# Building the next **Internet Of Things** device using a Raspberry Pi and simple electronics.

## **IoT** What is “it”

- Yet another buzz word and acronym or is it more?
  - The term started in 1999 by Kevin Asthon but the concept has been around since the early 90's and originally revolved around RFID tags and the concept that if everything had a unique license plate  number it could be tracked and managed by computers.
  - As with many terms the meaning has changed over the years as technology and needs have changed.

## **IoT** Today

- The term Internet of Things (commonly abbreviated as **IoT**) today is used to denote the connectivity of devices, systems and services that goes beyond the traditional machine to machine communications and covers a variety of protocols, domains and applications intended to collect and distribute information to people as well as systems and allow these systems to be managed by machines and people in an efficient secure and personalized manner.

## **IoT** Today

- So it goes beyond the traditional Internet into transports like mesh networks, point to point connections, near field, sonar and more. These devices can be interacted with or interact with us in ways that will help us manage the ever growing volume of information that we are collecting by having all of these devices running.

# **IoT** Today

- One example of **IoT** that few people think about is our own cell phones. These devices are connected to radio networks and have GPS or cell tower triangulation built in that allows them to be used to determine traffic patterns and congestion. The cameras located on the freeways in conjunction with the cell phones in the cars allows the people who can access this information the ability to show us via the map programs on our phones where the traffic congestion is on the freeway heping us decide if we should take a detour or change our trip plans before we get stuck in a traffic jam.

## Challenges

26 smart things for every human on the planet!!

- **IoT** faces several challenges over the next few years the biggest I feel are address space, privacy and security. With an estimated 200 billion devices in 2020[1] we will not have enough addresses with IPv4 so IPv6 will play a big part in the growth of **IoT**. With the limits of IPv6 not having any NAT built into the standard raising privacy issues and security issues such as Heartbleed I expect **IoT** devices will be on the radar with security and privacy groups frequently in years to come.

1. Sources: IDC, Intel, United Nations

## Enough Terms and Stats lets get to how I can build my own IoT device

- The biggest challenge to anyone who already has some electrical engineering experience and who wants to build an **IoT** device is the Internet part. To build an embedded device that has a TCP stack and all of the magnetics and PHY chips and the complex communication protocols to talk to these chips it quickly becomes a massive undertaking.

## Enough Terms and Stats lets get to how I can build my own IoT device

- Sure it has gotten easier with such products as the WIZnet W5100 where a lot of the protocol stacks have been obfuscated to allow easy connecting to your Atmel or Pic project but at the end of the day you are still limited by what your Pic or Atmel chip can do with its limited code space so adding things like encryption make these solutions limited and costly to develop.

## Enough Terms and Stats lets get to how I can build my own IoT device

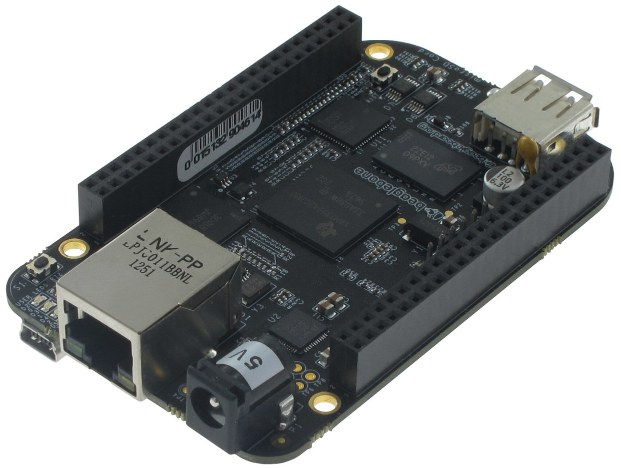
- At the end of the day its also about price. With the cost of the magnetics and PHY chips your already looking at $20 in parts and countless hours of C code to write before you can even get UDP packet out the door much less a full TCP session.

## A better way

- Leveraging the low cost educational and engineering tools one can have a full featured Linux or BSD based **IoT** device for as little as $40 and then add your own IP to make your own **IoT** device.
- These devices are known as credit card sized computers and all feature expansion or GPIO headers

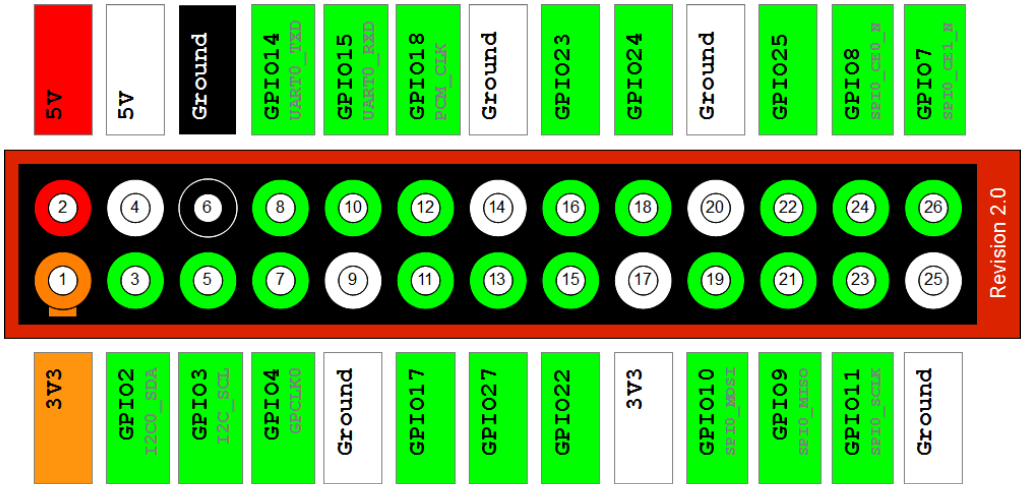
Raspberry Pi

Beaglebone Black (BBB)



## Ya but that GPIO buss looks scary do I need to learn about PCI bus and read lots of engineering papers?

- It is as easy as 3.1415927......
  - The GPIO header on the Pi is the simplest and its as simple as reading a PIN or talking serial on your uP.



## Ya but that GPIO buss looks scary do I need to learn about PCI bus and read lots of engineering papers?

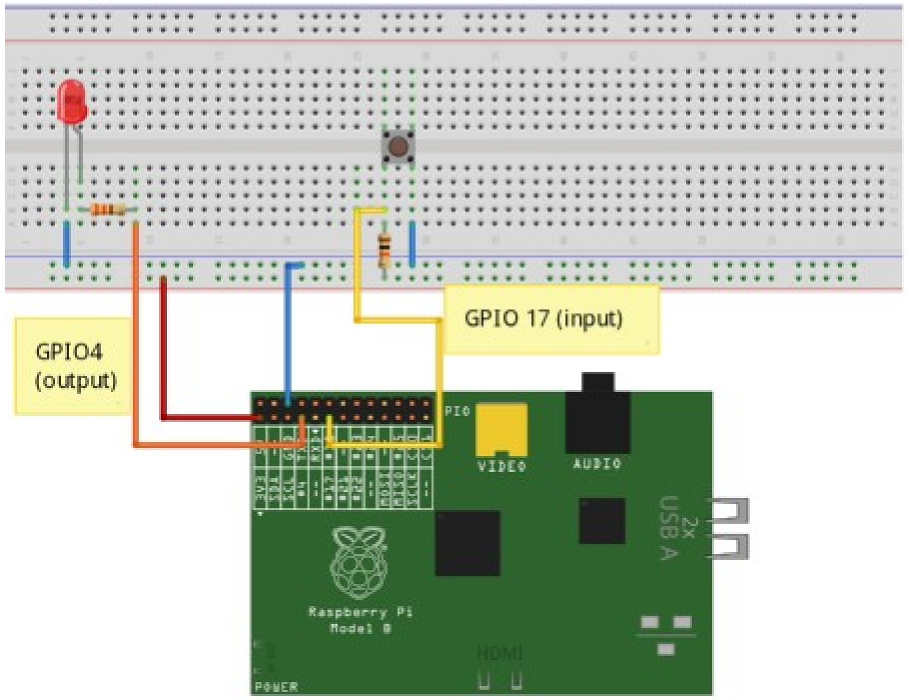
- The GPIO header on the BBB is more complex and has more features but again it can be as simple as reading and writing to a pin using Python but does require loading a cape manager template to configure the pins.
- A big advantage of the BeagleBone Black is that it also has 7 analog inputs, making it much easier to connect sensors. The Raspberry Pi has none and requires an extra analog to digital convertor chip to read analog sensors.

# Comparison Pi vs BBB

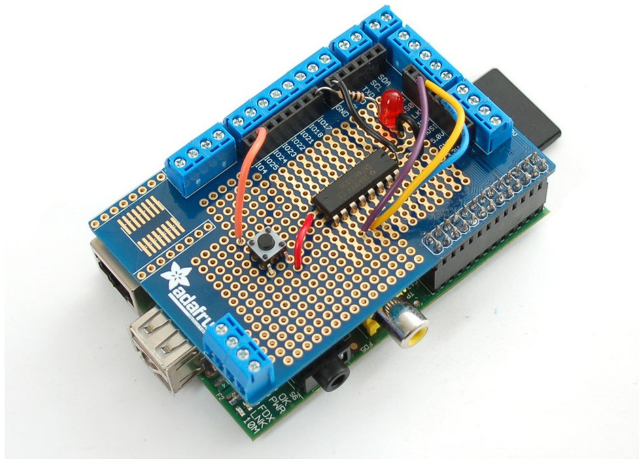
| | **BeagleBone Black** | **Raspberry Pi** |
|---|---|---|
| Base Price | 45 | 40 |
| Processor | 1GHz TI Sitara AM3359 ARM Cortex A8 | 700 MHz ARM1176JZFS |
| RAM | 512 MB DDR3L @ 400 MHz | 512 MB SDRAM @ 400 MHz |
| Storage | 2 GB on-board eMMC, MicroSD | SD |
| Video Connections | 1 Mini-HDMI | 1 HDMI, 1 Composite |
| Supported Resolutions | 1280×1024 (5:4), 1024×768 (4:3), 1280×720 (16:9), 1440×900 (16:10) all at 16 bit | Extensive from 640×350 up to 1920×1200, this includes 1080p |
| Audio | Stereo over HDMI | Stereo over HDMI, Stereo from 3.5 mm jack |
| Operating Systems | Angstrom (Default), Ubuntu, Android, ArchLinux, Gentoo, Minix, RISC OS, others… | Raspbian (Recommended), Android, ArchLinux, FreeBSD, Fedora, RISC OS, others… |
| Power Draw | 210-460 mA @ 5V under varying conditions | 150-350 mA @ 5V under varying conditions |
| GPIO Capability | 65 Pins (A/D) | 8 Pins ( NO A/D) |
| Peripherals | 1 USB Host, 1 Mini-USB Client, 1 10/100 Mbps Ethernet | 2 USB Hosts, 1 Micro-USB Power, 1 10/100 Mbps Ethernet, RPi camera connector |

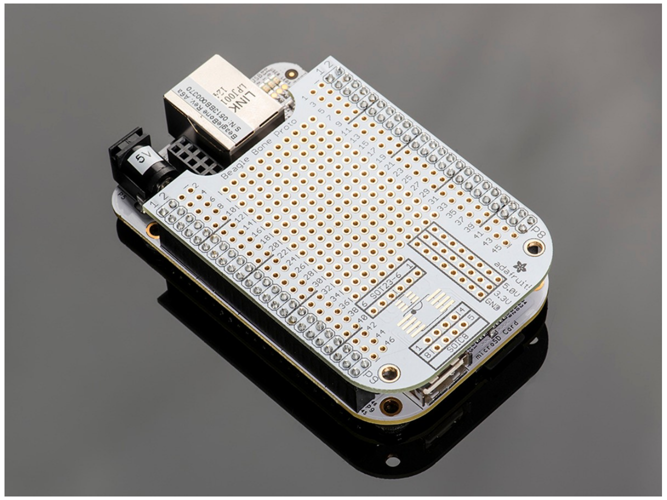
# Start with a prototype

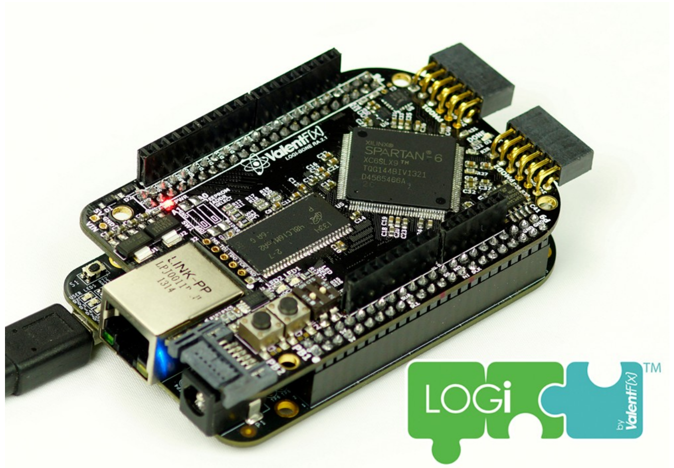
- As with all things electronic its best to start with a prototype and test it out before you build your own PCB.
- The Pi makes this easy with the GPIO header just use wires and a prototype board and built it on your bench before you draw it all up.



# Prototype tools


Pi Plate Kit


BBB Proto Cape


Xilinx FPGA http://valentfx.com/logi-bone/

## Get your Platform up and running

- Raspberry Pi does require a few steps before you can boot the board and start working.
  - The Raspberry Pi makes it very easy to start playing with Linux and GPIO. First download one of the approved images and install it onto your flash disk. It is not possible to brick your Pi so don't worry about that. Once you have your Pi up and running Linux you can start playing with the GPIO header. I recommend the Raspbian image it is very stable and easy to add packages. Because the Pi can only boot from the SD card its very simple but does require the correct tools to format the SD properly before you can start the next step.
  - http://www.raspberrypi.org/downloads/

# Get your Platform up and running

BeagleBone Black comes pre-configured with an OS so you can boot it up and immediately start using the GPIO header

- The BBB is a little more confusing because of the multiple devices it can boot from but just as impossible to brick. The BBB has a button (S2) that is designed to tell the ROM stage 1 boot loader to change the order of devices it looks for its stage 2 loader. If this is pressed it will try to load from the SPI0, MMC0 (uSD), USB0 and finally UART0. If not pressed the order is MMC1 (onboard eMMC), MMC0 (uSD), UART0 and finnally USB0. If it does not detect the necessary MLO file on the file system TOC or the correct  XMODEM control from the UART it will move on to the next device.
- I recommend erasing the eMMC with 0's or flashing a special MLO that jumps to the uSD by default as you are developing your **IoT** to make it simpler and be assured it will load from the uSD on ever boot.
- Because of the added complexity in the boot process it is recommended that one get an FTDI TTL serial adapter to watch the boot loader from the JTAG interface.

## BBB Angstrom

- It is Linux but things are a little different than most. It does not have a traditional syslog daemon but uses one that writes to a circular binary file and requires journalctl to read from it. I think this is smarter than syslog for embedded but another option is to move your syslog folder to a ram disk to avoid disk corruption or running out of disk space. I have been looking for other solutions but so far have not found any I like. On my own Linux distribution UFO I opted for a ram drive long ago and other than loosing data after reboot I have never had a corrupt file system or full disk. Also on Angstrom the package manager is opkg not dpkg or apt-get

## Python Fun

- Easy to install on Raspbian just make sure your system is updated and install the python module and start talking to the GPIO header.

```
$ sudo apt-get install python-rpi.GPIO
root@alarmdecoder:/home/pi# python
Python 2.7.3 (default, Mar 18 2014, 05:13:23)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import RPi.GPIO as GPIO
>>> GPIO.setmode(GPIO.BCM)  # note below
>>> GPIO.setup(4,GPIO.OUT)
>>> GPIO.output(4,True)
>>> GPIO.output(4,False)
```

GPIO.BCM Broadcom SOC channel
GPIO.BOARD board GPIO pin # on P1 header

## BBB Node.js fun

- The default OS on BBB is Angstrom and this contains the kitchen sink of tools to begin GPIO testing mostly based on Node.js

```
var b = require('bonescript');
var pin = 'P8_12';

b.pinMode(pin, b.OUTPUT);

var toggle = b.HIGH;
setInterval(flashPin, 1000);

function flashPin() {
        console.log("setting pin to %d",toggle);
     b.digitalWrite(pin, Number(toggle));
     toggle = !toggle;
}
```

# Now lets make a PCB

- Lots of people think that this is the hardest part and in some ways it is but I have a few tips today that will hopefully make it a lot less scary.
  - For starts it can be very expensive to make your own PCB

    Some fab shops charge hundreds of dollars to make a simple PCB and that is enough to stop most enthusiasts or even small companies in their tracks. I would like to remind everyone that we live in the Silicon Forest and we have lots of hobbyist community leaders that have made it so that everyone can enjoy working in this field.

    If you don't know about DorkbotPDX or Laen and his hard work in building OSH Park please check them out and support our local hobbyist efforts

    **Dorkbotpdx.org**

    Here you can get a 2 layer PCB with silk layer and mask for as little as $5 per square inch in as little as 12 days. A comparable local company I wont mention will charge over $100 with no mask or silk layer and as much as $200 for a full feature PCB.
  - Mistakes happen I have made plenty especially since I tend to hand draw everything and don't use any auto routing. I am sure Laen has seen me send what looks like the same PCB a few times in as many weeks and likely knows a face palm had to happen.

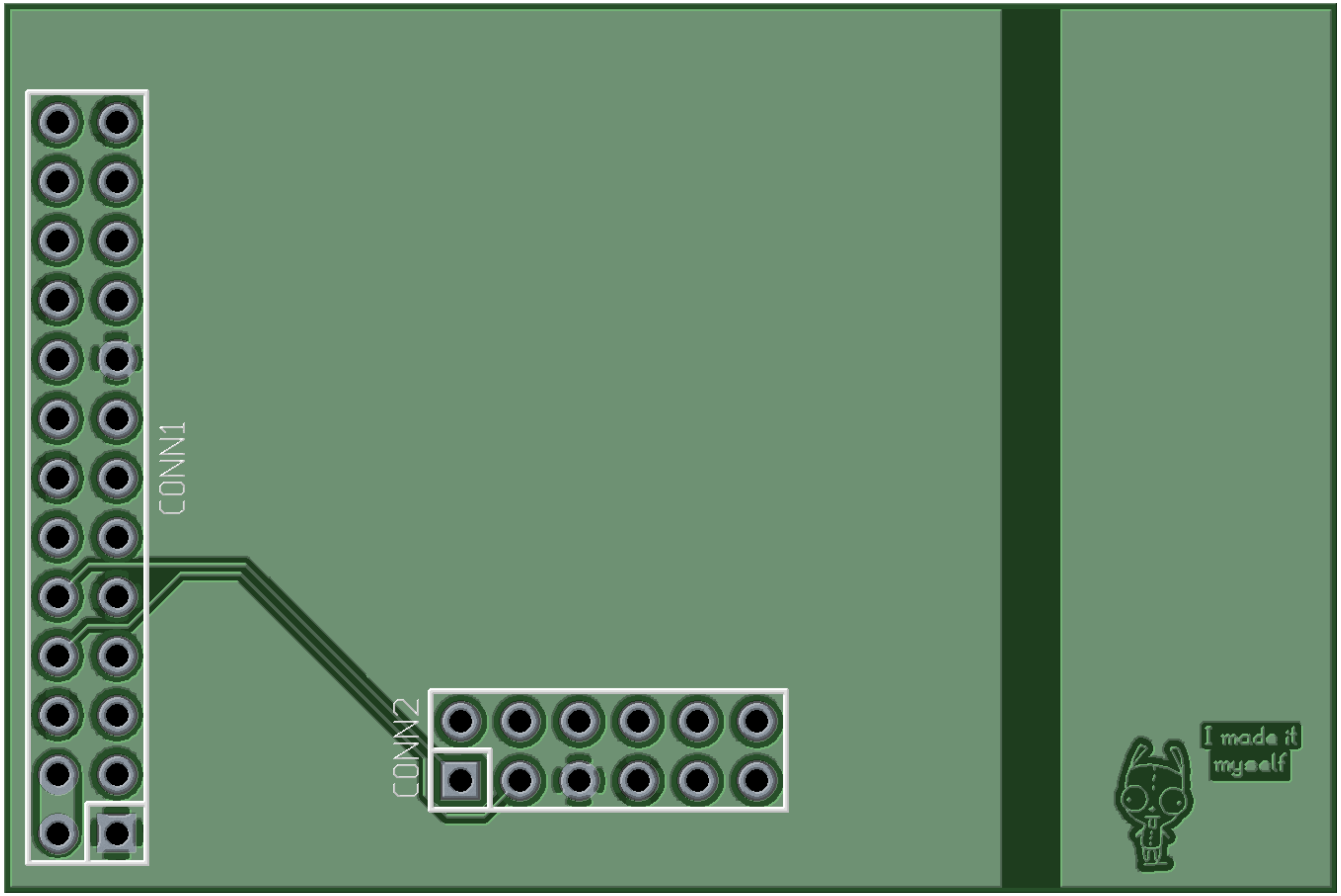
# Now lets make a PCB

- Second the tools can be expensive and complicated to use.
  - You may have your own tools.  Some are free such as Eagle Freeware but I would like to suggest an open source project called gEDA
    - http://www.geda-project.org/
    - You can download a pre done gEDA PCB layout for a Raspberry Pi GPIO board today from our site at
    - http://www.alarmdecoder.com/wiki/index.php/Raspberry_Pi

I can't help with the complicated part but with this existing layout hopefully it will make it a little easier to get started. I could easily stay on the topic of gEDA for another hour discussing the many free oss footprint files available and the many quirks I have had to work with over the years but not today. Maybe another day or even a workshop on using gEDA would be more appropriate so people can actually work with it directly.

# Now lets make a PCB

- Your first Pi GPIO **IoT** board starts here



## Now lets make a PCB

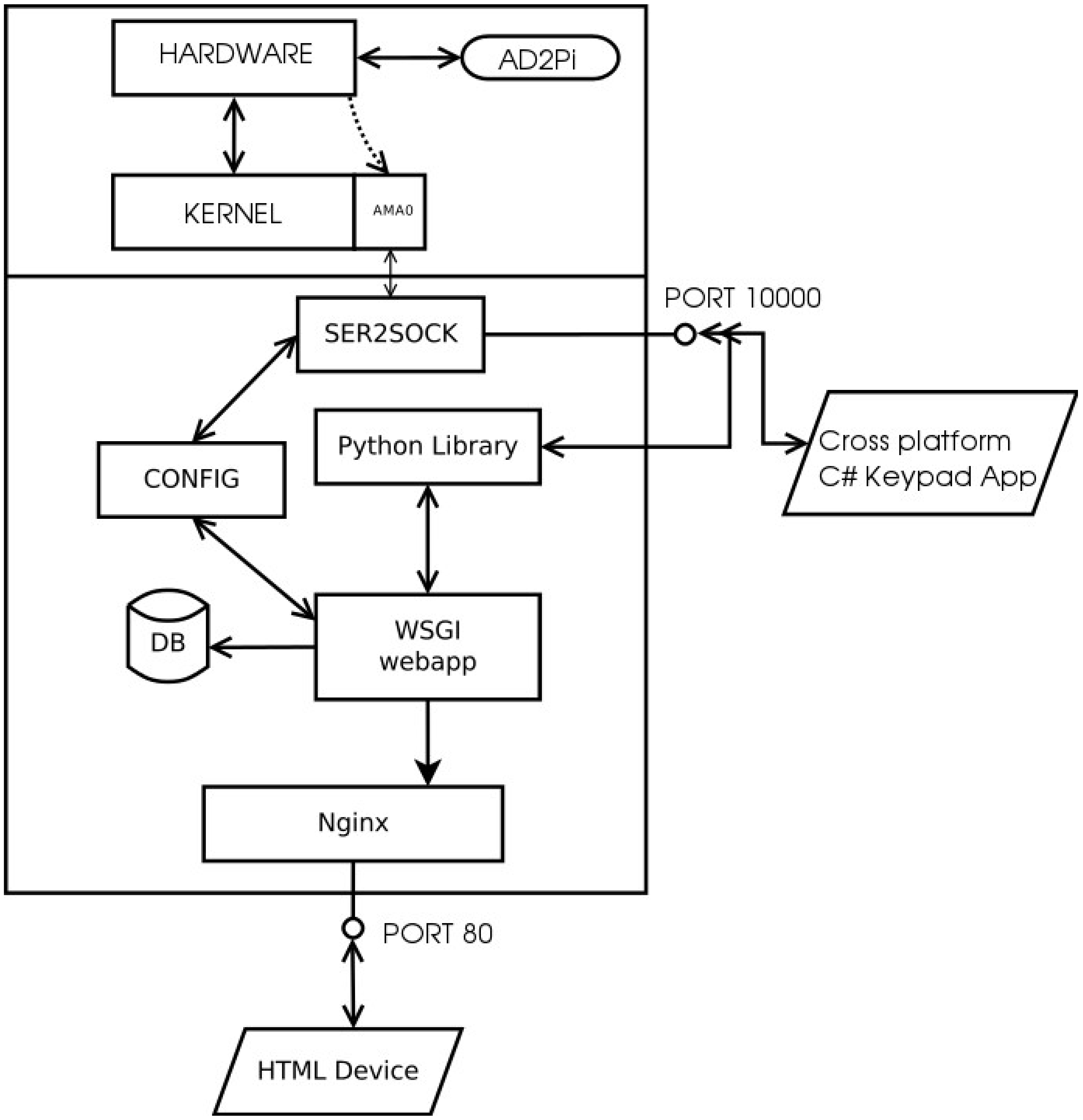
- In this example PCB I have connected the GPIO SPI TX/RX pins that would then be connected to a PIC or Atmel chip. In order to use these pins and have access to the port you only need to disable the Linux serial console that by default uses these pins as ttyAMA0.

  http://www.alarmdecoder.com/wiki/index.php/Raspberry_Pi

## Now for the Internet part

- Here is where its all up to you. I chose to keep my projects open source. Feel free to look at our projects and contribute to or borrow from for your own OSS efforts.

- https://github.com/nutechsoftware/ser2sock
- https://github.com/nutechsoftware/alarmdecoder
- https://github.com/nutechsoftware/alarmdecoder-webapp


HARDWARE
AD2Pi
KERNEL
AMA0
SER2SOCK
PORT 10000
Python Library
Cross platform C# Keypad App
CONFIG
DB
WSGI webapp
Nginx
PORT 80
HTML Device

## What next?

- I highly recommend heading out to Oregon Electronics and your local Dorkbot gathering to find ideas and parts. Oregon Electronics has Raspberry Pi boards and lots of sensors as well as components to play with and Dorkbot is an excellent way to connect with other hardware hackers in the area.

www.oregon-electronics.com

www.dorkbotpdx.org

Don't forget to share your experience and encourage others to make the next **IoT** device

Sean Mathews
coder at f34r dot com