



FAVORITAR

27/06/2014 Fábio Souza 30 Comentários

Arduino UNO – Taxa de amostragem do conversor A/D



-- publicidade --



-- publicidade --



ÍNDICE DE CONTEÚDO



- ▶ Conversor A/D do ATmega328
- ▶ Modos de operação
- ▶ Conversão simples
- ▶ Conversão contínua
- ▶ Clock



- ▶ Resolução
- ▶ Leitura padrão do A/D no Arduino
- ▶ Conversão com clock de 250 KHz
- ▶ Aumentando o clock para 500 KHz
- ▶ O que acontece a 1 MHz
- ▶ Conclusão
- ▶ Saiba mais
- ▶ Referências

Este post faz parte da série [Arduino UNO: Conversor A/D](#). Leia também os outros posts da série:

- [Arduino UNO – Taxa de amostragem do conversor A/D](#)
- [Arduino UNO – Sensor de temperatura interno](#)

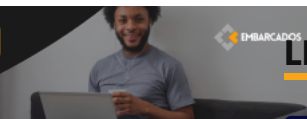
A plataforma Arduino traz em seu núcleo funções para leitura de sinais analógicos através da utilização do conversor analógico digital. O valor da taxa de amostragem é configurado internamente através das bibliotecas do Arduino. Desta forma não precisamos configurar os registradores internos do [ATmega328](#) antes da leitura de um sinal analógico. Essa camada de abstração auxilia os iniciantes para a leitura de sinais analógicos, porém em aplicações que necessitam de uma otimização da leitura, é necessário entender o funcionamento do conversor A/D do microcontrolador ATmega328 e a correta configuração dos seus registradores para que atenda as necessidades do projeto.

Nesse artigo vamos abordar o funcionamento do conversor A/D do [Arduino UNO](#) que é baseado no ATmega328, entendendo a sua configuração e desenvolver alguns teste para avaliarmos qual a máxima taxa de amostragem que podemos obter garantindo a resolução máxima do conversor.

-- publicidade --

Treinamentos sobre
Sistemas Embarcados e IoT


Conheça aqui




Conversor A/D do ATmega328

O ATmega328 possui internamente um conversor A/D de aproximação sucessivas de 10 bits de resolução com precisão de ± 2 LSBs. Possui até 8 canais de entradas multiplexados, dependendo do encapsulamento. No caso do ATmega328 PDIP, do Arduino UNO, apresenta apenas 6 canais, como pode-se verificar na placa. Por existir apenas 1 conversor A/D, só poderá ser selecionado 1 canal por vez para conversão, isso é feito através da configuração dos registradores internos. O diagrama de blocos do conversor A/D é exibido a seguir:

WEBINARS



Webinar: Como uma solução IoT baseada em Android Automotive pode ajudar a reduzir os acidentes de trânsito no Brasil?



NOVIDADES DO ZEPHYR RTOS

Webinar: Novidades do Zephyr RTOS

-- publicidade --

LATeRe

30 anos de experiência em Componentes Eletrônicos no Brasil.

Representante Exclusivo

Newark
AN AVNET COMPANY

CLIQUE AQUI

LEIA TAMBÉM

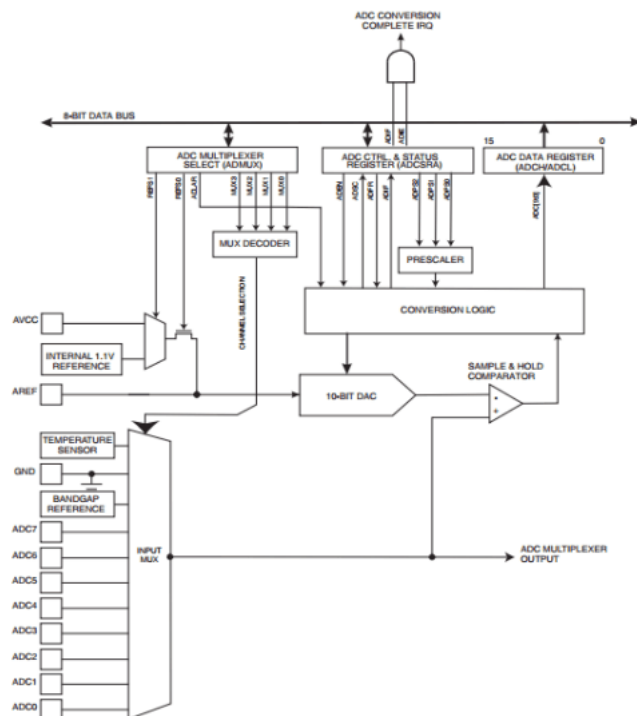
NEW

AWS Trainium2

Amazon anuncia novo chip para cloud computing



A bola da Eurocopa 2024, uma maravilha tecnológica



ESP32: Interfaceamento com microfones digitais

Como pode ser observado na figura acima, o bloco do conversor A/D possui fonte separada para a parte analógica, o pino AVcc. Essa tensão não pode variar mais do que $\pm 0,3V$ de Vcc.

O Atmega328 possui tensão de referência interna de 1,1 V, que pode ser selecionada por software. Apresenta também um pino externo para uma tensão de referência diferente de VCC ou a referência interna de 1,1 V. O valor de tensão de entrada deve estar entre 0V e o valor de tensão de referência, não ultrapassando o valor de VCC.

Ao final da conversão pode ser gerada uma interrupção, caso a mesma esteja habilitada.

A conversão gera um resultado de 10 bits, necessitando assim de 2 registradores, **ADCH** e **ADCL**.

A seguir serão apresentados os registradores de configuração do conversor A/D do ATmega328:

ADMUX – ADC Multiplexer selection Register:

Bit	7	6	5	4	3	2	1	0	
(0x7C)	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit 7:6 – REFS1:0 – Reference Selection Bits

Esses bits configuram a fonte de tensão de referência para o A/D, conforme a tabela abaixo:

Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal V_{ref} turned off
0	1	AV_{CC} with external capacitor at AREF pin

O EMBARCADOS ▾

CONTEÚDOS ▾

EM DESTAQUE ▾

OPORTUNIDADES ▾

PERFIL

LOG IN

**Bit 5 – ADLAR: ADC left adjust Result**

Configura a forma de exibição do resultado da conversão. **ADLAR** = 1, resultado justificado a esquerda, **ADLAR** = 0, justificado a direita. O resultado é exibido nos registradores **ADCL** e **ADCH**, conforme a configuração do **ADLAR**.

Bit4 – Não usado**Bits 3:0 – MUX3:0 – Analog Channel Selection Bits**

Seleciona qual entrada analógica será conectada ao conversor, conforme tabela abaixo:

Input Channel Selections

MUX3..0	Single Ended Input
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5
0110	ADC6
0111	ADC7
1000	ADC8 ⁽¹⁾
1001	(reserved)
1010	(reserved)
1011	(reserved)
1100	(reserved)
1101	(reserved)
1110	1.1V (V_{BG})
1111	0V (GND)

nota: 1. Sensor de temperatura

ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit 7 – ADEN: ADC Enable

Habilita o conversor A/D quando em nível lógico 1. Quando **ADEN** = 0 o conversor será desligado e caso isso ocorra enquanto uma conversão em progresso, a mesma será terminada antes de desligar o conversor A/D.

Bit 6 – ADSC: ADC Start conversion

No modo de conversão simples, **ADCS** = 1 fará iniciar a conversão, já no modo de conversão contínua será iniciada a primeira conversão. **ADCS** vai para nível lógico zero quando a conversão é finalizada. Se **ADCS** for escrito em nível lógico 1 ao mesmo tempo que **ADEN**, a primeira conversão levará 25 ciclos de clock ao invés dos 13 Ciclos de uma conversão.

Bit 5 – ADATE: ADC Auto Trigger Enable

Habilita o auto dispara, quando esse bit estiver em 1. O conversor iniciará uma conversão quando uma borda de subida ocorrer no sinal de disparo. O sinal de disparo é selecionado nos bits **ADTS** do registrador **ADCSR**.

Bit 4 – ADIF: ADC Interrupt Flag

Sinaliza o final de uma conversão e os registradores de dados são atualizados.

Bit 3 – ADIE: ADC Interrupt Enable

Habilita a interrupção no final da conversão. Porém os bits do registrador **SREG** deve estar ligado, para que ocorra a interrupção.

Bit 2:0 – ADPS2:0: ADC Prescaler Select Bits

Configura o fator de divisão entre o clock do sistema e a entrada de clock do ADC. Os valores possíveis são exibidos na tabela abaixo:

ADC Prescaler Selections

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

ADCSR – ADC Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
(0x7B)	–	ADSC	–	–	–	ADTS2	ADTS1	ADTS0	ADCSR
Read/Write	R	R/W	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit 2:0 – ADTS2:0: ADC Auto Trigger Source

Seleciona a fonte de disparo caso o bit **ADATE** esteja habilitado. As fontes possíveis são exibidas na tabela a seguir:

ADC Auto Trigger Source Selections

ADTS2	ADTS1	ADTS0	Trigger Source
0	0	0	Free Running mode
0	0	1	Analog Comparator
0	1	0	External Interrupt Request 0
0	1	1	Timer/Counter0 Compare Match A
1	0	0	Timer/Counter0 Overflow
1	0	1	Timer/Counter1 Compare Match B
1	1	0	Timer/Counter1 Overflow
1	1	1	Timer/Counter1 Capture Event

Vamos ver a seguir os modos de operação do conversor A/D do ATmega328.

Modos de operação

O conversor AD do Atmega328 possui dois modos de operação: conversão simples e conversão contínua.

Conversão simples

No modo de conversão simples é necessário a inicialização de cada conversão. Quando a conversão é finalizada os registradores de dados são preenchidos e o bit **ADIF** é colocado em 1. Para iniciar uma conversão deve-se ligar o bit **ADSC**. Esse bit permanecerá em 1 enquanto a conversão está em processo, e passará para 0 no final da conversão.

Conversão contínua

No modo de conversão contínua, você iniciará a primeira conversão e o conversor iniciará automaticamente as próximas conversões, logo após ser completada a anterior.

Clock

O clock recomendado para o conversor AD do Atmega328 é de 50KHz a 200 KHz para uma resolução de 10 bits. O bloco prescaler controla do clock do conversor A/D, assim o clock do conversor A/D será uma fração do clock do oscilador principal, conforme o fator do prescaler.

Os valores são selecionados no registrador **ADCSA** nos bits **ADPS2:0**. No caso da placa **Arduino UNO** que roda como um cristal de 16 MHz, o clock do conversor A/D pode assumir os seguintes valores:

- $16 \text{ MHz} / 2 = 8 \text{ MHz}$
- $16 \text{ MHz} / 4 = 4 \text{ MHz}$
- $16 \text{ MHz} / 8 = 2 \text{ MHz}$
- $16 \text{ MHz} / 16 = 1 \text{ MHz}$
- $16 \text{ MHz} / 32 = 500 \text{ kHz}$
- $16 \text{ MHz} / 64 = 250 \text{ kHz}$
- $16 \text{ MHz} / 128 = 125 \text{ kHz}$

Como mencionado anteriormente o clock do conversor A/D deve estar entre 50 KHz e 200 KHz para garantir a precisão de 10 bits na resolução. Assim, observando os valores anteriores só se pode usar o prescaler de 128. Caso esteja trabalhando com um cristal de 20 MHz, e for selecionado o prescaler de 128 o clock do conversor AD será 156 KHz.

Uma conversão normal necessita de 13 pulsos de clock no conversor A/D. A primeira conversão necessita de 25 pulsos de clock, conforme exibido nas figuras abaixo. Dessa forma o valor de amostragem do conversor A/D depende do pulsos de clock de cada conversão, ou seja, o valor do clock deve ser dividido por 13 para calcular a quantidade de amostras por segundo.

Primeira conversão:

Conversão Normal:

A configuração do conversor A/D do Arduino está no arquivo **wiring.c**, e encontra-se da seguinte forma:

```
1 #if defined(ADCSRA)
2   // set a2d prescaled factor to 128
3   // 16 MHz / 128 = 125 KHz, inside the desired 50-200 KHz range.
```

```
4 // XXX: this will not work properly for other clock speeds, and
5 // this code should use F_CPU to determine the prescaled factor.
6 sbi(ADCSRA, ADPS2);
7 sbi(ADCSRA, ADPS1);
8 sbi(ADCSRA, ADPS0);
9
10 // enable a2d conversions
11 sbi(ADCSRA, ADEN);
12 #endif
```

Conforme exibido na configuração acima, o prescaler com 128, provendo um clock de 125 KHz para o ADC, já que o Arduino roda com um cristal de 16MHz.

Com um clock de 125 KHz a taxa de amostragem será: $125 \text{ KHz} / 13 = 9600$ amostras por segundo.

Uma opção para o aumento da taxa de amostragem é a troca oscilador principal para uma frequência de 12 MHz, onde é possível chegar ao valor de 187 KHz de clock, que resultará em uma taxa de amostragem de $187 \text{ KHz} / 13 = 14384$ amostras por segundo. Caso se tenha um clock de 200 KHz, que é o máximo recomendado, a taxa de amostragem máxima que será conseguida com o conversor A/D do ATmega328 será: $200 \text{ KHz} / 13 = 15384$ amostras por segundo, ou seja, o AD do ATmega328 conseguirá no máximo 15KHz de amostragem com 10 bits de resolução.

Resolução

O conversor A/D do ATmega328 possui 10 bits de resolução, ou seja, os valores entre 0 e **Vref** serão convertidos entre 0 e 1023.

O clock máximo recomendado para essa resolução é 200 KHz, que dá uma taxa de amostragem de aproximadamente 15KHz. No [application Note AVR120: Characterization and Calibration of the ADC on an AVR](#), encontramos a seguinte declaração:

"The ADC accuracy also depends on the ADC clock. The recommended maximum ADC clock frequency is limited by the internal DAC in the conversion circuitry. For optimum performance, the ADC clock should not exceed 200 kHz. However, frequencies up to 1 MHz do not reduce the ADC resolution significantly. Operating the ADC with frequencies greater than 1 MHz is not characterized."

Isso significa que pode-se aumentar o clock acima de 200 KHz até 1MHz sem obter degradação na precisão do valor convertido. Para verificar esta afirmação, vamos testar a aquisição para alguns valores de prescaler, ou seja, aumentando o clock do conversor A/D.

Leitura padrão do A/D no

Arduino

Para testarmos a taxa de amostragem que vem configurada por padrão no Arduino vamos utilizar o código a seguir, que consiste em os valores de conversão marcando o tempo de início e o tempo de fim da leitura, exibindo o valor e o tempo decorrido para cada leitura.

```
1 // Variáveis para armazenar os resultados
2 unsigned long tempo_inicio;
3 unsigned long tempo_fim;
4 unsigned long valor;
5
6 void setup() {
7   Serial.begin(19200); //inicia a comunicação serial
8 }
9
10 void loop() {
11   // leitura
12   tempo_inicio = micros(); //marca tempo de início de leitura
13   valor = analogRead(0); //le valor convertido
14   tempo_fim = micros(); //le tempo no fim da conversão
15
16   //exibe valor lido e tempo de conversão
17   Serial.print("Valor = ");
18   Serial.print(valor);
19   Serial.print(" -- Tempo leitura = ");
20   Serial.print(tempo_fim - tempo_inicio);
21   Serial.println(" us");
22   delay(500);
23 }
```

Conforme exibido acima, o tempo decorrido em cada leitura está entre 108 us e 116 us. Se pegarmos o valor de 0,116 ms teremos uma frequência de amostragem de aproximadamente 8600 Hz, bem próximo de 9600 calculado anteriormente.

Conversão com clock de 250 KHz

Agora vamos mudar o prescaler do clock para 64, aumentando a frequência do clock para 250 KHz, já que estamos usando um cristal de 16MHz. Esse clock já está acima do valor recomendado. Para facilitar a configuração foram criadas constantes para facilitar a correta escrita no registrador **ADCSRA**, conforme é exibido no código abaixo:

```
1 // Variável para armazenar os resultados
2
```

```
3 unsigned long tempo_inicio;
4
5 unsigned long tempo_fim;
6
7 unsigned long valor;
8
9 // constante para configuração do prescaler
10 const unsigned char PS_16 = (1 << ADPS2);
11 const unsigned char PS_32 = (1 << ADPS2) | (1 << ADPS0);
12 const unsigned char PS_64 = (1 << ADPS2) | (1 << ADPS1);
13 const unsigned char PS_128 = (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);
14
15 void setup() {
16   Serial.begin(9600);
17
18   // configura o preescaler do ADC
19   ADCSRA &= ~PS_128; //limpa configuração da biblioteca do arduino
20
21   // valores possíveis de prescaler só deixar a linha com prescaler desejado
22   // PS_16, PS_32, PS_64 or PS_128
23   //ADCSRA |= PS_128; // 64 prescaler
24   ADCSRA |= PS_64; // 64 prescaler
25   // ADCSRA |= PS_32; // 32 prescaler
26   // ADCSRA |= PS_16; // 16 prescaler
27 }
28
29 void loop() {
30   // leitura
31   tempo_inicio = micros(); //marca tempo de inicio de leitura
32   valor = analogRead(0); //le valor convertido
33   tempo_fim = micros(); //le tempo no fim da conversão
34
35   //exibe valor lido e tempo de conversão
36   Serial.print("Valor = ");
37   Serial.print(valor);
38   Serial.print(" -- Tempo leitura = ");
39   Serial.print(tempo_fim - tempo_inicio);
40   Serial.println(" us");
41   delay(500);
42 }
```

Agora o intervalo de leitura está na ordem de 60us, o que nos dá uma frequência de aproximadamente 16KHz. O que já era esperado já que foi dobrado a frequência do clock.

Variando o valor do pontenciamento de 0 a 100%, nota-se que o valor da conversão está entre 0 a 1023, desta forma a resolução para esse clock ainda encontra-se em 10 bits.

Aumentando o clock para 500 KHz

Agora vamos mudar o prescaler para 32, dessa forma o ADC estará rodando com uma frequência de 500KHz.

O intervalo de leitura agora caiu para 32 us o que dá uma taxa de amostragem de aproximadamente 31K amostra por segundo. Variando o valor do potenciômetro verifica-se que a conversão ainda está sendo feita corretamente.

O que acontece a 1 MHz

Por ultimo vamos configurar o prescaler para 16 assim o ADC estará funcionando a 1MHz.

O intervalo de leitura agora está na faixa de 20 us o que dá uma taxa de amostragem de aproximadamente 50K amostra por segundo. Variando potenciômetro verifica-se que o valor de conversão ainda se encontra com 10 bits de resolução.

Conclusão

O correto uso do conversor A/D é essencial para a amostragem de sinais analógicos. Conhecer as configurações e os limites de hardware possibilita a otimização e confiabilidade do sinal lido e desta informação. Nesse artigo conhecemos os registradores do conversor AD do ATmega328 e testamos a sua taxa de amostragem. Percebeu-se que não houve prejuízo nos valores de conversão quando houve o aumento do clock do conversor AD, porém para diminuir possíveis erros ocasionados por ruídos na conversão (e quando não há a necessidade de uma alta taxa de amostragem) é aconselhado trabalhar em

uma frequência menor. Pode-se também aumentar a frequência de amostragem para valores acima de 1 MHz e trabalhar com uma resolução de 8 bits, configurando o ajuste à esquerda do resultado.

A partir do exemplo apresentado você poderá fazer teste de aquisições em frequências de amostragens diferentes do padrão que vem configurado na biblioteca do Arduino. Teste em seus projetos e nos conte suas experiências. No próximo artigo vamos testar o sensor de temperatura interno do Atmega328.

Saiba mais

[Arduino – Primeiros Passos](#)

[Arduino UNO](#)

[Arduino – Entradas Analógicas](#)

Referências

[Função analogRead\(\)](#) 


[Advanced Arduino ADC – Faster analogRead\(\)](#) 

<https://forum.arduino.cc/index.php/topic,6549.0.html>


[AVR120: Characterization and Calibration of the ADC on an AVR](#) 

Outros artigos da série

[Arduino UNO – Sensor de temperatura interno >>](#)

Fábio Souza**385 posts**<https://github.com/FBSelectronica> 

Fabio Souza é um engenheiro com ampla experiência no desenvolvimento de projetos eletrônicos. Atualmente, como diretor do portal Embarcados, dedica-se a promover a área de desenvolvimento de projetos eletrônicos, sistemas embarcados e IoT no Brasil. Com seu profundo conhecimento em eletrônica e programação, Fabio atua como professor de graduação e pós-graduação, bem como ministra cursos livres e exclusivos para empresas. Ele é um entusiasta do movimento maker, da cultura DIY e do compartilhamento de conhecimento, publicando diversos artigos, projetos open hardware e sendo autor de livros da área. Por meio de iniciativas como o projeto Franzininho e outros projetos na área de educação, Fabio leva a cultura maker para o Brasil, capacitando e incentivando professores e alunos a usarem a tecnologia em suas vidas. Sua dedicação é fundamental para impulsionar a inovação e o empreendedorismo no país.

Esta obra está licenciada com uma Licença Creative Commons Atribuição-Compartilhual 4.0 Internacional .



 Software, Hardware, Arduino # Iniciante, Eletrônica Analógica

[Home](#) » [Arduino](#) » Arduino UNO – Taxa de amostragem do conversor A/D



JUNTE-SE HOJE À COMUNIDADE EMBARCADOS**FAZER PARTE****COMENTÁRIOS:****SÉRIES**

ULWOS – Multitarefa no RL78

Controlador VGA

 Notificações ▾*Entre na discussão***B** *I* U        

30 COMENTÁRIOS

  recentes ▾

yuri

10/08/2023 11:26

Cara, poderia me tirar a seguinte duvida. para um microcontrolador blackpill, como eu calculo o prescale para uma resolução adc de 12bits?



0

 Responder

Víctor Paulo

24/01/2020 21:09

Boa noite, excelente artigo, muito explicativo, parabéns!!!

Uma dúvida:

Você mencionou acima: "Pode-se também aumentar a frequência de amostragem para valores acima de 1 MHz e trabalhar com uma resolução de 8 bits, configurando o ajuste à esquerda do resultado."

A dúvida é em como isso poderia ser feito, não estou familiarizado com a manipulação de registradores, poderia dar uma forcinha?

Agradeço desde já, Víctor.



0

 Responder

Carlos

13/06/2019 11:22

Ola Fábio, parabéns pelo excelente artigo!

Tenho uma pergunta: estou utilizando um LCD 16x2 para ler valores analógicos, tensão de 0 a 30V.

Porém devido à resolução não consigo uma indicação estável e definida pois o passo é de 4mV +/-.

Se eu utilizar como referência 1,1V conseguirei mesmo melhorar a indicação?

Grande abraço.

Carlos

cad.rj.2015@gmail.com



0

 Responder**Monitoramento de água com IoT****Trazendo o mundo real para dentro do processador****Sistemas Operacionais de Tempo Real****A arte de especificar e encontrar componentes****Projetos de desenvolvimento: Antes de começar****Boas práticas para o desenvolvimento de software****GNU ARM Cross-toolchain****Shape The World****Ver todas as séries** →

 Jose Antonio Gonzalez Gil

30/04/2019 09:09

Excelente artigo, muito bom parabéns



0



Responder

 Fábio Santos

17/11/2018 07:56



Olá Fábio, bom dia! Estou com um problema e não consigo um código que consiga satisfazer as seguintes condições: Utilizar Arduino como base (permitindo usar a serial) void setup() { } void loop() { } Problema: Desenvolver através da configuração/manipulação dos registradores (sem funções do Arduino) * Única exceção, uso da Serial nativa do Arduino "Serial.begin(9600);" no setup "Serial.println(texto/valor);" para imprimir algo no terminal Configurar o timer 1 para disparar o ADC a cada 1ms * Sinal analógico na entrada A0 * Potenciômetro (conectado ao Vcc e GND) * Realizar a média de 16 amostras do ADC * Minimizar ruídos... [Leia mais »](#)



0



Responder

 Fernando França 

09/11/2015 10:02

Bom dia Fábio. Excelente artigo, parabéns. Estou projetando um medidor inteligente para qualidade de energia como trabalho de conclusão e pensei em utilizar o ATmega 328P como μC para minha parte de aquisição de dados. Entretanto as normas do Prodist (ANEEL), estabelecem que instrumentos de medição devem considerar para fins de distorção da componente fundamental até a 25ª harmônica. Ou seja, $60 \times 25 = 1500$ Hz. Sendo então pela frequência de Nyquist 3 KHz de amostragem. Com os 10 bits de resolução dele, parece que estou dentro e seu artigo me deixou otimista quanto ao comportamento do ADC. Mas... [Leia mais »](#)



0



Responder

Carlos

 Reply to [Fernando França](#)

12/02/2016 17:22


Boa tarde, também estou pensando nesse mesmo assunto para o meu TCC. Acredito que com a frequência de amostragem não haverá problemas, mas a minha maior dúvida é quanto à resolução: medir -180 a 180Vpp transformados para 0 a 5V, com 10 bits de resolução leva à uma precisão de décimo de volt, que não sei se é aceitável, visto que a amplitude das últimas harmônicas deve ser muito pequena e provavelmente afetaria na questão dos fatores de distorção.



0



Responder

 **Matheus Tieppo**

30/10/2015 12:50

Boa Tarde Fabio, gostaria de saber se é possível ler e gravar no sd 4000 amostrar por segundo de uma senoide 60 hz.

desde já agradeço



0



Responder

Fabio_Souza_Embarcados Reply to [Matheus Tieppo](#)

30/10/2015 18:33

Com esse Arduino você ficará limitado. Talvez você consiga essa taxa de amostragem o Arduino Due ou outra arquitetura.

Abraços



0



Responder

 **Felipe Fontenele**

12/08/2015 09:19

Olá, gostaria de saber se tem como modificar os registradores de modo a eu obter um maior tempo de amostragem na casa dos milisegundos, por exemplo 3ms seria o que eu desejo.

Abraços!



0



Responder

Fabio_Souza_Embarcados Reply to [Felipe Fontenele](#)

12/08/2015 10:20

O maior tempo de amostragem será com prescaler de 128, que resultará em um tempo de conversão de aproximadamente 100 us para o cristal de 16MHz. Para o seu caso, você pode usar um timer para iniciar uma conversão a cada 3 ms e usar a configuração padrão do conversor AD no Arduino.



0



Responder

 **Fabio_Souza_Embarcados**

08/05/2015 14:20


Olá Wesley. Cara esse microcontrolador não é dedicado para amostragens de sinais nessa frequência. Além disso você precisará de uma amostragem de pelo menos 2 vezes a frequência do sinal. Não seria melhor usar um osciloscópio? Abraços





0



Responder

**Wesley Mata**
08/05/2015 09:05


Eai Fábio tudo bem?Gostaria de saber se com essa taxa de amostragem eu conseguiria medir o ruído e/ou ruído "RMS" (Spike) de saída de fontes chaveadas , onde as mesmas trabalham em uma frequência de no máximo 400KHZ.Desde já muito obrigado.

 0  Responder



Carregar mais comentários

TALVEZ VOCÊ GOSTE:



O que é firmware?

 10/01/2023  Fábio Souza

Algoritmos: Resolução de Exercícios Parte 3

 03/03/2021
 Elaine Cecília Gatto

Algoritmos: Resolução dos Exercícios Parte 1

 01/03/2021
 Elaine Cecília Gatto

NEWSLETTER

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

☐ Concordo com o [Termo de Uso](#) e [Política de Privacidade](#) do Embarcados *

CADASTRAR E-MAIL

INSTITUCIONAL

O Embarcados

Seja Colaborador

Contato

NAS REDES



COMUNIDADE

Oportunidades

Sites e Blogs

LEGAL

Legal

[Política de Privacidade](#)

[Política de Governança](#)

[Política de Cookies](#)

[Termos de Uso](#)

© Embarcados – Todos os direitos reservados. [Termos de Uso](#).

Desenvolvido por