



18/03/2019 04/03/2020 José Moraes 13

Comentários

ESP32 – Analisando e corrigindo o ADC interno

José Moraes

ÍNDICE DE CONTEÚDO



Talvez você já tenha visto em algum lugar que o ADC do ESP32 não é linear e tem muito erro nas leituras, mas o que isso significa na prática? Como



podemos resolver? Quem irá nos ajudar? É o que vamos analisar e tentar corrigir neste artigo dedicado ao ADC do ESP32.



Figura 1 - ESP32 ADC.

ESP32 e seu peculiar ADC

Depois de muitas reclamações a respeito do ADC no ESP8266, principalmente sobre conter apenas 1 canal (1 V, 10 bits, SAR ADC) para essa tarefa, a Espressif ouviu os seus usuários. No ESP32 incluiu 18 canais (1,1 - 3,9 V, 9 - 12 bits, SAR ADC), que foi separado em 2 controladores, ADC1 (8 canais) e ADC2 (10 canais). Isso nos permite, por exemplo, ler 2 canais paralelamente ou até com DMA para protocolos de áudio e vídeo como I2S, um baita avanço!

Alguns detalhes curiosos do ADC no ESP32

- Ambos controladores podem ser usados no domínio digital e RTC, que são focados em velocidade (2 M Samples/Seg) e baixo consumo (200 K Samples/Seg), respectivamente;
- Resolução configurável: 9, 10, 11 e 12 bits;
- Tensão máxima configurável (atenuação interna): 0 dB (1,1 V), 2,5 dB (1,5 V), 6 dB (2,2 V), 11 dB (3,9 V limitado pelo VDD_A);
- Faixas de tensão recomendadas para melhor precisão:
 - 0dB: 100 - 950mV;
 - 2.5dB: 100 - 1250mV;
 - 6dB: 150 - 1750mV;



- 11dB: 150 - 2450mV.
- O ADC2 não pode ser usado enquanto o Wi-Fi estiver ligado, pois o canal



depois ligá-lo novamente;

- As medições do ADC ficarão mais ruidosas enquanto WiFi estiver ligado, muitas vezes por conta de má alimentação e/ou filtragem do sinal;
- ULP permite leitura ADC mesmo durante *deep sleep*;
- Cada ESP32 pode ter até 6 % de diferença nas leituras, boa parte devido à tensão de referência (V_{ref}) do ADC haver grande variação (1000 - 1200 mV). Essa variação na V_{ref} ocasiona leituras diferentes **entre os chips**, veja na figura 1 abaixo a comparação de 2 ESP32 com V_{ref} diferentes.

Figura 2 - Leituras com V_{ref} diferentes.

Testando o ADC

Agora que já sabemos um pouco mais sobre os detalhes do ADC, vamos fazer alguns testes na prática para analisar suas curvas.

As medições foram feitas com média simples de 100 amostras intervaladas por 30 us cada, sem capacitores ou qualquer método além da média simples para filtragem. Em todos os testes, vamos adotar uma leitura com o V_{ref} padrão ("ADC Descal") e outra com a API de calibração utilizando o V_{ref} verdadeiro ("ADC Cal"), que vem gravado na memória dos ESP32 mais novos, fabricados de 2018 para frente. Após os testes, vamos entender como utilizar a API de calibração disponibilizada pela Espressif.



A fórmula utilizada nas medições sem a calibração foi:



V_m: 1,1 ou 3,3 V.

Resolução: 1024 ou 4096.

ADC: Valor lido do pino.

Teste A

Nesse teste (figuras 3 e 4), vamos analisar o ADC1 (GPIO36) e seu erro com 0 dB para as seguintes opções:

- Wi-Fi **OFF**;
- 10 e 12 bits;
- 0 dB (0 - 1100 mV);
- Vref ADC Descal: 1100 mV;
- Vref ADC Cal: 1072 mV.

Figura 3 - Curvas 0dB.





Figura 4 - Erro das curvas 0dB.

Observando as figuras 3 e 4 que mostram a leitura e erro do ADC para “0 dB, 10 e 12 bits com Wi-Fi OFF”, podemos ver que a curva utilizando a API de calibração se manteve com erro ≤ 5 mV em toda faixa de tensão recomendada, tendendo a 0 mV. As leituras de 10 e 12 bits foram praticamente idênticas, não havendo alguma discrepância considerável na ordem dos mV, mas caso você precise de leituras mais precisas e consistentes, refaça os testes com sua placa e análise os resultados.

Apesar do erro com a calibração tendendo a ~ 0 mV na faixa de tensão recomendada pela Espressif (100 - 950 mV), fora desses limites, o erro é grande e pode ser totalmente inaceitável, descartando essas leituras em muitos projetos.

Teste B

Nesse teste (figuras 5 e 6), vamos analisar o ADC1 (GPIO36) e seu erro com 0 dB para as seguintes opções:

- WiFi **ON**;
- 10 e 12 bits;
- 0dB (0 - 1100 mV);
- Vref ADC Descal: 1100 mV;
- Vref ADC Cal: 1072 mV.





Figura 5 - Curvas 0dB.

Figura 6 - Erro curvas 0dB.

Observando as figuras 5 e 6 que se diferenciam do **teste A** apenas pelo Wi-Fi ON, podemos ver que os erros na medição calibrada, próximo ao fim, foi maior que com o Wi-Fi OFF. Além disso, em toda a faixa de medição, houve ruído perceptível no gráfico, podendo haver necessidade de filtros analógicos ou digitais melhores.

Agora vamos mudar um pouco as coisas, onde a maioria que programa pela Arduino IDE reside. Os dois testes a seguir serão com a atenuação interna de 11 dB, que permite a leitura de 0 - 3,9 V (limitado pelo VDD_A), padrão na Arduino IDE. A própria Espressif diz que com 11 dB perde a linearidade enquanto 0 dB não, então vamos observar.

Teste C

Nesse teste (figuras 7 e 8), vamos analisar o ADC1 (GPIO36) e seu erro com 11 dB para as seguintes opções:

- WiFi **OFF**;



- 10 e 12 bits;
- 11 dB (0 - 3900 mV);



- Vref ADC Cal. 10/2 mV.
-

Figura 7 - Curvas 11dB.

Figura 8 - Erro curvas 11dB.

Podemos perceber que os erros aumentaram e também ficou bem menos linear, principalmente após 2 V. Mesmo utilizando a calibração com Vref verdadeiro, não foi o suficiente para aproximar o erro de ~0 mV, ficando próximos a ~25 mV na faixa recomendada.

Teste D

Nesse teste (figuras 9 e 10), vamos analisar o ADC1 (GPIO36) e seu erro com 11 dB para as seguintes opções:

- WiFi **ON**;
- 10 e 12 bits;



- 11 dB (0 - 3900 mV);
- Vref ADC Descal: 1100 mV;



Figura 9 - Curvas 11dB.

Figura 10- Erro curvas 11dB.

Novamente, repetimos o mesmo teste que o anterior, mas com o Wi-Fi ON. As curvas permanecem parecidas, entretanto, o ruído gerado pelo Wi-Fi é bem notável, sendo até desaconselhável usar o ADC com Wi-Fi ON em produtos que precise de leituras estáveis.

Algumas medidas para tentar melhorar a leitura é usar um atenuador externo como divisor resistivo em 0 dB e adicionar filtros analógicos/digitais.

API de calibração

A Espressif, após tantas reclamações dos usuários, começou a disponibilizar uma biblioteca (API) para calibração do ADC com tabelas para comparação que também utilizam dois métodos para afinar a leitura entre os chips, que



vêm gravados na memória (eFuse) dos ESP32 mais recentes fabricados de 2018 em diante:



naquele ESP32. Quando disponível, é utilizado pela API do nível do padrão (veremos mais a seguir);

- **Two Point:** A calibração de dois pontos tem as leituras do ADC1 e ADC2 para 150 mV e 850 mV naquele ESP32. Se disponível, este método tem mais prioridade que o Vref quando utilizado pela API.

A API do ADC faz todo o trabalho de compensar as curvas de acordo com estes e outros valores gravados na memória. Mais detalhes nas referências no fim do artigo. Se ainda não for o suficiente, você pode efetuar suas próprias análises e calibrações individuais das placas com alguma fórmula de correção, como a Espressif fez em sua linha de montagem, porém, de forma automatizada.

E se o meu ESP32 for anterior a 2018, vou poder utilizar essa calibração? Sim, é o que vamos testar agora, alterando as tensões de referência para os máximos, padrão e verdadeiro.

Teste E

Nesse teste (figuras 11 e 12), vamos analisar o ADC1 (GPIO36) e seu erro com 0 dB para as seguintes opções:

- WiFi **OFF**;
- 12 bits;
- 0 dB (0 - 1100 mV);
- Vref: 1000, 1100, 1200, 1072 mV.





Figura 11 - Curvas de Vref 0dB.

Figura 12 - Erros de curvas Vref 0dB.

Observando as figuras 10 e 11, podemos perceber que há grande variação assumindo os piores casos (1000 e 1200 mV). Apenas para a curva de 1072 mV que foi utilizado o Vref gravado na memória, sendo o restante definido no código. Qualquer uma dessas curvas ainda é melhor do que se utilizar o método sem calibração como visto nas figuras 2, 3, 4 e 5.

Teste F

Nesse teste (figuras 13 e 14), vamos analisar o ADC1 (GPIO36) e seu erro com 11 dB para as seguintes opções:

- WiFi **OFF**;
- 12 bits;
- 11 dB (0 - 3900 mV);
- Vref: 1000, 1100, 1200, 1072 mV.



ESP32 ADC Test: 0 - 3300mV, 11dB

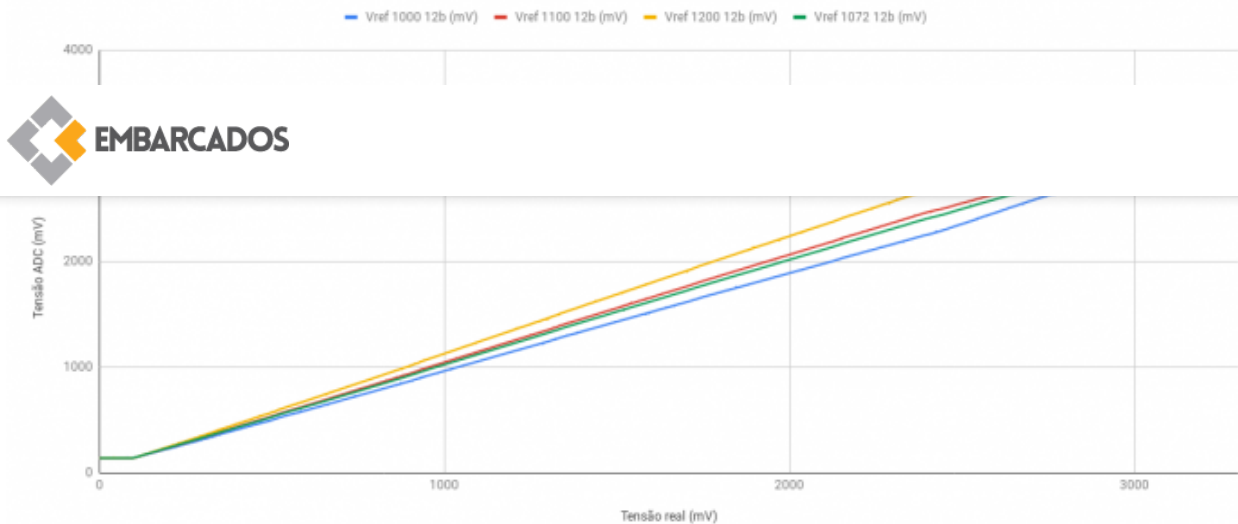


Figura 13 - Curvas de Vref 11dB.

ESP32 ADC Erro: 0 - 3300mV, 11dB

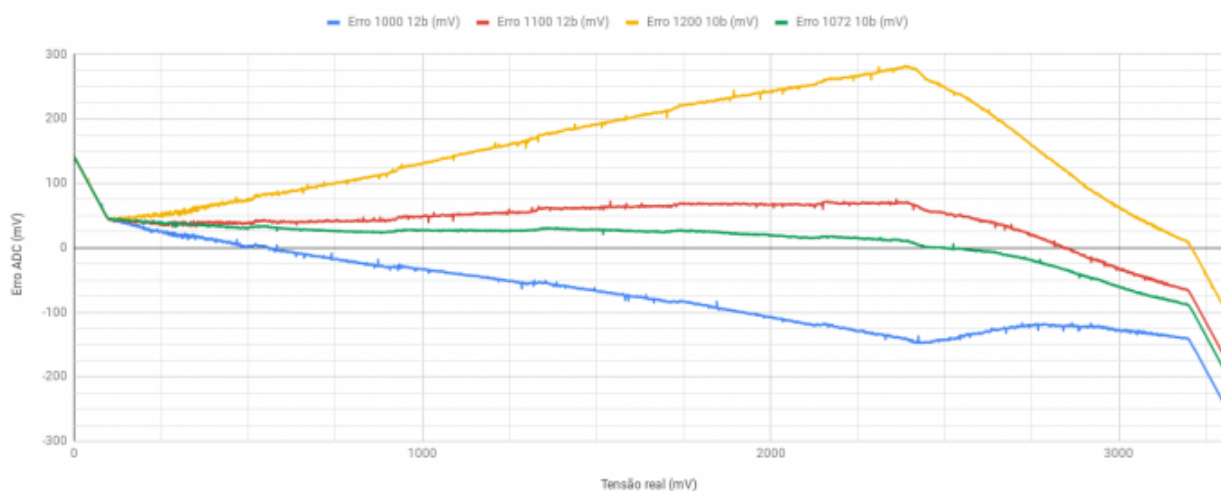


Figura 14 - Erro de curvas Vref 11dB.

Observando as figuras 13 e 14, os erros com Vref 1200 mV chegam até próximo aos 300 mV, sendo totalmente descartável em muitos produtos. Mesmo assumindo o Vref verdadeiro, não foi capaz do erro tender a ~0 mV como em 0 dB. Novamente, qualquer uma dessas curvas será muito melhor do que sem a API de calibração, como visto nas figuras 7, 8, 9 e 10.

Vamos então colocar a mão na massa e utilizar essa API de calibração, que faz todo o trabalho sujo para nós. O código é simples, bastando apenas inicializar a estrutura interna para calibração e depois converter o valor RAW

lido para mV com a função específica da API. A API utilizará o melhor método para afinar a leitura sempre que disponível.



Código

```
1  #include <driver/adc.h>
2  #include <esp_adc_cal.h>
3  #include <freertos/FreeRTOS.h>
4  #include <freertos/task.h>
5  #include <esp_err.h>
6  #include <esp_log.h>
7
8
9  esp_adc_cal_characteristics_t adc_cal; //Estrutura que contem as informacoes para calibrar
10
11
12 void app_main()
13 {
14     adc1_config_width(ADC_WIDTH_BIT_12); //Configura a resolucao
15     adc1_config_channel_atten(ADC1_CHANNEL_0, ADC_ATTEN_DB_11); //Configura a atenuacao
16
17
18     esp_adc_cal_value_t adc_type = esp_adc_cal_characterize(ADC_UNIT_1, ADC_ATTEN_DB_11, ADC_WIDTH_BIT_12, &adc_cal);
19
20     if (adc_type == ESP_ADC_CAL_VAL_EFUSE_VREF)
21     {
22         ESP_LOGI("ADC CAL", "Vref eFuse encontrado: %umV", adc_cal.vref);
23     }
24     else if (adc_type == ESP_ADC_CAL_VAL_EFUSE_TP)
25     {
26         ESP_LOGI("ADC CAL", "Two Point eFuse encontrado");
27     }
28     else
29     {
30         ESP_LOGW("ADC CAL", "Nada encontrado, utilizando Vref padrao: %umV", adc_cal.vref);
31     }
32
33
34
35     while (1)
36     {
37         /*
38          * Obtem a leitura RAW do ADC para depois ser utilizada pela API de calibracao
39          * Media simples de 100 leituras intervaladas com 30us
40          */
41         uint32_t voltage = 0;
42         for (int i = 0; i < 100; i++)
43         {
44             voltage += adc1_get_raw(ADC1_CHANNEL_0); //Obtem o valor RAW do ADC
45             ets_delay_us(30);
46         }
47         voltage /= 100;
48
49
50
51         voltage = esp_adc_cal_raw_to_voltage(voltage, &adc_cal); //Converte e calibra para mV
52         ESP_LOGI("ADC CAL", "Read mV: %u", voltage); //Mostra a leitura calibrada no terminal
53
54
55
56         vTaskDelay(pdMS_TO_TICKS(1000)); //Delay 1seg
57     }
58 }
```

Testando este código, você também vai descobrir se seu ESP32 tem os valores de Vref ou Two Point gravados na memória, além de verificar a tensão



Atenção, é necessário que as opções de Vref e/ou Two Point estejam habilitadas no MENUCONFIG para API conseguir utilizar, o que vem por padrão ON, mas pode ser necessário você olhar manualmente se está ativado. Mais informações nas referências.

Conclusão

Após vários testes com o ADC no ESP32, podemos chegar à conclusão que ele realmente não é tão bom para alguns produtos, mas a calibração da Espressif é muito útil e serve para muitos outros, ainda mais quando 0 dB. Se você precisa de algo ainda mais sofisticado, pode utilizar filtros digitais e/ou analógicos para deixar um sinal mais agradável, além de aplicar alguma fórmula matemática para correção da curva do ADC (Excel pode fazer isso!).

Caso seu produto demande um ADC melhor, seria aconselhável utilização de um ADC externo, que é dedicado a esta função e **costuma** ser muito melhor que ADC embutidos de microcontroladores. Continue estudando maneiras para calibração de ADC's e diga para gente, nos comentários, o método utilizado!

Saiba mais

[Conhecendo o co-processador ULP \(Ultra Low Power\) do ESP32](#)

[Configurando o ambiente de desenvolvimento do ESP32 no Windows](#)

[Controlando ESP32 via WiFi com validação por endereço MAC](#)

Referências



<https://docs.espressif.com/projects/esp-idf/en/latest/api-reference/peripherals/adc.html#>



[reference/peripherals/adc.html#adc-calibration](https://docs.espressif.com/projects/esp-idf/en/latest/api-reference/peripherals/adc.html#adc-calibration)

https://www.espressif.com/sites/default/files/documentation/esp32_technical_refe



Esta obra está licenciada com uma Licença [Creative Commons Atribuição-Compartilhual 4.0 Internacional](#) [↗](#).

Receba os melhores conteúdos sobre sistemas eletrônicos embarcados, dicas, tutoriais e promoções.

Seu e-mail

CADASTRAR E-MAIL

[Software](#) » ESP32 - Analisando e corrigindo o ADC interno



[Software](#) [Firmware, Intermediário](#)

COMENTÁRIOS:





B I U [+]



13 COMENTÁRIOS



recentes ▾

**Fernando Meira Rocha**

🕒 16/09/2020 12:39

E se eu estou com um sinal de temperatura (via DS18B20) ou umidade/temperatura (DHT22) esta variação na ADC faz variar a minha calibração? ou melhor ainda, eu posso calibrar minhas entradas de temperatura, umidade e pressão pelo conversor ADC?
Agradeço um norte...



0



Responder

**urbanze** Reply to [Fernando Meira Rocha](#)

🕒 16/09/2020 18:42

Esses sensores são digitais, não usam o ADC do ESP32 para leitura. Então não consegue usar essa calibração.



1



Responder

**Fernando Meira Rocha** Reply to [urbanze](#)

🕒 17/09/2020 08:29

Obrigado pela ajuda Urbanze. Vou tentar outras maneiras de calibração, pois alguns não vem dentro dos parâmetros do datasheet de fábrica. Achei que pudesse ser uma variação da tensão de alimentação.



0



Responder



**Odair Carlos**

🕒 16/06/2020 08:27



relativamente fácil de encontrar na lib do ESP32 o .h que contém as funções e incluir no projeto... espero que outros usuários arduino também encontrem com facilidade! Grande abraço e bom trabalho!



2



Responder

**Eliel**

🕒 11/08/2019 04:11

"Ambos controladores podem ser usados no domínio digital e RTC, que são focados em velocidade (2 M Samples/Seg) e baixo consumo (200 K Samples/Seg), respectivamente;"
Gostaria de saber como configuro o Esp32 para atingir essas taxas, pois o máximo que consegui foram míseros 26ksps.
Alguém sabe?



0



Responder

José Moraes💬 Reply to [Eliel](#)

🕒 06/06/2020 14:44

O periférico I2S permite ler o ADC através de DMA. Com isso, pode até ultrapassar 2MHz de leitura.



0



Responder

**Antonio**

🕒 31/05/2019 04:16

Os pinos do ADC2, posso utilizar para outros fins, como saída pra rele, mesmo quando o WIFI está ligado? o Wifi utiliza todos os 10 pinos do ADC2?




0



Responder



urbanze

 Reply to [Antonio](#) 01/06/2019 11:43

(digital, PWM, etc) mesmo com WiFi.

 0   Responder**JEAN CARLOS RIBEIRO** 29/04/2019 10:05

Bom dia
se for mais de um ADC,
muda alguma coisa nessa API ??
`esp_adc_cal_value_t adc_type = esp_adc_cal_characterize(ADC_UNIT_1,
ADC_ATTEN_DB_11, ADC_WIDTH_BIT_12, 1100, &adc_cal);`//Inicializa a
estrutura de calibracao

 0   Responder**José Morais** Reply to [JEAN CARLOS RIBEIRO](#) 29/04/2019 17:56

Não, a única coisa que mudará é na linha 45
"`adc1_get_raw(ADC1_CHANNEL_0);`" que será outro canal que você
deseja ler. A calibração usada permanece a mesma para todos pinos
do ADC1...

 0   Responder**JEAN CARLOS RIBEIRO** Reply to [José Morais](#) 30/04/2019 11:00

Blz valeuu

 0   Responder

Rudsom



🕒 26/04/2019 08:53

Excelente artigo! Obrigado!



0



Responder



Francisco

🕒 25/03/2019 16:22

Gostaria de agradecer por todo esse empenho em nos manter informados sobre as fragilidades de determinados componentes. Principalmente do ESP32 que está recebendo tanto interêsse da comunidade IoT.






1



Responder

TALVEZ VOCÊ GOSTE:



 07/11/2014  Diego Sueiro  Beaglebone Black, Linux Embarcado, Software



Comunicação Serial Java + Arduino

 14/02/2014  Klauder Dias  Arduino, Comunicações, Software

Beaglebone Black + Qt5 + Yocto – Parte 2

 10/11/2014  Diego Sueiro  Beaglebone Black, Linux Embarcado, Software



SÉRIES



NEWSLETTER

Receba os melhores conteúdos sobre sistemas eletrônicos embarcados, dicas, tutoriais e promoções.

CADASTRAR E-MAIL

INSTITUCIONAL

- > O Embarcados
- > Seja Colaborador
- > Contato

COMUNIDADE

- > Oportunidades
- > Lojas Online
- > Sites e Blogs

NAS REDES





Desenvolvido por

Este site utiliza cookies. Ao usá-lo você concorda com nossos Termos de Uso. [Saiba mais.](#)

Continuar

