

EASY TAZ MINI DEVELOPER'S GUIDE



LulzBot Easy TAZ Mini Developer's Guide
by Aleph Objects, Inc.
Copyright © 2014 Aleph Objects, Inc.
Permission is granted to copy, distribute and/or modify this document
under the terms of the Creative Commons Attribution 4.0 International
Public License (CC BY-SA 4.0).
Published by Aleph Objects, Inc., 626 West 66th Street, Loveland, Colorado,
80538 USA.
For more information, call +1-970-377-1111 or visit www.alephobjects.com.

20140913

Contents

Introduction	ix
Welcome Aboard	...
Audience	x
Open Source Hardware, Free Software	x
1 LulzBot Easy TAZ Mini	11
Developer Overview	...
1.1 Easy TAZ Mini	12
1.2 Versions	12
1.3 Begonia Photos	12
1.4 Schedule	21
2 Mechanical	23
Cartesian Bot in X, Y, Z	...
2.1 Intro	24
2.2 Begonia Renders	24
2.3 Begonia 3D Printed Parts	32
2.4 Begonia Bed	32
2.5 Begonia Extruder	35
2.6 Begonia LCD	37
2.7 Begonia Spool	40
2.8 Begonia X	43
2.9 Begonia Y	45
2.10 Begonia Z	47
2.11 Begonia Misc	50
2.12 Begonia Drawings	54
2.13 Camillia Drawings	54
3 Electrical	55
Power Supply, wiring	...

iii

List of Figures

1.1 Begonia Front Photo	13
1.2 Begonia Left Photo	14
1.3 Begonia Back Photo	15
1.4 Begonia Right Photo	16
1.5 Begonia Spool Arm Up Photo	17
1.6 Begonia Spool Arm Down Photo	18
1.7 Begonia Green Color Scheme Photo	19
1.8 Begonia Black Green Color Scheme Photo	20
2.1 Begonia Front Render	25
2.2 Begonia ISO Render	26
2.3 Begonia Left Render	27
2.4 Begonia Right Render	28
2.5 Begonia Right Render	29
2.6 Begonia Top Render	30
2.7 Begonia Bottom Render	31
2.8 Begonia 3D Printed Bed Corner Render	33
2.9 Begonia 3D Printed Bed Cover Render	33
2.10 Begonia 3D Printed Bed Fan Mount Render	34
2.11 Begonia 3D Printed Belt Clamp Render	34
2.12 Begonia 3D Printed Extruder Body Hex Render	36
2.13 Begonia 3D Printed Extruder Mount Render	36
2.14 Begonia 3D Printed LCD Back Cover Render	38
2.15 Begonia 3D Printed LCD Catch Render	38
2.16 Begonia 3D Printed LCD Hinge Render	39
2.17 Begonia 3D Printed LCD Mount Render	39
2.18 Begonia 3D Printed Spool Arm Render	41
2.19 Begonia 3D Printed Spool Hinge Render	41
2.20 Begonia 3D Printed Spool Mount Render	42
2.21 Begonia 3D Printed X End Idler Render	44
2.22 Begonia 3D Printed X End Motor Render	44
2.23 Begonia 3D Printed Y End Idler Render	46
2.24 Begonia 3D Printed Y Rod Mount Render	46
2.25 Begonia 3D Printed Upper Z Left Render	48
5.1 Core (center) and Base Board Photo	61
5.2 Base Board Layout	63

CONTENTS

CONTENTS

3.1 Electrical Layout	56
4 3D Printer Controller	57
Mini-RAMBo	58
4.1 Intro	58
5 Embedded Hardware	59
Freedom Sandwich	59
5.1 Overview	60
5.2 Specifications	62
5.3 Other Hardware	64
LCD	64
LED Lights	64
Camera	64
6 Free Software	65
Debian, Linux, GNU, Slic3r, et. al.	65
6.1 Intro	66
6.2 Bootloader	66
Intro	66
Git Repos	66
Commands	66
6.3 Linux Kernel	67
Intro	67
Kernel Branch	67
Kernel Version	68
Building the Kernel	68
6.4 Core OS	70
Creating Root Filesystem	70
Packages	71
Changes to Core Packages	71
Filesystem	72
Developer Host System Setup	72
OctoPrint	74
6.5 3D Object Processing	77
6.6 Misc	77
7 Contact	79
Phone, Email, Web, Location	79
7.1 Support	80
7.2 Sales	80
7.3 Websites	80

iv

v

vi

vii

Introduction

Welcome Aboard

Audience

This is a developer's guide to hacking on the LulzBot Easy TAZ Mini 3D Printer. It is meant for developers, not users, of the printer.

Open Source Hardware, Free Software

Aleph Objects, Inc. is a Loveland, Colorado, USA company that manufactures Open Source Hardware using Free Software.
For more info, visit <http://www.alephobjects.com>.

LulzBot Easy TAZ Mini

Developer Overview

LulzBot Easy TAZ Mini

1.3. BEGONIA PHOTOS

13

14

LulzBot Easy TAZ Mini

1.3. BEGONIA PHOTOS

15

1.1 Easy TAZ Mini

The LulzBot Easy TAZ Mini is a 3D Printer currently under development. The abbreviated name is EZTAZ.

The source files are available here:
http://devel.lulzbot.com/Easy_TAZ_Mini/

1.2 Versions

Each new version of the EZTAZ has a new name, with the next letter in the alphabet.

- Azalea - First Prototype
- Begonia - Second Prototype, being built now
- Camellia - Third Prototype
- Daffodil - First Production batch

1.3 Begonia Photos

Figure 1.1: Begonia Front Photo

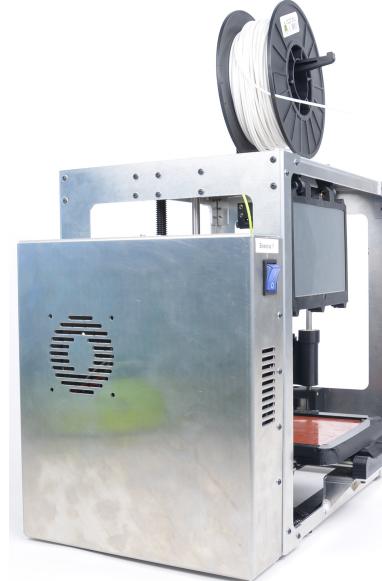


Figure 1.2: Begonia Left Photo

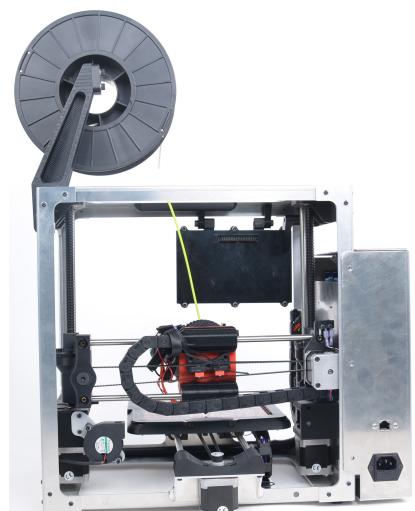


Figure 1.3: Begonia Back Photo



Figure 1.4: Begonia Right Photo



Figure 1.5: Begonia Spool Arm Up Photo



Figure 1.6: Begonia Spool Arm Down Photo



Figure 1.7: Begonia Green Color Scheme Photo

1.4 Schedule

The schedule is updated weekly. It is in Libre Office spreadsheet format.
The latest version is available here:

http://devel.lulzbot.com/Easy_TAZ_Mini/program_management/

Mechanical
Cartesian Bot in X, Y, Z

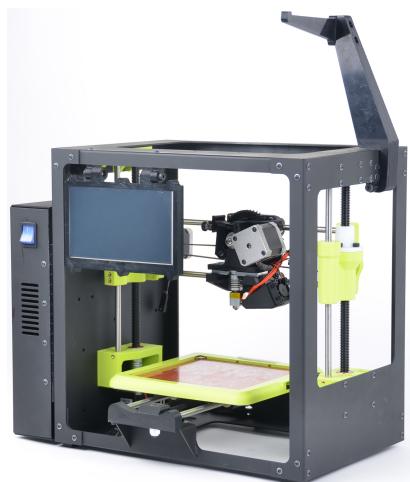


Figure 1.8: Begonia Black Green Color Scheme Photo

2.1 Intro

Mechanical hardware specs and parts are in these subdirectories:
http://devel.lulzbot.com/Easy_TAZ_Mini/

2.2 Begonia Renders

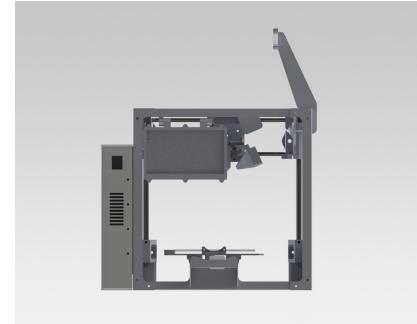


Figure 2.1: Begonia Front Render



Figure 2.2: Begonia ISO Render

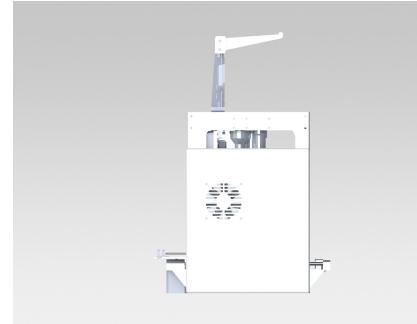


Figure 2.3: Begonia Left Render

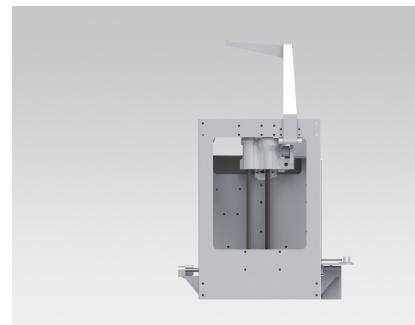


Figure 2.4: Begonia Right Render

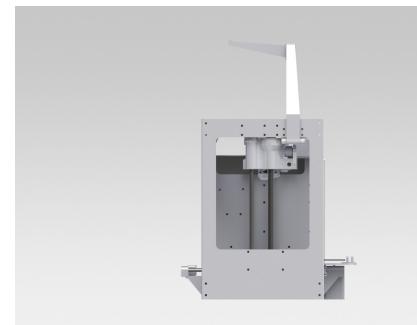


Figure 2.5: Begonia Right Render

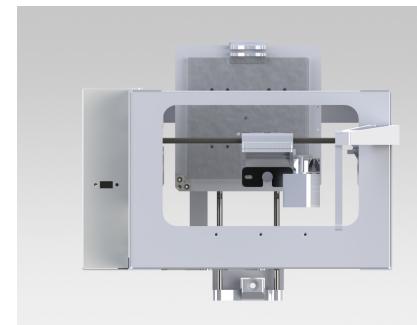


Figure 2.6: Begonia Top Render

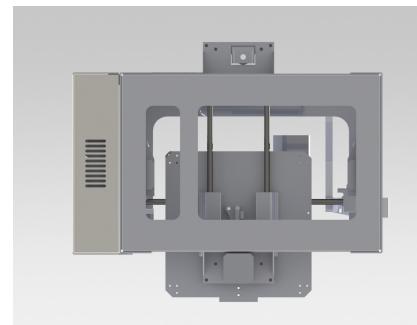


Figure 2.7: Begonia Bottom Render

2.3 Begonia 3D Printed Parts

2.4 Begonia Bed

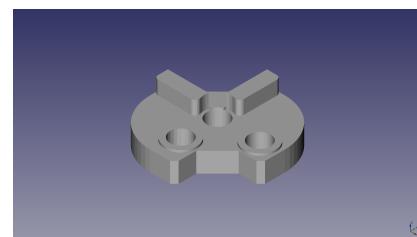


Figure 2.8: Begonia 3D Printed Bed Corner Render

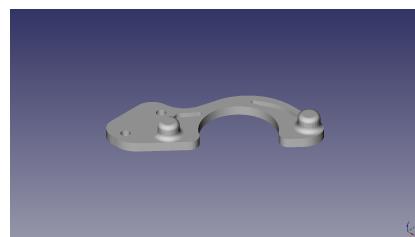


Figure 2.10: Begonia 3D Printed Bed Fan Mount Render

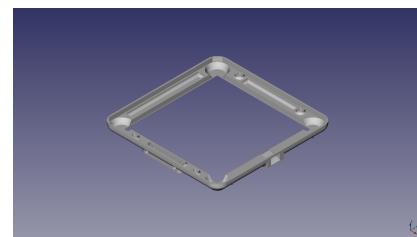


Figure 2.9: Begonia 3D Printed Bed Cover Render

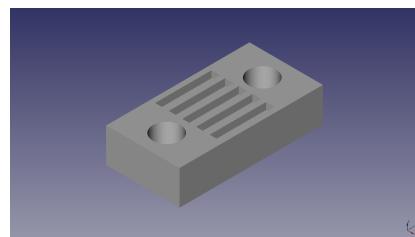


Figure 2.11: Begonia 3D Printed Belt Clamp Render

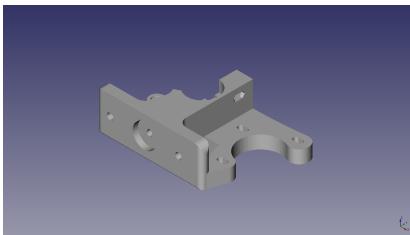


Figure 2.12: Begonia 3D Printed Extruder Body Hex Render

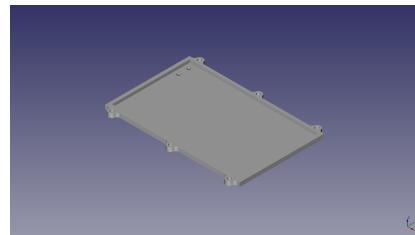


Figure 2.14: Begonia 3D Printed LCD Back Cover Render



Figure 2.13: Begonia 3D Printed Extruder Mount Render

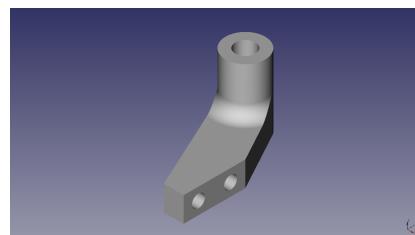


Figure 2.15: Begonia 3D Printed LCD Catch Render

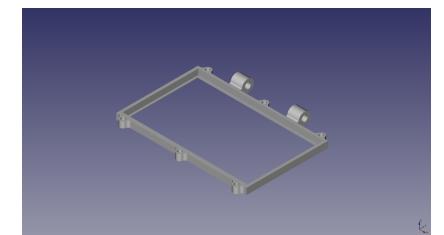


Figure 2.17: Begonia 3D Printed LCD Mount Render

2.7 Begonia Spool



Figure 2.18: Begonia 3D Printed Spool Arm Render

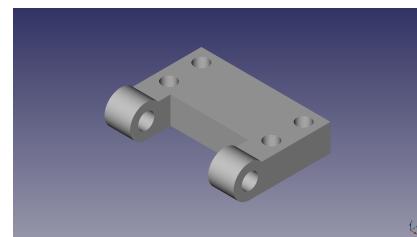


Figure 2.19: Begonia 3D Printed Spool Hinge Render

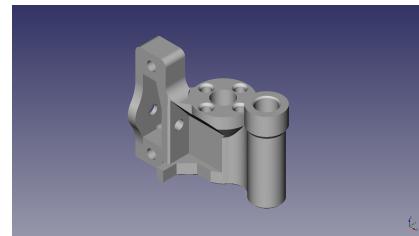


Figure 2.21: Begonia 3D Printed X End Idler Render

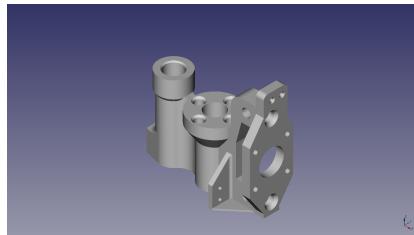


Figure 2.22: Begonia 3D Printed X End Motor Render

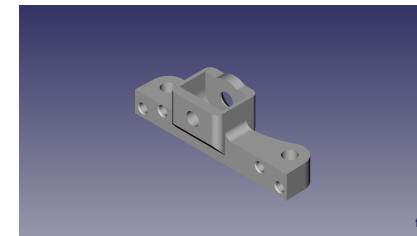


Figure 2.23: Begonia 3D Printed Y End Idler Render

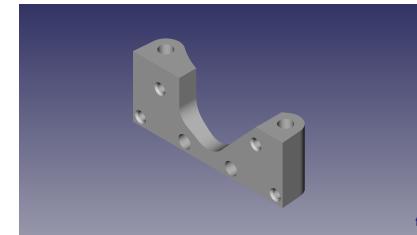


Figure 2.24: Begonia 3D Printed Y Rod Mount Render

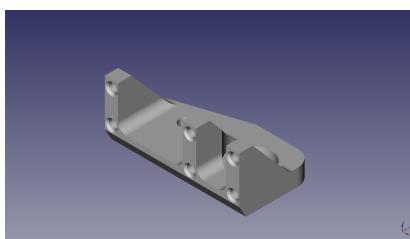


Figure 2.25: Begonia 3D Printed Upper Z Left Render

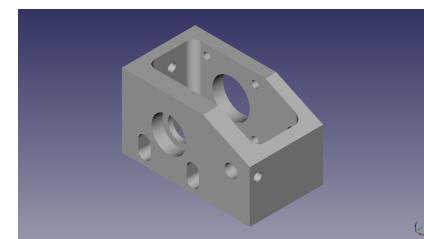


Figure 2.27: Begonia 3D Printed Lower Z Left Render

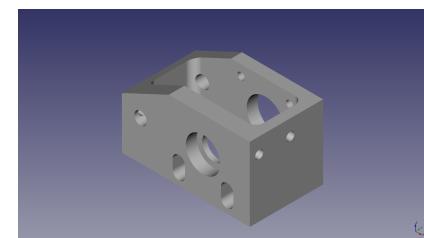


Figure 2.28: Begonia 3D Printed Lower Z Right Render

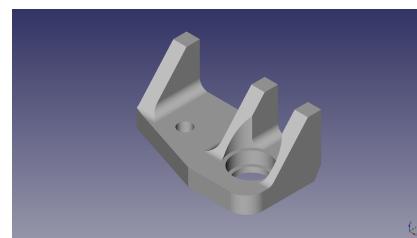


Figure 2.26: Begonia 3D Printed Upper Z Right Render



Figure 2.29: Begonia 3D Printed Double Bearing Holder Render

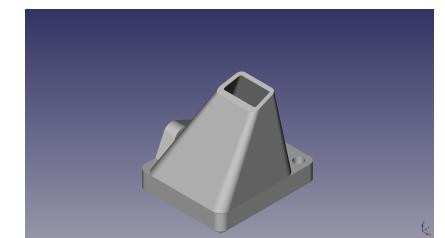


Figure 2.30: Begonia 3D Printed Fan Mount Render

2.12 Begonia Drawings

2.13 Camillia Drawings

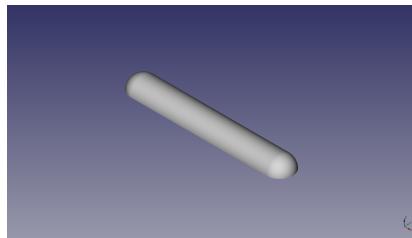


Figure 2.31: Begonia 3D Printed Handle Bar Render

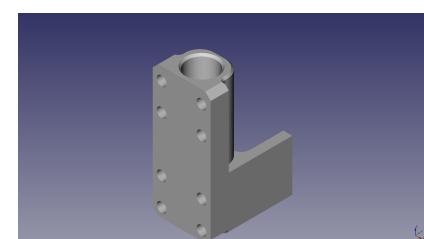


Figure 2.33: Begonia 3D Printed Extruder Mt Top Double Bearing Holder Render

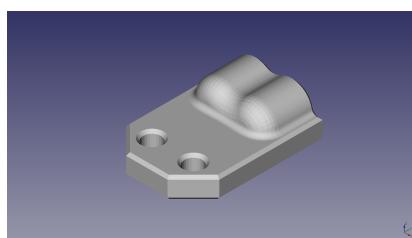


Figure 2.32: Begonia 3D Printed Cable Carrier Mount Render

Electrical
Power Supply, wiring

3.1 Electrical Layout

3D Printer Controller
Mini-RAMBo

56

Embedded Hardware

5.1 Overview

The final hardware will be a custom built board by LinkSprite (makers of the PCDuino). The system will require two PCBs. One will be the (soon-to-be) standard LinkSprite Core Board. The other board is a daughter board to the core board, and is a custom designed Base Board, populated with what we need for the printer. This combo system we are calling the EZGNU. The whole thing with core board plus base board plus LCD is the Freedom Sandwich, part of the Free Lunch series.

The first two revisions, Azalea and Begonia, use the Olimex A20 board. Testing for the kernel is done on the LinkSprite PCDuino Version 3.

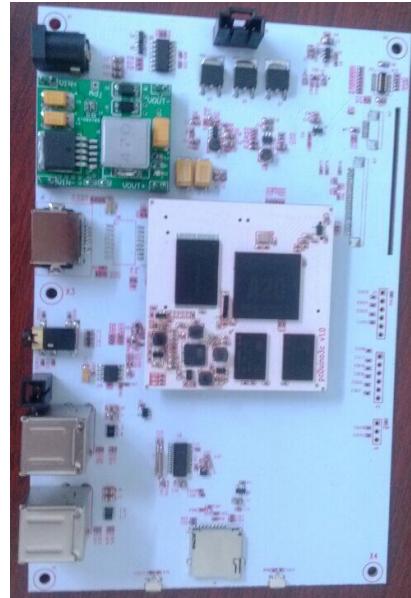


Figure 5.1: Core (center) and Base Board Photo

60

4.1 Intro

The printer controller will be the RAMBo-Mini.

Embedded Hardware
Freedom Sandwich

58

Embedded Hardware

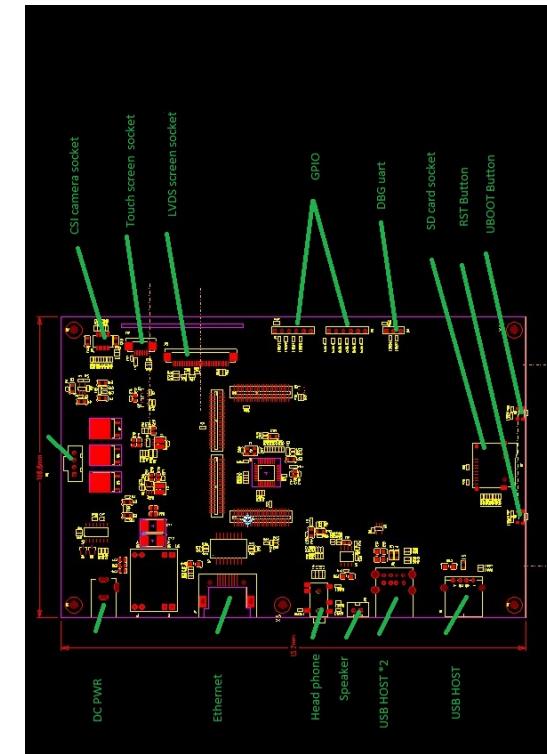
5.2 Specifications

Specs, Core Board:

- Allwinner A20 ARM Processor
- 1GHz CPU
- 1 Gig RAM

Specs, Base Board:

- MicroSD Card Slot
- 4 USB A Ports
- Line level audio in/out
- Amplified audio out (mono speaker headers)
- Ethernet
- MIPI/CSI Camera socket
- 10 GPIO pins
- 24V power input
- 12V RGB LED Strip driver
- uBoot button
- RST button
- DBG UART Pins
- LVDS LCD Socket
- I2C Touch screen socket



62

5.3 Other Hardware

LCD

The LCD screen will mount near the EZGNU. The board is being specified by Linkspire.

Specs LCD:

- 1280x800 Resolution
- LVDS interface

LED Lights

There will be RGB LED strips to indicate various printer states via colored lights.

Camera

The first version won't have a camera, though the EZGNU does have a header for common types.

Free Software

Debian, Linux, GNU, Slic3r, et. al.

- Linux Sunxi – This is the main kernel branch for the Sunxi platform built upon the Allwinner ARM chips. It is actively maintained. The Easy TAZ Mini core is built upon their various archives. Main website:

<http://linux-sunxi.org/>
<git://github.com/linux-sunxi/linux-sunxi.git>

Kernel Version

We will be using the main linux-sunxi git repo, using the sunxi-3.4 branch as the main base for the Linux kernel. The latest version is 3.4.90.

There are also various kernel versions we could choose from. Some options:

- sunxi-next – This looks pretty good. Worth exploring more.
- sunxi-devel – Probably not.
- 3.4.79+ – This is the kernel that gets built by the default a20-kernel archive from Linkspire. Known to work. Has non-free software.
- 3.14 – This is the latest version from the sunxi-3.14 branch of the main linux-sunxi kernel. It has not seen as much development or testing as sunxi-3.4. It does have -sunxi patches and is based on a much more recent upstream kernel. The one test kernel I built didn't fully boot, but it likely can be made to work without too much pain. As it hasn't seen as much real-world usage, sunxi-3.4 is preferred.
- 3.4.90 – This is the latest version from the sunxi-3.4 branch of the main linux-sunxi kernel. This is known to work. We will likely use a version of this kernel.

Building the Kernel

Quickie overview:

- Clone the kernel archive we want to use:

```
git clone git://github.com/linux-sunxi/linux-sunxi.git
cd linux-sunxi
```

Misc build commands

- This will build the default sun7i (pduino3) kernel (example):

6.1 Intro

This chapter covers the software that runs on the EZGNU embedded hardware board.

6.2 Bootloader

Intro

U-boot is the bootloader. The linux-sunxi branch is used as a base.

Git Repos

Upstream git repos used:

- <git://github.com/linux-sunxi/u-boot-sunxi.git>
- <git://github.com/linux-sunxi/tools.git>
- <git://github.com/linux-sunxi/sunxi-boards.git>
- <git://github.com/linux-sunxi/sunxi-bsp.git>
- We're not using it at present, but this is allegedly the best version of A20 u-boot at present, per
<https://wiki.debian.org/InstallingDebianOn/Allwinner>
<git://github.com/jvrdegoede/u-boot-sunxi.git>

Commands

Various commands for reference.

- ```
./sunxi-tools/fex2bin \
OUTPUT/ao-pcduino3.fex OUTPUT/script.bin
```
- ```
mkimage -C none -A arm -T script -d \
OUTPUT/boot.cmd OUTPUT/boot/boot.scr
```

```
LOADADDR=0x40008000 make -j4 ARCH=arm
CROSS_COMPILE=arm-linux-gnueabihf-
sun7i_defconfig
```

- If you need to clean up:

```
make clean ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
```

- You probably don't need to:

```
make mrproper ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
```

6.4 Core OS

- Debian Wheezy (stable) release
- armhf architecture

Creating Root Filesystem

The root filesystem is Debian Wheezy using the armhf architecture. Commands, briefly, to build a rootfs:

1. mkdir rootfs
2. sudo debootstrap --verbose --arch=armhf --foreign wheezy \
./rootfs
3. sudo cp -p /usr/bin/qemu-arm-static \
./rootfs/usr/bin/
4. sudo chroot ./rootfs
5. /debootstrap/debootstrap --second-stage

- sudo dd if=u-boot-sunxi-with-spl.bin of=/dev/sdb bs=1024 seek=8

- Boot splash, to boot.cmd:

```
$(fs)load ${dtype}:1 10000000 /splash.bmp && bmp d 10000000;
```

- make CROSS_COMPILE=arm-linux-gnueabihf- \
Linksprite_pcDuino3_config

- make CROSS_COMPILE=arm-linux-gnueabihf-

6.3 Linux Kernel

Intro

The EZGNU board uses the Linux kernel.

Kernel Branch

We will be using Linus Sunxi as the base source code for the kernel. There are various kernel branches that could be used as a base:

- Mainline Linus – This doesn't have much of the latest/greatest for A20.
<git://kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git>

- Linkspire – This is optimized for pcduino3, which is very similar to the board we'll be using. It uses non-free software, such as the Mali driver. Using it as a base will be a bit messy. It also does non-standard patching in the build process, instead of just committing everything to the kernel. The build system overall is nice and does more than just the kernel. It depends upon the non-free Allwinner fslut image tools. We will use another tree as a base, but will pick select drivers from this one.
<git://github.com/pcduino/a20-kernel.git>

Packages

The core packages will be straight from Debian's armhf archive, including wheezy-backports. Exceptions:

- u-boot bootloader - Not maintained in a package (at present).
- Linux Kernel - Not maintained in a package (at present).
- Slic3r - This has been (?) built for Debian Sid and Jessie.
- Slic3r dependencies - This is a long list. They have been built for Debian Sid and Jessie.
- OctoPrint - This has not been packaged yet.
- EZTAZ Applications - Need to be packaged.

The development repository will be located here:

<http://devel.lulzbot.com/debian/>

The release repository will be located here:

<http://download.lulzbot.com/debian/>

So the development repo line to add to `/etc/apt/sources.list` for Azalea is:

```
deb http://devel.lulzbot.com/debian/ azalea main
```

It will need to be populated with a sub-set of the main Debian archive. We won't carry the whole repo, to make it more efficient. The main Debian repo should remain compatible with the EZTAZ repo.

Changes to Core Packages

The following scripts/configurations to the standard Debian release:

- systemd – This should likely be used to improve boot time.
- read-only filesystem – So the SD card won't corrupt itself, nor the user wait for fscking.

Filesystem

The best filesystem for the SD card needs to be selected... Considerations:

- Robustness
 - Boot Speed
- Filesystem options for core OS:
- ext4
 - btrfs
 - squashfs
 - NILFS2

Filesystem options for users' USB drives:

- ext4
- btrfs
- FAT/VFAT – We won't use these in the core OS, but we will have to read them off users' USB drives.
- NTFS – Same as FAT? Probably don't need/want it at all.
- HFS+ – Same as NTFS?

Developer Host System Setup

- Debian Wheezy
- emdebian repo – cross compilers
- Fix compiler paths – cross compilers, use ccache

Cross Compiler

To compile ARM packages, a cross compiler must be set up.

1. Add emdebian Repo – Add this line to `/etc/apt/sources.list`:
`deb http://www.emdebian.org/debian/ unstable main`
2. Update –
`apt-get update`
3. Install key –
`sudo apt-get install emdebian-archive-keyring`
4. Install Cross Compilers –
`apt-get install \
 cpp-4.7-arm-linux-gnueabihf \
 g++-4.7-arm-linux-gnueabihf \
 gcc-4.7-arm-linux-gnueabihf \
 gcc-4.7-arm-linux-gnueabihf-base \
 gcc-4.7-plugin-dev-arm-linux-gnueabihf`

5. Install (not all needed? Note, some in backports):

```
libncurses-dev build-essential
ccache git autoconf autoconf2.13
gnu-standards libtool u-boot-tools
debootstrap qemu qemu-user-static
```

6. Update everything to latest in wheezy-backports:
`apt-get -t wheezy-backports dist-upgrade`
`apt-get install ccache git`

7. Add this to `~/.bashrc` –
`export PATH="/usr/lib/ccache:$PATH"`

8. Set up symlinks without version numbers (super crusty)

```
cd /usr/lib/ccache/
sudo ln -s ../../bin/ccache arm-linux-gnueabihf-g++
sudo ln -s ../../bin/ccache arm-linux-gnueabihf-gcc
cd /usr/bin/
sudo ln -s arm-linux-gnueabihf-cpp-4.7 arm-linux-gnueabihf-cpp
sudo ln -s arm-linux-gnueabihf-g++-4.7 arm-linux-gnueabihf-g++
sudo ln -s arm-linux-gnueabihf-gcc-4.7 arm-linux-gnueabihf-gcc
sudo ln -s arm-linux-gnueabihf-gcc-ar4.7 arm-linux-gnueabihf-gcc-ar
sudo ln -s arm-linux-gnueabihf-gcc-nm-4.7 arm-linux-gnueabihf-gcc-nm
sudo ln -s arm-linux-gnueabihf-gcov-4.7 arm-linux-gnueabihf-gcov
sudo ln -s arm-linux-gnueabihf-gcov-4.7 arm-linux-gnueabihf-gcov
```

fb

- X Windows
- EZTAZ
- OctoPrint Web Interface

OctoPrint

OctoPrint main site: <http://www.octoprint.org>

OctoPrint uses Python 2.7. Until OctoPrint is packaged in Debian, it will be built on a host system and installed to `/usr/local`.

- adduser octo
- su - octo
- git clone git://github.com/foosel/OctoPrint.git
- cd OctoPrint
- git checkout 1.2.0-dev
- This may be a good one to consider:
`git checkout remotes/origin/devlevel`

- See what dependencies there are:
`cat requirements.txt`

```
apt-get install -t wheezy-backports python-jinja2 python-six \
python-pymgments python-docutils
```

- As root, install unpackaged dependencies:
`pip install -r requirements.txt`

- Install to `/usr/local`:

```
python setup.py install
```

- Set up `/etc/default/octoprint`

```
OCTOPRINT_USER=octo
PORT=8080
DAEMON=/usr/local/bin/octoprint
DAEMON_ARGS="--port=$PORT"
UMASK=022
START=yes
```

- Set up `/etc/init.d/octoprint`

```
cp scripts/octoprint.init /etc/init.d/octoprint
chmod 755 /etc/init.d/octoprint
update-rc.d octoprint defaults
```

OctoPrint 1.2.0-dev dependencies:

- flask==0.9 – Wheezy has python-flask 0.8
- werkzeug==0.8.3 – Wheezy has python-werkzeug 0.8.3+dfsg-1
- tornado==3.0.2 – Wheezy has python-tornado 2.3
- socksjs-tornado>=1.0
- PyYAML==3.10 – Wheezy has python-yaml 3.10

- Flask-Login==0.2.2
- Flask-Principal==0.3.5
- pyserial>=2.6 – Wheezy has python-serial 2.5
- netaddr>=0.7.10 – Wheezy has python-netaddr 0.7.7
- mock>=1.0.1 – Wheezy has python-mock 0.8.0
- nose>=1.3.0 – Wheezy has python-nose 1.1.2
- sphinxcontrib-httplibdomain
- sphinx_rtd_theme

Most python modules for OctoPrint 1.2.0-dev are in Debian, but they are older versions. There may be newer versions that can be rebuilt from Jessie and Sid.

These are in Wheezy:
`apt-get install python2.7 python-werkzeug python-yaml python-pip`
These are pulled in by the dependencies of pip requirements.txt, that are in Wheezy, but aren't explicitly named as requirements.

Set up the octo user with our OctoPrint config, stored at `/home/octo/.octoprint/config.yaml`
`chown octo:octo /home/octo/.octoprint/config.yaml`
Change the key as needed.

```
accessControl:
  enabled: false
api:
  enabled: true
  key: 19A7C56E31B74257955E49E5561D019D
appearance:
  color: yellow
  name: Easy TAZ Mini
cura: {}
feature:
  gCodeVisualizer: false
  sdSupport: false
```

Areas to fix to speed up boot:

- hotplug
- /dev populated statically
- /tmp is tmpfs, so no need to "clean" it

- Slic3r
- Meshlab

Adding "noswap" to the kernel boot command line should make it skip trying to activate swap, but it doesn't in all cases. This hack:
`echo "NOSWAP=yes" >> /etc/default/rcS`

Contact

Phone, Email, Web, Location

7.1 Support

Email: support@alephobjects.com
Phone: +1-970-377-1111 x610
LulzBot Forum
<http://forum.lulzbot.com>

7.2 Sales

Email: sales@alephobjects.com
Phone: +1-970-377-1111 x600

7.3 Websites

Aleph Objects, Inc.
<http://www.alephobjects.com>
LulzBot 3D Printers
<http://www.lulzbot.com>

Colophon

Created with 100% Free Software
GNU/Linux
 \LaTeX Memoir
