**Luci, my Lamp**



<https://www.youtube.com/watch?v=dPnp3uZqJPc>

I will take a pictiure a

A lamp is quite useless. Does not talk, cannot dance and is boring. This is where I try to help.

Luci is an autonomous lamp with a webcam, a power LED in the shade, and five servo motors. She is controlled by an ODROID U3. Once switched on, she looks around checking the environment for any beings and does what she wants.

**Mechanics**

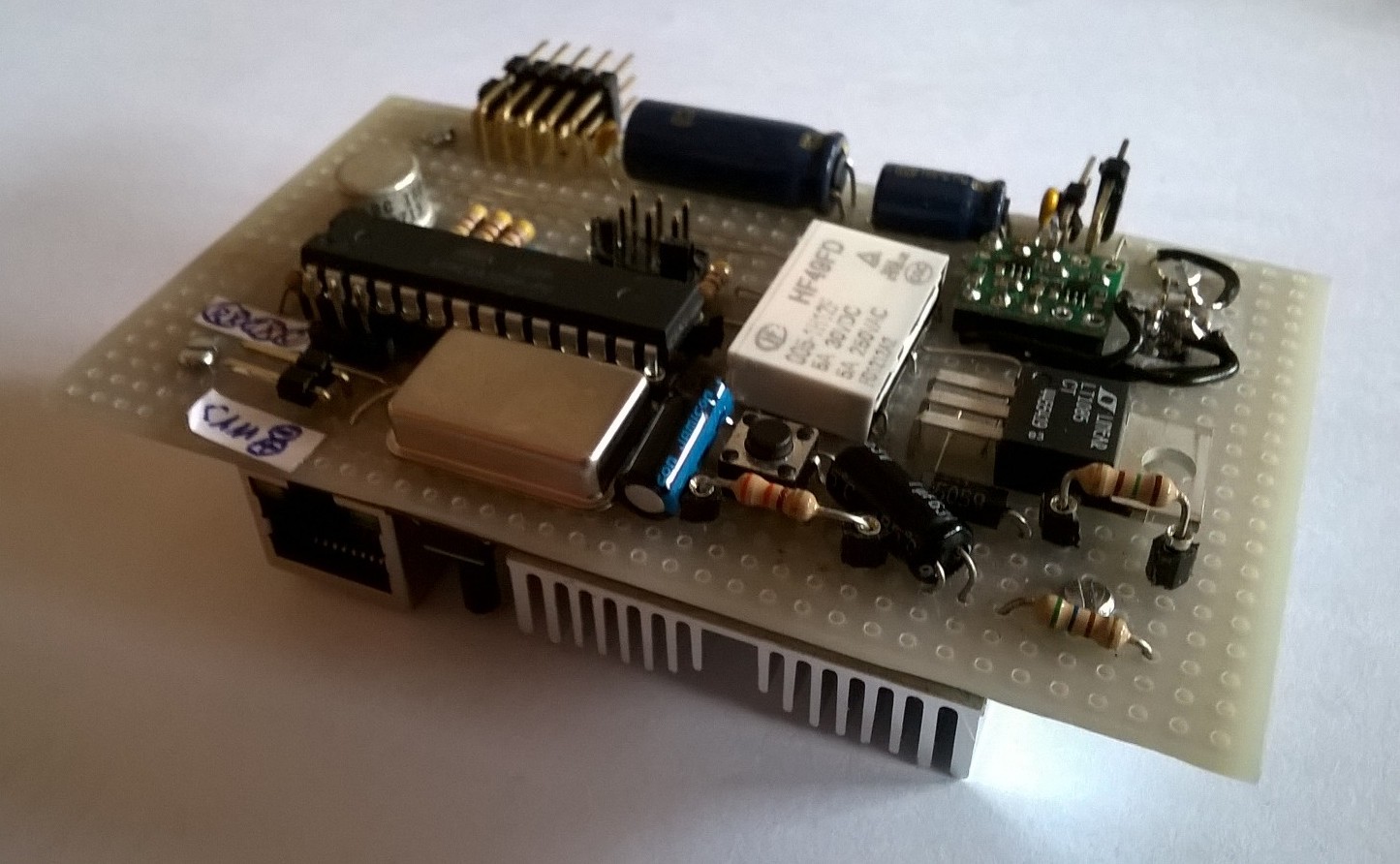
First, I bought a desk lamp, measured the basic dimensions and tried to give it more childlike characteristics. TurboCAD did help to model the lampshade and the hinges.

After having spent hours in the basement and cutting the hinges out of multi-layered birch wood, it turned out that the friction between pole and hinges was too high. Ok. I reconstructed it with ball bearings, went back to the basement, tried again, noticed that the springs need different mount points, changed that, went back in the dark and breathed in fine dust again, and again. The shade is not wooden, but has been 3d-printed of ABS.

**Hardware**

When the mechanics seemed fine, hardware was next. I started with a beagleboard, but it became clear very soon that it is too slow for proper face recognition, so I ordered an Odroid U3. Unfortunately the Odroid does not provide five PWM outputs for the servos, so there was the need of a separate ATmega that produces the signal for the servos as well as the power LED in the shade. The result is an Odroid with a self-made shield on top. The sandwich was supposed to be placed in the base of the lamp, but in the end I never placed it there since I was scared of having a hot odroid being under full load within a wooden housing.



**Software**The software runs in C++ on the ODROID with multiple threads to leverage most of the cpu power. The first thread grabs the image of the webcam; a second thread takes these images and tries to find all faces currently looking at Luci. The result is a list of faces; Luci focuses on the biggest one.



A third thread is running the *trajectory planning* generating movements, which is a sequence of point and orientation in 3-dimensional space. When no face is detected, Luci runs a search pattern looking for faces by sampling the environment until a face has been found. Then, Luci carries out a pattern simulating real emotions like nodding without knowing why, pretending to listen, coming closer or more distant depending on the movements of the face. It’s like in real life at home.



Finally, the main loop takes the current and next point of the trajectory and interpolates all intermediate points with 60Hz using a cubic bezier curve to smooth the movement. The support points of the bezier curve are geometrically derived from the trajectory’s previous and next point by the rule shown in the picture.

**Mass inertia**

This transformation step also computes the shade’s acceleration, since the bezier curve alone does not take into account that in total 400 grams are moved. This acceleration is limited by ½ g to prevent luffing caused by the elastic construction and the backslash of the servo motors. The position is amended accordingly such that the actual movement is following the bezier curve with a delay and distance once the acceleration becomes too high (professional robots do this on base of the fully modelled inertia, but at some point in time you get tired of doing maths for hours, so I did not try that).



**Kinematics**

The output of all this is just a point which is now passed to the kinematics module that computes the to-be angles of all servo motors. This part is textbook robotics, it works as follows:



Algorithm starts with the point / orientation (=tensor) of the head’s center *A*. I built a small library that is able to carry out geometric operations on tensors, to make the kinematics code compact. It provides the main class Tensor, Point and Orientation (In the following code I removed all C++ specific overhead and left the bare content only).

class Tensor {  
 Point pos; // contains x,y,z  
 Orientation ori; // contains x,y,z  
}

class Point {  
 Point(float x, float y, float z);  
 Point translate (Point p);  
 // rotate \*this around c with angles o with rotation matrices  
 Point rotate(Point c, Orientation o);   
 // return missing point c of a triangle via cosinus law  
 static Point triangle(Point a, Point B, float lengthAC, float lengthCB);   
 // compute distance to point b  
 float distance(Point b);   
 // more geometric constructors and operators like \*,+,- …  
}

void Kinematics::moveIt(Tensor a) {  
// compute Tensor B out of A  
// by taking unrotated B relative to A and rotating along orientation of shade   
float distanceAB = lengthShade; // length of the shade  
Point b = a.translate(Point(distanceAB,0,0).rotate(a.pos, a.ori));  
// compute servo angle at points E, C and B  
float angleBase = b.atan2(a.pos.z,a.pos.x); // servo at E  
float angleNeckTurn = a.ori.y – servoAngleBase; // servo at C  
float angleHeadNick = a.ori.z; // servo at B  
// compute Point C by taking B and rotating  
Point vectorBC(-lengthNeck, heightNeck,0);  
Point c = b.translate (vectorBC.rotate(Orientation(b,servoAngleBase,a.z)

// Compute point D via cosinus law out of points e,c  
Point d = Point::triangle(e,c,lengthLowerArm, lengthUpperArm);  
float angleLowerArm = atan(d.y-e.y, e.distance(e));  
float angleUpperArm = (180-angleLowerArm) + 90 + atan(c.y-d.y, c.distance(d));  
// …  
}

These angles are passed via I2C to the ATMega, where the Arduino library generates a 60Hz PWM signal and sends it to the servos.

**Face Recognition**

The face recognition module uses opencv 3.0 with haar cascades. Although the newer LBP cascades (without floats) were significantly faster, they had a lot more false positives, so I thought 10 fps with haar cascades is ok for me. Out of the 2d-position of a detected face, the 3d position is estimated assuming a standard face size which worked surprisingly well. Later on, Luci’s Trajectory Planning module moves towards the face if it is very close to simulate real interest and moves away if it violates the European intimacy distance. Following a face in realtime was a bit tricky, since the webcam has a latency of 0,4s, so the computation of the face’s 3d position needs to be done relatively to the position Luci had 0,4s ago. Still, during fast movements this did not work satisfyingly, so the orientation of the head is done towards the last stable face position until Luci moves slower and following the face in realtime becomes possible again.

**The project changed me**

not at all. Like in the office, everything just took longer than expected. Surprisingly the software and hardware worked out quickly, but getting the construction and the mechanics in a shape that worked, is not too heavy, with properly mounted servos and springs consumed two months‘ weekends in the basement. Maths were definitely challenging. Getting face recognition done was the simplest and fastest part, but still gives the most ahhs and ohhs. The guys from opencv did a pretty good job to make this really easy. Most fun was the Trajectory planning, i.e. how Luci should move when the webcam recognizes a face moving.

I have been thinking of enhancing Luci with an acoustic beat detection allowing her to do cool moves matching the music. First thoughts were come on, how difficult could that be, but rhythm detection seems to be really, really tricky. So, let’s see….

**Main Parts List**

* Odroid U3 running Ubuntu 14.04.02 with latest g++
* Software uses openCV 3.0 and Boost 1.57 as base libraries
* ATmega 328 running C++ firmware based on Arduino Library for Servos and I2C
* Servos from Hitec: 77B (for turning base & nicking the head), 7954SH (lower arm, strong & expensive), 7775MG (upper arm, also expensive), 5065MG (turn head inside the shade)
* 3D print of a TurboCAD model made of ABS
* Springs from my local dealer, 20 ball bearings, 0.5m2 multi-layered birch, and several brass axis



Jochen Alt  
Germany, jochen.alt@gmx.net  
*Two years ago, I was bored by too much Excel in the office (profession-wise I’m a software architect), started learning electronics and embedded stuff and made my first robot (https://www.youtube.com/watch?v=ak5ua5bKloc).*