Basic Custom Arduino Library for HX711

Nicholas Giacoboni

User's Manual v3.0
15 Decemberl 2019

This manual explains how to register a custom library and get started with the HX711 load cell amplifier in MATLAB® environment. You need to have already installed the MATLAB Support Package for ARDUINO® Hardware. It does require basic knowledge about MATLAB and his functionality, furthermore it assumes your load cell works properly and the connection is correct. This particular library has been tested with Arduino UNO and MEGA2560, if you are using Arduino DUE you can use this one `https://it.mathworks.com/matlabcentral/fileexchange/73693-custom-arduino-due-library-for-h`

Contents

1 Overview

An add-on library is a collection of MATLAB and C++ code that provides a user easy access to features on the Arduino hardware or attached shields [5]. The HX711 add-on library just develops the two wire comunication protocol between Matlab workspace and HX711 itself through ARDUINO, thus you can built your own commands to calibrate your load cell and acquire weight. In the HX711 folder you can find a basic example.

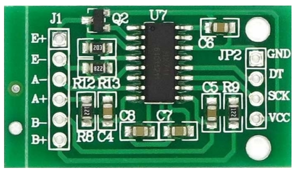


Figure 1

The HX711 is a precision 24-bit analog to digital converter (ADC) designed for weigh scales and industrial control applications to interface directly with a bridge sensor [8]. All you have to do is to specify the name of the digital pins in according to phisical connection. Only channel A with a gain of 128 is implemented. Please read section 2 and 3 before starting.

2 Limitations

- This library has been tested with ARDUINO UNO and MEGA2560 under windows environment. It does not work with ARDUINO DUE and it might not work with other board.
- You can make static or dynamic measurements with high precision even with low cost components, but in the second case do not exceed the sampling rate of 10 $[Hz]$ because you may have some comunication error (0 or NaN).
- Only channel A with a gain of 128 is implemented because I've obtained good results in terms of precision and stability.
- I've written this library for educational porpuse, so I want to keep the code easy to understand. If you are interested in a more sophisticated library you can download the following for Arduino Uno and Mega `https://it.mathworks.com/matlabcentral/fileexchange/73692-advanced-custom-` Instead if you are using Arduino DUE you can download the following one `https://it.mathworks.com/matlabcentral/fileexchange/73693-custom-arduino-du`

3 Caution

Here I sum up some tips you should follow in order to avoid wasting time and solve every problem quickly. Please don't ask for help in comment section if you are not going to check the following recommendations.

- I suggest to test your load cell with an usual library in Arduino IDE environment. You can find a very good one at `https://github.com/bogde/HX711`. After the calibration phase take note of the scale factor so you can compare it with the one you get from MATLAB.

- Make sure that the connection between ARDUINO and HX711 is correct and particularly the connection between the load cell and HX711 must be stable and strong (I personally recommend a soldered connection, it also reduces noise effect if it's well executed, instead of other temporary connection).
- The correct functioning of the load cell depends on the installation of the strain gauges, I warn you that on the market there are load cell with a low quality installation technique and even a wrong bridge connection between strain gauges. If the calibration phase is done properly you should be able to make measurements with at least $0.001\,[kg]$ of precision with a $0 \div 10\,[kg]$ load cell. In the best case the error is below $0.0001\,[kg]$.

4 How to Start

In this part you will learn how to use a *Custom Arduino Libraries* and then how to get data from HX711 with just one command. For more information see [6] and [4].

4.1 Register Custom Library

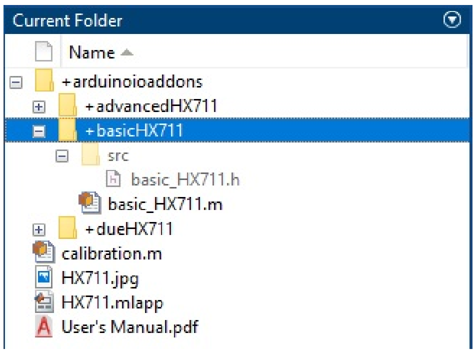
The current folder contains two files named `HX711.m` and `HX711.h`. The first one is the *MATLAB Add-On Class* that inherits from *arduino LibraryBase class* a variety of properties and methods. The C++ Header File is a class that includes `librarybase.h` and the code segments executed on the Arduino device.

First you should extract content from .zip file and paste it in your own personal folder, in the example it has the following path 'C:/work'. In order to register the HX711 custom library you have to add the path of the folder to Matlab search path. You can do that with the following command:

```
addpath('C:\work');
```

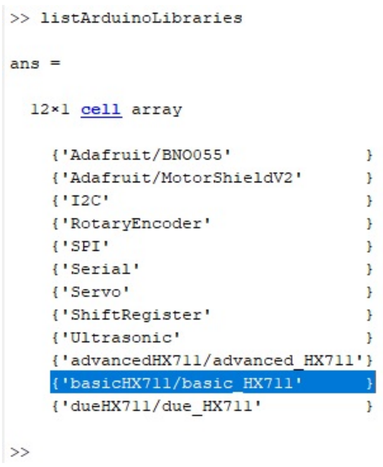
now the Matlab current folder should look like figure 2a. The structure in the Work folder is the following: +arduinoioaddons which contains several folders, one of these is +basicHX711 folder. You can have other +NameLibrary folder in +arduinoioaddons folder. The folder structure is really important so if you have some questions look here `https://it.mathworks.com/help/supportpkg/arduinoio/ug/create-custom-folder-structure.html`. Make sure the basicHX711/basic_HX711 library is available with `listArduinoLibraries` command which produces an output similar to figure 2b.

```
listArduinoLibraries
```



```
Current Folder
Name
+arduinoioaddons
  +advancedHX711
  +basicHX711
    src
      basic_HX711.h
    basic_HX711.m
  +dueHX711
calibration.m
HX711.jpg
HX711.mlapp
User's Manual.pdf
```

(a) *Add-On Package folder;*



```
>> listArduinoLibraries

ans =

  12×1 cell array

    {'Adafruit/BNO055'            }
    {'Adafruit/MotorShieldV2'     }
    {'I2C'                        }
    {'RotaryEncoder'              }
    {'SPI'                        }
    {'Serial'                     }
    {'Servo'                      }
    {'ShiftRegister'              }
    {'Ultrasonic'                 }
    {'advancedHX711/advanced_HX711'}
    {'basicHX711/basic_HX711'     }
    {'dueHX711/due_HX711'         }

>>
```

(b) *List of Arduino libraries;*

Figure 2

4.2 Upload Arduino Server

Now you have to initialize the connection between Matlab and Arduino. You can do that in two ways. The first one consists of using the following command, which requires as argument the Arduino Board you're currently using and the name of serial port (*UNO* and *COM6* in the example).

```
a=arduino('com6','Uno','libraries','basicHX711/
    basic_HX711');
```

The second way consists of using the hardware setup procedure which is initialized by the following command:

```
arduinosetup
```

it is shown in figure 3 step by step.

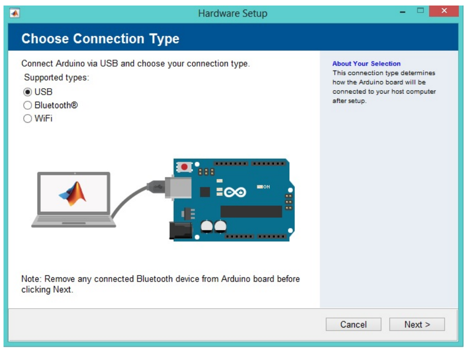
Now you can start creating the objects of the classes: figure 4 shows the output you should obtain. If an error message occures see section 5.

```
a=arduino('com6','Uno') // If you've used Hardware Setup;
// You must specify the digital pin you've chosen:
// in the example I use the digital pins 2 and 3;
LoadCell=addon(a,'basicHX711/basic_HX711',{'D2','D3'})
```

Finally you can get data from HX711 with the following command:

```
read_HX711(LoadCell)
```

Now you can write your own function to calibrate the loadcell or use the `calibration` class (section 6); you can also use the HX711 app (section 7).

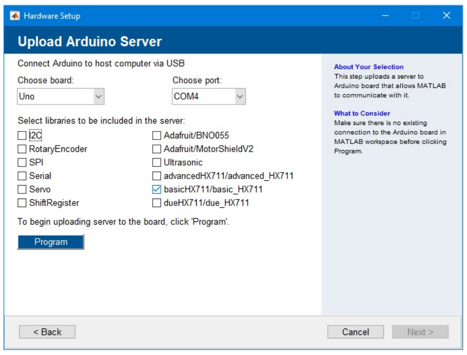


Connect Arduino via USB and choose your connection type.
Supported types:
[✓ USB]
Bluetooth®
WiFi

Note: Remove any connected Bluetooth device from Arduino board before clicking Next.

About Your Selection
This connection type determines how the Arduino board will be connected to your host computer after setup.

(a) *Choose connection type;*



Connect Arduino to host computer via USB
Choose board: Uno
Choose port: COM4

Select libraries to be included in the server:

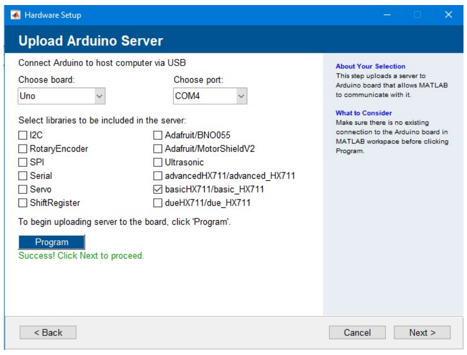
I2C	Adafruit/BNO055
RotaryEncoder	Adafruit/MotorShieldV2
SPI	Ultrasonic
Serial	advancedHX711/advanced_HX711
Servo	✓ basicHX711/basic_HX711
ShiftRegister	dueHX711/due_HX711

To begin uploading server to the board, click 'Program'.

About Your Selection
This step uploads a server to Arduino board that allows MATLAB to communicate with it.

What to Consider
Make sure there is no existing connection to the Arduino board in MATLAB workspace before clicking Program.

(b) *Choose the board and the port, then make sure basicHX711/basic_HX711 library is selected;*



Connect Arduino to host computer via USB
Choose board: Uno
Choose port: COM4

Select libraries to be included in the server:

I2C	Adafruit/BNO055
RotaryEncoder	Adafruit/MotorShieldV2
SPI	Ultrasonic
Serial	advancedHX711/advanced_HX711
Servo	✓ basicHX711/basic_HX711
ShiftRegister	dueHX711/due_HX711

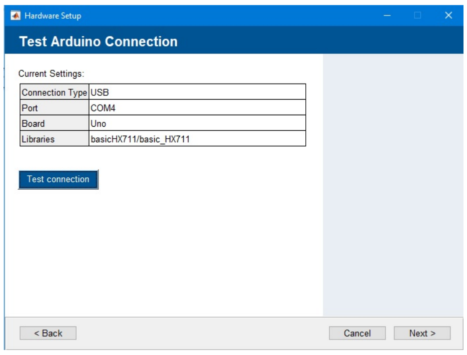
To begin uploading server to the board, click 'Program'.

Success! Click Next to proceed.

About Your Selection
This step uploads a server to Arduino board that allows MATLAB to communicate with it.

What to Consider
Make sure there is no existing connection to the Arduino board in MATLAB workspace before clicking Program.

(c) *Upload arduino server with Program button then click next;*



Current Settings:

Connection Type	USB
Port	COM4
Board	Uno
Libraries	basicHX711/basic_HX711

(d) *Make sure the basicHX711/ basic_HX711 library is in the list, then proceed;*

Figure 3

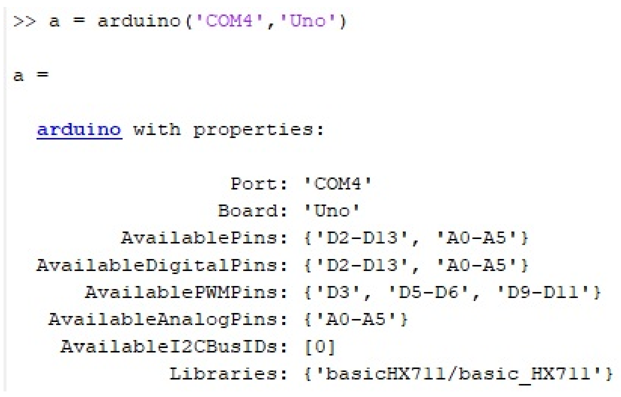
5 Troubleshooting

If Custom Arduino Library class is not detected visit *Custom Arduino Library Issues* page: `https://it.mathworks.com/help/supportpkg/arduinoio/ug/custom-arduino-library-issues.html`.

6 Calibration Class

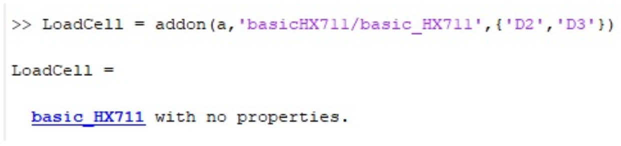
In this part you'll see an example of the calibration phase. First I get the tare weight T, then I place a known weight on the load cell in order to calculate the scale factor: the result of readings P is devided by your known weight P_t.

$$f_{scale} = \frac{P - T}{P_t} \tag{1}$$



```
>> a = arduino('COM4','Uno')

a =

  arduino with properties:

                   Port: 'COM4'
                  Board: 'Uno'
          AvailablePins: {'D2-D13', 'A0-A5'}
   AvailableDigitalPins: {'D2-D13', 'A0-A5'}
       AvailablePWMPins: {'D3', 'D5-D6', 'D9-D11'}
    AvailableAnalogPins: {'A0-A5'}
     AvailableI2CBusIDs: [0]
              Libraries: {'basicHX711/basic_HX711'}
```

(a)



```
>> LoadCell = addon(a,'basicHX711/basic_HX711',{'D2','D3'})

LoadCell =

  basic_HX711 with no properties.
```

(b)

Figure 4

Now you can check the calibration quality by a raw reading or start making measurements, if P is the result of a raw reading you can calculate weight W with the following formula:

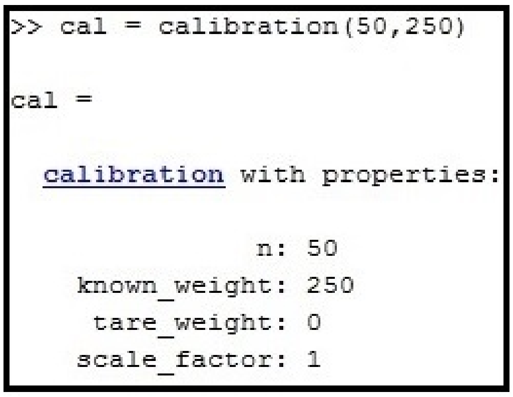
$$W = \frac{P - T}{f_{scale}} \tag{2}$$

In order to get more precise results I use the average of n readings instead of a single reading.

In the `calibration` class there are all the functions you need to calibrate the loadcell wich implement the previous formulas.

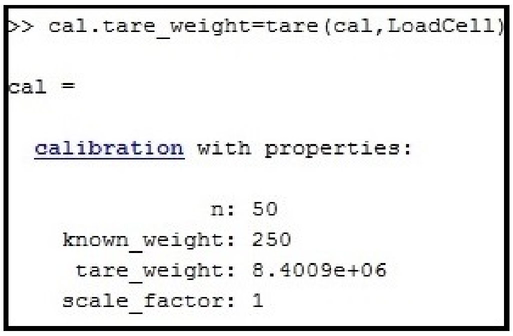
1. Create the object of the calibration class (figure 5a);
2. Use the `tare_weight` function (figure 5b);
3. Use the `scale_factor` function (figure 5c);
4. Now use the `get_weight` function, you can also add an argument which is a positive scalar so you'll obtain the average of multiple readings (figure 5d);
5. If you are interested in how your loadcell behaves with a specific static load in term of precision you can use the `stat` function in order to get the average and the standard deviation of multiple readings (figure 6a);

6. Finally if you are interested in a visual representation of the `stat` function you can use the `plot_data` function (figure 6b);



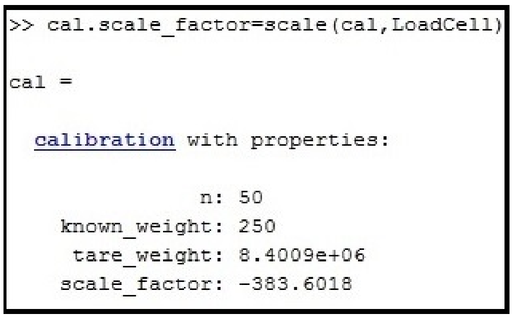
```
>> cal = calibration(50,250)

cal =

  calibration with properties:

               n: 50
    known_weight: 250
     tare_weight: 0
    scale_factor: 1
```

(a) *Input argument: 1- Number of Readings; 2- Known weight.*



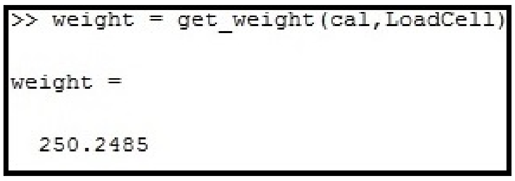
```
>> cal.tare_weight=tare(cal,LoadCell)

cal =

  calibration with properties:

               n: 50
    known_weight: 250
     tare_weight: 8.4009e+06
    scale_factor: 1
```

(b) *Input argument: 1- Calibration object; 2- HX711 object.*



```
>> cal.scale_factor=scale(cal,LoadCell)

cal =

  calibration with properties:

               n: 50
    known_weight: 250
     tare_weight: 8.4009e+06
    scale_factor: -383.6018
```

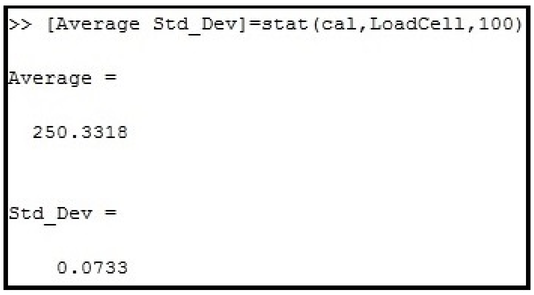
(c) *Input argument: 1- Calibration object; 2- HX711 object.*



```
>> weight = get_weight(cal,LoadCell)

weight =

  250.2485
```

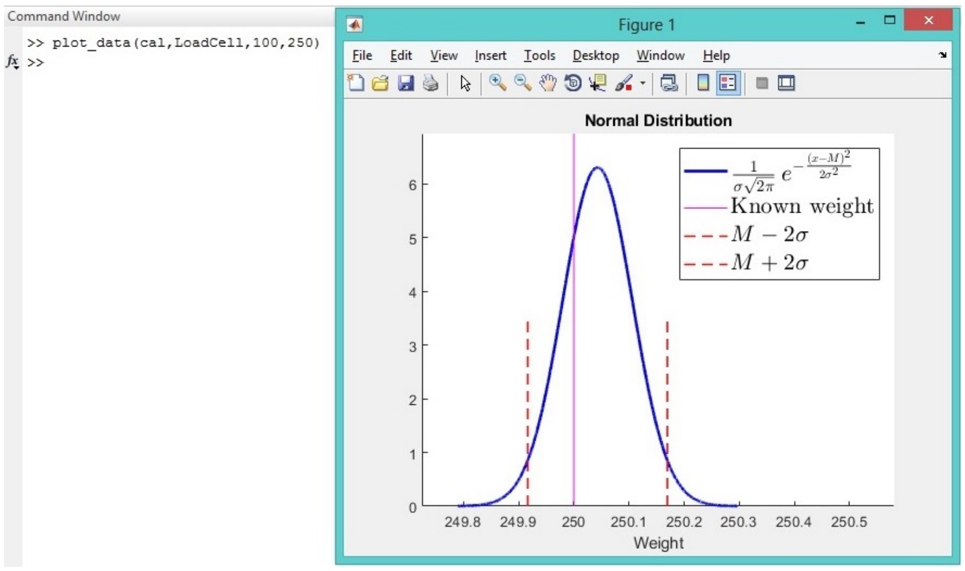
(d) *Input argument: 1- Calibration object; 2- HX711 object; 3-(optional) Number of readings.*

Figure 5



```
>> [Average Std_Dev]=stat(cal,LoadCell,100)

Average =

  250.3318


Std_Dev =

    0.0733
```

(a) *Input argument: 1- Calibration object; 2- HX711 object; 3- Number of readings.*



```
Command Window
  >> plot_data(cal,LoadCell,100,250)
fx >>
```

(b) *Input argument: 1- Calibration object; 2- HX711 object; 3- Number of readings; 4- Known Weight*

Figure 6

7 HX711 App

In this section you'll learn step by step how to use the HX711 Matlab App (`HX711.mlapp`) which can be used to calibrate a loadcell, plot the data in real time and finally save them in `.txt` or `.xls` file.

1. Type HX711 in the command window to open the app;

2. Set properly the connection settings and then click on the CONNECT button (figure 7);

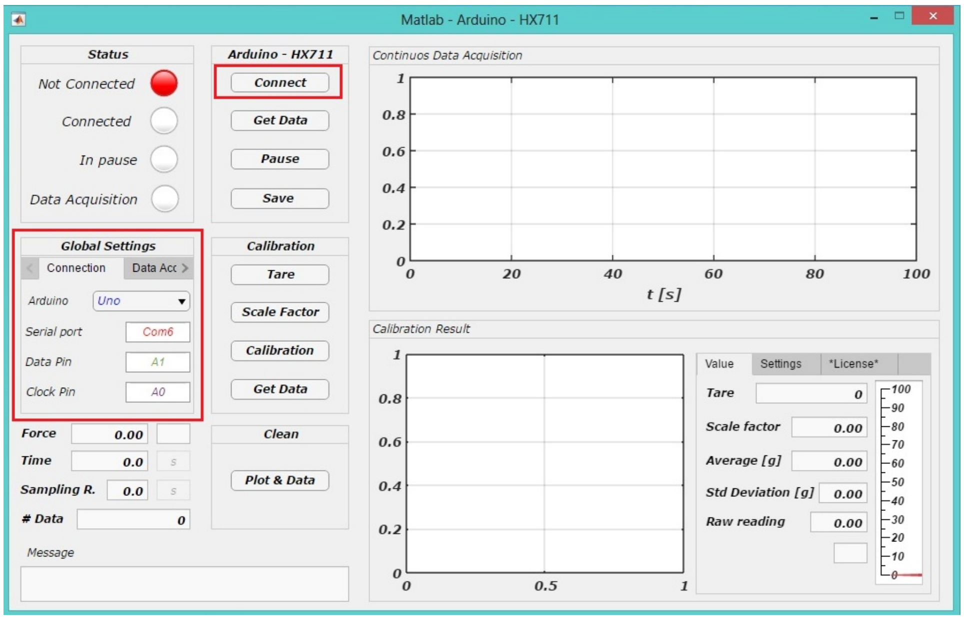
3. Set the number of readings ($50 \div 500$) and the known weight, you can choose the unit and also set the maximum load the loadcell can deal with, thus a linear gauge will show when you reach the limit, then click on the TARE and the SCALE button in that order (wait for data acquisition) so you'll obtain the tare weight and the scale factor (figure 8);

OPT. If you are interested in how your loadcell behaves with a specific static load in term of precision, set the known weight and click on the CALIBRATION button (figure 9);

4. You can start getting data by clicking on the GET DATA button in the upper or a single raw read by the GET DATA button in the bottom; thanks to PAUSE button you can stop data acquisition and start again later; there are also two main settings: the sampling rate which has the minimum value of $0.1\,[s]$ and the session time which allows you to set the time beyond which data acquisition stops (figure 10);

5. Now you can save all data you've collected (by clicking on the GET DATA button in the upper): set up the name and extention then click on the SAVE button. Now you can start getting data again. If you'll click on the SAVE button again you'll actually save data you've collected in the last session since you've saved data the previous time. Note that time array doesn't start from zero in the file you've created. If you want to set to zero the beginning of the time in every further file click on the RESET TIME check box (figure 11).

6. If you need to clean up data and the graphs just click on the PLOT & DATA button (now you can start getting data again maybe with a different unit) (figure 12).



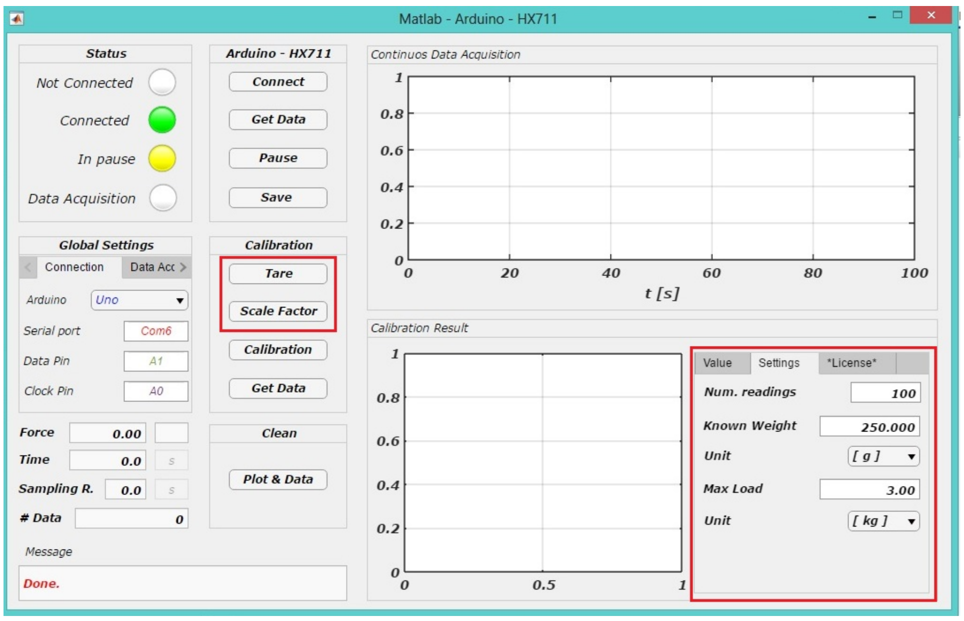
Not Connected ●
Connected
In pause
Data Acquisition

Global Settings — Connection | Data Acc

Arduino	Uno
Serial port	Com6
Data Pin	A1
Clock Pin	A0
Force	0.00
Time	0.0 s
Sampling R.	0.0 s
# Data	0

Tare	0
Scale factor	0.00
Average [g]	0.00
Std Deviation [g]	0.00
Raw reading	0.00

Figure 7



Not Connected
Connected ●
In pause ●
Data Acquisition

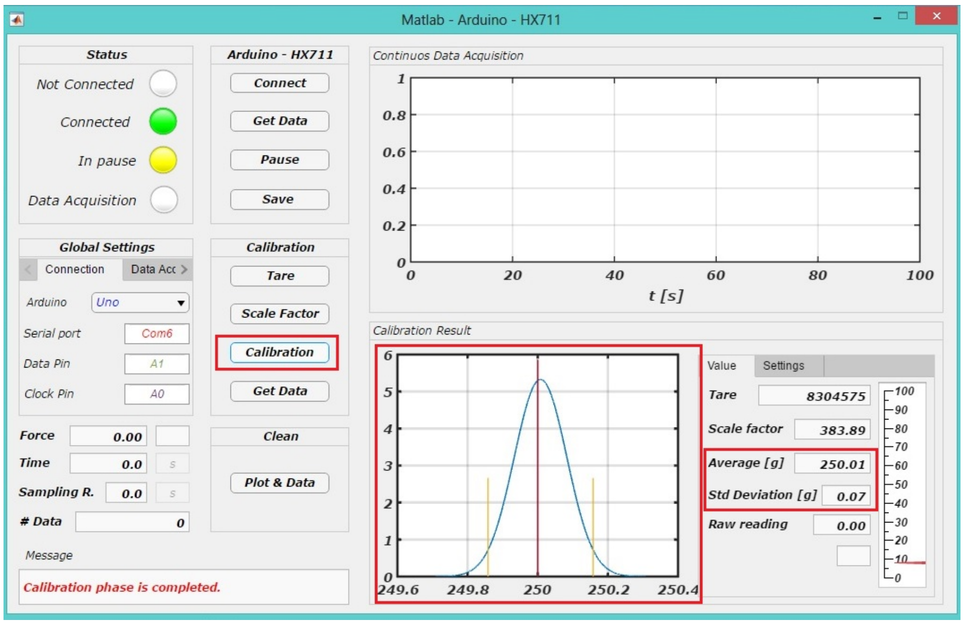
Global Settings — Connection | Data Acc

Arduino	Uno
Serial port	Com6
Data Pin	A1
Clock Pin	A0
Force	0.00
Time	0.0 s
Sampling R.	0.0 s
# Data	0

Num. readings	100
Known Weight	250.000
Unit	[g]
Max Load	3.00
Unit	[kg]

Message: Done.

Figure 8



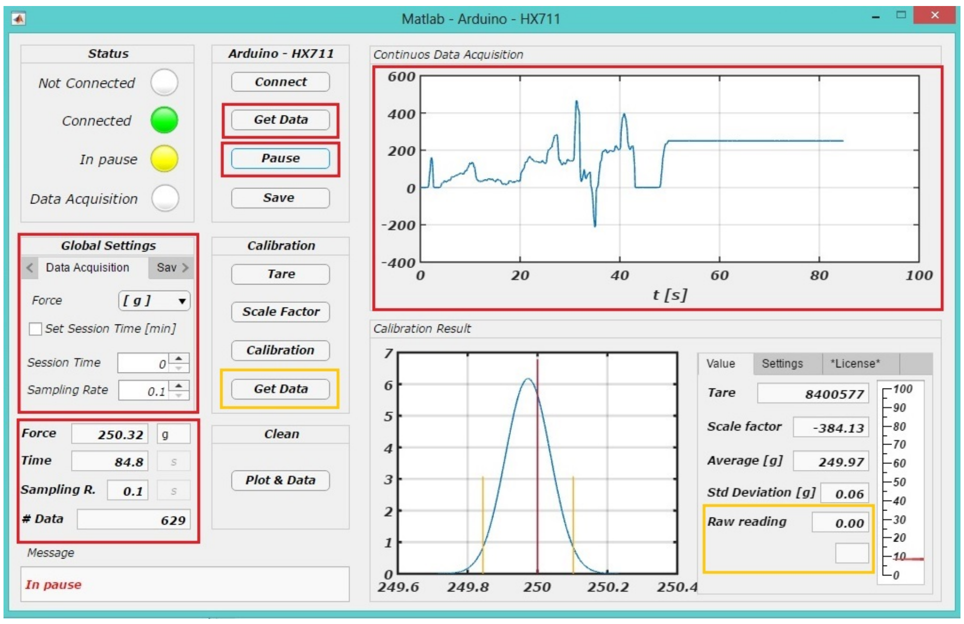
Arduino: Uno
Serial port: Com6
Data Pin: A1
Clock Pin: A0

Force: 0.00
Time: 0.0 s
Sampling R.: 0.0 s
Data: 0

Message: Calibration phase is completed.

Tare: 8304575
Scale factor: 383.89
Average [g]: 250.01
Std Deviation [g]: 0.07
Raw reading: 0.00

Figure 9



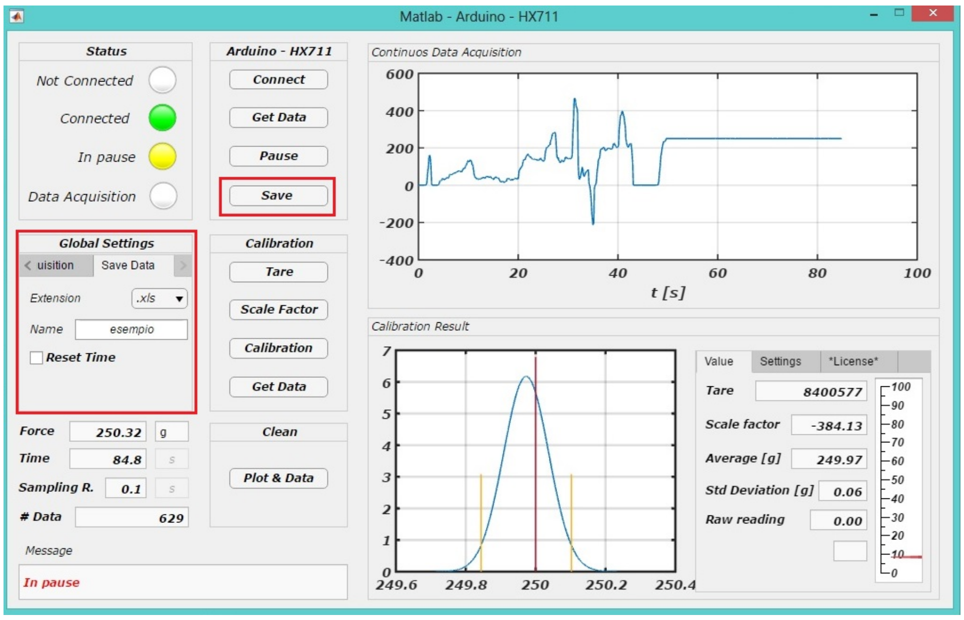
Force: [g]
☐ Set Session Time [min]
Session Time: 0
Sampling Rate: 0.1

Force: 250.32 g
Time: 84.8 s
Sampling R.: 0.1 s
Data: 629

Message: In pause

Tare: 8400577
Scale factor: -384.13
Average [g]: 249.97
Std Deviation [g]: 0.06
Raw reading: 0.00

Figure 10



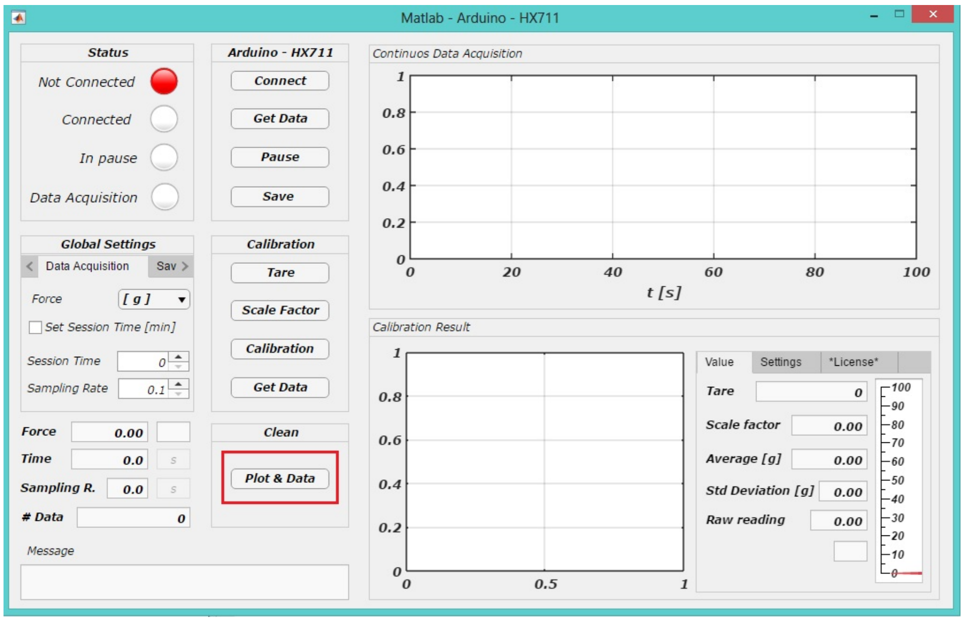
Extension .xls
Name esempio
Reset Time

Force 250.32 g
Time 84.8 s
Sampling R. 0.1 s
Data 629

Tare 8400577
Scale factor -384.13
Average [g] 249.97
Std Deviation [g] 0.06
Raw reading 0.00

Message: In pause

Figure 11



Force [g]
Set Session Time [min]
Session Time 0
Sampling Rate 0.1

Force 0.00
Time 0.0 s
Sampling R. 0.0 s
Data 0

Tare 0
Scale factor 0.00
Average [g] 0.00
Std Deviation [g] 0.00
Raw reading 0.00

Figure 12

References

[1] ARDUINO. *Arduino MEGA2560 rev3*. Documentation. 2017. URL: `https://store.arduino.cc/arduino-mega-2560-rev3`.

[2] ARDUINO. ***shiftIn()***. Reference, Language, Functions, Advance IO. 2017. URL: `https://www.arduino.cc/reference/en/language/functions/advanced-io/shiftin/`.

[3] IEEE. *Guide to the Software Engineering*. 2004.

[4] MathWorks. *Add-on Package Folder*. 2018. URL: `https://it.mathworks.com/help/supportpkg/arduinoio/ug/create-custom-folder-structure.html`.

[5] MathWorks. *Custom Add-On Library Concepts*. 2018. URL: `https://it.mathworks.com/help/supportpkg/arduinoio/ug/custom-library-concepts.html`.

[6] MathWorks. *Custom Arduino Libraries*. 2018. URL: `https://it.mathworks.com/help/supportpkg/arduinoio/custom-arduino-libraries.html`.

[7] MathWorks. *Object-Oriented Programming*. MATLAB. 2017.

[8] AVIA Semiconductor. *24-Bit Analog-to-Digital Converter (ADC) for Weigh Scales*. 2017. URL: `https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/hx711_english.pdf`.