**SDR DBPSK GROUP FINAL REPORT**

**Introduction**

Our group worked on implementing a DBPSK system through GNURadio blocks in order to transmit files over a USRP. Our system takes bytes of a file, encodes each bit with an encoder, converts the information to a series of waveforms with a DPSK mod block sends the information either acoustically or with a USRP, reconverts it on the receiver side, and then scans for the proper access code in order to distinguish the signal from noise, and then writes the information to a file. A detailed explanation of the function of each block can be found in the documentation below.

**Meeting Notes**

**Monday 3/7**

* Gave presentation to Prof. Wang. It was...good :)
  + BPSK: show use of sinc/root-raised-cosine filter wave rather than square wave
  + Problem in BPSK: using square wave bit stream will result in sudden change in time-domain wave (180-turn), which results in huge amounts of noise in frequency domain between bits. Therefore, we need to use a sinc/root-raised cosine filter wave to reduce “transition noise.”
  + Understand use of PLL in PSK.
  + Root-raised cosine filter prevents “intersymbol” interference in time-domain (when bit waveform reaches zero, next bit waveform can begin; no overlap).
  + DBPSK/DQPSK: Needs “preamble” in system (send 6 low bits then 2 high bits to tell the system to begin receiving and the difference between low and high).
* Still need to understand and rebuild Simulink model.

**Friday 3/11**

* Met with Alejandro for him to explain the current Simulink model.
  + Problems:
    - Create PSK module
    - Improve FSK module (audio sink?)

**Monday 3/14**

* Lu, Charles, and Kevin
* Created simple phase modulation (successful!)
  + Needs sinc/clock signal for system to read data (distinguish between 0s and 1s) without feedback or memory
  + How to set “initial state”
  + Feedback not possible
* Look up (specifically) DPSK on Simulink

**Monday, 3/21**

* Met with Professor Wang. It was better!
* Understand features of each block (Gray code, excess BW, vector length of Throttle, etc.)
* Understand packet encoder (exactly how many bits are sent)
* Start working with audio sink.
* DPSK vs. PSK
  + Bandwidth of PSK modulation much longer than DPSK.

**Weddnsknadsday**

* Didnt do much
* Looked at spectrum analyser
* Got homework:loook into the psk blkc, actually know how it works

**Sunday, 3/27**

* Packet Encoder
  + Sends -92.0, 54.0, 32.2, 0.0 on Number Sink (when samp\_rate = 10)
  + Sends 13 extra values for each bit (when samp\_rate = 1000)
    - 18 values in one period, 13 are the same in each period
  + Compared two different values at once
    - 5 values are different, 13 are the same (shown in Time Sink)
    - Values in Number Sink switch at different times
  + Sending 0 to 127 results in positive value, >128 results in negative value (Number Sink)
  + Changing Payload Length to 2 (1 for automatic), sent one waveform with P.L. 2, and one with default value
    - 20 values in one period for P.L. 2 waveform
    - 18 values in one period for default waveform
    - 13 values in each waveform are still the same
  + Problems:
    - Don’t know how to set preamble (don’t see it in waveform)
    - Don’t see effect of samples/symbol, bits/symbol in waveform
* PSK Mod
  + Samples/symbol: makes curve “smoother”
    - Decreases “bandwidth” (works in smaller range of frequency)
  + Excess BW: controls “bandwidth” of waveform in frequency domain
  + Problems:
    - Don’t see effect of Gray Code, Differential Encoding on waveform
      * Because system only sends 1s and 0s, gray code will not matter (and possibly because of small sampling rate)

**Monday, 3/28**

* Met with Professor Wang
  + Packet encoder
    - Packet encoder outputs samples (discrete values)
    - Understand effect of “Samples/Symbol” and “Bits/Symbol” on waveform
    - Understand Access Code
    - “Bits/Symbol” related to QAM (repeats value in complex plane X times)
    - Preamble needs to be proportional to size of input
  + PSK Mod: Excess BW
    - Understand value put in; what value should we put in Excess BW for a desired bandwidth?
      * Relates to the Root-Raised Cosine Filter (f(t) = cos(alpha,t), alpha comes from Excess BW)
    - Understand Differential Encoding (send more than 1s and 0s)

**Wednesday, 3/30**

* Working communication between transmitter and receiver (few wrong characters, but still very successful).
* Successful when using DBPSK, however.

**Monday, 4/4**

* Met with Professor Wang
  + Understand blocks!
  + Test with one or two bytes, predict output with one or two bytes rather than long string.
  + Packet Encoder
    - If output is float value, then what is the point of using Constellation Points in PSK Mod?
* Send Professor Wang email (no meeting next Monday)

**Thursday, 4/14**

* Throttle rate
  + Runs application at maximum speed (which overheats the computer)
  + Controls the “rate” the application is running in
* Samples/Symbol
  + Is relevant only for decimation/interpolation of signal; in our case, we aren’t upscaling/downscaling, so Samples/Symbol must be equal in both modulation and demodulation.
  + May also be related to bit rate (128kbps vs. 20kbps)
    - Hardware decides “symbols/second” (from DAC)
    - samples/second divided by samples/symbol results in symbols/second
* Vector length
  + Specifies the vector length for vector processing
  + System outputs discrete values, complex, float, etc. (no vectors)
  + Ideally no effect on system.
  + Can only work when 1 (sending out (x), not (x,y,z))
* Packet encoder takes in number of bytes (payload length), sends out packet
  + Header: 2x repetition of the payload length (16 bits for each field)
  + Access code: need to match on decoder
  + Preamble: need to match on decoder
* Bit/Symbol
  + If we send 24 bits, the packet encoder will take 24 bytes (if bits/symbol = 1)
  + If we send 24 bits, the packet encoder will take 12 bytes (if bits/symbol = 2)
  + Need to test this (does payload length control this?)

**Monday, 4/18**

* Source code for Packet Decoder and Packet Encoder:
  + <http://gnuradio.sourcearchive.com/documentation/3.2.2.dfsg/packet_8py-source.html>
* Threshold for Packet Decoder
  + In source code, if threshold < 0, system takes “default threshold” (default threshold = 12)
    - Threshold: detect access code with up to “threshold” bits wrong
    - If system detects more than 12 errors in access code (or preamble?), will not output.
* Distributed MIMO (multiple input, multiple output)
* Samples/Symbol
  + <https://www.ee.oulu.fi/research/ouspg/psk_demod.py>
  + Sampling rate 125 kHz is used in the script by default and can only be changed from the source code (decimation values). Samples per symbol = 32 would correspond to 125/32 ≈ 3.9 ksym/s symbol rate or 3.9 kbit/s bitrate in 2-PSK.
* Need to understand Phase BW, Timing BW, and Frequency BW on PSK Demod block.
* PSK Demod needs a “frequency component.” Need to understand how PSK Mod block outputs a “frequency component.”
* Eventually turn “Differential Encoding” on PSK Demod off (bit rate goes down when turned on).
* Packet Encoder to PSK Mod
  + “Bits/Symbol” and “Number of Constellation Points” need to match
  + Samples/Symbol on both blocks need to match
  + Need to understand effect of “Bits/Symbol” in Packet Encoder
  + Need to understand output of packet encoder (seeing access code, preamble, etc.)

**Sunday, 4/24**

* Testing the Packet Encoder
  + **Observing Preamble, Access Code, etc.**
  + Preamble: bunch of 1s with 2 0s
  + Access code: bunch of 0s with 2 1s
  + Screenshot of Time Sink on “testEncoder” file on desktop
  + Sent Constant Source input of 0; saw bunch of 0s, 1,0,1, bunch of 0s
  + Preamble -> access code -> data
  + *Was able to see preamble, access code, data, and header in waveform (in “0” and “1” png files)*
  + Waveform depends on Sampling Rate of system, so did not thoroughly examine waveform point-by-point; when sampling rate of system is decided, will thoroughly look at it.
  + **Observing “Samples/Symbol”**
  + Changed “Samples/Symbol” from 1 to 2 to 8 (screenshots saved); saw no effect on waveform
  + **Observing “Bits/Symbol”**
  + Changed “Bits/Symbol” from 1 to 2 to 8 (screenshots saved); saw no effect on waveform
  + Changed Payload Length to 8, then changed “Bits/Symbol;” same waveform, just phase-shifted (possibly slightly longer data)s
  + Changed “Bits/Symbol” and tested if modulation/demodulation still worked with text file; text file still transferred correctly even with “incongruent” settings
  + Since Packet Encoder will take 8 bits of data per packet, changing “Bits/Symbol” won’t change anything.
  + If we send 24 bits, the packet encoder will take 24 bytes (if bits/symbol = 1)
  + If we send 24 bits, the packet encoder will take 12 bytes (if bits/symbol = 2)
    - How do we see how many bytes the Packet Encoder is sending?

**Monday, 4/25**

* Need to send in report
  + What we did during semester (lab log, etc.)
  + What we learned
  + Advanced documentation on several GNURadio blocks (what does each feature on Packet Encoder does)
* Source code on making packet
  + <https://sourcecodebrowser.com/gnuradio/3.0.2/namespacegnuradio_1_1packet__utils.html#a6444ba5b431817c6e476f17a1737498f>
  + <https://sourcecodebrowser.com/gnuradio/3.0.2/packet__utils_8py_source.html>

**SDR PSK-Transmission Group Final Report**

Project Summary

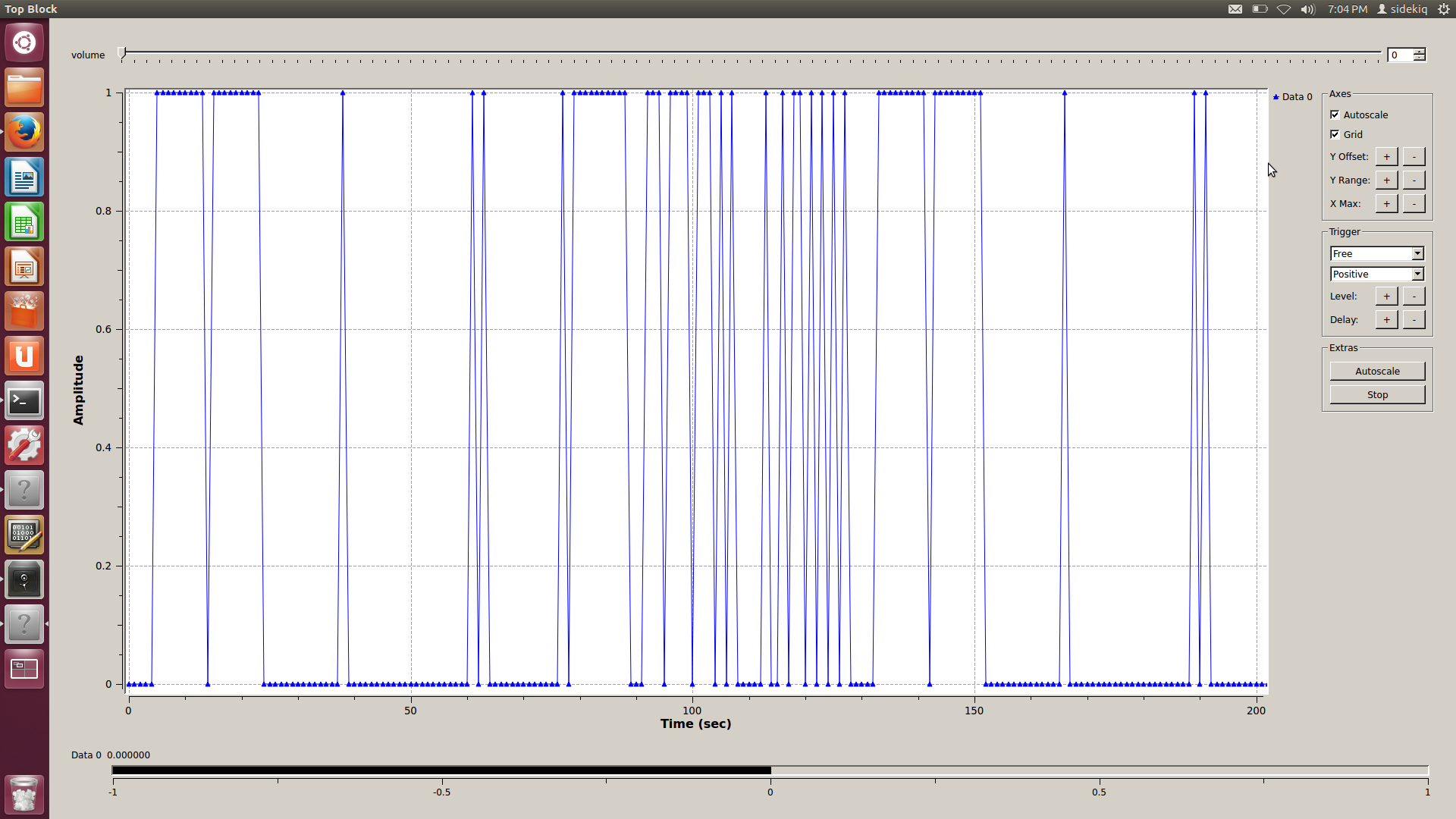
In this semester, our team focused on the Phase Shift Keying transmission research and successfully implemented the system. Besides, we digged deep into the function of each block and the exact meaning of every settings of the blocks. Followed below are our documentations of our research, and what we learned throughout the semester.

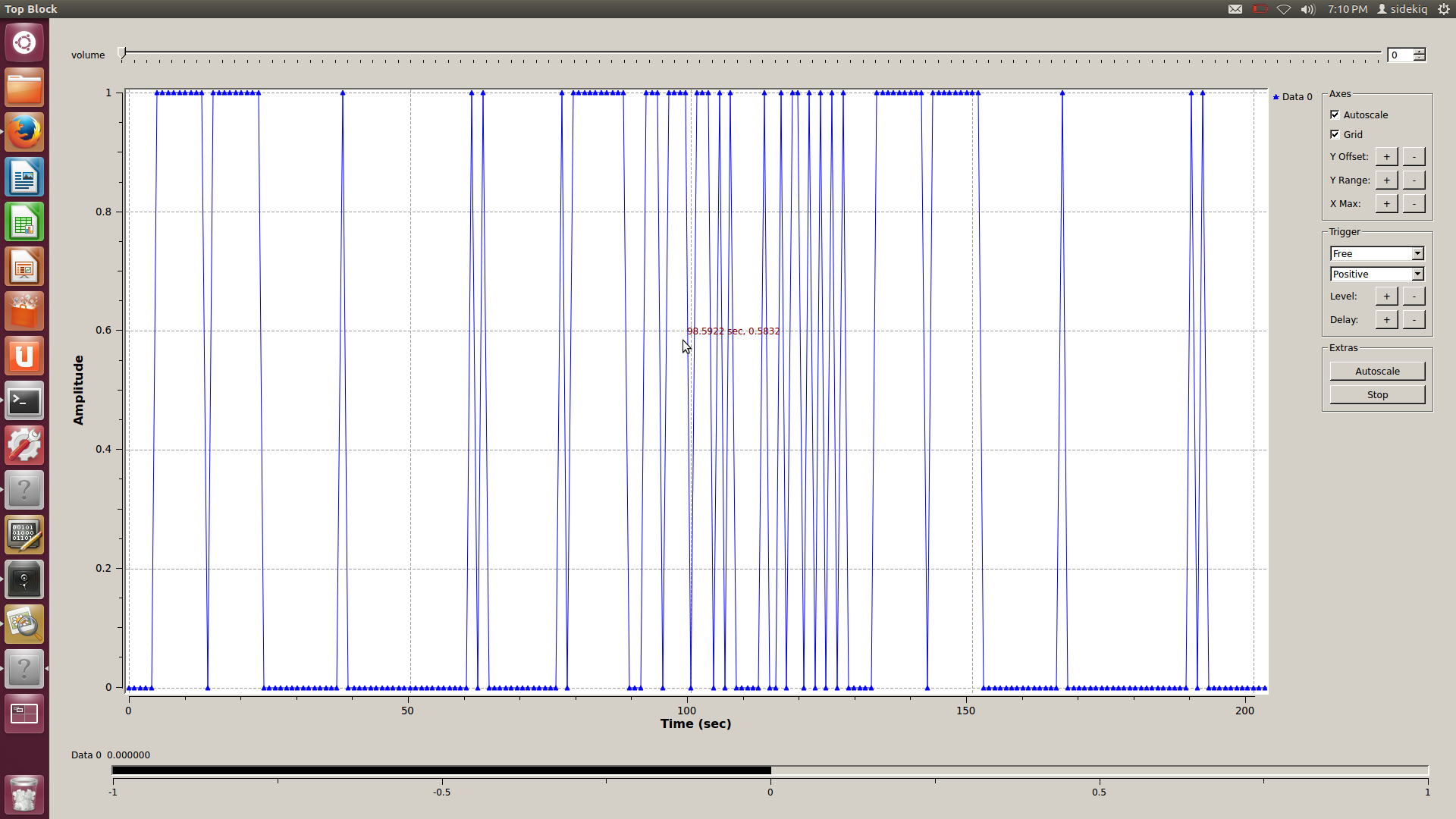
Advanced Documentation

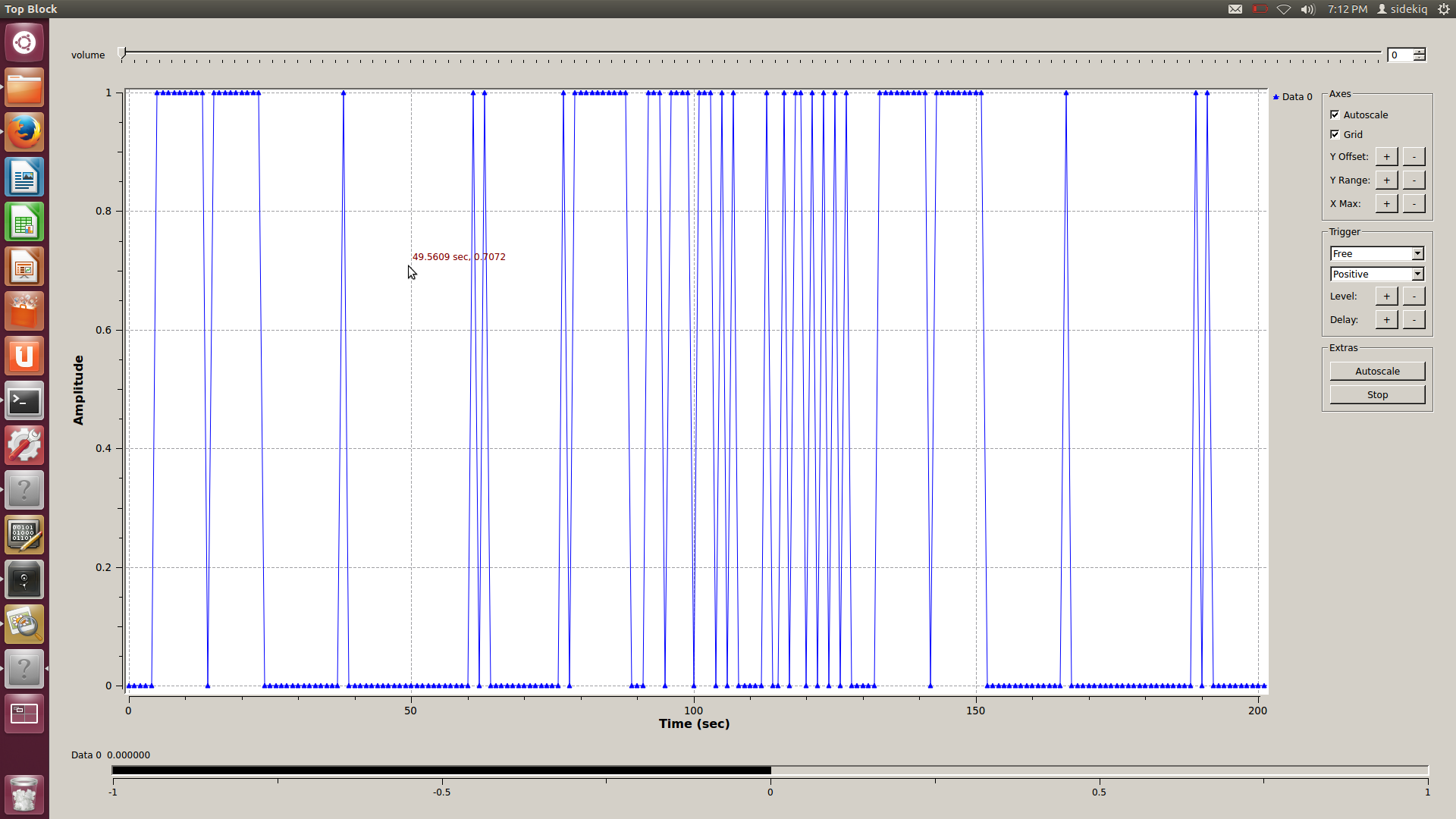
**File Source:** Takes a selected file and outputs the file information as a series of bytes

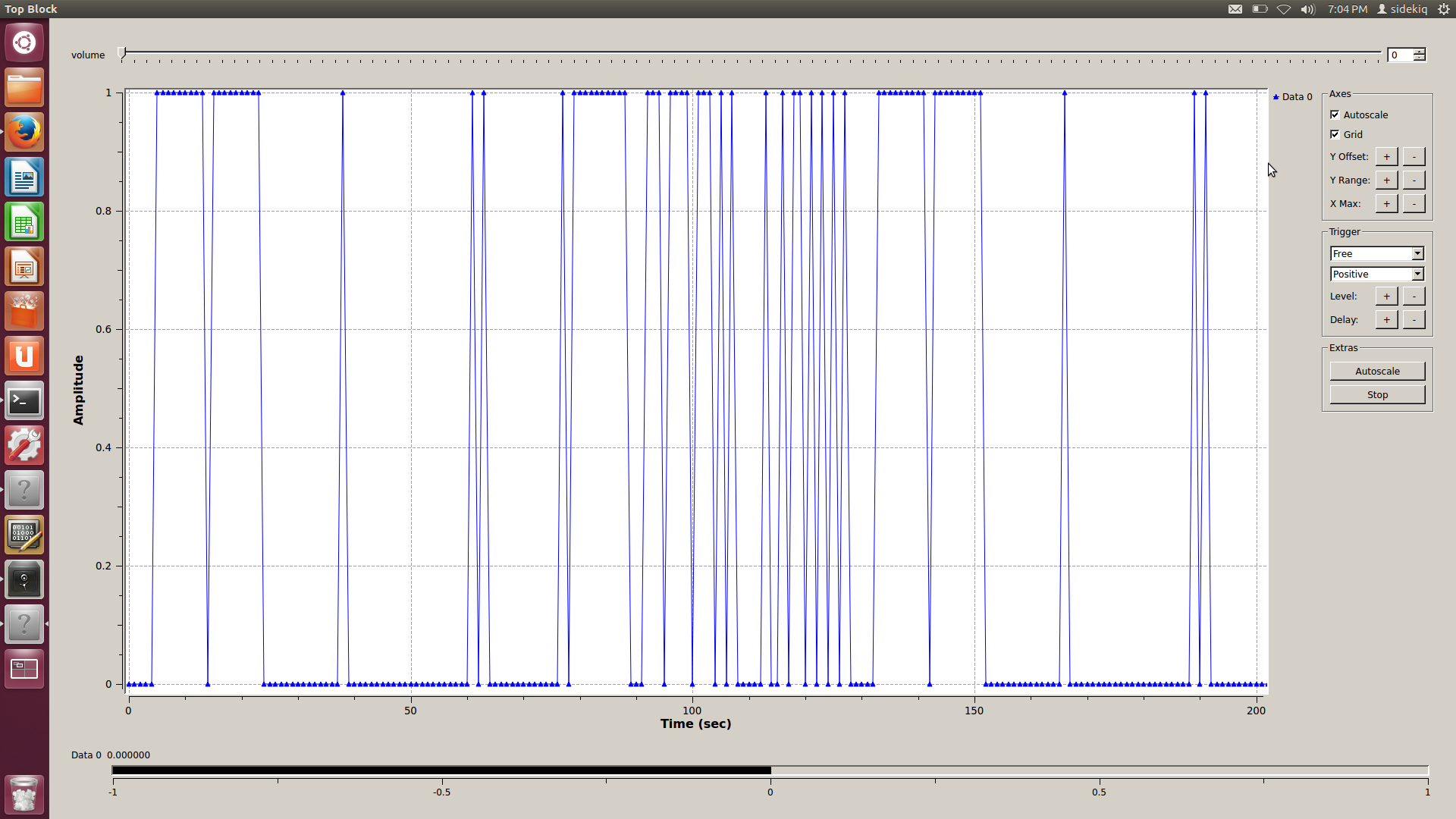
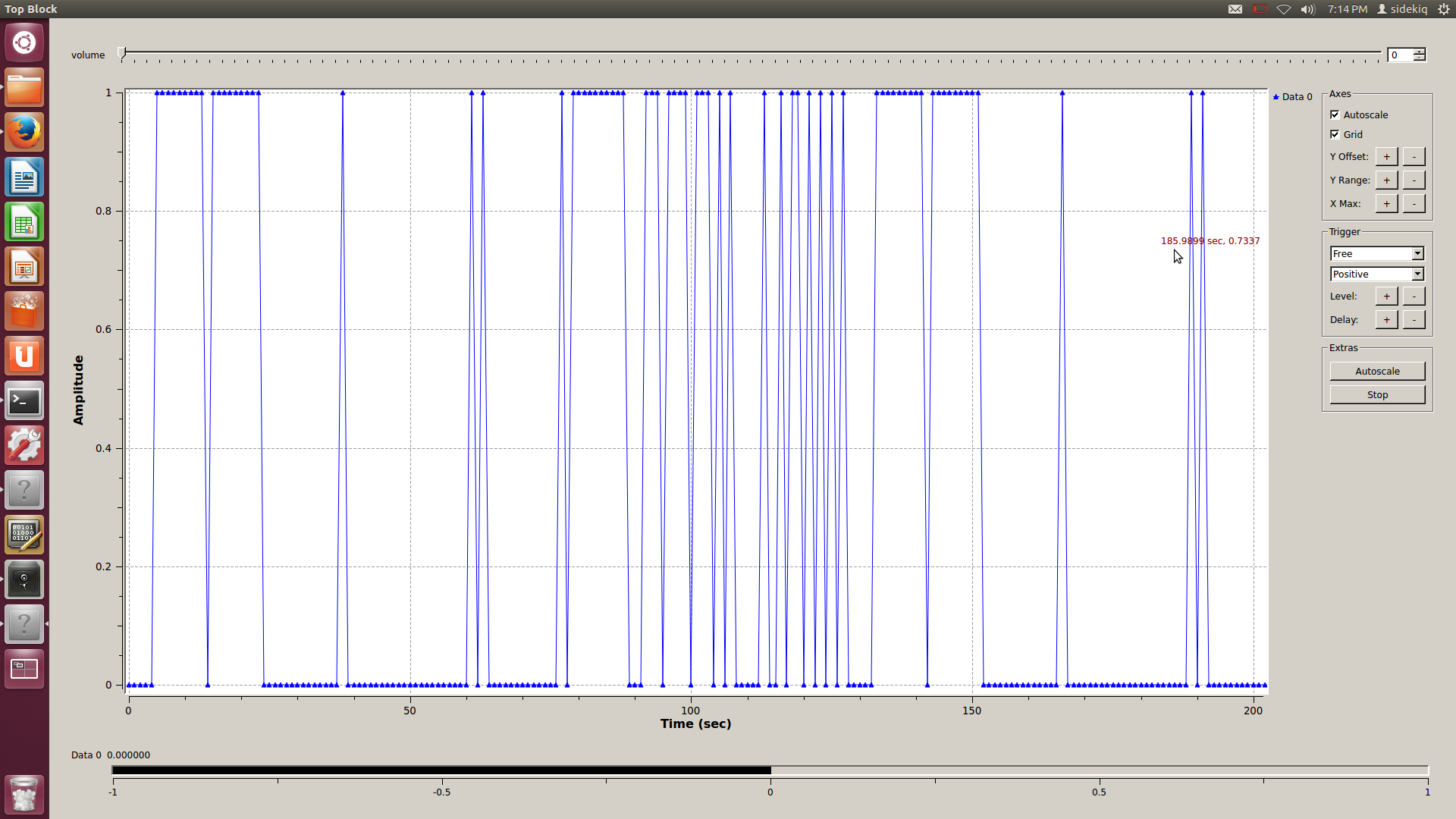
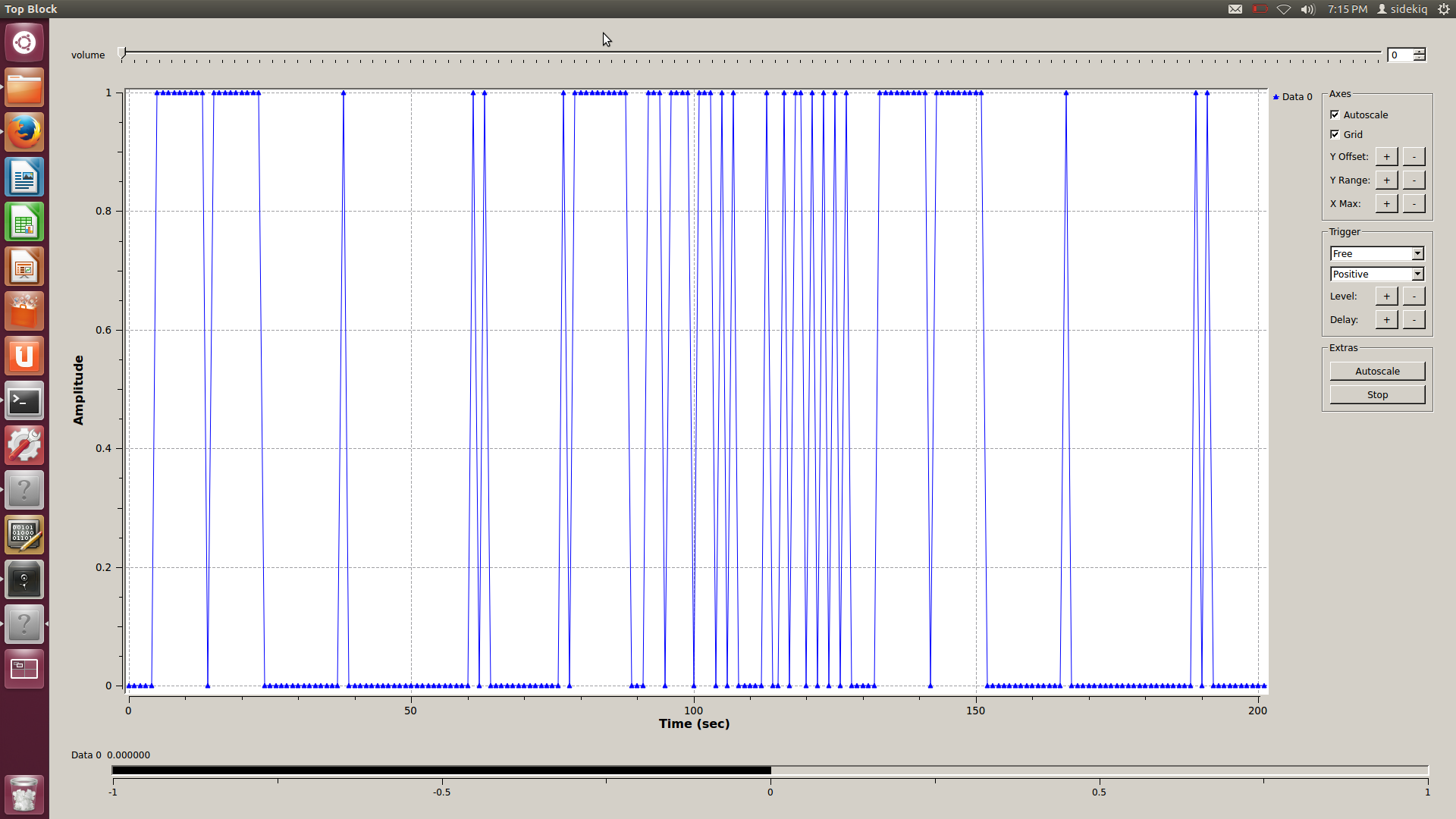
**Packet Encoder:**

* Samples/Symbol: This allows you to set the symbol rate, as the block has a coded frequency of symbols per second. Samples per symbol specifies the number of samples dedicated to each symbol.
  + Testing effect of Samples/Symbol on waveform
  + Set “Samples/Symbol” to 0:



* + Setting “Samples/Symbol” to 1:
  + Setting “Samples/Symbol” to 8:



* + Conclusion: setting Samples/Symbol had no effect on transmission. However, was tested using only one laptop, so remote transmission and turning on USRP will matter.
* Bits/Symbol: Allow users to set how many bits are contained in one ‘symbol’. For example, for BPSK, as one symbol only contains one bit, we should set to one. For QPSK, as we are sending 2 bits in one symbol, we’ll set it as 2.
  + Testing effect of “Bits/Symbol” on waveform
  + Setting “Bits/Symbol” to 0:
  + Setting “Bits/Symbol” to 2:  
    
  + Setting “Bits/Symbol” to 8:
  + Conclusion: setting Bits/Symbol had no effect on transmission. However, was tested using only one laptop, so remote transmission and turning on USRP will matter.
* (Bits/Symbol and Samples/Symbol are only useful while Pad for USRP is turned on)
* Preamble: The preamble is a series of bits added to the beginning of the bit to be sent. If no value is given, it is set to a default.
* Access code: The access code is a series of bits which is added in front of the bit to be sent, but behind the preamble, if no value is given, it is set to the default
* Pad for USRP: Pads data until string is a multiple of 128 bits. Allows use for USRP.
* Payload length: The actual “data.” Increasing Payload Length will increase the number of bits assigned to the input data. (Ex: 5 unique discrete values for 0 when Payload Length = 1 but 7 unique discrete values for 0 when Payload Length = 2).

**DPSK Mod:**

* Can select between DBPSK and DQPSK (Transmitter works when DBPSK is selected).
* Samples/Symbol: Samples per symbol specifies the number of samples dedicated to each symbol.
* Excess BW: Not a major concern for system
* Gray code: allows for gray code binary system in transmitting data. Recommended for reducing noise (have not experimented this).

**Unpack K bits:**

* This block picks the K least significant bits from a byte, and expands them into K bytes of 0 or 1.
* Example: k = 4 in = [0xf5, 0x08] out = [0,1,0,1, 1,0,0,0] (Binary 5 and binary 8)

**DPSK Demod**

* Samples/symbol: Should match the transmitter’s setting.
* Excess BW: Told to disregard for now
* FLL BW: Disregarded; will understand later in project
* Phase loop BW: Disregarded; will understand later in project
* Timing BW: Disregarded; will understand later in project
* Grey Code: different binary system which only changes one bit for each iteration. Helps with noise
* Sync Out: Disregarded; will understand later in project

**Packet Decoder**

* Access code: This is the series of bits the encoder will look for in order to accept the subsequent bit as a proper bit. Needs to match the packet encoder access code being used for the transmission
* Threshold: This sets the number of errors accepted by Decoder before disabling output. When Threshold < 0, sets to default value.

**What we Learned**

Zijin Lu

* Learned the basic principle of remote digital transmission using Phase Shift Keying, and different forms of PSK.(Higher transmittion rate: QPSK, 8PSK…. PSK with differential encoding: DPSK)
* Learned the difficulties to implement the practical remote data transmission(need to consider frequency, error, access code, preamble…)
* Learned the exact function of each block in our system and what the settings do to the output.
* Learned how to research the function of blocks using the Python source code online.

Charles Li

Kevin Choi

* Learned different methods of data transmission (phase shift keying, frequency shift keying, etc.) as well as the various sub-methods of phase shift keying (differential, binary, quadrature, etc.). Was very valuable in understanding how radios work.
* Worked with GNURadio and Simulink and observing effects on signal in each block. Overall, finding new ways to observe and test the signals.
* Learned practical parts needed in data transmission, i.e. use of a preamble, access code, payload length.
* Finding Python source code of the GNURadio blocks and understanding what happens at the basic level.
* Was able to see things mentioned in ECE 301 being applied and used.