Prototyping a device for monitoring in apiculture

Gabriele Labanca

July 27, 2018

Abstract

This document briefly describes the prototype of a device for weight measurements in apiculture, developed in collaboration with *Re.Te.*, part of *Sermig* onlus [1]. I would like to aknowledge Roberto Verzino in particular, which followed me in learning and applying the necessary knowledge. The project is open source [2].

[1] `www.sermig.org`
[2] `https://github.com/GabrieleLabanca/Pesatura_Arnie`

Contents

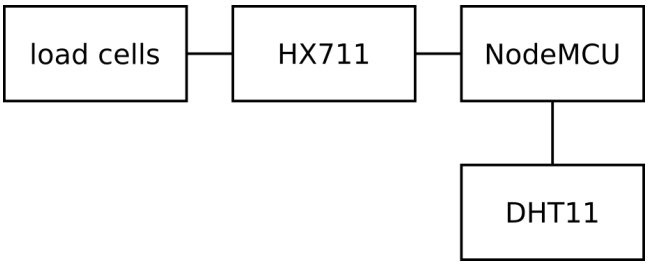

Figure 1: General scheme of the device.

1 The device

The device is based upon a NodeMCU wi-fi board [3]: sensors to measure weight, temperature and humidity are used, which will be described in the following; see last chapters to see some further implementations of a battery-level monitor and a resting capability.

1.1 NodeMCU

Programming the NodeMCU is as easy as writing Arduino code, provided that the support for Esp8266 is installed and the appropriate board is selected [4].

1.2 Weight - load cells + HX711

A Wheatstone bridge configuration for four load cells is used [5] [6]; see Figure 2. Since the signal is too weak to be detected directly by the board, a HX711 amplifier is used; in the figure the schematics of connections to the board are shown.

In order to make the HX711 work, the library `HX711.h` is used [7]. The code highlights follow:

```
#include "HX711.h"
// set the pins used by the amplifier
#define HX711_SCK_PIN  D1
#define HX711_DOUT_PIN D2
// create a HX711 object
HX711 scale;
scale.begin(HX711_DOUT_PIN,HX711_SCK_PIN);
scale.power_up();
// the value of myscale is obtained by calibrating
```

[3] This choice has been guided exlusively by the smaller dimension and cost of this board; an Arduino board with an appropriate wi-fi shield can of course do as well.

[4] https://www.instructables.com/id/Quick-Start-to-Nodemcu-ESP8266-on-Arduino-IDE/

[5] https://www.aliexpress.com/item/1PCS-DIY-50Kg-Body-Load-Cell-Weighing-Sensor-Resistance-strain-Half-bridge/32597969753.html?spm=2114.13010608.0.0.pC56uP

[6] http://www.instructables.com/id/Make-your-weighing-scale-hack-using-arduino/ ,https://www.sparkfun.com/products/13878?_ga=1.186640489.1126097763.1485380550

[7] https://github.com/bogde/HX711

```
//   the scale with known weights
scale.set_scale(myscale);
// reset the scale to 0
scale.tare();
// get weight (tare and scale)
float weight = scale.get_units();
```

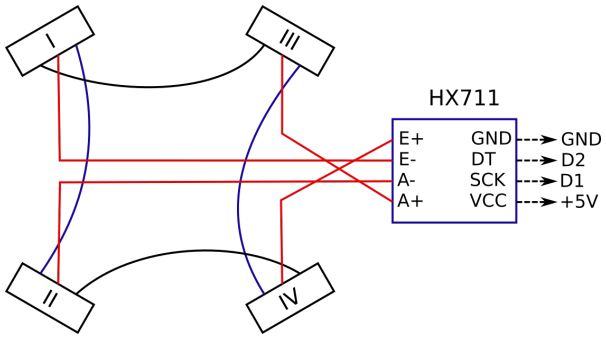

Figure 2: The Wheatstone bridge configuration for load cells, connected to the HX711 amplifier; connections from HX711 to NodeMCU board.

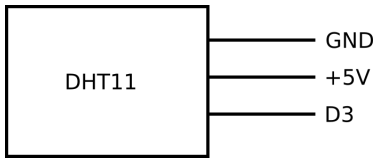

Figure 3: The schematics of DHT11

1.3 Temperature and humidity - DHT11

A DHT11 sensor is used to get measures of temperature and humidity. The related schematics is in Figure 1.2.

The library used is `DHT.h` and the relevant code is [8]:

```
#include "DHT.h"
#define DHTTYPE DHT11
#define DHT11_PIN D3              // signal pin (has to be digital)
DHT dht(DHT11_PIN, DHTTYPE);      // create a DHT11 object
float t = dht.readTemperature(); // read values
float h = dht.readHumidity();
```

[8] `https://github.com/adafruit/DHT-sensor-library`, **needs** `https://github.com/adafruit/Adafruit_Sensor`

2 Thingspeak.com

As far as data visualization is concerned, the website Thingspeak.com, powered by Matlab plugins, has been the platform of choice. Provided that one has an account (a free plan exists), the website allows the creation of a *channel*, which can contain up to 8 fields remotely updated with the provided API.

On NodeMCU boards, the following code starts a server connection

```
#include <ESP8266WiFi.h>
const char* server = "api.thingspeak.com";
String apiKey = ".........";     //  Enter the Write API key from
    ThingSpeak
WiFiClient client;
WiFi.begin(ssid, pass);
  while (WiFi.localIP().toString() == "0.0.0.0")
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println("WiFi connected");
```

while a String must be created to post it to the server:

```
if (client.connect(server, 80))
  {
    String postStr = apiKey;
    postStr += "&field1=";
    postStr += String(my_measure);
    postStr += "\r\n\r\n";
    client.print("POST /update HTTP/1.1\n");
    client.print("Host: api.thingspeak.com\n");
    client.print("Connection: close\n");
    client.print("X-THINGSPEAKAPIKEY: " + apiKey + "\n");
    client.print("Content-Type: application/x-www-form-urlencoded\n
    ");
    client.print("Content-Length: ");
    client.print(postStr.length());
    client.print("\n\n");
    client.print(postStr);
  }
  client.stop();
```

3 Data

A very simple, yet satisfactory, approach has been taken to the elaboration of data: a sample over one or more days is considered, of which the weight-temperature points are fitted with a linear function. Assuming that the weight variation is neglectable with respect to the dependence on temperature of the response of the load cells, which is usually a decent assumption, the relation found corrects the instrumental noise:

$$w_{\text{vero}}[i] = w_{\text{misurato}}[i] - (w_0 + m * T[i])$$

It is worth noting that without correction the uncertainty is of order $0.01kg$, so depending on the aim of the project may be that no correction is necessary to obtain an acceptable result.

The Matlab code, which can be directly used in a *visualization* on Thingspeak.com, is presented:

```
%%%%%%%%%%%%%%%%%%%%%%%%
% this code gets data (weight, temperature) from Pesatura Arnie (
    ChID 350718)
% the correlation between the two variables is used to clean weight
    data
% so they are not subject to temperature variation
% the results are then averaged and displayed (original data vs.
    clean data)

% read/write variables
writeChId = ......; % channel write id
writeKey = '.....'; % channel write API key
readChId = .......; % channel read id
readKey  = '.....'; % channel read API key

% retrieve last 100 data from Fields 1 and 2
npoints = 3*24*7;
[dataWeight,timestamps] = thingSpeakRead(readChId,'NumPoints',
    npoints,'Fields',1);%,'ReadKey',readKey);
dataTemperature = thingSpeakRead(readChId,'NumPoints',npoints,'
    Fields',2);%,'ReadKey',readKey);
dataHumidity = thingSpeakRead(readChId,'NumPoints',npoints,'Fields'
    ,3);
oldW = dataWeight; % keeps old measure
% clean data, using correlation with Temperature
m = 0.01; % correlation coefficient
n_back = 2; % NB the temperature affects weight with a DELAY!
T0 = 3
for i=1:npoints
    if(i<(n_back+1))
            Delta2 = dataWeight(i)-dataWeight(1)-m*(dataTemperature
    (i)-T0);
    else
            Delta2 = dataWeight(i)-dataWeight(1)-m*(dataTemperature
    (i-n_back)-T0);
    end
    dataWeight(i) = Delta2;
end
```

```
% find mean values
nmean = 40;
mean = [0,0];
mean_dataWeight = [];
mean_oldW = [];
mean_timestamps = timestamps(1:npoints/nmean);
for i=1:npoints
    mean(1) = mean(1) + dataWeight(i);
    mean(2) = mean(2) + oldW(i);
      if isequal(mod(i,nmean),0)
           mean_dataWeight(i/nmean) =  mean(1)/nmean;
           mean_oldW(i/nmean) = mean(2)/nmean;
           mean_timestamps(i/nmean) = timestamps(i-nmean/2); %
    center of interval
           mean = [0,0];
      end
end
numel(mean_dataWeight);
numel(mean_oldW);
numel(mean_timestamps);

%% Visualize Data %%
prevision = [];
for i=1:npoints
    prevision(i) = -dataWeight(1)+m*(dataTemperature(i)-T0);
end
A = horzcat(dataTemperature,transpose(prevision));
B = horzcat(transpose(mean_dataWeight),transpose(mean_oldW));
thingSpeakPlot(mean_timestamps,B,'XLabel','time','YLabel','weight(
    kg)','Title','weight with respect to original weight','Legend'
    ,{'corrected','measured'});
```

4 Appendice

4.1 Battery monitoring

If the device is powered by a battery, a simple way to monitor its discharging is reading from pin A0 with the configuration shown in Figure 4.

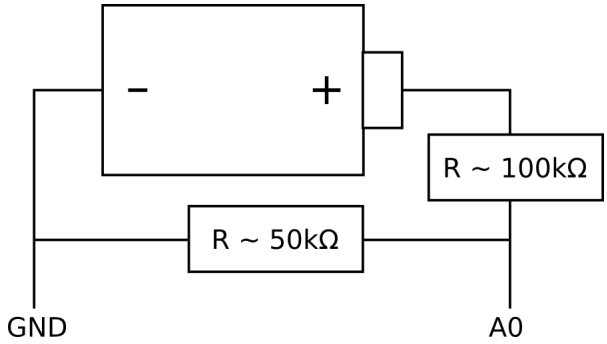

Figure 4: The configuration needed to read battery level from pin A0.

4.2 Resting

The device can be put to rest when inactive. This can be done connecting **D0 to RST** and with the following code lines:

```
#define SECONDS_DS(seconds)  ((seconds)*1000000UL)
ESP.deepSleep(SECONDS_DS(600), WAKE_RF_DEFAULT);
```

4.3 Complete code

```
////////////////////
// CONFIGURE HX711
#include "HX711.h"//The library used for arduino  https://github.com/bogde/HX711
// HX711.DOUT   - D1  (Arduino: 8 (BEFORE: pin 10))
// HX711.PD_SCK - D0 (Arduino: 7 (BEFORE: pin 11))
#define HX711_SCK_PIN  D1
#define HX711_DOUT_PIN D2
//scale(DOUT,SCK)
HX711 scale;//(HX711_DOUT_PIN, HX711_SCK_PIN); // parameter "gain" omitted; default value 128

////////////////////
// CONFIGURE DHT11
#include "DHT.h" // https://github.com/adafruit/DHT-sensor-library NEEDS https://github.com/adafruit/Adafruit_Sensor
#define DHTTYPE DHT11   // DHT 11
#define DHT11_PIN D3 //signal pin (has to be digital)
DHT dht(DHT11_PIN, DHTTYPE);
```

```
//DHT dht(DHTPIN, DHT11);

// CONFIGURE WIFI
#include <ESP8266WiFi.h>


// Configure WIFI
  const char *ssid =  "dlink_DWR-730_2F6E";
  const char *pass =  "arsenaleterra2017";

// CONFIGURE SERVER
const char* server = "api.thingspeak.com";
String apiKey = "AG5BH0BV8ITOCAUL";     //  Enter your Write API key from ThingSpeak
WiFiClient client;


void setup()
{
  delay(1000);
  Serial.begin(9600);
  Serial.println("HX711_DHT11_wifi");


  // SETUP HX711
  scale.begin(HX711_DOUT_PIN,HX711_SCK_PIN);

  scale.power_up();
  delay(1000);
  Serial.println("Before setting up the scale:");
  Serial.print("read: \t\t ");
  Serial.println(scale.read());     // print a raw reading from the ADC
  float myscale = 114. / .005600966442953021;
  scale.set_scale(myscale); // this value is obtained by calibrating the scale with known weights;
  scale.tare();   // reset the scale to 0
  Serial.println("After setting up the scale:");
  Serial.print("read: \t\t");
  Serial.println(scale.read());                 // print a raw reading from the ADC
  delay(15);
  // print the average of 20 readings from the ADC
  Serial.print("read average:\t\t ");
  delay(15);
  Serial.println(scale.read_average(20));
  delay(15);
  // print the average of 5 readings from the ADC minus the tare weight, set with tare()
  Serial.print("get value: \t\t ");
  Serial.println(scale.get_value(5));
  delay(15);
  Serial.print("get units: ");
  Serial.println(scale.get_units(5), 1);
  delay(15);
  // print the average of 5 readings from the ADC minus tare weight,
```

```
68    //divided by the SCALE parameter set with set_scale

70    //SETUP WIFI
      Serial.println("Connecting to ");
72    Serial.println(ssid);   Serial.println("HX711_DHT11_wifi");
      WiFi.begin(ssid, pass);
74    while (WiFi.localIP().toString() == "0.0.0.0") //while (WiFi.
        status() != WL_CONNECTED)
      {
76      delay(500);
        Serial.print(".");
78    }
      Serial.println("WiFi connected");
80    //Serial.println("HX711_DHT11_wifi READY");

82  }
    void loop()
84  {
      // GET DATA  M
86    // weight
      float weight = scale.get_units();//* .005600966442953021;
88    Serial.print("Weight: ");
      Serial.print(weight, 3); //tara con 3.870kg
90    float weight_raw = scale.read();
      Serial.print(" Raw weight: ");
92    Serial.print(weight_raw, 3);
        // temperature
94    float t = dht.readTemperature();
      // humidity
96    float h = dht.readHumidity();
      Serial.print(" Temperature: ");
98    Serial.print(t);
      Serial.print(" degrees Celsius, Humidity: ");
100   Serial.print(h);
      Serial.print('\n');
102
      // SEND TO THINGSPEAK
104   Serial.println("%. Send to Thingspeak.");
      if (isnan(t) || isnan(h))
106   {
        Serial.println("Failed to read from DHT sensor!");
108     return;
      }
110   if (client.connect(server, 80))  //   "184.106.153.149" or api.
        thingspeak.com
      {
112     String postStr = apiKey;
        postStr += "&field1=";
114     postStr += String(weight);
        postStr += "&field2=";
116     postStr += String(t);
        postStr += "&field3=";
118     postStr += String(h);
        postStr += "&field4=";
120     postStr += String(weight_raw);
        postStr += "\r\n\r\n";
122     client.print("POST /update HTTP/1.1\n");
```

```
      client.print("Host: api.thingspeak.com\n");
      client.print("Connection: close\n");
      client.print("X-THINGSPEAKAPIKEY: " + apiKey + "\n");
      client.print("Content-Type: application/x-www-form-urlencoded\n
      ");
      client.print("Content-Length: ");
      client.print(postStr.length());
      client.print("\n\n");
      client.print(postStr);
    }
    client.stop();

    // DELAY
    Serial.print('\n');
    int n_min = 10;
    int delay_time = 1000*60*n_min;
    delay(delay_time);
}
```

arduino/Working/HX711_DHT11_wifi/HX711_DHT11_wifi/HX711_DHT11_wifi.ino