How to build a cheap 3D-scanner mostly out of spare parts

by Till Handel, April 20th 2014

This article is published under GPLv3

Abstract

This paper describes how to build a 3D scanner out of parts for less than 60 Euro and parts that were extracted from old printers, notebooks and so forth.

The scanner will be good for scanning a 360° field around its own position at distances of approx. 0.3 to 5 m. So basically this scanner is optimized to scan rooms. It creates a point cloud that resembles the visible surface of the surroundings as viewed from the position of the scanner.

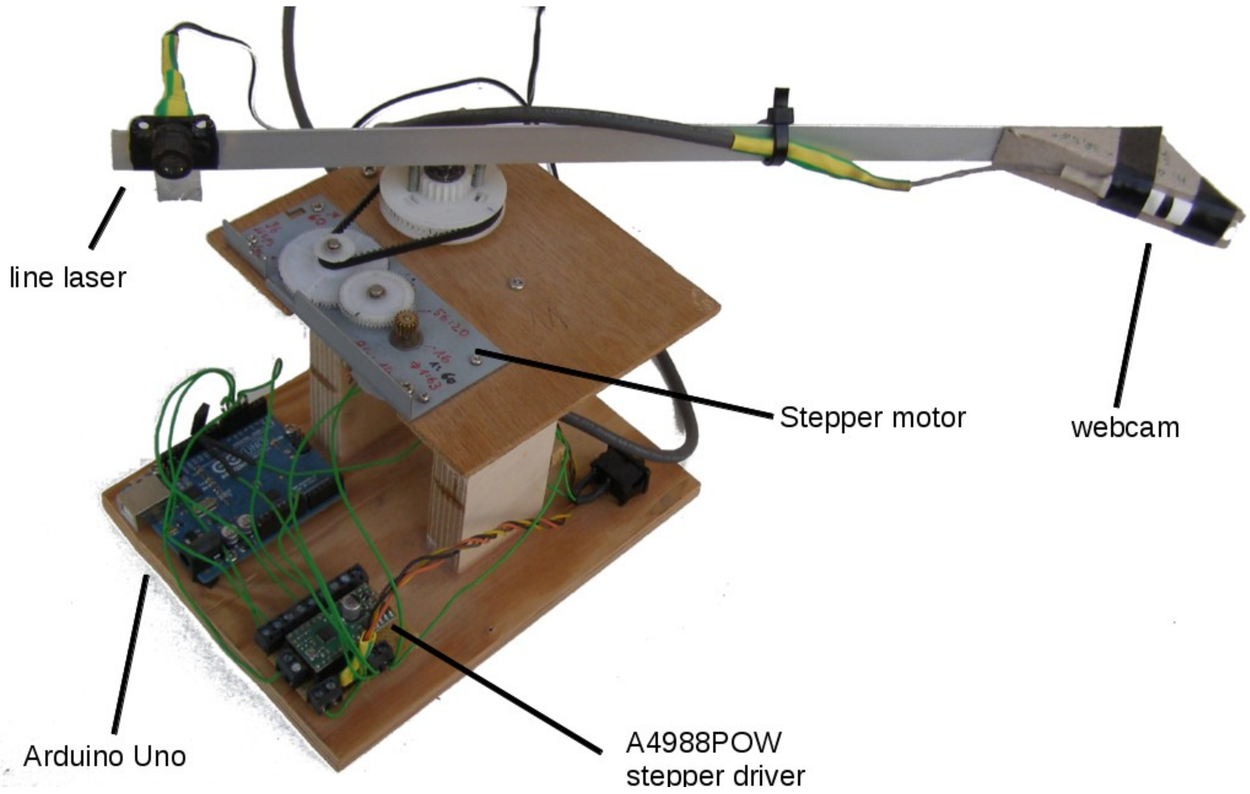

Figure 1: 3D Laserscanner; image created by the author

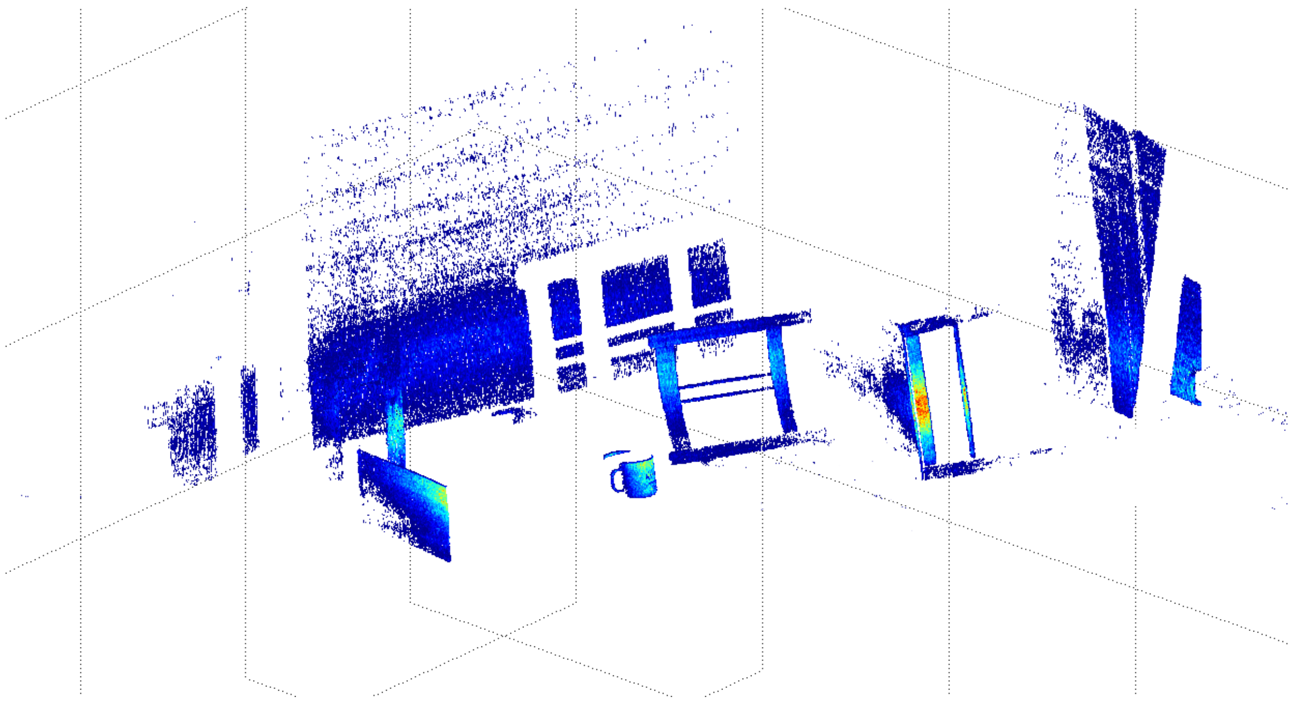
Figure 2: point cloud of a chair and a mug scanned with the 3D-scanner, the chair casting a 'shadow' on the wall behind it; image created by the author

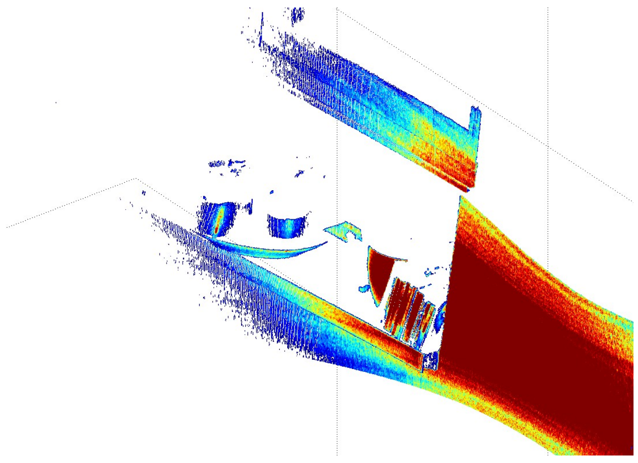
Figure 3: a window board scanned from slightly below with the 3D-scanner; image created by the author

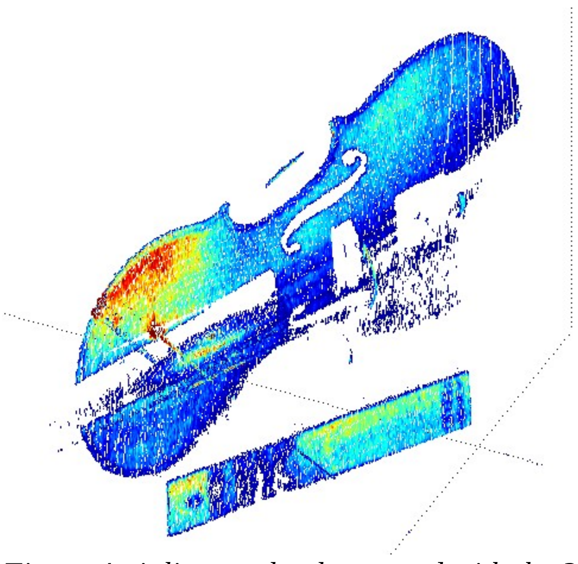
Figure 4:violin on a book scanned with the 3D-scanner; image created by the author

Table of Contents

Required tools

- **soldering iron + solder** - soft solder for electronics
- **simple multimeter** to check voltage, resistance and connectivity
- **PC with 2 unoccupied USB-ports** - one to communicate with Arduino the other to receive the webcam signal
- The software provided here was run on a Fedora 19 OS. Tor program the microcontroller the **Arduino programming software & compiler** 1.0.5 was used. The **GCC**-compiler was used for the c++ code utilizing the **OpenCV** package. The analysis software was written to run on **Matlab** 2013a (older versions will probably do just as well).

Required parts

- **Arduino Uno** or another microcontroller that can communicate with a PC and has at least 3 digital outputs - ~28 Euro
- **stepper motor driver -** A4988POW was used here which has Allegro’s A4988 DMOSMicrostepping Driver onboard - ~24Euro

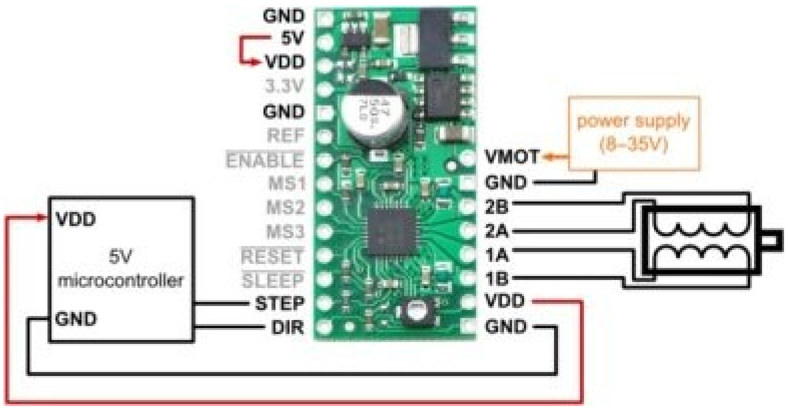

Figure 5: A4988POW stepper driver; image taken from:http://www.shop.robotikhardware.de/shop/catalog/product_info.php?products_id=237

another stepper driver that I can recommend is the one based on the L297 and L298N IC, it is slightly cheaper but much more work and more bulky - ~19 Euro

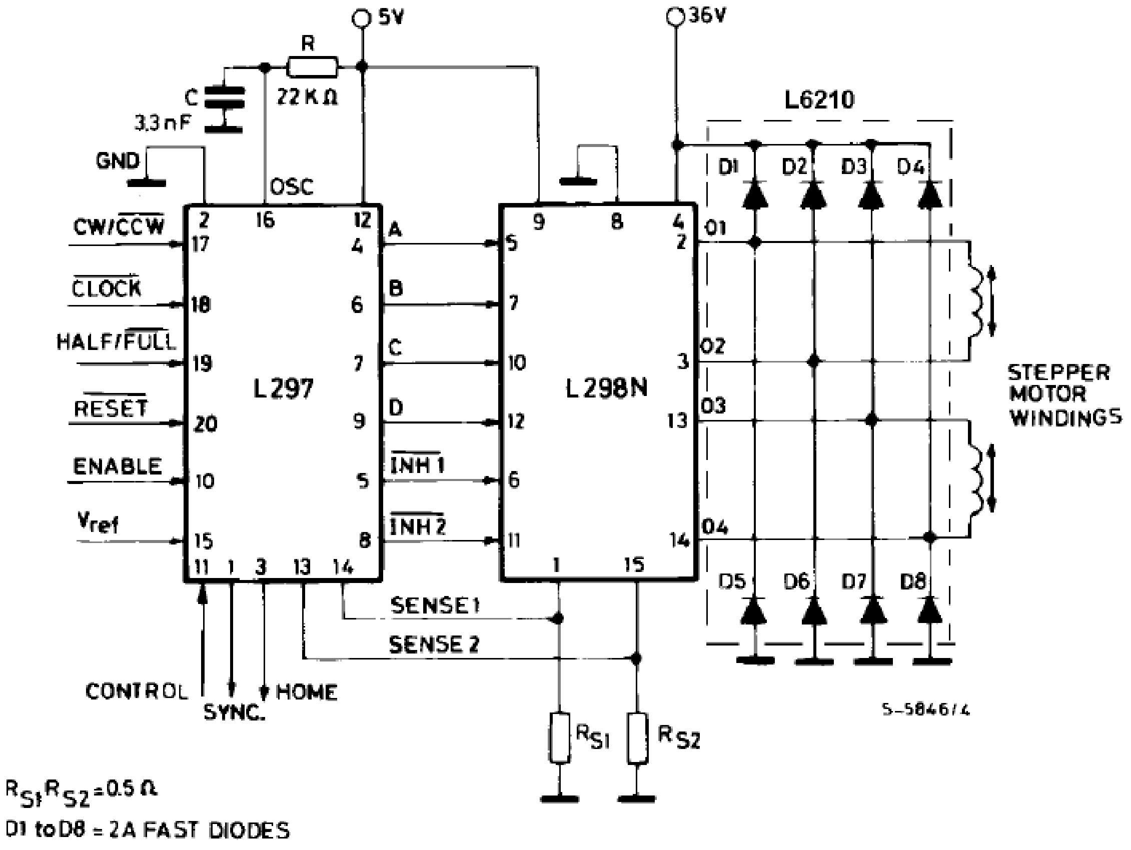

Figure 6: L297 & L298N stepper driver; image taken from: ST L298 datasheet

- **stepper motor -** I used a M35SP-11NK with a 60:1 translation that I got from a broken laser-printer I found lying on the street. That gives me an angular resolution of 5760 steps / 360° which is more than enough. If you have an ordinary stepper with 200 steps/360° and are using the A4988POW which supports 16 microsteps per wholestep you get a resolution of 3200 steps/360° which is pretty good. Thus you don't really need the translation gear.
- **webcam** - I used the webcam that was integrated into the display of my old Lenovo notebook. Those display webcams are tiny, come cheap on Ebay, have increasingly awesome resolutions and are nothing but ordinary USB-webcams that are connected to an intertnal USB-port; They require 3.3V instead of the 5V your USB provides so you have to put a voltage regulator in between:

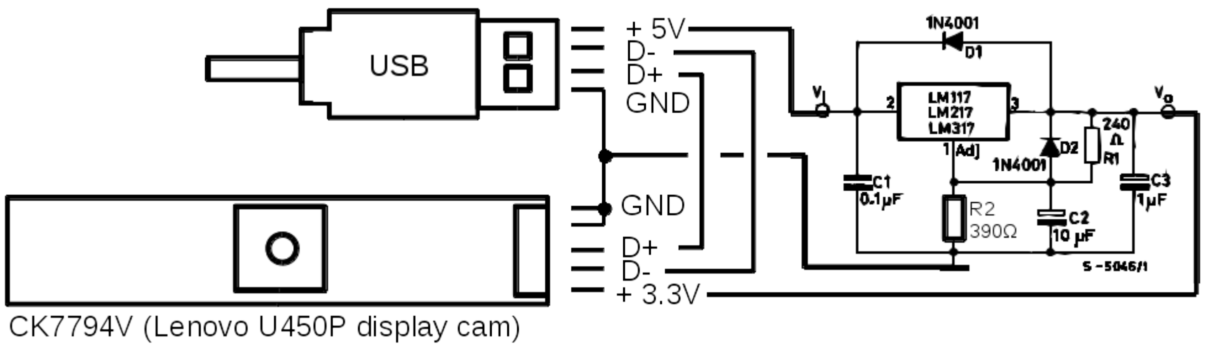

Figure 7: display cam circuit diagram; image taken from: ST LM317 datasheet and modified by author

- **line laser** – preferably a strong green line laser which can be build by extracting the laser module and driver from a cheap chinese green laser pointer and putting a prism from a red spirit level laser in front of it.
- **8-35V power supply**

Construction

The whole thing should be mounted on some kind of solid frame. A simple wooden set-up as shown in figure 1 will suffice. Webcam and laser are mounted on a rotatable arm, the laser pointing forward at 90° to the arm, the webcam being tilted inward to point at a small angle (in my case 31°) inward. The laser projects its line vertically.

The stepper motor drives the arm.

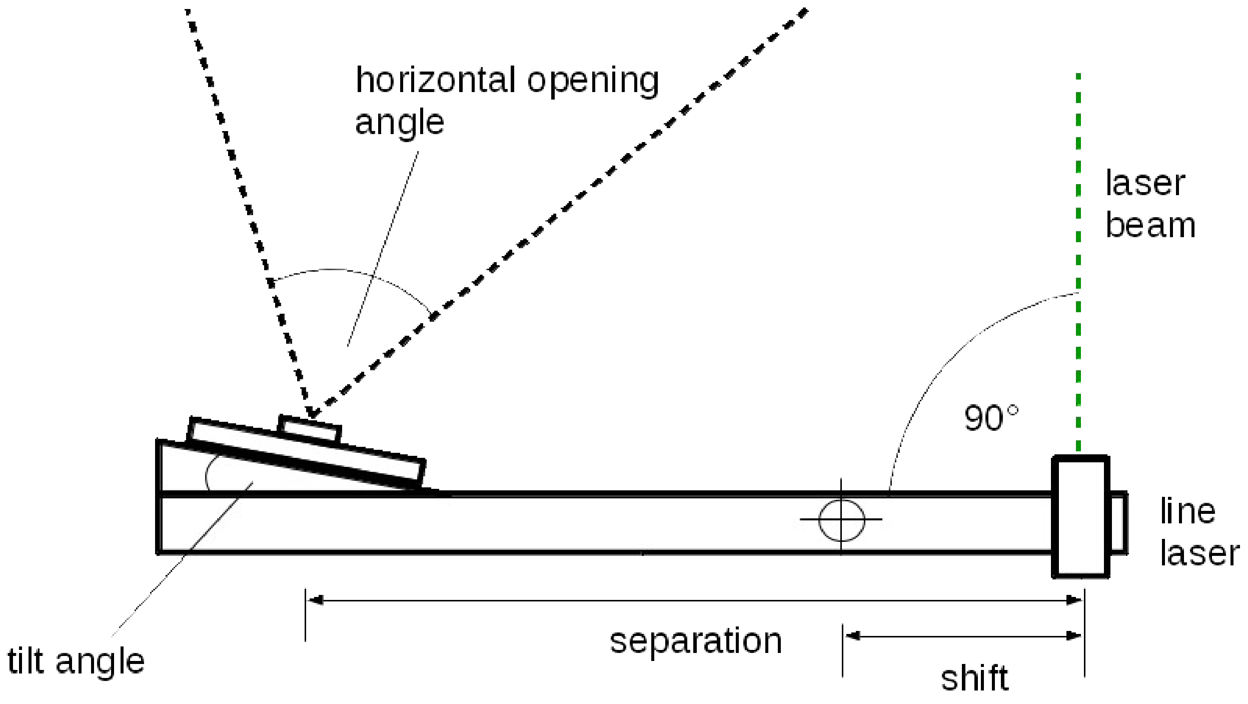

Figure 8: arm top view; image created by author

The angles and distances marked here are important input parameters that are required by the Matlab script in order to calculate the true 3D-cartesian coordinates of each point out of the data caputred by the webcam.

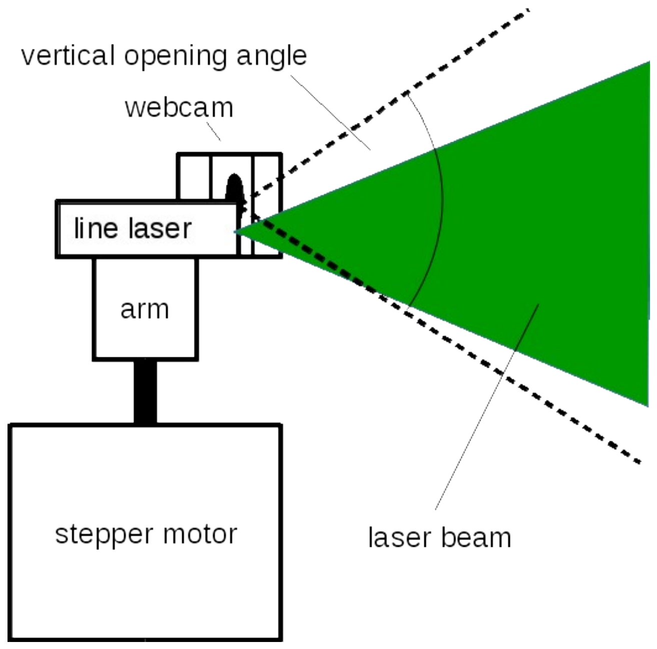

Figure 9: arm side view; image crated by author

The circuit diagram is as follows:

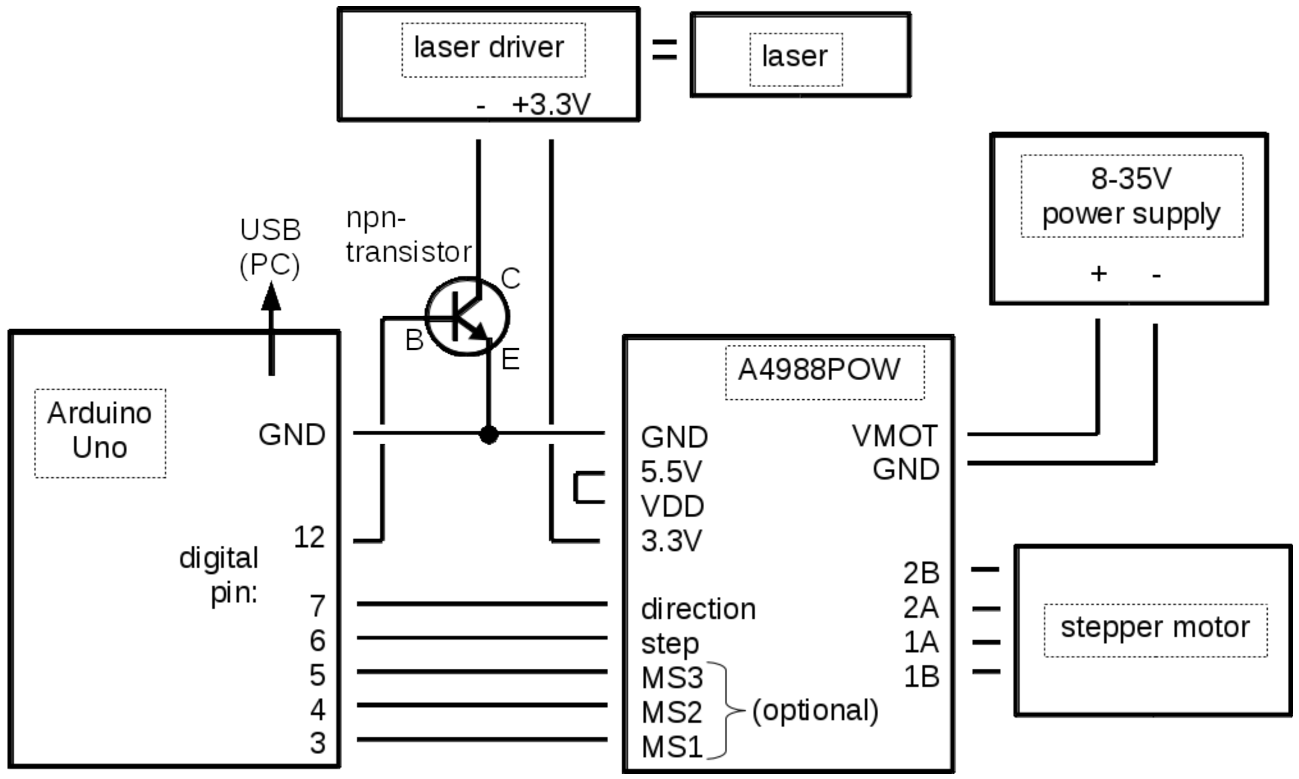

Figure 10: circuit diagram of the 3D scanner; image created by the author

Be careful with the laser driver! Sometimes they are simple voltage regulators. However this is not sufficient since the laser diode will draw too much current, burning out eventually. A current regulator based on the LM317 will be the simplest laser driver that actually works. (However you will need to know at how much current your laser diode operates safely.)

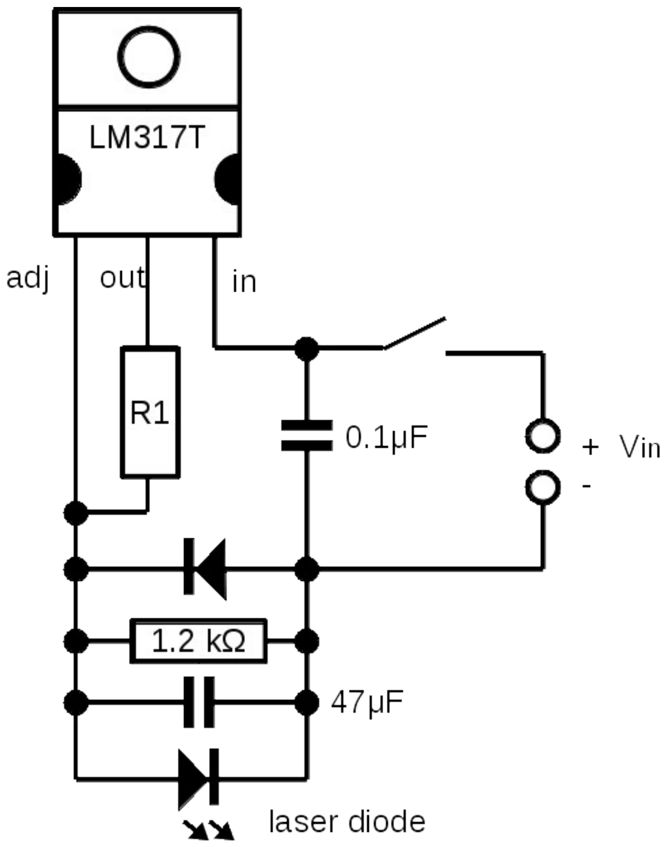

Figure 11: Laser driver based on LM317 (current regulator); image created by the author

Where the operational current (Io) of the laser diode is calculated by Io = (Vref / R1) = 1.25 V / R1. The voltage will adjust itself.

Software listings

Arduino

The program that runs on the Arduino Uno is a simple parser that receives one-letter ASCII-command-strings and acts accordingly on the stepper and laser:

```
// parser from serial to d-out
// by Till Handel 01/2014, (C)GPLv3

//define variables
int Dout3 = 0;
int Dout4 = 0;
int Dout5 = 0;
int Dout6 = 0;
int Dout7 = 0;
int Dout12 = 0;
int Dout13 = 0;

int laser = 0;  //laser on=1; off=0
int inbyte = 0;

void setup() {
 // put your setup code here, to run once:

 //configure digital outputs
 pinMode(3, OUTPUT); //MS1
 pinMode(4, OUTPUT); //MS2
 pinMode(5, OUTPUT); //MS3
 pinMode(6, OUTPUT); //step
 pinMode(7, OUTPUT); //direction LOW->counter-clockwise, HIGH->clockwise
 pinMode(12, OUTPUT); //laser
 pinMode(13, OUTPUT); //on-board LED

 //start serial connection
 Serial.begin(57600);
 while (!Serial) {
  ; // wait for serial port to connect.
 }

 establishContact(); // send a byte to establish contact until receiver responds
}

void loop() {

 if (Serial.available() > 0){
```

```
inbyte = Serial.read();


//f = forward step
if (inbyte == 'f') {
 digitalWrite(7, HIGH);
 digitalWrite(6, HIGH);
 digitalWrite(13, HIGH);
 delay(50);
 digitalWrite(6, LOW);
 digitalWrite(13, LOW);
 delay(50);
}
//b = backward step
if (inbyte == 'b') {
 digitalWrite(7, LOW);
 digitalWrite(6, HIGH);
 digitalWrite(13, HIGH);
 delay(50);
 digitalWrite(6, LOW);
 digitalWrite(13, LOW);
 delay(50);
}

//l = switch laser on
if (inbyte == 'l') {
 if (laser == 0) {
  digitalWrite(12, HIGH);
  laser = 1;
  delay(100);
 }
}

//d = switch laser off
if (inbyte == 'd') {
 if (laser == 1) {
  digitalWrite(12, LOW);
  laser = 0;
  delay(100);
 }
}

//p = pause mode
if (inbyte == 'p') {
 digitalWrite(12, LOW);
 digitalWrite(6, LOW);
```

```
   digitalWrite(13, LOW);
   delay(200);
  }
 }
}


void establishContact() {
 while (Serial.available() <= 0) {
  Serial.println("ready"); // send an initial string
  delay(300);
 }
}
```

C++ -script that executes the scan

The following script executes the scan. It is written in C++ and compiled with GCC - in this case (GCC) 4.8.2 20131212 (Red Hat 4.8.2-7). Required packages are OpenCV as well as a few standard packages listed in the '#include'-section.

This program opens a video stream to a specified USB-webcam using some predefined OpenCV routines. It then instructs Arduino to switch the laser on and to make 'forward steps' by sending one-letter ASCII-strings. Notice that it does this by opening a stream to a serial-port although Arduino Uno is connected to the PC by USB. This is because Arduino has an internal RS232 to USB-converter and thus shows on the PC as serial device.

The program captures a frame for each step. At the same time it extracts the position and intensity of the maximum of each row of the image-frame, assuming that the maximum is at the laser-line. (Thus it works best if you switch off the light in the room while scanning.) The positions and intensities are subsequently written to .txt-files that can be read and converted to 3D-coordinates later.

After scanning and writing the files the program switches off the laser, winds back the arm and closes all communication channels to Arduino.

```
//serial port communication with arduino uno
//sending ASCII-string
//by Till Handel
/Jan 2014, (C)GPLv3


#include <iostream>
#include <fstream>
#include <cstdlib>
#include <string.h>
#include <highgui.h>
#include <math.h>
#include <stdio.h>
#include <cv.h> //OpenCV
#include <fstream>

using namespace std;
using namespace cv;


//*************************************************************
//set parameters
const int nsteps = 5760;          //# of steps per 360°
const double waitf = 0.1;         //time between forward steps in s
const double waitb = 0.1;         //time between backward steps in s
const double waittosend = 1;      //time to wait after sending command
const int imcol = 2;              //colour of laser (0=blue, 1=green, 2=red)
const int thres = 100;          //laser detection threshold [0,255}
const int cam = 1;
const int irows = 1024;         //webcam resolution
const int icols = 1280;
const int intersteps = 1;       //# of steps in between images
const int gkernel = 5;          //size of gaussian blur kernel
//************************************************************

//secondary functions
void wait (float seconds)
{
clock_t endwait;
endwait = clock () + seconds * CLOCKS_PER_SEC ;
while (clock() < endwait) {}
}




int main()
{

        //*********************************************************
        //establish serial connection

        // activate the tty connection with Arduino
        system("stty -F /dev/ttyACM0 cs8 57600 ignbrk -brkint -icrnl -imaxbel -opost -onlcr -isig
-icanon -iexten -echo -echoe -echok -echoctl -echoke -noflsh -ixon -crtscts");
        // open the tty connection as an ifstream - stuff received from Arduino
        ifstream arduino_from("/dev/ttyACM0");
        // open the tty connection as an ofstream - stuff sent to Arduino
        ofstream arduino_to("/dev/ttyACM0");




        //define variables
        double Time;
        char buf [20];
        char ard_from [20];
        char ard_to [20];

        Point point_tmp;
        Mat im1;
        vector<Mat> rgb;
        double val_tmp;
        double pos [irows];
```

```
//Wait five seconds for the Arduino to start up
cout << "wait 5 sec .." << endl;
wait(5);

//connect to arduino
arduino_to << "0,0,0" << endl;
cout << "connecting .." << endl;
wait(2);




//check whether webcam 1 is connected
VideoCapture cap1(cam); // open camera 1
cap1.set(CV_CAP_PROP_FRAME_WIDTH, icols);
cap1.set(CV_CAP_PROP_FRAME_HEIGHT, irows);
if(!cap1.isOpened()){
        cout<<"webcam is not connected"<<endl;
}else{
        cout<<"webcam is successfully connected"<<endl;

        wait(2);

        namedWindow("image",CV_WINDOW_NORMAL|CV_GUI_EXPANDED);
        namedWindow("green",CV_WINDOW_NORMAL|CV_GUI_EXPANDED);


        for(int i = 0;i<20;i++){
                cap1 >> im1; //get a new frame from camera 1

                imshow("image", im1);
                imshow("green", im1);
                waitKey(50);
        }



        //create vector storing r,g and b image
        rgb.push_back( Mat(irows, icols, CV_8UC1));
        rgb.push_back( Mat(irows, icols, CV_8UC1));
        rgb.push_back( Mat(irows, icols, CV_8UC1));



        while(!arduino_from.eof()){          //while the eof flag isn't set
```

```
//switch laser on
sprintf(ard_to, "l");
arduino_to << ard_to << endl;
arduino_from.clear();
arduino_to.clear();
cout << "laser on" << endl;

wait(waittosend);


//write positions &intensities to file
ofstream posfile;
posfile.open ("3Dscan_positions.txt", ios::out | ios:: trunc);

ofstream intfile;
intfile.open ("3Dscan_intensities.txt", ios::out | ios:: trunc);


//forward scannning movement
for(int i=0; i<nsteps; i++){
        for(int l = 0; l< intersteps; l++){
                sprintf(ard_to, "f");
                arduino_to << ard_to << endl;
                cout << "forward step " << i << endl;

                wait(waitf);

                arduino_from.clear();
                arduino_to.clear();
        }

        //grab frame
        cap1 >> im1; //get a new frame from camera 1
        GaussianBlur(im1, im1, Size(gkernel,gkernel), 1.5, 1.5);
        split(im1, rgb);
        threshold(rgb[imcol],rgb[imcol],thres,255,THRESH_TOZERO);

        //extract laser position in image
        for(int j=0;j<im1.rows;j++){
                minMaxLoc(rgb[imcol].row(j),NULL,&val_tmp,NULL,&point_tmp);
                posfile << point_tmp.x << "    ";
                intfile << val_tmp << "    ";
                pos[j] = point_tmp.x;
        }
        //show in im1 where laser was detected
        for(int j=0;j<im1.rows;j++){
                im1.row(j).col(pos[j]) = Scalar(0,255,0);
        }

        posfile << endl;
        intfile << endl;
        imshow("image", im1);
        imshow("green", rgb[imcol]);
        waitKey(30);
}


//switch laser off
sprintf(ard_to, "d");
arduino_to << ard_to << endl;
arduino_from.clear();
arduino_to.clear();
cout << "laser off" << endl;

wait(waittosend);




posfile.close();
intfile.close();



//backward movement
for(int i=0; i<nsteps*intersteps; i++){
        sprintf(ard_to, "b");
        arduino_to << ard_to << endl;
        cout << "backward step " << intersteps*nsteps - i << endl;

        wait(waitb);

        arduino_from.clear();
        arduino_to.clear();
}




//switch to pause mode
sprintf(ard_to, "p");
arduino_to << ard_to << endl;
arduino_from.clear();
arduino_to.clear();
```

```
                cout << "pause mode" << endl;

                wait(waittosend);




                break;

            }


            arduino_from.clear();
            arduino_to.clear();

            arduino_from.close();  //Close the ifstream
            arduino_to.close();    //Close the ofstream
            exit(0);
        }

    }
```

Makefile for the C++ -script

I used this simple makefile to compile the above C++ script in GCC.

```
all:
	g++ -o main main.cpp `pkg-config opencv --cflags --libs`
clean:
	rm -rf *.o main
```

Matlab script

Matlab script that reads the scan-data, calculates 3D-coordinates from it and displays the resulting point-cloud.

```
%program to convert data recorded with the 3D-scanner into cartesian
%coordinates and to plot them
%jan 2014, by Till Handel, (C)GPLv3

clear all;
clc;


%set parameters
posfilename = '3Dscan_positions.txt';   %file containing positions
intfilename = '3Dscan_intensities.txt';   %file containing intensities (not
really necessary for the 3D reconstruction)

a_rot = 360;    %rotational angle [deg]
a_camh = 49.5;    %horizontal opening angle of webcam [deg]
a_camv = 38.8;    %vertical opening angle of webcam [deg]
a_camt = 31;    %tilt of webcam [deg]
sep = 0.314;   % distance betw. laser & cam [m]
shift = 0.084; % distance betw. laser & rotational axis [m]
icols = 1280;  % column resolution of webcam



%load text file into matrix
M = dlmread(posfilename);
L = dlmread(intfilename);


%normalize intensities
L = L-min(L(:));
L = L./max(L(:));


%filter outliers
M(M<10) = NaN;
M(M>max(M(:)-10));


% surf(M)


irows = size(M,2);
nsteps = size(M,1);


%transform angles to rad
a_rot = a_rot*pi/180;
a_camh = a_camh*pi/180;
a_camv = a_camv*pi/180;
a_camt = a_camt*pi/180;
```

```
%transform pixel-coordinates to proper 3D cartesian coordinates
%calculate distances and put them in the same matrix M
M=icols-M;
M = sep./tan(a_camt+atan((1-M.*2./icols).*tan(a_camh./2)));
%calculate heights
H = M.*(1-2.*repmat((1:irows),nsteps,1)./irows).*tan(a_camv./2);


%matrix containing rotation angles per point
A=(nsteps-(1:nsteps).'*ones(1,irows));
A=A./nsteps.*a_rot;


K(:,1) = cos(A(:)).*shift-sin(A(:)).*M(:);
K(:,2) = sin(A(:)).*shift+cos(A(:)).*M(:);
K(:,3) = H(:);




%delete NaN-rows
L = L(~isnan(K(:,1)));
K = K(~isnan(K(:,1)),:);
%remove identical points
[K,ia,ic] = unique(K,'rows');
L = L(ia);



%plot everything

h=figure(1)
set(h,'Renderer','opengl');
scatter3(K(:,1),K(:,2),K(:,3),0.1,L(:),'.');
%surf(M);
colormap(jet);
set(gca,'Drawmode','fast','DataAspectRatio',[1 1 1],'PlotBoxAspectRatio',[1 1
1]);

% figure(2)
% DT = delaunayTriangulation(K);
% tetramesh(DT);
% set(gca,'Drawmode','fast','DataAspectRatio',[1 1 1],'PlotBoxAspectRatio',[1 1
1]);
```

Possible improvements

- Everything on this 3D-scanner is kept as simple as possible. Especially the software is not very sophisticated. For instance the C++/OpenCV code does not run stable. I sometimes have to manually reconnect the webcam after every scan, probably due to some problem with video4linux.
- The communication with Arduino does not always run smoothely either. It sometimes pauses or misses steps when it shouldn't. That is probably due to buffer errors, and shouldn't be surprising with all that buffering, reading and erasing that is happening between PC, RS232-converter and Atmega328. A microcontroller with native USB support would likely do better.
- If your webcam is not so great you will get a lot of noise. Simple thresholding as done here won't do to keep noise from registering as maxima and thus creating noisy point clouds in your 3D image. You could accumulate several images per step and add them up (=average them). This will improve your SNR. Another way is to get a stronger laser => a strong one will allow you to take scans in daylight conditions instead of the dark (select a laser-colour that is not very common in the scenery you want to scan – you could even use a NIR-laser and put a NIR-bandpass filter in front of the webcam).
- Point-clouds are not very useful. If you want to use your scan for 3D-modelling, you could spatially filter and triangulate your data with the PCL-library (http://pointclouds.org/)
- You could also save the RGB-colour of the maxima-positions in a second scan-run without the laser and thus create a proper colour map instead of the intensity map. This obviously does not work in the dark and thus your SNR will be worse.

References

ST LM317 datasheet:

http://www.st.com/web/en/resource/technical/document/datasheet/CD00000455.pdf

ST L279 datasheet:

http://www.st.com/st-web-
ui/static/active/en/resource/technical/document/datasheet/CD00000063.pdf

ST L279N datasheet:

http://www.st.com/st-web-
ui/static/active/en/resource/technical/document/datasheet/CD00000240.pdf

A4988POW datasheet:

http://www.robotikhardware.de/download/A4988POW.pdf

How to build a laser (-driver):

http://laserpointerforums.com/f51/i-want-build-laser-thread-52972.html

http://www.loneoceans.com/projects/project405/index.htm

Arduino Uno:

http://arduino.cc/en/Main/ArduinoBoardUno

Arduino serial communication:

http://arduino.cc/en/Tutorial/SerialCallResponseASCII

Arduino reading ASCII-strings:

http://arduino.cc/en/Tutorial/ReadASCIIString

Arduino interfacing:

http://playground.arduino.cc//Interfacing/CPlusPlus

OpenCV documentation:

http://docs.opencv.org/

Point Cloud Library:

http://pointclouds.org/

That helped a lot when I did the trigonometric calculations that are necessary to reconstruct 3D-coordinates from the pixel-coordinates extracted by the scan (or any maths really):

http://www.mathcentre.ac.uk/resources/uploaded/43799-maths-centre-ff-for-web.pdf

License

This article as well as all the ideas, text, images (those images that are solely created by the author) and source code it contains are published under GNU General Public License Version 3.0 (GPLv3). See `https://www.gnu.org/copyleft/gpl.html` for reference.

The author does not claim that anything described in this article will necessarily work, nor be safe on people or property, nor be in accord with electronic or other safety regulations and standards.