Prototyping a device for monitoring in apiculture

Gabriele Labanca

July 31, 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

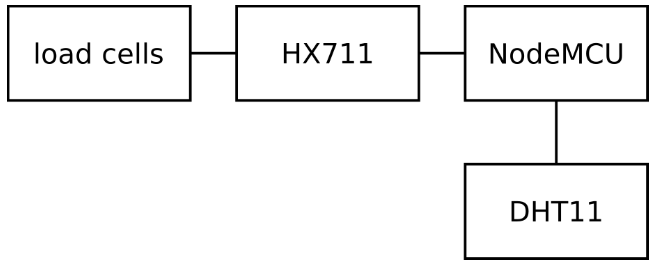

Figure 1: General scheme of the device.

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 relevant code follows:

```
#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(); // turn on the scale
scale.power_down(); // turn off the scale
```

[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 value of myscale is obtained by calibrating
//   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();
// get value of weight without tare
float weight_raw = scale.read();
```

Although, as shown in the code, it is possible to tare the scale with methods internal to the HX711 library, it is preferable to set up the tare with an external code: in order to reduce the calculations made on the NodeMCU and to make it more portable (i.e. requiring the least hands-on manteinance, allowing a remote intervention) such setup is done on Thingspeak: see 3.
Apart of that, there are other advantages here. A big problem with the scale was its recalibration at each "reboot", so that the weight had to be removed and put on again; since the calibration is now done online, once set there is no need to remove weights at reboots.

A **known problem** whith this system: bumps (even small ones) could cause a rescaling of the non-calibrated readings, compromising the values.

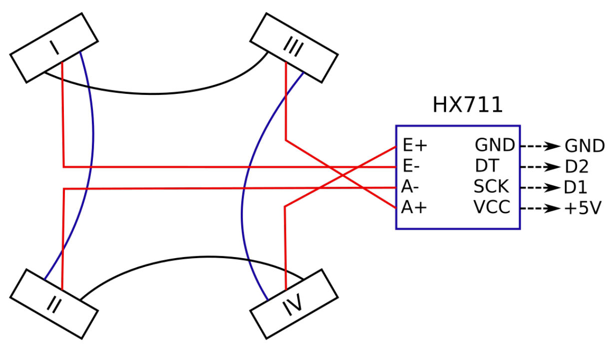

Figure 2: The Wheatstone bridge configuration for load cells, connected to the HX711 amplifier; connections from HX711 to NodeMCU board.

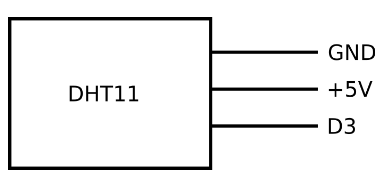

Figure 3: The schematics of DHT11

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 analysis

As it has been written in 1.2, the not-calibrated weight is elaborated on Thingspeak.com in order to obtain the true weight:

```
%%%%%%%%%%%%%%%%%%%%%%%%
% This code visualizes the true weight starting from raw readings (ch. 4),
%  starting from two readings to calculate the tare
% Averaged values!

% read/write variables
readChId = ...;    // put your own keys here (see Thingspeak.com)
readKey  = '...';

% retrieve last 100 data from Fields 1 and 2
npoints = 100;
[dataWeight,timestamps] = thingSpeakRead(readChId,'NumPoints',npoints,'Fields',4);

% tare: the relation is assumed to be linear
tw0 = 0.0;       // the true weight put on the scale at time 0
rw0 = -318718;   // the value read on the scale when not-calibrated at time 0
tw1 = 1.061;     // the true weight put on the scale at time 1
rw1 = -299786;   // the value read on the scale when not-calibrated at time 0
k_tare = (tw1-tw0)/(rw1-rw0);
q_tare = tw1 - rw1*k_tare;

dataWeight = dataWeight*k_tare + q_tare;

% average
nmean = 40;
mean = 0.;
mean_dataWeight = [];
mean_timestamps = timestamps(1:npoints/nmean);
for i=1:size(dataWeight)
    mean = mean + dataWeight(i);
      if isequal(mod(i,nmean),0)
          mean_dataWeight(i/nmean) =  mean/nmean;
          mean_timestamps(i/nmean) = timestamps(i-nmean/2); % center of interval
          mean = 0.0;
      end
end

%% Visualize Data %%
thingSpeakPlot(mean_timestamps,mean_dataWeight,'XLabel','tempo','YLabel','peso (kg)','Title','Peso arnia','Legend',{'peso'});
```

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 on the average 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. Moreover, calibrating online as in 1.2 appears to reduce the dependence on temperature.

The Matlab code, which can be directly used in a *visualization* on Thingspeak.com, is presented (the weight is calibrated as in the previous code):

```
% read/write variables
readChId = ...;
readKey  = '...';

% retrieve last 100 data from Fields 1 and 2
npoints = 500;%3*24*7;
[dataWeight,timestamps] = thingSpeakRead(readChId,'NumPoints',
    npoints,'Fields',4);%,'ReadKey',readKey);
dataTemperature = thingSpeakRead(readChId,'NumPoints',npoints,'
    Fields',2);%,'ReadKey',readKey);
dataHumidity = thingSpeakRead(readChId,'NumPoints',npoints,'Fields'
    ,3);

% transform raw data in true weights (see "tara")
tw0 = 0.0;
rw0 = -318718;
tw1 = 1.061;
rw1 = -299786;
k_tare = (tw1-tw0)/(rw1-rw0);
q_tare = tw1 - rw1*k_tare;

dataWeight = dataWeight*k_tare + q_tare;
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:
%  set = 3 if want to compare old measures
%  set = dataTemperature(1) if only a estimate of the variation in
    last days is wanted
T0 = dataTemperature(1);
prevision = [];
for i=1:npoints
    if(i<(n_back+1))
            prevision(i) = dataWeight(1)-m*(dataTemperature(i)-T0);
    else
            prevision(i) = dataWeight(1)+m*(dataTemperature(i-
    n_back)-T0);
    end
    dataWeight(i) = dataWeight(i) - prevision(i);
end

```

```matlab
% find mean values
nmean = 40;
mean = [0,0,0];
size(mean)
mean_dataWeight = [];
mean_oldW = [];
mean_prevision = [];
mean_timestamps = timestamps(1:npoints/nmean);
for i=1:npoints
    mean(1) = mean(1) + dataWeight(i);
    mean(2) = mean(2) + oldW(i);
    %mean(3) = mean(3) + prevision(i);
      if isequal(mod(i,nmean),0)
          mean_dataWeight(i/nmean) =  mean(1)/nmean;
          mean_oldW(i/nmean) = mean(2)/nmean;
          %mean_prevision(i/nmean) = mean(3)/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 %%
select = 'B'
if(select == 'A') % cfr old weight and temperature and prevision
    M = horzcat(dataTemperature,oldW,transpose(prevision));
    tax = timestamps;
end
if(select == 'B') % THE INTERESTING GRAPH
    M = horzcat(transpose(mean_dataWeight),transpose(mean_oldW),mean_prevision);
    tax = mean_timestamps;
end
if(select == 'C') % cfr temperature and weight => correlation
    M = oldW;
    tax = dataTemperature;
end
thingSpeakPlot(tax,M,'XLabel','tempo','YLabel','peso (kg)','Title','Peso arnia (rispetto a peso originale)','Legend',{'corretto','misurato'});
```

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.

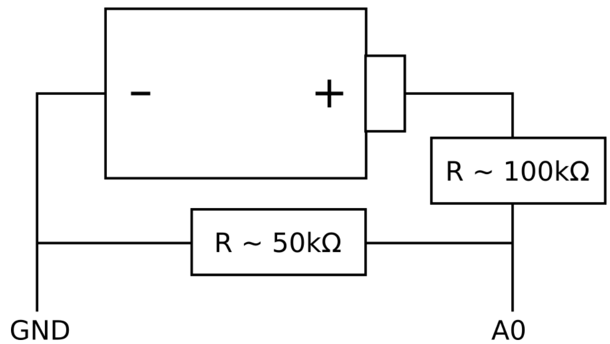

Figure 4: The configuration needed to read battery level from pin A0.

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>


// WIFI CREDENTIALS
const char *ssid = "FASTWEB-D68BC7";
const char *pass = "CY3APAPE3J";

// 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);

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

}
void loop()
{
  // GET DATA
  // weight
  scale.power_up();
  float weight_raw = scale.read();
  Serial.print(" Raw weight: ");
  Serial.print(weight_raw, 3);
  scale.power_down();

  // temperature
  float t = dht.readTemperature();
  // humidity
  float h = dht.readHumidity();
  Serial.print(" Temperature: ");
  Serial.print(t);
  Serial.print(" degrees Celsius, Humidity: ");
  Serial.print(h);
```

```
  Serial.print('\n');

  // SEND TO THINGSPEAK
  Serial.println("%. Send to Thingspeak.");
  if (isnan(t) || isnan(h))
  {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  if (client.connect(server, 80))  //   "184.106.153.149" or api.thingspeak.com
  {
    String postStr = apiKey;
    //postStr += "&field1=";
    //postStr += String(weight);
    postStr += "&field2=";
    postStr += String(t);
    postStr += "&field3=";
    postStr += String(h);
    postStr += "&field4=";
    postStr += String(weight_raw);
    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();

  // DELAY
  Serial.print('\n');
  int n_min = 1;
  int delay_time = 1000 * 60 * n_min;
  delay(delay_time);
}
```

arduino/HX711_DHT11_wifi/HX711_DHT11_wifi/HX711_DHT11_wifi.ino