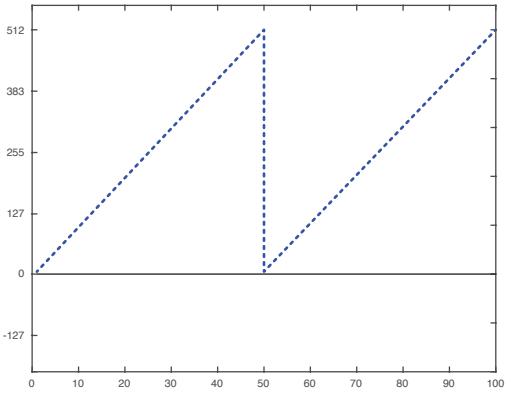
Modulo och bitwiseAND



Originalsignal i blått, den går mellan 0 och 512.

https://www.computerhope.com/jargon/m/modulo.htm
https://www.geeksforgeeks.org/bitwise-operators-in-c-cpp/

https://se.mathworks.com/help/matlab/ref/rem.html?searchHighlight=rem
https://www.arduino.cc/en/pmwiki.php?n=Reference/Modulo
https://en.wikipedia.org/wiki/Bitwise_operation#AND
https://se.mathworks.com/help/matlab/ref/mod.html?searchHighlight=mod
https://www.arduino.cc/reference/en/language/structure/bitwise-operators/bitwiseand/

Med `rem` i Matlab fås det som kvarstår efter division. Det kallas modulo (`%`) i Arduino (och flera andra programspråk). Här visat med en signal i rött.

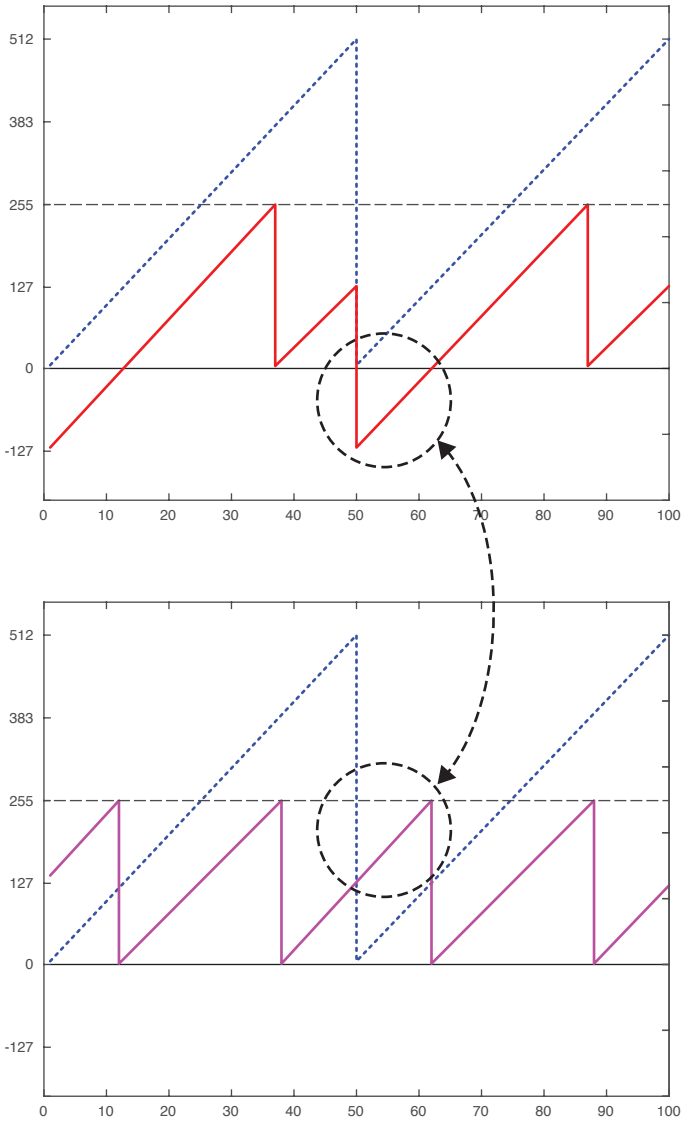
```
value = 255;
```
I Matlab:
```
r = rem((signal - 127), value);
```
I Arduino:
```
signal = signal - 127;
r = signal % (value + 1);
```

Value behövs ökas med 1 för att värdet 255, ska användas. Här börjar signalen om från 0 när den överstiger value. Negativa tal förblir negativa.

Med `mod` i Matlab fås också det som kvarstår efter division men bara i positiva värden. För att få den funktionen används i Arduino bitwiseAND (`&`). Här visat med en signal i lila.

```
value = 255;
```
I Matlab:
```
m = mod((signal - 127), value);
```
I Arduino:
```
signal = signal - 127;
m = signal & value;
```

Även här börjar signalen om från 0, men bara positiva värden används eftersom 8 bitar binärkod anger värden mellan 0 och 255.



Om value är 255 så anges det som `11111111` i 8 bitar binärkod. BitwiseAND i Arduino fungerar genom att göra en logisk operation (`AND`) på en `int` som ett 16-bitars binärtal, om värdet överstiger value annars förblir värdet detsamma.

```
   100  00000000 01100100  |    255  00000000 11111111  |    300  00000001 00101100
 & 255  00000000 11111111  |  & 255  00000000 11111111  |  & 255  00000000 11111111
 = 100  00000000 01100100  |  = 255  00000000 11111111  |  =  44  00000000 00101100
-------------------------------------------------------------------------------------
   450  00000001 11000010  |    512  00000010 00000000  |   -100  11111111 10011100
 & 255  00000000 11111111  |  & 255  00000000 11111111  |  & 255  00000000 11111111
 = 194  00000000 11000010  |  =   0  00000000 00000000  |  = 156  00000000 10011100
```