How To:Quantum Encryption Device

short line

Last updated:

February, 2018

# How to build

1. Introduction
2. Materials
3. Software
4. Hardware
5. Code Explanation
6. Conclusion

Introduction:

Quantum Encryption is an increasingly important pursuit as the race for functional quantum computers draws closer to an end. Having a reliable encryption method is important if we want to transfer sensitive data, just as it is important in a classical computer. Classical encryption methods rely heavily on factoring; for a standard computer it takes a long time to find the factor of very large numbers. However quantum computers can factor much faster (how much faster depends on the number of qubits in a system), and because of this new methods of encryption need to be made.

This device was built from the research paper “Hybrid Quantum Encryption”[[1]](#footnote-0) by Anthony B. Kunkel at St. Cloud State University. Basic methodology: The Arduino microcontroller reads information from the Vernier Radiation Monitor and sends that information to the Raspberry Pi. From there the information is displayed on the monitor connected to the Raspberry Pi.

The purpose of this approach is to provide and encryption that can be applied to both quantum computers and classical computers. To do this we need random numbers either to generate keys to the encryption or as part of the encryption method depending on the approach. We use this device to generate truly random numbers as random number generators in computers are never truly random and have some sort of algorithm to produce the next number.

Code is available on github at this address: <https://github.com/NotoriousCrayon/Quantum_Encryption_Device>

# 

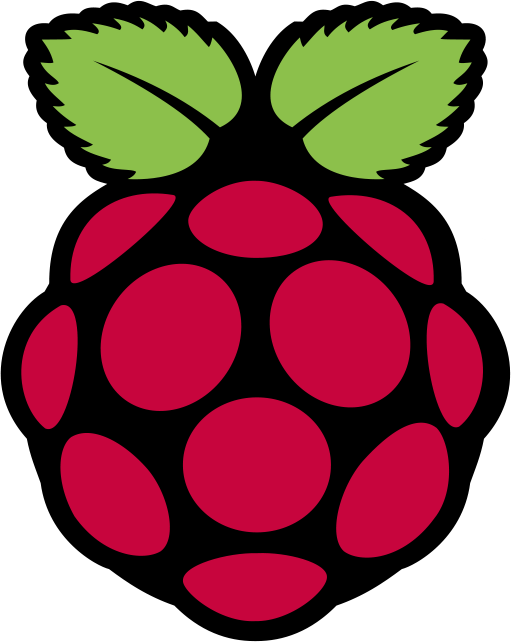
# Materials

Tip: Find links to all of these products in the README File of Github

Device List

1. Arduino Uno Microcontroller (with
2. Raspberry Pi 3
3. Raspberry Pi 3 2.5A Power Supply
4. Vernier Radiation Monitor
5. Vernier Arduino Interface Shield (SparkFun)
6. MicroSD (at least 8gb)
7. MicroSD to USB
8. USB cable A to B
9. Cesium - 137 Microcurie Disk Source
10. HDMI Cable
11. Monitor (needs HDMI output)
12. USB keyboard
13. USB Mouse

# Software

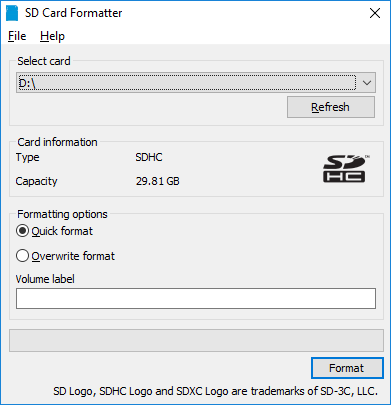


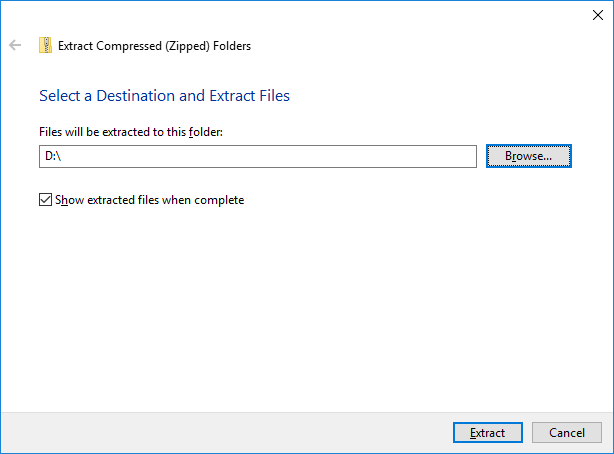
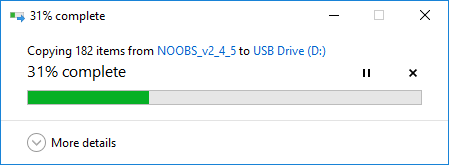
NOOBS on Raspberry Pi 3:

Your Raspberry pi is just a mini-computer. The specs are much lower than a typical household computer but they will be good for our project because it is so small. First we will Need to get an operating System for our Raspberry Pi. The easiest one to download is called NOOBS. To do this you will also need an SD formatter, download the files from here. Find the Updated Files on the README file in Github.

SD formatter : NOOBS for Raspberry Pi: <https://www.raspberrypi.org/downloads/noobs/>

SD Formatter: <https://www.sdcard.org/downloads/formatter_4/>

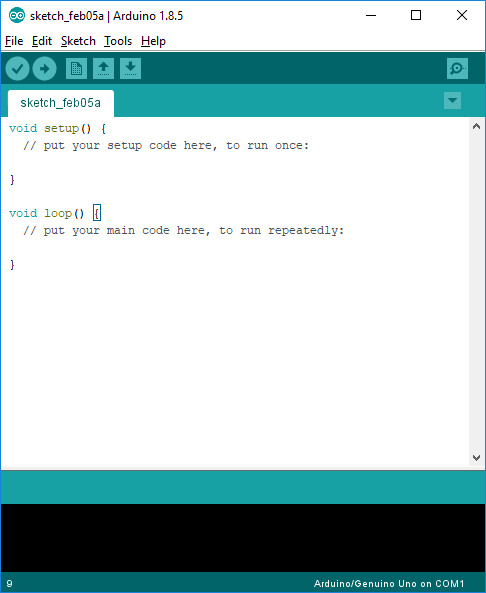
1. Plug your MicroSD card into your MicroSD to USB reader. Plug that into your computer
2. Open SD Card Formatter. Under “Select Card” Select your MicroSD card.
3. Click Format. (note this will erase everything on the SD card).
4. Right Click on the NOOBS Zip File (likely in Downloads Folder) and select “Extract All” - Extract to your newly formatted SD card.
5. Once the files are done transfering you will have installed the operating system on to the SD card. This will plug in on the back of the Raspberry Pi.



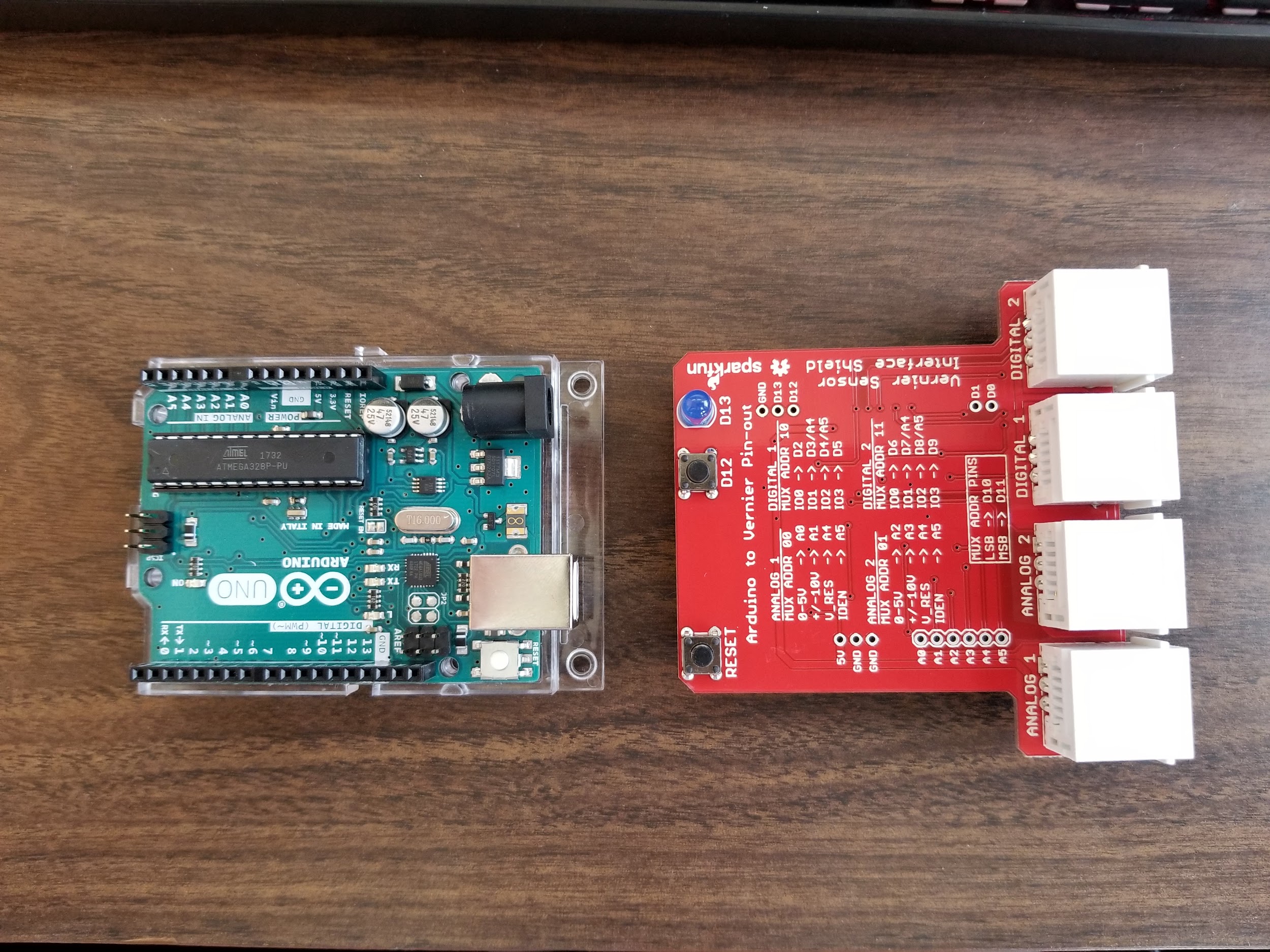
Arduino IDE and Set-Up

To Download the Arduino IDE follow this link: <https://www.arduino.cc/en/Main/Software> (Updated Version in README on github). Download according to your operating system (Windows, Mac OS, Linux). Follow the installation Instructions.

Once you have the IDE installed and run the program you’ll have something like this.

1. Select Tools > Board > Arduino / Genuino Uno
2. This is the Environment where you can edit code. Arduino is based on C, and has many tutorials on how to use it. However in this case we have provided the Code for you.
3. To upload the Code to the Arduino, select File > Open > VernerFile2
4. Then Compile the sketch by selecting the check mark in the top left.
5. Save, later when you need to upload the code to the arduino, just plug the arduino into your computer, open this code, and select the upload button (the arrow). On the board you will see the small LED blink a couple of times to let you know it has been uploaded.

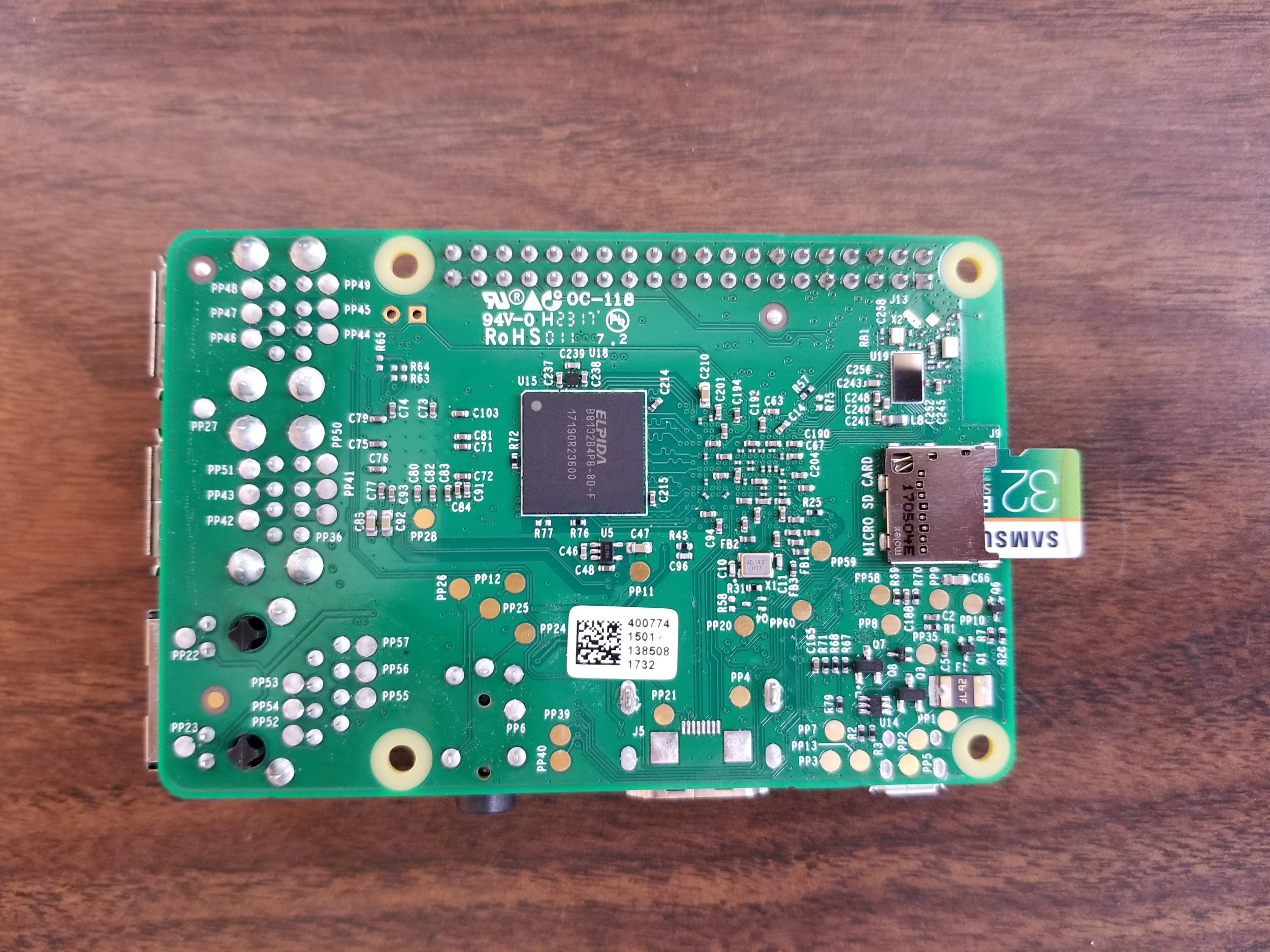
# Hardware



Arduino Set-up

1. Make sure you have uploaded the arduino sketch to the board (Connect the board via USB to computer, Start the Arduino IDE, open the code VernierFinal2.ino, and click the upload arrow button).
2. Disconnect the Arduino. Insert the Vernier Interface Shield into the designated pin inputs on the Arduino. (vernier input slots point to the back of the arduino, away from the usb plug)
3. Connect the Vernier Radiation Monitor to the “Digital 1” input.
4. Your Radiation source Cesium - 137 can be detected through the end of of the Vernier Radiation Monitor.

The Arduino is the part of this device that reads the radiation source. It then passes the information serially to the raspberry pi. This is done via the usb cable which also will provide the power to the USB.

Raspberry Pi 3 Set-up

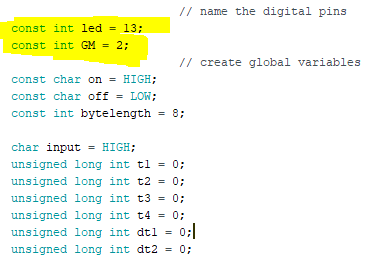
1. Connect your MicroSD card with the new operating system to the underside of the Raspberry Pi.
2. Plug The USB cable into the Raspberry Pi and Arduino.
3. Connect the 2.5A power supply into the micro USB slot near the HDMI port.
4. Connect HDMI into Raspberry Pi and Monitor
5. Connect USB keyboard to the Raspberry Pi
6. Connect the 2.5A power supply to the wall to start up the NOOB OS

# 

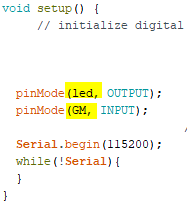
# 

# Code Explanation (and how this thing works!)

Arduino

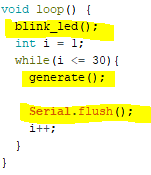
This portion of the code is where we are declaring our variables to use later on in the code. 

Pay close attention to the first two initializations, led and GM anre part of the Arduino. “led” is obviously an LED and GM is the Vernier Radiation Monitor. These are set equal to INT values that are equal to the pins on the physical Arduino board.

In the next portion of the code, you can see we use “pinMode” to set the variables led and GM as inputs or outputs. 

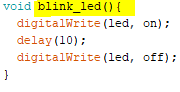
Most of the pins on the Arduino can be set to either an input or an output, so we have to set them so the board knows where to expect incoming data and what to do with outputs.

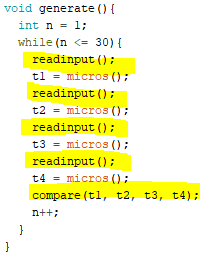
The Serial.begin portion is setting the baud rate (rate of serial transmission from arduino to raspberry pi). Here we have set it to 115200 baud, which is the maximum.



In this portion of the code we call a couple methods. “blink\_led( )” and “generate( )”. The “blink\_led” lets us know that our code is working by blinking the led that we declared in “void setup ( )”, and “generate( )” handles the incoming data from the Vernier Monitor.

“Serial.flush ( )” forces the arduino to wait for the transmission of data to the Raspberry Pi to complete before generating more numbers.

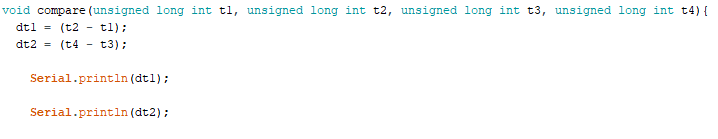
This is the method called in previous explanation. DigitalWrite sends the data to the declared variable (led, which we set to 13 where the led is on the board). Then it turns it on, waits 10 milliseconds and turns it off, creating a blinking.

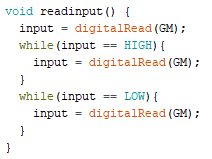
Here is the other method we called in “void loop ( )”. We set n = 1 and in the while loop we will continue looping until n <= 30. This will give us 30 readings.

The next method we call is the “readinput ( )” method, which will read the data incoming from the vernier radiation monitor.

Each variable is set to be measured in micros( ) or microseconds.

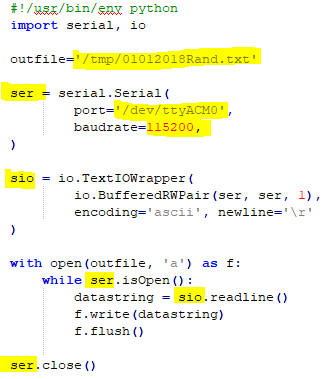
The Last method we call is compare ( ), which compares the data just as explained in the research paper which this model is based on. (see below to “void compare ( )”)

Note: Serial.prinln(dt1) or (dt2); is what is sending the data down to the Raspberry Pi.

Here we are reading the data incoming from the Vernier Radiation Monitor. There is a downtime in between readings which is what we are after (this is the random portion).

So we read the Vernier Monitor while it is on or HIGH and that saves in the method “generate ( )” as t1 (and then t2, t3, and t4). Then we find the difference in void compare ( ).

Raspberry Pi 3 Code

The import is how we call in other modules. Here we want to use serial and input/output to gather the data from the arduino.

Here we are setting the variable “ser” to take in the Serial data. The port is set to the arduino board, and the baudrate is the same as we have in the arduino

The “sio” is how we are going to read the data. Which is why we use the “io” module to state we have a pair of data coming in (dt1 and dt2 from arduino) we set each line to ascii and print on a new line for easier viewing.

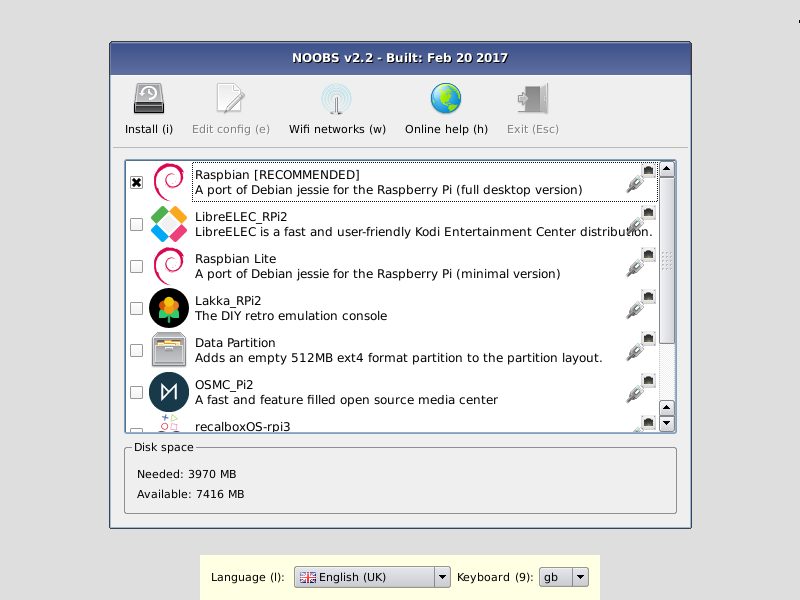
This last statement is opening the output file previous state and using .isOpen() to know when the data is incoming and uses the “sio” to see how we are going to read the data.

# Putting It All Together.

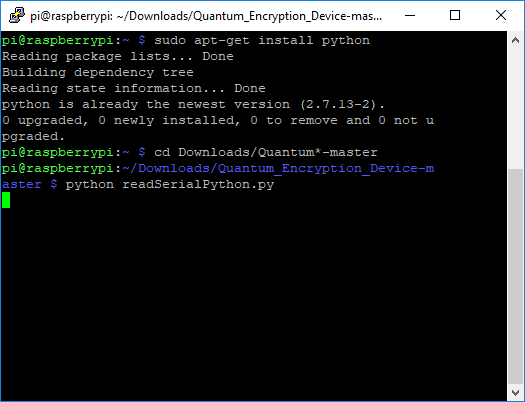
The Raspberry Pi

For this part of the tutorial you should have your raspberry pi connected to the monitor, keyboard, and mouse (as well as connected to the arduino setup via usb). The raspberry pi should be connected to the power supply and you will see the start-up checks scrolling through.

When prompted, select “Raspbian”

The first thing we need to do is to retrieve the python code for the Raspberry Pi to execute. You can find the Code on the GitHub page, or if you have it on a USB drive copy it and save it in “Downloads” through the Raspbian File System (just like on any other computer).

If you download from the github page, you will be downloading a .zip file, so make sure to extract the files (In this tutorial I everything was done in the Downloads folder but can be done from other folders as well.

Next Open the terminal by “CTRL + ALT + T” and run the following commands.

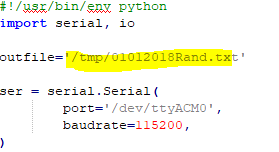
“sudo apt-get install python”

“cd Downloads/Quantum\*-master”

“python readSerialPython.py”

The program has now started!

To view the file that has all of the random numbers that we needed to generate we need to go to the output file mentioned in the python code.

As we can see here the file is located in the “tmp” directory.

To find the folder and view file type the following commands:

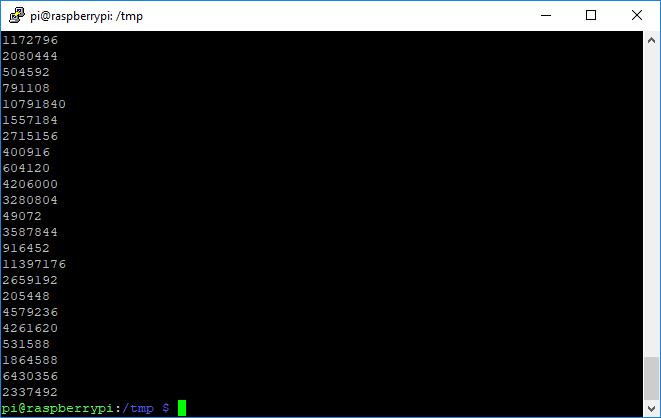
“Cd /tmp”

“ ls “

We can see the file here (my file date may be different than yours in the picture. To open the file and view the random numbers type the following command.

“ cat {filename}.txt”

Note: as our filenames might be different, use whatever yours is. For mine I would type “cat 02272017Rand.txt”



Now you have all the numbers that you will need to begin the Encryption which will combat Quantum computers. Congrats.

# Conclusion

Building the Quantum Encryption Device is just the first step towards a truly enabled quantum encryption. Implementing new encryption methods is a group effort in the Information Technology community. Encompassing more than just security professionals and end users, this is no small task. However it is important to keep building on the work that others are doing, so when quantum computing is readily available we also have an easy way to transmit data securely. With a bit more work, this project could be used to communicate with another user and send data between them with a random number generated key.

Hopefully this project will inspire more people to dive into quantum computing and this knowledge can be built upon.

short dash

1. Kunkel, Anthony B., "Hybrid Quantum Encryption Device using Radioactive Decay" (2017). Culminating Projects in Information Assurance. 31. <http://repository.stcloudstate.edu/msia_etds/31> [↑](#footnote-ref-0)