# Open ForceSense MR: An Open-Hardware MR-Compatible Force Sensor

*Francesco Santini*

## Contents



## 1 Overview

This document contains the instructions for the building of the OpenForceSense MR force sensor, an MR-Compatible force sensor composed of cheap, off-the shelf components. This sensor, the associated files, the arduino firmware, and the host software included in this repository is open-source, distributed under the GNU General Programming License (GPL) v3.

In this current embodiment, the sensor is composed of 4 20-kgf parallel beam load cells mounted on a custom aluminum frame, thus having a maximum load capacity of 80kgf (784N), and an arduino shield built around the standard HX711 amplifier.

In order to make the sensor MR-compatible, nonmagnetic materials are used, and accurate electromagnetic shielding is used to minimize the effects of interference from the MR components.

## 2 Building the sensor head

Please refer to the attached bill-of-materials (BOM) to acquire the required components. The prices indicated in the BOM are for indication and they refer to generic sources available on eBay.

### 2.1 Aluminum frame

The aluminum frame is composed of a top and bottom plates (one with 5-mm holes, the other with 4mm), 4 spacers with two 4-mm holes each, and 4 spacers with two 5-mm holes each. The single pieces can be obtained from a single 150x150mm piece of aluminum (thickness: 4mm) using a desktop CNC router/milling machine (we used a Shapeoko 3 with a Kress 1050E spindle), or by

manual machining following the provided svg file (see fig 2.1). Manually chamfer the holes of the top and bottom plates in order to have the countersunk screws fit at the level of the surface.

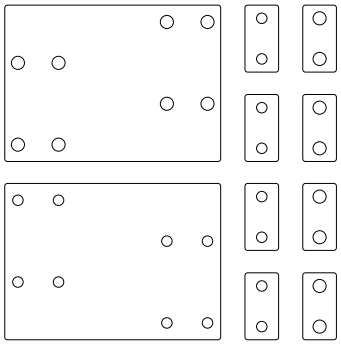

Fig. 2.1: Shape for machining the aluminum parts.

The beam cells work by measuring the deformation in the longitudinal direction, therefore they appropriately measure the torque applied between the two sides of the cell, rather than the force *per se*. For this reason, each cell must be mounted in a way that allows the deformation. Assemble the four cells on the frame using the spacers and the M4 and M5 brass screws as shown in figure 2.2 (a 3D model as FreeCAD file is also provided). Make sure to route the cables in a way that they all come out from a single slit between two cells.

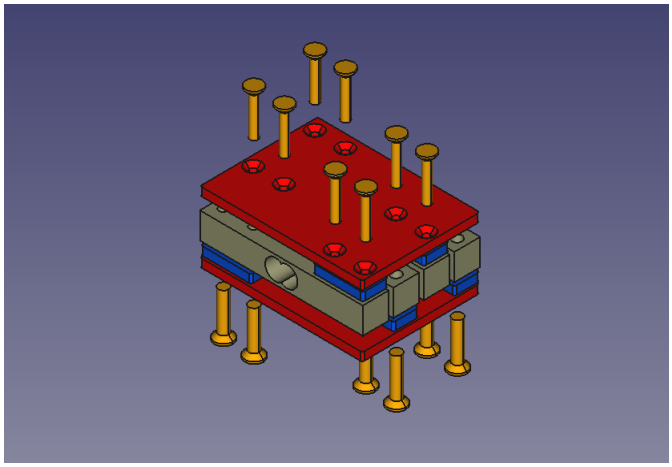

Fig. 2.2: Mechanical assembly.

## 2.2 Electrical connections

Cut all the cables coming out of the slit at the same length and solder the cables of the same color together. This ensures a parallel connection of the Wheatstone bridges that can therefore act as a single sensor.

Cut the connector from one end of the ethernet patch cable and expose the internal cables. Cut approximately 20-cm-long piece of the copper braid and slide it over the ethernet cable (push the extremities of the braid to enlarge the hole).

Connect the sensor cables to two twisted-pairs of the ethernet cable, and cover the connections with heatshrink tube. We used the following connections (sensor -> cable):

- Red -> Orange
- Black -> Orange/White
- Green -> Green
- White -> Green/White

Fig. 2.3: Mounted sensor with shielding braid (left) and connection of the braid to the cable shielding (right)

Slide the braid over the sensor cables and attach it to the aluminum frame by securing it between a spacer and a plate. At the other end of the braid, remove a piece of insulation from the ethernet cable and fix the braid to the shielding with a cable tie (see figure 2.3). For extra shielding, the whole sensor can also be wrapped in aluminium foil.

## 3 Building the Arduino shield

The Arduino shield is a simple shield built around the HX711 breakout board that does not need any additional discrete component. Solder the HX711 and a female RJ45 connector on the shield prototyping board and connect (see fig. 3.1):

- HX711 Digital side (from Arduino to HX711):
  - +5V -> Vcc
  - GND -> GND
  - Pin 6 -> CLK
  - Pin 7 -> DT
- HX711 Analog side (from HX711 to Ethernet connector):
  - E+ -> Orange cable
  - E- -> White/orange cable
  - A+ -> White/green cable
  - A- -> Green cable

Solder the headers on the appropriate holes of the board in order to create a pass-through shield.

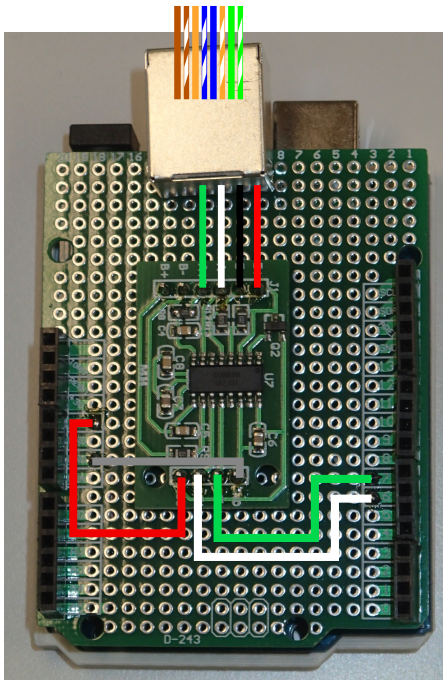

Fig. 3.1: Picture of the Arduino shield.

# 4 Software and calibration

## 4.1 Arduino firmware

The arduino firmware is based on the HX711 library (https://github.com/bogde/HX711). Download the latest arduino interface from https://www.arduino.cc/, load the sketch and upload it onto the Arduino Uno board. This firmware calculates the zero offset at initialization, and then reads a value from the sensor every 100ms, converts it into some unit using the SCALE factor, and prints it to serial.

The sketch can be configured by changing the DATA_PIN and CLK_PIN constants if needed, and the SCALE factor should be calibrated before first use (see below).

## 4.2 Host interface

A simple python program is provided as an interface to the force sensor. It depends on the following installable modules:

- PySerial
- PySide (Qt implementation for Python)
- pyqtgraph (Plotting library for python based on PySide)

Before first use, the pyForceSense.py script should be adapted by defining the SERIAL_PORT constant to the serial port value (OS dependent).

Once started, the program initializes the serial port to the Arduino and starts logging values. In the green box on the left, the current measured value is displayed; in the red box on the right, the maximum value is shown. The last 100 measured values (corresponding to 10s) are shown as a plot on the bottom, and the serial log is displayed on the right side of the window (see figure ).

## 4.3 Calibration

Before first use, the SCALE parameter in the Arduino sketch must be calibrated. The procedure is simple and it involves a known weight or reliable dynamometer. Load the default sketch on the Arduino and open the host interface. After initialization, place the known weight on the sensor. The new scale factor can be calculated as:

$$SCALE_{new} = \frac{SCALE_{old} \cdot WEIGHT_{measured}}{WEIGHT_{known}}$$

Note that the scale is inversely proportional to the weight. Replace the new SCALE value in the Arduino sketch and upload the firmware again.

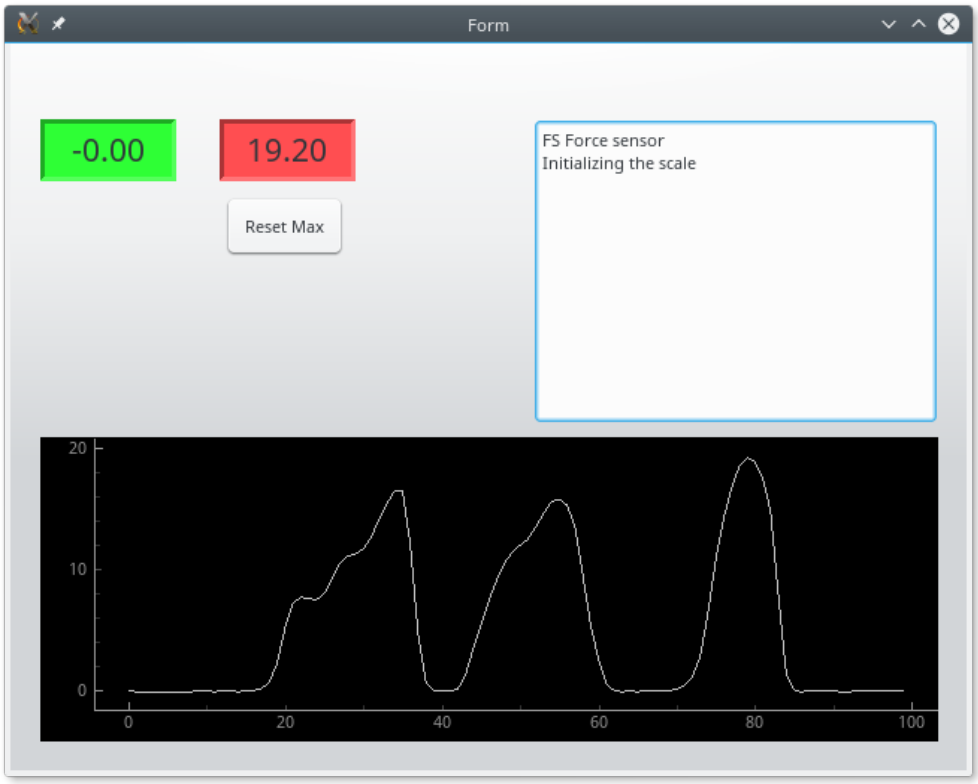


Fig. 4.1: Screenshot of the host interface.