Popoto **Modem**

Application Note AN002
Streaming Data

delResearch LLC

Document Versions

Version	Purpose	Author	Sw Ver	Date
1.00	Initial version	Jim DellaMorte	2.7.0	5/28/2021

Contents

0.1 References

Document	URL
Popoto PMM3511 User's Guide	https://www.popotomodem.com/wp-content/uploads/2021/05/PopotoUsersGuidePMM3511.pdf
Popoto PMM5021 User's Guide	https://www.popotomodem.com/wp-content/uploads/2021/05/PopotoUsersGuidePMM5021.pdf
Popoto Python API	https://www.popotomodem.com/wp-content/uploads/2020/07/PopotoAPI.pdf

1 Overview

This application note covers the use of PopotoModem for streaming data applications. In a streaming data scenario, data is continuously provided to the Popoto Modem, and that data is packetized, modulated and transmitted to the receiving modem.
Figure 1 shows the setup for this application note. In this setup there are 2 laptop computers connected to PopotoModem devices (M2000 depicted in the diagram, but any Popoto Modem PMM, M, or S series modem will work). The laptops are running the provided python scripts to send generated data from the source modem on the left to the sink modem on the right.

Figure 1.1: Streaming Data setup with 2 Popoto Modems and 2 laptops. The modems are connected via ethernet.

Two python scripts are provided, `StreamingTx.py` and `StreamingRx.py`. These 2 scripts setup the modems for streaming operation, and the StreamingTx.py script generates data and sends it at maximum bitrate to the StreamingRx.py script, which displays the data on the screen. This setup demonstrates the following functionality.

- Setting up the Modems for transparent transmission
- Connecting to the Data socket
- Setting appropriate transmission block lengths
- Receiving data
- Setting Power levels for transmission

2 About Streaming On Popoto

In the Popoto System, all transmissions begin with a Frequency Hopped acquisition followed by a Frequency Hopped (FH) header. These Frequency hopped signals are modulated at 80 bits per second, and they setup the transmission parameters of the optionally attached cargo packet. Refer to Figure 2, this figure shows a single PSK packet sent by the Popoto Modem.

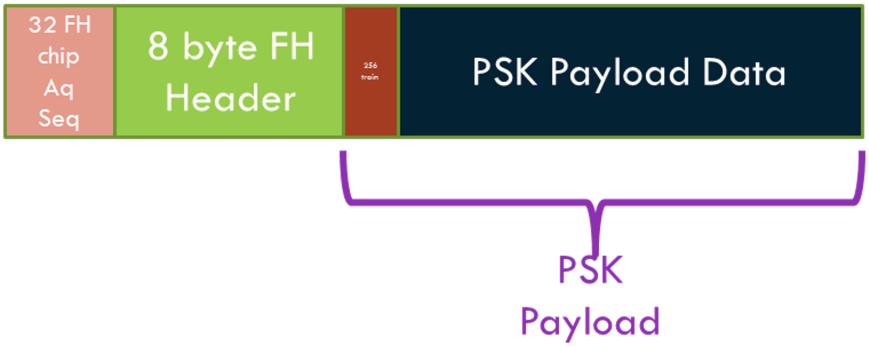

Figure 2.1: A single PSK Payload Packet

In streaming mode, (shown in Figure 2) A single FH acquisition and header sequence is shared among several PSK Packets. This allows the modem to have increased throughput, as the overhead associated with the Frequency hopped header is amortized over several PSK packets.

Talk about Variables here TxStreamingLen etc. etc.

Figure 2.2: PSK Data Sent in Streaming Mode.

3 Test Setup

3.1 Requirements

3.1.1 Required Hardware

In order to run this application note, you will need the following hardware setup as depicted in Figure 1.

- Two Laptop or Desktop computers with ethernet interfaces
- An ethernet switch or router
- Two Popoto Modems configured with ethernet interfaces. These can be deckboxes, OEM units, or enclosed modems. Note that some enclosed modems do not bring the ethernet interface outside of the bottle. Check with Popoto Modem if you are unsure about the type of modem that you have.
- Appropriate ethernet cables.

3.1.2 Required Software

The following software is required to be installed and running on the laptop/desktop computers used for running this application note:

- A serial terminal (i.e. Teraterm, Serial, Minicom, etc.)
- Python 2.7
- popoto.py
- StreamingTx.py
- StreamingRx.py

4 Example Code

The following source code listings show the steps necessary to enable streaming mode and to transfer data. These files are intended to be a starting point for development of a customer's application. The configuration provided in the script file represents the optimal settings for streaming operation. It is not recommended to change these configuration items.

4.0.1 StreamingTx.py

The following code listing shows the steps required to transmit streaming data on the Popoto modem.

```
from popoto import popoto
import time

# Select Hardware Modem or Virtual Modems
hardware = True

if hardware:
    # Connect to the Unit for transmission
    # Here our IP Address is 10.0.0.242  Change to
    # your IP address as necessary
    p1 = popoto('10.0.0.242',17000,0)
else:
    p1 = popoto('localhost',17500,0)

# Connect to the streaming Data socket.
p1.openDataSocket()

# These are the optimal packet sizes for streaming operation.
# No need to changes these
PacketSize = 256
StreamingLen = PacketSize*8

p1.drainReplyQ()


# Set the timeout value for transmission if we don't amass a full
# packet.   This is an important value if you are using a telnet script
# but for streaming set it to a large value to effectively disable the
# transmit on timeout feature
p1.setValueI('ConsoleTimeoutMS', 10000)

# Set the size of the individual PSK Packets
p1.setValueI('ConsolePacketBytes', PacketSize)

# Select the Raw Data Port mode.    1 = Raw Byte 0= Telnet filtering
p1.setValueI('DataPortMode', 1)

# Set the Bitrate to 5120
p1.setValueI('PayloadMode',1)

# Set a nominal transmit power (4 Watts is a good level for close operation or an airtest)
p1.setValueF('TxPowerWatts',4)

# Set the bytelength of the "Super Packet" of PSK packets
# This is the number of bytes to transfer before sending a FH
# Header again.
p1.setValueI('StreamingTxLen', StreamingLen)


check = p1.replyQ.get()
while(not any('StreamingTxLen' in str(s)for s in check.items())):
    check = p1.replyQ.get()
    time.sleep(0.25)
```

```
TxMsg = [int(n // 256) for n in range(StreamingLen)]
# Send first message
p1.datasocket.sendall(bytearray(TxMsg))
print('sending message 0')

npackets=20
for i in range(1,npackets):

    # Build a test message of streaminglen (2048 bytes) with least digit indicating packet number, other digits indicating message number
    TxMsg = [int(10 * i +n//256) for n in range(StreamingLen)]

    # Send subsequent message
    p1.datasocket.sendall(bytearray(TxMsg))
    print('sending message', i)

print(' ')
print('All buffers sent to socket')
#  Exit on key press
raw_input(' ')
p1.exit()
exit()
```

4.0.2 StreamingRx.py

The following code listing shows the steps required to transmit streaming data on the Popoto modem.

```
from popoto import popoto
import sys
from select import select

import time

def kbhit():
    dr,dw,de = select([sys.stdin], [], [], 0)
    return dr <> []


#Using real hardware or the simulation
hardware = True

# Open the receiver port
if hardware:
    p2 = popoto('10.0.0.241',17000,0)
else:
    p2 = popoto('localhost',18500,0)


p2.openDataSocket()

PacketSize = 256
StreamingLen = PacketSize*8

p2.setValueI('DataPortMode', 1)
p2.receive()

time.sleep(2)

while(~kbhit()):
        try:
                RxMsg =  p2.datasocket.recv(256)
        except:
                continue
        if(len(Rxsg)>0):
                print(RxMsg)
                RxMsg=''

p2.exit()
quit()
```

4.0.3 Returning to Data Packet Mode

To return back to normal packet operation, simply set the StreamingTxLen variable to 0.

```
f
# Set the bytelength of the "Super Packet" of PSK packets
# This is the number of bytes to transfer before sending a FH
# Header again.
p1.setValueI('StreamingTxLen', StreamingLen)
```