آزمایشگاه طراحی سیستمهای دیجیتال
گزارش آزمایش ۷

۲۰ مرداد ۱۴۰۳

امیرحسین ملک محمدی ۴۰۱۱۰۶۵۷۷
متین محمدی ۴۰۱۱۱۰۳۲۹
صادق محمدیان ۴۰۱۱۰۹۴۷۷

۱ شرح آزمایش

در این آزمایش می‌خواهیم یک UART طراحی کنیم. این قطعه از دو بخش فرستنده و گیرنده تشکیل شده است.

۱.۱ ورودی‌ها

ورودی‌های دستگاه شامل سیگنال کلاک، ورودی شروع ارسال داده، داده ۷ بیتی، و ورودی سریال است.

۲.۱ خروجی‌ها

خروجی‌های قطعه شامل خروجی سریال، سیگنال اتمام ارسال، داده دریافت شده، صحت داده دریافت شده(مقایسه بیت توازن)، و اتمام دریافت داده است.

۲ کد وریلاگ

۱.۲ قطعه اصلی

در شکل .Fig ۱ کد قطعه اصلی آمده.

این قطعه از دو بخش فرستنده و گیرنده تشکیل شده و ورودی‌ها را به آن‌ها می‌دهد و خروجی‌ها را از آن‌ها می‌گیرد. این ماژول یک کلاک ۵۰ MHz را در ورودی می‌گیرد و یک کلاک ۱۱۵۲۰۰Hz به ماژول‌های زیرین می‌دهد. برای این کار یک رجیستر clk۲ معرفی می‌کنیم که در هر ۲۱۷ کلاک مغدارش تغییر می‌کند.

۲.۲ ارسال‌کننده

در شکل .Fig ۲ کد ارسال‌کننده آمده.

این بخش یک متغییر ۴ بیتی به نام index دارد که نشان می‌دهد بیت چندم را باید ارسال کنیم. اول یک بیت که همان بیت شروع است را ارسال می‌کنیم. سپس داده ورودی را به ترتیب ارسال میکنیم و بعد بیت توازن را ارسال می‌کنیم. در نهایت نیز یک بیت ۱ که بیت پایان است را ارسال می‌کنیم.

در هر لبه بالارونده کلاک، اگر ورودی start فعال بود، مراحل فرستادن را شروع می‌کنیم و گرنه یا ارسال کردن را ادامه می‌دهیم یا اگر ارسال تمام شده بود، ۱ را خروجی می‌دهیم و خروجی اتمام ارسال را فعال می‌کنیم

۳.۲ گیرنده

در شکل .Fig ۳ کد گیرنده آمده.

این مدار دو حالت دارد، یکی حالت pre-start است (یعنی هنوز ورودی شروع نیامده) و حالت دیگر حالت recive است که یعنی ورودی شروع آمده و مدار در حال ورودی گرفتن است.

```
module UART(
    input clk,
    input start_sending,
    input [6:0] data_to_send,
    input Rx,

    output Tx,
    output sent,
    output [6:0] recived_data,
    output valid,
    output recived

);
reg [7:0] counter;
reg clk2;
always @(posedge clk)begin
    counter = counter + 8'd1;
    if(counter < 8'd 217) clk2 = 1'b0;
    else if (counter < 8'd 434)begin
        clk2 = 1'b1;
    end
    else begin
        counter = 8'd0;
        clk2 = 0;
    end
end

UART_sender sender (clk2, start_sending, data_to_send, Tx, sent);
UART_reciver reciver (clk2, Rx, valid, recived, recived_data);

endmodule

```

شکل ۱: کد وریلاگ قطعه اصلی

```
1   module UART_sender (
2       input clk,
3       input start,
4       input [6:0] data_in,
5
6       output reg data_out,
7       output reg done
8   );
9   wire [9:0] data_to_send ;
10  assign data_to_send[0] = 1'b0;
11  assign data_to_send[7:1] = data_in[6:0] ;
12  assign data_to_send[8] = ^data_in;
13  assign data_to_send[9] = 1'b1;
14  reg [3:0] index = 4'd0;
15
16  always @(posedge clk) begin
17      if (start) begin
18          index = 4'd0;
19          done = 1'b0;
20      end
21      else if (index < 10)begin
22          data_out = data_to_send[index];
23          index = index + 1;
24          if(index == 9)begin
25          done = 1;
26          end
27      end
28      else begin
29          data_out = 1'b1;
30          done = 1'b1;
31      end
32  end
33  endmodule
```

شکل ۲: کد وریلاگ ارسال‌کننده

```verilog
module UART_reciver (
    input clk,
    input data_in,

    output valid,
    output reg done,
    output [6:0]data_out
);

reg [7:0] data;

assign data_out = data[7:1];

wire parity = ^data[7:1];
wire parity_recived = data[0];
assign valid = (parity == parity_recived) & done;

localparam pre_start = 1'b0;
localparam recive = 1'b1;
reg state = pre_start;

reg [2:0] index = 3'd0;

always @(posedge clk) begin
    if (state == pre_start) begin
        if (data_in == 1'b1) state = recive;
        else state = pre_start;

        index = 3'd0;
        done = 1'b0;
    end
    else if (state == recive) begin
        if (index ==  3'd7) begin
            state = pre_start;
            done = 1'b1;
        end
        data[index] = data_in;
        index = index + 1;
    end
end
endmodule
```

شکل ۳: کد وریلاگ دریافت‌کننده

ورودی شروع همان بیت ۰ است. در اکثر اوقات در ورودی سریال بیت ۱ می‌آید ولی وقتی در هنگام ارسال داده در ابتدا یک بیت ۰ ارسال می‌شود و ما به حالت recive می‌رویم.

در این بخش یک متغییر ۳ بیتی به نام index داریم که نشان می‌دهد بیت ورودی را در کدام خانه ذخیره خواهیم کرد. زمانی که این متغییر به مقدار ۷ برسد یعنی فرایند ورودی گرفتن تمام شده.

خروجی valid وقتی ۱ می‌شود که فرایند دریافت تمام شده باشد و بیت توازن صحیح باشد.

۳ پیاده‌سازی روی بورد

این قطعه را دوبار روی FPGA پیدا می‌کنیم. بار اول Rx و Tx را به هم وصل می‌کنیم. در این مرحله انتظار داریم ورودیی که می‌دهیم را در خروجی ببینیم.

ابتدا یک پروژه کوارتوس باز می‌کنیم و فایل‌های وریلاگ را به آن اضافه می‌کنیم و سپس پروژه را کامپایل می‌کنیم.

در مرحله بعد پین‌های ورودی و خروجی را معین می‌کنیم. در شکل .Fig ۴ نحوه مشخص کردن پین‌های ورودی و خروجی آمده.

ورودی clk را از کلاک داخلی FPGA گرفتیم که فرکانسش ۵۰MHz بود. کلیدهای شماره ۰ تا ۶ برای ورودی داده استفاده می‌شد و کلید شماره ۷ برای شروع ارسال داده بود. LED های سبز شماره ۰ تا ۷ داده دریافت شده را نشان می‌دادند. از LED قرمز شماره ۱۷ برای نشان داده پایان ارسال داده استفاده می‌کنیم.

در شکل .Fig ۵ نتیجه این پیاده‌سازی را می‌بینیم. همانطور که در تصویر مشخص است ورودی به خروجی منتقل شده است.

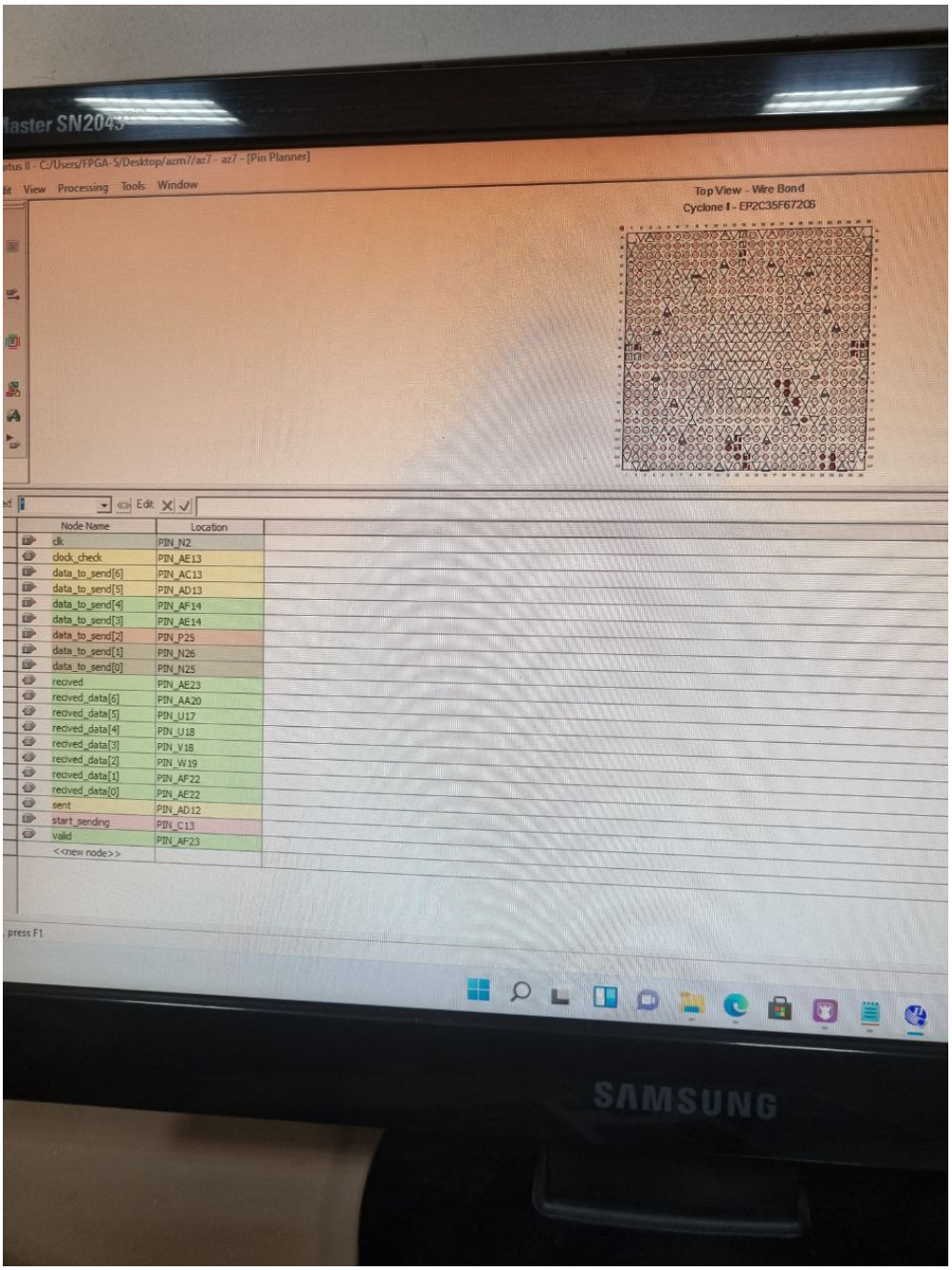
در پیاده‌سازی دوم، ورودی و خروجی را به مداری خارجی متصل می‌کنیم که آن خود به یک سرور وصل است. با این آزمایش نشان می‌دهیم که قطعه ما می‌تواند به هر مداری که روی فرکانس ۱۱۵۲۰۰Hz و با همین پروتکل کار می‌کند ارتباط برقرار کنیم.

در شکل .Fig ۶ نحوه مشخص کردن پین‌های ورودی و خروجی این پیاده‌سازی آمده. clk و خروجی مشابه قسمت قبل است و تنها تفاوت در ورودی سریال گرفتن و خروجی سریال دادن است. ورودی سریال را از پین Connection GPIO ۰ شماره ۰ می‌گیریم و خروجی سریال را در Con- GPIO nection ۰ شماره ۱ خروجی می‌دهیم. همچنین باید به دستگاه ورودی زمین دهیم که زمین فرستنده و گیرنده یکی باشد تا ۰ و ۱ ها به درستی ارسال شوند. این اتصالات در تصویر .Fig ۷ قابل مشاهده‌اند.

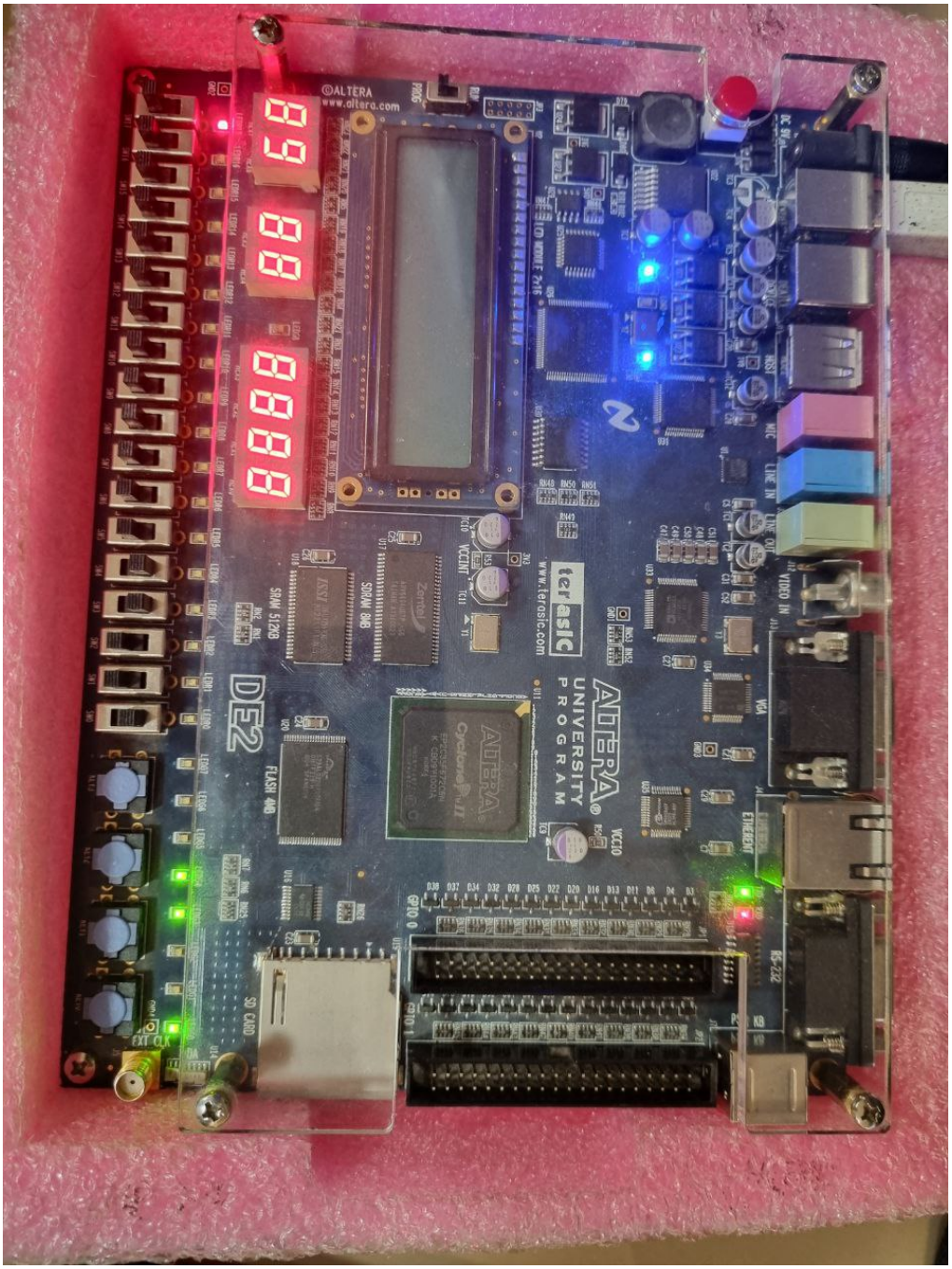
حال با اتصال دستگاهمان به یک مدار دیگر می‌توانیم از آن ورودی بگیریم یا خروجی دهیم. یک مثال از عملکرد درست دستگاه در ارسال کلمه به سرور در تصویر .Fig ۸ زیر نمایان است.

۴ نتیجه‌گیری

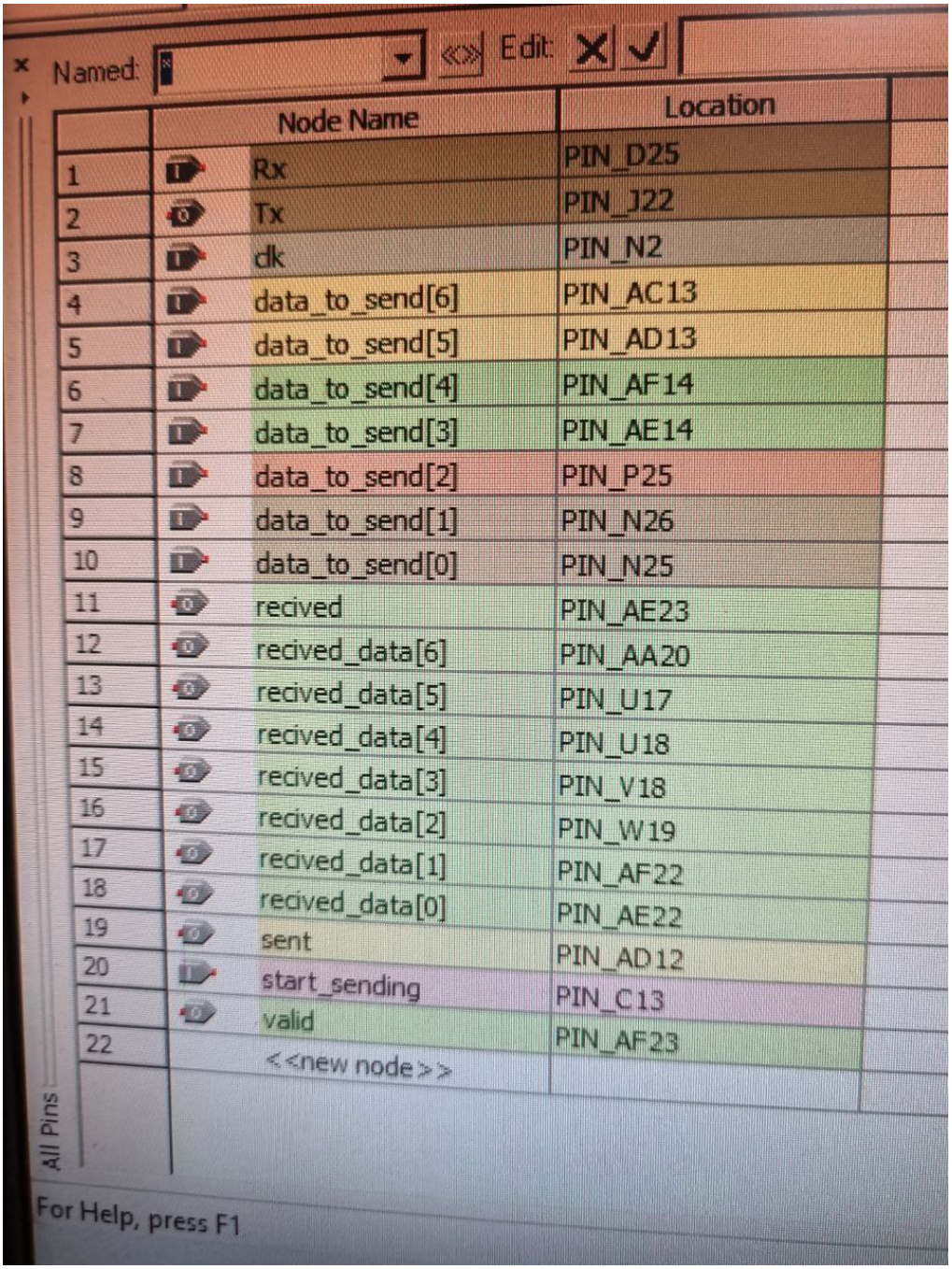
در این آزمایش ابتدا با زبان وریلاگ یک UART ساختیم. سپس دوبار آن را روی FPGA آپلود کردیم. یک بار خروجی سریالش را به ورودی خودش دادیم و از عملکرد قطعه اطمینان حاصل کردیم و بار دوم قطعه را به یک مدار دیگر وصل کردیم و بین این دو ارتباط برقرار کردیم. بعد از انجام این



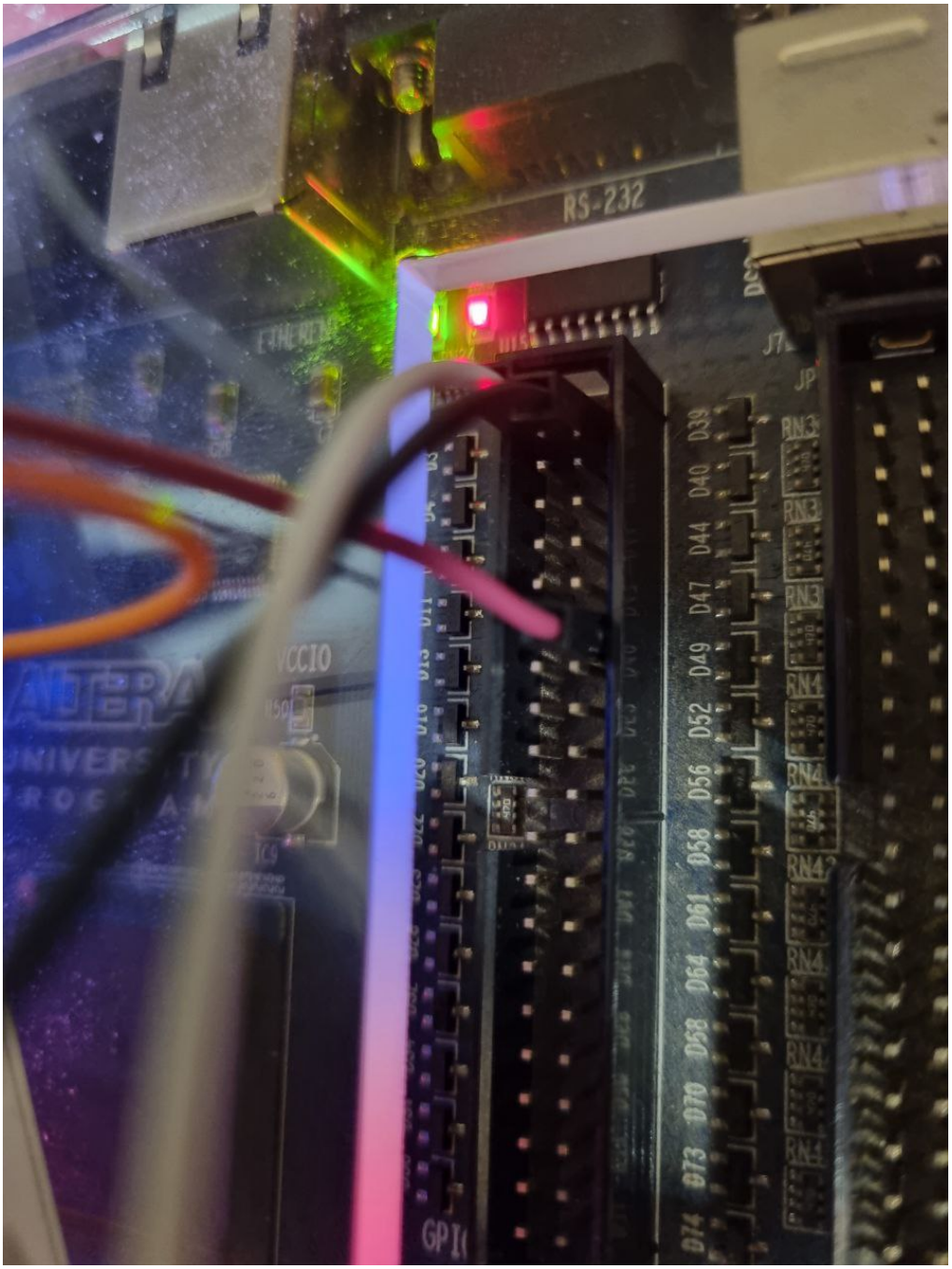
شکل ۴: پین‌های ورودی و خروجی پیاده‌سازی اول



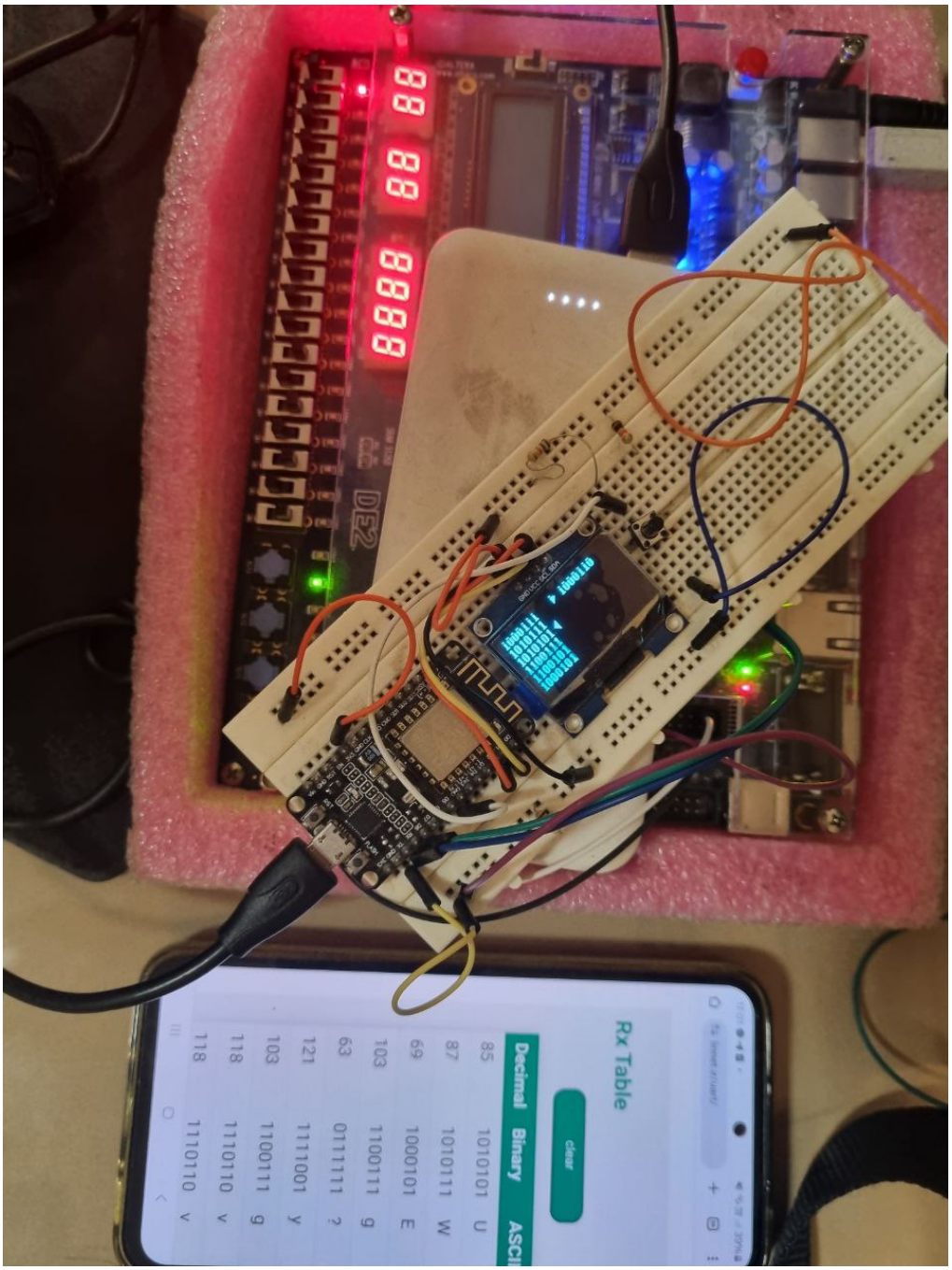
شکل ۵: نتیجه پیاده‌سازی اول



شکل ۶: پین‌های ورودی و خروجی پیاده‌سازی دوم



شکل ۷: پین‌های سریال و زمین



شکل ۸: نتیجه ارسال از قطعه به سرور

آزمایش دیدیم که چطور می‌توان به صورت سریال بین دو دستگاهی که حتی کلاک مشترک ندارند نیز ارتباط برقرار کرد.