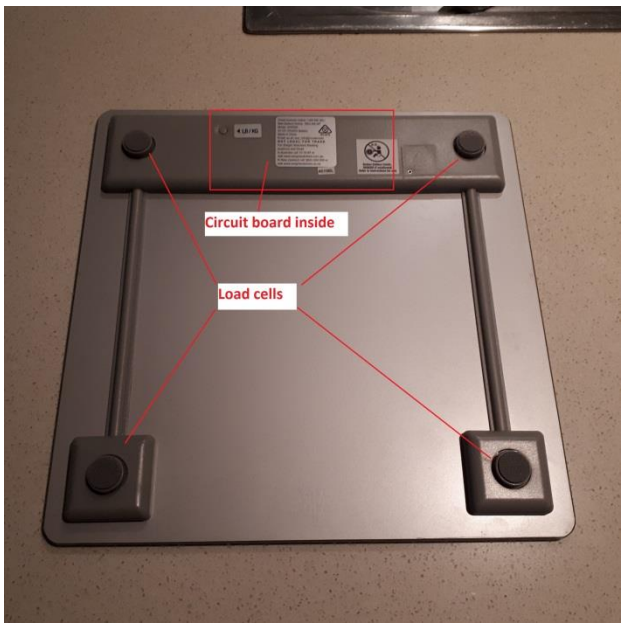
How to build a set of Wheelchair Scales

Mark I

I had a problem to solve – how to weigh a person who cannot stand on a set of bathroom scales unassisted? After discarding some wild plans about installing strain gauges (aka load cells) under the 4 feet of the bed, I thought of starting from an ordinary set of bathroom scales.

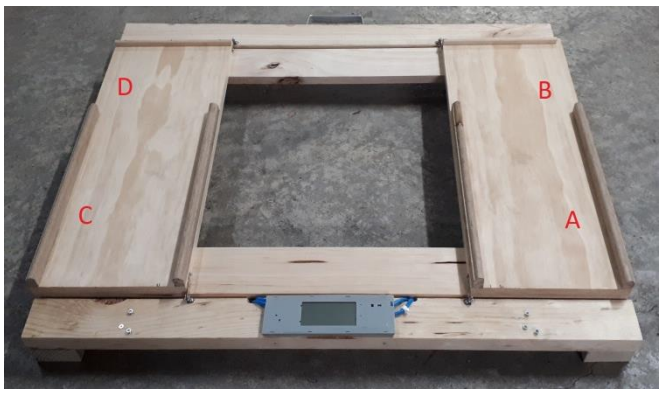


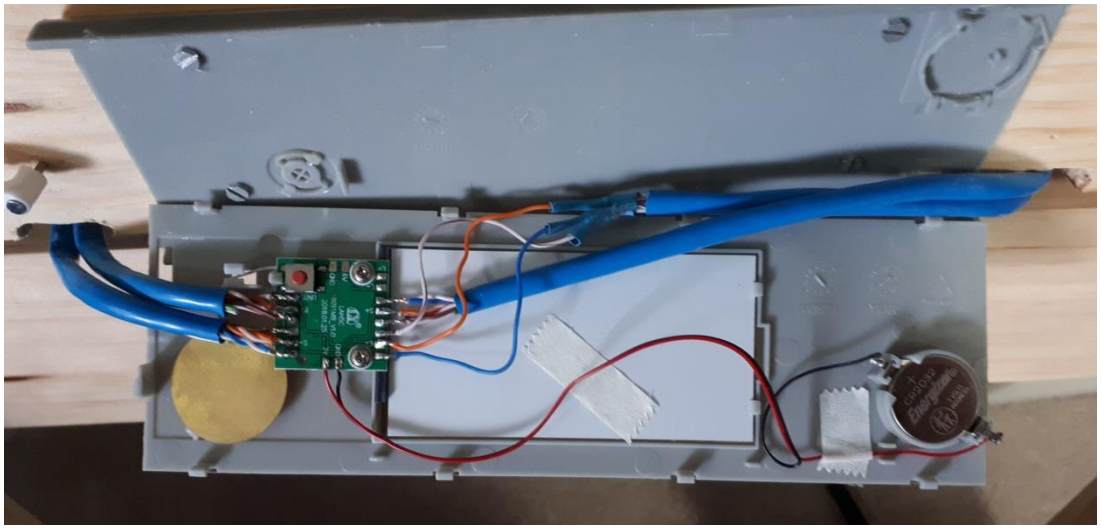
After Building a frame with pine wood, I dismantled a $25 bathroom scales, unglued the 4 load cells from the glass plate, extended the cells 3 wires through extension cables, and Voila!

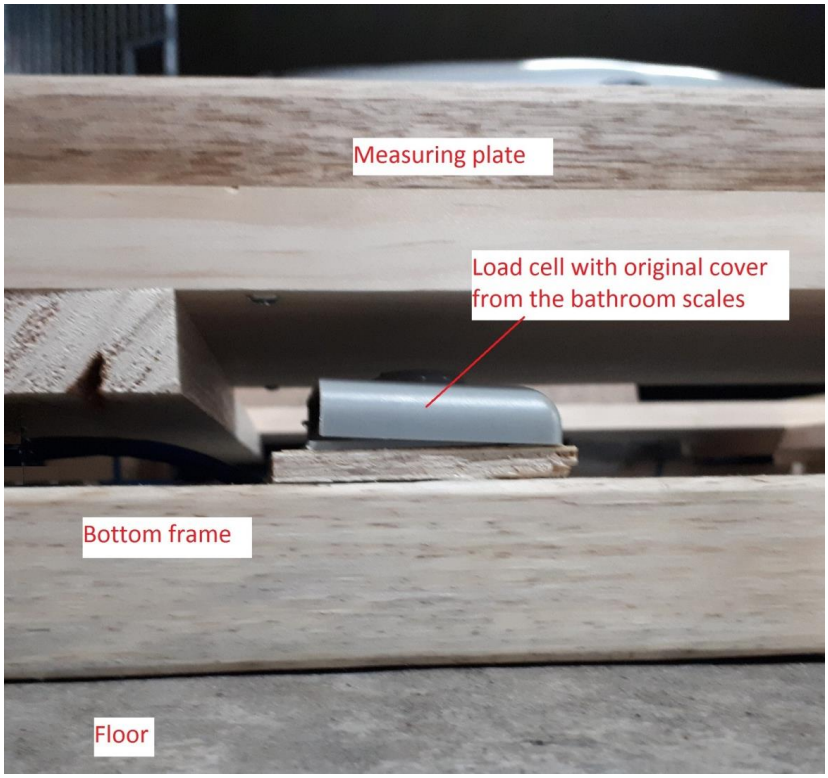
This set of scales worked for a while but became unreliable. Successive measurement would sometimes have a difference of up to 3kg. On these scales, the load cells were underneath the frame on which the wheelchair would sit.

Mark II

So, I decided to get another set of bathroom scales and build another frame. This one has the load cells glued on top of the frame, and a measuring plate resting on them.

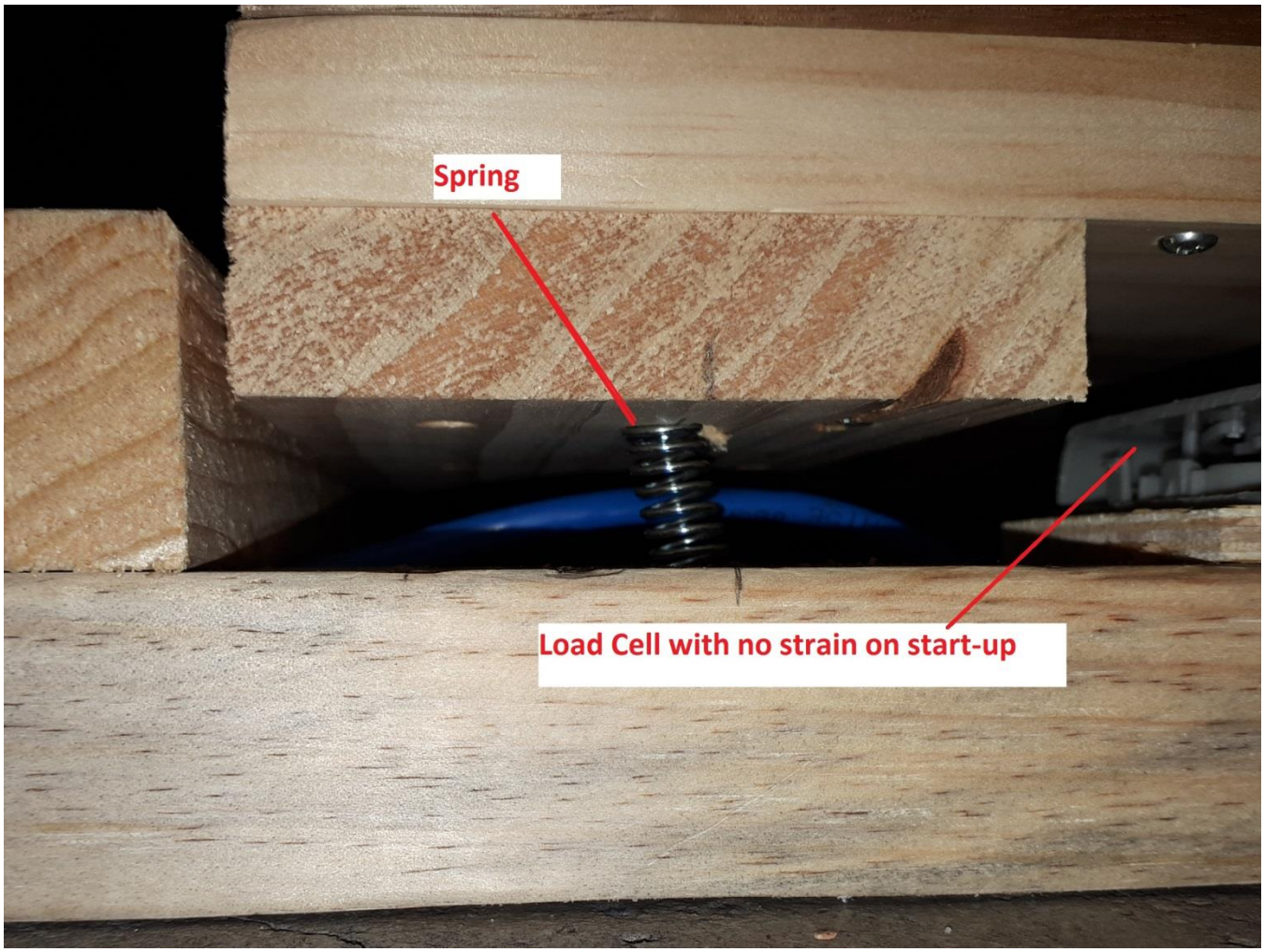






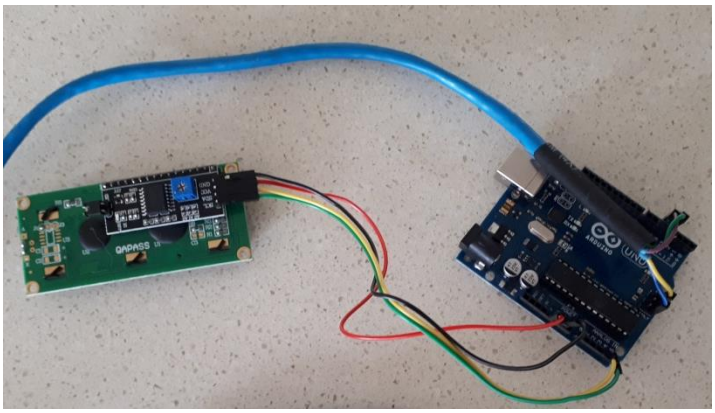
The problem with this one is that the measuring plate being a few kg in weight, the original bathroom scales circuit could not work out the zero. This did not happen with the first bathroom scales, but this one was too smart for its own good! It would display **Err** just after start-up.

So I installed a set of springs to support the frame on start-up. It worked, in the sense that it managed to get to zero. Of course you had to subtract the weight of the frame after a measurement. But as you have to subtract the weight of the wheelchair anyway, so, this was no obstacle.



Maybe because of the springs, maybe for other reasons, this circuit also had a margin of error of a few kg out of 70kg (including measuring plate).

Mark III



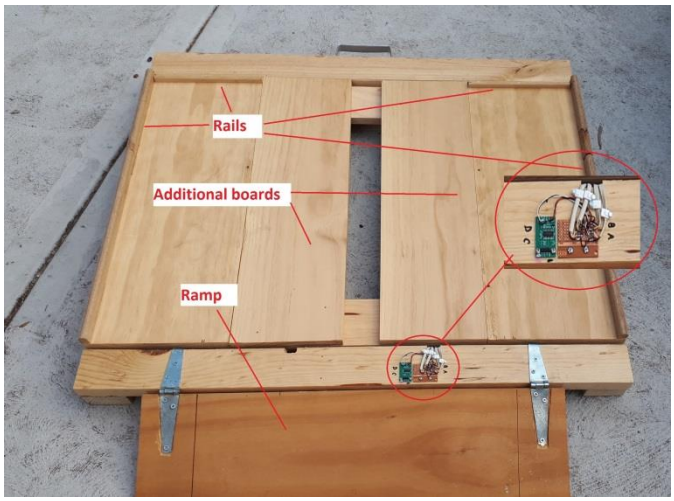
Undefeated, after finding an Arduino Uno R3 and an LCD Display with I2C interface at the bottom of a drawer, and chatting with my son, I decided to have a second try with this new fame. Here are the Arduino and the 2IC LCD Display (face down):

Inspired by Indrek Luuk and his YouTube video **https://www.youtube.com/watch?v=LIuf2egMioA** about building a scales with an Arduino, I decided to keep the frame, remove the springs and build my own circuit.

Indrek, you are a saviour!

The Arduino Uno Analog to Digital Convertors (ADC) convert to a 10 bit integer. So for a full scale of 100kg, the resolution would be 100/(2^10-1)=97 g. However, Indrek recommends using the HZ711 board with a resolution of 24 bits. This gives a resolution of about 6g, well under the expected error of the load cells.

The measuring plate is secured on the sides via a loose arrangement of brackets.



Two new boards were installed in the centre of the measuring plate to prevent the wheelchair to fall in the gap. Small rails were positioned on the outside and end of the measuring plate to prevent the wheelchair to slide out. A ramp was installed at the front to facilitate pushing the wheelchair onto the scales.

The HZ711 board is wired to the 4 Load Cells via a mustering Vero-board and 4-wire telephone cables.

Using an old aluminium plate, I formed an enclosure for theHZ711, Auduino, display and 9V battery as shown below:

And now for the acid test! Measurement with person + wheelchair, then wheelchair by itself.

Reading: Top line shows raw value from HZ711. Bottom line shows value converted in kg.

Code

```
/*
   --------------------------------------------------------------------------------------
   HX711_ADC
   Arduino library for HX711 24-Bit Analog-to-Digital Converter for Weight Scales
   Olav Kallhovd sept2017
   --------------------------------------------------------------------------------------
*/

/*
   Settling time (number of samples) and data filtering can be adjusted in the config.h file
   For calibration and storing the calibration value in eeprom, see example file "Calibration.ino"

   The update() function checks for new data and starts the next conversion. In order to acheive maximum
effective
   sample rate, update() should be called at least as often as the HX711 sample rate; >10Hz@10SPS,
>80Hz@80SPS.
   If you have other time consuming code running (i.e. a graphical LCD), consider calling update() from
an interrupt routine,
   see example file "Read_1x_load_cell_interrupt_driven.ino".

   This is an example sketch on how to use this library
*/
//YWROBOT
//Compatible with the Arduino IDE 1.0
//Library version:1.1
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27,20,4);  // set the LCD address to 0x27 for a 16 chars and 2 line display

#include <HX711_ADC.h>
#if defined(ESP8266)|| defined(ESP32) || defined(AVR)
#include <EEPROM.h>
#endif

//pins:
const int HX711_dout = 4; //mcu > HX711 dout pin
const int HX711_sck = 5; //mcu > HX711 sck pin

//HX711 constructor:
HX711_ADC LoadCell(HX711_dout, HX711_sck);

const int calVal_eepromAdress = 0;
unsigned long t = 0;

void setup() {
  Serial.begin(19200); delay(10);
  Serial.println();
  Serial.println("Starting...");

  LoadCell.begin();
  //LoadCell.setReverseOutput(); //uncomment to turn a negative output value to positive
  float calibrationValue; // calibration value (see example file "Calibration.ino")
  calibrationValue = 696.0; // uncomment this if you want to set the calibration value in the sketch
#if defined(ESP8266)|| defined(ESP32)
  //EEPROM.begin(512); // uncomment this if you use ESP8266/ESP32 and want to fetch the calibration
value from eeprom
#endif
  //EEPROM.get(calVal_eepromAdress, calibrationValue); // uncomment this if you want to fetch the
calibration value from eeprom

  unsigned long stabilizingtime = 2000; // preciscion right after power-up can be improved by adding a
few seconds of stabilizing time
  boolean _tare = true; //set this to false if you don't want tare to be performed in the next step
  LoadCell.start(stabilizingtime, _tare);
  if (LoadCell.getTareTimeoutFlag()) {
    Serial.println("Timeout, check MCU>HX711 wiring and pin designations");
    while (1);
  }
  else {
```

```
    LoadCell.setCalFactor(calibrationValue); // set calibration value (float)
    Serial.println("Startup is complete");
  }
  {
  lcd.init();                      // initialize the lcd
  lcd.init();
  lcd.backlight();
  }
}

void loop() {
  static boolean newDataReady = 0;
  const int serialPrintInterval = 100; //increase value to slow down serial print activity (default 0)

  // check for new data/start next conversion:
  if (LoadCell.update()) newDataReady = true;

  // get smoothed value from the dataset:
  if (newDataReady) {
    if (millis() > t + serialPrintInterval) {
      float i = LoadCell.getData();
      // Serial.print("Load_cell output val: ");
      lcd.setCursor(0,0);
      lcd.print("Load_cell:      ");
      lcd.setCursor(10,0);
      lcd.print(i);
      float w = 0 - (66*i/2332); // 66kg corresponds to raw reading of 2332 as checked with another
bathroom scales
      lcd.setCursor(0,1);
      lcd.print("Weight:         ");
      lcd.setCursor(7,1);
      lcd.print(w);
      newDataReady = 0;
      t = millis();
    }
  }

  // receive command from serial terminal, send 't' to initiate tare operation:
  if (Serial.available() > 0) {
    char inByte = Serial.read();
    if (inByte == 't') LoadCell.tareNoDelay();
  }

  // check if last tare operation is complete:
  if (LoadCell.getTareStatus() == true) {
    Serial.println("Tare complete");
  }

}
```

Acknowledgments

I will be forever indebted to the Aduino community, and

Olav Kallhovd for the HZ711 library

YWROBOT for the LCD 2IC library

Indrek Luuk for his insightful YouTube video

Eric Borrey

xborrey@gmail.com

Sydney, Australia