

ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL
Facultad de Ingeniería en Electricidad y Computación



LABORATORIO DE SISTEMAS EMBEBIDOS

Paralelo del laboratorio: 111

Proyecto:

Sistema de monitoreo de signos vitales para pacientes
que requieren de una observación constante.

Estudiantes:

Abel Enrique Valenzuela Castro

Kevin Ricardo Rivera Moreira

Profesor teórico:

Ronald David Solís Mesa

Profesor práctico:

Alisson Asunción Constantine Macías

II TÉRMINO - 2021

Simulaciones

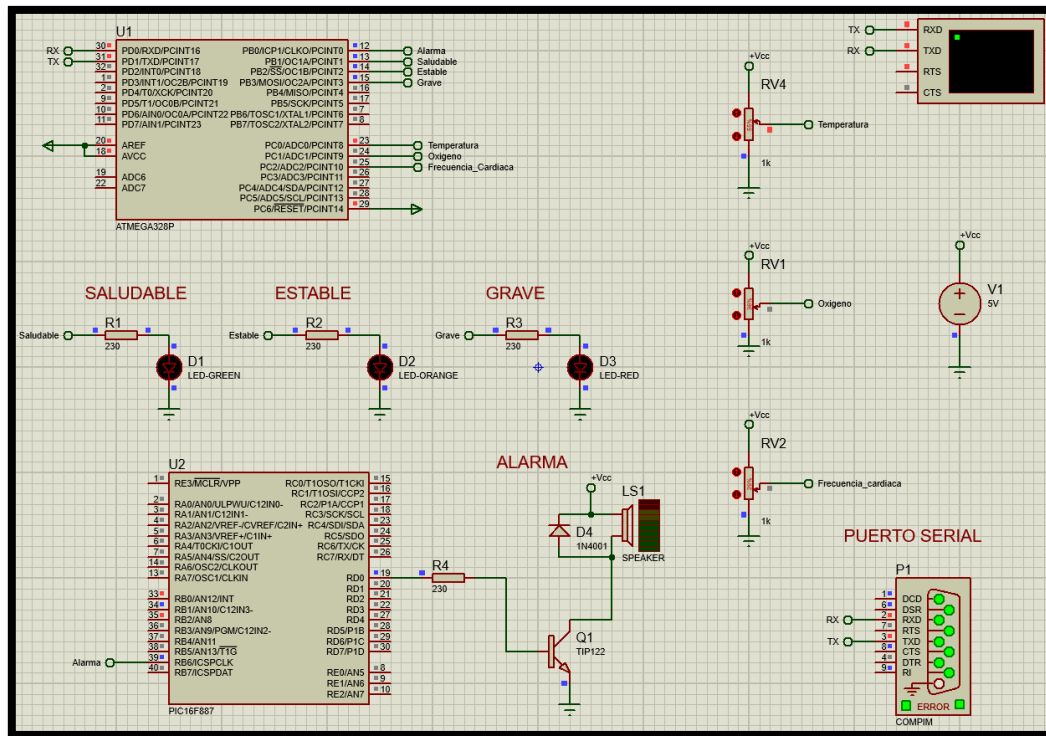


Figura 1: Circuito de proteus con el Atmega328p y conexión con la raspberry

```

Proyecto_SE_ATMEGA328P - Proteus 8 Professional - Source Code

main.c
1 #include <inttypes.h>
2 #include <avr/io.h>
3 #include <avr/interrupt.h>
4 #include <avr/sleep.h>
5 #include <util/delay.h>
6 #include "uart.h"
7
8 #include <stdio.h>
9
10 char* mensaje = "";
11
12 void ADC_Init(){
13     ADMUX &= ~(1<<ADLAR); //ajuste de salida hacia la derecha
14
15     //Voltaje de referencia AVcc +5V
16     ADMUX |= (1<<REFS0);
17     ADMUX &= ~(1<<REFS1);
18
19     //Divisor de frecuencia ===== 16000MHz / 128 = 125KHz
20     ADCSRA |= (1<<ADPS0);
21     ADCSRA |= (1<<ADPS1);
22     ADCSRA |= (1<<ADPS2);
23 }
24
25 int ADC_GetData(int puerto){
26     //selecciona el pin que va a muestrear
27     ADMUX &= 0x0F;
28     ADMUX |= puerto;
29
30     //activo el ADC
31     ADCSRA |= (1<<ADEN);
32     _delay_us(10);
33
34     //muestraa
35     ADCSRA |= (1<<ADSC);
36
37     //ESPERAMOS A QUE MUESTREE
38     while (!(ADCSRA & (1<<ADIF)));
39
40     //REINICIAMOS EL FLAG
41     ADCSRA |= (1<<ADIF);
42
43     //APAGAMOS ADC
44     ADCSRA &= ~(1<<ADEN);
45

```

```

Proyecto_SE_ATMEGA328P - Proteus 8 Professional - Source Code

main.c
42
43 //APAGAMOS ADC
44 ADCSRA &= ~(1<<ADEN);
45
46 return ADC;
47
48 float temperatura;
49 float oxigeno;
50 float frecuencia_cardiaca;
51 int ola;
52 //char string;
53
54 int main(void)
55 {
56     serial_begin();
57     DDRB = 0b00001111;
58     ADC_Init();
59     PORTB &= ~(0b00001111); //APAGO TODO
60
61     while (1){
62
63         if ((ADC_GetData(0) > 500) && (ADC_GetData(0) < 600) && (ADC_GetData(1) > 500) && (ADC_GetData(2) > 500)){
64
65             PORTB &= ~(0b00001111); //APAGO ALARMA, APAGO LUZ ESTABLE, APAGO LUZ GRAVE
66             PORTB |= (0b00000101); //PRENDIENDO LUZ SALUDABLE
67             PORTB &= ~(0b00001111); //APAGO TODO
68
69             if ((ADC_GetData(0) < 500) && (ADC_GetData(1) < 500) && (ADC_GetData(2) < 500)){
70                 PORTB &= ~(0b00001111); //APAGO ALARMA, APAGO LUZ SALUDABLE, APAGO LUZ GRAVE
71                 PORTB |= (0b00000100); //PRENDIENDO LUZ ESTABLE
72                 PORTB &= ~(0b00001111); //APAGO TODO
73             }
74             if ((ADC_GetData(0) == 1023) && (ADC_GetData(1) == 1023) && (ADC_GetData(2) < 1023)){
75                 PORTB &= ~(0b00000110); //APAGO LUZ SALUDABLE, APAGO LUZ ESTABLE,
76                 PORTB |= (0b00001001); //PRENDIENDO ALARMA Y PRENDIENDO LUZ GRAVE
77                 PORTB &= ~(0b00001111); //APAGO TODO
78                 char string=ADC_GetData(0);
79                 itoa(string,10);
80                 //printf("Set ola");
81                 serial_println_str(string);
82                 _delay_ms(2000);
83             }
84             return 0;
85         }
86     }

```

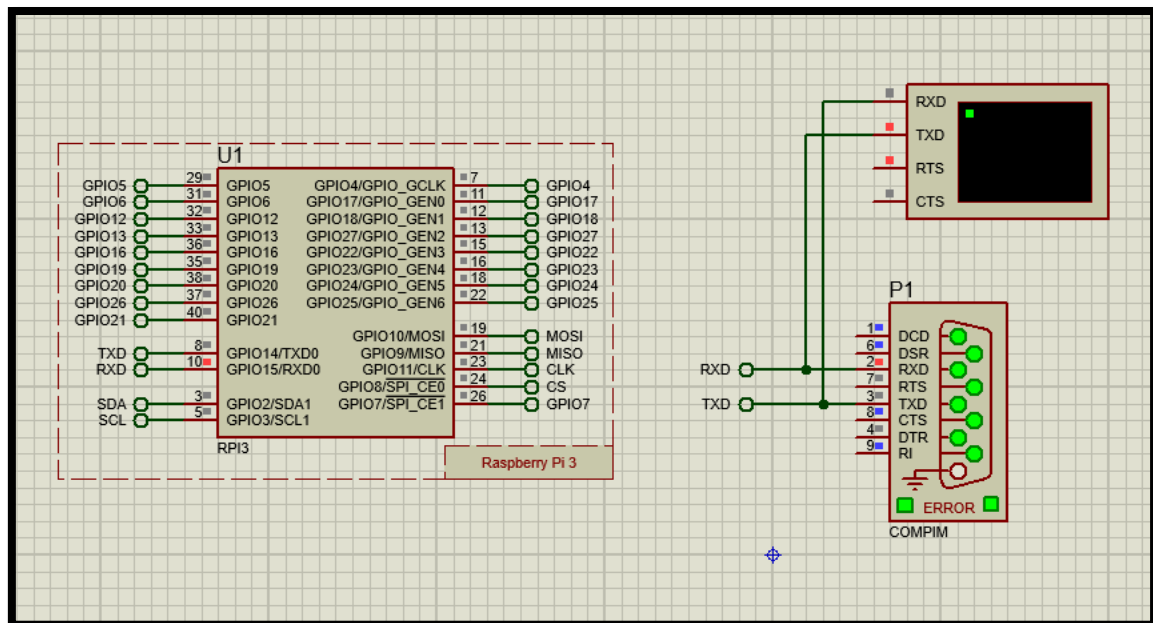


Figura 2: Circuito de proteus con la raspberry, la cual permite subir los datos a la nube

```

1
2 from time import*
3 import requests
4 import math
5 import random
6
7 TOKEN = "B8FF-5b4ER30BmWmyaEzDEzPc8NaULVva" # Put your TOKEN here
8 DEVICE_LABEL = "biotecc" # Put your device label here
9 VARIABLE_LABEL_1 = "temperatura" # Put your first variable label here
10 VARIABLE_LABEL_2 = "oxigeno" # Put your second variable label here
11 VARIABLE_LABEL_3 = "frecuenciaCardiaca" # Put your second variable label here
12
13
14 def build_payload(variable_1, variable_2, variable_3):
15     # Creates two random values for sending data
16     value_1 = random.randint(0, 100)
17     value_2 = random.randint(0, 200)
18     print ("CONECTADO...")
19
20
21 # Creates a random gps coordinates
22 lat = random.randrange(34, 36, 1) + \
23 random.randrange(1, 1000, 1) / 1000.0
24 lng = random.randrange(-83, -87, -1) + \
25 random.randrange(1, 1000, 1) / 1000.0
26
27 payload = {variable_1: value_1,
28            variable_2: value_2,
29            variable_3: {value_3: 1, "context": {lat: lat, "lng": lng}}}
30
31 return payload
32
33
34 def post_request(payload):
35     # Creates the headers for the HTTP requests
36     url = "http://industrial.api.ubidots.com"
37     url = "[/api/v1.6/devices/{}].format(url, DEVICE_LABEL)
38     headers = {"X-Auth-Token": TOKEN, "Content-Type": "application/json"}
39
40 # Makes the HTTP requests
41 status = 400
42 attempts = 0
43 while status >= 400 and attempts <= 5:
44     req = requests.post(url=url, headers=headers, json=payload)
45     status = req.status_code
46     attempts += 1
47     sleep(1)
48
49 # Processes results
50 if status >= 400:
51     print([ERROR] Could not send data after 5 attempts, please check \
52         your token credentials and internet connection)
53     return False
54
55 print([INFO] request made properly, your device is updated")
56 return True
57
58
59 while(1):
60     payload = build_payload(
61         VARIABLE_LABEL_1, VARIABLE_LABEL_2, VARIABLE_LABEL_3)
62     print([INFO] Attempting to send data")
63     post_request(payload)
64     print([INFO] finished")
65     sleep(10)
66
67
68

```

```

1
2 random.randrange(1, 1000, 1) / 1000.0
3
4 payload = {variable_1: value_1,
5            variable_2: value_2,
6            variable_3: {value_3: 1, "context": {lat: lat, "lng": lng}}}
7
8 return payload
9
10
11 def post_request(payload):
12     # Creates the headers for the HTTP requests
13     url = "http://industrial.api.ubidots.com"
14     url = "[/api/v1.6/devices/{}].format(url, DEVICE_LABEL)
15     headers = {"X-Auth-Token": TOKEN, "Content-Type": "application/json"}
16
17 # Makes the HTTP requests
18 status = 400
19 attempts = 0
20 while status >= 400 and attempts <= 5:
21     req = requests.post(url=url, headers=headers, json=payload)
22     status = req.status_code
23     attempts += 1
24     sleep(1)
25
26 # Processes results
27 if status >= 400:
28     print([ERROR] Could not send data after 5 attempts, please check \
29         your token credentials and internet connection)
30     return False
31
32 print([INFO] request made properly, your device is updated")
33 return True
34
35
36 while(1):
37     payload = build_payload(
38         VARIABLE_LABEL_1, VARIABLE_LABEL_2, VARIABLE_LABEL_3)
39     print([INFO] Attempting to send data")
40     post_request(payload)
41     print([INFO] finished")
42     sleep(10)
43
44
45

```

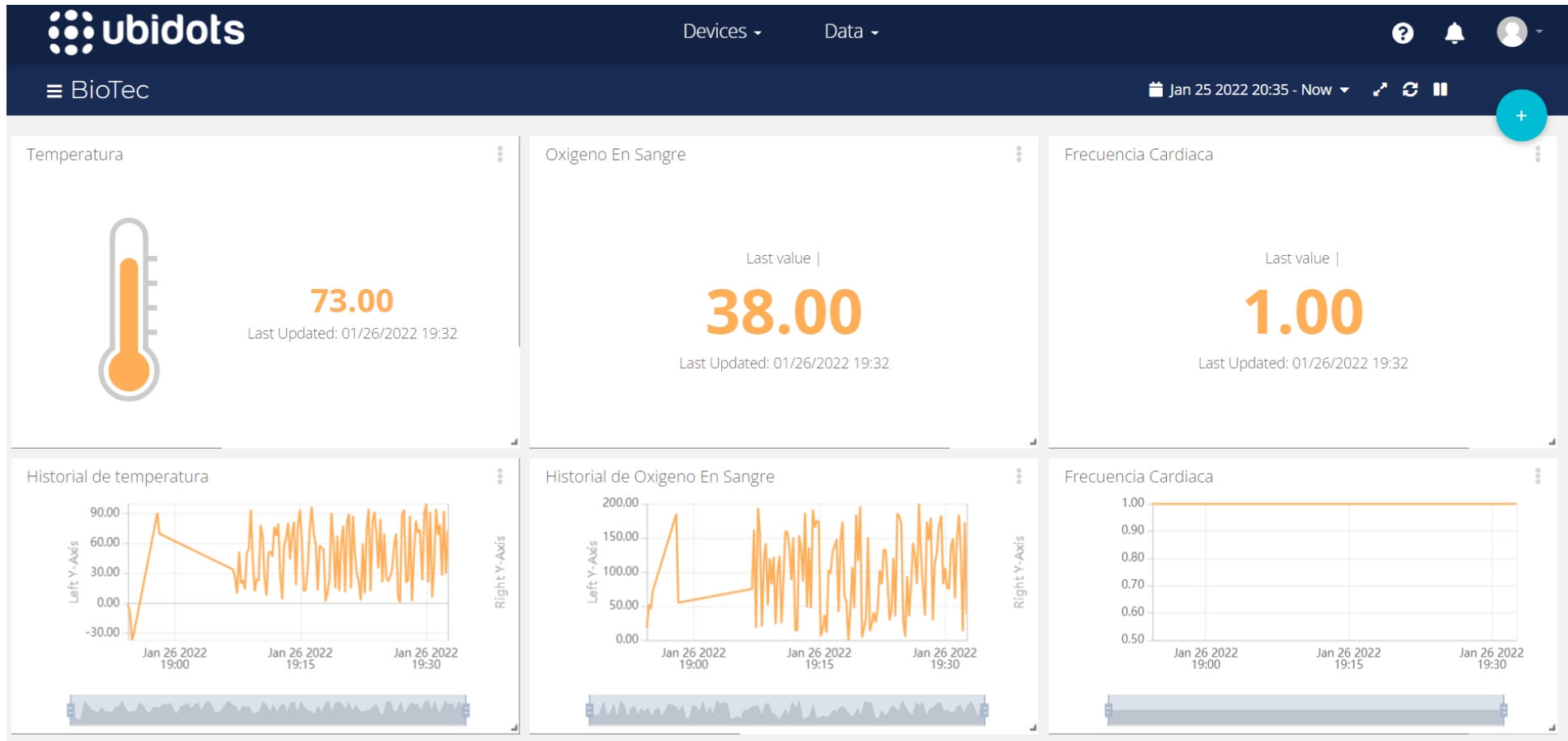


Figura 3: Datos del paciente que el doctor puede observar en la nube