# Interfacing with STM32 with components:

1. Temperature sensor : we need to connect it with analog pin
2. Battery (9V): we need to divide the voltage as it cannot be directly fed to the board and may damage it, use resistors for that.
3. LED’s: Only thing which necessary to be considered in case of LED’s is to calculate the resistors and also take its voltage and current characteristics.

Note: Consider the datasheets of each components as ultimate guide.

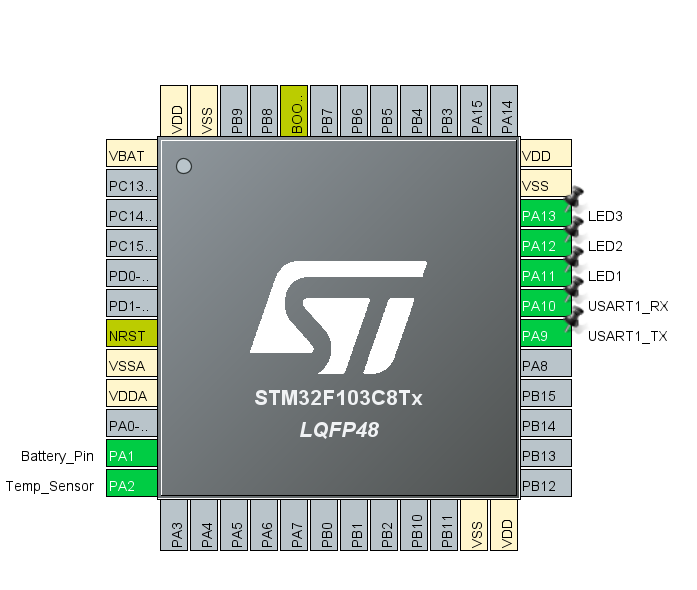


Figure 1 STM Pinout

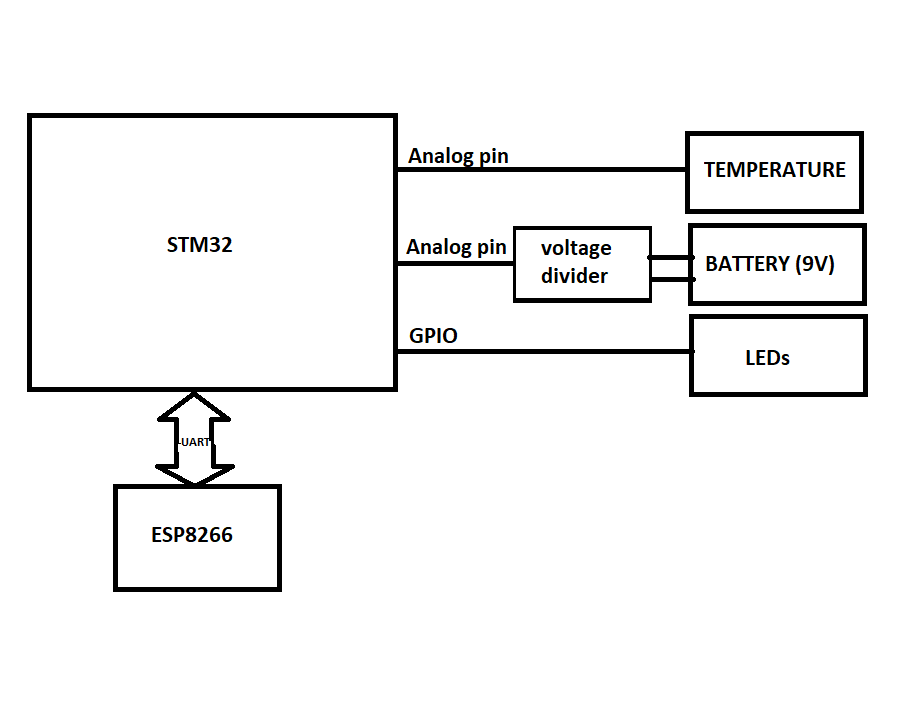


Figure 2 BLOCK diagram

# Interfacing ESP8266 with the STM32:

1. Use UART to send/receive data with each other.
2. STM32 will send the temperature and voltage rating.
3. ESP8266 will send the instruction to turn ON/OFF LED.

# Communication of ESP8266 with web server:

1. A web page should display temperature and voltage data received from ESP8266.
2. It should contain LED buttons to turn ON/OFF LEDs.

# Measurements:

We are using STM32F334 to measure the Battery voltage rating and temperature ratings

But before that Let’s understand ADC in STM32

## ADC

* There are two ADCs ADC1 and ADC2
* Each ADC converts analog signal into the 12 bit number
* So resolution is 12 bit

## ADC modes:

1. **Single conversion mode**

* It will perform one conversion and will stop the
* Again we have to perform another conversion

1. **Single Continuous conversion mode**
   * Will start the new conversion as long as old conversion is done
2. **Scan mode**
   * Used when there are more than channel
   * Every channel will be given the rank and depending upon that further process will be done.

## Internal Temperature calculation:

Example 1:

PCLK = 72MHz

Tconv = sampling time + 12.5 ADC clock cycles (sampling time >=2.2 microsecond)

example: 1.5 + 12.5 = 14 ADC cycles = (1/PCLK) \* 14 microsecond (this should be >= 2.2 microsecond)

Example 2:

Tconv = ST + 12.5 cycles

(ST + 12.5) \* 1/36 >= 2.2

(ST + 12.5) \* 0.028 >= 2.2

(ST + 12.5) >= 78.57

ST >= 66.07

# 4PIN NTC Thermistor Temperature Sensor Module

## Understanding the analog:

Controller works on 0 to 5V range

Whatever the value we get in the range of 0 to 5V controller convert it into 0 to 1023 numbers

By this method: ((0 to 5V) / (max Voltage ) \* 1024) -1 //1024 came from 2^10 because of 10 bit resolution of the ADC used in Arduino this number (1024) depends on the resolution of the ADC peripheral that is being used in microcontroller

Example: ((2.5/5)\*1024) -1) will give 512-1=511

## General information

* Low cost and small size
* Possible change to its threshold
* Temperature detection ranges from 20 to 80 degree Celsius
* Working voltage 3.3V-5V
* wide voltage comparator LM393

**Steps to get the value of the temperature out of voltage ratings:**

1 / ( log(SERIES\_RESISTOR / (1023/(analog value) – 1)) / BCOEFFICIENT + 1 / (TempNom + 273.15) ) - 273.15 is final temperature value in celcius

**MACROS**

#define THERMISTORNOMINAL 10000 // resistance of thermistor at 25 degrees C

#define TEMPERATURENOMINAL 25

#define NUMSAMPLES 10 //for we will be taking 10 samples

#define BCOEFFICIENT 3950 //Beta coefficient from datasheet

#define SERIESRESISTOR 10000 //the value of the R1 resistor

**Conversion:**

//resistence to temperature

float temperature;

temperature = average / THERMISTORNOMINAL;

temperature = log(temperature); //use math library for it

temperature /= BCOEFFICIENT;

temperature += 1.0 / (TEMPERATURENOMINAL + 273.15);

temperature = 1.0 / temperature;

temperature -= 273.15; // convert to C