Internet of Things
Department of Computer Engineering
Santa Clara University, California
Behnam Dezfouli, PhD

Santa Clara University
Internet of Things
RESEARCH LAB
SIOTLAB

Internet of Things

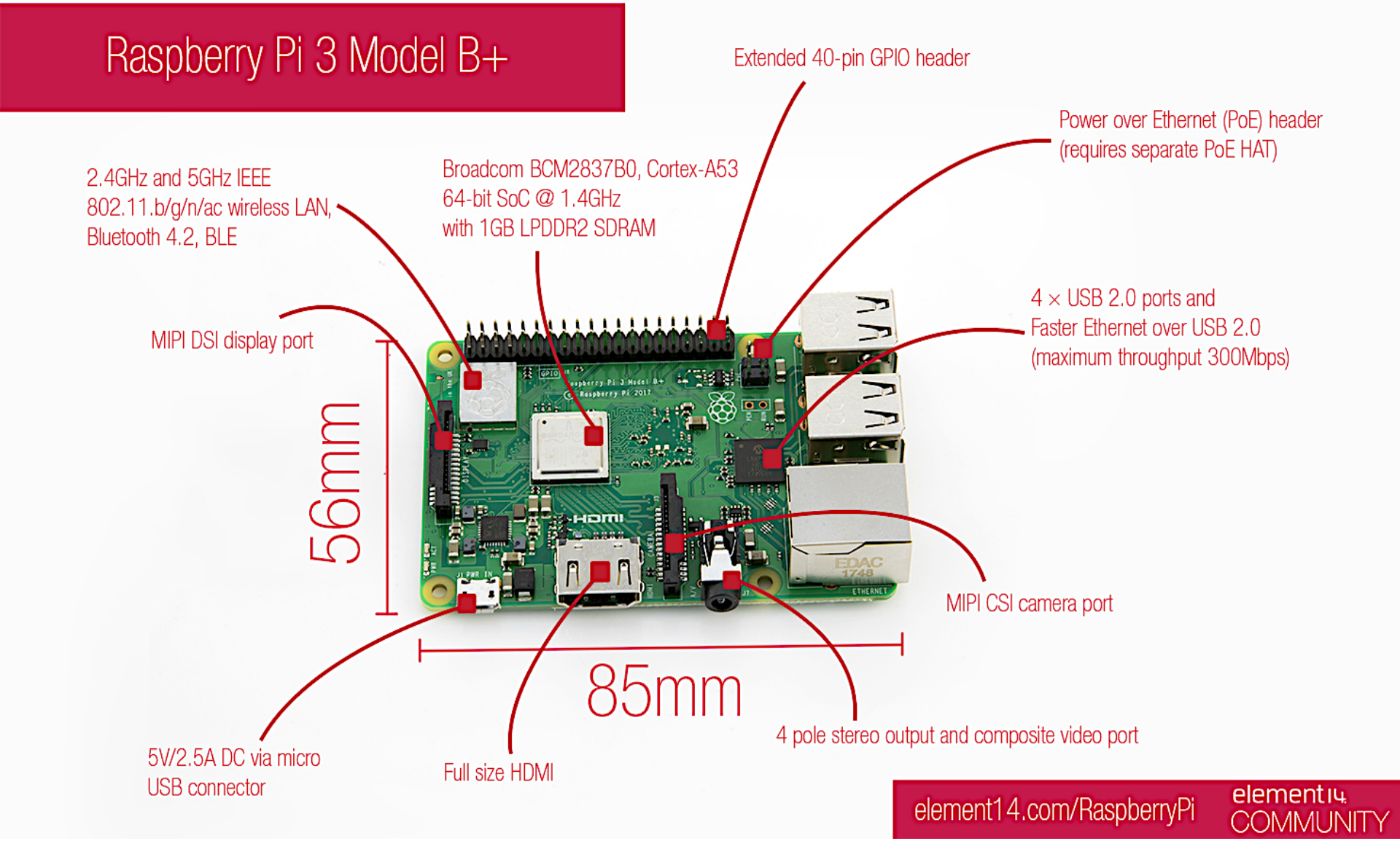
Lab 1
Setting up a Raspberry Pi

Introduction to Raspberry Pi

Setting up a Raspberry Pi

- Raspberry Pi is a credit card-sized computer originally designed for education
- Now widely used across the world both for **rapid prototyping and for industrial IoT solutions**
- It can integrate with USB, UART, SPI, I2C, and general-purpose GPIO devices to connect them to the internet with its onboard **WiFi card or Ethernet** port
- It is capable of running both **Linux** and Windows 10 IoT Edition (though we'll be using Linux exclusively in this class)
- Has an ARMv7 chipset
- It's powered by a microUSB cable and uses a micro SDCard for storage

Setting up a Raspberry Pi



Burning the SD Card

Finding the Device (the SD Card)

Finding the Device (the SDCard)

- In Linux, **block devices** are special files that refer to or represent a device (which could be anything from a hard drive to a **USB drive**)
- So naturally, there are command line tools that help you with your block devices-related work
- Once such utility is **`lsblk`**
- The first step is to see what *block devices* are already connected to the desktop, so that when we plug a new one in, we know its name

- Now, plug in the USB adapter (shown below) and run the list block devices command again:

➢ **`lsblk`**

Finding the Device (the SD Card)

- Note that a new entry has appeared in the file listing
  - **This is the name of our SDCard**

- Sometimes, operating systems will automatically **mount** the SDCard for you, trying to be helpful!

- Since we're burning the card ourselves, we don't want anything mounted, so we need to unmount any mounted partitions:

➢ **`umount /dev/<device name>*`**

  - Replace <device name> with your device name

Writing the Image

- Now, we need to install the operating system image to the SDCard
- For this, we have a disk image file, or .iso file, in a folder that is shared with the whole class
- The image file itself is over 1.7 GB, so rather than have every student make their own copy, we have a compressed (zipped) version of the file shared with the whole class
- This file is stored at **/ftp/pub/linux/raspbian/raspbian.zip**

- Run the following:

➢ **`unzip -p /ftp/pub/linux/raspbian/raspbian.zip | dd of=/dev/null bs=4M conv=fsync status=progress`**

  - In the above command, replace /dev/null with /dev/<device name> (such as sdb or mmcblk0)

➢ **`sync`**

Writing the Image

How it works?

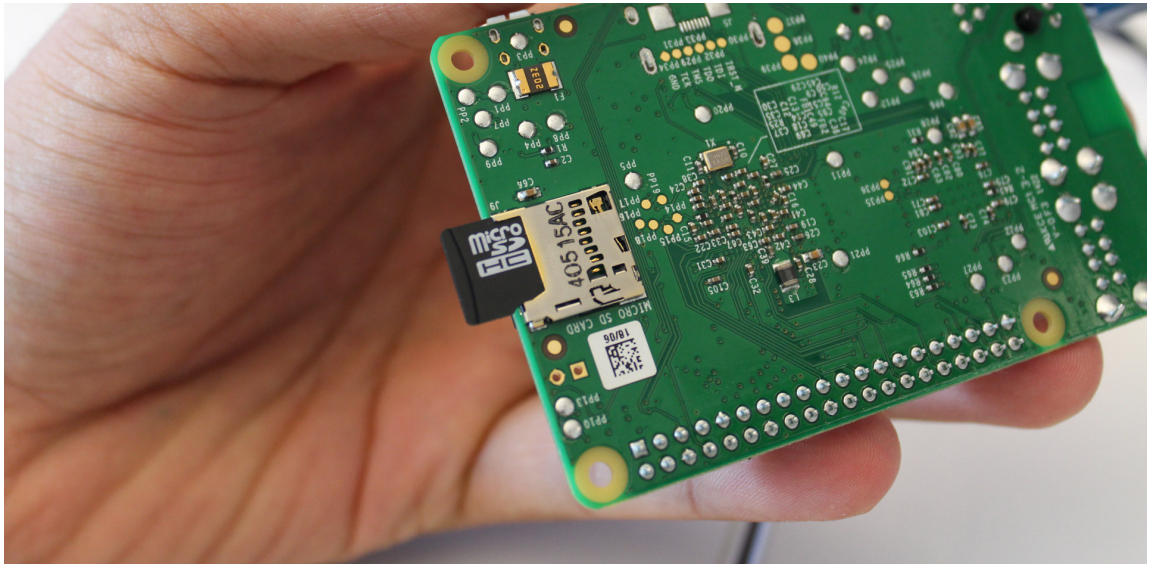
- The **first line** uses the unzip command to extract the .iso file from the zip folder we've shared with you
  - The `-p` flag tells unzip to extract the contents of the zip file to STDOUT, which we *pipe* (using "|") into the dd command
  - `dd` then writes the input file to `of`, the output file (which you specified as your SDCard's device name)
  - The block size was set to 4 MB, and `conv=fsync` tells `dd` to flush the writes to the disk at each block
  - The `status=progress` flag lets you see how much of the job has completed, as it may take several minutes

- In the **second line**, the `sync` command forces the OS to write all buffered pages to the disk to avoid unmounting the SDCard before it is ready

Writing the Image

- When the prompt returns, the image has been written to the SDCard!

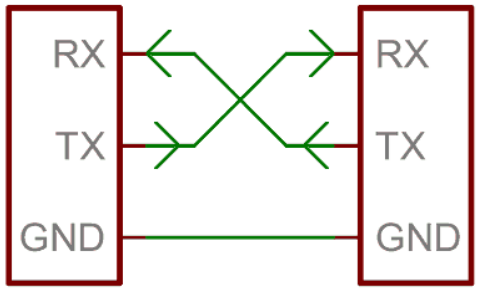
- **It's now safe to remove theSDCard and insert it into the RPi itself**



Logging Over UART

UART

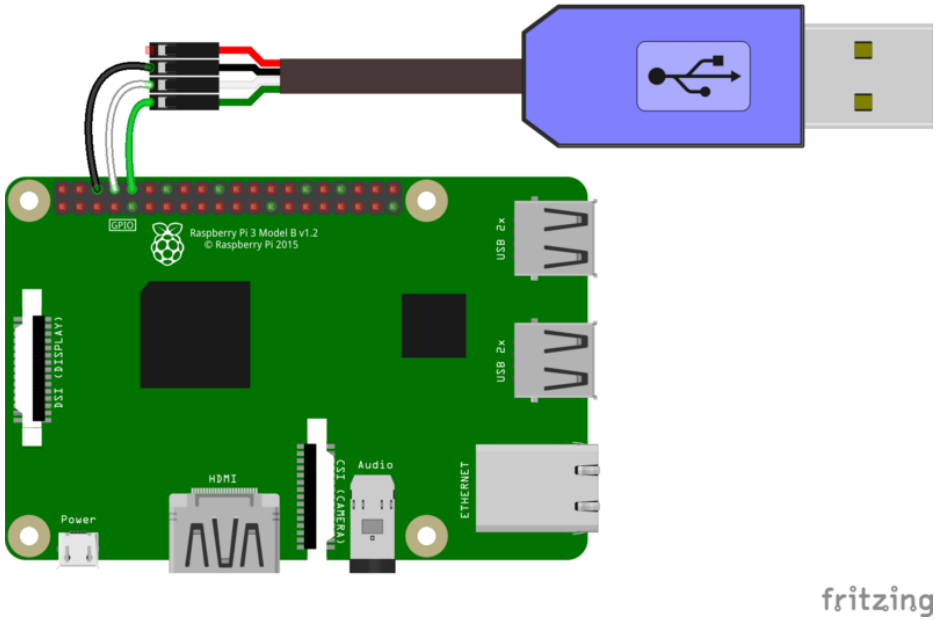
- UART stands for **Universal Asynchronous Receiver/Transmitter**
- A UART's main purpose is to transmit and receive **serial data**
- In UART communication, two UARTs communicate directly with each other
- The transmitting UART converts parallel data from a controlling device like a CPU into serial form, transmits it in serial to the receiving UART, which then converts the serial data back into parallel data for the receiving device
- **Only two wires are needed to transmit data between two UARTs**
- Data flows from the TX pin of the transmitting UART to the RX pin of the receiving UART:



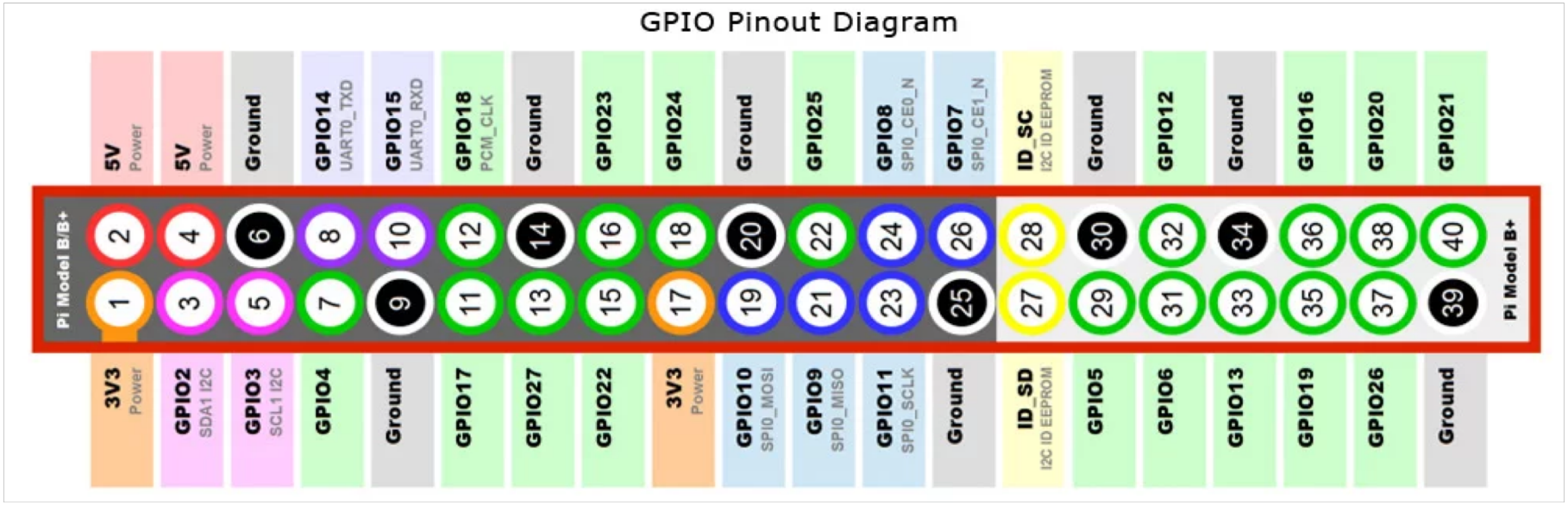
Connecting UART Cable

- Shown below is a photo of a USB to UART cable

- By plugging it in to the GPIO pins on the RPi, we are able to log in to the device and see a terminal
- The first step is to plug in the pins int GND, Pin 8, and Pin 10 as shown



Connecting UART Cable



On other words, we connect the:

- TX pin of the cable to the RX pin of the Raspberry pi
- RX pin of the cable to the TX pin of the Raspberry pi

Logging in Over UART

- Now, we need to see a list of the **TTY devices**
- A tty device gets its name from the very old abbreviation of *teletypewriter* and was originally associated only with the physical or virtual terminal connection to a Unix machine
- Over time, the name also came to mean **any serial port style device**, as terminal connections could also be created over such a connection
- Some examples of physical tty devices are serial ports, **USB-to-serial-port** converters, and some types of modems that need special processing to work properly
- tty virtual devices support virtual consoles that are used to log into a computer

Logging in Over UART

- To see the list of tty devices run the following command:

➢ `ls /dev/tty*`

- Now, **insert the UART cable into the desktop's USB port** and plug in the RPi's power cable
- Wait a couple of seconds, and now you should be able to run the **`ls`** command again to see what's changed

➢ `ls /dev/tty*`

- The **screen** utility can be used to log into the command line interface provided by the RPi:

➢ `screen </dev/ttyUSB0> 115200`

  - Replace </dev/ttyUSB0> with the new device

Logging in Over UART

- Now you should see the RPi's login presented to you
- **The default username and password are:**
  - **User name: pi**
  - **Password: raspberry**
- You should now see a prompt ready for commands
- **Note**: All the commands given for the rest of the RPi labs are to be run on the RPi over serial, not on the EEC server

Configuring the Raspberry Pi

Adding Your User

Adding Your User

- The first step is to change your username and password
- However, only the **root user** has the ability to do that, as adding and removing users from the system is a security issue
- Therefore, you need to become root:

➢ **`sudo su`**

- The above command tells the su (super user) do do something, which in this case is "run the su command”
- The su command is a shortcut that the superuser can use to log in to the system at any time
- If you are in the list of approved superusers, you have permission to run the sudo command

Adding Your User

- Since the pi user has been added to the sudo group, so you will have permission to become root

- It is really a good idea to **change the default user name** *to protect from brute force attack*

➢ **`useradd <username>`**

  - Replace *<username>* with **the user name you use for the engineering server (ECC) login** at scu so you don't forget it later

- Now that the user has been added, you need to **set a password** for it:

➢ **`passwd <username>`**

  - Again, remember to replace <username> here

Adding Your User

- Now, you need to grant sudo privileges to your new user so that you can become root again later after you delete the pi user:

➢ **`usermod -aG sudo <username>`**

- This usermod command adds you to the sudo group (-aG, or add Group), which is a list of users that are allowed to use the sudo command

- It is also important to **add a home folder** for your new user:

➢ **`cp -r /home/pi  /home/<username>`**

➢ **`chown -R <username>:<username> /home/<username>`**

Adding Your User

How it works?

- To make things simpler, we are copying *all of the files* in the pi user's home folder into a new folder for your new user
- The `-r` parameter stands for recursive and makes the cp command copy everything inside the folder instead of just the folder itself

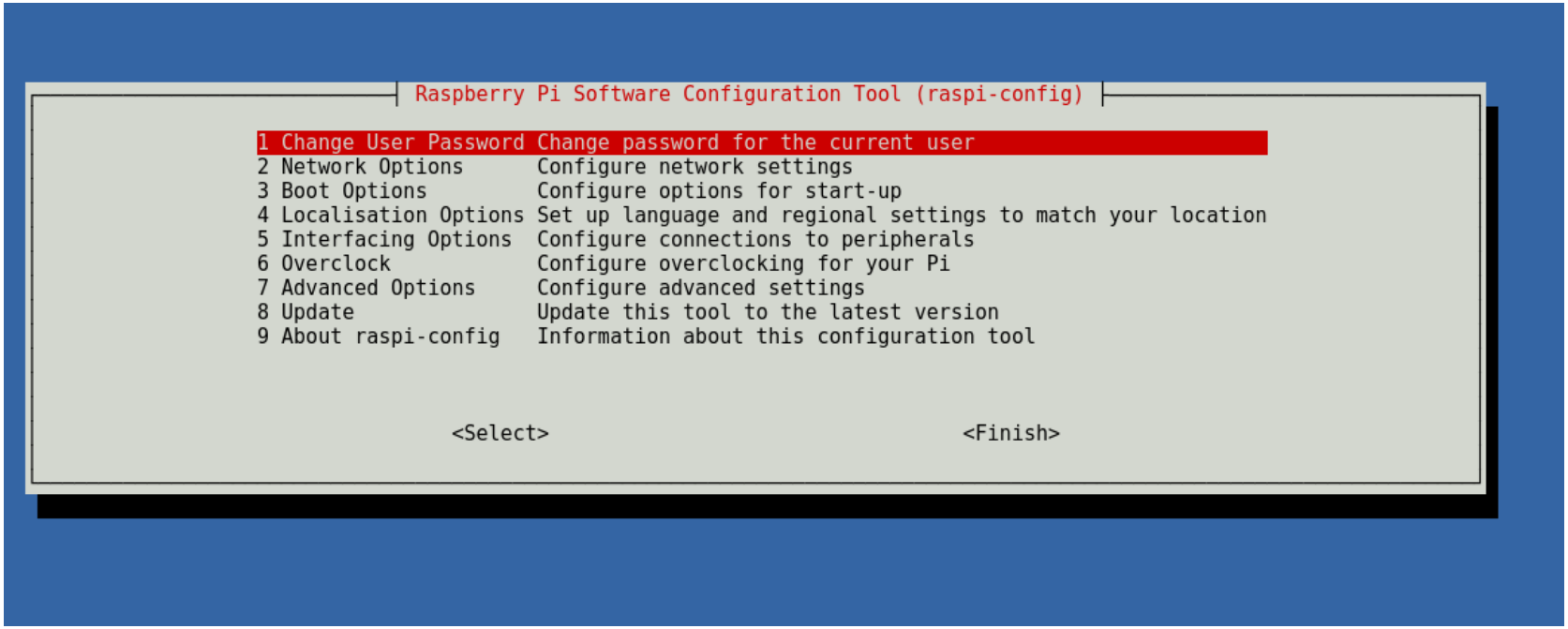
- The `chown`, or *change ownership* command, is necessary because you created the folder as root, which makes root the owner
- Since your user cannot access folders owned by others (unless they have specifically granted you access), we transferred ownership to your new user
- The `-R` parameter stands for *recursive*, changing the ownership on both the folder and everything inside to the <username> user

System Configuration

- There are several system configuration options we need to set as well. For that we will use the raspi-config utility:

➢ **`raspi-config`**

- You should see a screen that looks somewhat like the following:



System Configuration

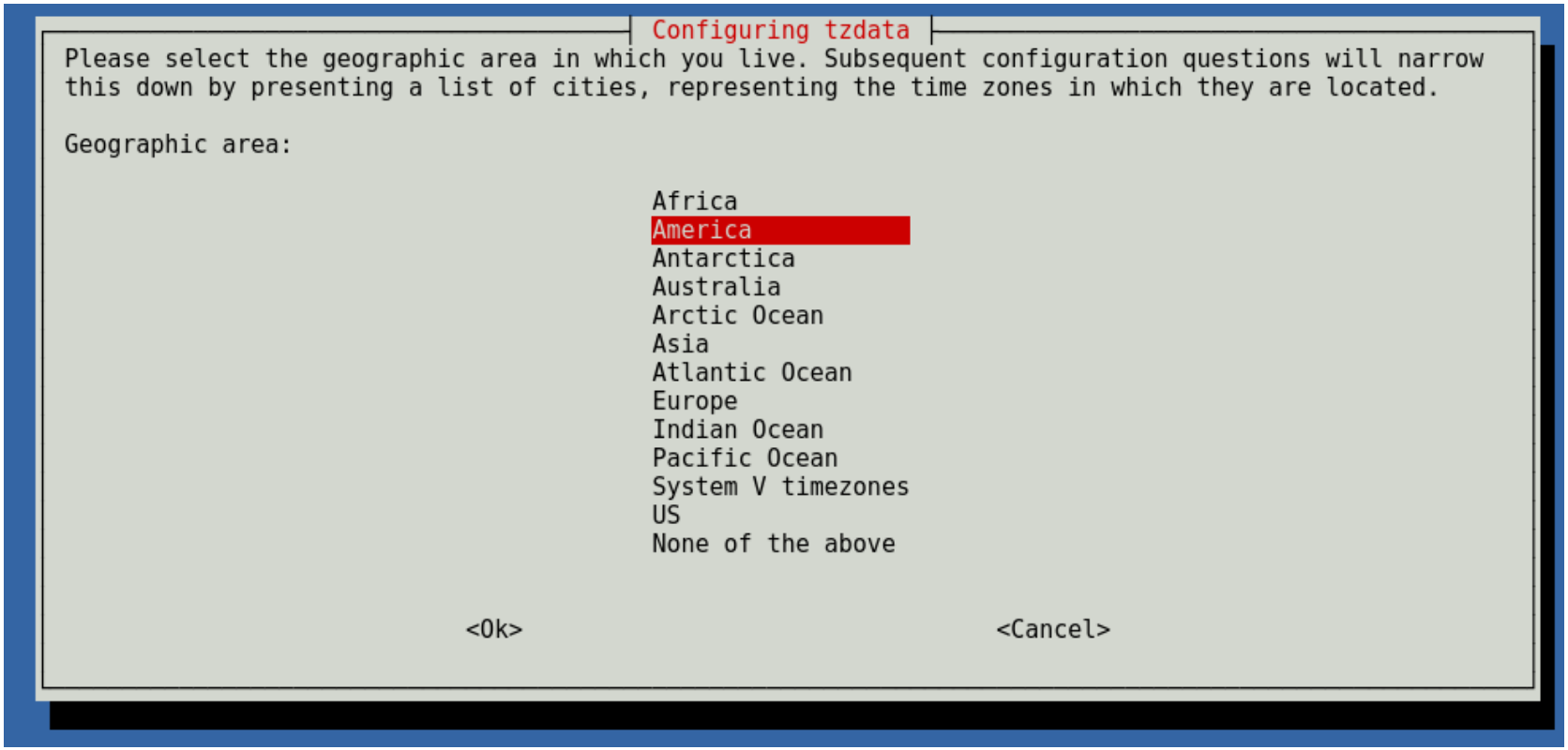
- Use the arrow keys to navigate the "Localization Options"
- **Change the locale to "en_US.UTF-8 UTF-8"**
- Do this by scrolling down to the entry and pressing the spacebar when it is highlighted
- Then press enter
- On the next page, select the UTF8 option in the same manner and hit enter

- You should see the following:

```
Generating locales (this might take a while)...
  en_US.UTF-8...
```

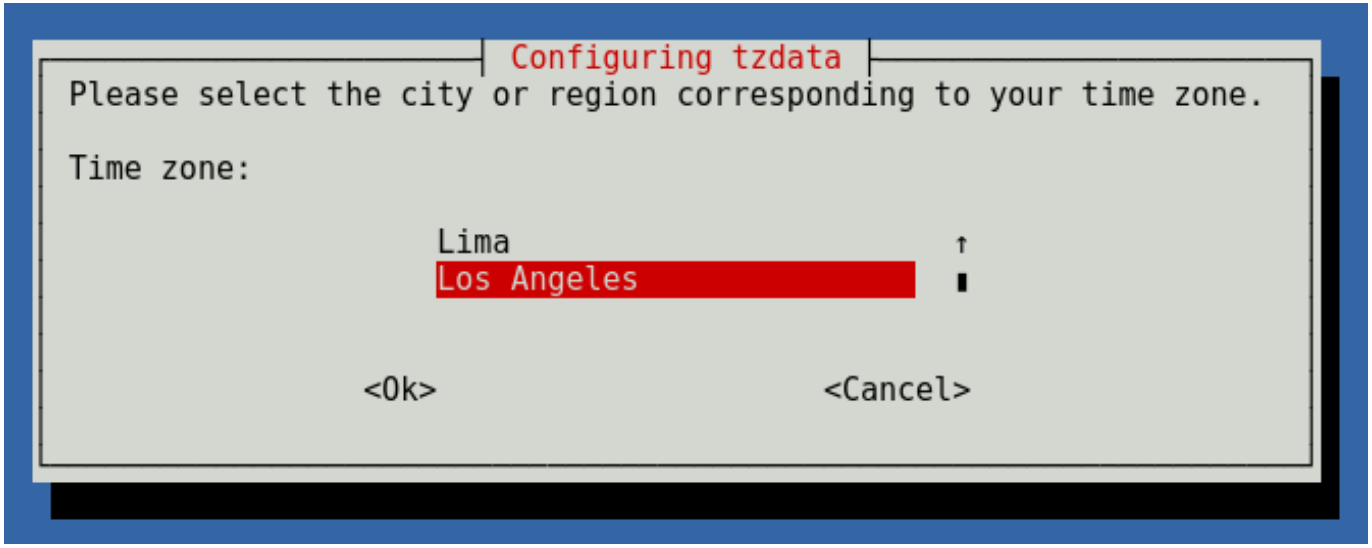
System Configuration

- Now, we need to change the timezone inside of localization options as well:
- First, choose America



System Configuration

- Then, choose Los Angeles:



- Now, you should see something similar to the following:

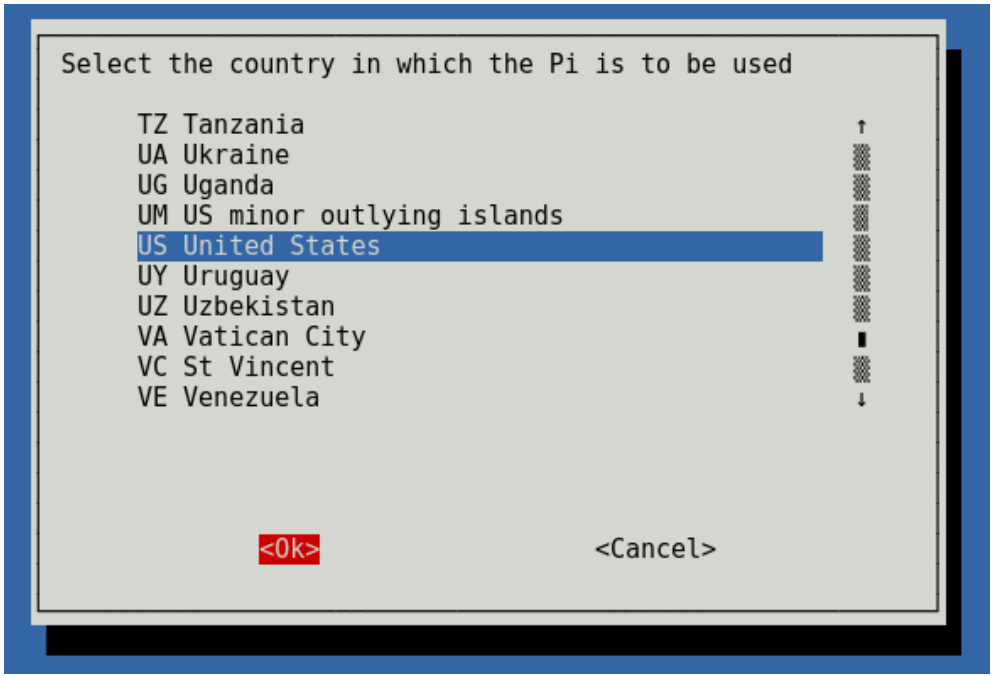
```
Current default time zone: 'America/Los_Angeles'
Local time is now:      Sat Feb 16 13:57:21 PST 2019.
Universal Time is now:  Sat Feb 16 21:57:21 UTC 2019.
```

System Configuration

- We have two more Localization settings to change, keyboard and WiFi country
- For keyboard, just select the entry and hit enter, it should automatically run, saying:

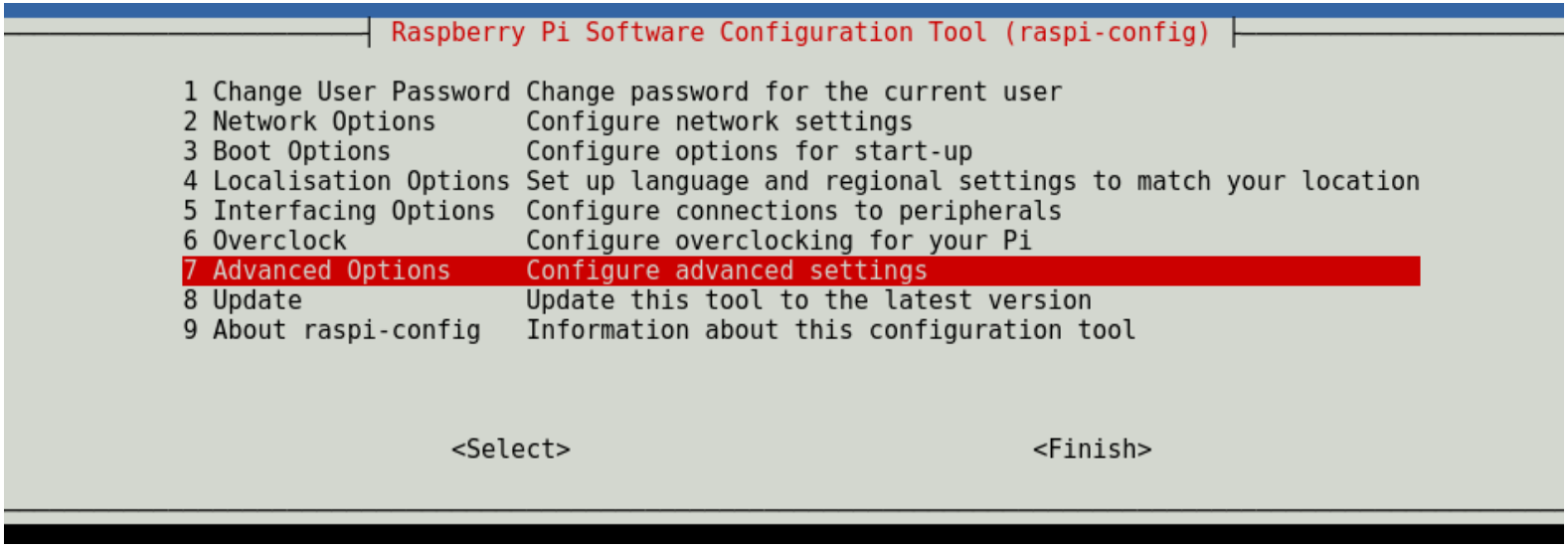
```
Reloading keymap. This may take a short while
```

- Lastly, change the WiFi country:



System Configuration

- By default, the size of the Raspbian root file system is **2GB**
- **If you have an SD card with more capacity, the portion of your disk space will be unused**
- So let's ***expand the filesystem!***
- This option enables you to expand your Raspbian installation to fill out the rest of the SD card, giving you more space to use for files



- Then choose **Expand File System**

Network Configuration

Lastly, let's set the **Network options**

- First, change your hostname to **u-iot-coen-rpi-<username>**

- Finally, we're ready to reboot
- Tell raspi-config to Finish, and it will ask you to reboot
- When you see the login prompt, your RPi is done rebooting

Cleaning Up

- Now, you need to log in to your RPi with the new user you just created
- Next, log in to the root account again using the sudo command from above:

➢ **`sudo su`**

- Now, we need to delete the pi user
- To prevent the pi user from initiating new processes when the system boots up:
  - Edit the file `/etc/systemd/system/autologin@.service`
  - Find and *comment* (#) the line
    - `#ExecStart=-/sbin/agetty --autologin pi --noclear %I $TERM`

- Then reboot the system

Cleaning Up

- Then we need to kill all the processes that belong to user **pi** using

➢ **`pkill -u pi`**

- Then run the following commands to delete the user and its home folder

➢ **`userdel pi`**

➢ **`rm -r /home/pi`**

- Congratulations, you have successfully set up your Raspberry Pi!

Student Work

- Task 1-1
  - How can we see the list of processes and their CPU and memory utilization?
  - How many processor cores are available on the board?
  - How much memory is currently being utilized?