Supporting information

A low-cost tabletop tensile tester with optical extensometer

Mogens Hinge[a*], Jeremiah A. Johnson[b*] and Martin L. Henriksen[a,b*]

[a]Plastic and Polymer Engineering, Department of Biological and Chemical Engineering, Aarhus University, Aabogade 40, DK-8200, Aarhus N, Denmark

[b]Department of Chemistry, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139-4307, USA.

Corresponding Author
* M. Hinge: hinge@bce.au.dk, J. A. Johnson: jaj2109@mit.edu, M. Henriksen: lahn@bce.au.dk

Contents

BOM

The frame is made from:

1 V-Slot® NEMA 17 Linear Actuator Bundle (Lead Screw) (1000 mm silver, **plus** NEMA 17, $162.98) https://openbuildspartstore.com/v-slot-nema-17-linear-actuator-bundle-lead-screw/

Includes: 1 V-Slot NEMA 17 Linear Actuator Bundle (Lead Screw), 2 Aluminum Spacers - 6mm, 4 Aluminum Spacers - 3mm, 4 Low Profile Screws M5 - 40mm, 6 Low Profile Screws M5 - 15mm, 1 5 x 8mm Flexible Coupling, 2 Eccentric Spacer - 6mm, 4 Xtreme Solid V Wheel™ Kit (4 Xtreme Solid V Wheel, 8 Ball Bearings - 625 RS, 8 5mm Precision Shims, 4 M5 Nylon Lock Nut), 2 Spacer Block, 1 V-Slot® Gantry Plate – Universal V4 (20 mm x 80 mm x 3 mm), 4 Self Tapping Screw – ¾" length, 4 M3 Cap Head Screws – 45 mm, 2 Ball Bearing 688Z 8x16x5, 2 Shim – 12 x 8 x 1 mm, 2 Lock Collar – 8 mm, 2 Threaded Rod Plate for NEMA 17 Stepper Motor V3, 1 Anti-Backlash Nut Block for 8mm Metric Acme Lead S, 1 NEMA 17 Stepper Motor, 1 V-Slot® 20 x 60 mm Linear Rail 1000 mm, 1 Metric Acme Lead Screw – 8 mm, 3 Aluminum Spacers - 1-1/2"

1 T-Slotted Framing Silver Mounting Foot for 40 mm High Dbl/Quad Rail. (5537T421, $16.31) https://www.mcmaster.com/t-slotted-framing-feet
2 Silver Mounting Foot for 20 mm Ht Single Rail T-Slotted Framing (5537T411, $10.54) https://www.mcmaster.com/t-slotted-framing-feet
6061 Aluminum 1/2" Thick, 8" x 8" (9246K31, $33.07) https://www.mcmaster.com/9246K31
4 Black-Oxide Alloy Steel Socket Head Screw M7 x 1 mm Thread, 20 mm Long (91290A165, $4.38) https://www.mcmaster.com/91290a165

Electronics used are:
1 Arduino Uno - R3 (DEV-11021, $24.95) https://www.sparkfun.com/products/11021
1 Load Cell - 10kg, Straight Bar (TAL220, $8.50) https://www.sparkfun.com/products/13329
1 Load Cell Amplifier (HX711, $9.95) https://www.sparkfun.com/products/13879
1 Big Easy Driver (ROB-12859, $19.95) https://www.sparkfun.com/products/12859
3 Screw Terminals 3.5mm Pitch (2-Pin) (PRT-08084, $0.95) https://www.sparkfun.com/products/8084
1 150W 24V 6.5A Enclosed Switchable AC-to-DC Power Supply (Mean well, $23.40) https://www.jameco.com/z/RS-150-24-MEAN-WELL-150W-24V-6-5A-Enclosed-Switchable-AC-to-DC-Power-Supply_323855.html

The extensometer is a HD Webcam - USB 2.0 Digital Video Webcamera (YoLuke, $10.88)

Calibrations are made with Plastic Tubular Spring Scale, 250g/2.5N and 5000g/50N (Ajax Scientific) https://ajaxscientific.com/

Materials and tools to make the custom made parts:
Black-Oxide Alloy Steel Socket Head Screw M4 x 0.7 mm Thread, 16 mm Long (91290A154, $9.96) https://www.mcmaster.com/91290A154
Black-Oxide Alloy Steel Socket Head Screw M5 x 0.8 mm Thread, 20 mm Long (91290A242, $12.69) https://www.mcmaster.com/91290A242
6061 Aluminum 3/4" Thick x 1-1/4" Wide 1ft long (8975K486, $9.52) https://www.mcmaster.com/8975k486

10 in. Standard Hacksaw ($5.47) https://www.homedepot.com/p/ANVIL-10-in-Standard-Hacksaw-12150/303858480
Metric Steel Tap and Die Set (DeWalt, $29.97) https://www.homedepot.com/p/DEWALT-Metric-Steel-Tap-and-Die-Set-17-Pieces-DWA1450/204787392

Custom-made components - drawings

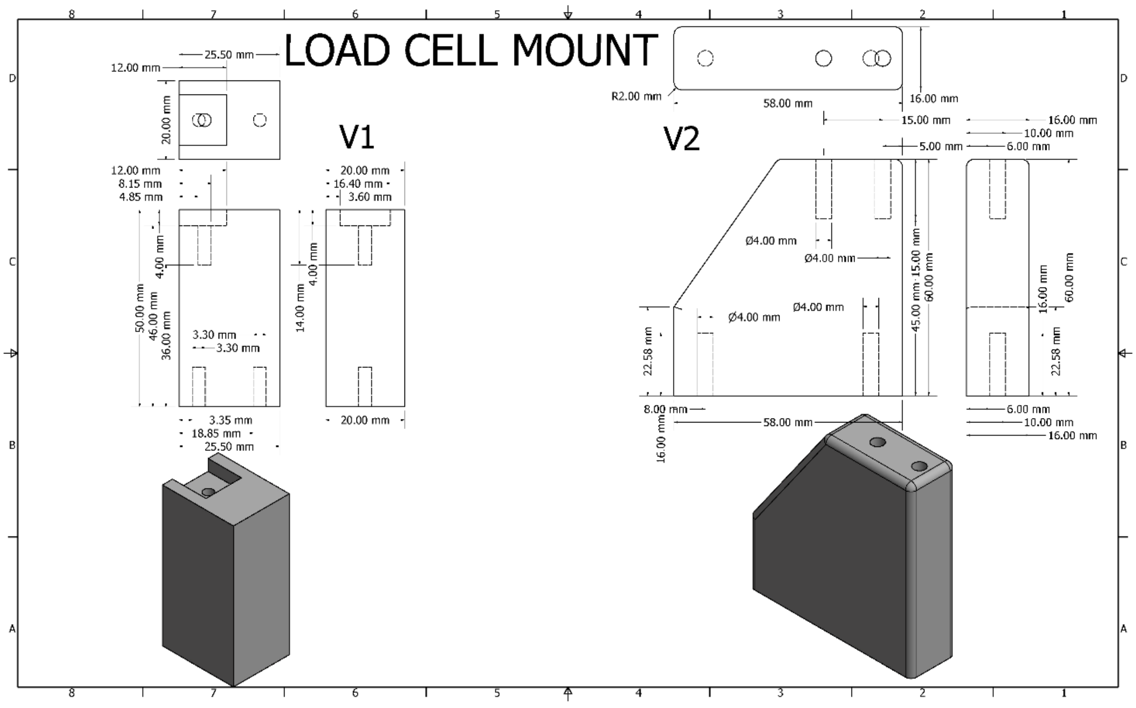


Figure S1 Custom-made load cell holders.

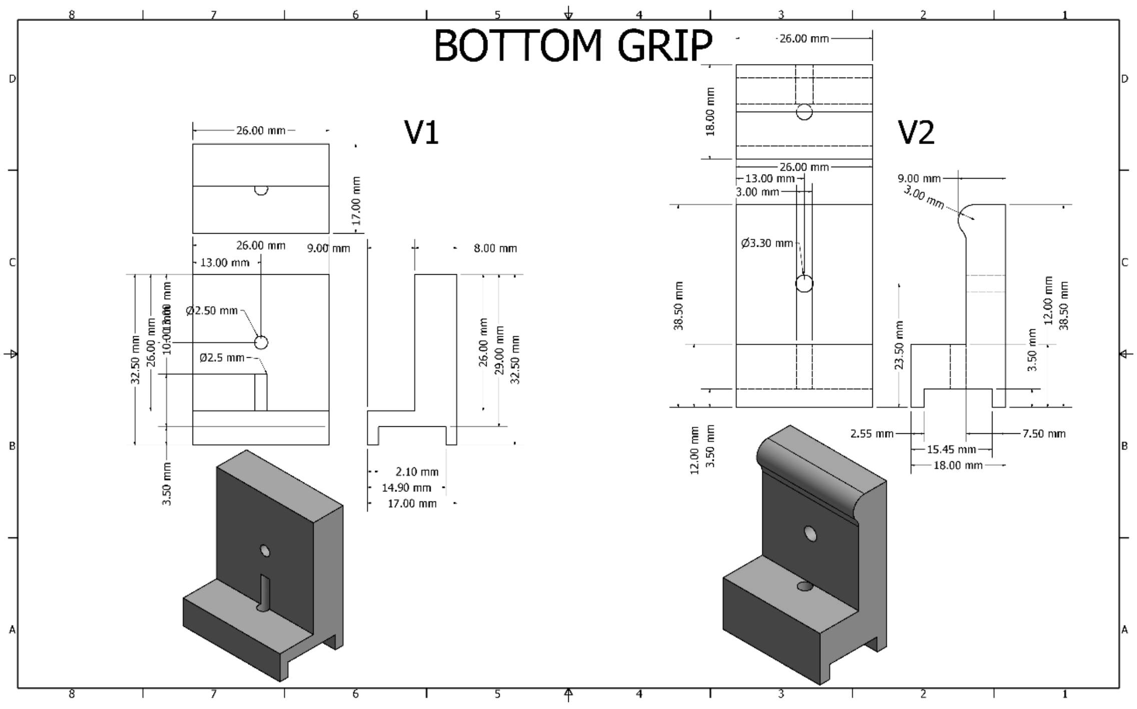


Figure S2 Custom-made bottom grips mounted on the load cell bar.

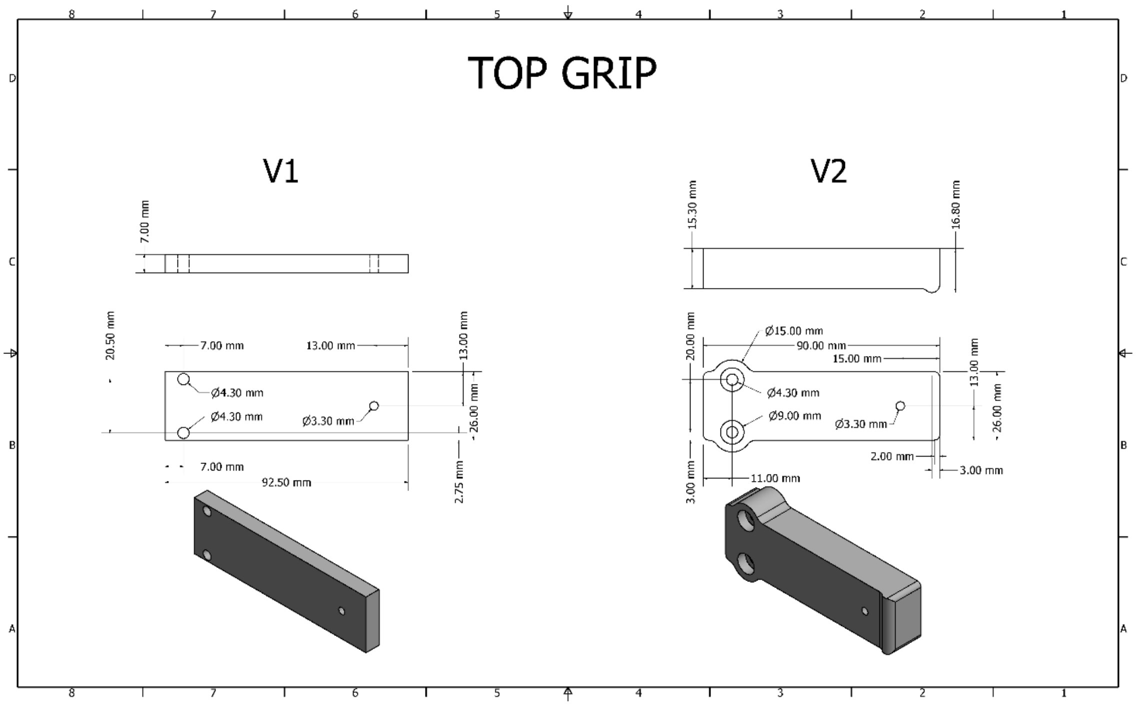


Figure S3 Custom-made top grips mounted on the gantry plate.

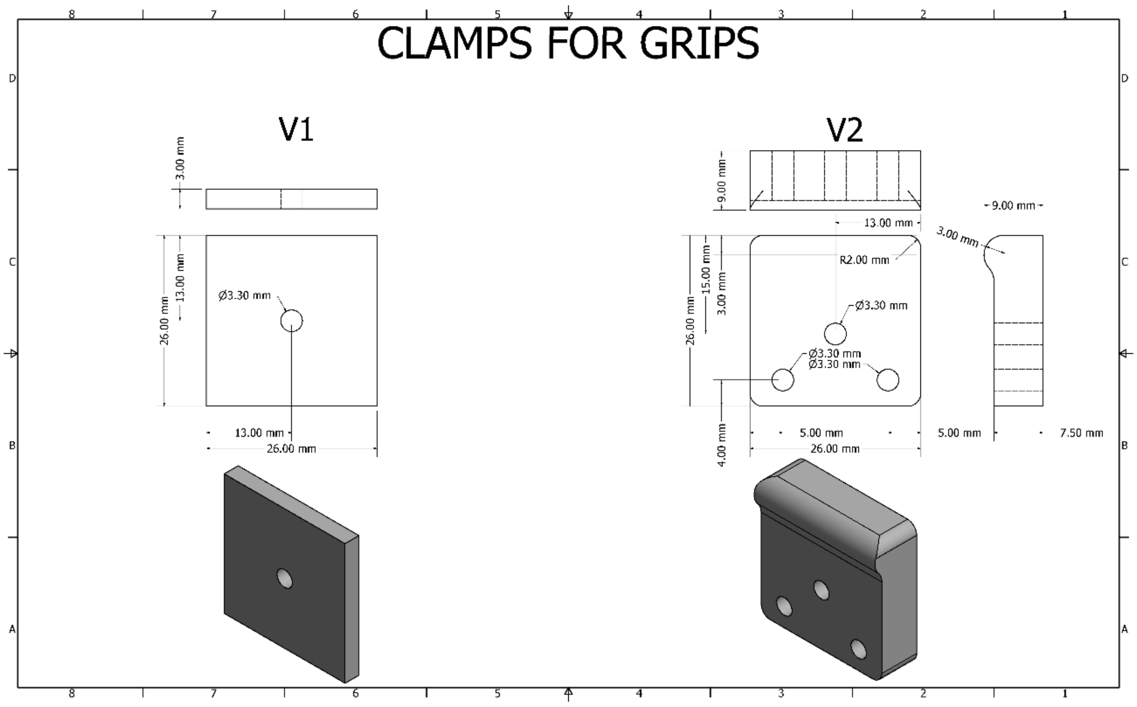


Figure S4 Custom-made grips used as both the top and bottom grips.

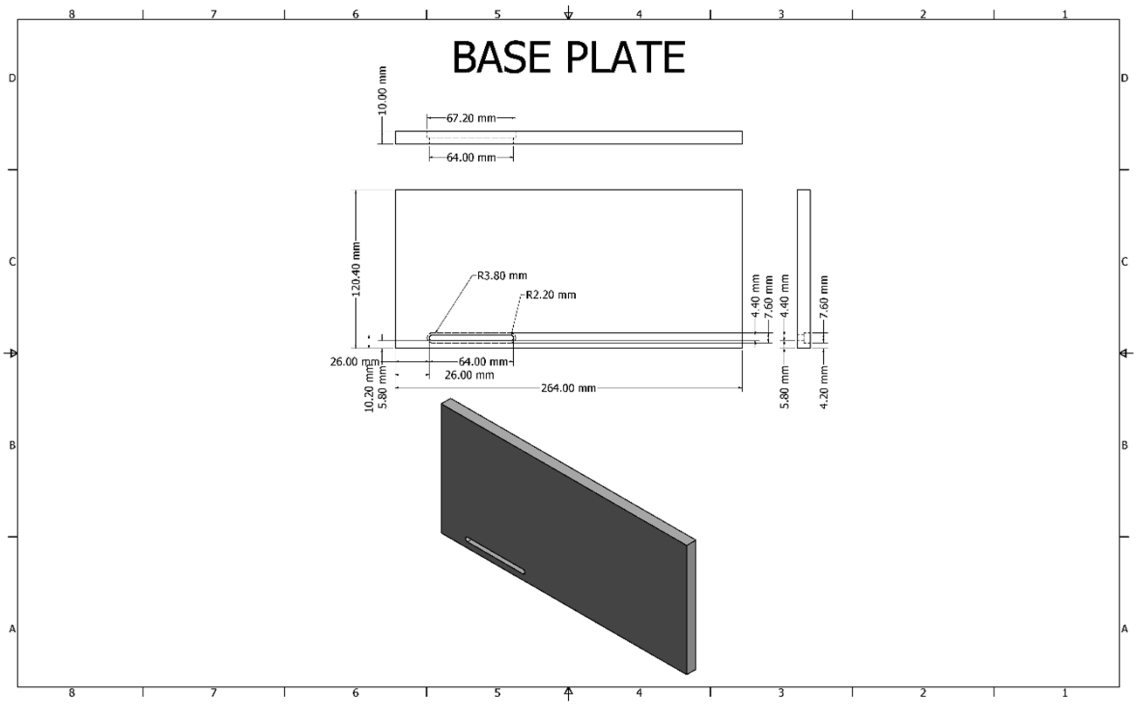
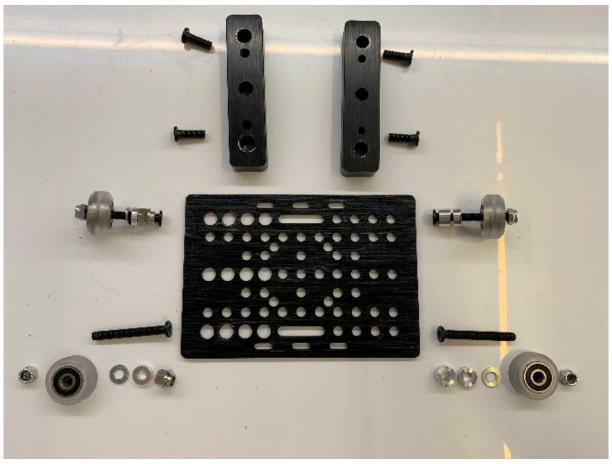
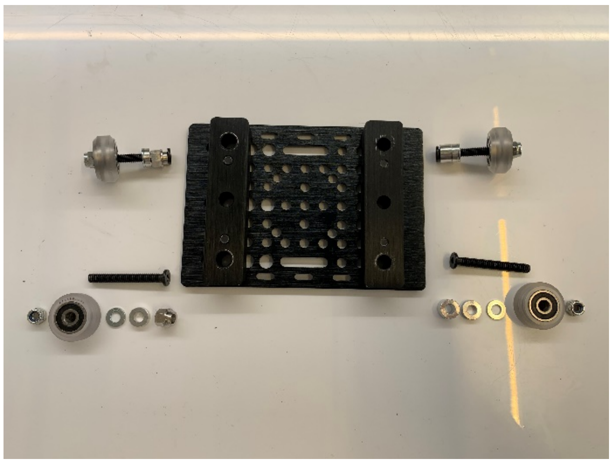
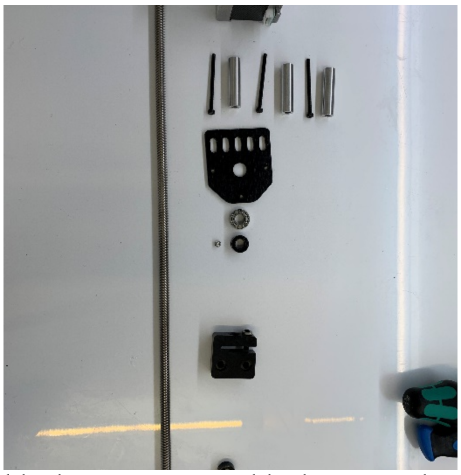


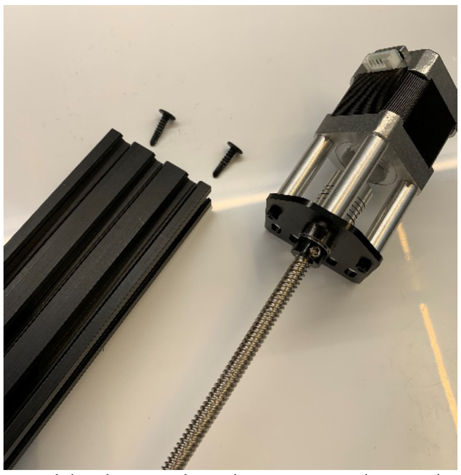
Figure S5 Custom-made base for mounting the tensile tester and load cell unit on. It has a slide mechanism made for M5 to adjust one direction. It is recommended to mount the load cell before the universal tensile tester to make a perfect alignment; where after small adjustments can be done to align with the sliding mechanism.

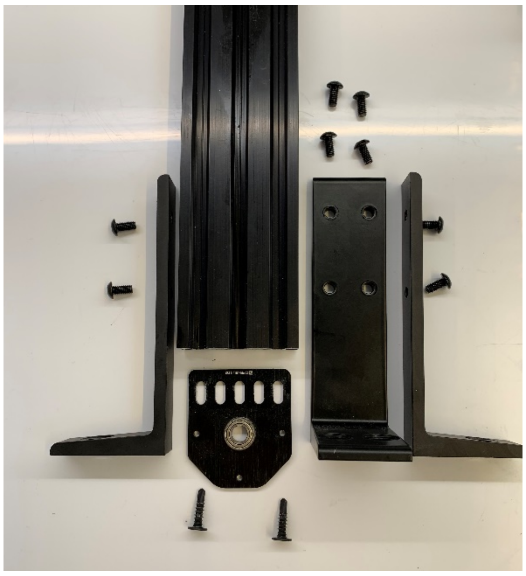
Step-by-step assembly guide


Start with the sledge and mount the spacer blocks on the gantry plate.

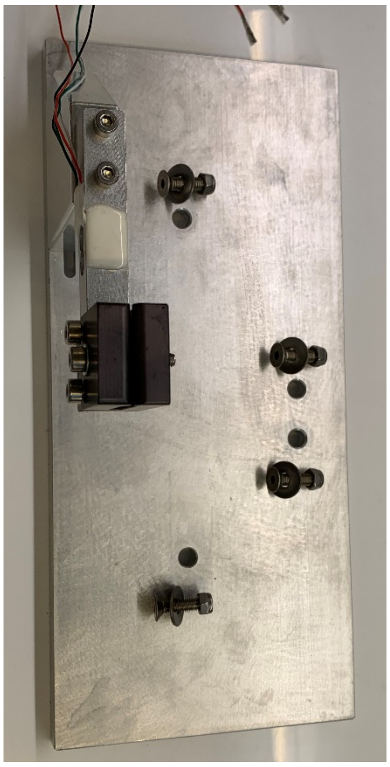

Mount the wheels. One side should have the eccentric spacers for a later adjustment of the sledge.


Assemble the motor mount block. Remember to add the anti-backlash nut block to the threaded rod.


The motor block can then be mounted one the linear rail.

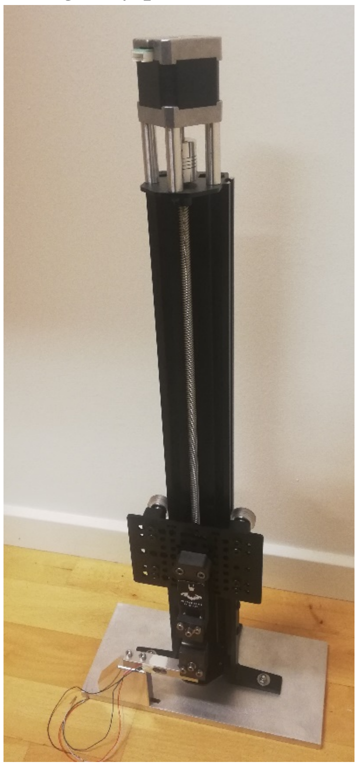


Mount the threaded rod plate and the mounting feets at the other end.

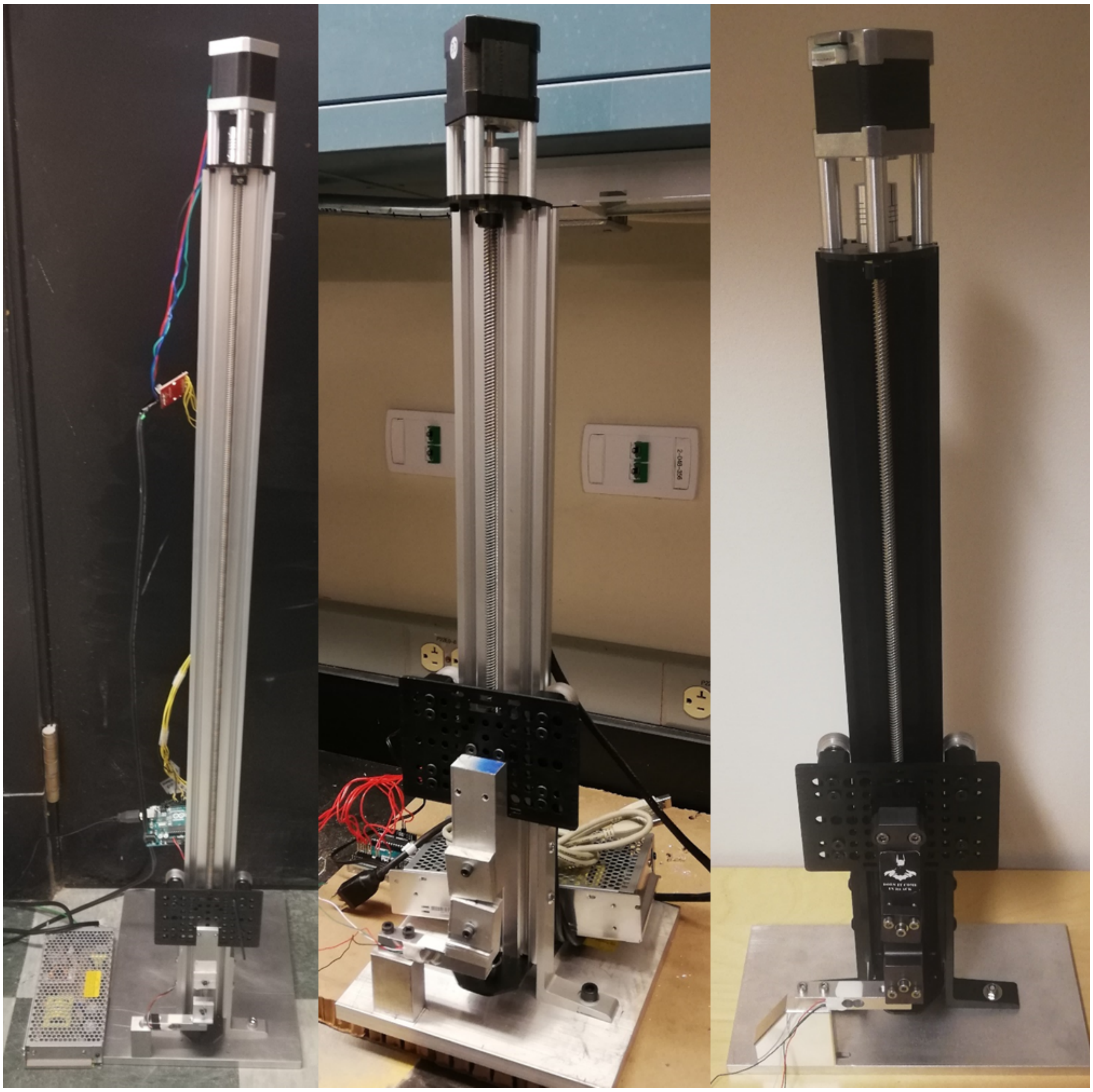


The load cell mount, load cell and bottom grip can be assmebled and mounted on the base plate. Afterwards, the linear acturator can be mounted on the base plate.

The top grip and clamp can be assembled and mounted on the gantry plate.



The final setup.
OBS: It is advantageous to assemble everything, and then drill the wholes in the base plate while everything is alligned. It is critical, that thegrips are alligned exactly on top of each other.


Several models has been made and tested with both an Arduino Uno and Nano. The setup can be fully custimized to fit the given needs with respect to clamps, three point bending, compression, adhesion testing etc. Further, the hight can be adjusted absed on the linear rail, the shown models are 100, 50, and 50 cm, respectivly.

Further, help can be found on OpenBuilds own webpage and YouTube channel, which supports how to build the Linear Actuator. "*V-Slot® NEMA 17 Linear Actuator Bundle (Lead Screw)*" https://www.youtube.com/watch?v=X0Z_gfl-iGY

Electronics

Several possibilities exist for breakout boards suitable for the final electronics. I all cases one needs to have a microcontroller e.g. Arduino, an industrial amplifier e.g. Hx711 or an INA125A for handling the load cell signals and finally a motor driver e.g. Pololu stepstick or a Big Easy Drive. Here two configurations are demonstrated but all sorts of combinations are possible.

Configuration one:		Configuration two:	
Microcontroller:	Ardunio Uno	Microcontroller:	Ardunio Nano
Load cell amplifier:	HX711	Load cell amplifier:	HX711
Stepper motor driver:	Big Easy Drive	Stepper motor driver:	Pololu drive
Connection:	Wiring	Connection:	Designed PCB

Connection between the breakout boards can be performed with wires or printed circuit boards (PCB). For PCB two approaches can be taken; a) Vero-boards (or the like) or b) specially designed PCB's. Further options exist for soldering directly onto the PCB or adding sockets. Here only four types are demonstrated i.e. wiring, designed single sided PCB with and without sockets. Schematics for configuration one and two are given in figure S6 and figure S7, respectively.

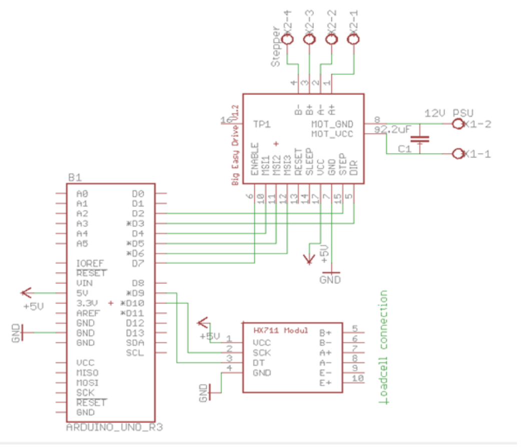


Figure S6 Schematics for electronic configuration one.

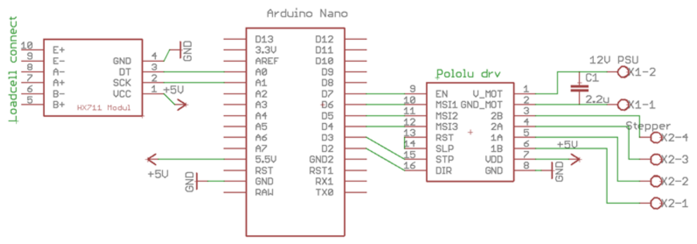


Figure S7 Schematic for electronic configuration two.

Configuration one was made by wiring, thus only a PCB for configuration two was designed and made. PCB for configuration two is given in figure S8.

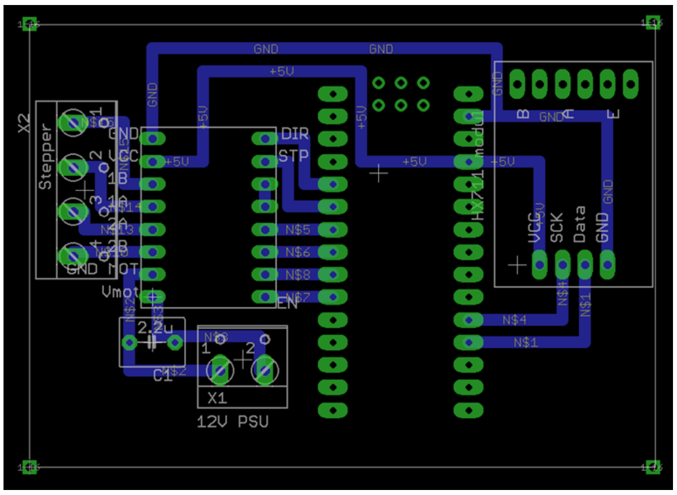


Figure S8 PCB for configuration two.

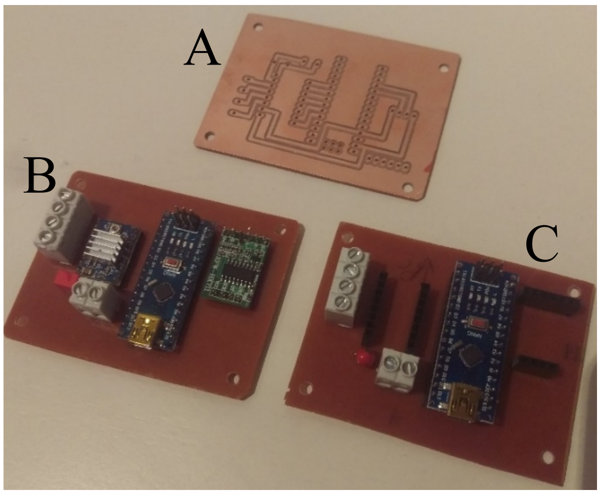


Figure S9 Images of the PCB. A) Cupper side before soldering, B) direct soldered, and C) with soldered sockets (Pololu drive and HX711 removed for clarity).

The PCB in figure S8 was made by CNC milling in a PCB cupper plate (figure S9 A) and can then be used with (figure S9 C) and without (figure S9 B) sockets. From figure S9 C it is seen that the sockets gives the possibility of exchanging the brake out boards. On the other hand, sockets increase the potential of “bad connections”, increase board complexity and height. The direct soldered PCB (figure S9 B) is thus preferred.

Calibration procedure

1. Open the Arduino IDE software.
2. Install the HX711_ADC and AccelStepper library v1.59 (Sketch→Include Library→Add .ZIP library).
3. Upload the “Calibrate” program from the HX711_ADC library (file → examples→HX711_ADC→calibrate) to the microcontroller.
4. Use spring scales (Ajax Scientific) to measure coarse (i.e. 5000 g) and fine changes (i.e. 250 g) by rotating the screw with the fingers.
5. Change the *Calibration Factor* so it matches the strain in grams to the one on the scale.
6. Change the *Calibration Factor* (Load cell.setCalFactor(xxx.x)) in “DCTT.ino” and upload it to the microcontroller.

Running an experiment

1. Open DCTT.exe, go to “Setup”.
2. Choose your ASCII data file for experimental data storage.
3. Choose COMport, speed in mm/min, threshold and sample rate. (If Arduino IDE is open, better shut it down, as it blocks the COMport)
4. Optional: “Cam Settings” can adjust all effects as contrast, brightness etc.
5. Choose “Define AOI” to take a picture and drag a box where a desirable Area of Interest (AOI) can be found. Take into consideration how far the test specimen will be in full extension at the end of the test.
6. A pop-up window will help the user to decide on the threshold, based on the contrast and with of the AOI.

7. “Controls” can be used to adjust speed, position and homing position of the grips together with taring the load cell.
8. “Start” starts the experiment, and it has to be stopped after it is done.
9. “Reset all” resets the user values and makes the program ready for a new experiment

Test specimens

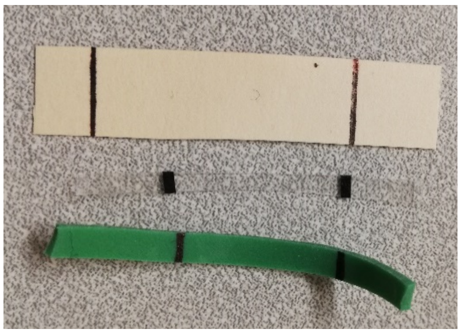


Figure S10 Test specimens, manila paper (top), poly(propylene) (middle), rubber (bottom). Black marks, drawn with a fine marker, are used for optical extensometer reading.

Optical extensometer

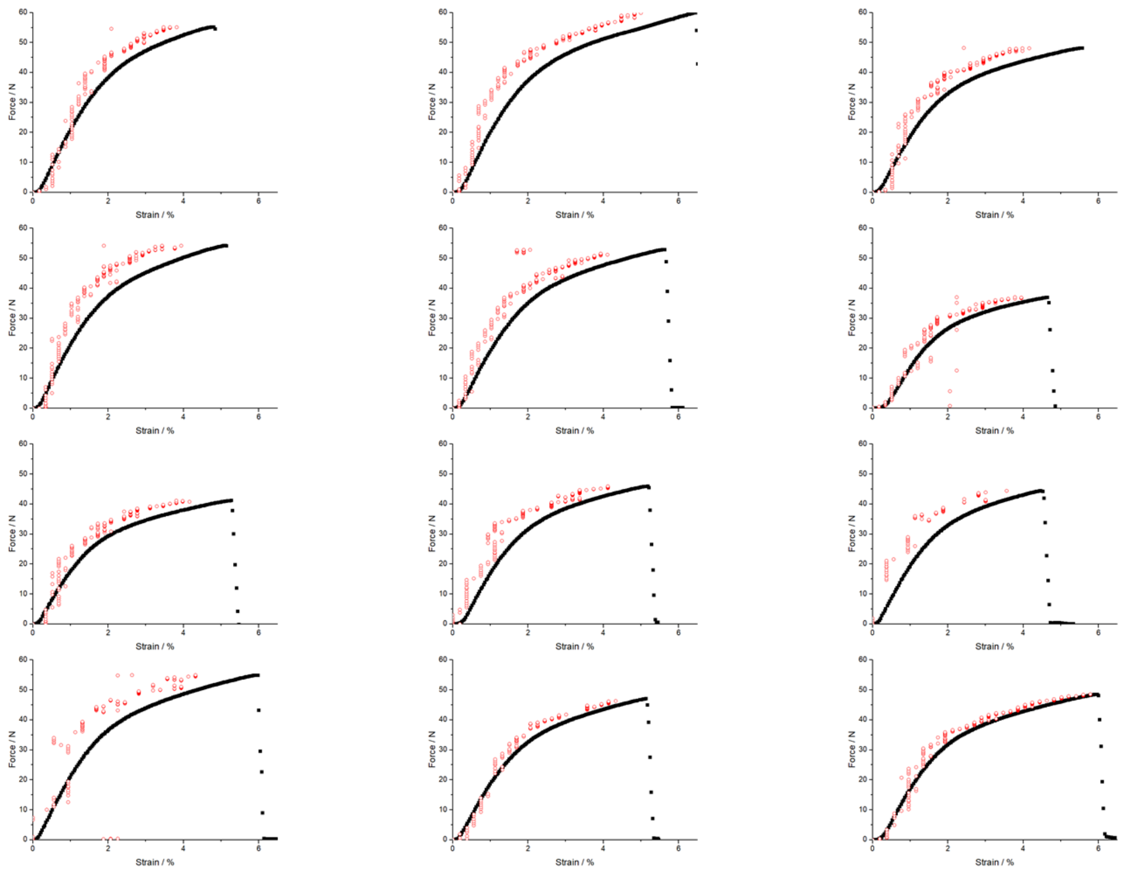


Figure S11 Extensometer measurements (dot) together with grip-to-grip (line) for Manila paper experiments in order to verify the two different strain measurement methods. In general, the video extensometer shows a more ductile material with a shorter extensibility, as was seen for the same type of material measured on the Instron tensile tester.

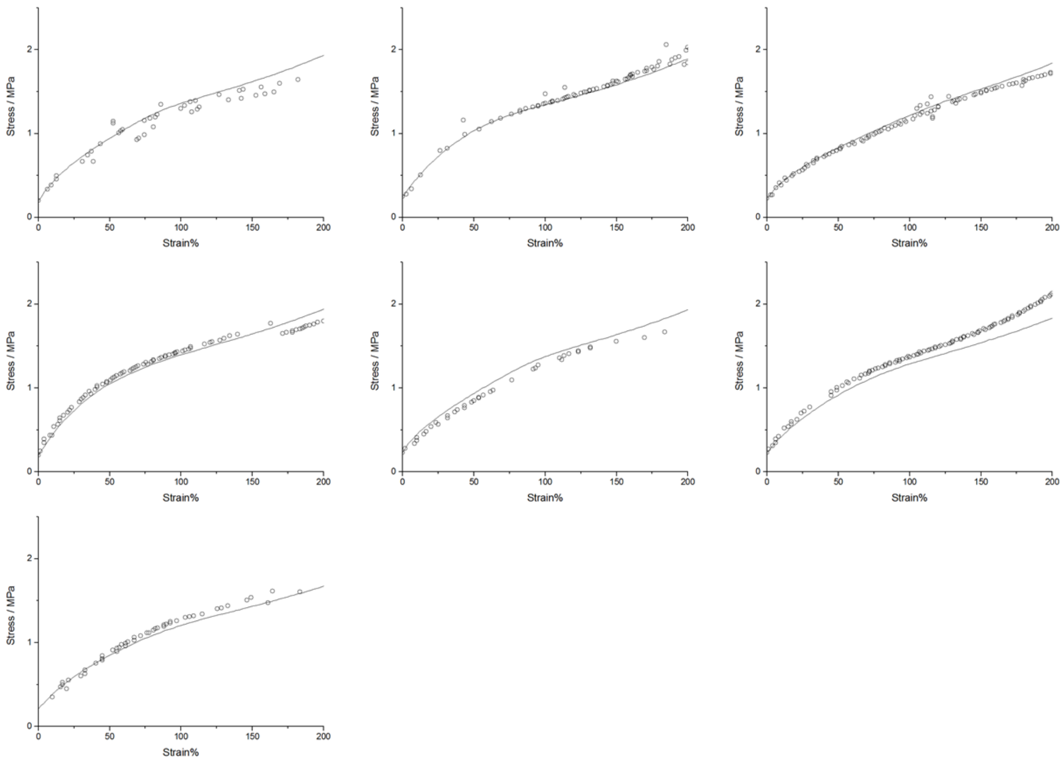


Figure S12 Extensometer measurements (dot) together with grip-to-grip (line) for rubber experiments in order to determine the elastic moduli. *E* 1.4(0.3) MPa for strain: 0%-50% and 0.5(0.1) MPa for strain: 100%-200%.

Manila paper cut in CD and MD

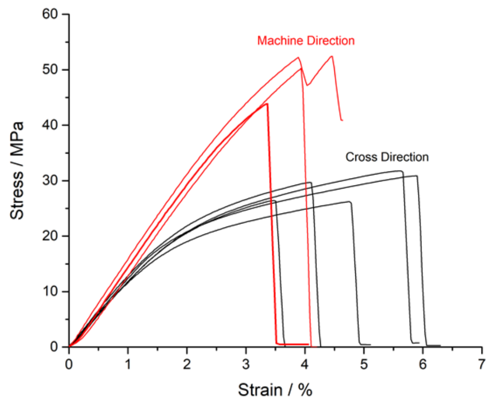


Figure S13 Manila paper cut in cross (CD) and machine direction (MD) and mechanically tested, hereby showing to different mechanical behaviors due to the manufacturing process. CD *E*: 1209(77) MPa; *UTS*: 27.6(3.3) MPa; ε_b: 4.2(0.8) %; *U*: 0.8(0.2) MJ m^{-3}. MD *E*: 1483(132) MPa; *UTS*: 48.8(4.5) MPa; ε_b: 3.9(0.3) %; *U*: 1.1(0.3) MJ m^{-3}. MD.

Arduino Code

The microcontroller code is identical for the two configurations, except for the pin configuration.

```
//---- Start of program----
/*******************************************************************************
This is the software for the Custom-Made Tensile Testing (DCTT) developed at MIT under the supervision of Prof. Jeremiah Johnson, MIT, and
Assco. Prof. Mogens Hinge, Aarhus University.
Written in Arduino IDE v1.8.5
Version 1.0
Tested with:
a) Arduino Uno, HX711 amplifier, 10 kg Load cell (TAL220), Big Easy Driver, NEMA 17 motor
b) Arduino NANO, HX711 amplifier, 10 kg Load cell (TAL220), Pololu Driver, NEMA 17 motor
Disclaimer:
This code is beerware i.e. if you see us at the local, and you have found this code helpful, please buy us a beer and let us have a chat! :)
*******************************************************************************/
//including packages
#include <AccelStepper.h>
#include <HX711_ADC.h>

//defining variables
//#define SAMPLES     16; // running average can be 4, 8, 16, 32, 64, 128
int SetMotorSpeedMax = 6667.5; // steps pr. second. I.e. 6400 would be one revolution (1/16 step is by default activated). And there is
8mm/revolution. Hence, 6400 corresponds to 8 mm/s pr second. initial velocity corresponds to 2 mm/min (6400 / 4 / 60 = 26.67), 500 mm/min =
6667,5
long t=0;
int SetMotorSpeed=0; // the cross head speed
long stabilisingtime = 2000; // tare precision can be improved by adding a few seconds of stabilizing time
int MotorRev = 400; // define your motors steps per revolution. It is typically 200 or 400 steps per revolution.
int MotorStep = 16; // 1=Full step, 4, 8 and 16 = micro steps. The code puts it default to 16 micro step via "resetBEDPins()". You lose some 20-30%
torque, but can do much slower and controlled velocities on the motor.
int RevDist = 8; // [mm] The distance moved in one revolution. For at setup with a 8 mm Metric Acme Lead Screw it moves 8 mm per revolution...
int StepPrDist = (MotorRev * MotorStep) / RevDist; //[step/mm] calculates the number of steps needed pr mm.
int debug=0; //set to 1 for debugging and 0 for not debugging

//if you use an UNO Uncomment the next 3 lines:
    AccelStepper stepperA(1, 2, 3);
    HX711_ADC LoadCell(9, 10);
    int Board = 1;

//if you use a NANO Uncomment the next 3 lines:
   //AccelStepper stepperA(1, 3, 2); // Nano  pin 3:Step and Pin 2:Dir
   //HX711_ADC LoadCell(14, 15); // Nano pin 14(analog A0):Data and pin 15(analog A1):Clock
   //int Board = 2;

void setup()
{
Serial.begin(115200); //fastest stable com speed we could run with.
if(Board==1){
  pinMode(2, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  resetBEDPins();
  Serial.println("Version UNO ...Wait for it..."); //indicating that all is good
  } else {
  pinMode(2, OUTPUT); // NANO Dir
  pinMode(3, OUTPUT); // NANO Step
  pinMode(4, OUTPUT); // NANO MSI3
  pinMode(5, OUTPUT); // NANO MSI2
  pinMode(6, OUTPUT); // NANO MSI1
  pinMode(7, OUTPUT); // NANO Enable
  pinMode(11, OUTPUT);// NANO not used.
  resetBEDPins();
  Serial.println("Version NANO ...Wait for it...");//indicating that all is good
  };

  LoadCell.begin(); //Setting up the HX711
  LoadCell.start(stabilisingtime); // Tare
  LoadCell.setCalFactor(212.0); // user set calibration factor (float)
  stepperA.setCurrentPosition(0); //Reset position
  stepperA.setMaxSpeed(SetMotorSpeedMax); //setting maximum speed
  stepperA.setAcceleration(2000); //setting acceleration
  stepperA.moveTo(0); //stay put and don’t move
  Serial.println("Startup and tare is complete"); // indicating that the system is ready
} //setup

void loop()
{
    LoadCell.update(); //Updating the load cell value.
    stepperA.run(); //Running the motor at the speed setting defined
    communication(); //See subroutine description below
}

//Reset motor Driver pins to default states
void resetBEDPins()
{
  digitalWrite(2, LOW);
  digitalWrite(3, LOW);
  digitalWrite(4, HIGH);
  digitalWrite(5, HIGH);
  digitalWrite(6, HIGH);
  if(Board==1){
     digitalWrite(7, HIGH);
    }else{
     digitalWrite(7, LOW);
    }; //pololu drive enable when set to a logic high, the outputs are disabled when LOW. Big Easy Drive opposite
} // done resetBEDPins

void communication(){ //handling communication received from the serial port
if(Serial.available()>0){ // if values in the buffer then...
   String user_input=""; //empty the message
   String vaerdi=""; //Need a variable for the number (in text form).
   float UserValue=0; //Need a variable for the number (in number form).
   delay(10); // wait to secure the full message to be received
   while(Serial.available()>0){ //while still char in the buffer
    user_input += char(Serial.read()); //Read user input and trigger appropriate function
    delay(1); // for stabilization
   };
    if(Board==1){digitalWrite(7, HIGH);}else{digitalWrite(7, LOW);}; //Pull enable pin low to set FETs active and allow motor control

 //An evaluation of what is sent and what to do now. The following commands are included
 //'F': send data now,
 //'S': set speed,
 //'T': terminate everything
 //'O': tare scale
 //'G': Start experiment
 //'H': Home
 //'P': Go to exact position
 //'U' : Move to relative
 //'Z' : set pos to zero
 //  --- end of commands----

  //'F': send data now,
     if(user_input[0]=='F'){ //send data now
        LoadCellData(); //obtaining the load cell data from the HX711
     }; //if F

  //'S': set speed,
     if(user_input[0]=='S'){ //A new speed have arrived
        for(int i = 1 ; i < user_input.length() ; ++i ){vaerdi+=user_input[i];} //Removing identification tag
        char carray[vaerdi.length() + 1]; //creates an array to keep the chars
        vaerdi.toCharArray(carray, sizeof(carray)); //Puts the user input in an array
        UserValue = atoi(carray); //converts the text to integer
        UserValue = UserValue * StepPrDist / 60; //turns mm/min into steps/min
        if (UserValue <= 0){UserValue=0;} // minimum speed is 0, negative velocities will be set to 0 mm/sec
      // if (UserValue >= SetMotorSpeedMax){UserValue=6667;} // Max speed limit is 500 mm/sec
        SetMotorSpeed=UserValue; //Motor speed is updated
        if(debug==1){Serial.println(SetMotorSpeed);}; //printout the set speed
     }; //if S

  //'T': terminate everything
     if(user_input[0]=='T'){ //Terminate
        stepperA.stop();
     }; //if T
  //'O': tare scale
     if (user_input[0] == 'O'){ //Tare of load cell
        LoadCell.tareNoDelay();
        if(debug==1){Serial.println("Tare complete");}; //Printout that tear is done
     }; // if O

  //'G': Start experiment
     if(user_input[0]=='G'){ //starts the movement and thereby the experiment
       stepperA.setCurrentPosition(0); //reset position
       stepperA.setMaxSpeed(SetMotorSpeed); //start the movement
       stepperA.moveTo(-64000); // go 80 cm up
     }; // if G

  //'H': Home
     if(user_input[0] =='H'){ // go home
        stepperA.setMaxSpeed(SetMotorSpeedMax);  //go as fast as possible
        stepperA.moveTo(0);// move back to start
     }; // if H

  //'P': Go to exact position
     if (user_input[0] =='P'){ //setting an absolute position and moves to it
      for(int j = 1 ; j < user_input.length() ; ++j ){vaerdi+=user_input[j];} //removes identification tag
        char carray[vaerdi.length() + 1]; //creates an array to keep the chars
        vaerdi.toCharArray(carray, sizeof(carray)); //Puts the user input in an array
        UserValue = atoi(carray); //converts the text to integer
        stepperA.moveTo(UserValue); // move to the new relative position
        if(debug==1){Serial.println(UserValue);}; // printing out the distance to be moved
     }; // if P

  //'U' : Move to relative
      if (user_input[0] =='U'){ //setting a relative position and moves to it
      for(int j = 1 ; j < user_input.length() ; ++j ){vaerdi+=user_input[j];} //removes identification tag
        char carray[vaerdi.length() + 1]; //creates an array to keep the chars
        vaerdi.toCharArray(carray, sizeof(carray)); //Puts the user input in an array
        UserValue = atoi(carray); //converts the text to integer
        stepperA.setCurrentPosition(0); //reset position
        stepperA.moveTo(UserValue); // move to the new relative position
        if(debug==1){Serial.println(UserValue);}; // printing out the distance to be moved
     }; // if U

  //'Z' : set position to zero
     if (user_input[0] == 'Z'){ //Tare of load cell
        stepperA.setCurrentPosition(0); //reset position
        if(debug==1){Serial.println("Stepper pos set to zero");}; //Printout that tare is done
     }; // if O
   }//end off if serial available
}//end of void communication.

void LoadCellData(){ //acquiring the load cell data from the HX711
   //get smoothed value from data set + current calibration factor
  if (millis() > t + 75) {
    float i = LoadCell.getData();
    float v = LoadCell.getCalFactor();
    float j = stepperA.currentPosition();
    if(debug==1){
     //sends back all data for debugging
     Serial.print(millis());
     Serial.print(";");
     Serial.print(j);
     Serial.print(";");
     Serial.print(v);
     Serial.print(";");
     Serial.println(i);
    } else {
     t = millis(); //updating the time
     //sends data back to the main program as "Pos;load"
     Serial.print(j); //printout the position
     Serial.print(";"); //data separator
     Serial.println(i); // printout the load value
    } //sending signal
  }//is time
}; // end of load cell data
//---- end of program----
```

Post-processing by ImageJ

The following procedure is used to post-process the pictures taken during an experiment. To use it, open ImageJ (v. 1.51r)

```
//----------macro start -------------
startbillede=5672; //First picture
slutbillede=8717;  //Last picture
base="C:\\Users\\....\\PPcup\\PP4\\PP1PP2PP3PP4_"; //Folder\\Subfolder…\\Subfolder\\Image_name
billedetype=".bmp"; \\Picture file format
tress = 180;    // Threshold Grey-value for analysis

//Remember to change rectangle. (0,0) is top left corner in pictures.
startx = 160;
starty = 275;
deltax = 400;
deltay = 10;

setBatchMode(true); //Run in background and do not show images
k=0; //Indicate which column the results should be saved in

for(j=startbillede; j<=(slutbillede); j++){ //Running thought all images
showProgress(j, slutbillede);
sti=base+j+billedetype;
open(sti);

run("Rotate... ", "angle=90 grid=1 interpolation=Bilinear"); //Rotates image if cam was not aligned.
makeRectangle(startx, starty, deltax, deltay);  //Add AOI to image
run("Crop"); //Crops the picture to AOI
run("Subtract Background...", "rolling=50 light"); //Adjusting background light;

//Creates 8-bit picture to reduce the amount of computation
makeRectangle(0, 0, deltax, deltay); // Rectangle of whole picture - to analyze the grey value
indbil=getImageID(); //Saves name of the picture
profile = getProfile(); //obtains the average intensity for the AOI

hen=0; // Occurrence - assuming two peaks
etStart=0; //Entering the first peak along the threshold line
etSlut = 0; // Exit the first peak
etLinje = 999; //Pre-set value for the first line (ridiculous high for spotting errors)
toStart =0; // Entering the second peak along the threshold line
toSlut = 0; // Exit the second peak
toLinje = 999; //Pre-set value for the second line (ridiculous high for spotting errors)

//Analyzing the profile and finding the two lines
for(i=0; i<profile.length-1; i++){

                if((profile[i]>tress) && (hen==3)){ // Exit of peak two (right side of peak two)
                                hen=4;
                                toSlut=i;
                                };//if

                if((profile[i]<tress) && (hen==2)){ //Entering peak two, Left side of peak two
                                hen=3;
                                toStart=i;
                                };//if

                if((profile[i]>tress) && (hen==1)){ //Exit of peak one, Right side of peak one
                                hen=2;
                                etSlut=i;
                                };//if

                if((profile[i]<tress) && (hen==0)){ //Entering peak one, Left side of peak one
                                hen=1;
                                etStart=i;
                                };//if
                if(hen==4) {
                                i=6000; //set I to a large number for stopping the analysis.
                                }; // All peaks has been found, and calculation are stopped.
 };//for
// Finding the average of the peak/line
etLinje = (etStart+etSlut)/2;
toLinje = (toStart+toSlut)/2;

//saving the results
setResult("Filnavn",k,sti);
setResult("Startpix",k,etLinje);
setResult("Slutpix",k,toLinje);
k=k+1; //move to next result line

selectImage(indbil); //Selects the picture again
close(); // closing the picture
}; //For loop stops

setBatchMode(false); //show images again

//Saving results
gemnavn = base+"_result.txt"; //concatenating the result filename
updateResults(); //updating the result table
saveAs("text",gemnavn); //saving the data

//---------end of macro-------
```