Lukas DiBeneditto, Mechanical Engineering Technology, Undergraduate
Paul Hagmann, Laboratory Manager
Christy Snip, Student Services Coordinator
MET 392 Laboratory Assistant In Engineering Technology
Purdue University, Purdue Polytechnic New Albany
13 Apr. 2020

3D Printer Cartridge Research Project Closure Document

Reverse Engineering a Stratasys Dimension SST 768 Fused Deposition Modeling (FDM) 3D Printer Filament Cartridge

Executive Summary

For this independent study research project, the Advanced Manufacturing Lab Manager requested that I find and implement a solution to save the school money by making non-refillable 3D printer cartridges, refillable. This document details the process of reverse-engineering the 3D printer cartridge EEPROM chip and the demonstration of a working reproducible prototype solution. This project required electrical engineering, mechanical engineering, computer programming, and Linux experience. The result is that more students now have access to a high-end commercial-grade 3D printer to add value to their education and cost savings of 86% for the school for each 3D printer cartridge.

Fig. 1. Stratasys Dimension SST 768 3D Printer [MIT]

Background and Context

The school has a high-end commercial grade Stratasys Dimension SST 768 Fused Deposition Modeling (FDM) 3D Printer which when released in 2013 retailed for $40,000. [3DPC] To use this expensive piece of equipment it requires consumable 3D printer ABS filament cartridges that cost ~$200 each. [AET] With about 250 students

conceivably each printing multiple 3D prints for multiple projects each semester purchasing enough 3D printer filament would quickly exhaust the budget. This is probably why the school purchased multiple LulzBot TAZ 5 3D printers (retailing for ~$2,200 each) for the students to use, which consume 3D printer ABS filament spools that cost ~$28 each. With education volume pricing it may be possible to purchase the filament spools at a lower price.

Project Challenges

The challenge was seeing if it would be possible to use the $28 LulzBot filament in place of the $200 Stratasys filament. Luckily previous research showed that others have done this. [ERESET] [HAVEBLUE] [PCB] [STEETO] This was complicated by three main obstacles to overcome:

1. **Roll vs. Cartridge:** The $28 LulzBot filament comes on rolls (Fig. 1), the $200 Stratasys filament comes in cartridges (Fig.2). The cartridges are proprietary and are designed to not be refilled.
2. **EEPROM Serial:** The $200 Stratasys filament cartridges contain an embedded electronics chip (EEPROM) with a unique serial number that the Stratasys Dimension SST 768 3D printer records, compares the amount of filament used, and marks the cartridge as empty. Even if the cartridge was opened and the filament replaced, the 3D printer would not print with the new filament because the cartridge has been recorded "marked as empty". It is assumed that Stratasys did this in an attempt to vendor lock-in path dependency for its customers. [LOCKIN]
3. **EEPROM Encryption:** The embedded electronics chips are encrypted to actively prevent modification to their firmware and resetting the filament count.

Fig. 2. LulzBot Filament Rolls [LB]

Fig. 3. Stratasys Filament Cartridges [AET]

Solution and Addressing the Challenges

Successful resetting of the $200 Stratasys filament cartridges EEPROM was accomplished by building a system consisting of a Raspberry Pi 3 B+, 1.5 KΩ Resistor, Alligator Clips, some wires, and use of purpose-built software, with an estimated budget ~$40. The basic process idea is to remove the Stratasys chip from the cartridge, connect to the alligator clips, run some commands in the Linux Terminal and this reads, make edits, overwrite the EEPROM, assigning a new serial number and changing the amount of filament that the Stratasys 3D Printer has stored.

1. **Roll vs. Cartridge:** The scope of the project was constrained to provide a working prototype and process. Rolling the filament onto the roll inside of the cartridge was sidelined for a future project per the Lab Manager.
2. **EEPROM Serial:** This required building a prototype electronics system and learning how to use the software, which was a very difficult task considering the instructions were from 2013 and 2016 and the software has since been updated to a new version (2019) and the instructions match very little.
3. **EEPROM Encryption:** The software "stratatool" manages all the decryption, editing, and encryption of the chip.

Hardware Required

1. Raspberry Pi 3 B+ (B Plus)
2. 16 GB SanDisk Micro SD Card (Class 10)
3. 2.5A USB Power Supply with Micro USB Cable and Noise Filter
4. Raspberry Pi 3 B+ Case
5. High Quality HDMI Cable
6. HDMI Monitor
7. USB Keyboard
8. USB Mouse
9. Prototyping Board
10. Prototyping Jumper Wires
11. 1 Black Alligator Clip
12. 1 Red Alligator Clip
13. 1.5 KΩ Resistor
14. 1 Stratasys 3D Printer Cartridge Chip

Software Required

Raspberry Operating System:
NOOBS v3.2.1 [NOOBS]

Programming Language:
Python v2.7

python-crypto:
pycrypto v2.6.1 [PYCR]

EEPROM Flashrom Read/Write:
Stratasys EEPROM tool [STEETO]

Work Breakdown Structure

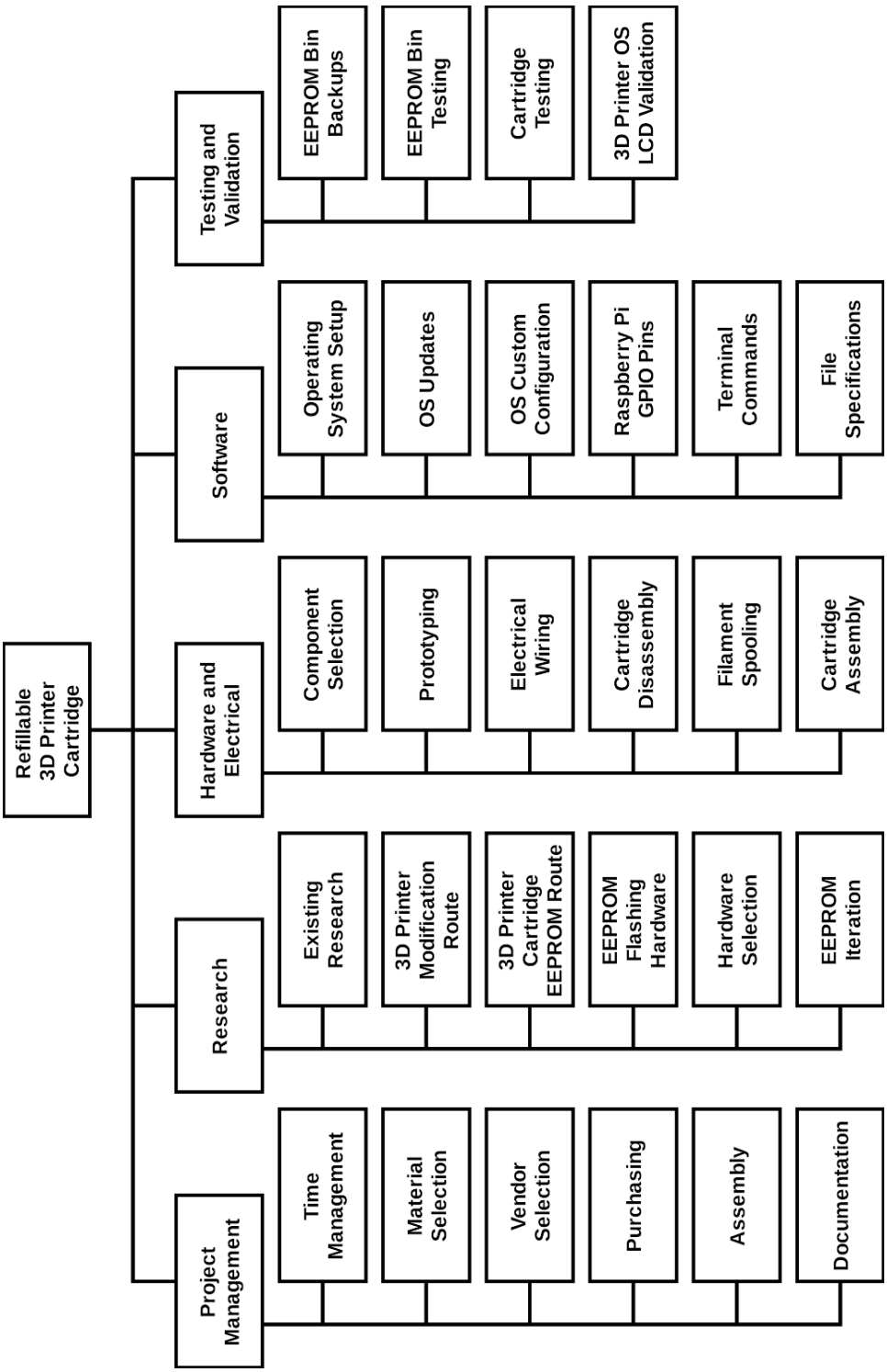


Fig. 4. Work Breakdown Structure

Procedure and Steps

The majority of the commands will be typed in the Linux Terminal. Here is a link to a refresher on how to use the Linux Terminal just in case.

Terminal - Raspberry Pi Documentation
https://www.raspberrypi.org/documentation/usage/terminal/

The Stratasys 3D Printer Cartridge Chip will be referred to as the Chip, the data stored on the Chip is a flashrom binary referred to as the Binary, Raspberry Pi 3+ will be referred to as the Computer, NOOBS operating system will be referred to as the Linux, and the Terminal emulator running on Linux will be referred to as the Terminal.

NOTE:

1. **It is highly recommended that you make a backup copy of the original EEPROM flash binary. Just in case something happens you can always write the original back to the Chip.**
2. Once you put a modified Chip into the Stratasys 3D printer it will record that serial "serial-number" in it's database, so you can't just change the "current-material" amount to a higher amount. A lower "current-material" amount would work, but not a higher amount. You will need to generate a new unique serial number for that Chip using the Raspberry Pi again to give the modified Chip a higher amount.
3. The maximum amount for "initial-material" and "current-material" is 53.6.
4. The "material-name" must match a valid Stratasys material type: "P400SR" works, and it is recommended to collect the "material-name" of the original cartridges to find one that can be used for the melting temperature of the material that replaces the original material spool. For example, does P400SR material have the same melting point as ABS or similar thermal properties?
5. The Terminal commands could be automated with a Bash or Python script however that was beyond the scope of this project but could be extended for a future project for another student per the Lab Manager.
6. Long lines for commands to be typed in Terminal have "\" a backslash which continues the command to the next line.

Summary of Steps

This represents all the commands that you need to run to change or reset the Chip, a more detailed step by step is listed on the following pages. This assumes that you have wired the chip correctly, see STEP 3.

Run Once

Assuming a new Raspberry Pi computer, fresh operating system install, or these commands have never run on the Raspberry Pi. This updates, then upgrades the Raspberry Pi, installs the python crypto, downloads the startatools program, unzips it, builds it, and installs it to get the dependencies, then prompts the user to configure the Raspberry Pi, and restarts.

I suggest running these commands line by line instead of copy and pasting the whole thing.

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install python-crypto
cd ~
wget https://github.com/bvanheu/stratatools/archive/master.zip
unzip master.zip
cd stratatools-master
sudo python2 ./setup.py build
sudo python2 ./setup.py install
sudo raspi-config
sudo shutdown -r
```

Run Every Time the Raspberry Pi has been Restarted

This changes the general-purpose input/output (GPIO) Raspberry Pi pin 7 to output then sets it to high, then it modifies the Linux Kernel to use the 1-wire protocol for the Broadcom BCM2837B0, then we look for the actual Linux device, output in postscript continuous hexdump of the chip id to use in the decoding, copy the EEPROM binary, decode the EEPROM, display the decoded EEPROM, create a new EEPROM from a template, encode the EEPROM to a binary, overwrite the old EEPROM with the new EEPROM binary, then display the new EEPROM data.

I suggest running these commands line by line instead of copy and pasting the whole thing, because you will need to make changes to these commands based on your unique chip. I have highlighted the important text that needs to change every time for every new Chip.

```
gpio mode 7 out
gpio write 7 1
sudo modprobe w1-gpio giopin=4
sudo modprobe w1-ds2433
cd /sys/bus/w1/devices/
ls
cd 23-0000015834c2
ls
xxd -p id
cp eeprom ~/original23-0000015834c2.bin
stratatools eeprom_decode --machine-type prodigy \
    --eeprom-uid 23c23458010000ad \
    eeprom > ~/orginal23-0000015834c2-info.txt
cat ~/orginal23-0000015834c2-info.txt
stratatools eeprom_create --serial-number 185372110.0 \
    --material-name P400SR --manufacturing-lot 5108 \
    --manufacturing-date "2001-01-01 01:01:01" \
    --use-date "2002-02-02 02:02:02" \
    --initial-material 53.6 --current-material 53.6 \
    --key-fragment 55aa5537ece659be --version 1 \
    --signature STRATASYS > ~/cartridge.txt
stratatools eeprom_encode --machine-type prodigy \
    --eeprom-uid 23c23458010000ad ~/cartridge.txt ~/cartridge.bin
sudo cp ~/cartridge.bin eeprom
stratatools eeprom_decode --machine-type prodigy \
    --eeprom-uid 23c23458010000ad eeprom
```

STEP 1. Required Cryptography Software: Install the required software and dependencies. Type these commands into the Terminal:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install python-crypto
```

If anything prompts you to upgrade choose yes. Now we need to get the software to decrypt the Chip.

```
cd ~
wget https://github.com/bvanheu/stratatools/archive/master.zip
```

After the archive file has downloaded we need to unzip it.

```
unzip master.zip
```

This should create a directory called "stratatools-master". Go into that directory.

```
cd stratatools-master
```

Now we need to build from source.

```
sudo python2 ./setup.py build
```

Now we need to install the build.

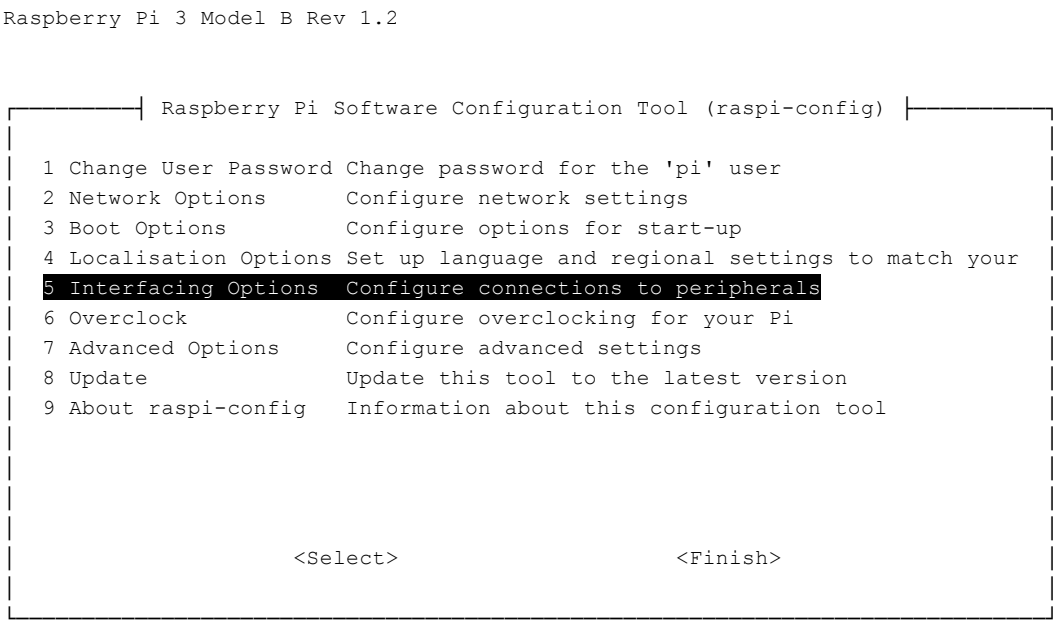
```
sudo python2 ./setup.py install
```

Once the installation is finished then go to STEP 2.

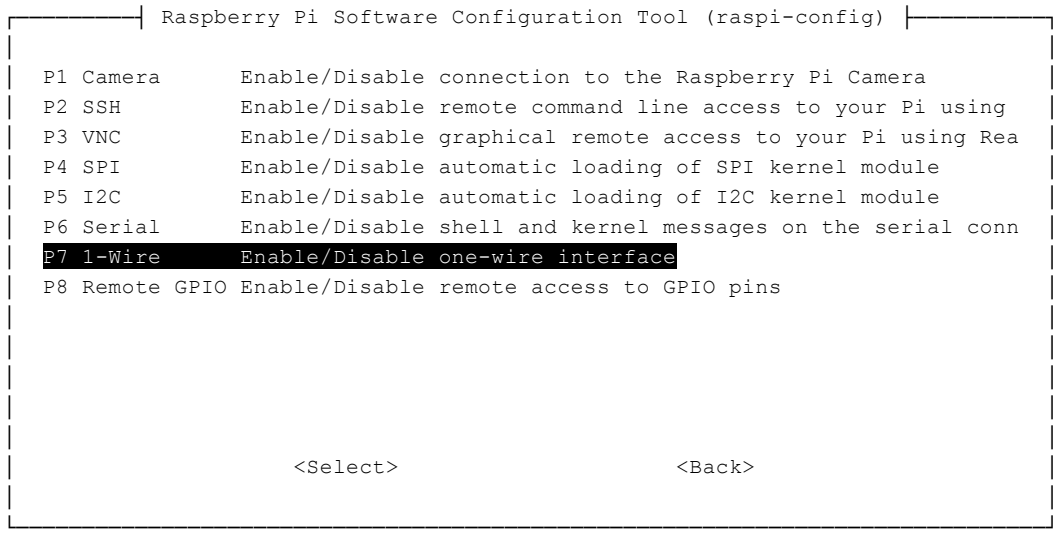
STEP 2. Setup the Raspberry Pi is configured to communicate using the 1-wire protocol. Type this command into the Terminal.

```
sudo raspi-config
```

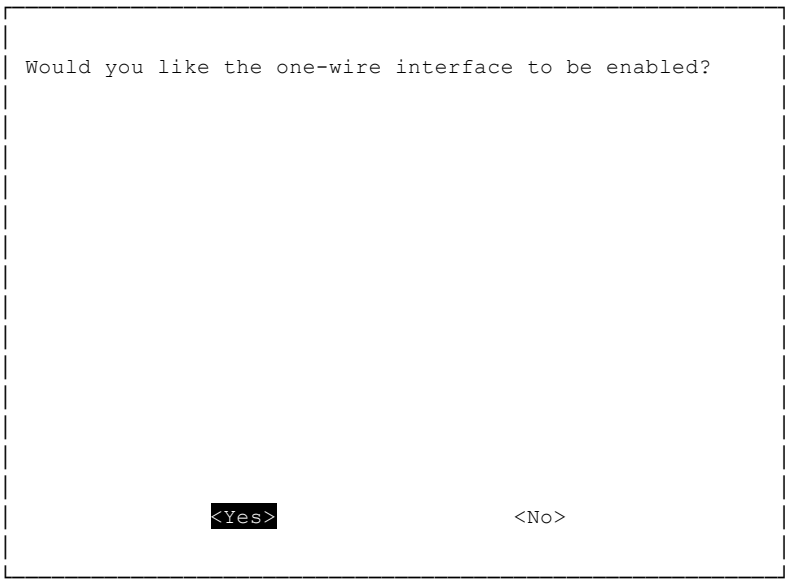
You should see a screen like this below, for the Raspberry Pi configuration program. Use the arrow buttons on your keyboard to select "5 Interfacing Options" then press enter:



You should see a screen like this below, for the Interfacing Options. Use the arrow buttons on your keyboard to select "P7 1-Wire" and press enter.



You should see a screen like this below, asking if you want the 1-wire interface enabled. Use the arrow buttons on your keyboard to select "<Yes>" and press enter.



You should see a screen stating "The one-wire interface is enabled", indicating that the 1-wire interface is now enabled. Press enter for "<Ok>".

Press the "Esc" key to exit out of the raspi-config program.

> **NOTE: That the Raspberry Pi is not yet able to communicate with the Chip until the Raspberry Pi has been restarted, but wait to restart until the directions below to save time.**

STEP 3. Connect the alligator clips to the Chip.

Wiring Diagram

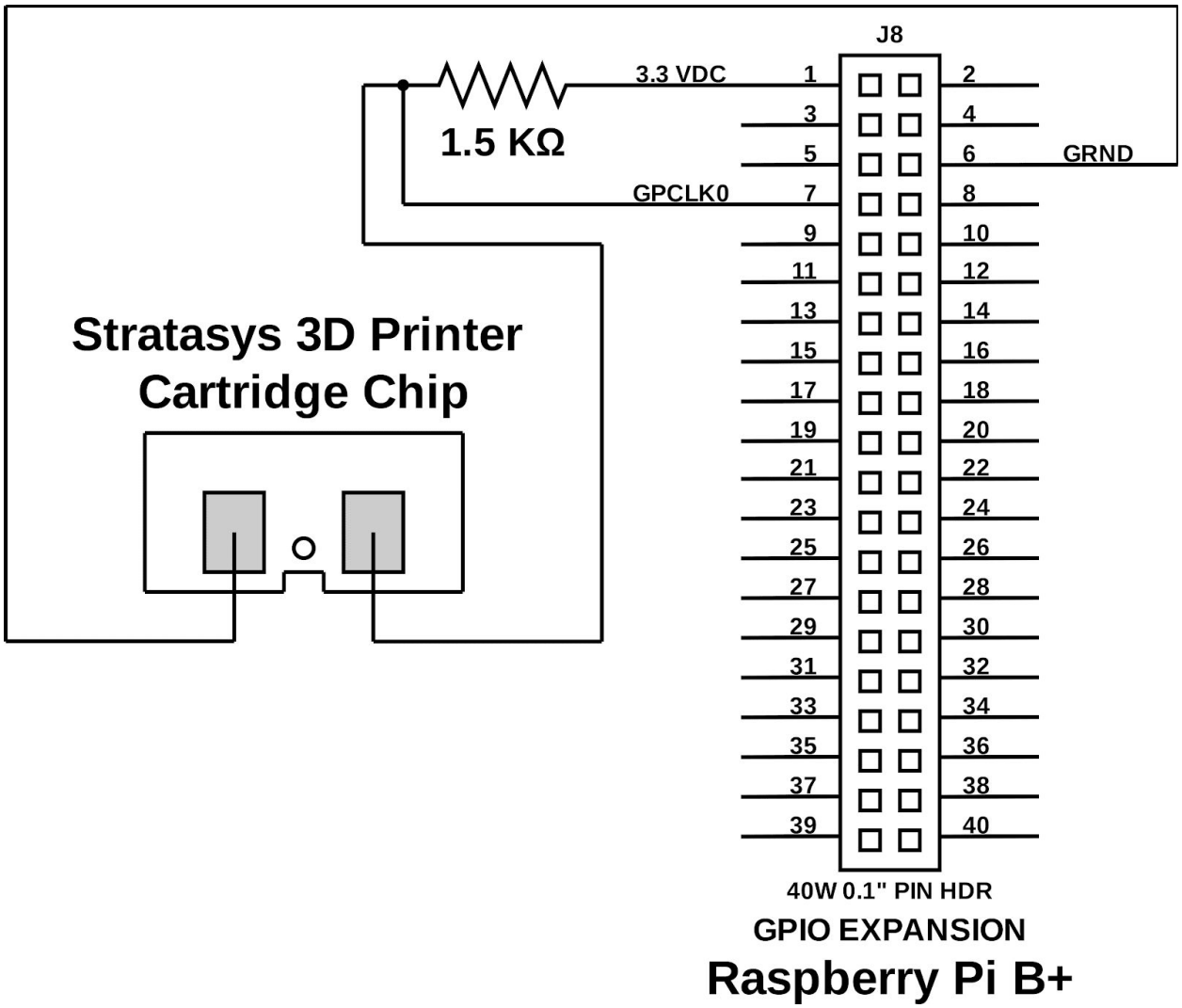


Fig. 5. Wiring Diagram

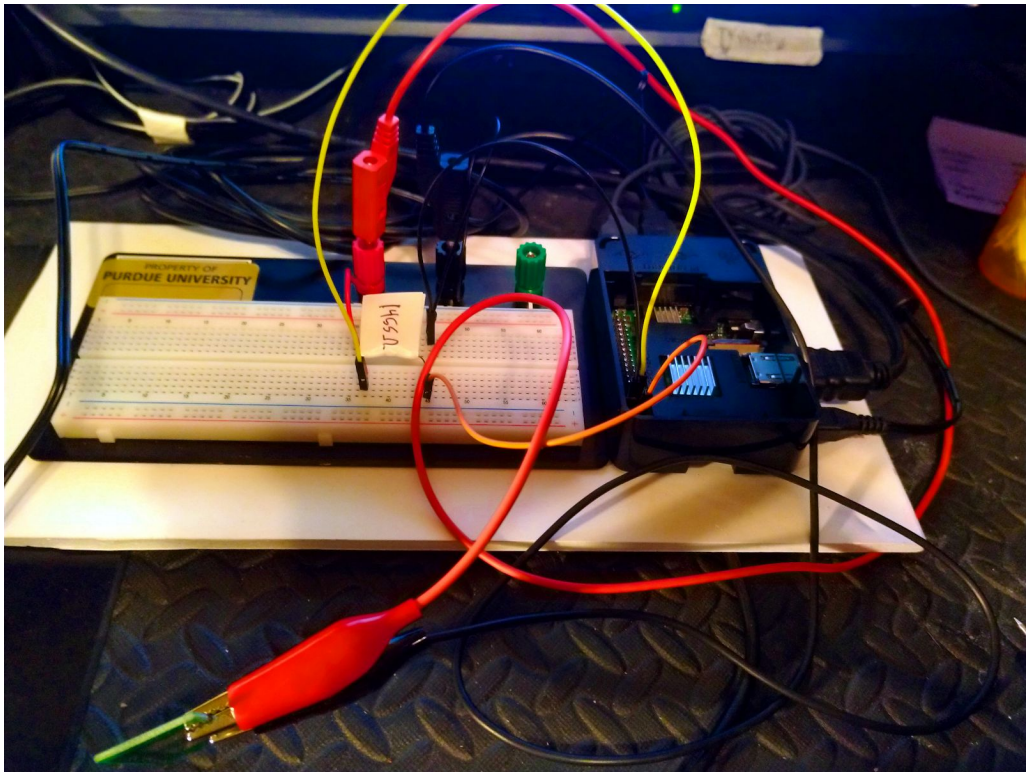


Fig. 6. Wiring setup for system, with a Raspberry Pi 3 B+ computer in a plastic case on the right.

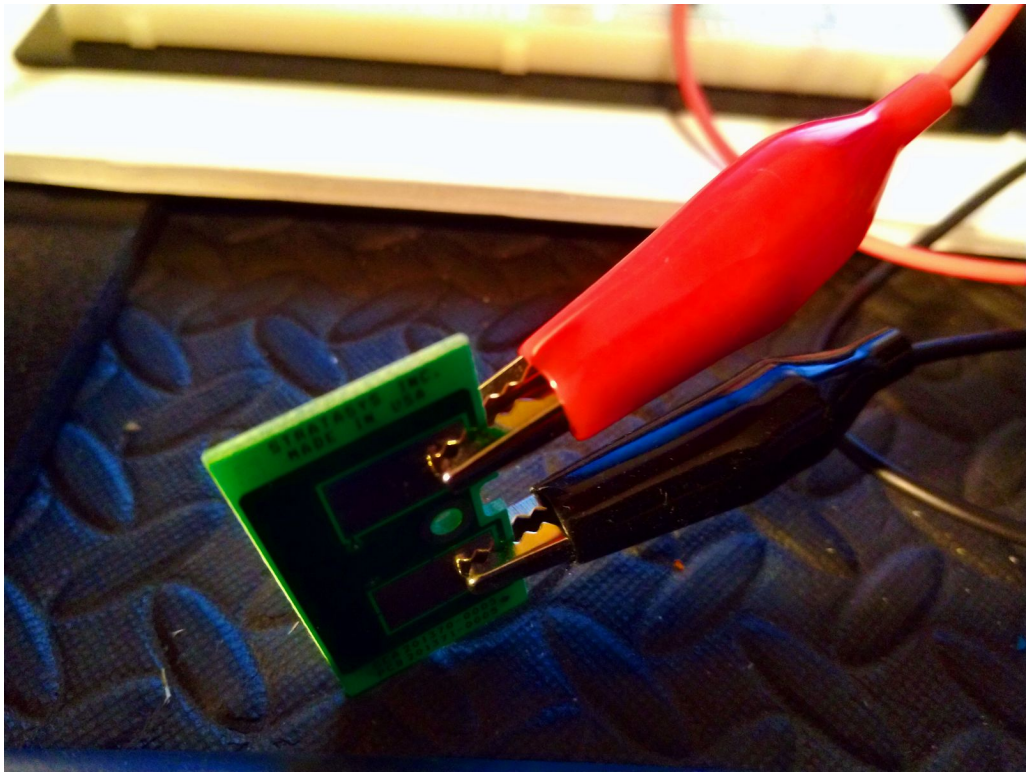


Fig. 7. Alligator Clips connected and "teeth" touching the pads on the Stratasys 3D Printer Chip.

STEP 4. Restart the Raspberry Pi.

NOTE: The reason we are restarting here is that occasionally the Chip will "timeout" with Linux devices. If that is the case then simply restart the Raspberry Pi.

STEP 5. After Every Restart of the Raspberry Pi.

NOTE:

Every time the Raspberry Pi reboots the following commands will need to be typed or copied and pasted into the Terminal. Otherwise, the Raspberry Pi may not "see" the Chip to use the 1-Wire communication protocol.

Once the Raspberry Pi has been restarted, open another Terminal and type commands into the Terminal. (They could be put into a Bash Script to make life easier.)

```
gpio mode 7 out
gpio write 7 1
sudo modprobe w1-gpio giopin=4
sudo modprobe w1-ds2433
```

What the above commands do is change the general-purpose input/output (GPIO) Raspberry Pi pin 7 to output then sets the pin to high, then it modifies the Linux Kernel to use the 1-wire protocol for the Broadcom BCM2837B0.

STEP 6. Optional Troubleshooting: Verify that the Raspberry Pi sees the Chip.

This part is mainly for troubleshooting because if there is an issue it can be with either the Chip, the wiring, the Raspberry Pi setup, or Linux. You can skip STEP 6, but it shows you what is going on with STEP 5.

After you have verified the wiring visually and if you want to verify that everything is set up correctly up to this point, type this command into the Terminal.

```
gpio readall
```

I have highlighted the GPIO 7 Pin that you are looking for.

```
+-----+-----+---------+------+---+---Pi 3B--+---+------+---------+-----+-----+
| BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
+-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
|     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
|   2 |   8 |   SDA.1 |   IN | 1 |  3 || 4  |   |      | 5v      |     |     |
|   3 |   9 |   SCL.1 |   IN | 1 |  5 || 6  |   |      | 0v      |     |     |
|   4 |   7 | GPIO. 7 |   IN | 1 |  7 || 8  | 0 | IN   | TxD     | 15  | 14  |
|     |     |      0v |      |   |  9 || 10 | 1 | IN   | RxD     | 16  | 15  |
|  17 |   0 | GPIO. 0 |   IN | 0 | 11 || 12 | 0 | IN   | GPIO. 1 | 1   | 18  |
|  27 |   2 | GPIO. 2 |   IN | 0 | 13 || 14 |   |      | 0v      |     |     |
|  22 |   3 | GPIO. 3 |   IN | 0 | 15 || 16 | 0 | IN   | GPIO. 4 | 4   | 23  |
|     |     |    3.3v |      |   | 17 || 18 | 0 | IN   | GPIO. 5 | 5   | 24  |
|  10 |  12 |    MOSI |   IN | 0 | 19 || 20 |   |      | 0v      |     |     |
|   9 |  13 |    MISO |   IN | 0 | 21 || 22 | 0 | IN   | GPIO. 6 | 6   | 25  |
|  11 |  14 |    SCLK |   IN | 0 | 23 || 24 | 1 | IN   | CE0     | 10  | 8   |
|     |     |      0v |      |   | 25 || 26 | 1 | IN   | CE1     | 11  | 7   |
|   0 |  30 |   SDA.0 |   IN | 1 | 27 || 28 | 1 | IN   | SCL.0   | 31  | 1   |
|   5 |  21 | GPIO.21 |   IN | 1 | 29 || 30 |   |      | 0v      |     |     |
|   6 |  22 | GPIO.22 |   IN | 1 | 31 || 32 | 0 | IN   | GPIO.26 | 26  | 12  |
|  13 |  23 | GPIO.23 |   IN | 0 | 33 || 34 |   |      | 0v      |     |     |
|  19 |  24 | GPIO.24 |   IN | 0 | 35 || 36 | 0 | IN   | GPIO.27 | 27  | 16  |
|  26 |  25 | GPIO.25 |   IN | 0 | 37 || 38 | 0 | IN   | GPIO.28 | 28  | 20  |
|     |     |      0v |      |   | 39 || 40 | 0 | IN   | GPIO.29 | 29  | 21  |
+-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
| BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
+-----+-----+---------+------+---+---Pi 3B--+---+------+---------+-----+-----+
```

You are looking for the GPIO 7 pin, here it saying that the mode is IN, Voltage is set to 1 high:

```
+-----+-----+---------+------+---+---Pi 3B
| BCM | wPi |   Name  | Mode | V | Physical
+-----+-----+---------+------+---+----+

|   4 |   7 | GPIO. 7 |   IN | 1 |  7 |
```

When you type the commands:

```
gpio mode 7 out
gpio readall
```

What you are looking for is GPIO 7 to have changed the mode to OUT and Voltage set to 0 low:

```
 +-----+-----+---------+------+---+---Pi 3B
 | BCM | wPi |   Name  | Mode | V | Physical
 +-----+-----+---------+------+---+----+

 |   4 |   7 | GPIO. 7 |  OUT | 0 |  7 |
```

Now type the commands:

```
gpio write 7 1
gpio readall
```

What you are looking for is GPIO 7, mode is OUT, and Voltage is 1 high.

```
 +-----+-----+---------+------+---+---Pi 3B
 | BCM | wPi |   Name  | Mode | V | Physical
 +-----+-----+---------+------+---+----+

 |   4 |   7 | GPIO. 7 |  OUT | 1 |  7 |
```

You can also type this in the Terminal if you want and it works just the same.

```
gpio mode 7 out | gpio write 7 1 | gpio readall
```

STEP 7. Get the unique device id of the chip as Linux sees it.

To get the unique device ID, which represents the Chip, it is necessary to query the Linux device system, this is done by simply getting a directory listing of the correct directory.

```
cd /sys/bus/w1/devices/
ls
```

This is what it should look like, you are looking for a directory that represents the chip which should start with "23-". If you don't see it but see the w1_bus_master1 then you need to restart the Raspberry Pi as the chip has probably timed out. Here the directory is "23-0000015834c2" (highlighted) but that is going to be different for your unique chip.

```
pi@raspberrypi:/sys/bus/w1/devices $ ls
23-0000015834c2  w1_bus_master1
```

Now go into that directory and list the contents. **This is important because the commands need to be run while pointing to the Linux device directory for the Chip.**

```
cd 23-0000015834c2
ls
```

It should look like this and we are looking for "eeprom" and "id" (highlighted).

```
pi@raspberrypi:/sys/bus/w1/devices $ cd cd 23-0000015834c2
pi@raspberrypi:/sys/bus/w1/devices/23-0000015834c2 $ ls
driver  eeprom  id  name  power  subsystem  uevent
```

To confirm the serial that the Chip has been assigned by Stratasys or by having already completed this process run this command, we need this for the "eeprom-uid":

```
xxd -p id
```

It should look something like this:

```
pi@raspberrypi:/sys/bus/w1/devices/23-0000015834c2 $ xxd -p id
23c23458010000ad
```

A note on the "eeprom-uid" is to keep it in this order, it seems as though the software "stratatools" has changed since 2013 and 2016, and does not need to be reversed. In this example the "eeprom-uid" is "23c23458010000ad".

STEP 8. Copy and decode the EEPROM.

Here is the "EEPROM" on the Chip, we can see the printable hexadecimal contents with this command:

```
xxd -p eeprom
```

It should look something like this:

```
pi@raspberrypi:/sys/bus/w1/devices/23-0000015834c2 $ xxd -p eeprom
aca1baea84cb524c4b2c12ccc8cc3833cf912926a5a97be4ceb7412af5ca
8985e76b0cf2428520dfa6953262c7f0d12fa6953262c7f0d12f2abcfb7f
143c3d08bfb900000000e44c55aa5537ece659be598f000000000000409d
95af15d71676ba56ff5400000000535452415441535953e8d368a8d39e94
9baee5f23fa87b83bebc8b598956ee28eaafcf7b6b96bfa0009a6b310243
be9320b02e2bdc24c75312cc02fd226971afcbecc335c1e166bab767648e
5340d88d834ee66856f0206ecf2d482ea6f79c2fbd1071d9a9c3841f4334
40d104551198c9e415fdf481d89d4df5d105111047cef019f0af94da47d8
32f79d98b6ce669f0a7d456ec2c66a984506231c1950b16f1b959b213c1e
1328d0bafbacb702ff3c03c5f6f19f8b450a8d11536fa2406e1d034b8242
18e7031675cf3587055c2507d1af010f575242fa31eff820891f293d4c0c
4b288ff8d88dec079935854480bdd05b7d798a1e48f072a6010cdb576885
1098eab41765700e672c2d2f07c48fb77b58803757fd700df1cedf163bc9
b9f0dba6243de65327316087abf852a9534ef454cf7761f1c3e480573b60
898b42d9b78310226b6757b3ad3316bd39d654d5a38062f92f06abe8410e
d52d6b9903142348355dab89465ce6d69b627bcc895461face9de1616f63
476ee510ce8314dc82209322ffc6a71563e62052750ff106798dbfbbbdde
ad1a
```

But this is not particularly helpful to us so we need to pass it to the "stratatools" program and to do that we are going to copy the binary eeprom. **Type this command to make a backup.**

```
cp eeprom ~/original23-0000015834c2.bin
```

We now have a copy of the EEPROM binary backed up in the file "original23-0000015834c2.bin" located in the current logged in user's home directory "~/".

Now type this command into the Terminal.

```
stratatools eeprom_decode --machine-type prodigy \
    --eeprom-uid 23c23458010000ad \
    eeprom > ~/orginal23-0000015834c2-info.txt
```

Remember that the "eeprom-uid" is what we got from the "xxd -p id" command. What the above command says is, "stratatools" decode the EEPROM, with machine type prodigy (which works for Dimension 3D models), using the eeprom-uid from the Linux device "eeprom" of the chip, then output that to a text file in the user's home directory.

Type this into the Terminal.

```
cat ~/orginal23-0000015834c2-info.txt
```

What the above command says is to display the text contents to the user in the Terminal. We could also use the "less" or "more" command here instead of "cat". You are going to want to get the most of the information from that text file to use in STEP 9. Since this is a text file you can edit and save a new text file if you want. You should see something like this below.

```
serial_number: 185371110.0
material_name: "P400SR"
manufacturing_lot: "5108"
manufacturing_date {
  seconds: 1369792725
}
last_use_date {
  seconds: 1369792725
}
initial_material_quantity: 56.3
current_material_quantity: 3.1855483
key_fragment: "55aa5537ece659be"
version: 1
signature: "STRATASYS"
```

STEP 9. Create the New EEPROM and modify the contents before overwriting it to the Chip.

Type this command into the Terminal.

```
stratatools eeprom_create --serial-number 185372110.0 \
    --material-name P400SR --manufacturing-lot 5108 \
    --manufacturing-date "2001-01-01 01:01:01" \
    --use-date "2002-02-02 02:02:02" \
    --initial-material 53.6 --current-material 53.6 \
    --key-fragment 55aa5537ece659be --version 1 \
    --signature STRATASYS > ~/cartridge.txt
```

What the above command says is, "stratatools" create a EEPROM with this serial number, material type, manufacturing lot number, date, use date, initial material, current material, key fragment, version, signature and save it to the file "cartridge.txt" in the user's home directory.

NOTE:

While you can get the "serial-number" from the original chip, it needs to be incremented or changed. The "serial-number" should be changed every time a new EEPROM is to be written, because once you put a modified Chip into the Stratasys 3D printer it will record that serial "serial-number" in its database, so you can't just change the "current-material" amount to a higher amount. A lower "current-material" amount would work, but not a higher amount. You will need to generate a new unique serial number for that Chip using the Raspberry Pi again to give the modified Chip a higher amount.

The maximum amount for "initial-material" and "current-material" is 53.6. This cannot be changed as it is set by the 3D printer.

The "material-name" must match a valid Stratasys material type: "P400SR" works, and it is recommended to collect the "material-name" of the original cartridges to find one that can be used for the melting temperature of the material that replaces the original material spool. For example, does P400SR material have the same melting point as ABS, or similar thermal properties?

Get the "manufacturing-lot", "manufacturing-date", "use-date", and "key-fragment" from STEP 8. Note that the date is in UNIX time if displayed in seconds.

STEP 10. Encode the EEPROM into a Binary

Type this command into the Terminal.

```
stratatools eeprom_encode --machine-type prodigy \
    --eeprom-uid 23c23458010000ad ~/cartridge.txt ~/cartridge.bin
```

What the above command says is, "stratatools" encode the EEPROM using the machine type and EEPROM uid from the cartridge text file into a cartridge binary file in the user's home directory.

Remember that the "eeprom-uid" is what we got from the "xxd -p id" command.

STEP 11. Write the New EEPROM Binary to the Chip

NOTE:

If you have not made a copy of the original EEPROM this is your last chance before you run the next command. The original EEPROM will be lost forever. Seriously, go make a backup now. See STEP 8.

Type this command into the Terminal.

```
sudo cp ~/cartridge.bin eeprom
```

What the above command says is to run this command with elevated admin privileges and copy the new cartridge binary file into the EEPROM Linux file, overwriting the original EEPROM.

Now you have a new EEPROM chip as far as the 3D printer is concerned.

STEP 12. Confirm that the New EEPROM has been written to the Chip.

Type this command into the Terminal.

```
stratatools eeprom_decode --machine-type prodigy \
    --eeprom-uid 23c23458010000ad eeprom
```

What the above command says is, "stratatools" decode the EEPROM assuming machine type and the EEPROM uid.

It should say something similar to this:

```
pi@raspberrypi:/sys/bus/w1/devices/23-0000015834c2 $ stratatools
eeprom_decode --machine-type prodigy --eeprom-uid 23c23458010000ad
/sys/bus/w1/devices/w1_bus_master1/23-0000015834c2/eeprom

serial_number: 185372110.0
material_name: "P400SR"
manufacturing_lot: "5108"
manufacturing_date {
  seconds: 1369792725
}
last_use_date {
  seconds: 1012615322
}
initial_material_quantity: 56.3
current_material_quantity: 56.3
key_fragment: "55aa5537ece659be"
version: 1
signature: "STRATASYS"
```

Note that the Unix time of 1586809628 seconds is the same as "2001-01-01 01:01:01". You should see that the serial_number and the current_material_quantity have been changed from the original. If it has, congratulations you have completed the process successfully!

Works Cited

[3DPC] "Dimension SST 768 in San Carlos California,Stratasys for Sale | 3D Printer Classifieds". *3DPrinterClassifieds.com*. 26 Dec. 2013. http://www.3dprinterclassifieds.com/3d-printers/dimension-sst-768/. Accessed 7 Apr. 2020.

[AET] "Dimension 768 Material Cartridges | AET Labs". *AET Labs*. https://aetlabs.com/product/model-material-cartridge-p-400/. Accessed 7 Apr. 2020.

[ERESET] "Tomblarom/Pi-Cartridge-Resetter: Build project for EEPROM resetter". *Tomblarom*. 9 Nov. 2017. https://github.com/Tomblarom/Pi-Cartridge-Resetter. Accessed 7 Apr. 2020.

[GPIO] "1-WIRE at Raspberry Pi GPIO Pinout". *Pinout.xyz*. ND. https://pinout.xyz/pinout/1_wire. Accessed 7. Apr. 2020.

[HAVEBLUE] "Stratasys EEPROM hack revisted – Have Blue [dot org]". *Have Blue*. 3 Mar. 2016. https://haveblue.org/?p=1988. Accessed 7 Apr. 2020.

[LB] "ABS by Village Plastics - ABS | LulzBot". *LulzBot*. https://www.lulzbot.com/store/filament/abs. Accessed 7 Apr. 2020.

[LOCKIN] Liebowitz, S. J.; Margolis, Stephen E. "Path Dependence entry - Look, I understand to little too late.". *The New Palgraves Dictionary of Economics and the Law*. MacMillan. 1998. https://personal.utdallas.edu/~liebowit/palgrave/palpd.html. Accessed 7 Apr. 2020.

[MIT] "dimension BST 768, SST 768, & Dimension Elite User Guide". *Stratasys.* 2007. http://fab.cba.mit.edu/content/tools/dimension/Dimension%20768%20Elite%20User%20Guide.pdfAccessed 7 Apr. 2020.

[MODPRO] "Adding Modprobe w1-gpio / Modprobe w1-therm to boot - Raspberry Pi Forums". *Raspberry Pi Foundation*. 13 Feb. 2013. https://www.raspberrypi.org/forums/viewtopic.php?t=35508. Accessed 7 Apr. 2020.

[NOOBS] "Index of /NOOBS/images/NOOBS-2019-09-30". *Raspberry Pi Foundation*. 30 Sep. 2019. https://downloads.raspberrypi.org/NOOBS/images/NOOBS-2019-09-30/. Accessed 7 Apr. 2020.

[PCB] "Nothing Doing: PCB for Stratasys Cartridges". *chasenachtmann.com*. 14 Jan. 2015. https://blog.chasenachtmann.com/2015/01/pcb-for-stratasys-cartridges.html. Accessed 7 Apr. 2020.

[PYCR] Litzenberger, Darsey. "PyCrypto - The Python Cryptography Toolkit". *dlitz.net*. 2013. https://www.dlitz.net/software/pycrypto/. Accessed 7 Apr. 2020.

[STEETO] Vanheuverzwijn, Benjamin. "GitHub - bvanheu/stratatools: Stratasys EEPROM tool". *bvanheu*. 7 Oct. 2019. https://github.com/bvanheu/stratatools. Accessed 7 Apr. 2020.

[WIRDIA] Thornton, Roger. "Raspberry Pi 3 Model B+ (Reduced Schematic), Page 2, GPIO EXPANSION". *Raspberry Pi Foundation*. 19 Mar. 2018. https://www.raspberrypi.org/documentation/hardware/raspberrypi/schematics/rpi_SCH_3bplus_1p0_reduced.pdf. Accessed 7 Apr. 2020.