Table 1: Relative Position of LED and Photdector Components in the MAX 30101 Chip

| Component | Red LED | IR LED | Green LED | Photodetector |
|---|---|---|---|---|
| Relative Position in the MAX 30101 Chip in x and y coordinates (mm) | (4.4,10.6) | (4.4,10.6) | (4.4,10.6) | (4.4,2.8) |

*all the LEDs are the same source

Effective area of the detector = 1.4 $mm^2$

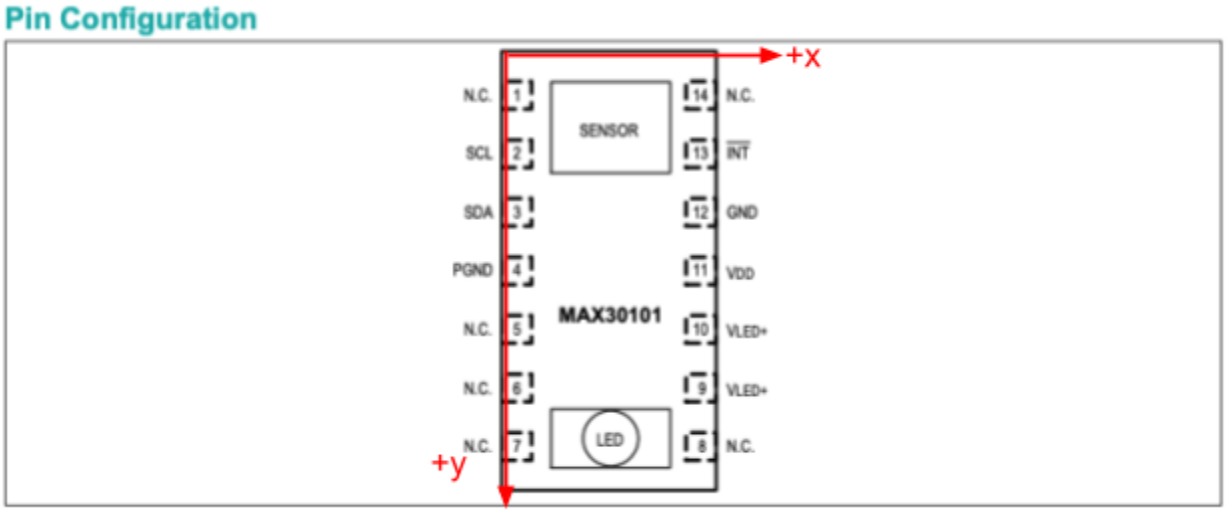



Table 2: Wavelength and Output Power for Each LED in the MAX 30101 Chip

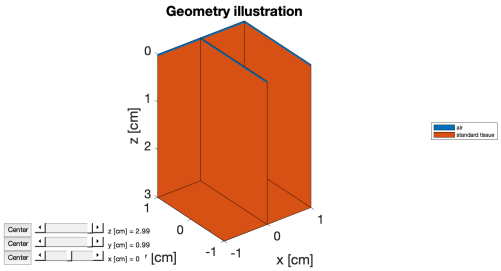
| LED | Red LED | IR LED | Green LED |
|---|---|---|---|
| Peak Wavelength (nm) | 660 (650-670) | 880 (870-900) | 537 (530-545) |
| Output Power (mW) (Radiant Power) | 9.8 | 6.5 | 17.2 |
| LED Supply Voltage (V) | 3.3 (3.1-5.0) | 5.0 (4.5-5.5) | 3.3 (3.1-5.0) |

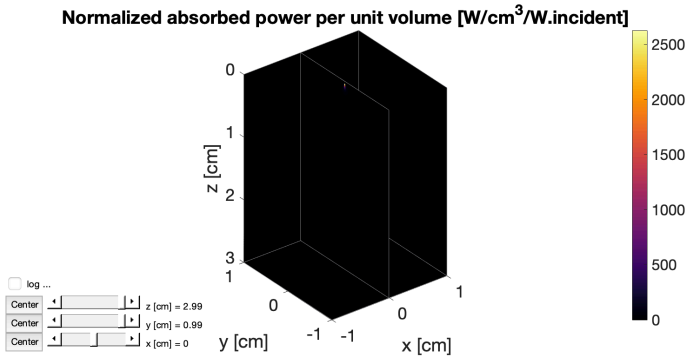
Simulation Parameters:

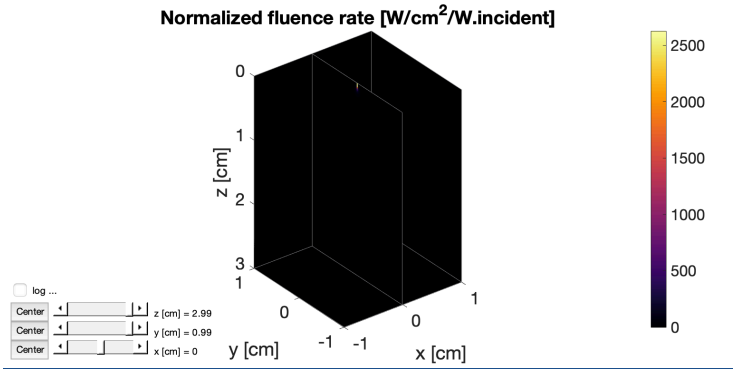
- Standard tissue: Example1_StandardTissue.m Simulation File
- Dimensions of simulation box: 2,2,3 cm
- Run time: 0.1 minute

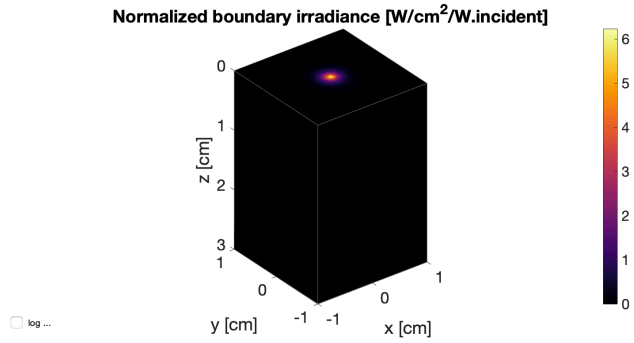
Red:

59.8% of incident light was absorbed within the cuboid.
40.2% of incident light hits the cuboid boundaries.
Mean fluence at the detector: 0.0012 W/cm^2/W.incident
Power detected by the detector: 1.6607e-09 W



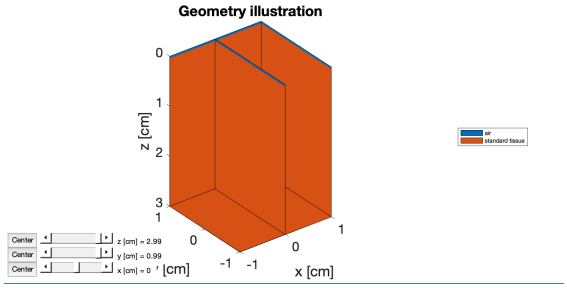



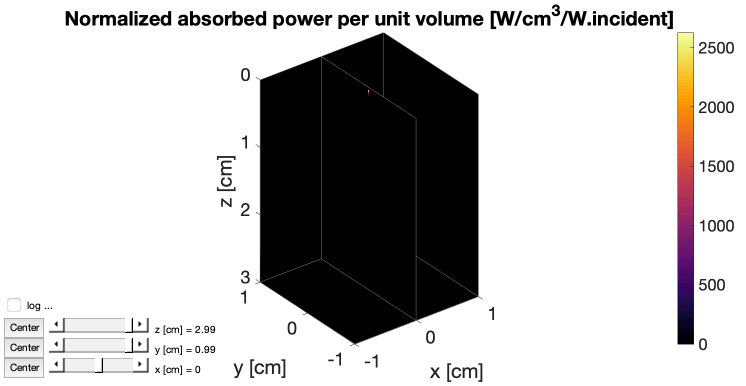



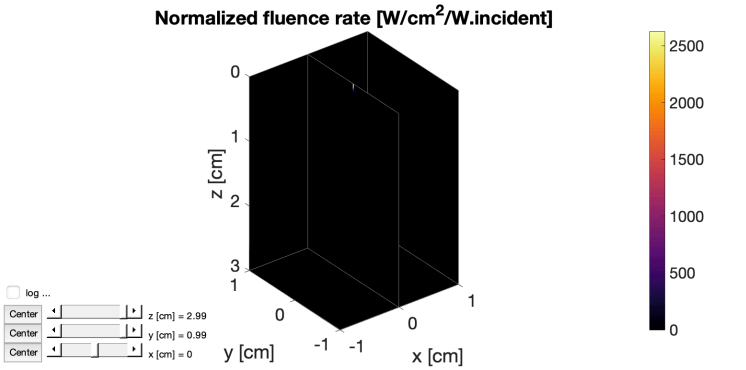



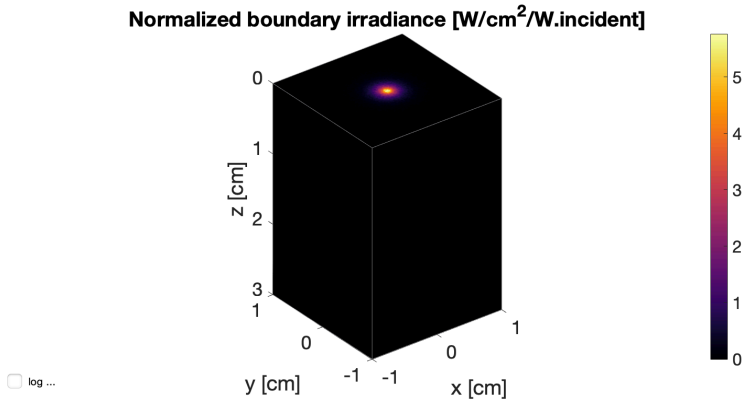
IR:

59.5% of incident light was absorbed within the cuboid.
40.5% of incident light hits the cuboid boundaries.
Mean fluence at the detector: 0.0004 W/cm^2/W.incident
Power detected by the detector: 3.3503e-10 W



Green:

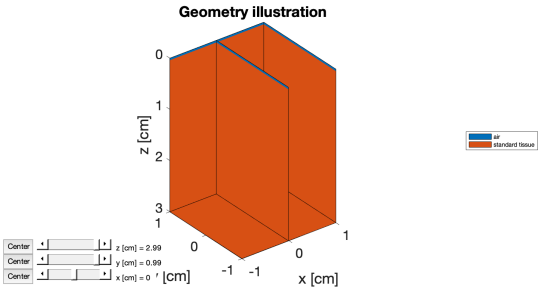
Power Received by Detector:

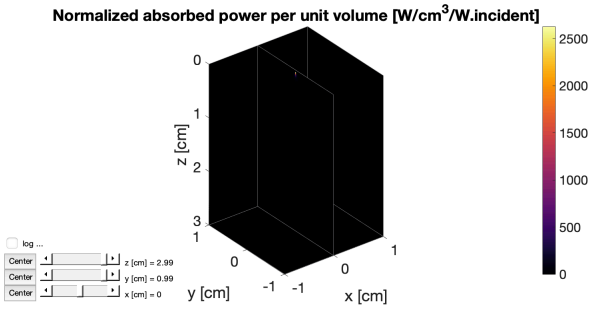
59.7% of incident light was absorbed within the cuboid.

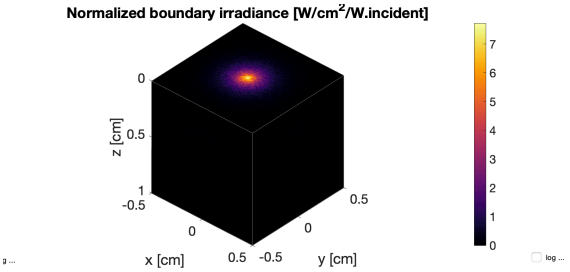
40.3% of incident light hits the cuboid boundaries.

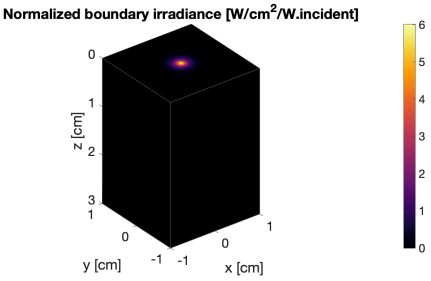
Mean fluence at the detector: 0.0006 W/cm^2/W.incident

Power detected by the detector: 1.54e-09 W

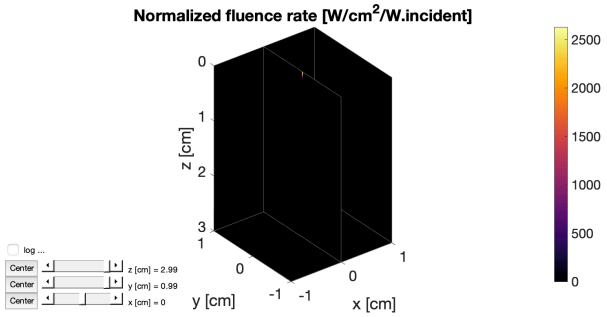



*same graphs just different scales




According to the results, the differences in wavelength and output power between the green, red, and IR LEDs revealed differences in the power detected by the detector. However, all cases

maintained similar values in the incident light absorbed by the cuboid and the incident light hitting the cuboid boundaries. These similar values suggest that the absorption and scattering properties of the cuboid were relatively uniform for the different wavelengths.

The order of detected power from most to least by the pulse oximeter detector is: red (1.6607e-09 W), green (1.54e-09 W), and IR (3.3503e-10 W). The red light was more effectively detected by the pulse oximeter. Factors like the detector's sensitivity to specific wavelengths or the absorption properties of the tissue at different wavelengths could explain these differences.

The order of wavelengths from most to least is IR (880 nm), red (660 nm), and green (537 nm). The wavelength affects the interaction with biological tissues and the detector's ability to measure light. The order of output power from most to least is green (17.2 mW), red (9.8 mW), and IR (6.5 mW). Despite the higher output power of the green LED compared to the red and IR LEDs, the detected power was lower for green than red. This discrepancy could be due to factors like scattering, absorption, or reflection properties of the tissue and the way light interacts with it.

In the end, especially with pulse oximeters, the detector is most sensitive to red light wavelengths, as seen with red having the highest detected power. Red and infrared light wavelengths are known to have distinct absorption properties depending on the oxygenation state of hemoglobin in tissue, which is critical for pulse oximetry measurements. IR light tends to penetrate deeper into tissues but is absorbed less by hemoglobin molecules (hemoglobin's absorption spectrum peaks in the visible range of 660 nm (close to red wavelength)). This leads to lower detected power from the IR LED compared to the red LED. As a result, the pulse oximeter is less sensitive to IR compared to red wavelengths in order to optimize unoxygenated blood oxygen level analysis. Oxygenated hemoglobin absorbs light differently, with a stronger absorption in the IR range (around 880 nm), explaining why the pulse oximeter has the IR LED. The light from the green LED interacts with hemoglobin in a way that can help improve the accuracy of pulse oximetry readings, especially in situations where blood flow is low, perfusion is poor, or when the system has low pulsatility. In the end, red light is most efficiently absorbed by deoxygenated hemoglobin molecules, making it the ideal wavelength for a pulse oximeter to be sensitive to compared to green and IR LEDs.

Detailed Requirements:

- Determine the relative position of each LED and the photodetector in the MAX30101 chip by using its datasheet (and possibly educated guesses)
- Determine the wavelength and output power of each LED (use the Datasheet)
- Build and run an MCMatlab simulation for each case of LED-detector pair. Assume standard tissue. Choose reasonable dimensions for the simulation box and a reasonable run time (or number of photos).
- Report the results in a Word document, indicating the power received by the detector for each LED illuminating. Include figures of the model geometry and reflectance fluence.
- Submit your project to a private Github repository.
- Share your repo with user: shai-BME

Source: https://www.analog.com/media/en/technical-documentation/data-sheets/max30101.pdf

Code:

```
%% Description
% In this introductory example, a block of "standard tissue" (mu_a = 1,
% mu_s = 100, g = 0.9) is illuminated by a pencil beam (infinitely thin
% beam). A small slice of air is present in the top of the simulation
% volume. nx and ny are set to odd values so that when the pencil beam is
% launched at x = y = 0 and travels straight down, it travels along a
% well-defined center column of voxels (the middle of the 51st column). Use
% the log10 plot checkbox in the visualizations to better see the fluence
% rate and absorption distribution in the MC result.
%% MCmatlab abbreviations
% G: Geometry, MC: Monte Carlo, FMC: Fluorescence Monte Carlo, HS: Heat
% simulation, M: Media array, FR: Fluence rate, FD: Fractional damage.
%
% There are also some optional abbreviations you can use when referencing
% object/variable names: LS = lightSource, LC = lightCollector, FPID =
% focalPlaneIntensityDistribution, AID = angularIntensityDistribution, NI =
% normalizedIrradiance, NFR = normalizedFluenceRate.
%
% For example, "model.MC.LS.FPID.radialDistr" is the same as
% "model.MC.lightSource.focalPlaneIntensityDistribution.radialDistr"
%% Geometry definition
MCmatlab.closeMCmatlabFigures();
model = MCmatlab.model;
model.G.nx                = 101; % Number of bins in the x direction
model.G.ny                = 101; % Number of bins in the y direction
model.G.nz                = 150; % Number of bins in the z direction
model.G.Lx                = 2; % [cm] x size of simulation cuboid (101, 101,
225) mm box
model.G.Ly                = 2; % [cm] y size of simulation cuboid
model.G.Lz                = 0.5; % [cm] z size of simulation cuboid
model.G.mediaPropertiesFunc = @mediaPropertiesFunc; % Media properties defined
as a function at the end of this file
model.G.geomFunc          = @geometryDefinition; % Function to use for defining
the distribution of media in the cuboid. Defined at the end of this m file.
model = plot(model,'G');
%% Monte Carlo simulation
model.MC.simulationTimeRequested  = 0.1; % 0.1 [min] Time duration of the
simulation
model.MC.matchedInterfaces        = true; % Assumes all refractive indices are
the same
model.MC.boundaryType             = 1; % 0: No escaping boundaries, 1: All
cuboid boundaries are escaping, 2: Top cuboid boundary only is escaping, 3: Top
and bottom boundaries are escaping, while the side boundaries are cyclic
```

```
model.MC.wavelength                = 537; % [nm] Excitation wavelength, used for
determination of optical properties for excitation light
% red = 660, IR = 880, Green = 537
Pout = 17.2/1000; % mW
% red = 9.8 mV, IR = 6.5 mV, Green = 17.2 mV
model.MC.lightSource.sourceType    = 0; % 0: Pencil beam, 1: Isotropically
emitting line or point source, 2: Infinite plane wave, 3: Laguerre-Gaussian
LG01 beam, 4: Radial-factorizable beam (e.g., a Gaussian beam), 5: X/Y
factorizable beam (e.g., a rectangular LED emitter)
% For a pencil beam, the "focus" is just a point that the beam goes
% through, here set to be the center of the cuboid:
model.MC.lightSource.xFocus        = 0; % [cm] x position of focus
model.MC.lightSource.yFocus        = 0; % [cm] y position of focus
model.MC.lightSource.zFocus        = 0.05; % [cm] z position of focus
% These lines will run the Monte Carlo simulation with the provided
% parameters and subsequently plot the results:
model = runMonteCarlo(model);
model = plot(model,'MC');
%% Define detector position (as given earlier)
detector_center = [0, 7.8/10, model.MC.lightSource.zFocus]; % cm
detector_area = 1.4 / 10000; % cm² (convert from mm² to cm²)
aspect_ratio = 1;
% Calculate width and height of the detector
detector_width = sqrt(detector_area * aspect_ratio); % cm
detector_height = detector_width / aspect_ratio; % cm
% Grid resolution (in cm)
grid_dx = model.G.Lx / model.G.nx;
grid_dy = model.G.Ly / model.G.ny;
% Convert detector center to grid indices
detector_z_index = round(detector_center(3) / model.G.Lz * model.G.nz);
detector_x_index = round(detector_center(1) / model.G.Lx * model.G.nx);
detector_y_index = round(detector_center(2) / model.G.Ly * model.G.ny);
% Clamp indices to valid ranges
detector_x_index = max(1, min(detector_x_index, model.G.nx));
detector_y_index = max(1, min(detector_y_index, model.G.ny));
detector_z_index = max(1, min(detector_z_index, model.G.nz));
% Calculate half-width and half-height in grid units (for region of interest)
half_width_idx = ceil((detector_width / 2) / grid_dx);
half_height_idx = ceil((detector_height / 2) / grid_dy);
% Initialize fluence sum
fluence_sum = 0;
num_points = 0;
% Loop through grid points within the rectangular bounds around the detector
for i = -half_width_idx:half_width_idx
   for j = -half_height_idx:half_height_idx
       % Calculate grid indices
       x_idx = detector_x_index + i;
       y_idx = detector_y_index + j;
```

```
        % Ensure indices are within bounds
        if x_idx > 0 && x_idx <= model.G.nx && y_idx > 0 && y_idx <= model.G.ny
            fluence_value = model.MC.normalizedFluenceRate(x_idx, y_idx,
detector_z_index);
            fluence_sum = fluence_sum + fluence_value;
            num_points = num_points + 1;
        end
    end
end
% Calculate the mean fluence at the detector position
mean_fluence = fluence_sum / num_points;
% Display the mean fluence
fprintf('Mean fluence at the detector: %.4f W/cm^2/W.incident\n',
mean_fluence);
% Ensure Pout is defined
if ~exist('Pout', 'var') || Pout == 0
    error('Pout is not defined or is zero.');
end
% Calculate power detected by the detector
P_detected = Pout * mean_fluence * detector_area;
% Display power detected by the detector
disp(['Power detected by the detector: ', num2str(P_detected), ' W']);
%% Geometry function(s) (see readme for details)
% A geometry function takes as input X,Y,Z matrices as returned by the
% "ndgrid" MATLAB function as well as any parameters the user may have
% provided in the definition of Ginput. It returns the media matrix M,
% containing numerical values indicating the media type (as defined in
% mediaPropertiesFunc) at each voxel location.
function M = geometryDefinition(X,Y,Z,parameters)
 zSurface = 0.03;
 M = ones(size(X)); % Air
 M(Z > zSurface) = 2; % "Standard" tissue
end
%% Media Properties function (see readme for details)
% The media properties function defines all the optical and thermal
% properties of the media involved by filling out and returning a
% "mediaProperties" array of "mediumProperties" objects with various
% properties. The j indices are the numbers that are referred to in the
% geometry function (in this case, 1 for "air" and 2 for "standard tissue")
% See the readme file or the examples for a list of properties you may
% specify. Most properties may be specified as a numeric constant or as
% function handles.
%
% The function must take one input; the cell array containing any special
% parameters you might specify above in the model file, for example
% parameters that you might loop over in a for loop. In most simulations
% this "parameters" cell array is empty. Dependence on wavelength is shown
% in examples 4 and 23. Dependence on excitation fluence rate FR,
% temperature T or fractional heat damage FD can be specified as in
```

```
% examples 12-15.
function mediaProperties = mediaPropertiesFunc(parameters)
 % Always leave the following line in place to initialize the
 % mediaProperties array:
 mediaProperties = MCmatlab.mediumProperties;
 % Put in your own media property definitions below:
 j=1;
 mediaProperties(j).name  = 'air';
 mediaProperties(j).mua   = 1e-8; % Absorption coefficient [cm^-1]
 mediaProperties(j).mus   = 1e-8; % Scattering coefficient [cm^-1]
 mediaProperties(j).g     = 1; % Henyey-Greenstein scattering anisotropy
 j=2;
 mediaProperties(j).name  = 'standard tissue';
 mediaProperties(j).mua   = 1; % Absorption coefficient [cm^-1]
 mediaProperties(j).mus   = 100; % Scattering coefficient [cm^-1]
 mediaProperties(j).g     = 0.9; % Henyey-Greenstein scattering anisotropy
end
```