

Intro to the Raspberry Pi (RPi) & The Internet of Things (IoT)

Class 7 – 2018 / 04 / 29

Step 10 – Final Solution from Last Class

- <https://www.michael-wessel.info/class6/step10.py>
- Let's walk through this together
- Take some time to understand the program!
- Today: add the LCD to show the score!



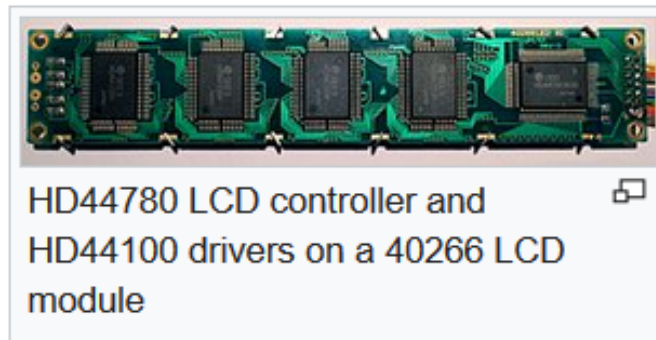
First Laptop – Epson HX 20 – 7 / 1982



Hitachi HD44780 LCD controller

From Wikipedia, the free encyclopedia

The **Hitachi HD44780 LCD controller** is an alphanumeric [dot matrix liquid crystal display](#) (LCD) controller developed by [Hitachi](#). It was made commercially available around 1987.^[*citation needed*] The character set of the controller includes [ASCII](#) characters, Japanese [Kana](#) characters, and some symbols in two 28 character lines. Using an extension driver, the device can display up to 80 characters.^[1]



Contents [\[hide\]](#)

- 1 [Architecture](#)
 - 1.1 [Mode Selection](#)
- 2 [Instruction set](#)

1. Ground
2. VCC (+3.3 to +5V)
3. Contrast adjustment (VO)
4. Register Select (RS). RS=0: Command, RS=1: Data
5. Read/Write (R/W). R/W=0: Write, R/W=1: Read (This pin is optional due to the fact that most of the time you will only want to write to it and not read. Therefore, in general use, this pin will be permanently connected directly to ground.)
6. Clock (Enable). Falling edge triggered
7. Bit 0 (Not used in 4-bit operation)
8. Bit 1 (Not used in 4-bit operation)
9. Bit 2 (Not used in 4-bit operation)
10. Bit 3 (Not used in 4-bit operation)
11. Bit 4
12. Bit 5
13. Bit 6
14. Bit 7
15. Backlight Anode (+) (If applicable)
16. Backlight Cathode (-) (If applicable)



Amber backlight on a HD44780 display

Instruction set [\[edit \]](#)

The HD44780 instruction set is shown below:^[3]

HD44780U based instruction set

Instruction	Code										Description	Execution time (max) (when $f_{cp} = 270 \text{ kHz}$)
	RS	R/W	B7	B6	B5	B4	B3	B2	B1	B0		
Clear display	0	0	0	0	0	0	0	0	0	1	Clears display and returns cursor to the home position (address 0).	1.52 ms
Cursor home	0	0	0	0	0	0	0	0	1	*	Returns cursor to home position. Also returns display being shifted to the original position. DDRAM content remains unchanged.	1.52 ms

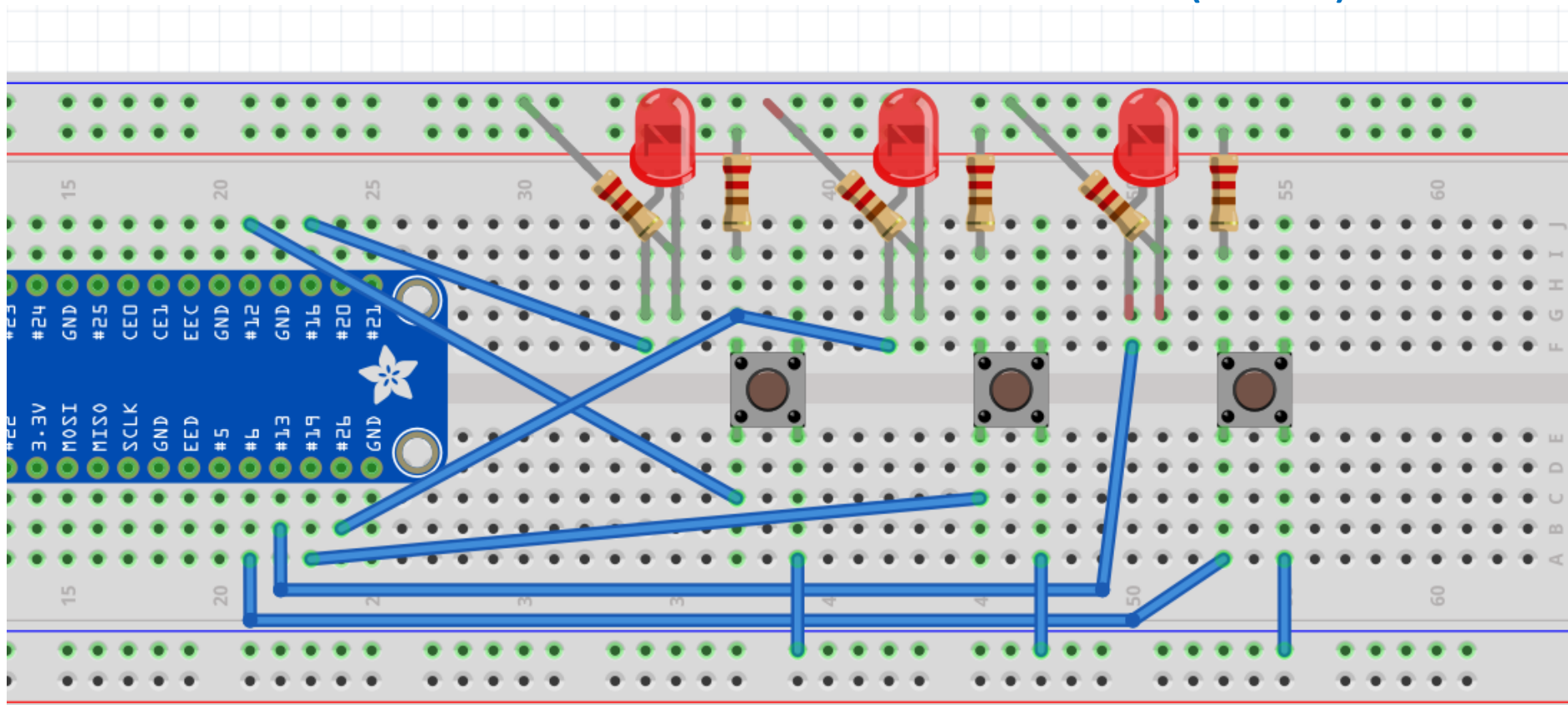
LEDs: 16 26 13 via cables to long leads of LEDs

220 Ohm Resistors from SHORT LEAD to GND (BLUE)

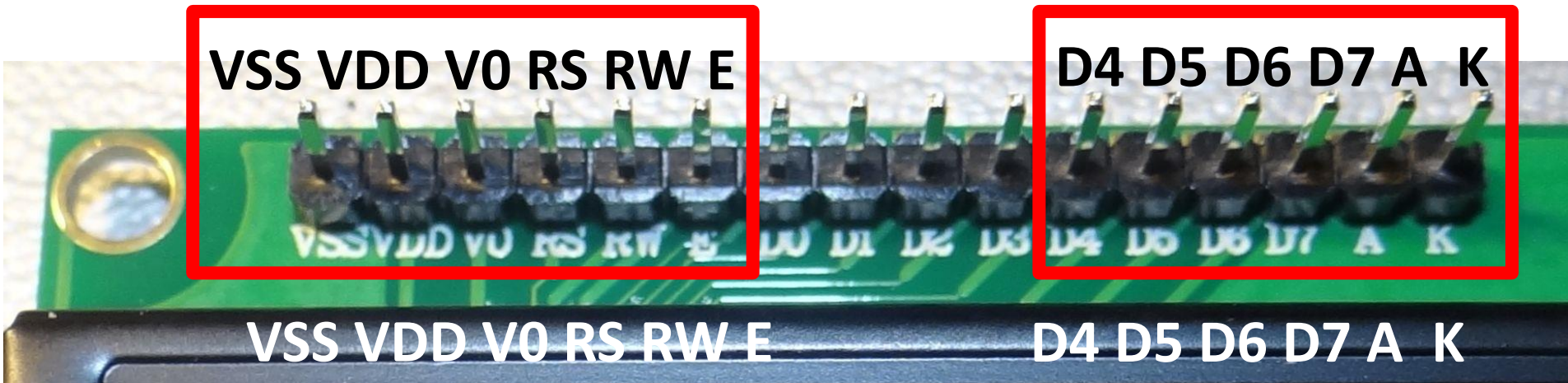
BUTTONS: 12 19 6 via cables to left side pins of buttons

10 kOhm PULL-UP RESISTORS TO PLUS 5 V (RED)

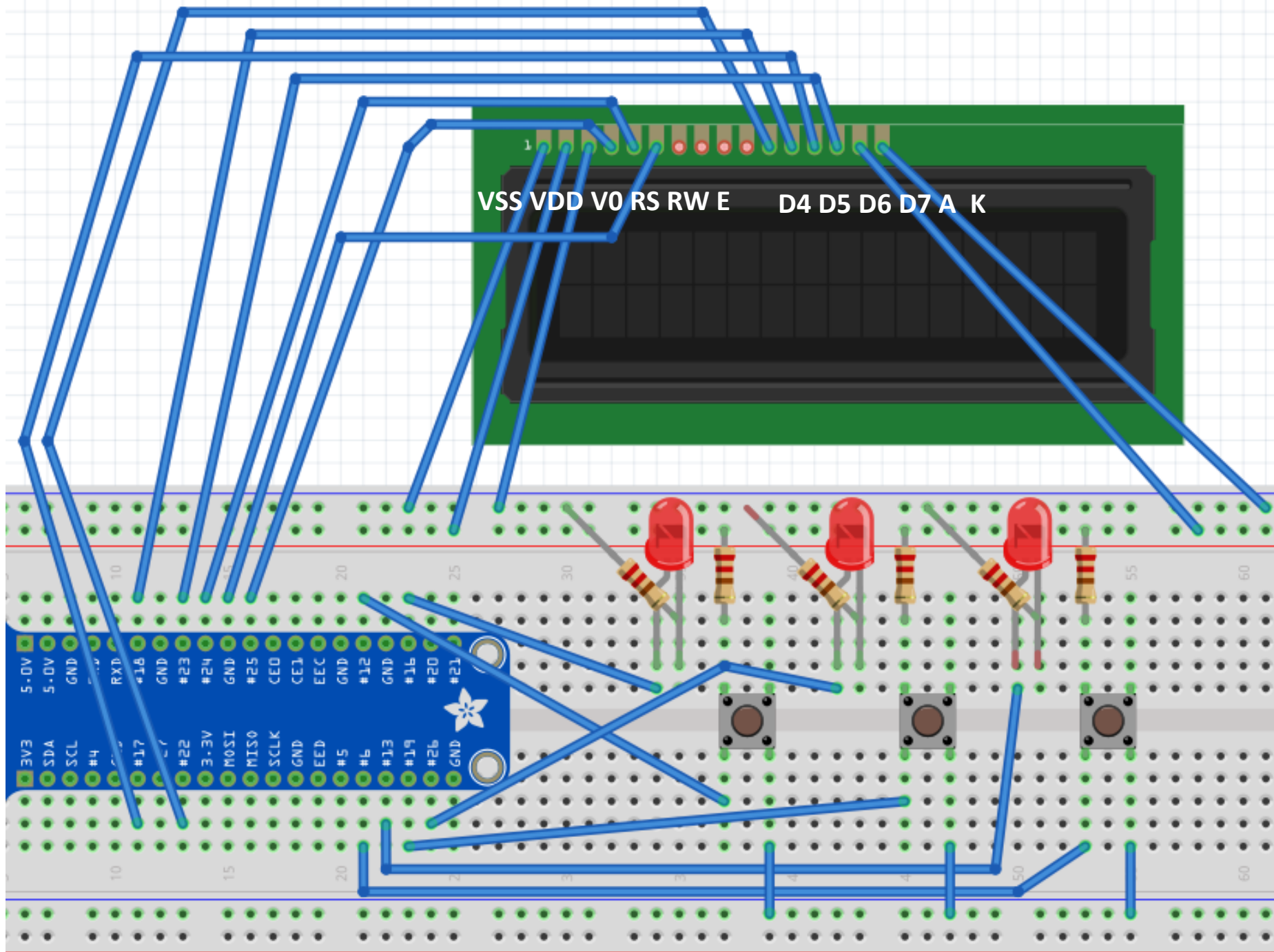
BUTTON RIGHT SIDE PINS TO GND (BLUE)



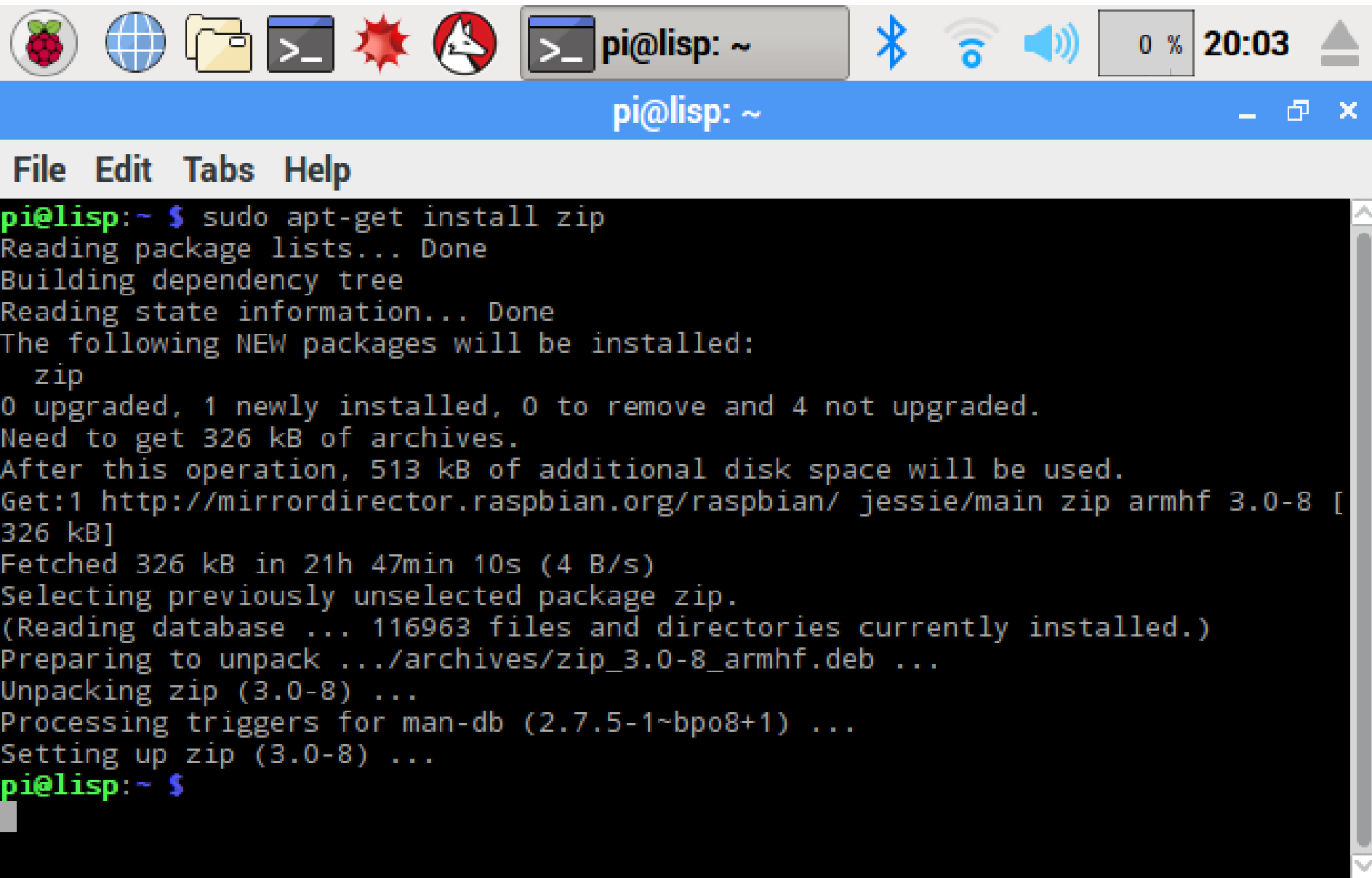
Let's hook up the LCD!



- Background Light Power Supply
 - K (Cathode)
= MINUS (BLUE, GND)
 - A (Anode)
= 5 V (RED)
- Data Lines to LCD
 - D7 = 22
 - D6 = 18
 - D5 = 17
 - D4 = 23
- Data Control Lines
 - E = 24
 - RW = MINUS (BLUE, GND)
 - RS = 25
 - VO = MINUS (BLUE, GND)
- LCD Main Power Supply
 - VDD = 5 V (RED)
 - VSS = MINUS (BLUE, GND)



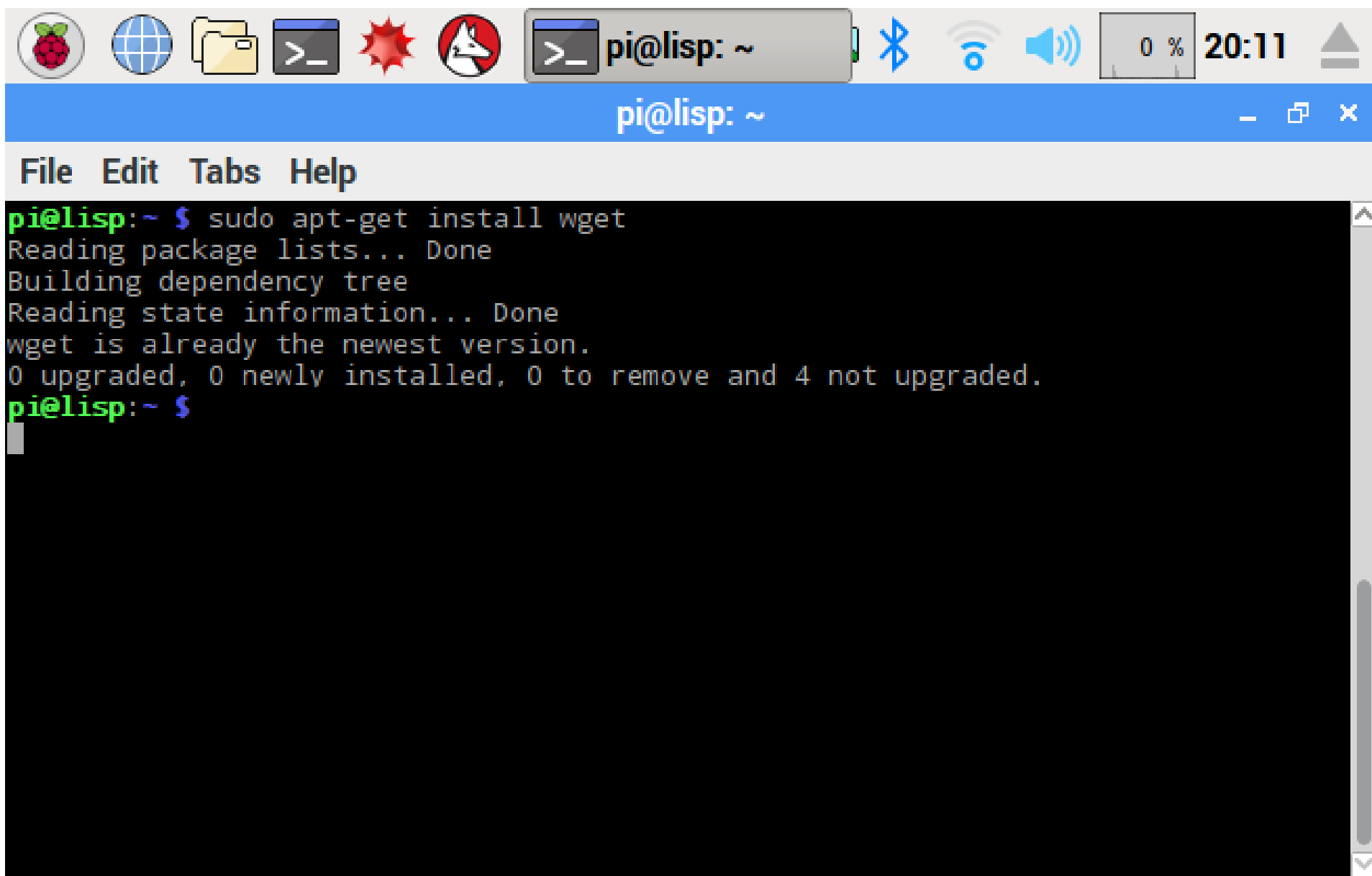
Software Installation - ZIP



The image shows a terminal window on a Raspberry Pi. The top bar contains various system icons: Raspberry Pi logo, network, file manager, terminal, a red star, a dog icon, and a terminal icon with the prompt 'pi@lisp: ~'. To the right are Bluetooth, Wi-Fi, and volume icons, followed by a battery indicator at 0% and the time 20:03. The terminal window has a blue title bar with the prompt 'pi@lisp: ~' and standard window controls. The menu bar includes 'File', 'Edit', 'Tabs', and 'Help'. The terminal output shows the command 'sudo apt-get install zip' and its execution details.

```
pi@lisp:~ $ sudo apt-get install zip
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  zip
0 upgraded, 1 newly installed, 0 to remove and 4 not upgraded.
Need to get 326 kB of archives.
After this operation, 513 kB of additional disk space will be used.
Get:1 http://mirrordirector.raspbian.org/raspbian/ jessie/main zip armhf 3.0-8 [
326 kB]
Fetched 326 kB in 21h 47min 10s (4 B/s)
Selecting previously unselected package zip.
(Reading database ... 116963 files and directories currently installed.)
Preparing to unpack .../archives/zip_3.0-8_armhf.deb ...
Unpacking zip (3.0-8) ...
Processing triggers for man-db (2.7.5-1~bpo8+1) ...
Setting up zip (3.0-8) ...
pi@lisp:~ $
```

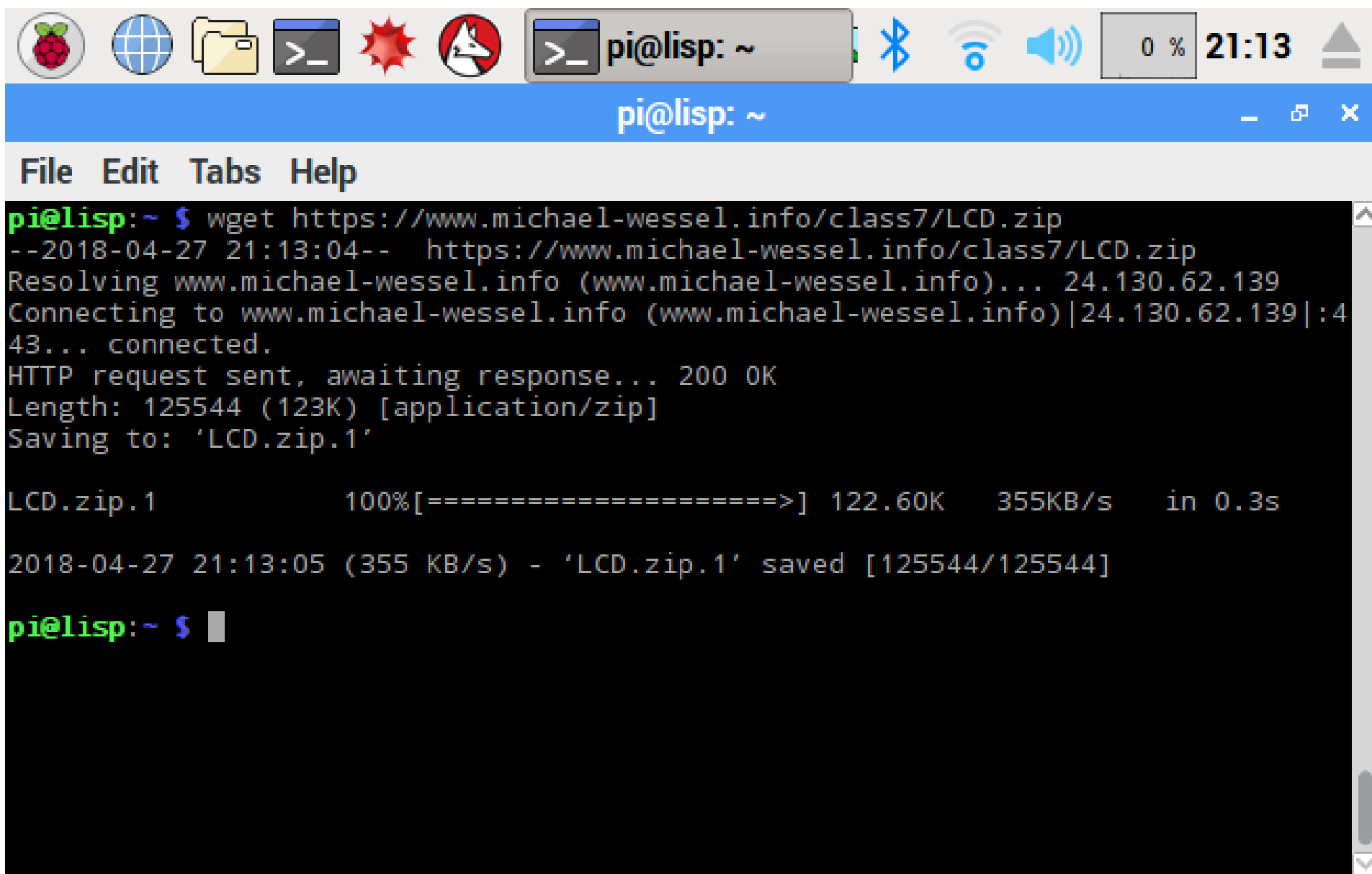
Software Installation – WGET



The screenshot shows a Raspberry Pi desktop environment. The top panel contains several icons: a Raspberry Pi logo, a globe, a folder, a terminal icon, a red star, a cat icon, and a taskbar showing 'pi@lisp: ~'. To the right of the taskbar are icons for Bluetooth, Wi-Fi, and a speaker, along with a battery indicator at 0% and a clock showing 20:11. Below the taskbar is a blue header bar for the terminal window, labeled 'pi@lisp: ~'. The terminal window has a menu bar with 'File', 'Edit', 'Tabs', and 'Help'. The terminal output shows the command 'sudo apt-get install wget' being executed, followed by the output: 'Reading package lists... Done', 'Building dependency tree', 'Reading state information... Done', 'wget is already the newest version.', and '0 upgraded, 0 newly installed, 0 to remove and 4 not upgraded.' The prompt 'pi@lisp:~ \$' is shown at the end of the output.

```
pi@lisp:~ $ sudo apt-get install wget
Reading package lists... Done
Building dependency tree
Reading state information... Done
wget is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 4 not upgraded.
pi@lisp:~ $
```

Download LCD.zip via WGET



The screenshot shows a terminal window on a Raspberry Pi. The top panel displays system icons including the Raspberry Pi logo, network status, Bluetooth, and battery level. The terminal title bar reads 'pi@lisp: ~'. The menu bar includes 'File', 'Edit', 'Tabs', and 'Help'. The terminal output shows the execution of the 'wget' command to download 'LCD.zip' from 'https://www.michael-wessel.info/class7/LCD.zip'. The download is successful, with a status of '200 OK' and a file size of 125544 bytes (123K). The file is saved as 'LCD.zip.1'. A progress bar indicates 100% completion. The final prompt shows the user is ready for the next command.

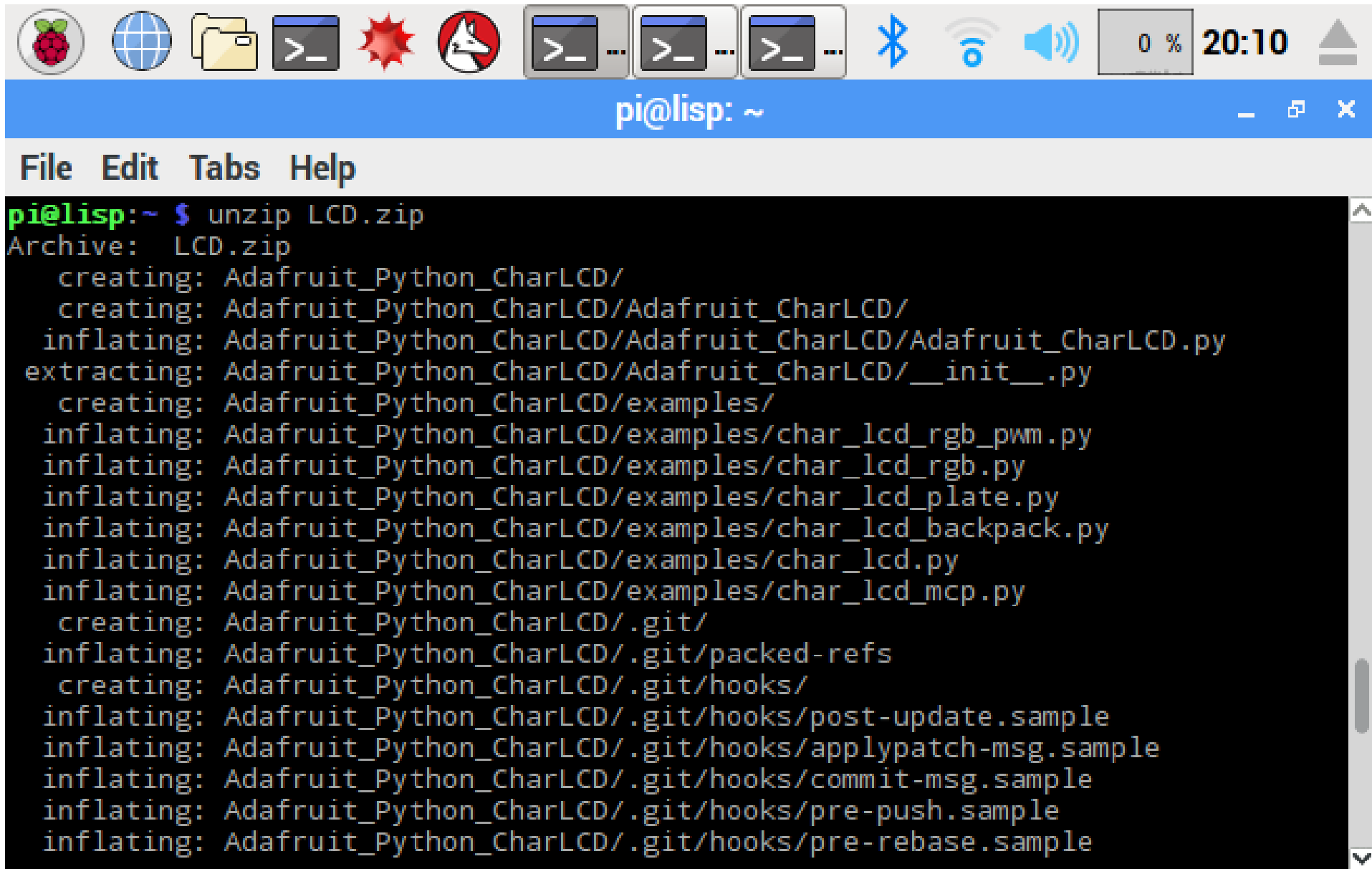
```
pi@lisp:~ $ wget https://www.michael-wessel.info/class7/LCD.zip
--2018-04-27 21:13:04-- https://www.michael-wessel.info/class7/LCD.zip
Resolving www.michael-wessel.info (www.michael-wessel.info)... 24.130.62.139
Connecting to www.michael-wessel.info (www.michael-wessel.info)|24.130.62.139|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 125544 (123K) [application/zip]
Saving to: 'LCD.zip.1'

LCD.zip.1          100%[=====>] 122.60K   355KB/s   in 0.3s

2018-04-27 21:13:05 (355 KB/s) - 'LCD.zip.1' saved [125544/125544]

pi@lisp:~ $
```

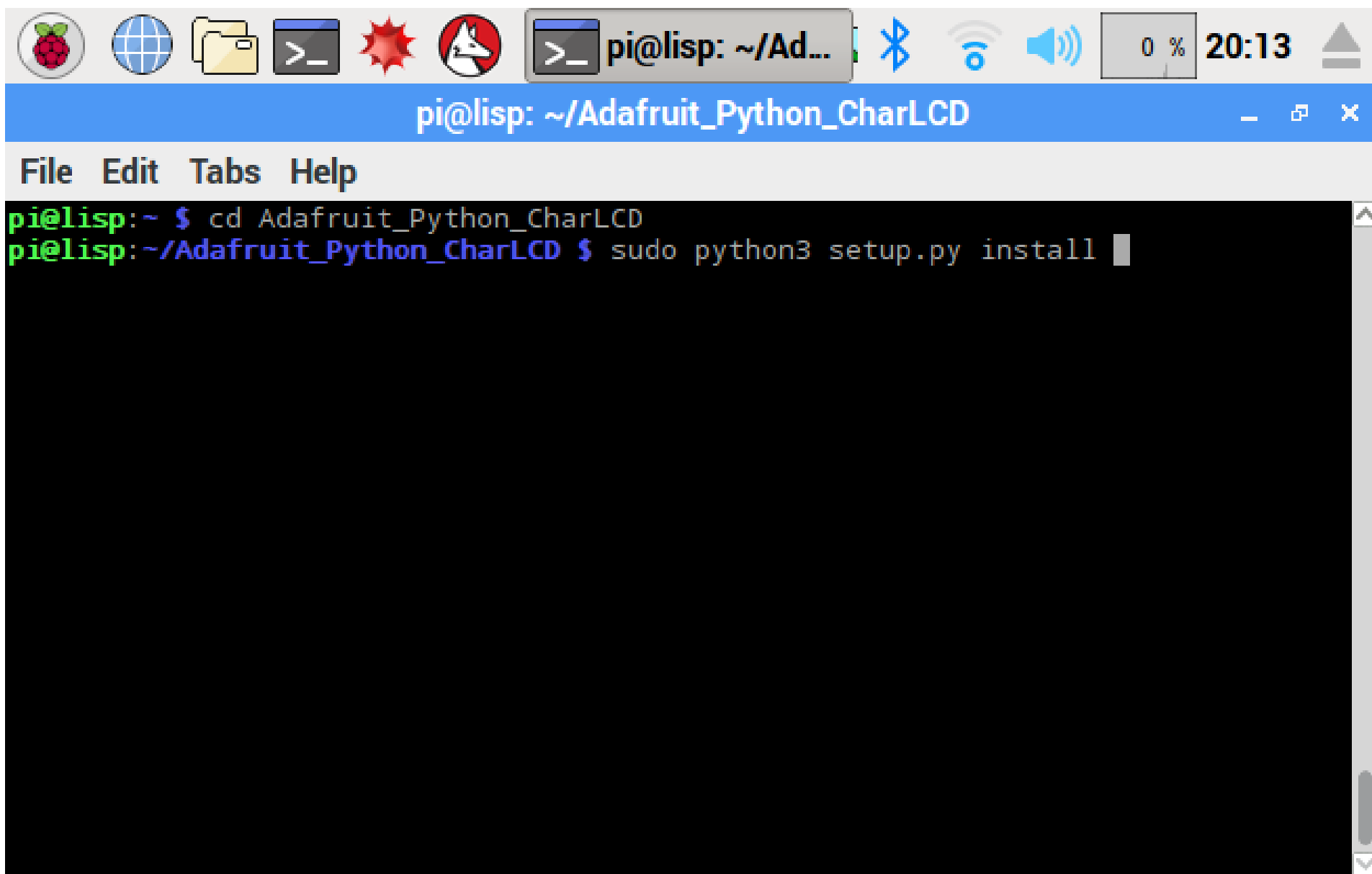
Unzip LCD Library



A terminal window on a Raspberry Pi desktop. The desktop background is white. The top panel shows various icons: Raspberry Pi logo, a globe, a folder, a terminal icon with a red star, a terminal icon with a white horse, three terminal icons with a blue bar, a Bluetooth icon, a Wi-Fi icon, a speaker icon, a battery icon at 0%, and the time 20:10. The terminal window has a blue title bar with the text 'pi@lisp: ~'. Below the title bar is a menu bar with 'File', 'Edit', 'Tabs', and 'Help'. The terminal content shows the command 'unzip LCD.zip' and its output, which lists the files being extracted from the archive.

```
pi@lisp:~ $ unzip LCD.zip
Archive:  LCD.zip
  creating: Adafruit_Python_CharLCD/
  creating: Adafruit_Python_CharLCD/Adafruit_CharLCD/
  inflating: Adafruit_Python_CharLCD/Adafruit_CharLCD/Adafruit_CharLCD.py
  extracting: Adafruit_Python_CharLCD/Adafruit_CharLCD/__init__.py
  creating: Adafruit_Python_CharLCD/examples/
  inflating: Adafruit_Python_CharLCD/examples/char_lcd_rgb_pwm.py
  inflating: Adafruit_Python_CharLCD/examples/char_lcd_rgb.py
  inflating: Adafruit_Python_CharLCD/examples/char_lcd_plate.py
  inflating: Adafruit_Python_CharLCD/examples/char_lcd_backpack.py
  inflating: Adafruit_Python_CharLCD/examples/char_lcd.py
  inflating: Adafruit_Python_CharLCD/examples/char_lcd_mcp.py
  creating: Adafruit_Python_CharLCD/.git/
  inflating: Adafruit_Python_CharLCD/.git/packed-refs
  creating: Adafruit_Python_CharLCD/.git/hooks/
  inflating: Adafruit_Python_CharLCD/.git/hooks/post-update.sample
  inflating: Adafruit_Python_CharLCD/.git/hooks/applypatch-msg.sample
  inflating: Adafruit_Python_CharLCD/.git/hooks/commit-msg.sample
  inflating: Adafruit_Python_CharLCD/.git/hooks/pre-push.sample
  inflating: Adafruit_Python_CharLCD/.git/hooks/pre-rebase.sample
```

Install LCD Library

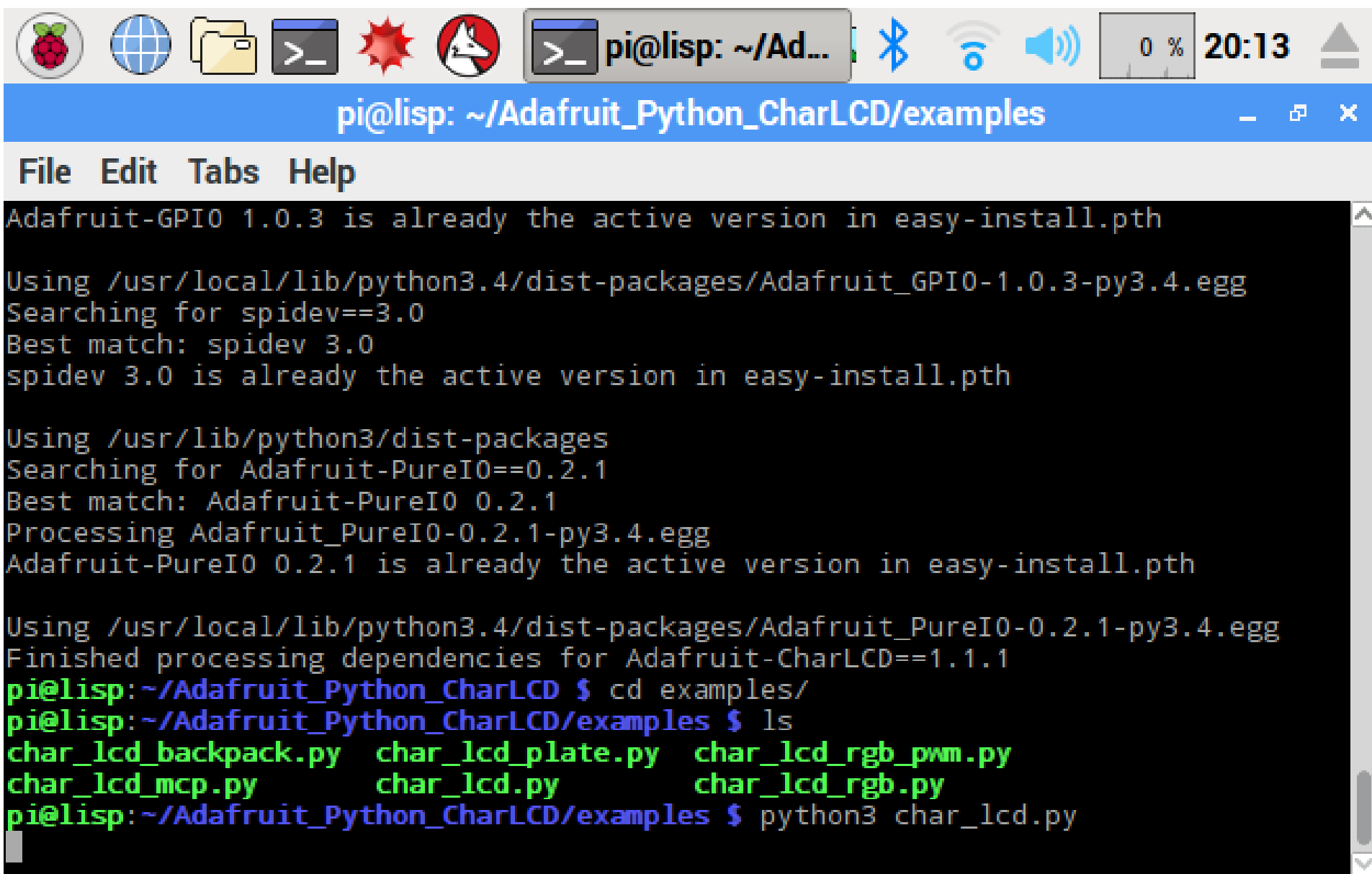


The image shows a terminal window on a Raspberry Pi. The window title is "pi@lisp: ~/Adafruit_Python_CharLCD". The terminal displays the following commands and output:

```
pi@lisp:~ $ cd Adafruit_Python_CharLCD
pi@lisp:~/Adafruit_Python_CharLCD $ sudo python3 setup.py install
```

The terminal window has a menu bar with "File", "Edit", "Tabs", and "Help". The system tray at the top shows various icons including the Raspberry Pi logo, a globe, a folder, a terminal icon, a red star, a red circle with a white horse, a Bluetooth icon, a Wi-Fi icon, a speaker icon, a battery icon at 0%, and the time 20:13.

Test the LCD



The image shows a terminal window on a Raspberry Pi. The title bar indicates the user is 'pi@lisp' in the directory '~/Adafruit_Python_CharLCD/examples'. The terminal output shows the installation of Adafruit-GPIO 1.0.3 and Adafruit-PureIO 0.2.1, followed by the execution of 'cd examples/' and 'ls', which lists several Python files. The final command shown is 'python3 char_lcd.py'.

```
pi@lisp: ~/Adafruit_Python_CharLCD/examples
File Edit Tabs Help
Adafruit-GPIO 1.0.3 is already the active version in easy-install.pth
Using /usr/local/lib/python3.4/dist-packages/Adafruit_GPIO-1.0.3-py3.4.egg
Searching for spidev==3.0
Best match: spidev 3.0
spidev 3.0 is already the active version in easy-install.pth

Using /usr/lib/python3/dist-packages
Searching for Adafruit-PureIO==0.2.1
Best match: Adafruit-PureIO 0.2.1
Processing Adafruit_PureIO-0.2.1-py3.4.egg
Adafruit-PureIO 0.2.1 is already the active version in easy-install.pth

Using /usr/local/lib/python3.4/dist-packages/Adafruit_PureIO-0.2.1-py3.4.egg
Finished processing dependencies for Adafruit-CharLCD==1.1.1
pi@lisp:~/Adafruit_Python_CharLCD $ cd examples/
pi@lisp:~/Adafruit_Python_CharLCD/examples $ ls
char_lcd_backpack.py  char_lcd_plate.py  char_lcd_rgb_pwm.py
char_lcd_mcp.py       char_lcd.py         char_lcd_rgb.py
pi@lisp:~/Adafruit_Python_CharLCD/examples $ python3 char_lcd.py
```

Warming up with the LCD

- Download
<https://www.michael-wessel.info/class7/lcd-warmup.py>
- Clear the LCD:
lcd.clear()
- Set Cursor for Output (where to write next):
lcd.set_cursor(<row>,<column>)
 - <row> from 0 to 3
 - <column> from 0 to 19
- Print:
lcd.message("Catch The LED!")
- Print with arguments:
use „%s“ and „%“
lcd.message(„Hello %s!“ % name)

Continue the Game Development

- Add the LCD initialization and setup
- Download

<https://www.michael-wessel.info/class7/step11.py>

Step 12

- Extend
<https://www.michael-wessel.info/class7/step11.py>
- Add, to the main loop, display of
 - Number of tries left
 - Current points

Step 13

- Extend
<https://www.michael-wessel.info/class7/step12.py>
- Add random encouragement messages
 - When user got a point
 - When user hit the wrong button
 - When user hit a button, but too late!
More difficult, think about it...
 - When user cheated

**Make sure message are readable long enough!!
(e.g., 0.3 seconds)**

Step 14

- Extend
<https://www.michael-wessel.info/class7/step13.py>
- At the end of the game, report
 - GAME OVER
 - Overall number of points
 - Give a rating / grade:
 - > **25 points: Excellent**
 - > **20 points: Very Good**
 - > **15 points: Good**
 - > **10 points: Not so great**
 - > **5 point: No comment**
 - Otherwise: terrible**

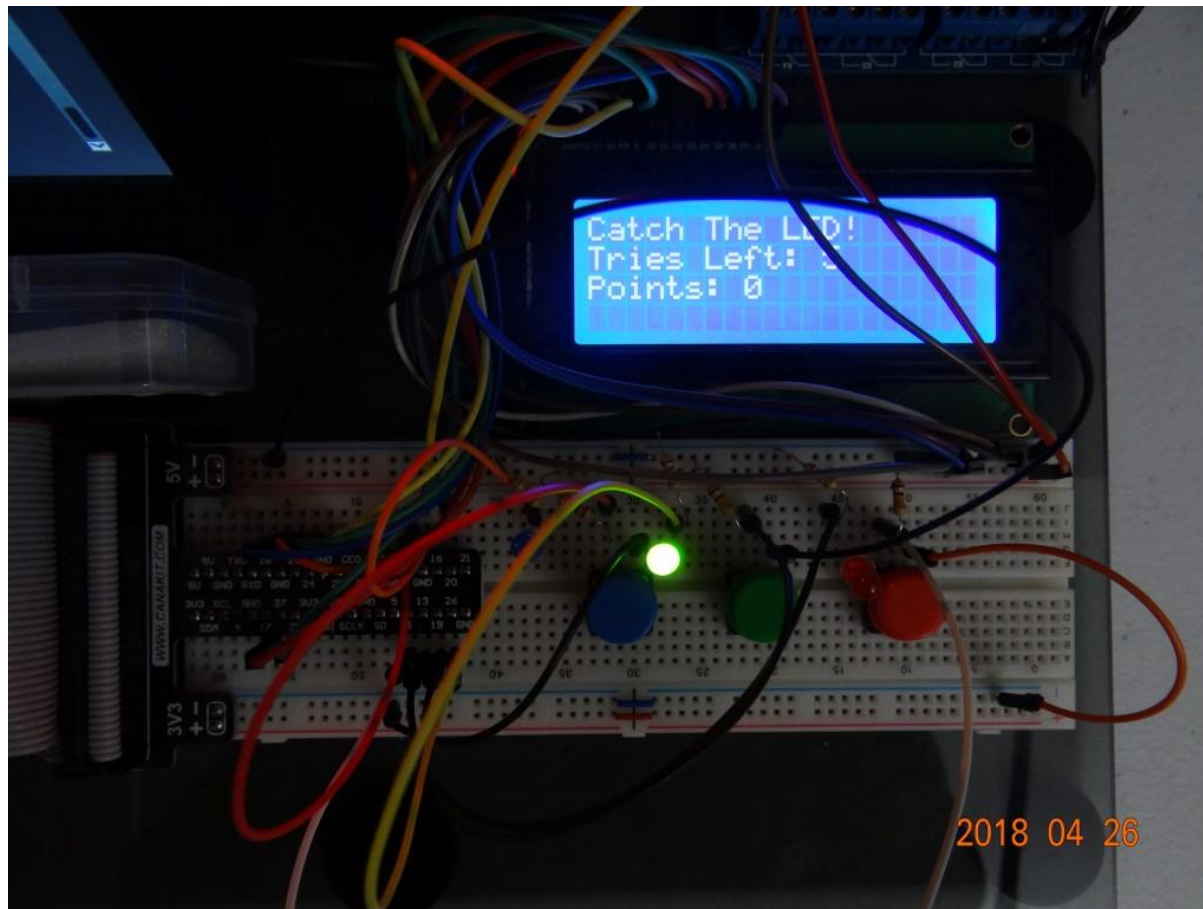
Step 15

- Extend
<https://www.michael-wessel.info/class7/step14.py>
- Put the grading and report function at the end into a function:

```
def report_score(points):  
    ... < to be written > ...
```

Final Solution

<https://www.michael-wessel.info/class7/step15.py>



The Game as a Standalone Device

- Put it in a project box

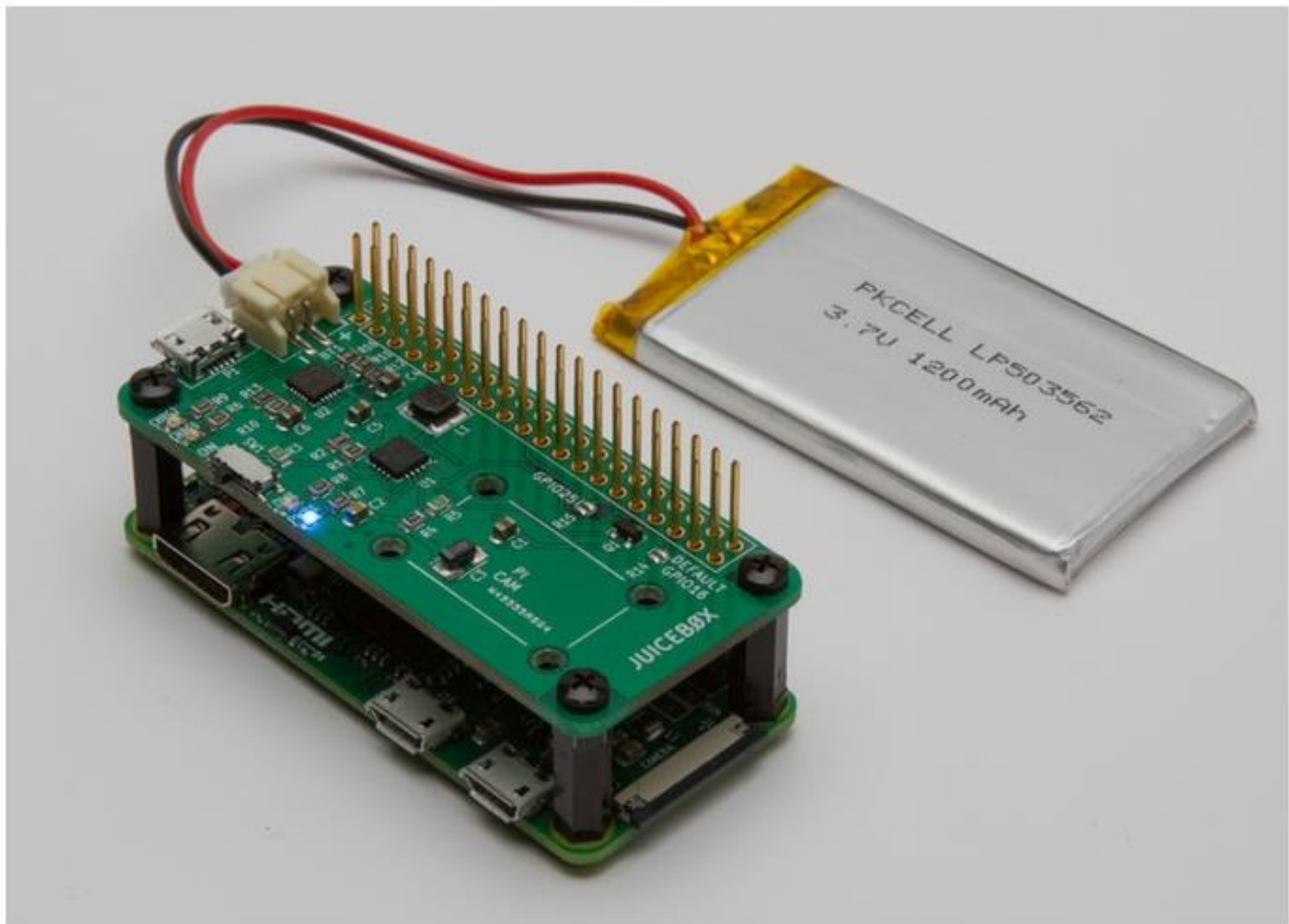


- Use a Raspberry Pi Zero !
- Configure such that the game starts automatically when turned on. E.g., use **/etc/rc.local**
- Add battery power (e.g., **JuiceBox Zero**)
- Add an on / off switch
- It would probably require some soldering though

Protoboard for RPi Zero



Option: Protoboard Soldered with Female Pin Header



Battery powered Pi Zero w/stacking header

JuiceBox Zero is the *simplest* way to properly power your Pi Zero with a Li-Ion battery.

Sound without a Monitor



**NEW! One free
RCA connector
included !**



Still Needed – Little Speaker



Thank You!

- This is the end of this class
- We hope you had fun!
- Keep on hacking and making!
 - <https://www.raspberrypi.org/magpi/issues/>
 - <https://www.raspberrypi.org/forums/>
 - <https://www.raspberrypi.org/community/>

Good bye! Hope to see you soon again!