**Pulse Oximetry and Heart-Rate**

Introduction:

In this project, we seek to monitor a patient’s heart rate and blood-oxygen level using a pulse oximeter. The pulse oximeter is designed using infrared and visible (red) light detection from light that passes through a patient’s finger from an emitter. The absorption will tell when blood is moving through the finger and how much of this is oxygen-rich. The output of this analog circuit will be fed into an NodeMCU microcontroller, which will compute the pulse and oxygen level from these numbers. The values are uploaded to a cloud computing web host called Thingspeak from where it can be viewed.

High Level Design / Background:

Pulse oximeters have been used in medical settings for many years. In many cases, such as during an operation, in intensive care, the emergency room, even an unpressurized aircraft, a person’s oxygen level may be unstable and needs monitoring. In addition, from these readings, the person’s heart rate can also be determined. This project is an attempt to construct a working version of a pulse oximeter from a relatively cheap set of parts – including a microcontroller. An off-the-shelf microcontroller has enough processing power to perform the tasks required for this design; however, in any commercial application, specialized hardware will be designed that is specifically suited to the task.

The sampling portion of this design requires an infrared emitter (around 940 nm wavelength) and a red light emitter (around 660 nm wavelength). Absorption of oxyhemoglobin and the deoxygenated formdiffers significantly between these wavelengths. Therefore, using the ratio of the two absorption values gives the percentage of arterial hemoglobin for oxyhemoglobin. The detectors do not give a very high voltage, so the output from the detector needs to be amplified using op amps before passing into the microcontroller for analysis. If not, the relative change will not be seen when the microcontroller makes the input a discreet value.

This attempt at a pulse oximeter is fairly crude and does not take into consideration some important facts if it were to be used in a serious situation. For instance, it does not take into account other gasses in the blood stream. If a person has been rescued from a burning building, they may have carbon monoxide poisoning. In order to distinguish the difference between CO and O2, absorption at additional wavelengths must be performed. Another example is a person suffering from poor gas exchange in the lungs. Their blood may have a 100% oxygen level, but may still be suffering from too much carbon dioxide (CO2) that cannot be exchanged and exhaled.

The microcontroller is required to perform a discrete Fourier transform to determine the pulse. This transform will take a collection of data over time and extract the amplitude of each of the frequencies it contains. In the case of our data, there should be a pretty distinct pattern of when there is blood movement. Therefore, we should obtain one frequency where the amplitude is much higher than any other frequencies detected. This should correspond to the frequency of the pulse of the person using the device.

In most design projects, there is a tradeoff to what should be done with hardware or with software. In our project, there is not much of a design comparison. The sampling and amplification must be done in hardware with analog values to obtain the correct results. For the calculations and the video generation, we need a device with enough processing power and features to perform meet all the timing requirements. In this case, the NodeMCU is a good fit at a low cost.

**Hardware Design:**

The preprocessing of the photodiode and led would take considerable time and resources to get the desired signal. Hence we use a sensor called Protocentral MAX30100. The sensor contains the necessary circuits to obtain the desired signals.The MAX30100 is an integrated pulse oximetry and heart-rate monitor sensor solution. It combines two LEDs, a photodetector, optimized optics, and low-noise analog signal processing to detect pulse oximetry and heart-rate signals. The MAX30100 operates from 1.8V and 3.3V power sup­plies and can be powered down through software with negligible standby current, permitting the power supply to remain connected at all times.

## Electrical Characteristics

(VDD = 1.8V, VIR\_LED+ = VR\_LED+ = 3.3V, TA = +25°C, min/max are from TA = -40°C to +85°C, unless otherwise noted.) (Note 2)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **PARAMETER** | **SYMBOL** | **CONDITIONS** | | **MIN** | **TYP** | **MAX** | **UNITS** |
| **POWER SUPPLY** | | | | |  |  | |
| Power-Supply Voltage | VDD | Guaranteed by RED and IR count tolerance | | 1.7 | 1.8 | 2.0 | V |
| LED Supply Voltage  (R\_LED+ or IR\_LED+ to PGND) | VLED+ | Guaranteed by PSRR of LED Driver | | 3.1 | 3.3 | 5.0 | V |
| Supply Current | IDD | SpO2 and heart rate modes,  PW = 200µs, 50sps | |  | 600 | 1200 | µA |
|  | |  |  |  |
|  |  | Heart rate only mode,  PW = 200µs, 50sps | |  | 600 | 1200 |  |
| Supply Current in Shutdown | ISHDN | TA = +25°C, MODE = 0x80 | |  | 0.7 | 10 | µA |
| **SENSOR CHARACTERISTICS** | | | | |  |  | |
| ADC Resolution |  |  | |  | 14 |  | bits |
| Red ADC Count  (Note 3) | REDC | Propriety ATE setup  RED\_PA = 0x05, LED\_PW = 0x00,  SPO2\_SR = 0x07, TA = +25°C | | 23,000 | 26,000 | 29,000 | Counts |
| IR ADC Count  (Note 3) | IRC | Propriety ATE setup  IR\_PA = 0x09, LED\_PW = 0x00,  SPO2\_SR = 0x07, TA = +25°C | | 23,000 | 26,000 | 29,000 | Counts |
| Dark Current Count | DCC | RED\_PA = IR\_PA = 0x00,  LED\_PW = 0x03, SPO2\_SR = 0x01 | |  | 0 | 3 | Counts |
| DC Ambient Light Rejection  (Note 4) | ALR | Number of ADC counts with finger on sensor under direct sunlight (100K lux)  LED\_PW = 0x03,  SPO2\_SR = 0x01 | RED LED |  | 0 |  | Counts |
| IR LED |  | 0 |  |

**System Block Diagram :**

1



ADC

CONTROL

SIGNAL

PROCESSING

COVER GLASS

10

0.1

RED

IR

HbO

2

Hb

NO INK

## Pin Configuration

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | | | | 1 | 12  14  13  SENSOR  **MAX30100**  11  10 | | |  | | 2 | |  | | 3 | |  | | 4 | |  | | 5 | |  | | 6 | LED | 9 | |  |  | | 7 | 8 | |  |  |   N.C.N.C.  SCLINT  SDAGND  PGNDVDD  IR\_DRVIR\_LED+ R\_DRVR\_LED+ N.C.N.C. |

## Pin Description

|  |  |  |
| --- | --- | --- |
| **PIN** | **NAME** | **FUNCTION** |
| 1, 7, 8, 14 | N.C. | No Connection. Connect to PCB Pad for Mechanical Stability. |
| 2 | SCL | I2C Clock Input |
| 3 | SDA | I2C Clock Data, Bidirectional (Open-Drain) |
| 4 | PGND | Power Ground of the LED Driver Blocks |
| 5 | IR\_DRV | IR LED Cathode and LED Driver Connection Point. Leave floating in circuit. |
| 6 | R\_DRV | Red LED Cathode and LED Driver Connection Point. Leave floating in circuit. |
| 9 | R\_LED+ | Power Supply (Anode Connection) for Red LED. Bypass to PGND for best performance. Connected to IR\_LED+ internally. |
| 10 | IR\_LED+ | Power Supply (Anode Connection) for IR LED. Bypass to PGND for best performance. Connected to R\_LED+ internally. |
| 11 | VDD | Analog Power Supply Input. Bypass to GND for best performance. |
| 12 | GND | Analog Ground |
| 13 | INT | Active-Low Interrupt (Open-Drain) |

## Functional Diagram

nm

880

nm

660

ADC

AMBIENT LIGHT

CANCELLATION

ANALOG

ADC

TEMP

OSCILLATOR

DIGITAL

FILTER

DIGITAL

DATA

REGISTER

LED DRIVERS

I

2

C

COMMUNICATION

INT

SDA

SCL

V

DD

IR\_LED+

R\_

LED+

IR\_DRV

R\_DRV

GND

PGND

RED

IR

RED+IR

### SpO2 Subsystem

The SpO2 subsystem in the MAX30100 is composed of ambient light cancellation (ALC), 16-bit sigma delta ADC, and proprietary discrete time filter.

The SpO2 ADC is a continuous time oversampling sigma delta converter with up to 16-bit resolution. The ADC output data rate can be programmed from 50Hz to 1kHz. The MAX30100 includes a proprietary discrete time filter to reject 50Hz/60Hz interference and low-frequency residual ambient noise.

### Temperature Sensor

The MAX30100 has an on-chip temperature sensor for (optionally) calibrating the temperature dependence of the SpO2 subsystem.

The SpO2 algorithm is relatively insensitive to the wavelength of the IR LED, but the red LED’s wavelength is critical to correct interpretation of the data. The temperature sensor data can be used to compensate the SpO2 error with ambient temperature changes.

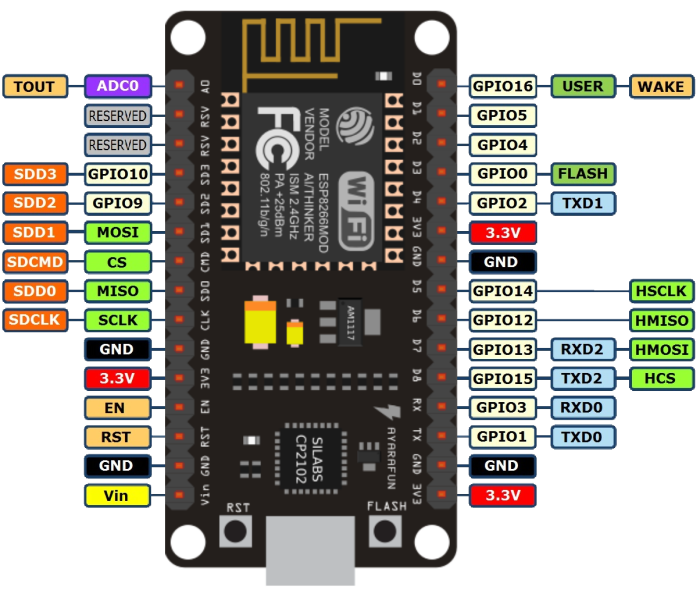
**LED Driver**

The MAX30100 integrates red and IR LED drivers to drive LED pulses for SpO2 and HR measurements. The LED current can be programmed from 0mA to 50mA (typical only) with proper supply voltage. The LED pulse width can be programmed from 200µs to 1.6ms to optimize measurement accuracy and power consumption based on use cases.

**Microcontroller:**

The microcontroller used for interfacing the sensor is the NodeMCU. It is a lua based microcontroller consisting of an ESP8266 WiFi module. This is the controller used for communicating the results to online web server.

The Pin diagram for the controller is as follows :



**Program:**

/\*

NodeMCU-MAX30100 oximetry / heart rate program project

\*/

#include <Wire.h>

#include "MAX30100\_PulseOximeter.h"

#include <ESP8266WiFi.h>

// Replace with your network details

const char\* ssid = "Kuruganti"; //Replace with your WiFi Name

const char\* password = "rockstar"; //Your WiFi password

//Thingspeak Server Host and API Key

const char\* server\_thingspeak = "api.thingspeak.com"; //The Thingspeak Host Server

String apiKey = "C53MOK04AYJ1RAME"; //Replace API Key with your Thingspeak Channel Key

// Web Server on port 80

WiFiServer server(80);

WiFiClient thingspeak;

#define REPORTING\_PERIOD\_MS 5000

#define THINGSPEAK\_MS 15000

// PulseOximeter is the higher level interface to the sensor

// it offers:

// \* beat detection reporting

// \* heart rate calculation

// \* SpO2 (oxidation level) calculation

PulseOximeter pox;

uint32\_t tsLastReport = 0;

uint32\_t thingspeakLastTransmit = 0;

float hr,spo2,temp;

// Callback (registered below) fired when a pulse is detected

void onBeatDetected()

{

Serial.println("Beat!");

}

void setup()

{

Serial.begin(57600);

Serial.println("Initializing MAX30100");

// Initialize the PulseOximeter instance and register a beat-detected callback

pox.begin();

pox.setOnBeatDetectedCallback(onBeatDetected);

// Connecting to WiFi network

Serial.println();

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED)

{

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

}

void loop()

{

// Make sure to call update as fast as possible

pox.update();

// Asynchronously dump heart rate and oxidation levels to the serial

// For both, a value of 0 means "invalid"

if (millis() - tsLastReport > REPORTING\_PERIOD\_MS) {

Serial.print("Heart rate:");

hr = pox.getHeartRate();

Serial.print(hr);

Serial.print("bpm / SpO2:");

spo2 = pox.getSpO2();

Serial.print(spo2);

Serial.print("% / temp:");

temp = pox.getTemperature();

Serial.print(temp);

Serial.println("C");

tsLastReport = millis();

if (millis() - thingspeakLastTransmit > THINGSPEAK\_MS)

{

if (thingspeak.connect(server\_thingspeak, 80)) {

// "184.106.153.149" or api.thingspeak.com

String postStr = apiKey;

postStr += "&field1=";

postStr += hr;

postStr += "&field2=";

postStr += spo2;

postStr += "&field3=";

postStr += temp;

thingspeak.print("POST /update HTTP/1.1\n");

thingspeak.print("Host: api.thingspeak.com\n");

thingspeak.print("Connection: close\n");

thingspeak.print("X-THINGSPEAKAPIKEY: " + apiKey + "\n");

thingspeak.print("Content-Type:application/x-www-form-urlencoded\n");

thingspeak.print("Content-Length: ");

thingspeak.print(postStr.length());

thingspeak.print("\n\n");

thingspeak.print(postStr);

//Serial.print(" degrees Celcius Humidity: ");

//Serial.print(h);

Serial.println(" Sending to Thingspeak...");

}

thingspeakLastTransmit = millis();

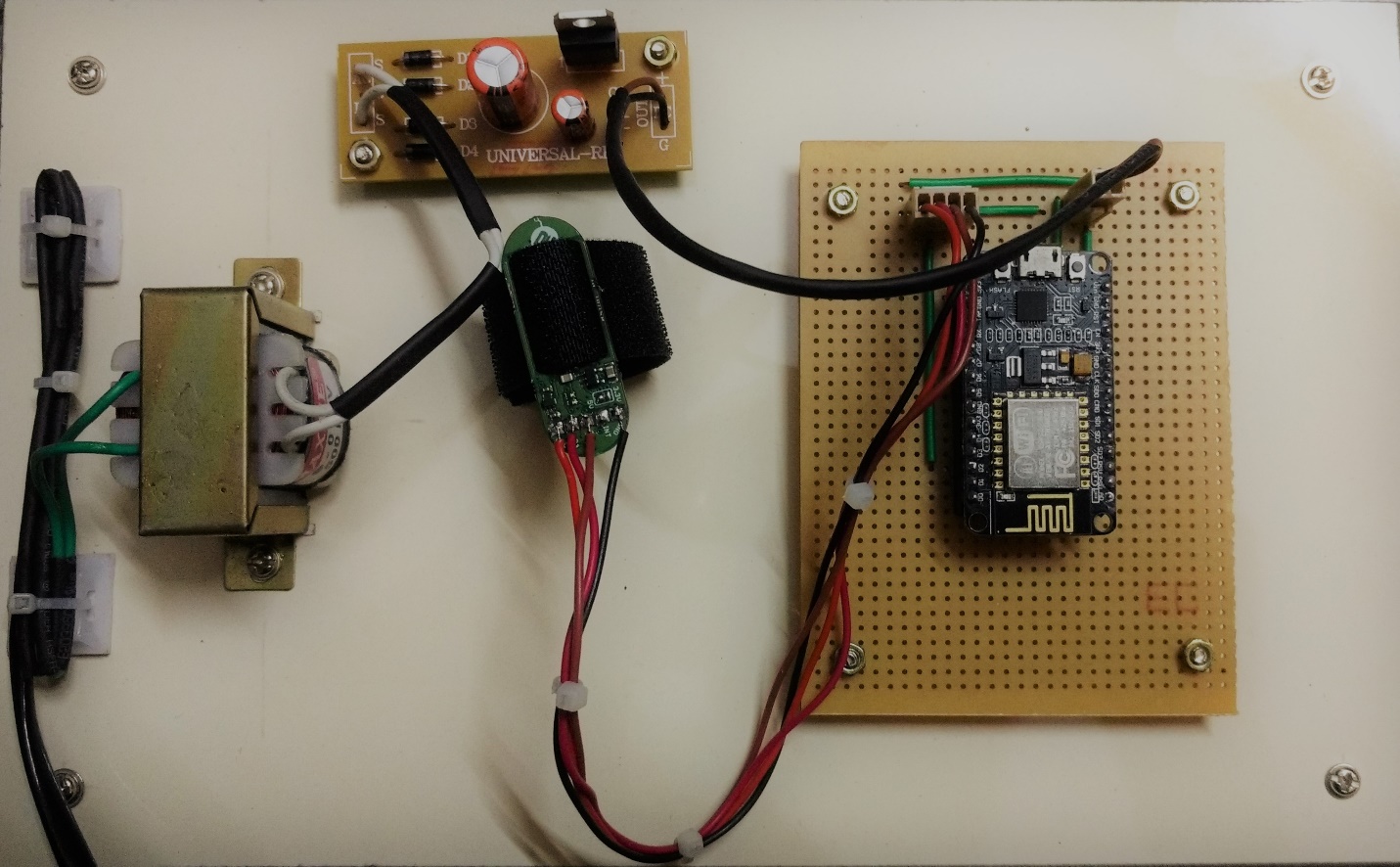
thingspeak.stop();

}

}

}

**Final Results:**

****

**Conclusion :**

The project has been a relative success compared to mainstream Pulse Oximetry and Pulse-Rate devices. Thus our aim to make the system accessible and portable has been proved and tested thorougly. With the connection with the Internet, even remote doctors can assess the condition of the person by checking the result from the web. Hence, the project has proved to be fruitful despite its challenges.