Strobe Generator - Randomised Pulses

This script generates a strobe sequence in which either or both of the on and off durations can be randomised (Makes use of Statistics and Machine Learning Toolbox). There are four different output modes of this script:

1) If both on and off durations are constant then the duty cycle can be set to a specified quantity.

2) If only one of the on or off durations are randomised and the other is constant then the number of flashes across the whole stimulus will remain somewhat constant.

3) If both of the on and off durations are completely randomised then the frequency of the strobe will vary greatly but should average across time to near the specified value

4) If the off duration is changed equal to the inverse of the on duration (or vice versa) then the frequency of the strobe will remain constant and the duty cycle will be randomised.

Common Parameters:

```
durationSeconds = 2;
baseStrobeHz = 10;
chosenMode = 2;

centralBrightness = 0;
ringBrightness = 255;

% Calculated parameter(s)
wavePeriod = 1/baseStrobeHz;
waveCount = durationSeconds * baseStrobeHz; % How many waves we can fit in
the duration
```

The following setup code calculates the on/off durations for each of the periods of the strobe waveform depending on the mode of the script.

Mode 1 Parameters:

Constant duty cycle.

```
if chosenMode == 1
    dutyCycle = 0.3; % Constant duty cycle

    onDuration = dutyCycle * wavePeriod;
    offDuration = (1-dutyCycle) * wavePeriod;

    % repeat duration values for every wavenumber
    onDurations = repmat(onDuration, 1, waveCount);
    offDurations = repmat(offDuration, 1, waveCount);
end
```

Mode 2 Parameters:

Fixed on duration, varying off time.

```
if chosenMode == 2
    averageDutyCycle = 0.5;
    minOffPercent = 0.2;
    maxOffPercent = averageDutyCycle + (averageDutyCycle - minOffPercent); %
50% +- 30%
    minOffTime = minOffPercent * wavePeriod;
    maxOffTime = maxOffPercent * wavePeriod;

    onDuration = averageDutyCycle * (1/baseStrobeHz);
    onDurations = repmat(onDuration, 1, waveCount);

    durationDistrib = makedist('Uniform', minOffTime, maxOffTime);
    offDurations = random(durationDistrib, 1, waveCount);
end
```

Mode 3 Parameters:

Random on time, random off time.

```
if chosenMode == 3
    averageDutyCycle = 0.5;

    minOnPercent = 0.2;
    maxOnPercent = averageDutyCycle + (averageDutyCycle - minOnPercent);
    minOnTime = minOnPercent * wavePeriod;
    maxOnTime = maxOnPercent * wavePeriod;

    minOffPercent = 0.2;
    maxOffPercent = averageDutyCycle + (averageDutyCycle - minOffPercent);
    minOffTime = minOffPercent * wavePeriod;
    maxOffTime = maxOffPercent * wavePeriod;

    % As the frequency varies, the wave period varies and we don't know how
    % many waves are required. Generate wave periods until we fill the
entire duration
    sumWavePeriods = 0;
    onDurations = [];
    offDurations = [];
    while sumWavePeriods < durationSeconds
        onDuration = random('Uniform', minOnTime, maxOnTime);
        offDuration = random('Uniform', minOffTime, maxOffTime);
        onDurations = [onDurations, onDuration];
        offDurations = [offDurations, offDuration];
        sumWavePeriods = sum(onDurations) + sum(offDurations);
    end
end
```

Mode 4 Parameters:

Random duty cycle per wave.

```
if chosenMode == 4
    averageDutyCycle = 0.5;

    minOnOffPercent = 0.2;
    maxOnOffPercent = averageDutyCycle + (averageDutyCycle -
minOnOffPercent);
    minOnOffTime = minOnOffPercent * wavePeriod;
    maxOnOffTime = maxOnOffPercent * wavePeriod;

    durationDistrib = makedist('Uniform', minOffTime, maxOffTime);
    onDurations = random(durationDistrib, 1, waveCount);
    offDurations = wavePeriod - onDurations; % The wave is off for the
period minus the on time, so the duration of each wave is contant
end
```

Sample Generation

Using the calculated durations we will now calculate the on/off state for each of the samples.

```
frameDurationS = (1/2000); % Time duration of each frame
sampleTimes = (0:frameDurationS:durationSeconds - frameDurationS)'; %
Generate a list sample timestamps
```

Use the on/off durations to calculate the start time, trainsition point, and end time of each wave;

```
wavePeriods = onDurations + offDurations;
waveEndTimes = cumsum(wavePeriods);
waveStartTimes = waveEndTimes - wavePeriods;
waveMidPoints = waveStartTimes + onDurations;
```

It is worth noting that due to the uniformly randomised nature of the wave periods, the total duration of all waveforms may differ from the generated stimuli as shown here:

```
disp('Waveform duration: ' + string(waveEndTimes(end)));
```

```
Waveform duration: 2.0074
```

```
disp('Playback duration: ' +  string(sampleTimes(end) + frameDurationS));
```

```
Playback duration: 2
```

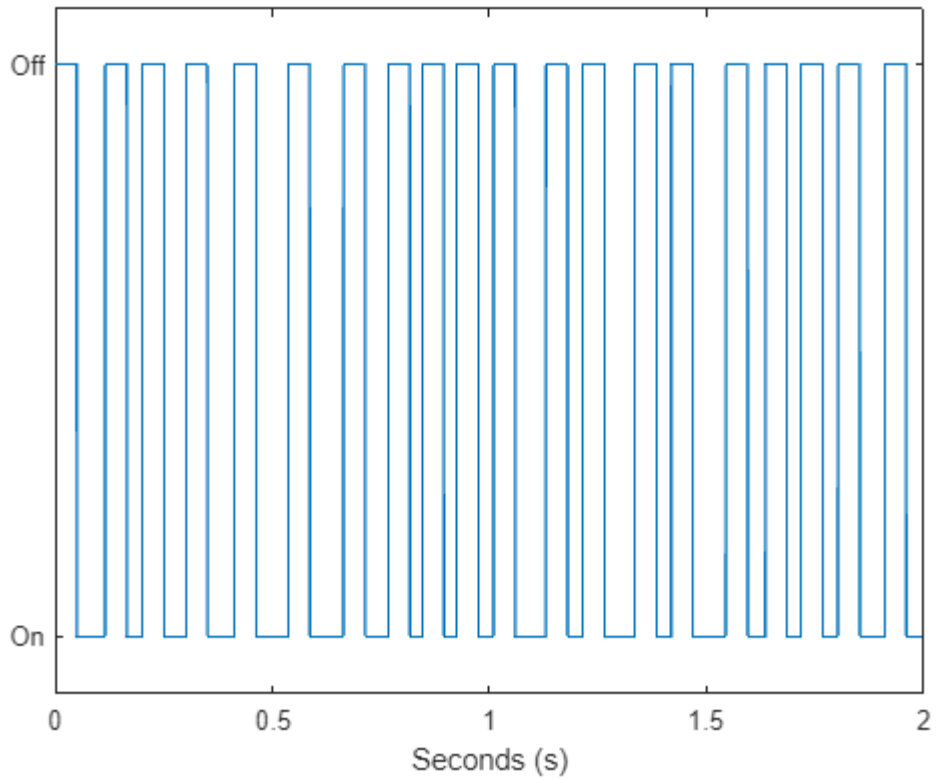
Using some MATLAB matrix math, calculate which samples are (for at least one wave) both after the start time and earlier than the on-to-off transition time:

```
strobePerSample = sum((sampleTimes >= waveStartTimes & sampleTimes <
waveMidPoints), 2);
```

The strobe signal can be seen in the plot below:

```
plot(sampleTimes, strobePerSample)
ylim([-0.1, 1.1])
```

```
yticks([0,1]);
yticklabels(["On", "Off"])
xlabel("Seconds (s)")
```



We have generated the on/off state for each sample and since we are controlling all LEDs with the same signal we need to repeat that value for all 8 LEDs:

```
ledONOFFsamples = repmat(strobePerSample, 1, 8);
```

Next we need to convert this array of 8 bits (1's or 0's) for each sample into a single 8-bit unsigned integer using this small helper function:

```
ledONOFFBitmap = binary8ToUint8(ledONOFFsamples); % Use the strobe signal to
turn on and off the ring LED states
```

(this function is defined at the end of this file)

We need to append to the right of each of these values the brightness values for each channel, these values are all the same so we generate a single row with the following command:

```
dacChannelValues = [centralBrightness, ringBrightness, ringBrightness,
ringBrightness, ringBrightness];
```

And then repeat that row for every single sample:

```
dacChannelValuesPerSample = repmat(dacChannelValues, [length(sampleTimes),
1]);
```

and append them to the ledONOFFBitmap values:

```
preparedStrobeData2D = [ledONOFFBitmap, dacChannelValuesPerSample];
```

This should be a 2D matrix where each row contains a single data packet and there is a row for each displayed sample.

In order to transmit this data we need to append the rows into a single 1D sequence using the following command:

```
preparedStrobeData1D = reshape(preparedStrobeData2D',
[size(preparedStrobeData2D, 1) * size(preparedStrobeData2D, 2), 1])';
```

This data can then be used with the DeviceUsageFromFileExample.m

```
function value = binary8ToUint8(bitArray)
    value = sum([2^7 2^6, 2^5, 2^4, 2^3, 2^2, 2^1, 2^0] .* bitArray, 2);
    return;
end
```