Grave Robbing Our Way out of the Global Chip Shortage: Buildroot to Run Nerves on a Gen 1 Amazon Echo

Background

Elixir is a functional programming language built on Erlang's BEAM VM. There is an Elixir library called Nerves which is an "open-source platform that combines the BEAM virtual machine and Elixir ecosystem to easily build and deploy production embedded systems."

Nerves supports a variety of targets by default, which include Raspberry Pi 0, 2, 3,4, BeagleBone Black, Green, and others. There are also community-supported boards that are unofficially supported.

There is a global chip shortage which has made the supported Nerves targets difficult to obtain. Amazon's Alexa devices have been deemed a total failure, these devices include the Amazon Echo, Echo Dot, etc.. As mentioned in this YouTube video, these devices are packed with great hardware and are cheap to buy now.

Let's make an Amazon Alexa device a community-supported Nerves device.

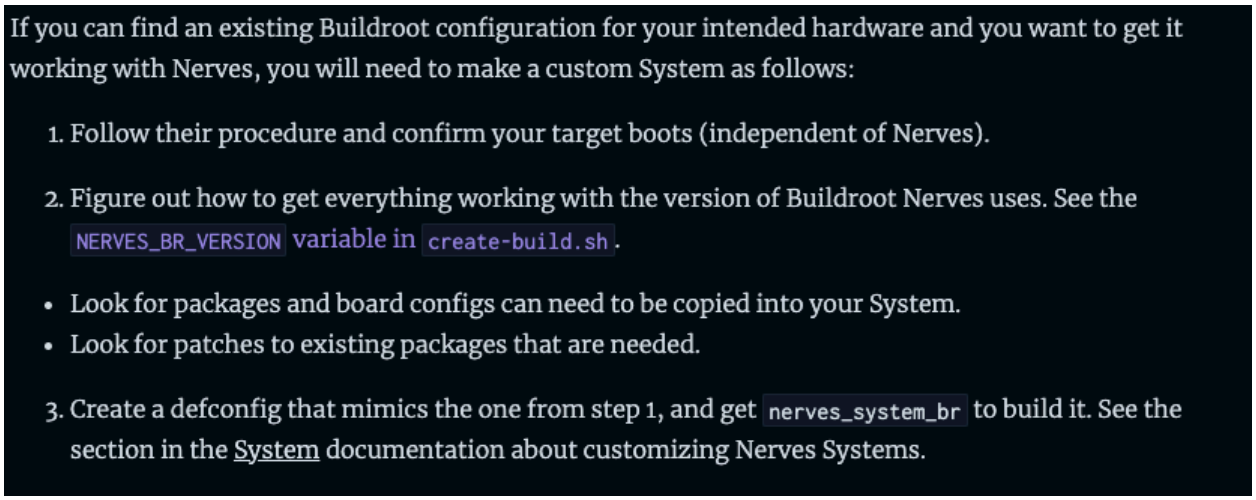
Overview
This document will:

- Explain the goal
- Link to Clinton Cook whitepaper, which provides a proof of concept for booting the Gen 1 Amazon Echo from an external SD card via exposed debug pads
- Link to Amazon's open-source Gen 1 Amazon Echo source code
- Explain that the Gen 1 Echo uses the TI DM3725 processor
- Show how to setup/install the TI DM3725 SDK, which is the SDK for the DM3725 processor (may or may not be needed to accomplish the goal)
    - The end of the document provides a sources.list file needed for the SDK
- Link to a blog that demonstrates creating a bootable SD card for the Gen 1 Amazon Echo by obtaining Amazon source code and pairing it with TI DM3725 SDK files
- Link to a blog and Gitlab repo that demonstrates using Makefiles to create a bootable SD card for the Gen 1 Amazon Echo

Goal

The goal is to make an Amazon device compatible with Nerves by using the Buildroot version (2022.11.3) that Nerves uses. The first task is specifically for Buildroot. This involves booting the device separately from Nerves, copying packages and board configurations (presumably from publicly available Amazon source code provided in a link below), and applying patches. **The minimum viable product (MVP) for this task does not need to support all the services of the device, but it should be able to support wifi and SSH**. **It should also be done in a way that allows for adding support for other services gradually in the future, and it should allow Nerves to run.** The plan for this project is to open source it once complete, so that other developers can contribute in adding support for the other services of the Amazon device.

The Nerves documentation lists what's required for supporting new hardware:



If you can find an existing Buildroot configuration for your intended hardware and you want to get it working with Nerves, you will need to make a custom System as follows:

1. Follow their procedure and confirm your target boots (independent of Nerves).
2. Figure out how to get everything working with the version of Buildroot Nerves uses. See the `NERVES_BR_VERSION` variable in `create-build.sh`.
   - Look for packages and board configs can need to be copied into your System.
   - Look for patches to existing packages that are needed.
3. Create a defconfig that mimics the one from step 1, and get `nerves_system_br` to build it. See the section in the System documentation about customizing Nerves Systems.

All the services provided by the Amazon device need not be supported initially as it is hoped that this will be an open-source repo with community-driven contributions to add the services in time. The MVP is to be able to use Buildroot to create an SD card such that the Amazon device boots and can run Nerves with wifi and SSH working, in such a manner such that support for other Amazon device services can be added later.

Technical Data

- The device we will use will be the generation 1 Amazon Echo
  - This is because research has already been done to show that debug pads on the bottom of the gen 1 devices allow for booting the device from an external SD card, UART is also exposed. The details of this research can be seen on the Clinton Cook white paper

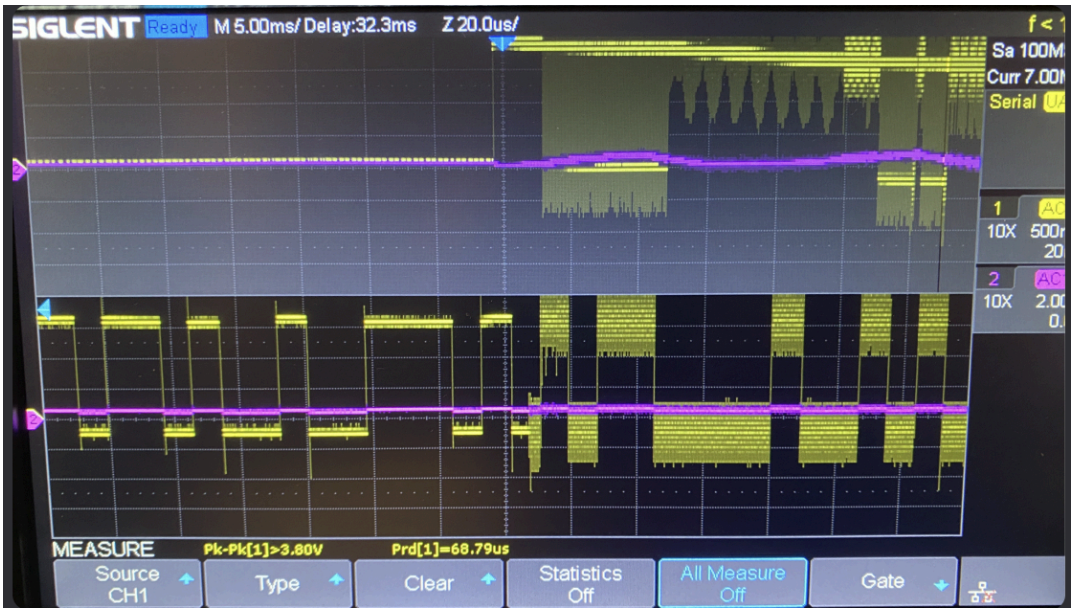
    Here you can see the log as read through UART as far as I have gotten. However, - I read through the Amazon source code I have and here it would report something like " Starting X-loader on mmc-0...custom_efi: invalid

signature". However, this proved to be a misleading error message based on what I saw in the source code. That error message is printed from line 119 of Echo_src_5.5.3.7_build653619520/ x-load/disk/part_custom.c which is a call to `get_partition_custom_efi`. I say it's misleading because it doesn't just error and exit, instead it tries to get the custom dos partition (via `get_partition_custom_dos`) IF `get_partition_custom_efi` returns a non-zero. By tracing through `get_partition_custom_dos` and looking at the attached log file it didn't appear that the printed error message was actually causing it to stop trying to boot from the SD card as that function logs more stuff in some error conditionals which I don't see being logged:

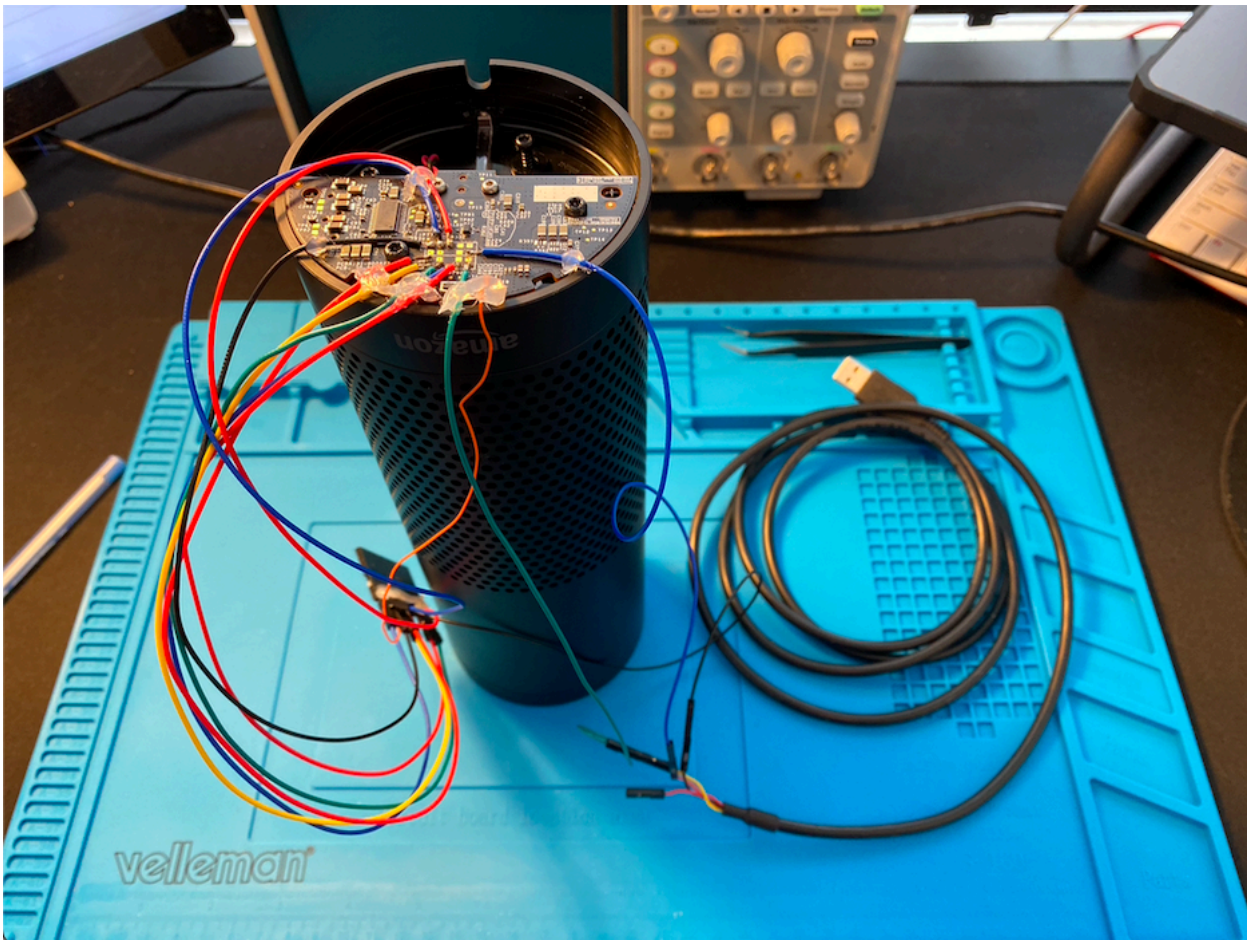
```
Texas Instruments X-Loader 1.51 (Jan 31 2017 - 01:24:36)
LAB126 Rev 0
Starting X-loader on mmc-0...custom_efi: invalid signature
Reading boot sector

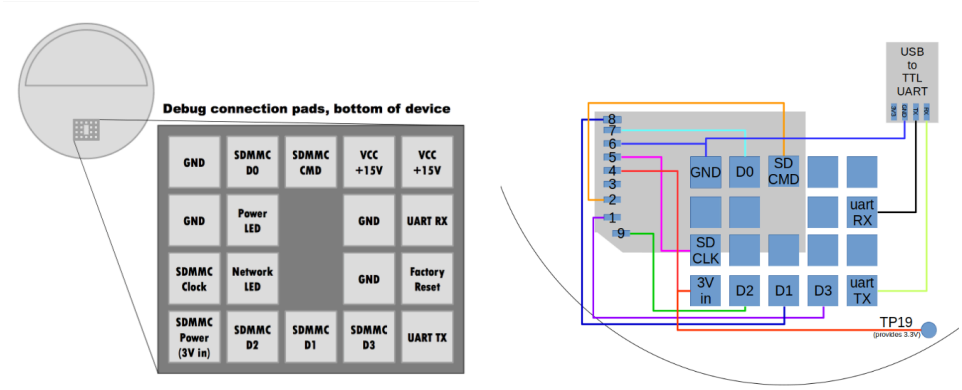
229036 Bytes Read from MMC
Starting OS Bootloader from MMC...
Starting OS Bootloader...(time = 855 ms)
```

However, when looking at the UART signal above, I saw that the next thing to happen was that a bunch of garbage was printed to the screen. It looked to me like a change in baud rate, so I hooked my scope up to it (attached a screenshot) to try to figure out what that baud rate may be. You'll see that the trace gets really weird where the garbage gets printed. Since it doesn't really look like normal UART signals where the garbage starts getting printed, I just tried a number of different typical baud rates- but nothing worked. I think that maybe the garbage is getting printed because I am using random u-boot stuff and random defconfigs, so maybe it's looking in memory for some valid u-boot stuff but is instead finding unintended data at the memory address. It looks like xboot is trying to read the filesystem but cant. Amazon changed it from ext2 to ext3. I was using FAT32 for the first partition, but ext 4 for the other 2 partitions- perhaps this is the issue…



- The script (mk3PartSDCard) for creating the 3 partition SD card that's mentioned in the Clint Cook paper is viewable here. That same page also lists a helpful command for copying the files into the intended partitions of the SD card (cpSDCard)
- My device, wired and ready to try SD cards





- The gen 1 Amazon Echo uses the Texas Instruments DM3725 processor

- Texas Instruments provides an SDK for the DM3725 should it be helpful to use their SDK to generate any needed files
    - SDK is available here (click “Download options”, then download “dvsdk_dm3730-evm_04_03_00_06_setuplinux”
    - The SDK must be ran in 32-bit Ubuntu 10.04 (checksums here), which likely means you’ll need a VM since no one has such an old system laying around
    - You will need to update your VM’s `/etc/apt/sources.list` because the sources for your VM’s apt will be outdated and not functional, and the SDK requires a couple packages to run. I can’t remember which 2 packages but it will be obvious because when running the SDK installer, the error message will tell you what’s missing. I provide a sources list at the end of this document, just append the contents of that to your VM’s `/etc/apt/sources.list` , the `sudo apt-get update`
    - The SDK’s Developer Guide is here, once you have your 32-bit Ubuntu 10.04 VM, you can download the SDK and follow the Developer Guide to run the installer. The TLDR is:
        - `export DVSDK="${HOME}/ti-dvsdk_dm3730-evm_xx_xx_xx_xx"`
        - ` ${DVSDK}/setup.sh`
        - The SDK installer will ask some questions. I mostly took the defaults, your guess is as good or better than mine as to what the correct answer for each of the installer’s questions are. But, for completeness I will tell you what I did:
            - *It will ask Select Linux kernel location and Select root file system* location
                - I choose `2` (SD card) for those
            - *The next question will be if you want to set up a script*
                - I answered `y`
            - *and then finally Would you like to run the setup script now (y/n)?* This requires you to connect the RS-232
                - I said `n` here because I think it’s assuming you’re setting up an evaluation module (some TI board that uses a DM3725) and that device is connected to your host via RS-232. Obviously, this does not apply for us
- The Amazon Echo Gen 1 source code is available in the Internet Archive’s Wayback Machine, not all the snapshots have a working download so if you don’t want to search for one this one works
- This blog has 2 article about rooting an Echo
    - Part 1
    - Part 2. In this part, he uses wireshark to sniff the Amazon Echo’s update download. He then says “We’ll need the files from the TI SDK to supplement what isn’t available in the Amazon packages”. He created a supposedly bootable SD card image, but it didn’t work for me. Anyway, his image does not use Buildroot and thus is not helpful
- This blog (repository here) does something similar to what I want to do, it uses the Amazon source files as well as a forked u-boot and creates a bootable SD card. The

problem here is that it uses a bunch of Makefiles rather than wrapping everything up in Buildroot. Since it uses these Make files, rather than Buildroot, I cannot use it to support a new Nerves target. But, maybe it's close or something we can use from Buildroot like telling Buildroot to use the u-boot from that repository?

- As an aside, this blog uses the linux 2.6.37 kernel because he wanted to support some microphone array, see issue here where a recent Linux kernel worked.
    - *"The reason for not using a new kernel is that you would lose microphone array "merging" which is done in the DSP, see https://andrerh.gitlab.io/echoroot/ .You might get the DSP working with 3.x, but it won't be easy and I didn't looked into it. Just search for omap dsp, the Beagleboard community has some ancient documentation for it. If you don't need the microphone array you could even use a mainline kernel, see https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/log/arch/arm/boot/dts/omap3-echo.dts …*
    You could in fact use a recent buildroot release, mine is only there to suit the 2.6 kernel which is not supported in later versions. My changes to it are only for newer package versions and build fixes for current build platforms. With a buildroot version from this decade you'll get a shiny up-to-date userspace for your 5.x kernel."
    - As mentioned, the MVP of this is to be able to run Nerves on the Echo, with wifi and SSH working. That means **that if we can set up buildroot such that the microphone array issue could be resolved later**, we could use a new kernel and a recent Buildroot release and let the community (or me) deal with fixing that issue later

Sources list for 32-bit Ubuntu 10.04 VM

append the following to /etc/apt/sources.list; then run sudo apt-get update

newer versions of the distribution.

deb http://old-releases.ubuntu.com/ubuntu/ lucid main restricted
deb-src http://old-releases.ubuntu.com/ubuntu/ lucid main restricted

Major bug fix updates produced after the final release of the
distribution.
deb http://old-releases.ubuntu.com/ubuntu/ lucid-updates main restricted
deb-src http://old-releases.ubuntu.com/ubuntu/ lucid-updates main restricted

N.B. software from this repository is ENTIRELY UNSUPPORTED by the Ubuntu
team. Also, please note that software in universe WILL NOT receive any
review or updates from the Ubuntu security team.
deb http://old-releases.ubuntu.com/ubuntu/ lucid universe

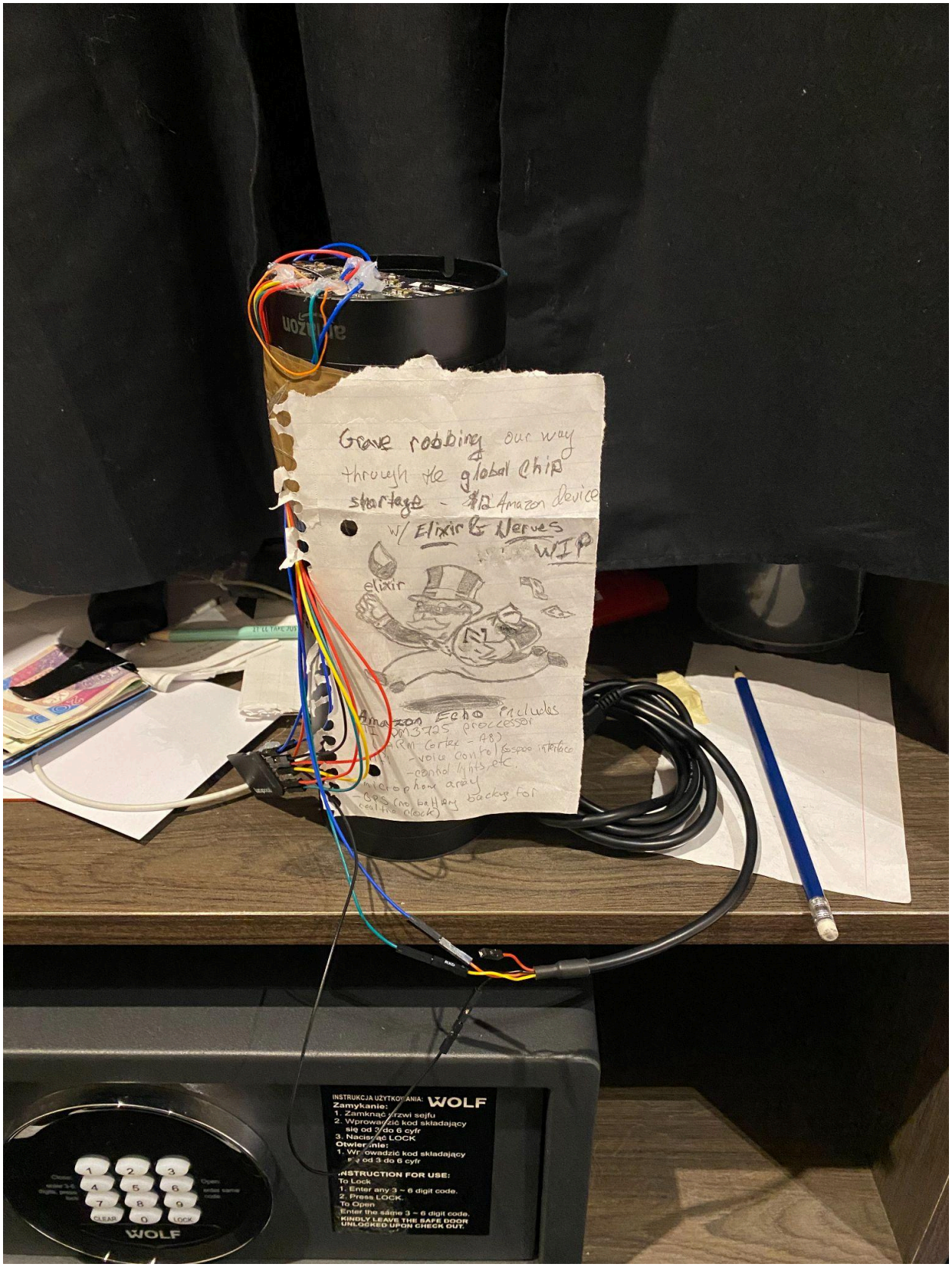
deb-src http://old-releases.ubuntu.com/ubuntu/ lucid universe
deb http://old-releases.ubuntu.com/ubuntu/ lucid-updates universe
deb-src http://old-releases.ubuntu.com/ubuntu/ lucid-updates universe

N.B. software from this repository is ENTIRELY UNSUPPORTED by the Ubuntu
team, and may not be under a free licence. Please satisfy yourself as to
your rights to use the software. Also, please note that software in
multiverse WILL NOT receive any review or updates from the Ubuntu
security team.
deb http://old-releases.ubuntu.com/ubuntu/ lucid multiverse
deb-src http://old-releases.ubuntu.com/ubuntu/ lucid multiverse
deb http://old-releases.ubuntu.com/ubuntu/ lucid-updates multiverse
deb-src http://old-releases.ubuntu.com/ubuntu/ lucid-updates multiverse

Uncomment the following two lines to add software from the 'backports'
repository.
N.B. software from this repository may not have been tested as
extensively as that contained in the main release, although it includes
newer versions of some applications which may provide useful features.
Also, please note that software in backports WILL NOT receive any review
or updates from the Ubuntu security team.
deb http://old-releases.ubuntu.com/ubuntu/ lucid-backports main restricted universe multiverse
deb-src http://old-releases.ubuntu.com/ubuntu/ lucid-backports main restricted universe multiverse

Uncomment the following two lines to add software from Canonical's
'partner' repository.
This software is not part of Ubuntu, but is offered by Canonical and the
respective vendors as a service to Ubuntu users.
deb http://old-releases.ubuntu.com/ubuntu lucid partner
deb-src http://old-releases.ubuntu.com/ubuntu lucid partner

deb http://old-releases.ubuntu.com/ubuntu lucid-security main restricted
deb-src http://old-releases.ubuntu.com/ubuntu lucid-security main restricted
deb http://old-releases.ubuntu.com/ubuntu lucid-security universe
deb-src http://old-releases.ubuntu.com/ubuntu lucid-security universe
deb http://old-releases.ubuntu.com/ubuntu lucid-security multiverse
deb-src http://old-releases.ubuntu.com/ubuntu lucid-security multiverse


Grave robbing our way
through the global chip
shortage - #2 Amazon device
w/ Elixir & Nerves
WIP
elixir
Amazon Echo includes
AM3725 processor
ARM Cortex - A8)
-voice control/espoo interface
-control lights, etc.
microphone array
-OS (no battery backup for
real time clock)
WOLF
INSTRUKCJA UŻYTKOWANIA:
Zamykanie:
1. Zamknąć drzwi sejfu
2. Wprowadzić kod składający
się od 3 do 6 cyfr
3. Nacisnąć LOCK
Otwieranie:
1. Wprowadzić kod składający
się od 3 do 6 cyfr
INSTRUCTION FOR USE:
To Lock
1. Enter any 3 ~ 6 digit code.
2. Press LOCK.
To Open
Enter the same 3 ~ 6 digit code.
KINDLY LEAVE THE SAFE DOOR
UNLOCKED UPON CHECK OUT.
1 2 3
4 5 6
7 8 9
CLEAR 0 LOCK
WOLF