```php
<?php
//To activate productionMode (display entering deep sleep), set http-header X-productionMode: true
#header("X-productionMode: true");
//To stop productionMode (no deep sleep, web config), set http-header X-productionMode: false
#header("X-productionMode: false");

// Set the sleep interval for the doorsigns via the server
#header("X-sleepInterval: 60 ");

error_reporting('E_ERROR');
# Supported displays:
# 1.54 inches: https://www.waveshare.com/wiki/1.54inch_e-Paper_Module
# 2.9 inches: https://www.waveshare.com/wiki/2.9inch_e-Paper_Module
# 4.2 inches: https://www.waveshare.com/wiki/4.2inch_e-Paper_Module
# 7.5 inches: https://www.waveshare.com/wiki/7.5inch_e-Paper_HAT
const DISPLAYS = array( "7.5"=>array("size"=>"640x384","rotate"=>"false"),
                        "7.5bwr"=>array("size"=>"640x384","rotate"=>"false", "red"=>"true"),
                        "4.2"=>array("size"=>"400x300","rotate"=>"false"),
                        "4.2bwr"=>array("size"=>"400x300","rotate"=>"false", "red"=>"true"),
                        "2.9"=>array("size"=>"296x128","rotate"=>"true"),
                        "1.5"=>array("size"=>"200x200","rotate"=>"true")
                        );

// Use Googles Noto fonts as the default font face
$DEFAULT_FONT = array(
    "test"=>realpath("./fonts/Wingdings_3.ttf"),
    "regular"=>realpath("./fonts/noto/NotoSans-Regular.ttf"),
    "bold"=>realpath("./fonts/noto/NotoSans-Bold.ttf"),
    "italic"=>realpath("./fonts/noto/NotoSans-Italic.ttf"),
    "bolditalic"=>realpath("./fonts/noto/NotoSans-BoldItalic.ttf"),
    "symbols"=>realpath("./fonts/noto/NotoSansSymbols-Regular.ttf"),
    "emoji"=>realpath("./fonts/noto/NotoEmoji-Regular.ttf"),
    "weathericons"=>realpath("./fonts/weathericons-regular-webfont.ttf")
    );


// To use LiberationSans font, uncomment the following lines
/*
$DEFAULT_FONT = array(
    "regular"=>realpath("./fonts/LiberationSans-Regular.ttf"),
    "bold"=>realpath("./fonts/LiberationSans-Bold.ttf"),
    "italic"=>realpath("./fonts/LiberationSans-Italic.ttf"),
    "weathericons"=>realpath("./fonts/weathericons-regular-webfont.ttf")
    );
*/

const THRESHOLDS = array("black" => 150, "red" => 240);

if (!extension_loaded('gd')) {
    echo "GD library is not installed. Please install GD on your server (http://php.net/manual/de/image.installation.php)";
    exit;
}

//Function to check if FreeType is installed. Not needed by static_image
function checkFreeType(){
    $gdInfo = gd_info();
    if($gdInfo['FreeType Support'] != 1){
        echo "FreeType is not enabled. FreeType is needed for creating text in images(http://php.net/manual/de/function.imagettftext.php)";
        exit;
    }
}

if(strlen($_GET['scale']) AND is_numeric($_GET['scale'])){
    $scale = $_GET['scale'];
}else{
    $scale = $_GET['scale'] = 32;
}

$displayType = $_GET['display'];
if(!isset(DISPLAYS[$displayType])){
    echo ("Not a valid display size. <br />");
    echo ("display=[");
    foreach (array_keys(DISPLAYS) as $display_key){
        echo ($display_key.", ");
    }
    echo ("]");
    exit;
}
$hasRed = DISPLAYS[$displayType]['red'];

$professor = htmlspecialchars($_GET["professor"]);

//Read existing contents
$contents = scandir('contents');

if(!count($contents)){
     echo "No content definitions";
     exit;
}

foreach ($contents as $content) {
    $contentFile = pathinfo("contents/".$content);

    if($contentFile['extension'] == "php"){
    $allContents[$contentFile['filename']] = "contents/".$content;
    }
}

$selectedContent = $allContents[$_GET['content']];

$displayWidth = explode("x",DISPLAYS[$displayType]['size'])[0];
$displayHeight = explode("x",DISPLAYS[$displayType]['size'])[1];
$im = imagecreate($displayWidth, $displayHeight);
$background_color = ImageColorAllocate ($im, 255, 255, 255);
$black = ImageColorAllocate($im, 0, 0, 0);
$red = ImageColorAllocate($im, 0xFF, 0x00, 0x00);


if(is_file($selectedContent)){
    include($selectedContent);
}else{
    echo "Not a valid content.";
    imagedestroy($im);
    exit;
}


if($_GET['debug'] == 'true'){
    header("Content-type: image/png");
    imagepng($im);
}
else{
    if(DISPLAYS[$displayType]['rotate'] == "true"){
        $im = imagerotate($im, 90, 0);
    }

    $im = imagerotate($im, 0, 0);
    //if you are using an older version of GD library you have to rotate the image 360°. Otherwise you get a white image due to a bug in GD library. Uncomment next lines:
    //$im = imagerotate($im, 180, 0);
    //$im = imagerotate($im, 180, 0);

    echo rawImage($im, $hasRed);
}

imagedestroy($im);

```

```php

function rawImage($im, $hasRed) {
    $bits = "";
    $bytes = "";
    $pixelcount = 0;

    for ($y = 0; $y < imagesy($im); $y++) {
        for ($x = 0; $x < imagesx($im); $x++) {

            $rgb = imagecolorat($im, $x, $y);
            $r = ($rgb >> 16) & 0xFF;
            $g = ($rgb >> 8 ) & 0xFF;
            $b = $rgb & 0xFF;
            $gray = ($r + $g + $b) / 3;

            if($hasRed == "true"){

                if(($r >= THRESHOLDS['red']) && ($g < 50) && ($b <50)) {
                    $bits .= "01";
                } else {
                    if ($gray < THRESHOLDS['black']) {
                        $bits .= "11";
                    }else {
                        $bits .= "00";
                    }
                }
            $pixelcount = $pixelcount+2;
            }else{
                  if ($gray < THRESHOLDS['black']) {
                $bits .= "1";
            }else {
                $bits .= "0";
            }
                $pixelcount++;
            }


            if ($pixelcount % 8 == 0) {
                $bytes .= pack('H*', str_pad(base_convert($bits, 2, 16),2, "0",
                STR_PAD_LEFT));
                $bits = "";
            }
        }
    }

    $size = strlen($bytes);

    header("Content-length: $size");
    return $bytes;
}
?>

```