

Station Automation

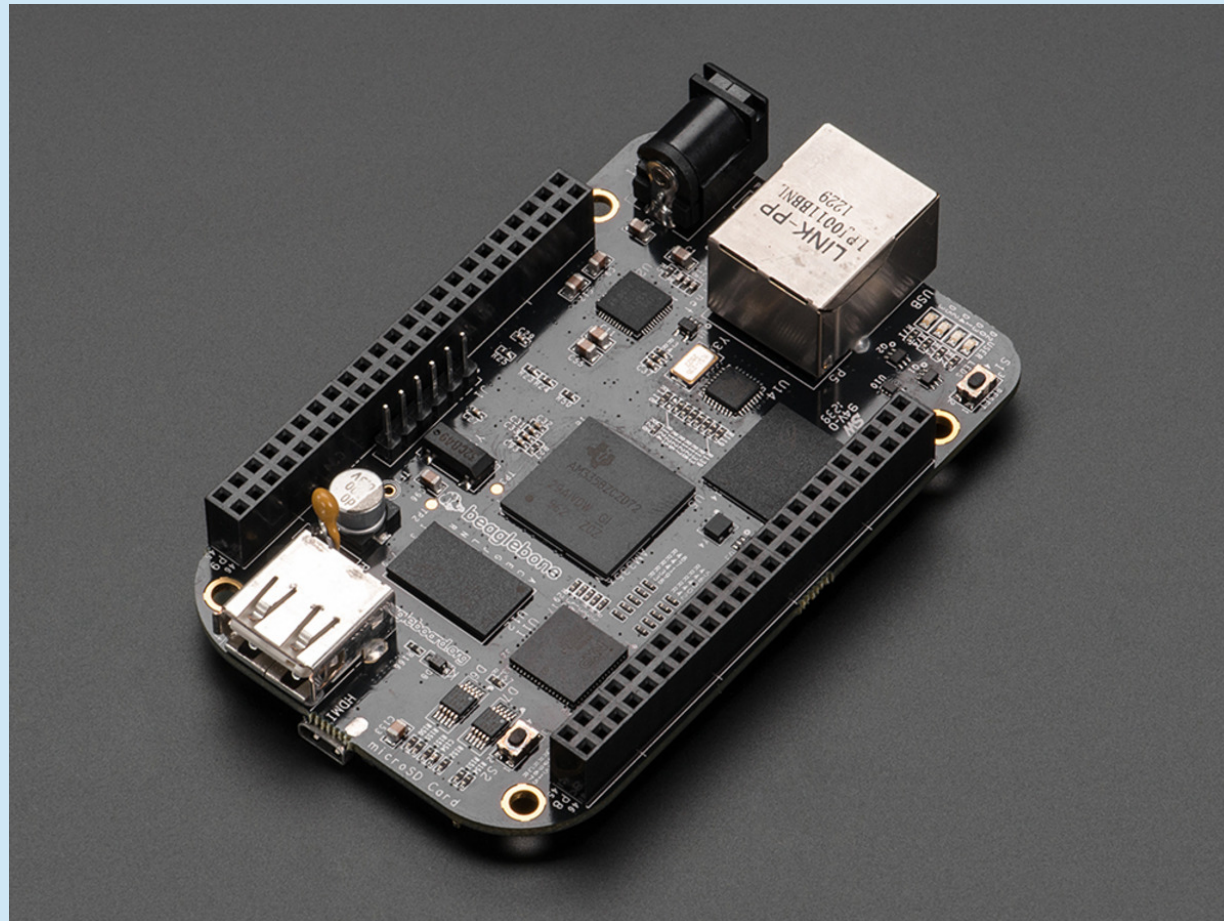
--W3SZ

The screenshot displays a complex software interface for station automation, featuring several key components:

- Signal Monitoring:** Multiple waterfall plots and spectrograms showing frequency activity across various bands, including 50.280 MHz and 144.140 MHz.
- Frequency Management:** A central window showing a list of frequencies with columns for UTC, dB, I, Freq, and Message. It includes controls for band activity and message transmission.
- Logging and Call History:** A window titled "General Logging - OK, M3.03h" displaying a table of call logs with columns for date, time, call sign, frequency, mode, and name.
- System Status:** A "WS3Z Multi-SDR Controller" window showing various system parameters and controls.
- Resource Monitoring:** A "Task Manager" window showing system performance metrics such as CPU usage (67%), memory usage (4.51 GB), and disk activity.
- Time and Date:** A digital clock showing 1:08:20 PM on 2017 Feb 24.

YYYY-MM-DD HH:MM	Call	Freq	Mode	Snt	Rcv	Pfx	Name	Comment
2017-01-22 06:52	WS2N	50276.50	MSK144	59	59	K	David	
2017-01-22 07:03	KESRV	50288.50	MSK144	59	59	K	Marshall	
2017-01-22 07:21	K5QE	50266.50	MSK144	59	59	K	Jim	
2017-01-22 07:29	K0RA	50286.50	MSK144	59	59	K	Larry	
2017-01-22 08:19	K0TPP	50281.50	MSK144	59	59	K	Peter	
2017-01-22 08:50	V3ELE	144173.50	J765	59	59	VE	Peter	
2017-01-22 09:27	V3ELE	129607...	J765	+00	59	VE	Peter	
2017-01-22 09:57	V3ELE	432071.50	J765	+00	59	VE	Peter	
2017-01-22 10:03	V3ELE	222071.50	J765	+00	59	VE	Peter	
2017-01-22 12:00	VEZDS	144128.70	J765	59	-15	VE	Dany	
2017-01-22 12:25	N8RA	50286.50	MSK144	59	-15	K	John	
2017-01-22 12:32	K1SIX	50281.50	MSK144	59	-15	K	John	
2017-01-22 18:45	WA3GFZ	230410...	J765	59	-15	K	Jeff	
2017-02-16 13:24	VE3VEY	50281.50	MSK144	+04	+06	VE	Larry	
2017-02-16 13:35	W3OFD	50281.50	MSK144	+08	+10	K	Larry	
2017-02-16 13:46	K8RTBW	50281.50	MSK144	+07	+02	K	Larry	
2017-02-16 13:46	K0TPP	50281.50	MSK144	+02	+00	K	Larry	
2017-02-16 13:46	K0TPP	50281.50	MSK144	+07	+02	K	Larry	
2017-02-16 14:01	K0TPP	50281.50	MSK144	+7	+2	K	Larry	
2017-02-16 14:10	W8JZW	50281.50	MSK144	+06	+06	K	Larry	
2017-02-16 14:14	N3ALN	50281.50	MSK144	+10	+12	K	Larry	
2017-02-16 14:26	WA1EAZ	50281.50	MSK144	+01	-01	K	John	

IF/Transverter Ethernet Bandswitching



IF/Transverter Bandswitching Ethernet

- NONE of these logging programs directly supports bandswitching of transverter devices, etc. by Ethernet
 - N1MM, VHFLog, RoverLog, WriteLog, VQLog, DXLabs
- Frequency and Antenna (band) data is provided by N1MM via Ethernet UDP Broadcast “Radio” data
 - Broadcast at least every 10 seconds, and every time radio frequency or mode changes
- User can write software and build hardware or use pre-existing hardware to harness these Ethernet UDP signals to control IF radio / transverter bandswitching

IF/Transverter Bandswitching Ethernet N1MM UDP Broadcast

- 1) Band switches in N1MM -->
 - 2) BeagleBone Black switches bands -->
 - 3) SainSmart 16-relay board or alternative device switches bands -->
 - 4) RF relay switches bands
- Writing some code is required

IF/Transverter Bandswitching Ethernet N1MM UDP Broadcast

The screenshot shows the N1MM Logger application window. The 'Config' menu is open, displaying various settings options. A blue arrow points to the 'Config' menu header. The interface includes a menu bar (File, Edit, View, Tools, Config, Window, Help), a frequency band selection table, a heading display, and a log window.

CW	PH
6m	6m
2m	2m
1.25m	1.25m
70cm	70cm
33cm	33cm
23cm	23cm
13cm	13cm
9cm	9cm
6cm	6cm
3cm	3cm
1cm	1cm
Light	Light

COM16 timeout. Count = 98

Heading
Call his

Telnet
0

- Configure Ports, Mode Control, Audio, Other...
- Change Your Station Data...
- Use Local Control (Set up Hardware, Function Keys, Digital Modes, Winkey, Mode Control)
- Enter Sends Message (ESM mode) Ctrl+M
- Spot All S&P QSO's
- QSYing Wipes the Call & Spots QSO in Bandmap (S&P)
- Grab Focus From Other Apps When Radio is Tuned
- Do Not Automatically Switch to Run on CQ Frequency
- Show Non-Workable Spots and Dupes in Bandmap
- Reset RX Freq to TX when QSO is Logged (Run & Split)
- Sub Receiver Always On Ctrl+Alt+D
- CQ Repeat Alt+R
- Set CQ Repeat Time... Ctrl+R
- CW / PH AutoSend Threshold...
- Enable Call History Lookup
- Change CW/SSB/Digital Function Key Definitions
- Change Band Plan
- Manage Skins, Colors and Fonts...
- Change Operator Callsign Stored in Log Ctrl+O
- Change Exchange Abbreviations
- SO2R
- WAE
- Clear *.ini File Settings
- SO2V Dual Receive...

IF/Transverter Bandswitching Ethernet N1MM UDP Broadcast

Configurer

Hardware Function Keys Digital Modes Other Winkey Mode Control Antennas Score Reporting **Broadcast Data** Audio

Select the type of data you wish to broadcast, and the the IP Address(es) and port(s) for the receiver(s) of the data. Use 127.0.0.1 for the local machine. Use 12060 as the port unless the receiving application requires a different port. 255 in the low order octet will broadcast to your current subnet.

Type of data	IP Addr:Port	IP Addr:Port...
<input checked="" type="checkbox"/> Application Info	127.0.0.1:12060	
<input checked="" type="checkbox"/> Radio	192.168.10.8:13063	
<input checked="" type="checkbox"/> Contact <input type="checkbox"/> All QSOS	127.0.0.1:12060	
<input type="checkbox"/> Spots	127.0.0.1:13063	
Rotor	127.0.0.1:12040	
<input type="checkbox"/> Score	127.0.0.1:12060	

Sets the IP Address and port that an external program can connect to N1MM+ via TCP Port.

Enable	IP Address	TCP Port
<input checked="" type="checkbox"/> Enable	127.0.0.1	52001

OK Cancel Help

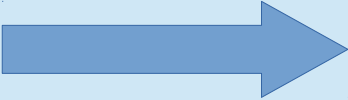
IF/Transverter Bandswitching Ethernet N1MM UDP Broadcast

- 1) Get UDP Data via Ethernet from N1MM
- 2) Parse UDP data to extract band information
- 3) Use Band information to switch bands

IF/Transverter Bandswitching Ethernet N1MM UDP Broadcast

- UDP Data looks like this:

```
<?xml version="1.0" encoding="utf-8"?>  
<RadioInfo>  
  <StationName>INTEL-I7</StationName>  
  <RadioNr>1</RadioNr>  
  <Freq>5012500</Freq>  
  <TXFreq>5012500</TXFreq>  
  <Mode>USB</Mode>  
  <OpCall>NN3Q</OpCall>  
  <IsRunning>False</IsRunning>  
  <FocusEntry>134626</FocusEntry>  
  <Antenna>-1</Antenna>  
  <Rotors>-1</Rotors>  
  <FocusRadioNr>1</FocusRadioNr>  
  <IsStereo>False</IsStereo>  
  <ActiveRadioNr>1</ActiveRadioNr>  
</RadioInfo>
```



IF/Transverter Bandswitching Ethernet N1MM UDP Broadcast

- Could use either MCU or SBC
- Lets use BeagleBone Black SBC to demonstrate its use
- BeagleBone Black comes with python 2.7. Lets use python because:
 - Python syntax is easy to learn
 - Python has an extensive library support
 - Python has plenty of examples and information on the web
 - Python is a good language to know!



Python 2.7 References

- To start python interpreter, type “python”
- To run myprogram.py, type “python myprogram.py”
- Info on setting up GPIO python library:

<https://learn.adafruit.com/setting-up-io-python-library-on-beaglebone-black/overview>

Introduction to BeagleBone Black

Getting Started

- Connect small B-type USB socket on BBB to A-type USB socket on computer
- USB-Network adapter on BBB will set up USB network connection 192.168.7.2 or 192.168.7.1 for BBB
- Browse to: <http://beagleboard.org/getting-started>
- On above web page, go to “Step 3” and click on <http://beaglebone.local> or <http://192.168.7.2>
- On web page that results, scroll down and click on “[Cloud9 IDE](#)” and web portal on BBB will open

Start your Beagle

Step 1:
Power and boot

Step 2:
Enable a network connection

Step 3:
Browse to web server on Beagle

Troubleshooting

Update to latest software

Other software options

Hardware documentation

Books

Beagles are tiny computers with the capability of modern systems, without the bulk, expense, or noise. Read the step-by-step getting started tutorial below to begin developing with your Beagle in minutes.

For user supplied tips on getting started, visit the eLinux (or other) community wiki pages:

- [BeagleBoard](#)
- [BeagleBoard-xM](#)
- [BeagleBoard-X15](#)
- [BeagleBone](#)
- [BeagleBone Black](#)
- [BeagleBone Black Wireless](#)
- [BeagleBone Blue](#)
- [SeeedStudio BeagleBone Green](#)
- [SeeedStudio BeagleBone Green Wireless](#)
- [SanCloud BeagleBone Enhanced](#)
- [Neuromeka BeagleBone Air](#)

If any step fails, it is recommended to update to the *latest software image* to use the instructions below.

Step 1 Power and boot

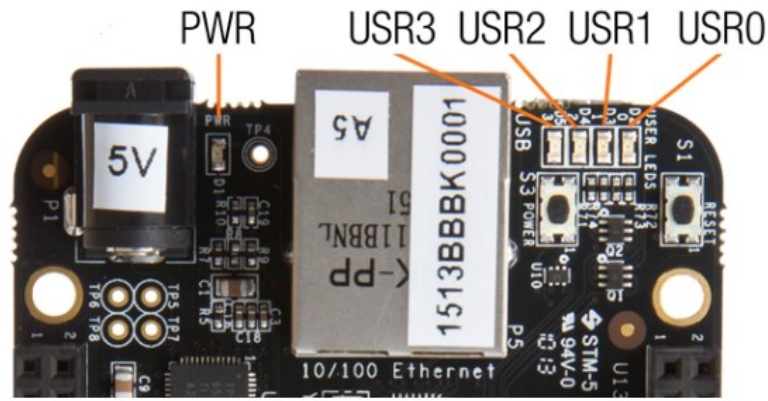
Most Beagles include a USB cable, providing a convenient way to provide both power to your Beagle and connectivity to your computer. If you provide your own, ensure it is of good quality. You'll connect the "type-B" plug of the USB cable to your Beagle and the "type-A" plug to your computer. *Note that BeagleBoard-X15 must always be powered instead by a 12V adapter with a barrel jack.*

Alternatively, for Beagles other than BeagleBoard-X15 and BeagleBone Blue that require 12V, you can utilize a 5V adapter connected to the barrel jack.

If your Beagle was provided with an [SD \(microSD\) card](#), make sure it is inserted ahead of providing power. Most Beagles include programmed on-board flash and therefore do not require an SD card to be inserted.

You'll see the power (PWR or ON) LED lit steadily. Within a minute or so, you should see the other LEDs blinking in their default configurations.

- USR0 is typically configured at boot to blink in a heartbeat pattern
- USR1 is typically configured at boot to light during SD (microSD) card accesses
- USR2 is typically configured at boot to light during CPU activity
- USR3 is typically configured at boot to light during eMMC accesses
- WIFI is typically configured at boot to light with WiFi network association (*BeagleBone Blue only*)



Linux

[mkudevrule.sh](#)

Driver installation isn't required, but you might find a few udev rules helpful.

Note: Additional FTDI USB to serial/JTAG information and drivers are available from www.ftdichip.com/Drivers/VCP.htm.

Note: Additional USB to virtual Ethernet information and drivers are available from www.linux-usb.org/gadget/ and joshuawise.com/horndis.

Step 3 Browse to your Beagle

Using either [Chrome](#) or [Firefox](#) (Internet Explorer will **NOT** work), browse to the web server running on your board. It will load a presentation showing you the capabilities of the board. Use the arrow keys on your keyboard to navigate the presentation.

- ▶ Click here to launch: <http://192.168.7.2>
Older software images require you to EJECT the BEAGLE_BOARD to load the latest software image, that step is no longer required.

The screenshot shows a web browser window with the address bar set to 192.168.7.2. The page displays the BeagleBoard.org logo and a green notification banner indicating a successful connection to a BeagleBone Black board. The main content area is titled 'BeagleBone: open-hardware expandable computer' and includes a grid of images showcasing various projects and hardware components. A left-hand navigation bar is visible on the page.

Step 1:
Power and boot

Step 2:
Enable a network connection

Step 3:
Browse to web server on
Beagle

Troubleshooting

Update to latest software

Other software options

Hardware documentation

Books

BeagleBone 101

Software

- Update image
- Node-RED
- Cloud9 IDE

Hardware

- Headers
- Capes

BoneScript

Functions

- getPlatform()
- pinMode()
- getPinMode()
- digitalWrite()
- digitalRead()
- shiftOut()
- analogWrite()
- analogRead()
- attachInterrupt()
- detachInterrupt()
- readTextFile()
- writeTextFile()

JavaScript

- console()
- setTimeout()
- clearTimeout()
- setInterval()
- clearInterval()
- typeof operator

Libraries

- require()

Demos

- Blink on-board LED
- Blink external LED
- Push button
- Potentiometer
- Joystick
- Ultrasonic sensor
- PIR motion sensor
- Accelerometer
- Temperature and pressure

Cape demos

- Bacon Cape

Grove demos

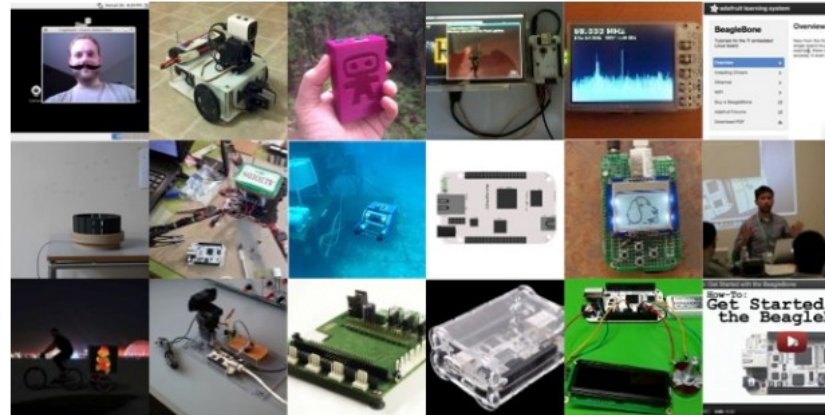
- Touch
- LCD RGB Backlight

 **Your board is connected!**
 BeagleBoard.org BeagleBone Black rev 00C0 S/N 2916BBBK148A running BoneScript 0.6.1 at beaglebone.local

BeagleBone: open-hardware expandable computer

Artist-tested, engineer approved

The left-hand navigation bar will help you explore your board and learn how to program it.



Latest ARM open source focused on easy hardware experimentation

- Ships ready to use
 - Debian Linux distribution with C++, Perl, Python, ...
 - Linux drivers support countless USB peripherals
 - Interactive tutorial to start learning about capabilities
- Open source means options
 - Texas Instruments releases: Android, Linux, StarterWare (no OS)
 - Linux: MachineKit, Debian, Fedora, Ubuntu, ArchLinux, Gentoo, Sabayon, Chromium, BeagleSNES, Asterisk, The Deck, BeagleMNT, Angstrom Distribution, Buildroot, Erlang
 - Other: QNX, FreeBSD, Minix, RTEMS, Windows Embedded, RISC OS
 - [Projects page](#)
- SD card images like get-out-of-jail-free card



```
xzcat XXX.img.xz | sudo dd of=/dev/sdX
```

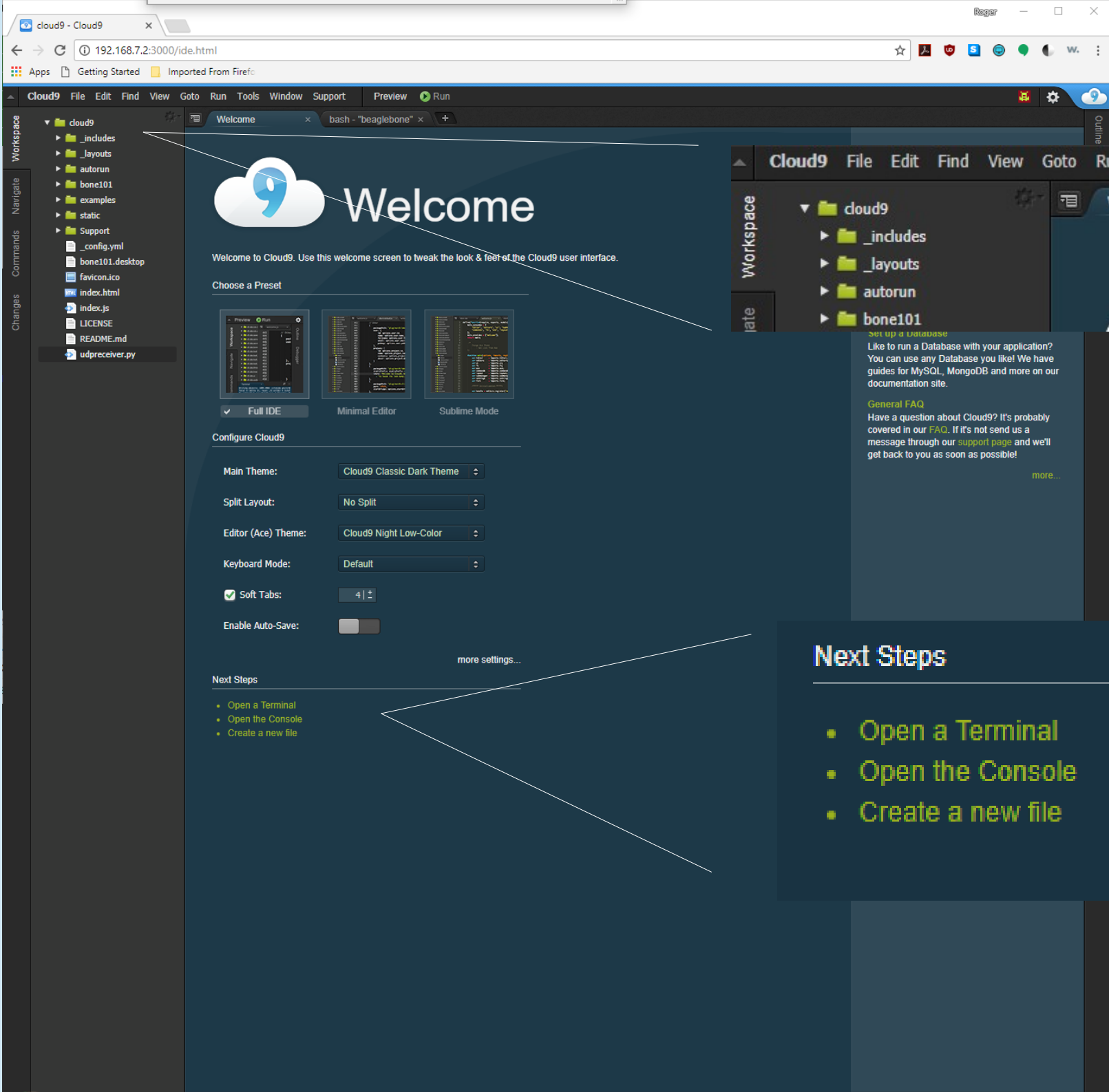
- Can be used just as easily for backups
- Board can be booted from SD using device ROM, so you can't "brick" it
- 7-zip and Ubuntu Win32DiskImager enable programming cards from Windows



Update board with latest software

There are multiple ways to run initial software on your board, but it is likely that the simplest way to get an update is to create an exact replica of a bootable microSD card and boot off of it. The BeagleBone Black Rev C has 4GB of eMMC storage that can be initialized by a program booted off of a microSD card. If you want to update to the latest software image for your board, this is a way to do that.

See [updates](#) for the step-by-step guide.



Welcome

Welcome to Cloud9. Use this welcome screen to tweak the look & feel of the Cloud9 user interface.

Choose a Preset



Full IDE Minimal Editor Sublime Mode

Configure Cloud9

Main Theme:

Split Layout:

Editor (Ace) Theme:

Keyboard Mode:

Soft Tabs:

Enable Auto-Save:

[more settings...](#)

Next Steps

- [Open a Terminal](#)
- [Open the Console](#)
- [Create a new file](#)

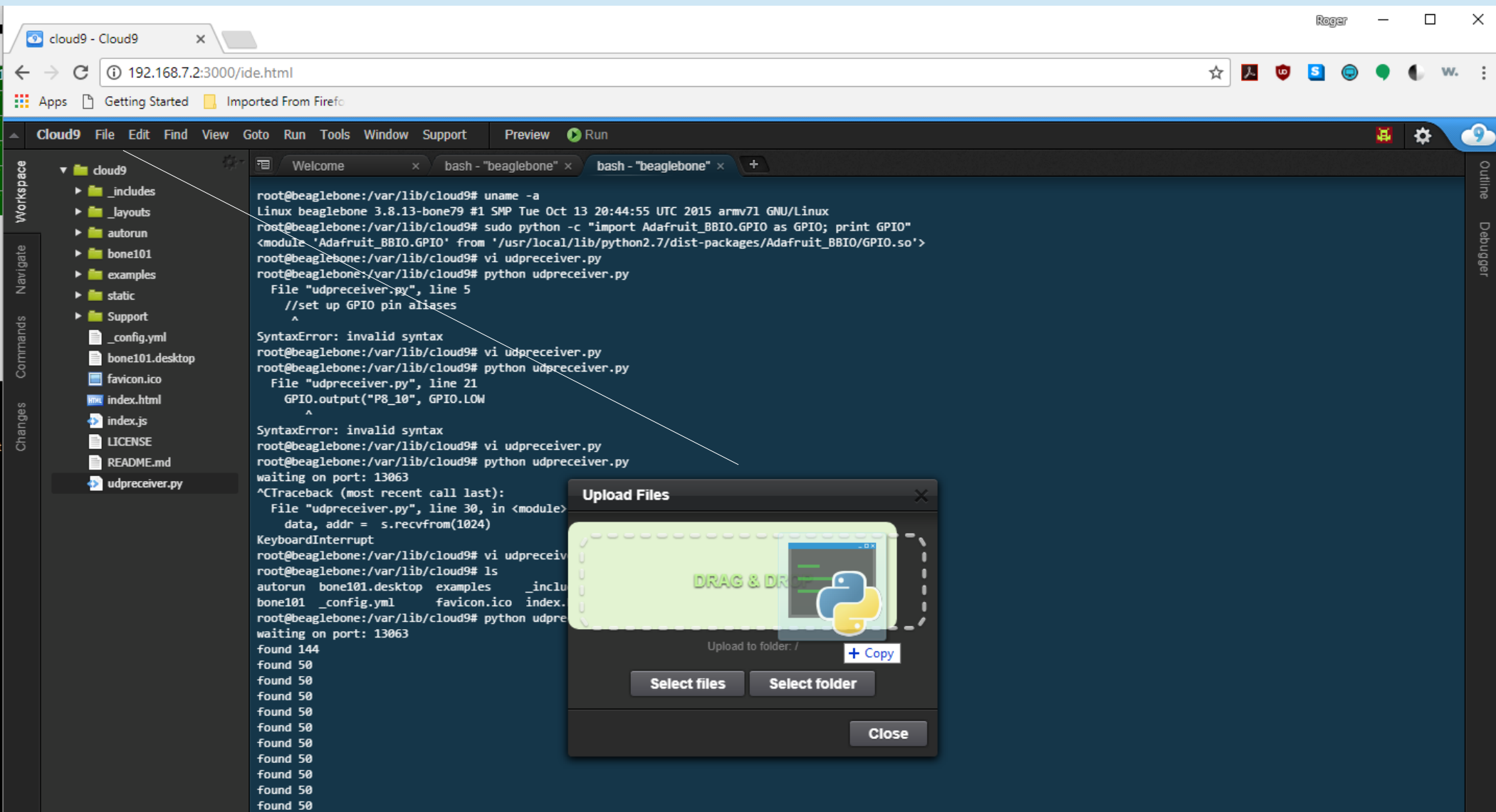
Next Steps

- [Open a Terminal](#)
- [Open the Console](#)
- [Create a new file](#)

Set up a Database
Like to run a Database with your application? You can use any Database you like! We have guides for MySQL, MongoDB and more on our documentation site.

General FAQ
Have a question about Cloud9? It's probably covered in our [FAQ](#). If it's not send us a message through our [support page](#) and we'll get back to you as soon as possible!

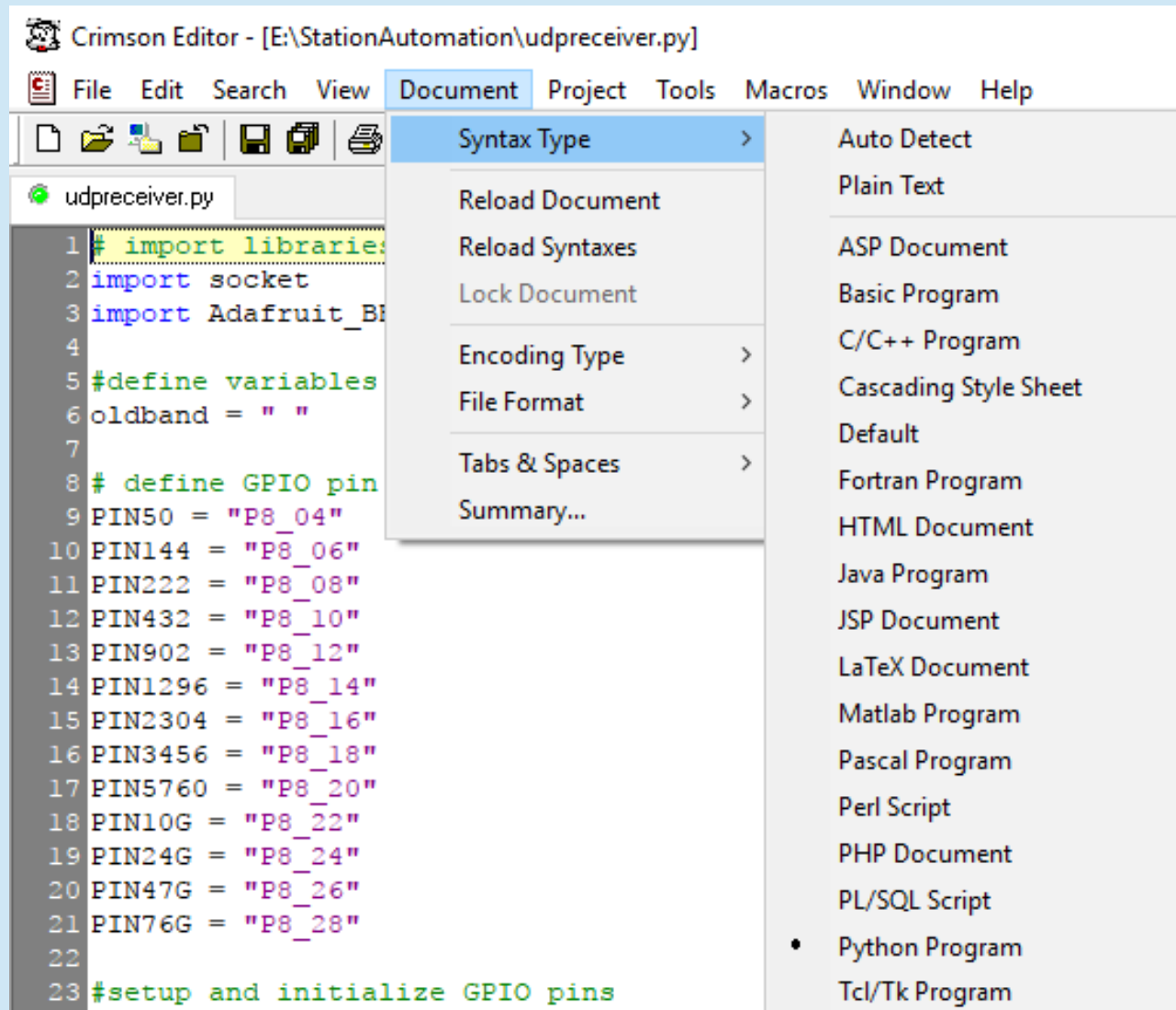
[more...](#)



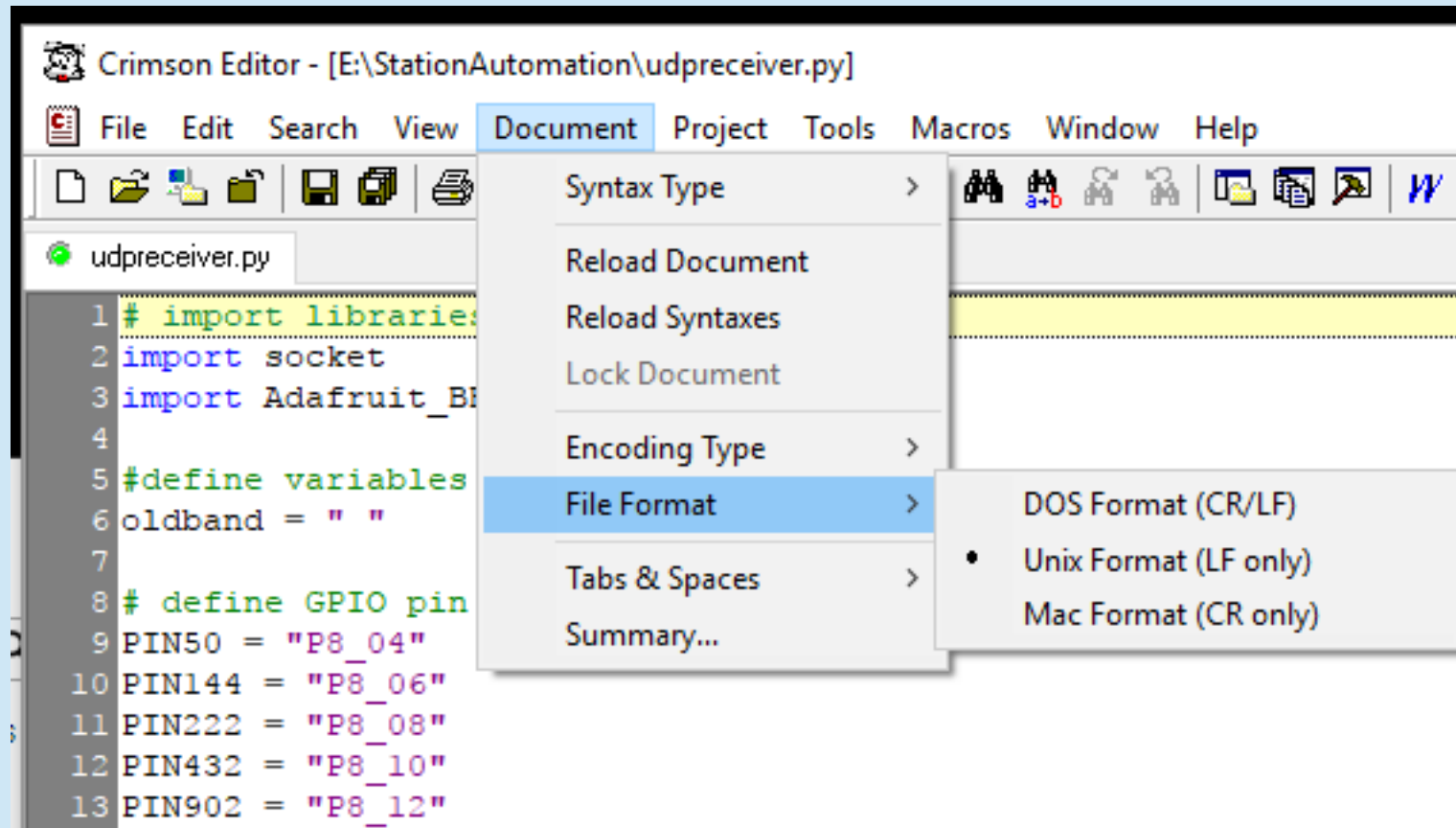
Use CRIMSON Editor in Windows to create, edit programs then drag and drop to BeagleBone Black!

Download Link for Crimson Editor is:
<https://sourceforge.net/projects/emeraldeditor/>

Crimson Editor – Free!



Crimson Editor – Free!



The image shows a terminal window on a BeagleBone Black (BBB) running a Python script named `udpreceiver.py`. The script is being edited in the `nano` text editor. The code defines GPIO pin aliases, sets up and initializes the GPIO pins, and starts a loop to receive UDP packets from a NIMM radio. The script also includes logic to parse incoming data and switch the band based on the received data.

```
GNU nano 2.2.6 File: udpreceiver.py
import libraries
import socket
import Adafruit_BBIO.GPIO as GPIO

# define variables
oldband = ""

# define GPIO pin aliases
PIN50 = "P8_24"
PIN144 = "P8_26"
PIN222 = "P8_28"
PIN432 = "P8_10"
PIN902 = "P8_12"
PIN1296 = "P8_14"
PIN2304 = "P8_16"
PIN3456 = "P8_18"
PIN5760 = "P8_20"
PIN1080 = "P8_22"
PIN246 = "P8_30"
PIN476 = "P8_32"
PIN766 = "P8_34"

# setup and initialize GPIO pins
GPIO.setup(PIN50, GPIO.OUT)
GPIO.output(PIN50, GPIO.LOW)
GPIO.setup(PIN144, GPIO.OUT)
GPIO.output(PIN144, GPIO.LOW)
GPIO.setup(PIN222, GPIO.OUT)
GPIO.output(PIN222, GPIO.LOW)
GPIO.setup(PIN432, GPIO.OUT)
GPIO.output(PIN432, GPIO.LOW)
GPIO.setup(PIN902, GPIO.OUT)
GPIO.output(PIN902, GPIO.LOW)
GPIO.setup(PIN1296, GPIO.OUT)
GPIO.output(PIN1296, GPIO.LOW)
GPIO.setup(PIN2304, GPIO.OUT)
GPIO.output(PIN2304, GPIO.LOW)
GPIO.setup(PIN3456, GPIO.OUT)
GPIO.output(PIN3456, GPIO.LOW)
GPIO.setup(PIN5760, GPIO.OUT)
GPIO.output(PIN5760, GPIO.LOW)
GPIO.setup(PIN1080, GPIO.OUT)
GPIO.output(PIN1080, GPIO.LOW)
GPIO.setup(PIN246, GPIO.OUT)
GPIO.output(PIN246, GPIO.LOW)
GPIO.setup(PIN476, GPIO.OUT)
GPIO.output(PIN476, GPIO.LOW)
GPIO.setup(PIN766, GPIO.OUT)
GPIO.output(PIN766, GPIO.LOW)

# setup ethernet UDP socket and start UDP server
port = 13063
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.bind(("",port))
print "waiting on port:", port

#start loop to receive UDP packets from NIMM
while 1:
    data, addr = s.recvfrom(1024)

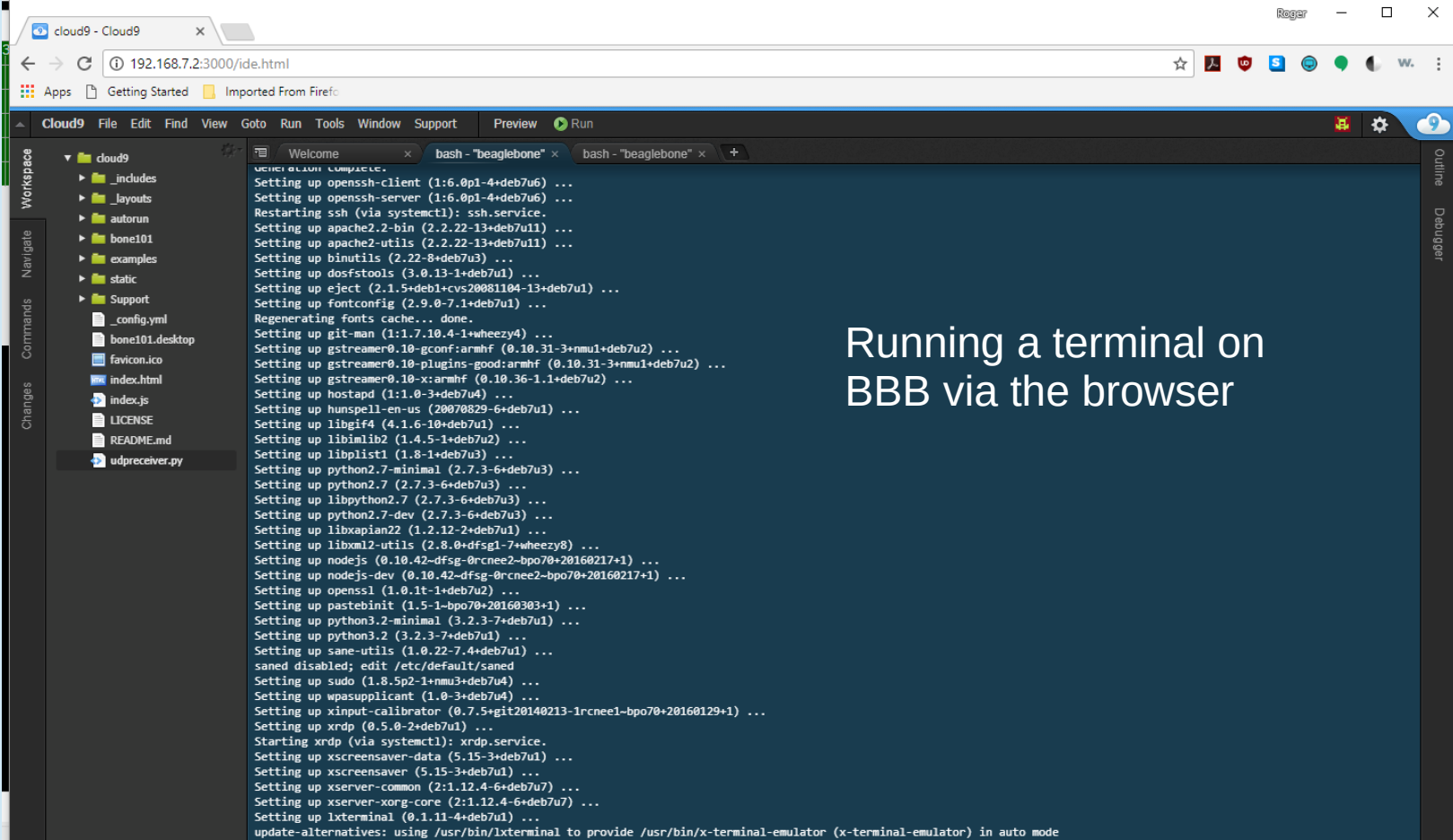
    #parse incoming data
    str1 = "<RadioNr>1</RadioNr>"
    str2 = "<Freq>"
    pos1 = data.find(str1)
    pos2 = data.find(str2)

    #if have valid XML data for NIMM Radio 1 then set band
    if pos1 >= 0:
        band = data[pos2+6:pos2+8]

    #start band switch code
    if band == "50" and band != oldband:
        print "found 50"
        oldband = "50"
        GPIO.output(PIN50, GPIO.HIGH)
        GPIO.output(PIN144, GPIO.LOW)
        GPIO.output(PIN222, GPIO.LOW)
        GPIO.output(PIN432, GPIO.LOW)
        GPIO.output(PIN902, GPIO.LOW)

[ Read 279 lines ]
Get Help WriteOut Read File Prev Page Cut Text Cur Pos
Exit Justify Where Is Next Page UnCut Text To Spell
```

Or you can use
the editor Nano
on the BBB



Running a terminal on BBB via the browser

```
root@beaglebone:/var/lib/cloud9# vi udpreceiver.py
root@beaglebone:/var/lib/cloud9# python udpreceiver.py
waiting on port: 13063
found 50
found 144
found 222
found 432
found 902
found 1296
found 2304
found 3456
found 5760
found 10 GHz
found 24 GHz
found 47 GHz
found 24 GHz
found 76 GHz
found 24 GHz
found 432
```

```
update-alternatives: using /usr/bin/lxterminal to provide /usr/bin/x-terminal-emulator (x-terminal-emulator) in auto mode
2-bpo70+20160217+1) ...
-bpo70+20160424+1) ...
po70+20160414+1) ...
0-bpo70+20170216+1) ...
com/codegen/esd/cgt_public_sw/PRU/2.1.4/ti_cgt_pru_2.1.4_armlinuxa8hf_busybox_installer.sh
... 23.199.194.206
com)|23.199.194.206|:80... connected.
K

ybox_installer.sh'

=====]>] 37,315,064 3.88M/s in 7.5s

'_2.1.4_armlinuxa8hf_busybox_installer.sh' saved [37315064/37315064]

2.1.4 into /

.

.

1) ...
u11) ...
vice.

mhf ...
```

```
Processing triggers for initscripts ...
update-initscripts: Generating /boot/initrd.img-3.8.13-bone79
root@beaglebone:/var/lib/cloud9#
```

IF/Transverter Bandswitching Ethernet N1MM UDP Broadcast

- 1) Import Libraries
- 2) Define constants and variables
 Define GPIO pin aliases
- 3) Setup and initialize GPIO pins
- 4) Setup Ethernet port and server socket
- 5) Get UDP data from Ethernet port
- 6) Parse UDP data to get band assignment
- 7) Use band information to set GPIO pins

Code Handout pages 9-13

BeagleBone Black Example

Import Libraries and Define Variables

```
1 # import libraries
2 import socket
3 import Adafruit_BBIO.GPIO as GPIO
4
5 #define variables
6 oldband = " "
7
```

Page 10 Code Handout

GOOGLED: "Programming UDP sockets in python":

4th hit was: <http://www.binarytides.com/programming-udp-sockets-in-python/>
That gave above code line 2.

GOOGLED: "setting up GPIO python on BeagleBone Black":

1st hit was:

<https://learn.adafruit.com/setting-up-io-python-library-on-beaglebone-black/gpio>
That gave above code line 3.

BeagleBone Black Example

Define GPIO Pin Aliases

```
10 # define GPIO pin aliases
11 PIN50 = "P9_12"
12 PIN144 = "P9_18"
13 PIN222 = "P9_24"
14 PIN432 = "P9_30"
15 PIN902 = "P9_31"
16 PIN1296 = "P9_42"
17 PIN2304 = "P8_9"
18 PIN3456 = "P8_15"
19 PIN5760 = "P8_18"
20 PIN10G = "P8_27"
21 PIN24G = "P8_33"
22 PIN47G = "P8_39"
```

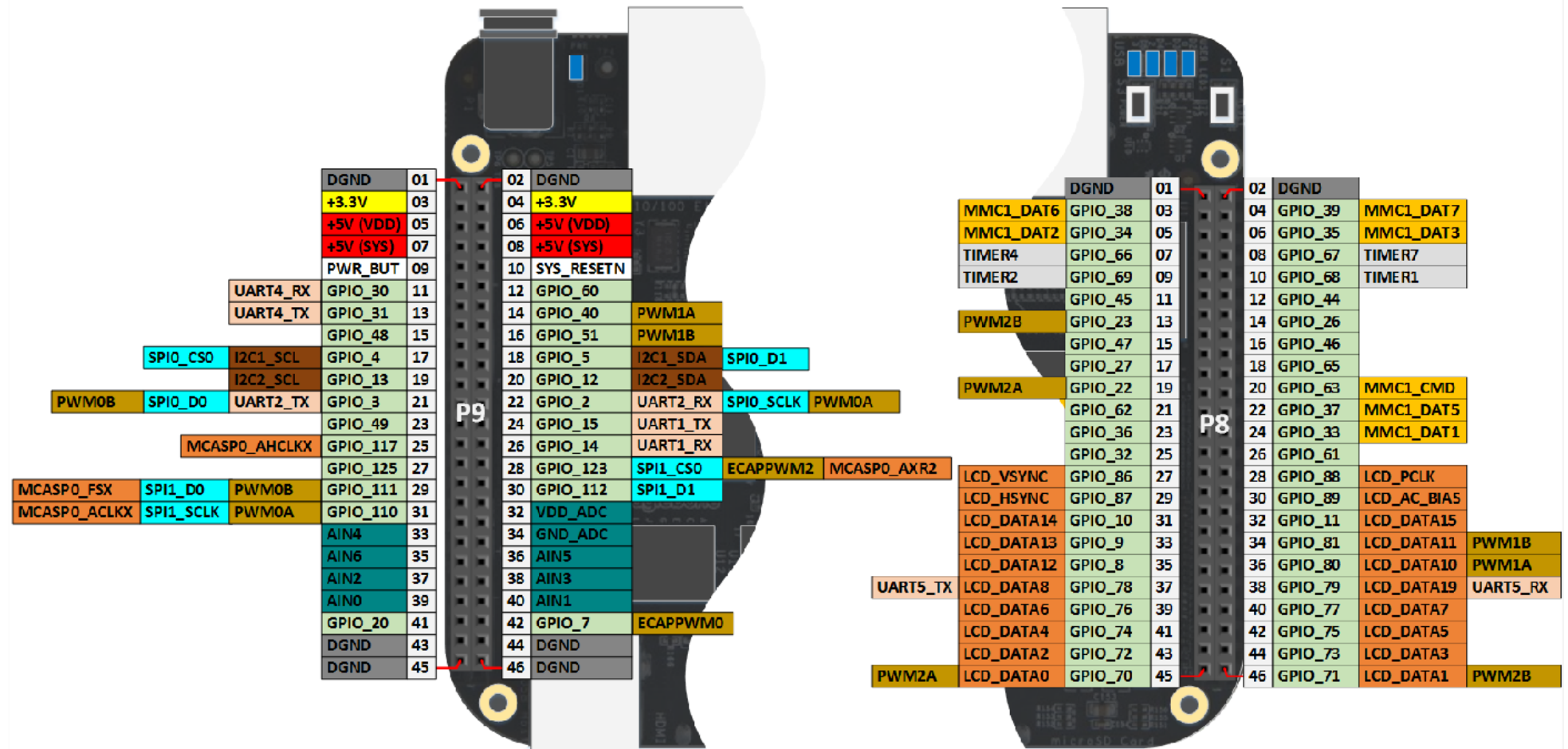
Page 10 Code Handout

BeagleBone Black Example GPIO Pin Aliases

The BeagleBone Black GPIO Pins

Rectangular Shp

The figure shows the full pin map produced by a call to showAllPins.



BeagleBone Black Example Setup and Initialize GPIO Pins (Zoomed next slide)

Page 10 Code Handout

```
24 #setup and initialize GPIO pins
25 GPIO.setup(PIN50, GPIO.OUT)
26 GPIO.output(PIN50, GPIO.LOW)
27 GPIO.setup(PIN144, GPIO.OUT)
28 GPIO.output(PIN144, GPIO.LOW)
29 GPIO.setup(PIN222, GPIO.OUT)
30 GPIO.output(PIN222, GPIO.LOW)
31 GPIO.setup(PIN432, GPIO.OUT)
32 GPIO.output(PIN432, GPIO.LOW)
33 GPIO.setup(PIN902, GPIO.OUT)
34 GPIO.output(PIN902, GPIO.LOW)
35 GPIO.setup(PIN1296, GPIO.OUT)
36 GPIO.output(PIN1296, GPIO.LOW)
37 GPIO.setup(PIN2304, GPIO.OUT)
38 GPIO.output(PIN2304, GPIO.LOW)
39 GPIO.setup(PIN3456, GPIO.OUT)
40 GPIO.output(PIN3456, GPIO.LOW)
41 GPIO.setup(PIN5760, GPIO.OUT)
42 GPIO.output(PIN5760, GPIO.LOW)
43 GPIO.setup(PIN10G, GPIO.OUT)
44 GPIO.output(PIN10G, GPIO.LOW)
45 GPIO.setup(PIN24G, GPIO.OUT)
46 GPIO.output(PIN24G, GPIO.LOW)
47 GPIO.setup(PIN47G, GPIO.OUT)
48 GPIO.output(PIN47G, GPIO.LOW)
```

BeagleBone Black Example Setup and Initialize GPIO Pins

```
23 #setup and initialize GPIO pins
24 GPIO.setup(PIN50, GPIO.OUT)
25 GPIO.output(PIN50, GPIO.LOW)
26 GPIO.setup(PIN144, GPIO.OUT)
27 GPIO.output(PIN144, GPIO.LOW)
28 GPIO.setup(PIN222, GPIO.OUT)
29 GPIO.output(PIN222, GPIO.LOW)
30 GPIO.setup(PIN432, GPIO.OUT)
31 GPIO.output(PIN432, GPIO.LOW)
```

GOOGLED: “setting up GPIO python on BeagleBone Black”:

1st hit was:

<https://learn.adafruit.com/setting-up-io-python-library-on-beaglebone-black/gpio>

That gave above code, all lines.

BeagleBone Black Example

Setup Ethernet Port and Server Socket

```
52 # setup ethernet UDP socket and start UDP server
53 port = 13063
54 s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
55 s.bind(("",port))
56 print "waiting on port:", port
57
```

Page 10 Code Handout

GOOGLED: "Programming UDP sockets in python":

4th hit was: <http://www.binarytides.com/programming-udp-sockets-in-python/>

That gave above code lines 53,54,55

BeagleBone Black Example

Setup Ethernet Port and Server Socket

```
52 # setup ethernet UDP socket and start UDP server
53 port = 13063
54 s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
55 s.bind(("",port))
56 print "waiting on port:", port
57
```

Page 10 Code Handout



Internet family socket



Datagram socket type

BeagleBone Black Example

Get UDP Data from Ethernet Port

```
58 #start loop to receive UDP packets from NIMM
59 while 1:
60     data, addr = s.recvfrom(1024)
61
```

Page 10 Code Handout

GOOGLED: "Programming UDP sockets in python":

4th hit was: <http://www.binarytides.com/programming-udp-sockets-in-python/>

That gave above code lines 59, 60

BeagleBone Black Example

Get UDP Data from Ethernet Port

```
58 #start loop to receive UDP packets from NIMM
59 while 1:
60     data, addr = s.recvfrom(1024)
61
```

Page 10 Code Handout

`socket.recvfrom(bufsize[, flags])`

Receive data from the socket. The return value is a pair (string, address) where string is a string representing the data received and address is the address of the socket sending the data.

BeagleBone Black Example Parse UDP Data to Get Band Assignment

```
61     #parse incoming data
62     str1 = "<RadioNr>1</RadioNr>"
63     str2 = "<Freq>"
64     pos1 = data.find(str1)
65     pos2 = data.find(str2)
66
67     #if have valid XML data for N1MM Radio 1 then set band
68     if pos1 >= 0:
69         band = data[pos2+6:pos2+8]
70
```

```
<?xml version="1.0" encoding="utf-8"?>
<RadioInfo>
    <StationName>INTEL-I7</StationName>
    <RadioNr>1</RadioNr>
    <Freq>5012500</Freq>
    <TXFreq>5012500</TXFreq>
    <Mode>USB</Mode>
    <OpCall>NN3Q</OpCall>
    <IsRunning>False</IsRunning>
    <FocusEntry>134626</FocusEntry>
    <Antenna>-1</Antenna>
    <Rotors>-1</Rotors>
    <FocusRadioNr>1</FocusRadioNr>
    <IsStereo>False</IsStereo>
    <ActiveRadioNr>1</ActiveRadioNr>
</RadioInfo>
```

Pages 10-11 Code Handout

BeagleBone Black Example Parse UDP Data

- To get “string.find” function, GOOGLED “python string find”

```
pos1 = data.find(str1)  
pos2 = data.find(str2)
```

2nd hit was:

https://www.tutorialspoint.com/python/string_find.htm

- To get string position function “data[a:b]”, GOOGLED “python string substring”

```
band = data[pos2+6:pos2+8]
```

3rd hit was: <https://www.dotnetperls.com/substring-python>

BeagleBone Black Example

Use Band Info to Set GPIO Pins

```
#start band switch code
if band == "50" and band != oldband:
    print "found 50"
    oldband = "50"
    GPIO.output(PIN50, GPIO.HIGH)
    GPIO.output(PIN144, GPIO.LOW)
    GPIO.output(PIN222, GPIO.LOW)
    GPIO.output(PIN432, GPIO.LOW)
    GPIO.output(PIN902, GPIO.LOW)
    GPIO.output(PIN1296, GPIO.LOW)
    GPIO.output(PIN2304, GPIO.LOW)
    GPIO.output(PIN3456, GPIO.LOW)
    GPIO.output(PIN5760, GPIO.LOW)
    GPIO.output(PIN10G, GPIO.LOW)
    GPIO.output(PIN24G, GPIO.LOW)
    GPIO.output(PIN47G, GPIO.LOW)
```

```
elif band == "14" and band != oldband:
    print "found 144"
    oldband = "14"
    GPIO.output(PIN50, GPIO.LOW)
    GPIO.output(PIN144, GPIO.HIGH)
    GPIO.output(PIN222, GPIO.LOW)
    GPIO.output(PIN432, GPIO.LOW)
    GPIO.output(PIN902, GPIO.LOW)
    GPIO.output(PIN1296, GPIO.LOW)
    GPIO.output(PIN2304, GPIO.LOW)
    GPIO.output(PIN3456, GPIO.LOW)
    GPIO.output(PIN5760, GPIO.LOW)
    GPIO.output(PIN10G, GPIO.LOW)
    GPIO.output(PIN24G, GPIO.LOW)
    GPIO.output(PIN47G, GPIO.LOW)
```

Page 11 Code Handout

GOOGLED "python if"

2nd hit was: <https://www.dotnetperls.com/substring-python>

That gave "if / elif" syntax.

BeagleBone Black Example

Use Band Info to Set GPIO Pins

```
elif band == "24" and band != oldband:  
    print "found 24 GHz"  
    oldband = "24"  
    GPIO.output(PIN50, GPIO.LOW)  
    GPIO.output(PIN144, GPIO.LOW)  
    GPIO.output(PIN222, GPIO.LOW)  
    GPIO.output(PIN432, GPIO.LOW)  
    GPIO.output(PIN902, GPIO.LOW)  
    GPIO.output(PIN1296, GPIO.LOW)  
    GPIO.output(PIN2304, GPIO.LOW)  
    GPIO.output(PIN3456, GPIO.LOW)  
    GPIO.output(PIN5760, GPIO.LOW)  
    GPIO.output(PIN10G, GPIO.LOW)  
    GPIO.output(PIN24G, GPIO.HIGH)  
    GPIO.output(PIN47G, GPIO.LOW)
```

```
elif band == "47" and band != oldband:  
    print "found 47 GHz"  
    oldband = "47"  
    GPIO.output(PIN50, GPIO.LOW)  
    GPIO.output(PIN144, GPIO.LOW)  
    GPIO.output(PIN222, GPIO.LOW)  
    GPIO.output(PIN432, GPIO.LOW)  
    GPIO.output(PIN902, GPIO.LOW)  
    GPIO.output(PIN1296, GPIO.LOW)  
    GPIO.output(PIN2304, GPIO.LOW)  
    GPIO.output(PIN3456, GPIO.LOW)  
    GPIO.output(PIN5760, GPIO.LOW)  
    GPIO.output(PIN10G, GPIO.LOW)  
    GPIO.output(PIN24G, GPIO.LOW)  
    GPIO.output(PIN47G, GPIO.HIGH)
```

GOOGLED "python if"

2nd hit was: <https://www.dotnetperls.com/substring-python>

That gave "if / elif" syntax.

Page 13 Code Handout

BeagleBone Black Example

IT DIDN'T WORK!!

- 50, 144, 222, 432, 1296, 2304, 3456 MHz worked
- 902 MHz and 5, 10, 24, 47 GHz didn't work
- 902 MHz and 10 GHz were always ON!

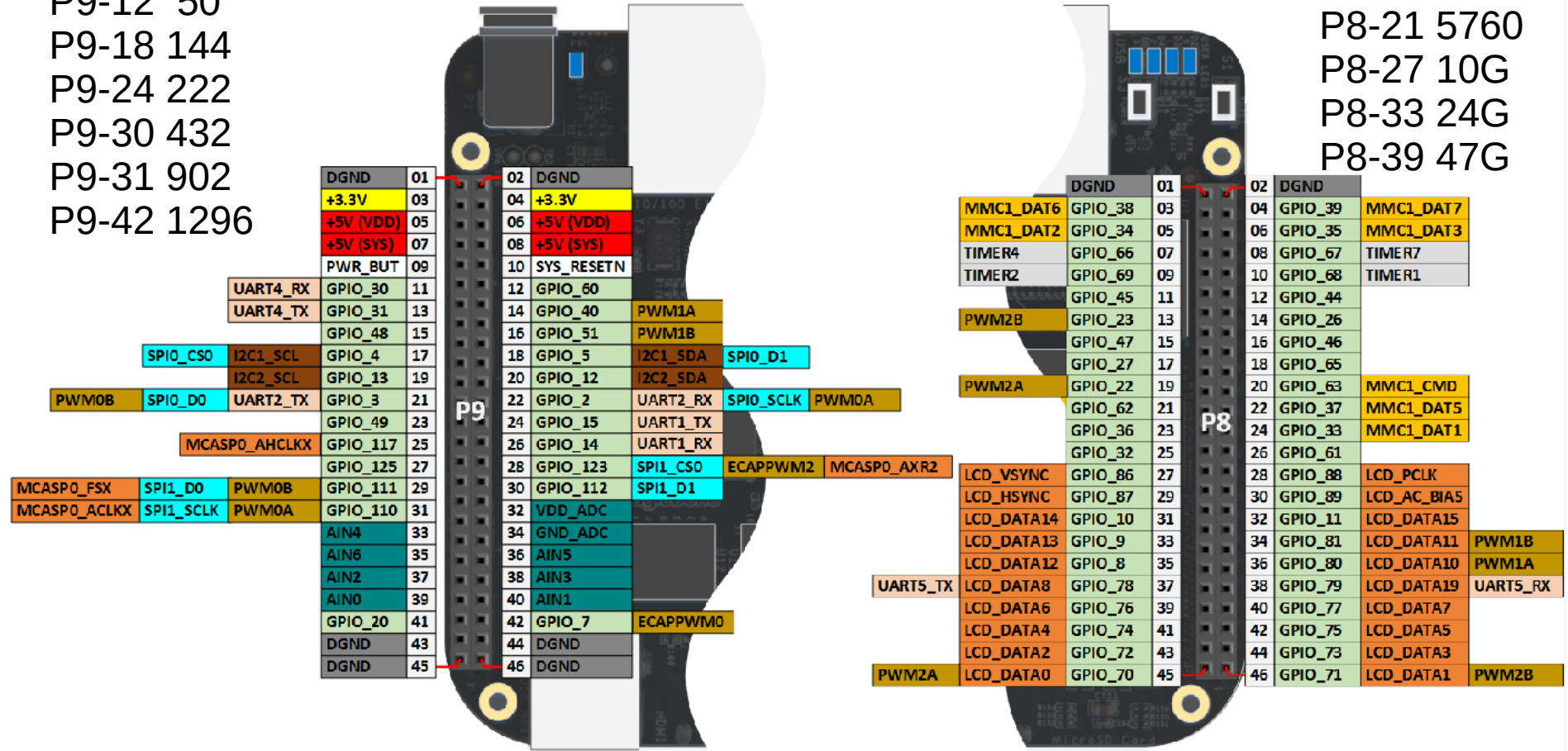


The BeagleBone Black GPIO Pins

Rectangular Strip

The figure shows the full pin map produced by a call to showAllPins.

P9-12 50
 P9-18 144
 P9-24 222
 P9-30 432
 P9-31 902
 P9-42 1296



P8-9 2304
 P8-15 3456
 P8-21 5760
 P8-27 10G
 P8-33 24G
 P8-39 47G

- 50, 144, 222, 432, 1296, 2304, 3456 MHz worked
- 5, 10, 24, 47 GHz didn't work
- 902 MHz and 10 GHz were always ON!
- Googled "BeagleBone Black P8-27 always HIGH -->

BBB Example Didn't Work!

- Google →

<https://github.com/AbhraneelBera/wiringBone>

If pins- P9.25, P9.28, P9.29, **P9.31** are used hdmi-audio cape should be disabled first.

If pins- **P8.27**, P8.28, P8.29, P8.30, P8.31, P8.32, **P8.33**, P8.34, P8.35, P8.36, P8.37, P8.38, **P8.39**, P8.40, P8.41, P8.42, P8.43, P8.44, P8.45, P8.46 are used hdmi cape should be disabled first.

If pins- P8.3, P8.4, P8.5, P8.6, P8.20, **P8.21**, P8.22, P8.23, P8.24, P8.25 are used emmc cape should be disabled first.

P9-12 50
P9-18 144
P9-24 222
P9-30 432
P9-31 902
P9-42 1296

P8-9 2304
P8-15 3456
P8-21 5760
P8-27 10G
P8-33 24G
P8-39 47G

BBB Example Didn't Work!

- Same Google page gave reference for how to disable HDMI cape:
 - “Uncomment” one line in file /boot/uEnv.txt to disable HDMI Video and Audio:
dtb=am335x-boneblack-emmc-overlay.dtb
- After reboot, that fixed all except P8-21, which was due to eMMC.
 - eMMC is the flash memory and controller for the BBB, so can't delete it unless use SD for memory.
 - So I switched P8-21 to P8-18 and everything worked.

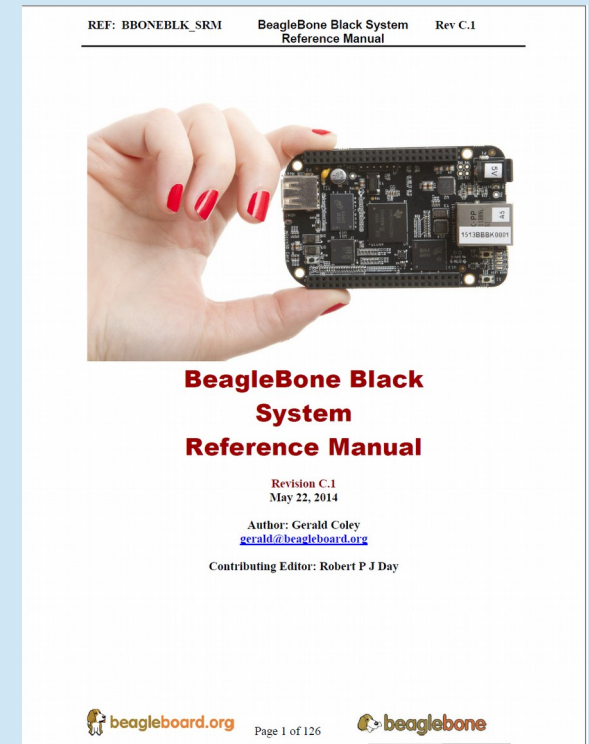
P9-12 50
P9-18 144
P9-24 222
P9-30 432
P9-31 902
P9-42 1296

P8-9 2304
P8-15 3456
P8-21 5760
P8-27 10G
P8-33 24G
P8-39 47G

When All Else Fails:

https://github.com/beagleboard/BeagleBone-Black/raw/master/BBB_SRM.pdf

Pages 96-98 discuss the issues with GPIO pins, HDMI, eMMC conflicts



NOTE: Do not connect 5V logic level signals to these pins or the board will be damaged.

NOTE: DO NOT APPLY VOLTAGE TO ANY I/O PIN WHEN POWER IS NOT SUPPLIED TO THE BOARD. IT WILL DAMAGE THE PROCESSOR AND VOID THE WARRANTY.

NO PINS ARE TO BE DRIVEN UNTIL AFTER THE SYS_RESET LINE GOES HIGH.

2_BBB_EthernetUDP_N1MM2_iPhonePIP.0

File Edit View Tools Config Window Help

CW PH Grid

6m 6m

2m 2m

1.25m 1.25m Run S&P

F1 S&P CQ	F2 Exch	F3 Spare	F4 K3WGR	F5 His Call	F6 Spare
F7 Rpt Exch	F8 Agn?	F9 Zone	F10 Spare	F11 Spare	F12 Wipe

Left Click, QSY to 3456101.9

Log It Edit Mark Store Spot It QRZ

9cm 9cm

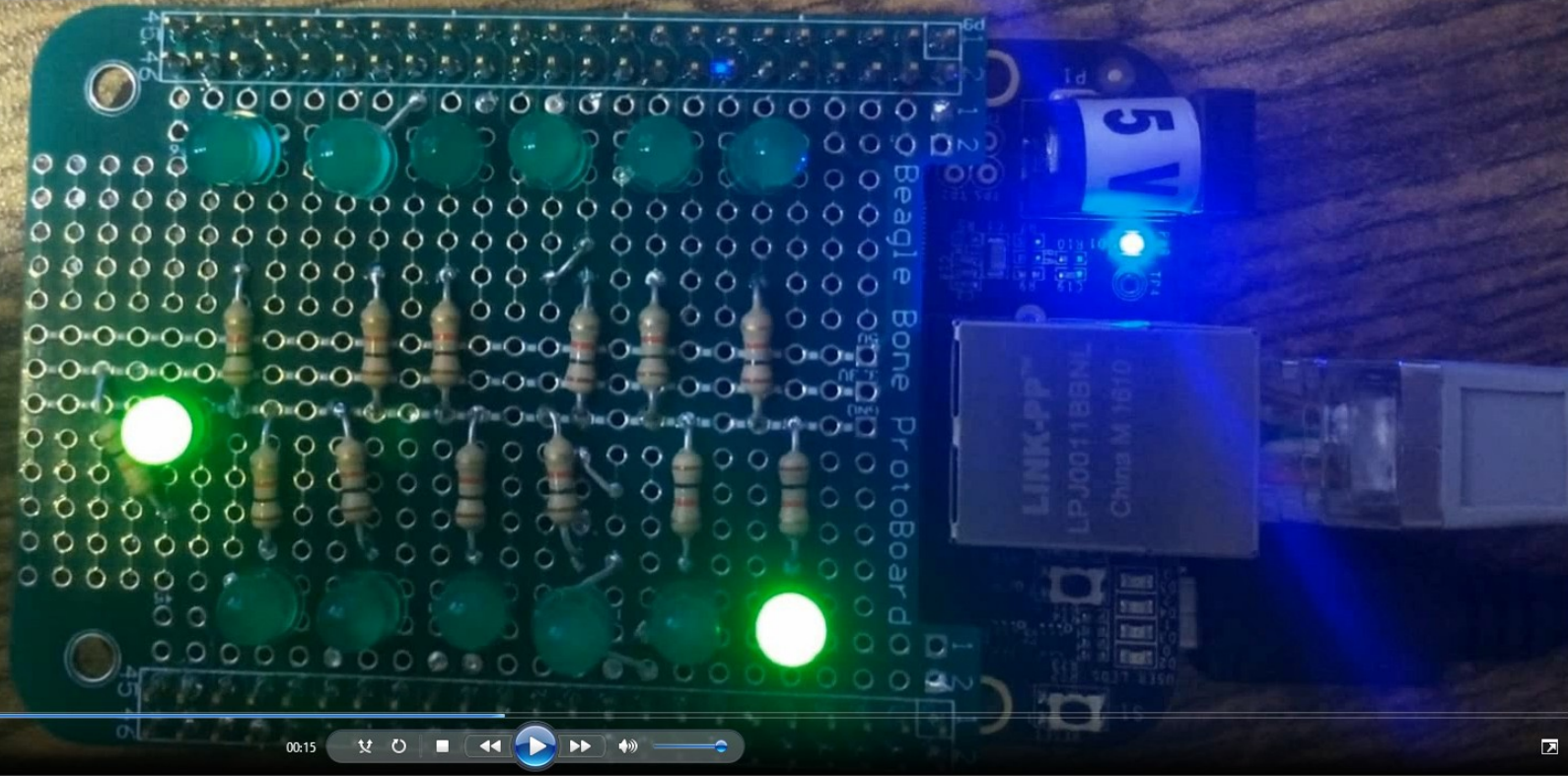
6cm 6cm Call history UserText appears here when enabled.

3cm 3cm

1cm 1cm

Light Light

GM: EU/SCOTLAND, Zn 14 141/0 47,496



00:15

⏪ ⏩ 🔊 🔍

E:\StationAutomation\PackRatsMiniTalk\2_BBB_EthernetUDP_N1MM2_iPhonePIP.wmv

Station Automation Coding

- **Very Simple:**
 - Got Some Input
 - Did Something With It
 - Produced Some Output

Station Automation Coding

1) Imported Libraries

- 1) Socket
- 2) Adafruit_BBIO.GPIO as GPIO

2) Defined constants and variables

Defined GPIO pin aliases

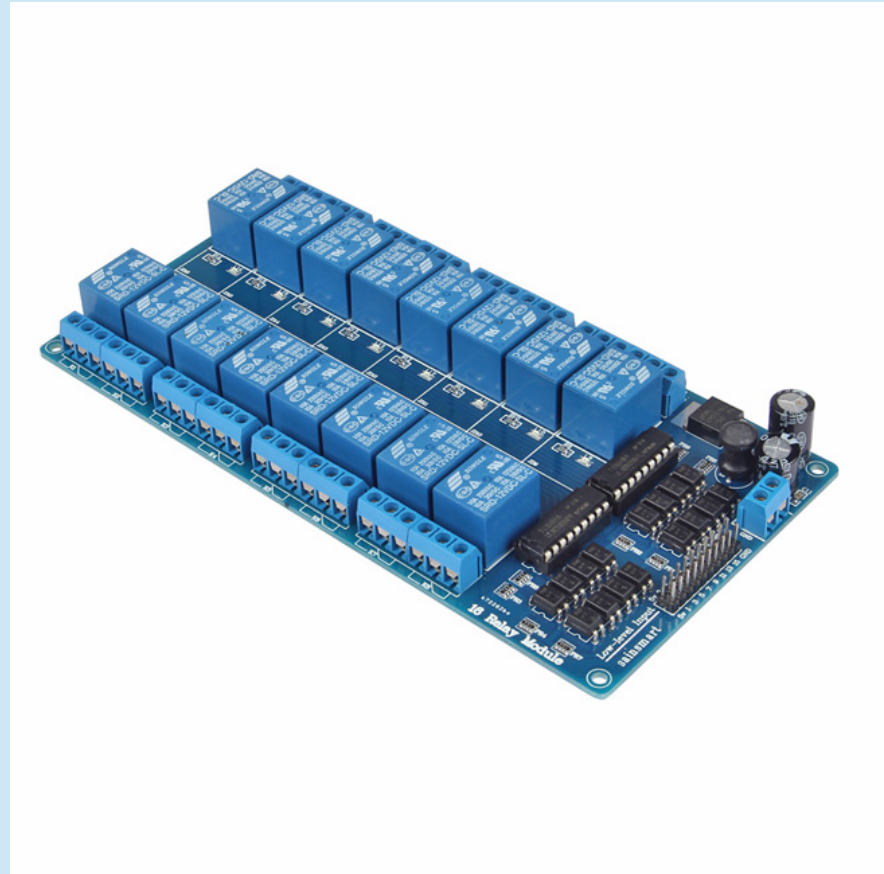
- 3) Setup and initialized GPIO pins
- 4) Setup Ethernet port and server socket
- 5) Got UDP data from Ethernet port
- 6) Parsed UDP data to get band assignment
- 7) Used band information to set GPIO pin outputs

Code Handout pages 9-13

IF/Transverter Bandswitching Ethernet N1MM UDP Broadcast

Sainsmart 16-relay
board (cost: \$14.99
with free shipping)

Requires 15-20 ma per pin
so need to buffer GPIO
outputs



What about the RF Relays?

- RF relay goes between the transverter (or antenna) port of your IF radio and the IF input/output of your transverters.
- If you have split IF ports (Rx, Tx) on your transverter, then you need two relays, one for Rx and one for Tx
- I use surplus SP8T, SP6T relays
- There are million ways to do this – see next slide

What about the RF Relays - 50MHz-24GHz?

- IF Radio-SP8T (50,144,222,432,903,1296, Micro) plus SP6T fed from Micro port (2,3,5,10,24 GHz)
- IF Radio-SPDT (Lo + Hi): Lo feeds SP6T for 50,144,222,432,903,1296; Hi feeds SP6T for 2,3,5,10,24 GHz
- IF Radio-SP6T(50,144,222,432,Hi) plus SP8T fed from Hi port (903, 1296, 2, 3, 5, 10, 24 GHz)
- Etc., etc., etc.

What about the RF Relays - 50MHz-24GHz?

SP8T \$48



SP6T \$61



SP6T \$54



SP6T \$25

HOW TO WRITE GOOD CODE:

