```
`include "uart_trx.v"

module top (
  output wire led_red,
  output wire led_blue,
  output wire led_green,
  output wire uarttx,
  input wire uartrx,
  input wire hw_clk
);

  wire        int_osc;
  reg  [27:0] frequency_counter_i;

  // 9600 Hz clock generation
  reg clk_9600 = 0;
  reg [31:0] cntr_9600 = 32'b0;
  parameter period_9600 = 625;

  // Message "hello"
  reg [7:0] message [0:4];
  initial begin
    message[0] = "h";
    message[1] = "e";
    message[2] = "l";
    message[3] = "l";
    message[4] = "o";
  end

  reg [2:0] char_index = 0;
  reg send = 0;
  wire busy;
  wire [7:0] current_char = message[char_index];

  // Trigger sending at various frequency_counter_i bits
  always @(posedge int_osc) begin
    frequency_counter_i <= frequency_counter_i + 1'b1;

    // Clock generation for UART
    cntr_9600 <= cntr_9600 + 1;
  if (cntr_9600 == period_9600) begin
    clk_9600 <= ~clk_9600;
    cntr_9600 <= 0;
  end

  // Send one character every few million cycles (adjust bit index)
  case (char_index)
    0: send <= frequency_counter_i[23];
    1: send <= frequency_counter_i[24];
    2: send <= frequency_counter_i[25];
    3: send <= frequency_counter_i[26];
    4: send <= frequency_counter_i[27];
  endcase

  // Advance to next character after sending one
  if (send && !busy) begin
    char_index <= char_index + 1;
  end
end

uart_tx_8n1 DanUART (
  .clk(clk_9600),
  .txbyte(current_char),
  .senddata(send),
  .tx(uarttx),
  .busy(busy)
);

// RGB driver (reuse UART RX pin as dummy PWM)
SB_RGBA_DRV RGB_DRIVER (
  .RGBLEDEN(1'b1),
  .RGB0PWM(uartrx),
  .RGB1PWM(uartrx),
  .RGB2PWM(uartrx),
  .CURREN(1'b1),
  .RGB0(led_green),
  .RGB1(led_blue),
  .RGB2(led_red)
);
defparam RGB_DRIVER.RGB0_CURRENT = "0b000001";
```

```
  defparam RGB_DRIVER.RGB1_CURRENT = "0b000001";
  defparam RGB_DRIVER.RGB2_CURRENT = "0b000001";

endmodule
```