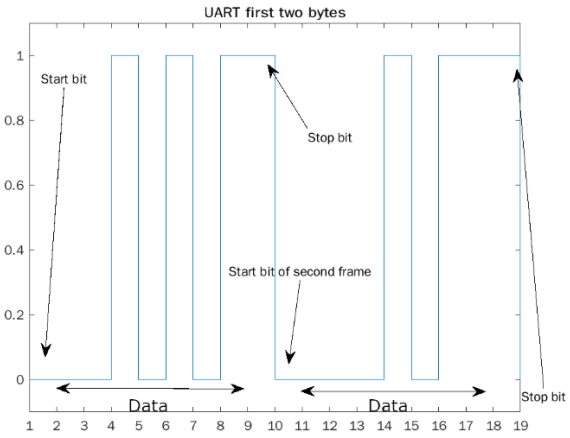
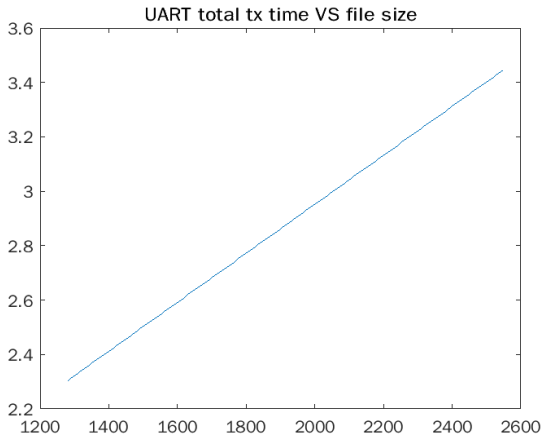
UART

In this protocol every character is converted to 7 binary bits to form a stream of bits for the whole file then we divide them into groups of 7 or 8 bits according to the required number of data bits. We do zero padding for the remaining non data bits in the last frame.

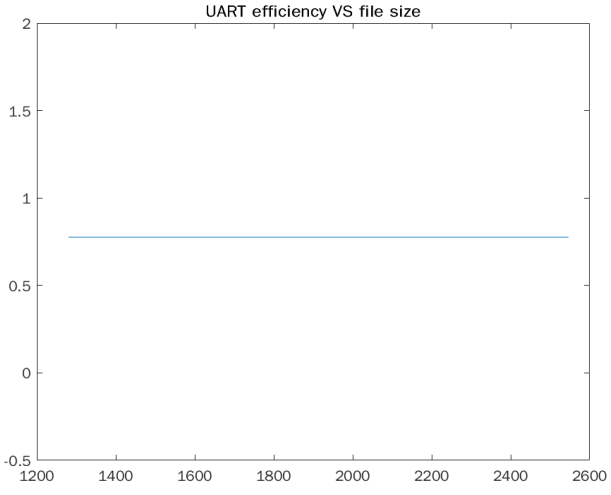
Some cases for UART

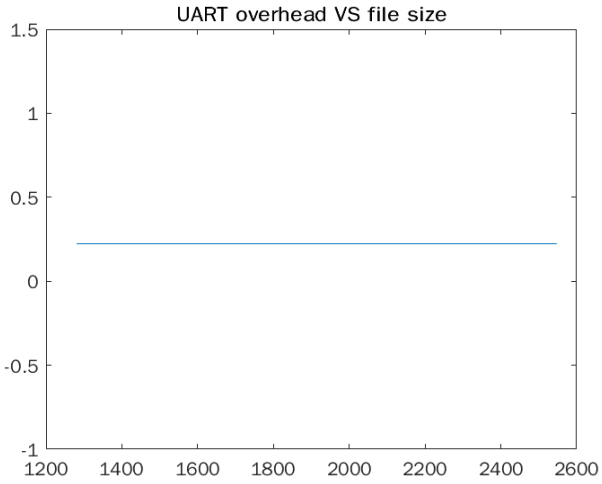
For 7-bit none parity and one stop bit





- As size of the data decreases in the packet the speed decreases. Therefore, the time for sending 7-bit is slower than 8-bit with same number of stop bit and parity.



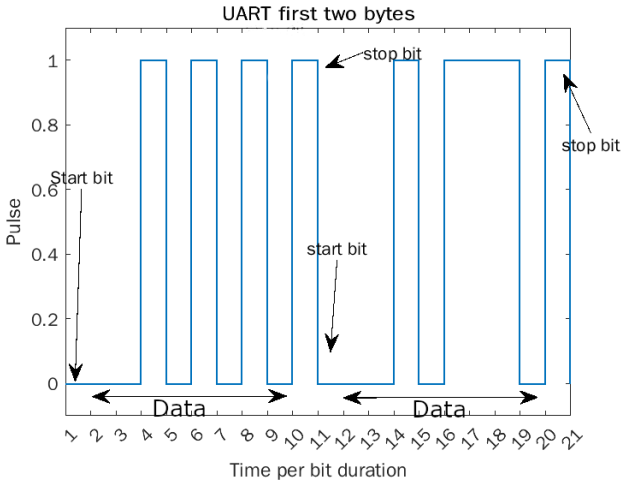


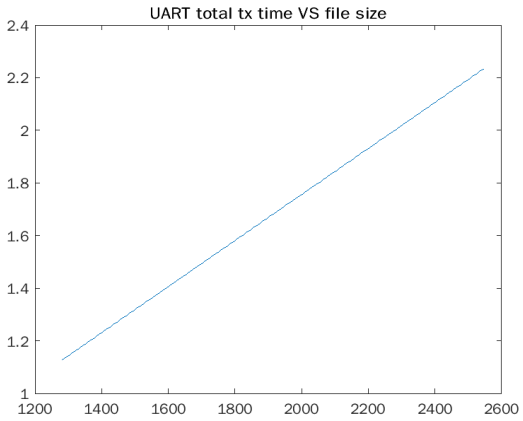
- Since, we send 7-bit data, and the character is 7-bit then each packet is responsible for one character and no data will be shifted to the next packet. Hence, the overhead and efficiency are constant regardless the number of characters we want to send.

  Overhead for the whole file (Overhead of one frame) $= \frac{2}{9} = 0.2222$

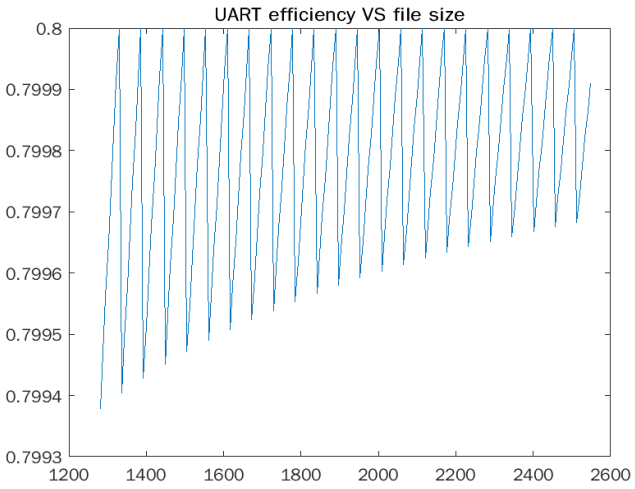
  efficiency for the whole file (efficiency of one frame) $= \frac{7}{9} = 0.7778$

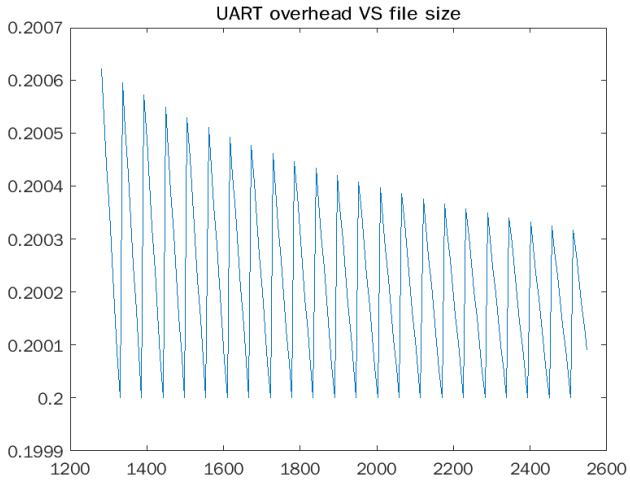
For 8-bit none parity and one stop bit





- Each packet Framed with start bit at the beginning and stop bit at the end and the data in the middle
- The Time is directly proportion with file size, since total time to transmit is file size in binary * bit duration.



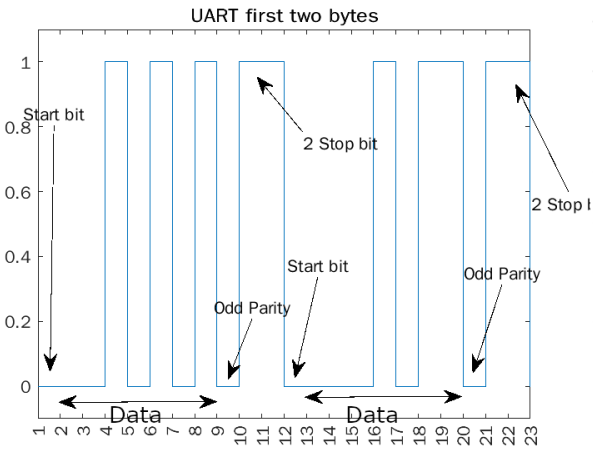


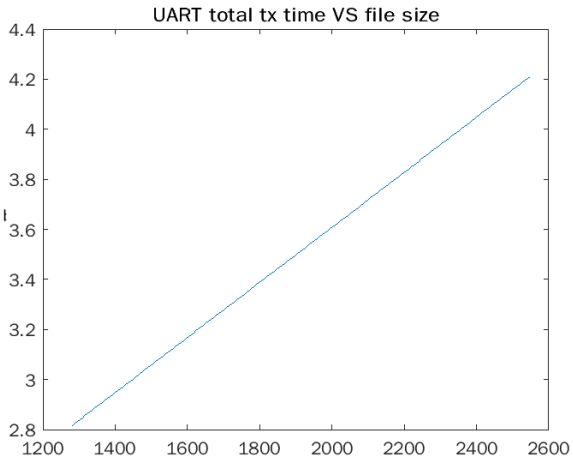
- The fluctuation for efficiency and overhead is because the character is 7-bits while the frame has 8-bits. Therefore, as data increases the last frame will contain some empty bits and the number of empties is changing with increasing the data, and when the more characters are streamed the packet tends to be full. So, overhead decreases and efficiency increases.

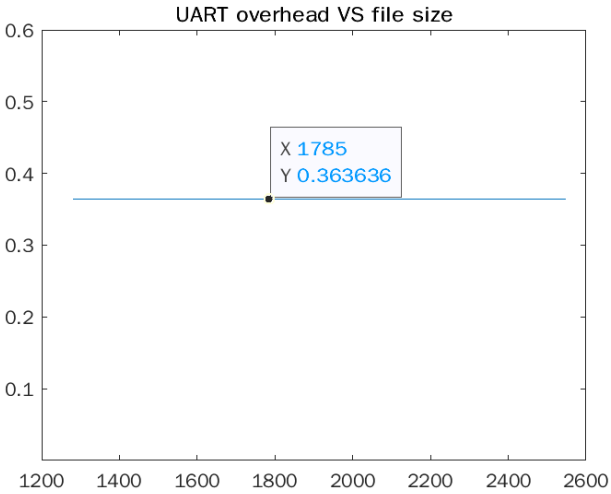
$$\text{Overhead for the whole file} = \frac{1120frames\ (1(stop) + 1(start))}{1280 * 7 + 1120frames\ (1(stop) + 1(start))} = 0.2$$

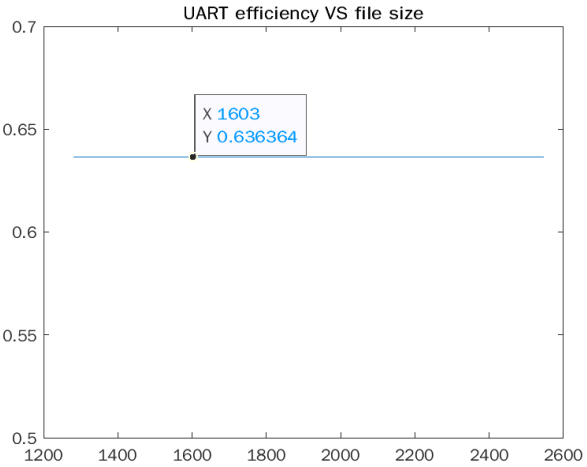
$$\text{efficiency for the whole file} = \frac{1280 * 7}{1280 * 7 + 1120frames\ (1(stop) + 1(start))} = 0.8$$

For 7-bit odd parity and two stop bits







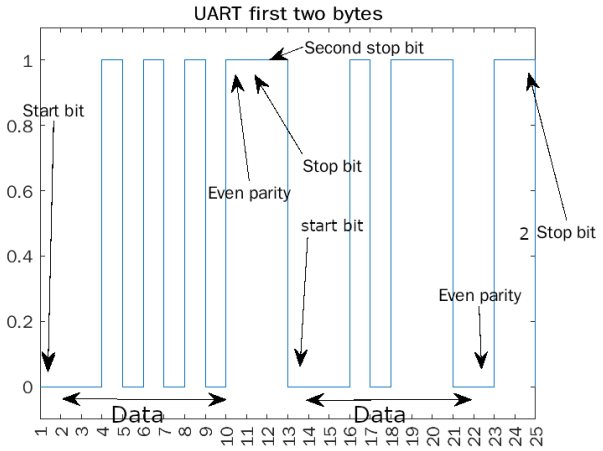


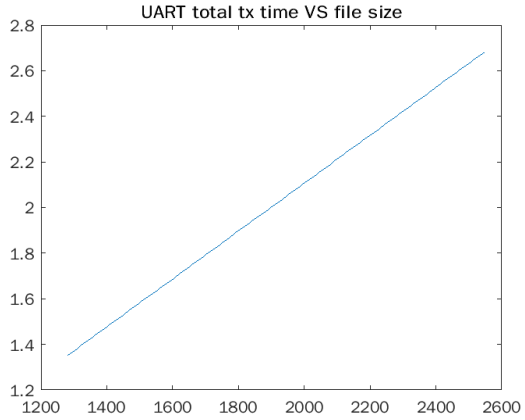
- The time is more than the previous time for sending 7-bit with no parity and one stop bit.
- The overhead is larger, and the efficiency is smaller
- The parity at first and packet are low because the sum of all ones are odd(3 in both).

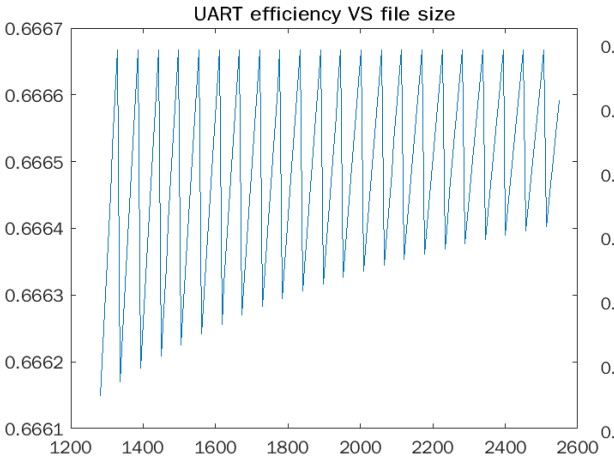
Overhead for the whole file (Overhead of one frame) $= \frac{4}{11} = 0.363636$

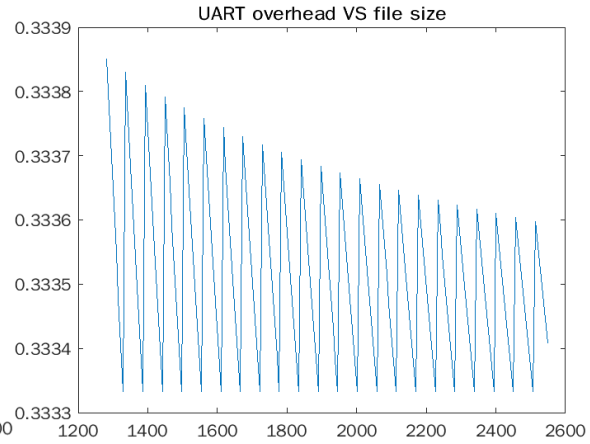
efficiency for the whole file (efficiency of one frame) $= \frac{7}{11} = 0.636364$

For 8-bit even parity and two stop bits







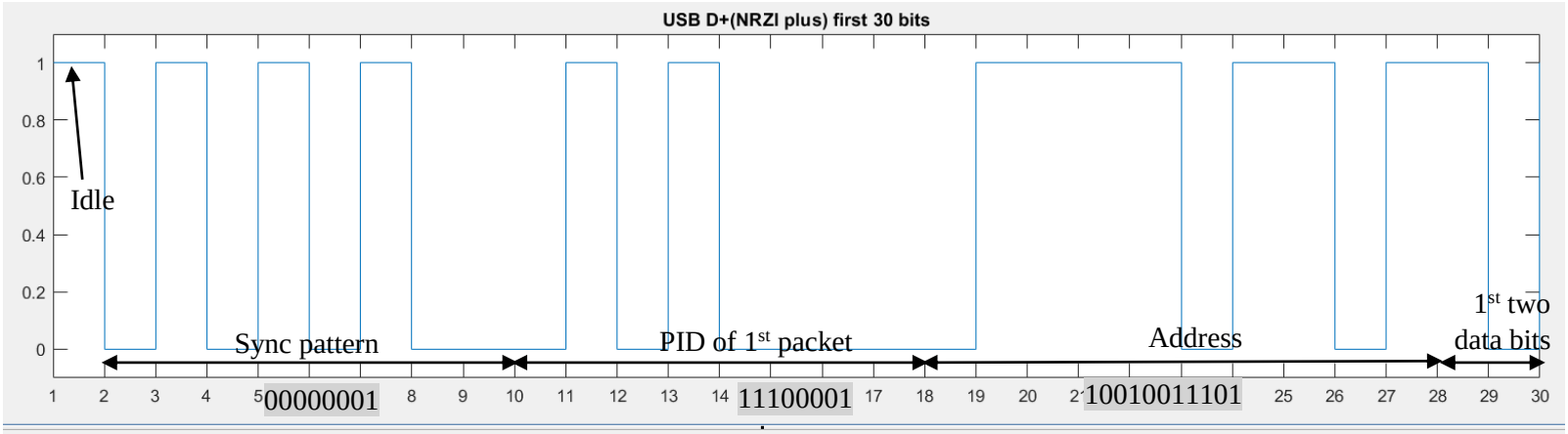


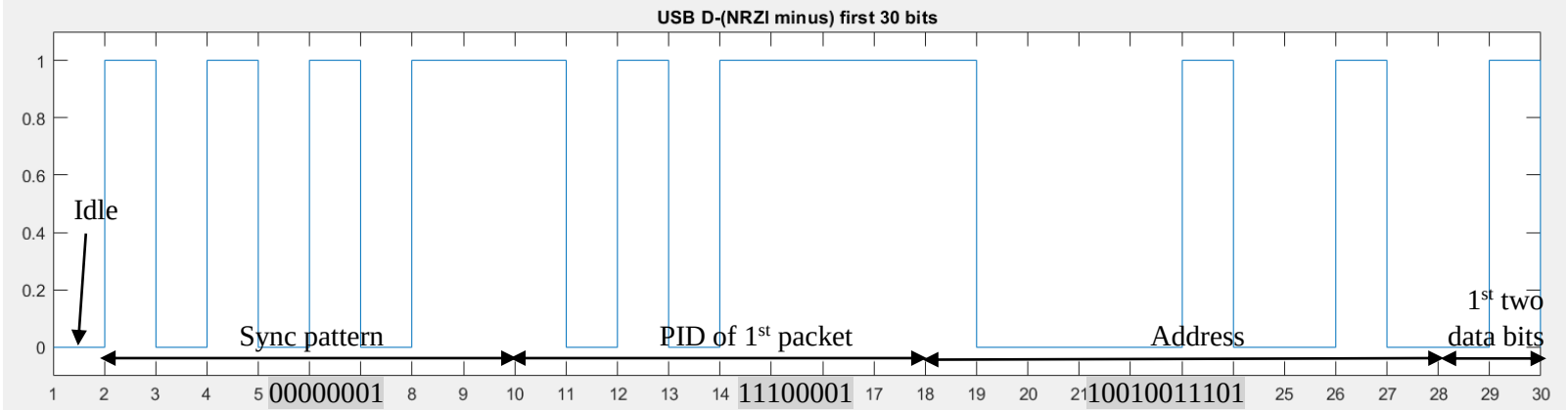
- The packet is framed with one start bit and 2 stop bit at the end and an even parity bit is inserted before the stop bit. Hence, the the overhead will increase and the transmission time will increases.
- Although, it have 2 stop bit and parity but it still faster than the previous one(7-bit with no parity).
- Comparing with the privious 8-bit with no parity and one stop, the previous has larger max and min efficiency and smaller max and min overhead.
- The parity at first packet is high because we have 3 ones, and the parity was the 4th.
- The parity at second packet is low because we have 4.

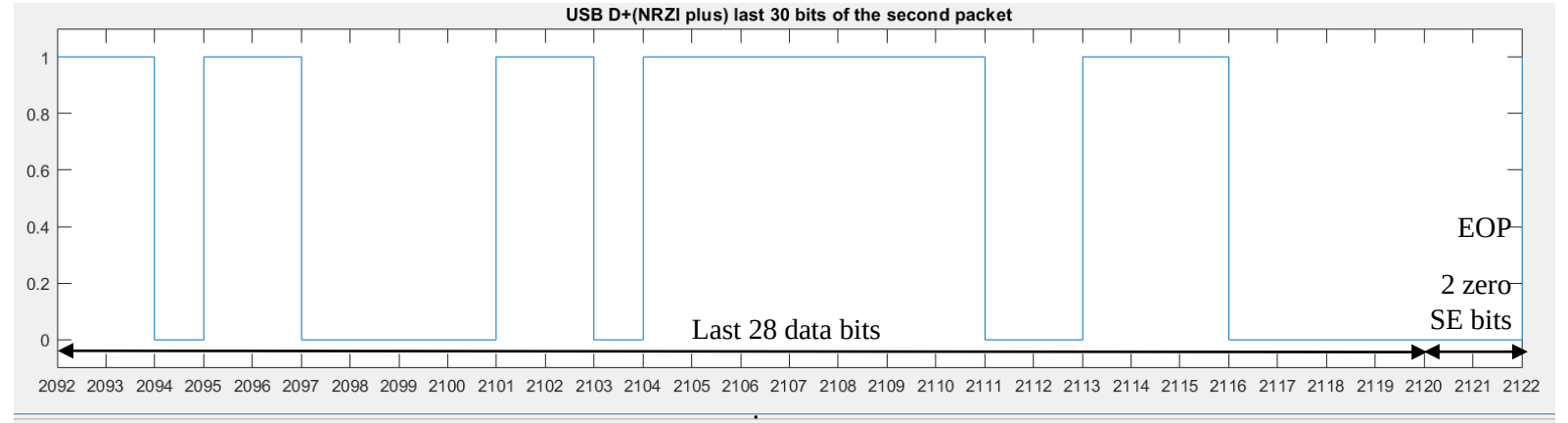
$$\text{Overhead for the whole file} = \frac{1120frames\ (2(stop) + 1(start) + 1(parity))}{1280 * 7 + 1120frames\ (2(stop) + 1(start) + 1(parity))} = 0.33333$$

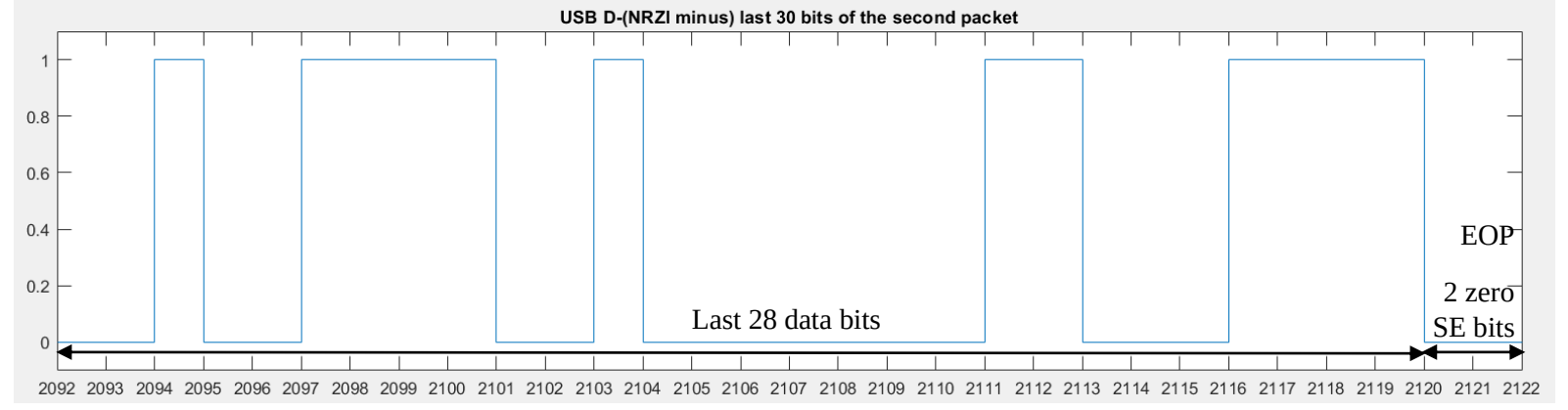
$$\text{efficiency for the whole file} = \frac{1280 * 7}{1280 * 7 + 1120frames\ (2(stop) + 1(start) + 1(parity))} = 0.66667$$

USB:


USB D+(NRZI plus) first 30 bits
Idle
Sync pattern
PID of 1st packet
Address
1st two data bits
00000001
11100001
10010011101


USB D-(NRZI minus) first 30 bits
Idle
Sync pattern
PID of 1st packet
Address
1st two data bits
00000001
11100001
10010011101


USB D+(NRZI plus) last 30 bits of the second packet
EOP
2 zero SE bits
Last 28 data bits


USB D-(NRZI minus) last 30 bits of the second packet
EOP
2 zero SE bits
Last 28 data bits

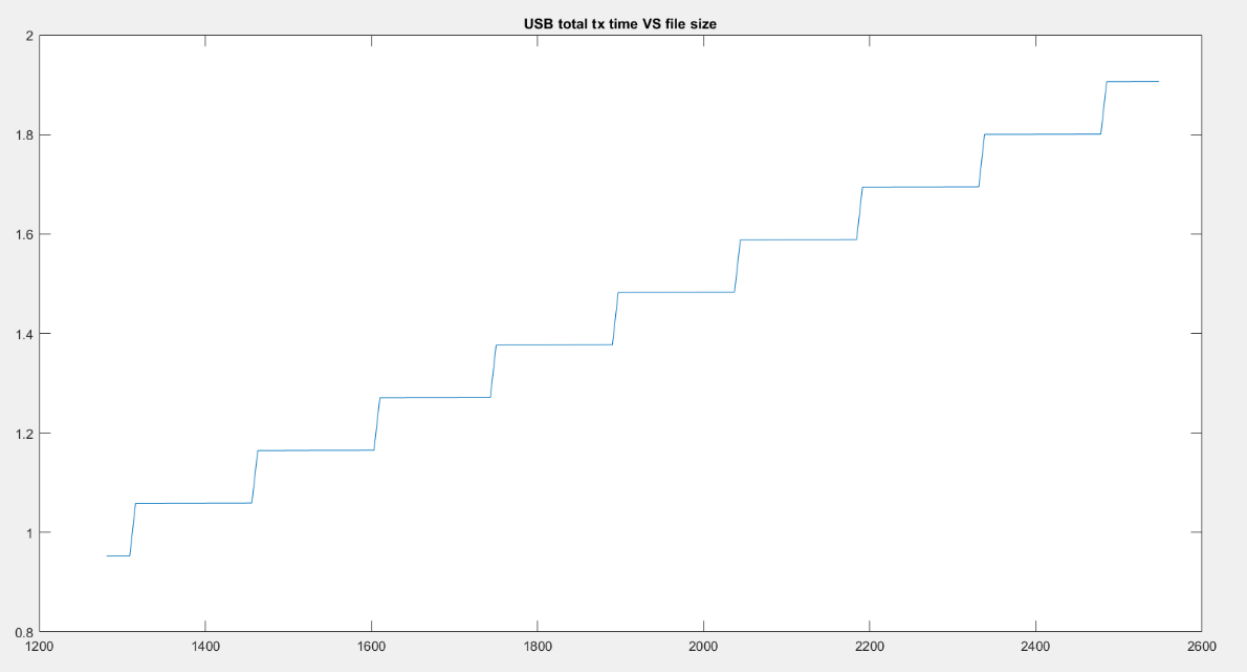
USB efficiency, overhead and total tx time:

In this protocol the last packet has a constant payload which contains the data bytes and zero padding for the rest of it (if there was any remaining bytes)

Total transmitting time:

The total transmitting time vs file size using USB protocol is as the following:

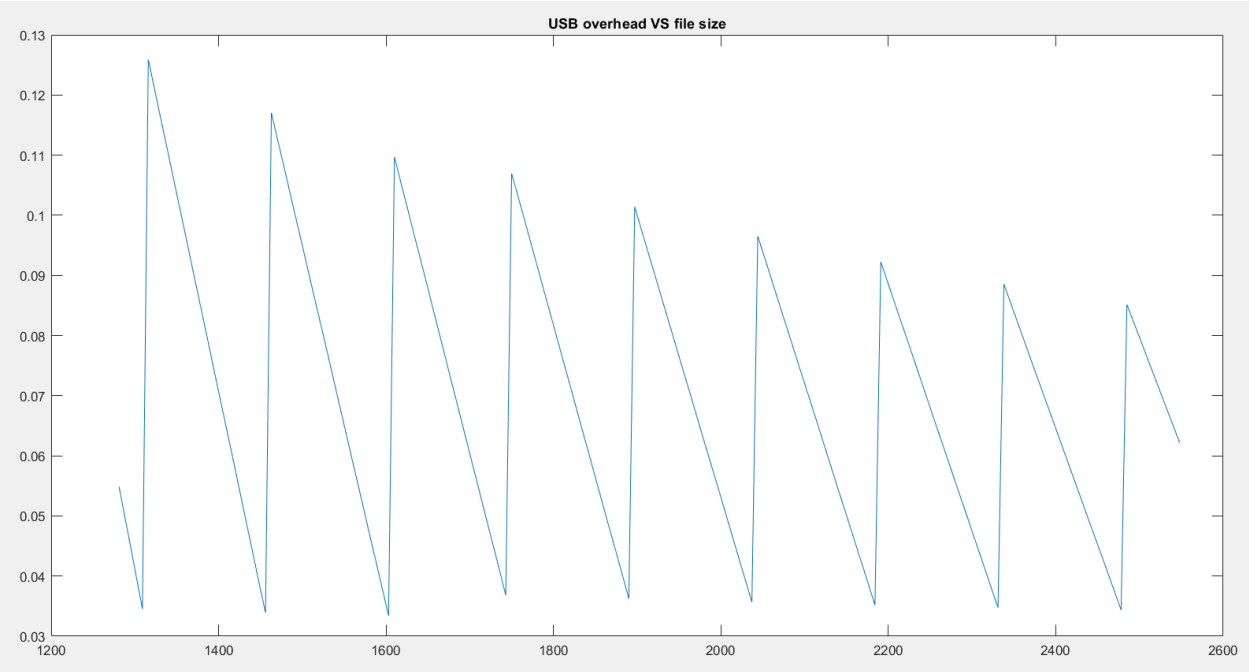
The total sending time depends on the number of packets.

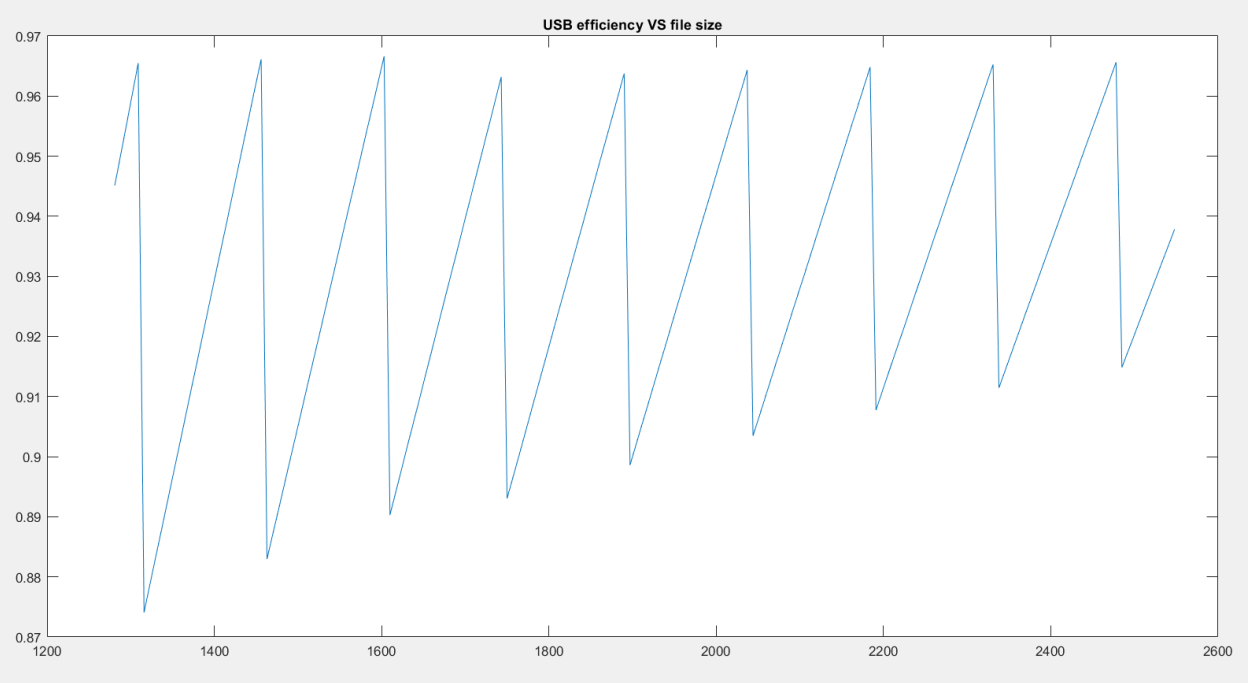


The graph has some points at which it has constant value of total sending time despite the increase of the file size, which can be explained that the data bytes are filling the packet but the number of packets are still the same so the total sending time is the same.

The steps in the graph shows that a new packet is added and the total sending time increases by (bit duration * payload *8)

USB overhead and efficiency:





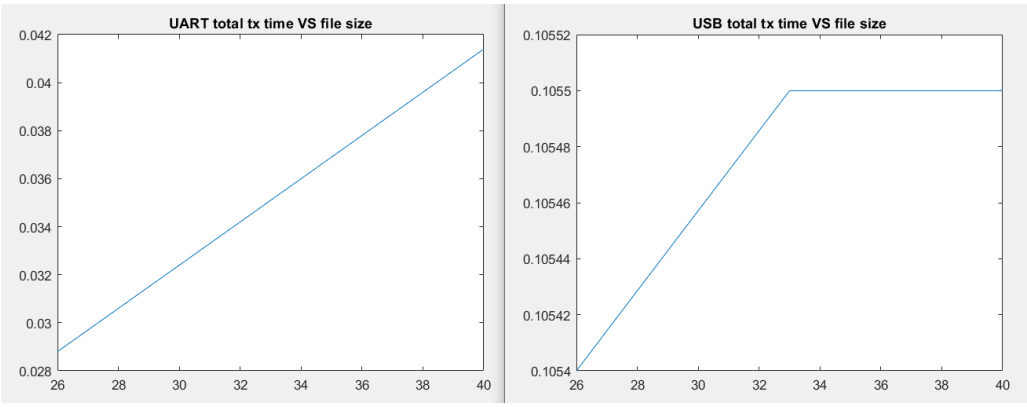
The efficiency and overhead graphs are as expected are rising and falling while increasing file size. The explanation for that behavior is that data bytes are filling the last packet so the overhead of zero padding decreases and hence the efficiency increases. Then at the end of this packet the efficiency and overhead reach their maximum and minimum. After that a new packet is created and the overhead of zero padding increases suddenly so the overhead reaches maximum and efficiency goes to minimum and so on…

The overall overhead decreases as the data sent increases and gets larger than overhead bits then the overall efficiency increases.

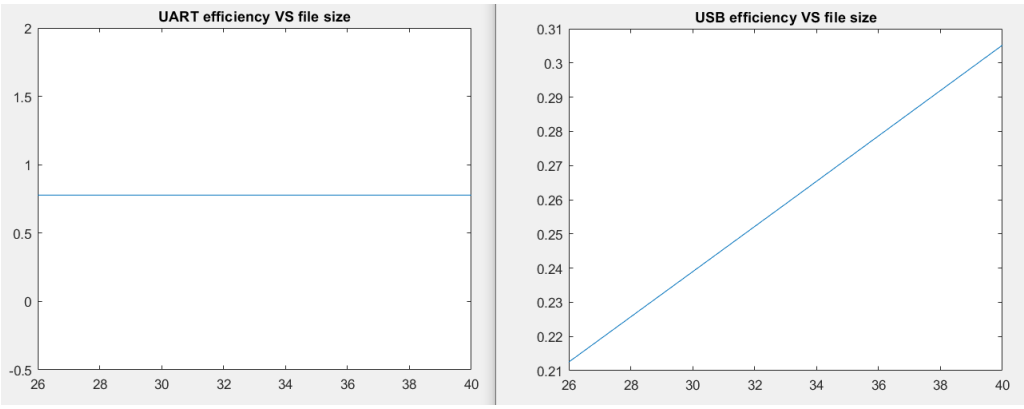
UART vs USB

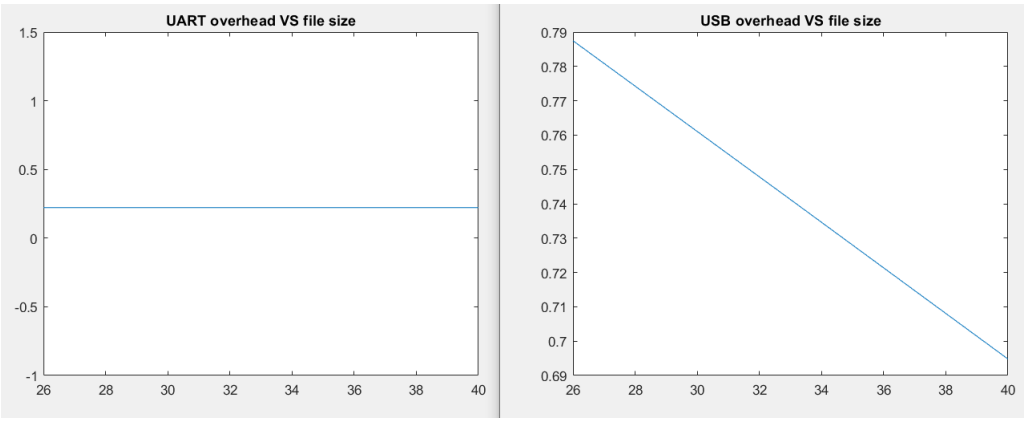
-As we noticed from the previous results the USB protocol has less transmitting time compared to the UART protocol so the USB is faster.

The UART is be faster (smaller tx time) than USB just in case of sending small size data as the following case:



-The overhead and efficiency of the USB are better than the UART in our case of 1280-byte file as the number overhead bits in UART is comparable with number data bits in a single frame, unlike the USB which has almost negligible number overhead bits compared with number of payload bits. But in case of small size file the UART has better efficiency and overhead as shown below:





Hence we can conclude that in case of large size data, the USB is more efficient and faster. And in case of small size data, the UART is more efficient and faster than the UART.