Project on Google Code by Basil Shikin: https://code.google.com/p/vintage-phone/

Introduction

The goal of this project was to re-create a XIX century phone call experience and esthetics. A user would pick up the receiver, tell "operator" whom to call and a call would be placed via VoIP.

I have used Android-based device, IOIO Board and some wood and plastic-cutting to re-create this device.

Here is small video that explains what I have managed to build:

http://youtu.be/nuUQQJHx-vg

Construction

Components

A small research led me to using following components:

Vintage phone for project enclosure. I was very attracted by the esthetics of this old device. Mix of materials, rich textures and shapes certainly add to the experience. I found a reasonably priced candlestick phone and a ringer box on eBay.

Archos 28 as a driving device. Archos 28 is a reasonably priced tablet that has all the features I need: 4Gb of internal memory, Wi-Fi, microphone, audio out and 800Mhz CPU.

One might ask: why not use a micro controller and a set of chips? It looked a bit simpler and more efficient to use Archos 28, as it has all components on its board and also comes with OS Android. Since my phone has to work 24/7 it has to remain plugged in all the time, so power consumption is not an issue.

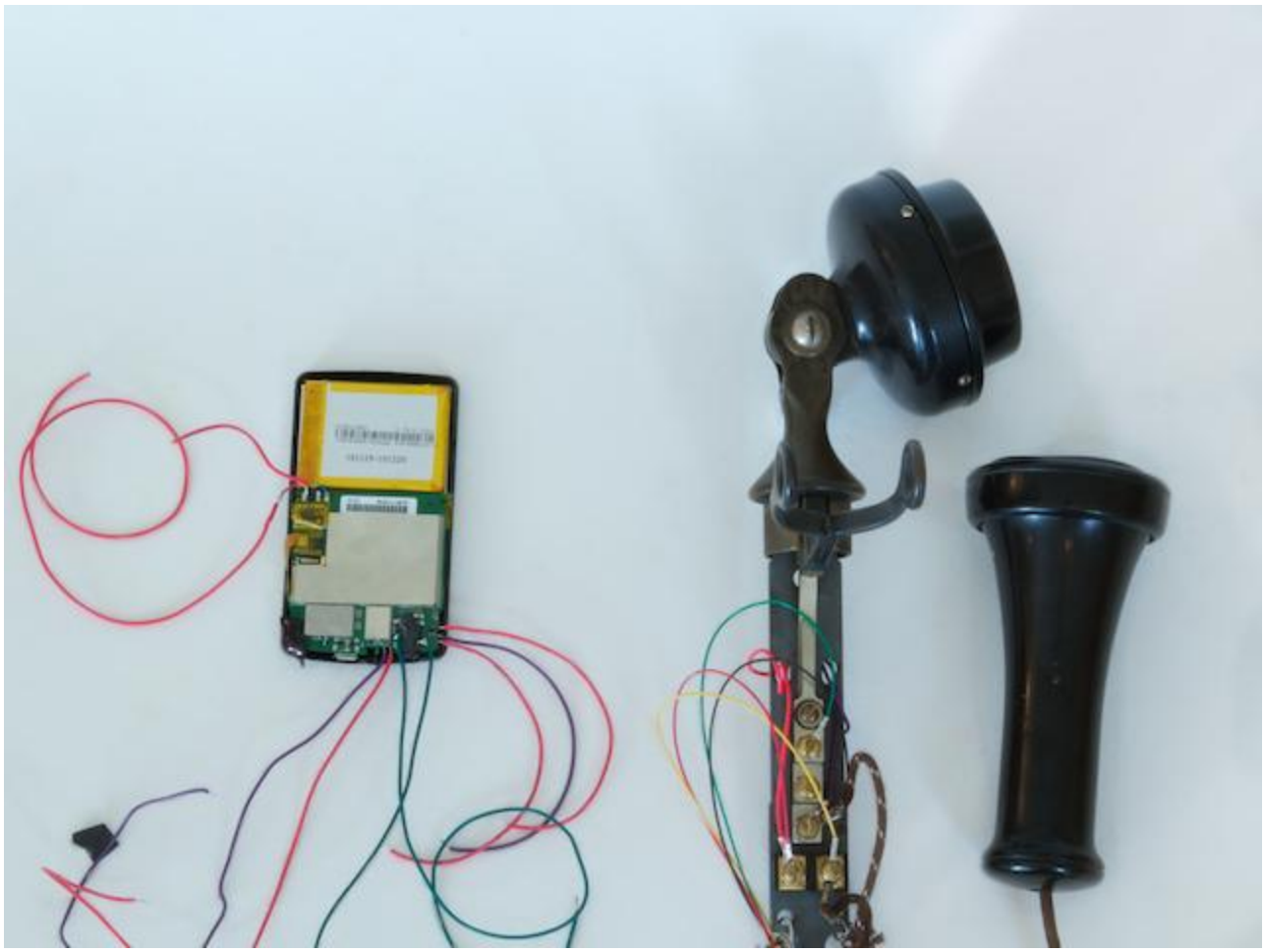
IOIO Board to interact with hardware. IOIO Board is an amazing device: it plugs into Android device via USB. Android device discovers it as an ADB host. There is a nice little API that allows any Android application read line state (either digitally or do analog read) and generate either digital or PWM signal on a line.

One might ask: why not use Android ADK? Unfortunately, ADK has been added only in Android 2.3. Archos 28 is running 2.2.

Metal, Wood and Plastic

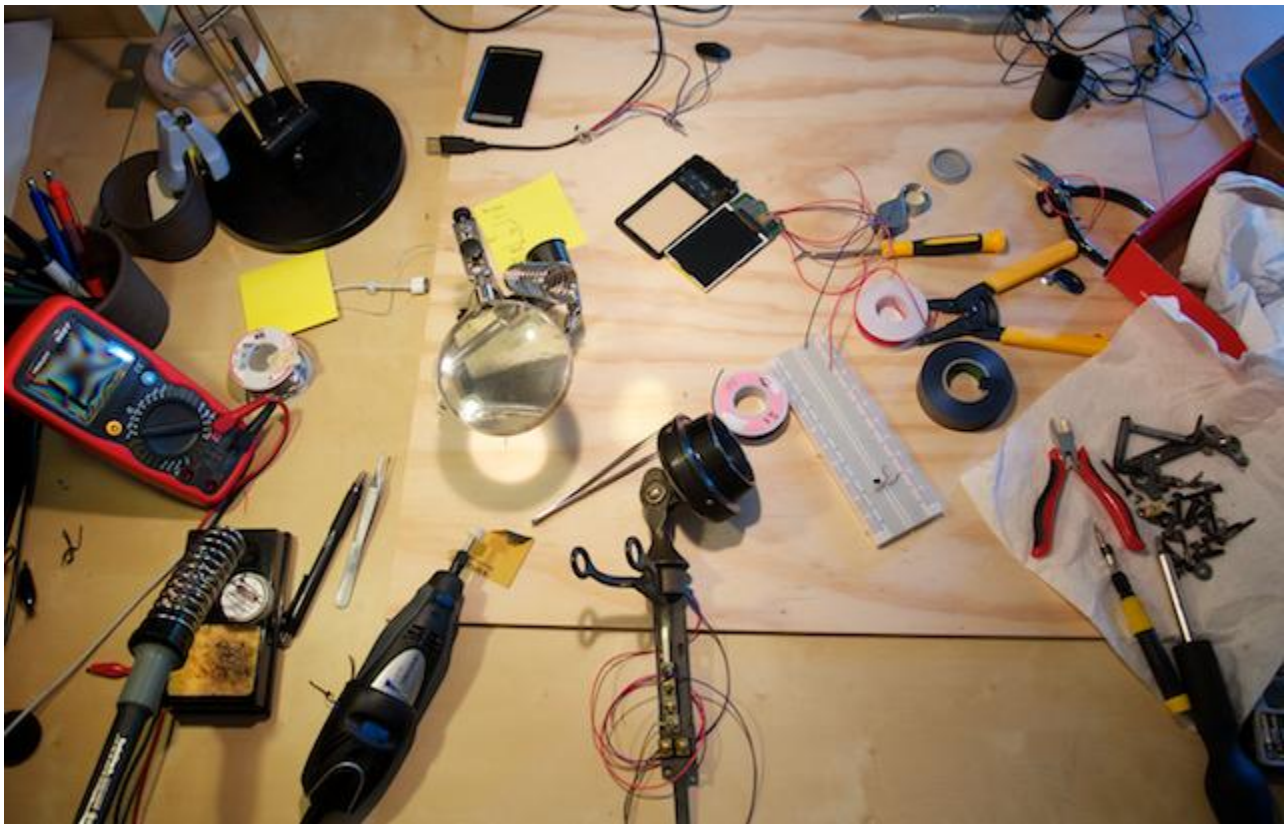
I disassembled vintage phone and phone box to clean it. Since the device was manufactured around 1910 it had quite a lot of dirt, dust and old glue in it. I used warm soapy water for wood and WD-40 for metals parts. Here are all the components laid out:

Although Archos 28 is certainly not intended for amateur tinkering it was not that difficult to wire out power button, microphone, audio out and Wi-Fi antenna. Candlestick holder of and old devices was also very simple to work with: it had a simple mechanism that would bring a line to the ground if the telephone is on hook. Here is how my devices looked like:

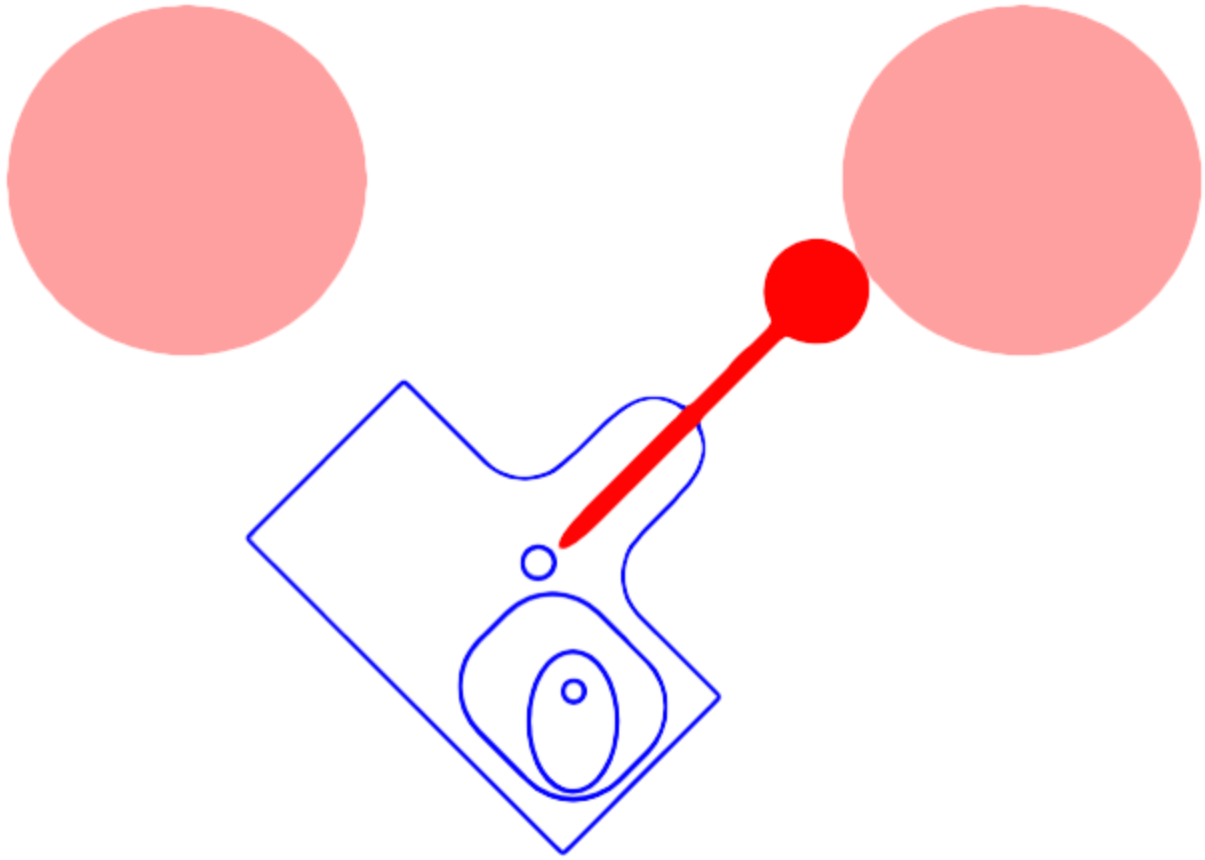


Archos 28 didn't fit the old ringer box very well. To be exact it was USB cable to was sticking out and preventing device to fit in. I used Dremel tool to fix this. I cut a small slope to make sure USB cable fits in nicely. I have also cut out holes for wires and installed couple hinges:

Somewhere along the process I was surprised to discover that my table resembled that of Time Traveler, who has just returned to his XIX-st century from our XII-st:

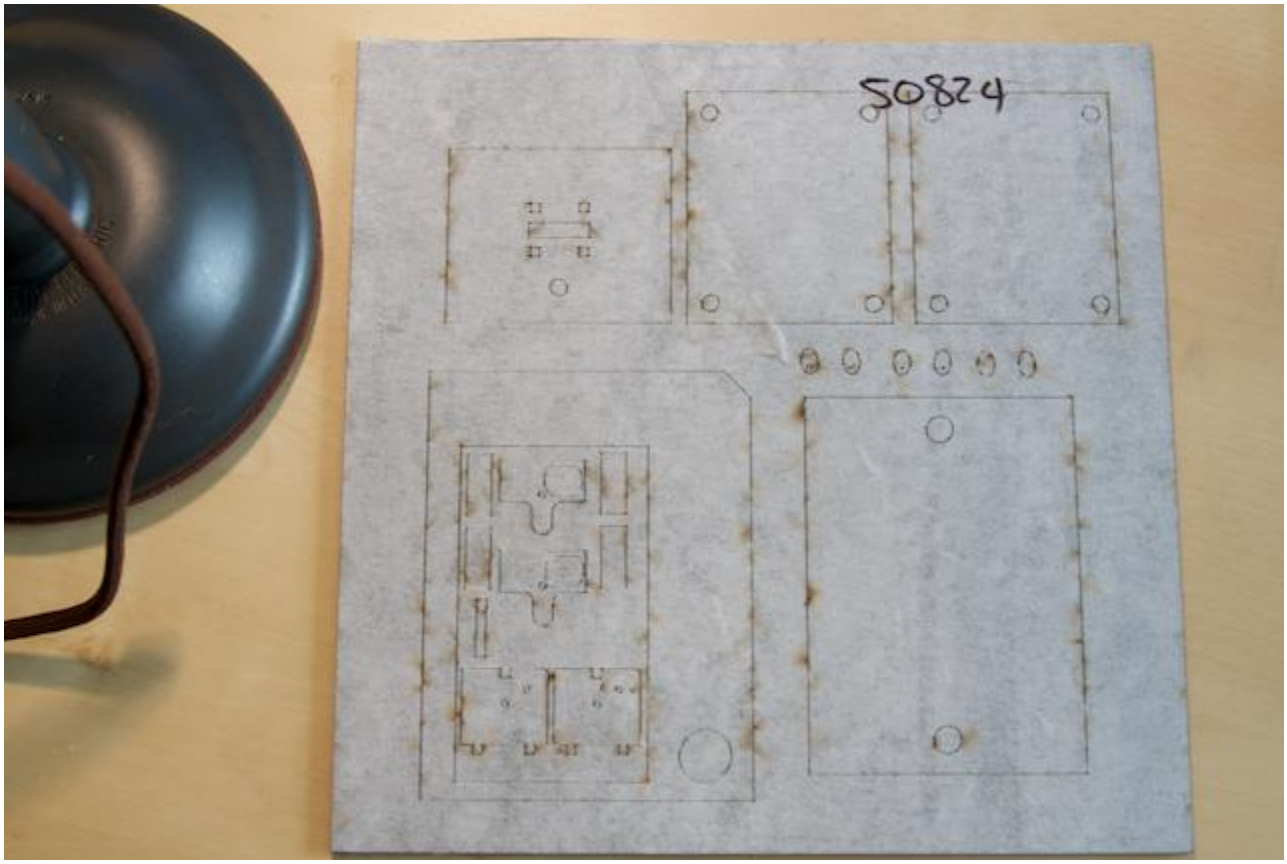


I really wanted to keep authentic vintage ringer. Unfortunately, I couldn't leverage any of the original designs: old electromagnets took up all space in the finder box. So, I had to throw everything away and design ringer mechanism from scratch. I mounted original metal ball (that hits the chimes) on a pivoting platform that it driven by a small servo motor. The motor has an ellipse on its axis. This ellipse rotates and pivots the platform. Here is how it looks on a diagram:



I used Inkscape to create 2D design. I must mention that using this tool very much feels like a torture. I really hope I'll be able to find better affordable tools for my next project.

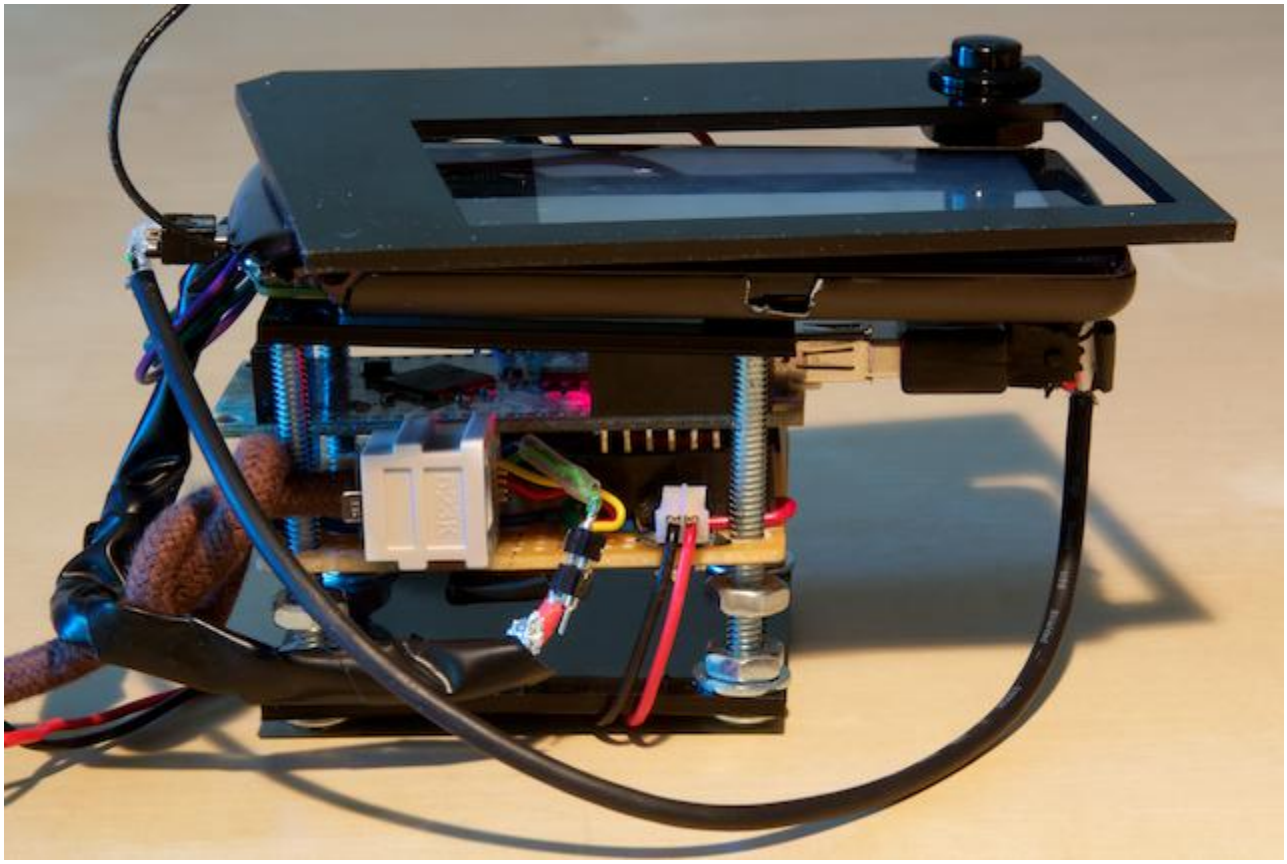
One the design was ready, I've submitted it to Ponoko for laster cutting. Laser-cut panel arrived couple weeks later:



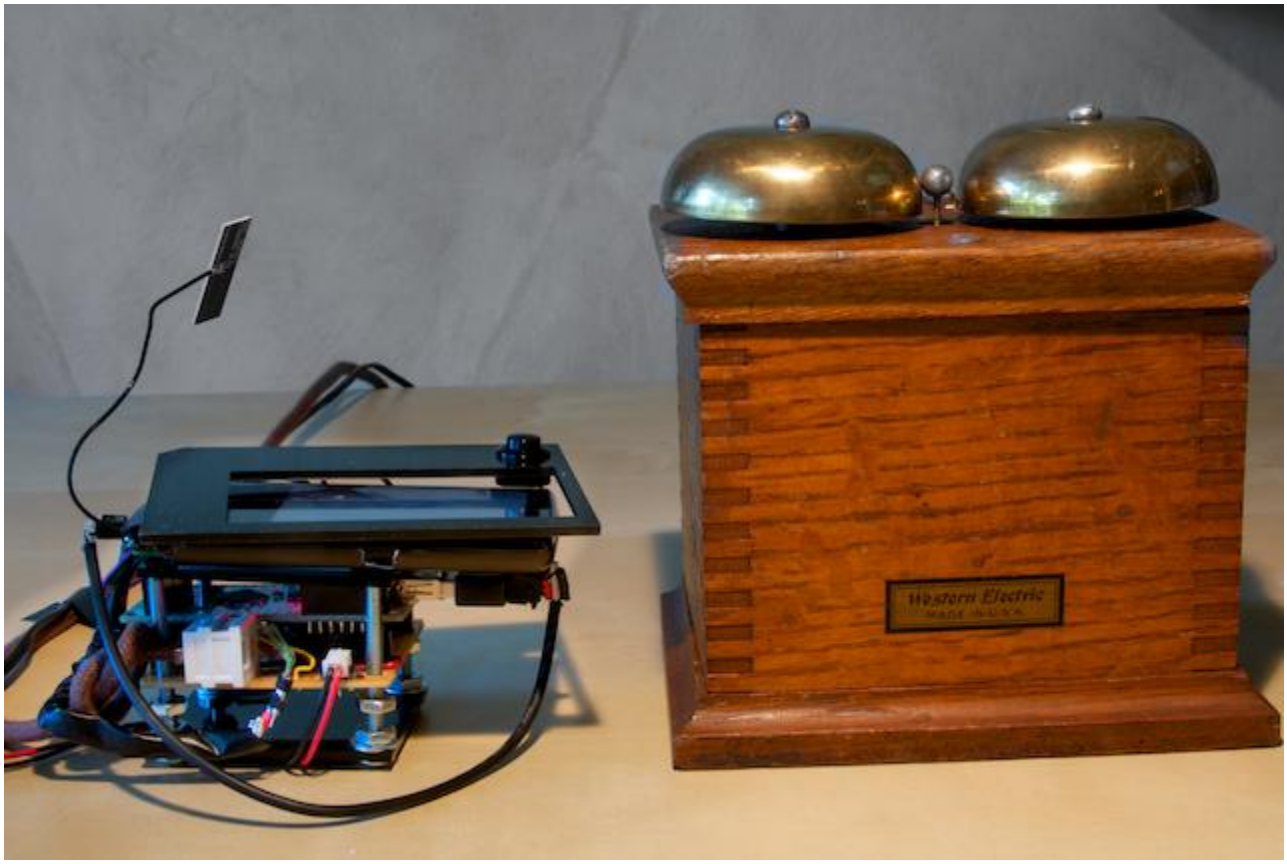
Add some glue and these components transform into a ringer mechanism:

Circuit Assembly

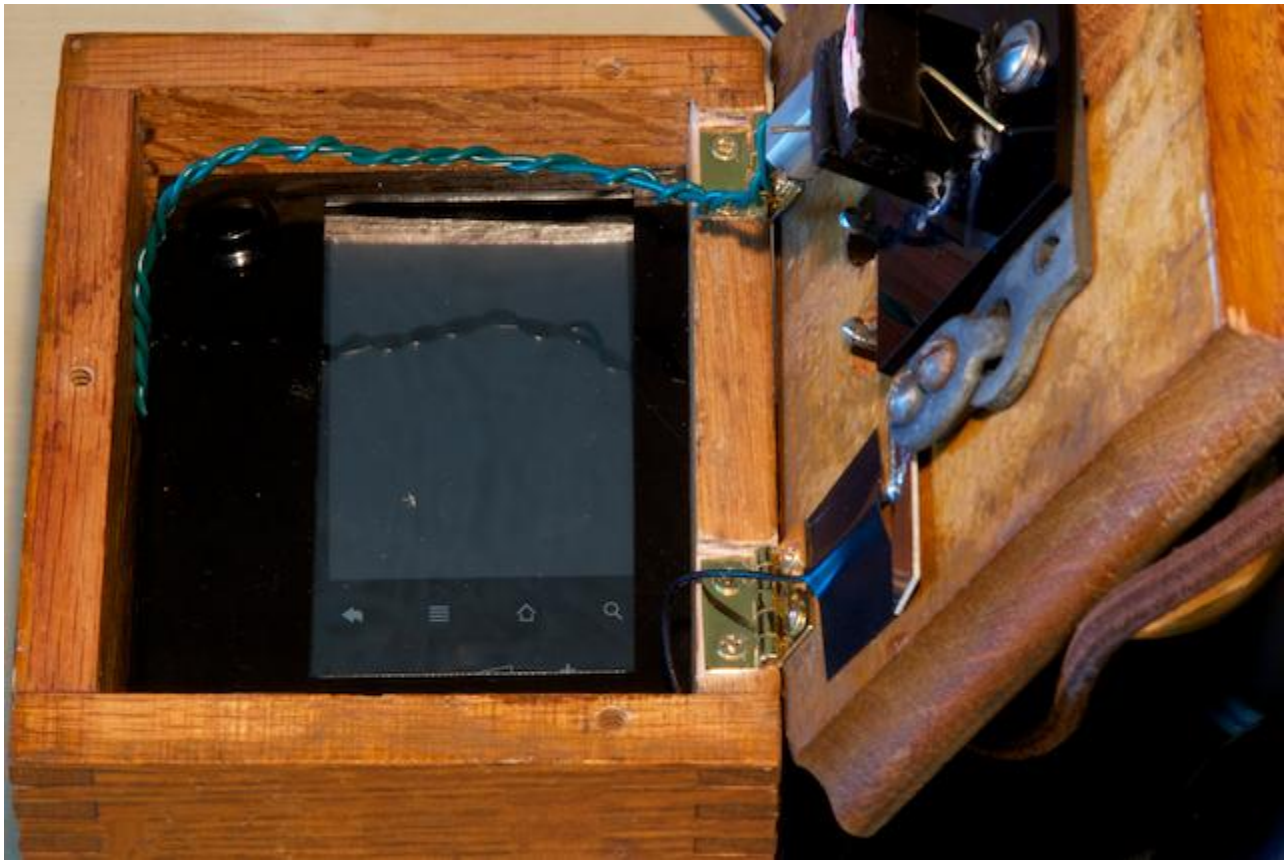
There was really not much to the device's circuit. Archos 28 was connected to IOIO Board via USB. The IOIO Board was connected to candlestick "on/off hook" line directly. It was also connected to the servo motor via TS1220-600T. Here is how it looked:



One can imagine, how it fit into the enclosure:



Once the circuit was in the ringer box Wi-Fi quality reduced significantly. So, I had to purchase and attach an external Wi-Fi antenna. Archos 28 happens to have a very well defined UF.L port. Here is how the system fit into the ringer box:



Here is how wires looked like:

Software

Primary Components

CMU Sphinx is an open source voice recognition project maintained by Carnegie Mellon. The system consists of two parts: recognizer code and files with voice model and language model. It was easy to compile library code for Android. There is a great example posted by CMU Sphinx's creators. One can teach CMU Sphinx their own pronunciation. All one has to do is to record 20 sentences and run generated files thought a supplied tool. This can significantly increase recognition quality. What is more, one can build a language model. This would basically tell recognizer what words and phrases to expect. In my case a primary phrase was "call *name*", where *name* is one of the names from my address book. Having such model also increases recognition quality.

One might ask: why not use Google Voice? Unfortunately, it is really bad at understanding my pronunciation. And it also not so good at recognizing names.

One might ask: why not use special micro controller? I have certainly considered this approach. One solution I found was Sensory. Unfortunately, it looked too expensive. Well, it seemed like I would have to do the same amount of work, as with CMU Sphinx and it will result in comparable quality, but I would have to pay for the chip.

"**No speech generator**" – I was very convinced in this after trying several different generators. All text-to-speech engines created a very un-natural voice. So, I had to ask a human to record all phrases that my phone can possibly tell. What is more, I had her read each phrase several time. During playback I pick a random version of the phrase; this creates a strong illusion of a real human on the other end.

PJSIP – is an open-source implementation of the SIP stack. In other words, it is open VoIP library. I didn't have much trouble with it: downloaded, compiled and used it. CSipSimple is a big project open source that also uses it. This project very helpful, as it contained some great usage examples.

One might ask: why not use Skype? This was my original idea. I've subscribed to Skype Developer Program. Unfortunately reading license agreement revealed that Skype SDK can not be installed on any devices controlled by Android.

One might ask: why not SIP stack that is built into Android? Unfortunately, the stack has been added only in Android 2.3. Archos 28 is running 2.2.

Workflow

When telephone is off the hook:

1. Wait one second
2. Say "Number, please!"
3. Start voice recognition
4. If recognized "call *name*", go to next, otherwise say "Sorry, I didn't get that" and go to 3
5. Say "Calling *name*..."
6. Start voice recognition
7. If recognized "no" or "stop" go to 2, otherwise go to next
8. Place a VoIP call
9. Say "Call placed"
10. Wait until the call is terminated
11. Say "Call terminated"

When incoming call is received ring the bell and wait until either telephone is picked up, or other end terminate a call or 20 seconds pass. Ring the bell with one second intervals.

Android App Format

Phone application is actually a background service. There is also a light-wait user application that displays current status. The services starts on app startup or on user app launch.