloat();
        float Hyg = H.toFloat();
```

```
      float Temp  = (Tem / 10);
      float Hygro = ((Hyg / 10) - 27);
      Serial.print("Temp.= "); Serial.print(Temp); Serial.print (char(186)); Serial.print("C"); Serial.print(" Hygro= "); Serial.print(Hygro);
Serial.println("%");
    }

    else if ((Wert < 10000) && (Wert > 999)) // fuer Temp-Werte zwischen 0,1 Grad und 0,9 Grad
    {
      Serial.println("Die Temperatur liegt < 1 C");

      // *** Es folgt das Trennen der beider Zahlen-Trios und die Ausgabe der Einzel-Werte ***  --> Der zusammengesetzte Wert lautet
bspw.: 242607
      String TH = String(Wert, DEC);
      Serial.print("Die String-Variable TH hat den Inhalt "); Serial.println(Wert);
      String T = TH.substring(0, 1); // nur 1.Stelle
      String H = TH.substring(1, 4); // 2.-4.Stelle
      // *** Dann sind die Einzelwerte in float-Variablen umzusetzen und durch '10' zu dividieren um wieder auf den Normalwert zu
kommen
      float Tem = T.toFloat();
      float Hyg = H.toFloat();
      float Temp  = (Tem / 10);
      float Hygro = ((Hyg / 10) - 27);
      Serial.print("Temp.= "); Serial.print(Temp); Serial.print (char(186)); Serial.print("C"); Serial.print(" Hygro= "); Serial.print(Hygro);
Serial.println("%");
    }

    // else if  --> Temp.-Wert ist unter '0' NULL

    else
    {
    // *** Es folgt das Trennen der beider Zahlen-Trios und die Ausgabe der Einzel-Werte ***  --> Der zusammengesetzte Wert lautet
bspw.: 242607
    String TH = String(Wert, DEC);
      Serial.print("Die String-Variable TH hat den Inhalt "); Serial.println(Wert);
      String T = TH.substring(0, 3); // 1.-3.Stelle
      String H = TH.substring(3, 6); // 4.-6.Stelle
      // *** Dann sind die Einzelwerte in float-Variablen umzusetzen und durch '10' zu dividieren um wieder auf den Normalwert zu
kommen
      float Tem = T.toFloat();
      float Hyg = H.toFloat();
      float Temp  = (Tem / 10);
      float Hygro = ((Hyg / 10) - 27);
      Serial.print("Temp.= "); Serial.print(Temp); Serial.print (char(186)); Serial.print("C"); Serial.print(" Hygro= "); Serial.print(Hygro);
Serial.println("%");
    } // ende else 'Wert >= 10.0 C'
  } // ende else 'nicht 5393'
 } // ende else 'unbekannte Codierung'
} // ende void
```

Man kann leicht erkennen, dass das gesamte entschlüsseln des Signals in der output.ino erfolgt. Hier ist auch schon eine Korrektur enthalten, die dafür sorgt, dass einstellige Temperaturen (unter 10°C), sowie Temperaturen unter 1°C korrekt dargestellt und entschlüsselt werden können.

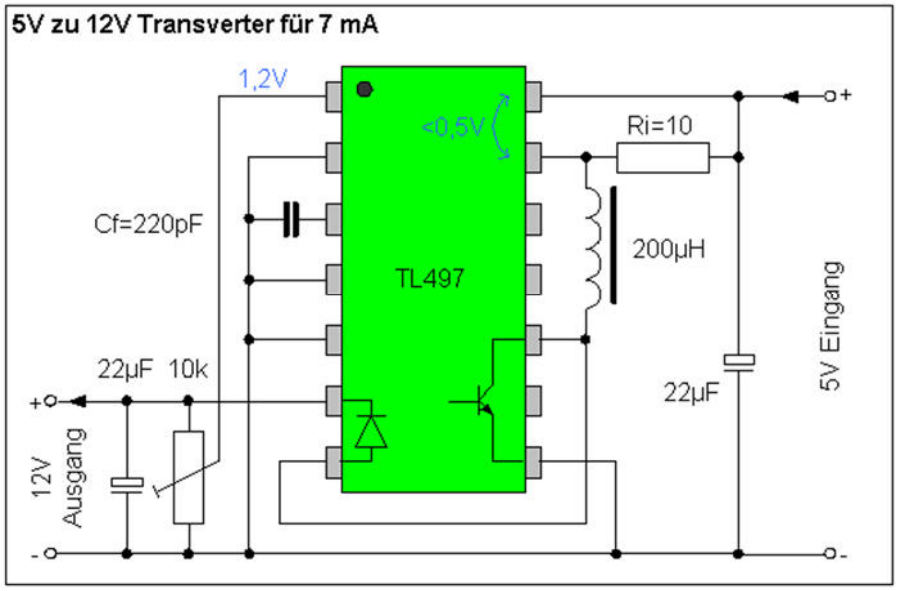
Es fehlt an dieser Stelle aktuell noch die Korrektur auch für Temperaturen unter 0°C.

Das wäre damit der erste Teil des Aufbaus einer drahtlosen Datenverbindung. Der zweite Teil wäre nun der, diese Funktionalität auch auf das letzte fertige Projekt einzubinden. Das war unser Projekt mit der (Innen-) Temperatur, dem Luftdruck, der (Innen-) Luftfeuchte, der Realtime-Uhr, der SD-Karten-Speicherung, sowie der LCD-, bzw. TFT-Anzeige. → *DS3231_Sensorik_SD_Interval4.ino*, bzw. *Sensorik_SD_TFT5.ino*.

Und dann kommt noch der dritte Teil, bei dem diese Funktionalität in die GPS-Uhr eingebunden wird.

Aber alleine jedes Teil-Projekt lässt schon umfangreiche individuelle Erweiterungen zu !

Hier noch ein Beispiel-Aufbau für die Umsetzung von 5V auf 12V auf der Sender-Seite.



Stand: 28.April 2016, Bodo Schnare