Arduino Standalone: I

Abrantes Araújo Silva Filho

2025-11-18

Resumo

Este é o primeiro de uma série de artigos que discute e mostra, em detalhes, como montar um Arduino Standalone, ou seja, um Arduino montado apenas em uma protoboard. Neste artigo aprenderemos a fazer a montagem da fonte de alimentação, do microcontrolador e de outros componentes fundamentais, como o cristal oscilador. Em outros artigos desta série discutiremos as opções disponíveis para a programação do microcontrolador diretamente na protoboard.

Sumário

1 Introdução

Depois de algum tempo utilizando o Arduino e ganhando experiência em diversos projetos interessantes com sensores e atuadores, você começa a sentir necessidade de entender realmente como essa placa funciona e como o microcontrolador realiza a mágica de transformar o código de seu programa em ações no mundo real.

Montar um "Arduino Standalone" (usar apenas o microcontrolador — geralmente o ATmega328P — na protoboard sem a placa do Arduino) é um verdadeiro **rito de passagem** no aprendizado de eletrônica, programação e sistemas embarcados: é o momento em que você deixa de ser apenas um usuário do Arduino para entender como a engenharia por trás dele funciona.

Pense no Arduino como uma bicicleta com rodinhas de treinamento: você consegue fazer muitas coisas legais e interessantes mas chega um momento em que, para avançar, as rodinhas de treinamento precisam ser removidas. Montar seu primeiro Arduino Standalone equivale à seu primeiro passeio de bicicleta sem as rodinhas de treinamento.

Existem diversas razões pelas quais isso é interessante e importante:

- **Desmistificação do hardware**: quando usamos o Arduino, muitas coisas estão pré-configuradas e ocultas para nós. Ao montar o circuito independente na protoboard você aprenderá na prática a função de diversos componentes essenciais como o cristal oscilador, a regulação de tensão para a alimentação do microcontrolador e a montagem do circuito de reset. Além disso você entenderá de verdade o que o microcontrolador precisa obrigatoriamente para funcionar (alimentação estável, clock, reset em nível definido, capacitores de desacoplamento, etc.) e o que é opcional e fornecido pelo Arduino apenas como um "conforto" para o usuário (conversor USB-Serial, LEDs, etc.).
- **Eficiência energética**: a placa do Arduino possui componentes que constantemente consomem energia (LEDs, conversor USB-Serial) mesmo que seu código não esteja fazendo nada. Ao eliminar todos os componentes não essenciais e configurar corretamente o microcontrolador (usando o modo de *sleep*) você pode fazer seu Arduino Standalone funcionar por vários meses usando apenas duas pilhas AA (um Arduino completo acabaria com as pilhas em alguns dias).
- **Custo e permanência**: imagine que você criou um pequeno sistema de automação para o portão da garagem de sua casa. Você deixaria seu Arduino preso lá no portão para sempre? Correndo o risco de um fio se desconectar com o tempo? Você vai perder uma placa relativamente cara e que tem diversas outras possibilidades de usos além do portão? Provavelmente não. O melhor é que você utilize apenas o microcontrolador e os componentes básicos para fazer seu sistema de automação funcionar e, ao montar o Arduino Standalone, você aprenderá a como fazer isso.

- **Transição para o produto final**: nenhum produto final acabado tem um Arduino colado dentro dele. Produtos reais utilizam apenas o microcontrolador e os demais componentes eletrônicos soldados em uma placa de circuito impressa (PCB[1]) fabricada profissionalmente. Aprender a montar um Arduino Standalone é um passo intermediário obrigatório para que, no futuro, você aprenda a projetar suas próprias PCB profissionais. Você entenderá na prática a diferença entre uma placa de desenvolvimento e a versão de produção final.
- **Flexibilidade**: ao usar o microcontrolador independente você não fica preso aos $5{,}0$ V e 16 MHz do Arduino: você pode rodar o chip em $3{,}3$ V e se comunicar com sensores modernos sem conversores de nível, ou pode usar o clock interno de 8 MHz do microcontrolador e liberar mais dois pinos digitais extras para uso.
- **Entendimento do datasheet**: para entender certas funções mais avançadas do microcontrolador você será forçado a ler o datasheet e procurar coisas como os tipos de clock possíveis, configuração de fuses e a organização dos pinos.
- **Liberdade de forma e conectores**: ao usar o Arduino você é obrigado a utilizar os conectores pin header fêmea que, apesar de bons para prototipagem, são ruins para o produto final (não oferecem fixação segura dos fios jumper). Com o microcontrolador independente você é quem decide qual o tamanho e o formato da placa final, e quais serão os conectores mais adequados para seu produto (bornes de parafuso, conectores Molex e outros).
- **Exercício de engenharia**: montar um Arduino Standalone lhe proporcionará um exercício de engenharia de sistemas embarcados completo pois você precisará montar e/ou entender a parte elétrica (fonte, reguladores, capacitores, proteções), a parte digital (clock, reset, sinais de entrada/saída, interface com sensores e atuadores), o firmware (fuses, bootloader, programação ISP[2], debug), documentação (projeto esquemática, projeto da PCB) e fabricação do seu produto final.

Em resumo, ao montar seu próprio Arduino Standalone você estará dando seus primeiros passos para se tornar um verdadeiro engenheiro de sistemas embarcados. Obviamente esse é um objetivo ambicioso demais para ser tratado em uma pequena série de artigos como esta, então resolvi me limitar aqui aos aspectos básicos e mais simples de criar um Arduino Standalone na esperança de que munido desta base você seja capaz de aprender cada vez mais sobre sistemas embarcados e o fantástico mundo dos microcontroladores.

Neste primeiro artigo faremos a montagem básica do Arduino Standalone, ou seja, a montagem inicial do microcontrolador na PCB, incluindo a fonte de alimen-

[1] *Printed Circuit Board* (PCB).
[2] *In-System Programming* (ISP), também chamado por *In-Circuit Serial Programming* (ICSP).

tação, o circuito de reset, e o circuito de clock externo. Meu objetivo é que este artigo seja um tutorial completo e detalhado o suficiente para você montar seu microcontrolador independente de forma segura[3]. Alguns pressupostos que estou assumindo:

- Montaremos um Arduino Standalone usando o microcontrolador Microchip ATmega328P-PU, que é o utilizado no Arduino Uno.
- Utilizaremos como fonte de alimentação uma bateria ou fonte de 9 V, conectada diretamente à protoboard.
- Nossa montagem será a mais fiel possível ao Arduino Uno[4].

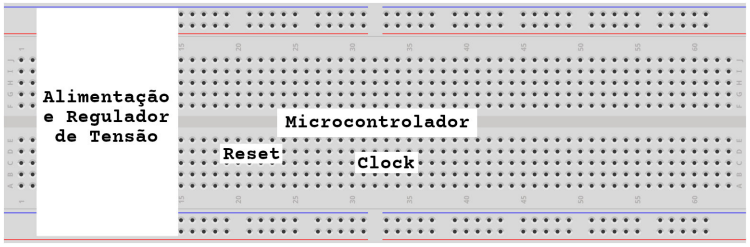
2 Materiais

Nosso Arduino Standalone terá, na protoboard, os seguintes grandes componentes:

1. **Alimentação**: usaremos uma bateria ou fonte de alimentação de 9 V mas o ATmega328P suporta, no máximo $5{,}5$ V. Precisaremos então de um circuito regulador de tensão que receba os 9 V e entregue 5 V para o microcontrolador.
2. **Microcontrolador**: é o ATmega328P em si. Você pode remover o microprocessador do Arduino, ou comprar um novo.
3. **Clock externo**: usaremos um circuito de clock externo de 16 MHz com um cristal oscilador.
4. **Circuito de reset**: um botão na protoboard permitirá que o microcontrolador seja reiniciado rapidamente, sem a necessidade de desconectar a fonte de energia.

A organização desses grandes componentes na protoboard será aproximadamente a exibida na Figura 1, a seguir:

Figura 1: Organização geral dos componentes



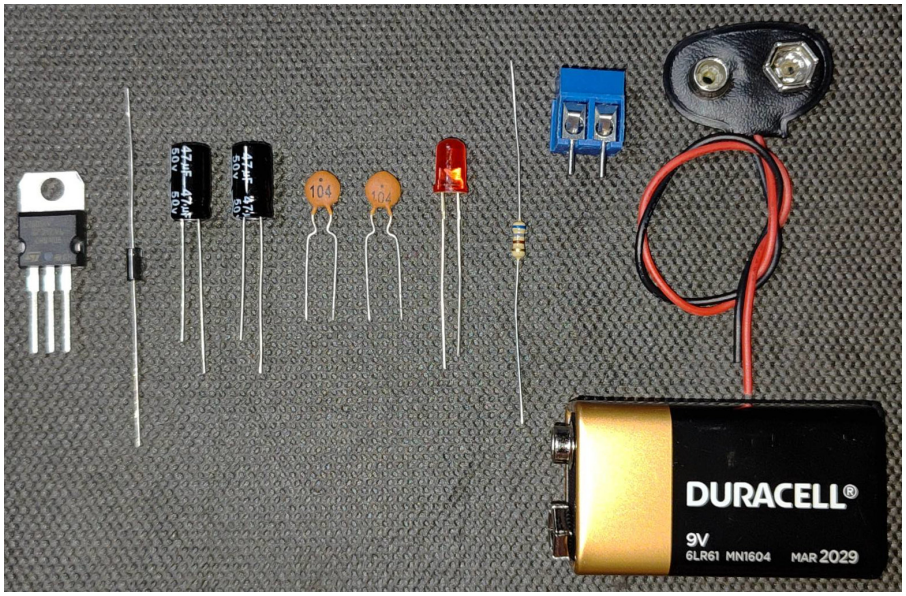
[3]Este artigo é, na verdade, o tutorial que eu queria ter tido acesso quando eu tentei montar um Arduino Standalone pela primeira vez.

[4]Este item parece um contra-senso pois acabamos de citar todas as vantagens de abandonar o Arduino e usar o microcontrolador independente. Sim, isso é meio que um contra-senso mesmo, mas estou partindo do pressuposto que você está montando seu primeiro Arduino Standalone e manter a arquitetura o mais próxima possível do Arduino, nesse momento, facilitará sua aprendizagem.

Os materiais e componentes necessários são os seguintes:

- **Alimentação**: a lista a seguir e a Figura 2 mostram os componentes que utilizaremos para a alimentação do microcontrolador. Alguns itens são opcionais mas recomendados neste estágio inicial de seu aprendizado.
  - 1 regulador linear de tensão L7805CV
  - 1 diodo 1N4007
  - 2 capacitores eletrolíticos de 47 µF
  - 2 capacitores cerâmicos de 100 nF
  - 1 LED vermelho de 5 mm (opcional, apenas para indicar que a protoboard está energizada)
  - 1 resistor de 680 Ω (opcional, apenas para indicar que a protoboard está energizada)
  - 1 conector borne de duas vias
  - 1 clip para bateria de 9 V e 1 bateria de 9 V

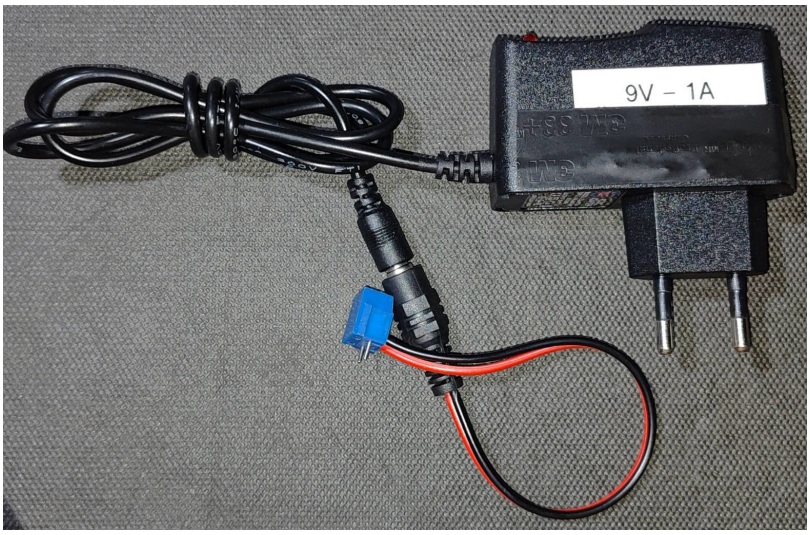
Figura 2: Componentes para a alimentação



Se você não quiser usar uma bateria, pode utilizar uma fonte de alimentação de 9 V (1 A)[5], com um adaptador com borne de duas vias conforme a Figura 3 (na página 6), para conectar a fonte à protoboard. Para mantermos nosso Arduino Standalone o mais portátil possível usaremos uma bateria comum mesmo.

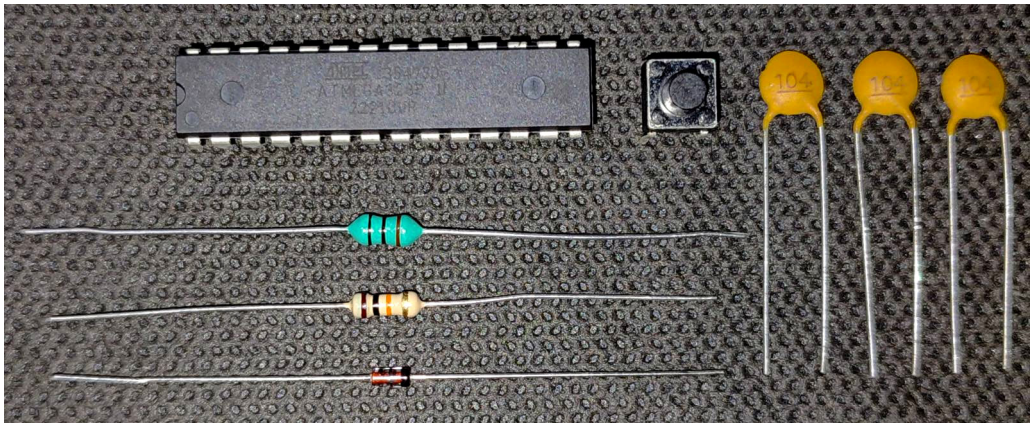
[5]Estou usando uma fonte que fornece corrente de até 1 A, mas você pode utilizar qualquer outra fonte de 9 V que tiver disponível.

Figura 3: Alternativa para alimentação



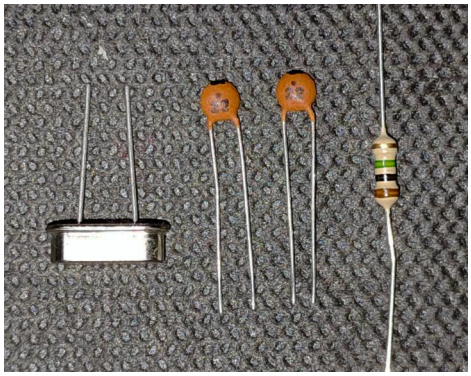
- **Microcontrolador** e **circuito de reset**: precisaremos dos materiais ilustrados na lista abaixo (e na Figura 4):
  - 1 microcontrolador ATmega328P-PU (você pode retirar, com muito cuidado, o microcontrolador de sua placa Arduino atual, ou comprar um novo microcontrolador)
  - 1 indutor axial de 10 µH
  - 1 resistor de 10 kΩ
  - 1 diodo 1N4148
  - 3 capacitores cerâmicos de 100 nF
  - 1 push button

Figura 4: ATmega328P-PU e circuito de reset



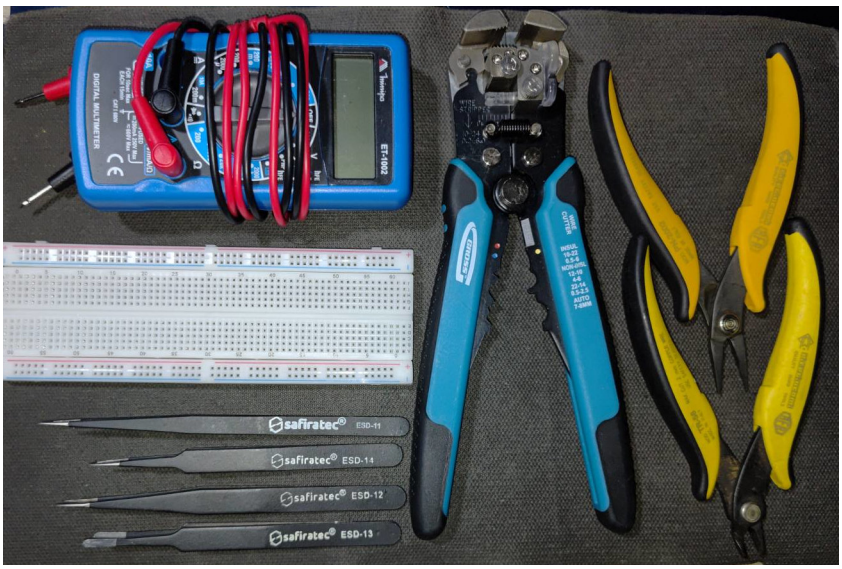
- **Clock externo**: para montar o circuito de clock precisamos dos materiais a seguir (ilustrados na Figura 5):
  - 1 cristal oscilador de 16 MHz
  - 2 capacitores cerâmicos de 22 pF
  - 1 resistor de 1 MΩ

Figura 5: Clock externo



Além dos materiais e componentes principais listados anteriormente, também será necessário uma protoboard de 830 furos (eu prefiro os modelos que têm quatro linhas de energia separadas), ferramentas diversas (alicates de corte, decapadores de fio, alicates de bico fino, pinças) e, se disponível, um multímetro para verificar tensão, corrente e resistência (Figura 6):

Figura 6: Ferramentas úteis



Por fim precisamos de fios para as conexões na protoboard e aqui há uma regra clara: **não use** fios jumper flexíveis, como aqueles que são comumente utilizados em kits de Arduino para iniciantes. Fios jumper flexíveis são grandes, ficam sobrando e fazem uma tremenda confusão da protoboard. Para montar um Arduino Standalone de modo "profissional" precisamos usar **fios rígidos sólidos** para eletrônica, com bitola entre 24 AWG[6] e 22 AWG (entre $0{,}511$ mm e $0{,}644$ mm). Tenha sempre à mão um estoque de diferentes cores desses fios rígidos, conforme a Figura 7 abaixo:

Figura 7: Fios rígidos 22 AWG

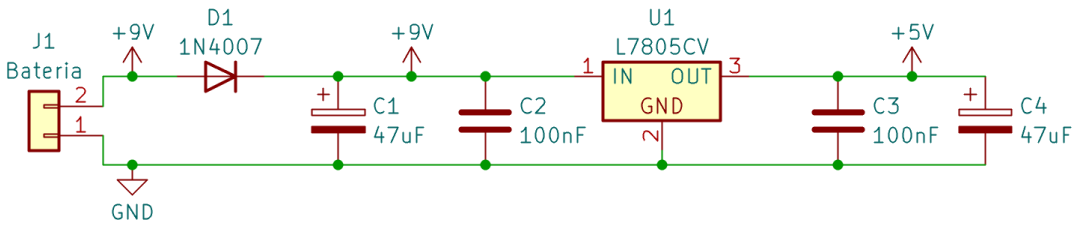
3 Montagem

3.1 Alimentação

A primeira cosia que montaremos na protoboard é o circuito regulador de tensão, que receberá os 9 V da bateria e fornecerá 5 V para alimentar o microcontrolador e demais componentes. O esquema desse circuito regulador está apresentado na Figura 8:

[6] *American Wire Gauge* (AWG): é uma escala logarítmica americana para a identificação das bitolas de fios rígidos, especialmente os fios utilizados em eletrônica.

Figura 8: Circuito regulador de tensão



O diodo D1 (1N4007) é uma proteção contra polaridade reversa. Se você inadvertidamente conectar a bateria ao contrário, o diodo bloqueará a corrente impedindo que ela flua para o regulador L7805CV e para o resto do circuito (potencialmente queimando os componentes). Esse diodo causa uma queda de tensão de até $1{,}1$ V dependendo do modelo e do fabricante[7].

Os capacitores eletrolíticos de 47 µF atuam fazendo filtragem de ruídos de baixa freqüência e como reservatório de energia (nessa função costumam ser chamados de *bulk capacitors*). O capacitor C1 (na entrada) garante que a tensão na entrada do regulador L7805CV permaneça estável mesmo se a bateria tiver uma queda momentânea de tensão. Já o capacitor C4 (na saída) melhora a resposta a transientes de carga e evita que a tensão de 5 V caia bruscamente (por exemplo, se um módulo que de repente puxa mais corrente).

Os capacitores cerâmicos de 100 nF atuam como capacitores de desacoplamento e fazem filtragem de ruídos de alta freqüência. O capacitor C2 (na entrada) filtra ruídos que venham da bateria, diodo ou fios. Já o capacitor C3 (na saída) filtra ruídos que venham do L7805CV ou da própria carga. Esses capacitores devem ser localizados o mais próximo possível do L7805CV e atual como um reservatório de energia local de resposta rápida.

O L7805CV é um regulador linear de tensão que aceita de 7 V a 25 V na entrada, fornece 5 V na saída, suporta correntes de até $1{,}5$ A, e conta com mecanismos de proteção contra sobrecarga térmica (se ele esquentar demais ele "desliga") e contra curtos circuitos. Ele é um regulador excelente para projetos de Arduino Standalone e pequenos projetos de sistemas embarcados de iniciantes, mas tem uma desvantagem: ele precisa dissipar bastante calor e pode esquentar muito dependendo da carga que será consumida no circuito. O cálculo da potência (em Watt) que o regulador precisará dissipar é dado por:

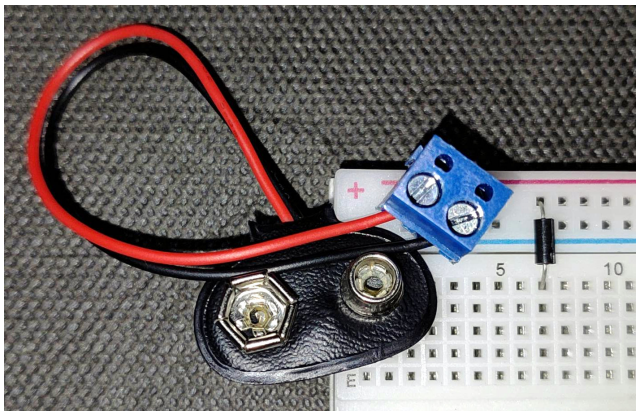
$$P = (T_i - T_o) \times I\,, \tag{1}$$

onde P é potência em Watt, T_i é a tensão na entrada, T_o é a tensão na saída e I é a corrente consumida em seu projeto (tensões em Volt e corrente em Ampere). Voltaremos a isso posteriormente.

[7] Note que devido à queda de tensão causada pelo diodo D1, a tensão indicada entre os capacitores C1 e C2 ($+9$ V) não é totalmente correta, deveria ser algo em torno de 8 V. Em nome da simplicidade e para deixar a explicação mais didática, preferi deixar o circuito com essa pequena imprecisão.

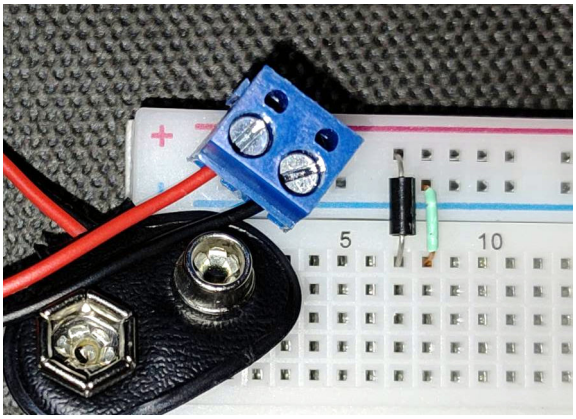
Inicie a montagem parafusando o clip da bateria no conector borne de duas vias, e coloque o borne na protoboard verificando corretamente se os polos positivos e negativos do borne estão alinhados com as trilhas vermelha e azul da protoboard (não coloque a bateria ainda). Depois coloque o diodo 1N4007 conectando a trilha positiva a uma linha da protoboard (lembre-se de que o diodo é polarizado, você precisa colocar o ânodo[8] do diodo na trilha positiva de 9 V e o cátodo em uma linha da protoboard) conforme a Figura 9:

Figura 9: Borne e diodo



Depois, com um fio rígido, faça uma conexão entre a trilha negativa a uma linha da protoboard ao lado do diodo, conforme a Figura 10:

Figura 10: Conexão do *ground* (GND)

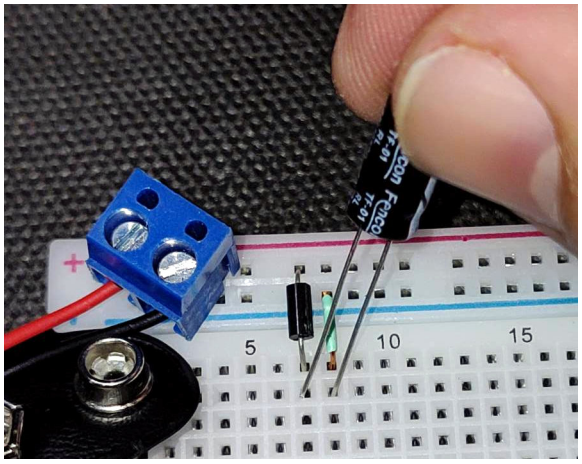


O próximo passo é conectar os capacitores de entrada. Coloque o capacitor eletrolítico de 47 µF na protoboard, conectando o ânodo do capacitor com o cátodo do

[8]Lembre-se de que o **ânodo** é o polo positivo e o **cátodo** é o polo negativo. O ânodo do diodo 1N4007 é o lado sem nenhuma marcação, e o cátodo é o lado que contém uma pequena faixa branca ou cinza.

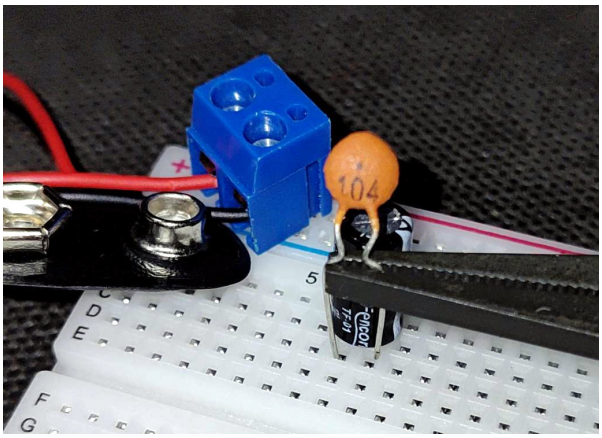
diodo, e o cátodo do capacitor na mesma linha do GND da protoboard, conforme a Figura 11. Cuidado ao fazer essa conexão pois capacitores eletrolíticos são polarizados e uma conexão invertida pode até causar uma pequena explosão (o cátodo do capacitor é indicado por uma faixa branca com sinais negativos).

Figura 11: Capacitor 47 µF de entrada



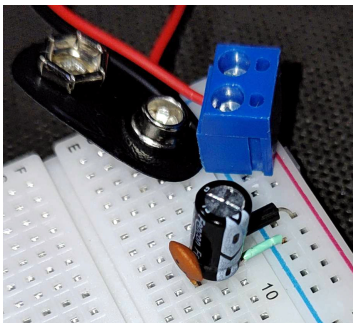
Agora devemos conectar o capacitor cerâmico de 100 nF de entrada. Esse capacitor não é polarizado e pode ser colocado de qualquer lado, conforme a Figura 12:

Figura 12: Capacitor 100 nF de entrada



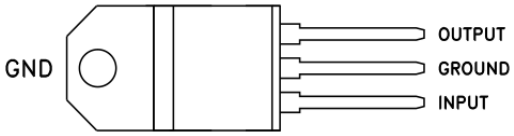
O resultado até aqui, com a colocação dos capacitores de entrada, está mostrado na Figura 13 (página 12).

Figura 13: Capacitores de entrada



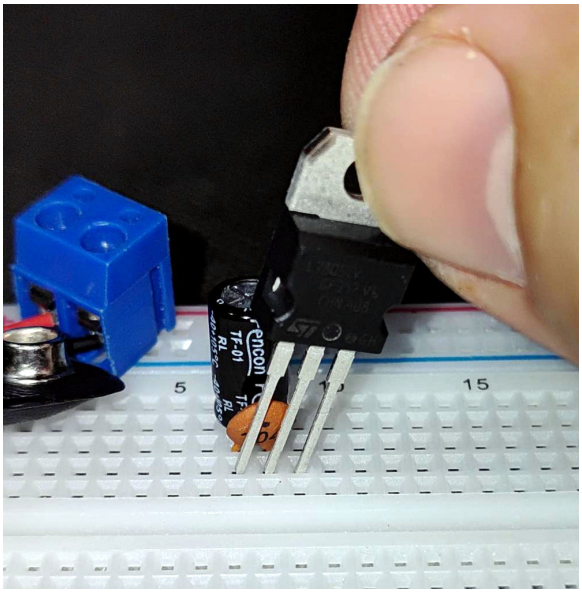
O próximo passo é colocar o regulador L7805CV. Conforme o *datasheet* do meu regulador os pinos são dispostos como na Figura 14 (talvez você tenha que conferir a pinagem do modelo exato do regulador que você está utilizando; na dúvida consulte o *datasheet*).

Figura 14: Pinos do L7805CV



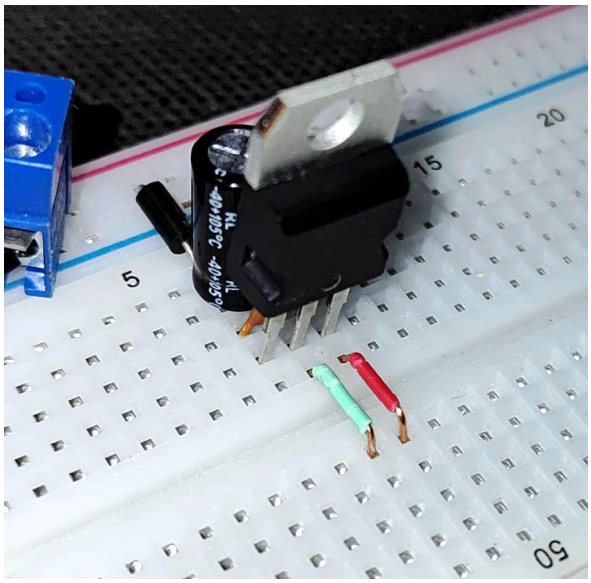
Para a colocação do L7805CV na protoboard: alinhe o pino de *input* do regulador com a linha de 9 V, e o pino de *ground* do regulador com a linha GND, conforme a Figura 15:

Figura 15: Regulador L7805CV



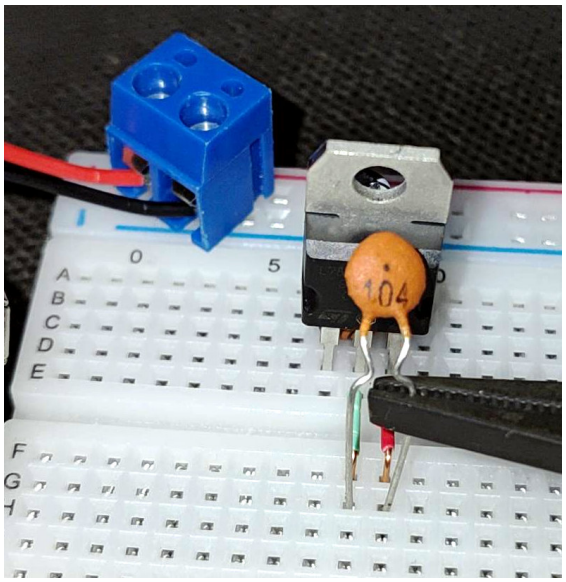
Para termos mais espaço para trabalhar, vamos conectar o GND e a saída de 5 V do regulador no outro lado da protoboard, conforme a Figura 16:

Figura 16: GND e 5 V



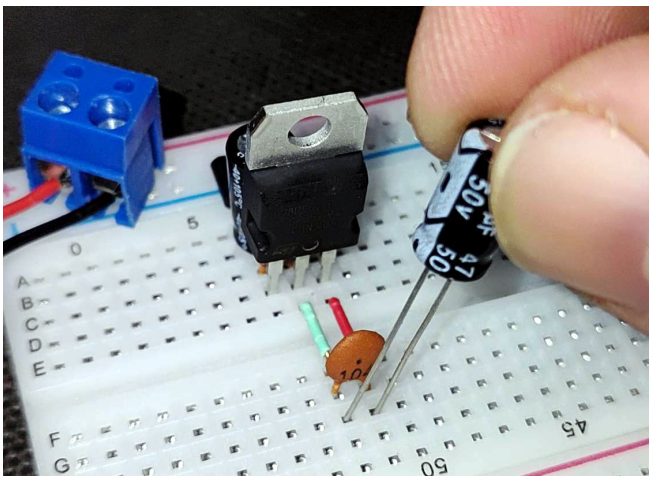
Coloque agora mais um capacitor cerâmico de 100 nF entre o GND e o 5 V, conforme a Figura 17. Idealmente esse capacitor deveria ficar o mais próximo possível do regulador de tensão mas pela falta de espaço na protoboard vamos deixá-lo do outro lado mesmo.

Figura 17: Capacitor de 100 nF da saída



O segundo capacitor eletrolítico de 47 µF deve ser colocado agora. Alinhe o ânodo do capacitor com a linha de 5 V e o cátodo do capacitor com a linha de GND, conforme a Figura 18:

Figura 18: Capacitor 47 µF de saída



Agora use fios rígidos para conectar a saída de 5 V e o GND do regulador de tensão às trilhas vermelha e azul da protoboard, conforme a Figura 19 e a Figura 20:

Figura 19: Alimentação de 5 V da protoboard

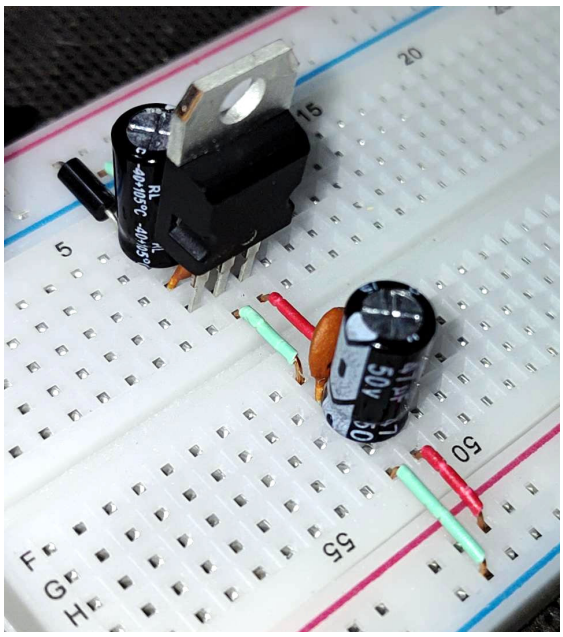
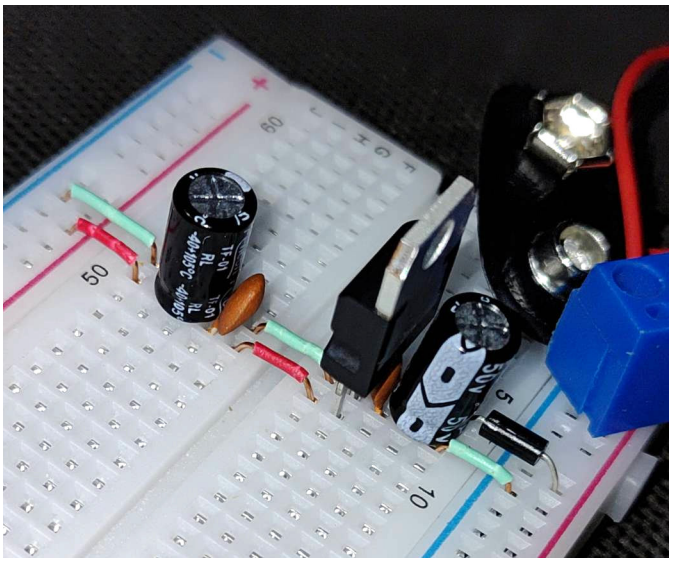
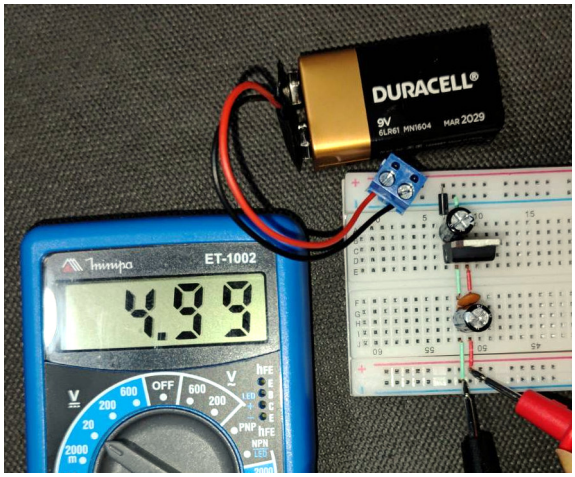


Figura 20: Alimentação de 5 V da protoboard



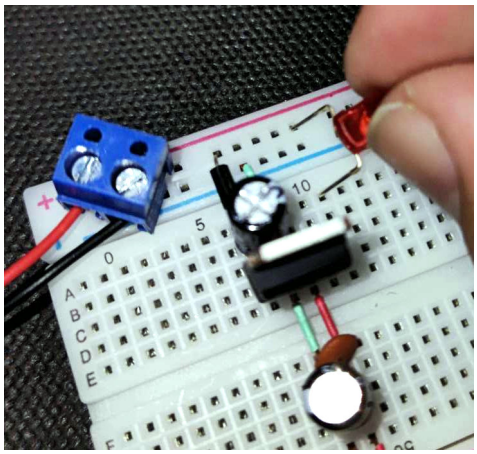
Coloque a bateria e verifique se as conexões estão corretas com um multímetro: ao medir a tensão na trilha inferior da protoboard, você deve obter 5 V conforme a Figura 21 (a tensão medida, de 4,99 V está dentro da margem de tolerância do regulador).

Figura 21: Verificação da tensão



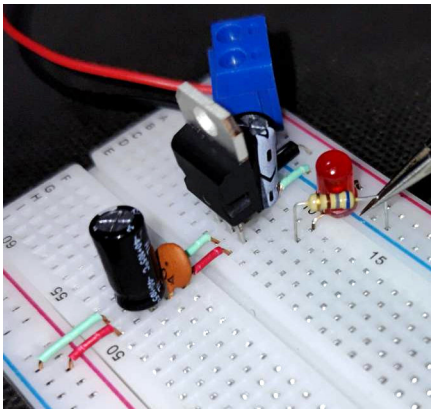
Agora já temos o circuito regulador de tensão montado e funcionando perfeitamente. Um pequeno inconveniente desse circuito é que não há nenhuma indicação visual de que o circuito está energizado. Assim, como um refinamento opcional, vamos colocar um LED indicativo de que o circuito está energizado e ligado. Coloque o ânodo do LED diretamente na trilha de 9 V da protoboard e o cátodo do LED em uma linha vazia conforme a Figura 22:

Figura 22: LED indicativo



Agora vamos colocar o resistor de $680\,\Omega$ entre o cátodo do LED e a trilha de GND da protoboard, conforme a Figura 23:

Figura 23: Resistor do LED



Esse resistor serve para limitar a corrente que percorrerá o LED que, segundo as especificações do *datasheet*, suporta correntes de até $20\,\text{mA}$ com uma *voltage forward* de $2{,}1\,\text{V}$. A corrente que percorre o LED será então de:

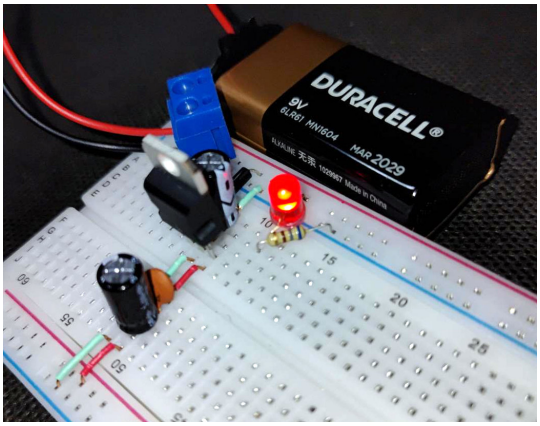
$$I = \frac{V}{R} = \frac{9{,}0 - 2{,}1}{680} = \frac{6{,}9}{680} \approx 10\,\text{mA} \tag{2}$$

A corrente de $10\,\text{mA}$ é segura para o LED mas representa um consumo de bateria a mais para o projeto (uma bateria alcalina comum tem capacidade entre $400\,\text{mA h}$ a $600\,\text{mA h}$ e o LED iria esgotar a bateria entre $40\,\text{h}$ a $60\,\text{h}$). Se você não precisa ou não quer esse consumo extra, você pode remover o LED completamente. Particularmente eu gosto de uma indicação visual de que o circuito está energizado.

Uma alternativa para diminuir o consumo da bateria é colocar resistores maiores para diminuir a corrente que percorre esse LED.

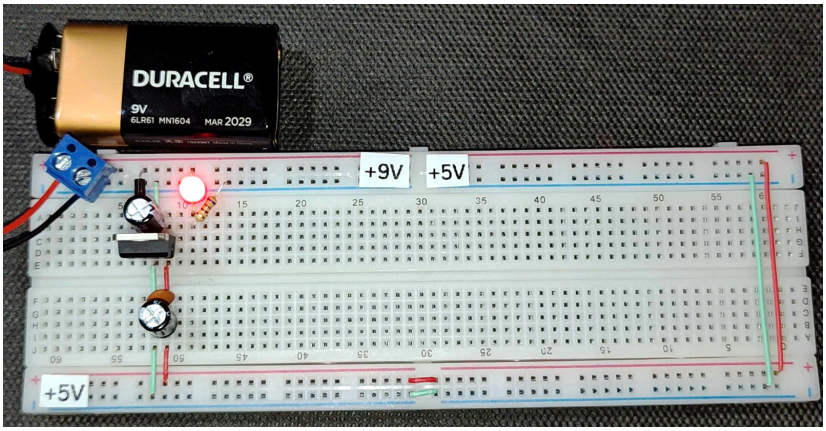
O circuito de alimentação está quase terminado. Coloque a bateria e verifique se o LED está funcionando conforme a Figura 24:

Figura 24: Regulador com LED indicativo



Por fim, como estou usando uma protoboard com quatro trilhas de alimentação independente, conectei as trilhas onde quero disponibilizar 5 V e coloquei pequenas etiquetas para identificar as tensões. A protoboard com o circuito de alimentação finalizado está mostrada na Figura 25.

Figura 25: Circuito de alimentação finalizado

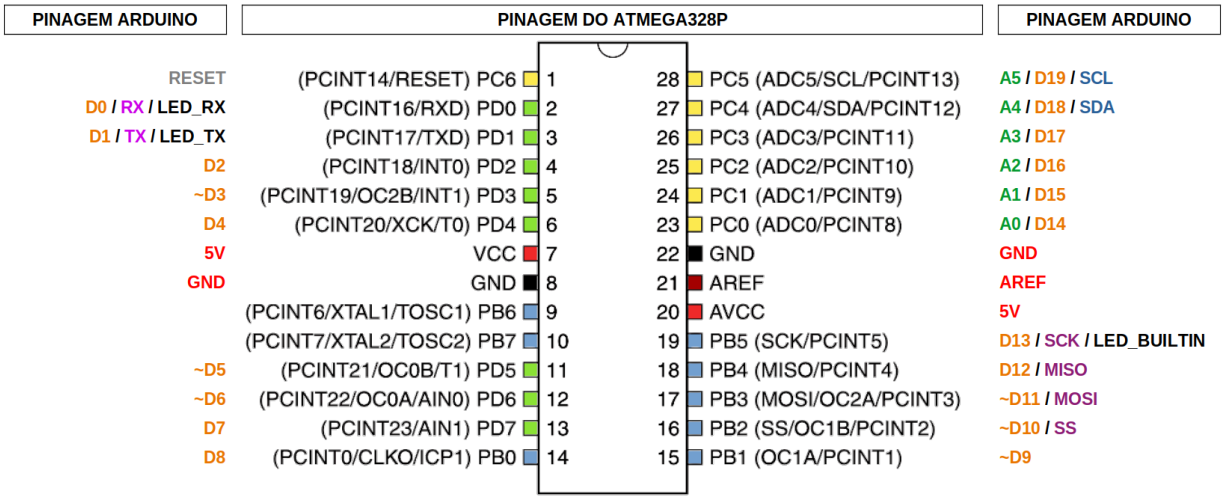


Ainda é possível fazer algumas outras pequenas melhorias no circuito de alimentação como, por exemplo, colocar um diodo adicional entre os pinos de *input* e *output* do regulador (para proteger contra tensão reversa) ou acrescentar outros diodos em série com o diodo D1 para diminuir a tensão que o regulador recebe. Entretanto essas melhorias são opcionais para esse nosso pequeno projeto e, por isso, não serão feitas.

3.2 Microcontrolador e reset

O próximo passo é fazer a montagem do microcontrolador ATmega328P-PU e do circuito de reset (que faz o microcontrolador reiniciar). Uma das primeiras coisas que você deve ter em mente é que os números dos pinos do Arduino são completamente diferentes dos números reais dos pinos do microcontrolador (o Arduino abstrai muito da complexidade do hardware e esconde as configurações e nomes verdadeiros dos componentes do microcontrolador). Em qualquer projeto de Arduino Standalone você precisa ter sempre à mão um mapeamento entre os pinos do microcontrolador e os nomes "bonitinhos" que o Arduino utiliza. A Figura 26 traz esse mapeamento:

Figura 26: Mapeamento ATmega328P × Arduino



Uma explicação mais detalhada sobre a pinagem do ATmega328P é necessária. O microcontrolador tem 28 pinos físicos, 14 de cada lado, numerados fisicamente de 1 a 28. O pino 1 é identificado por estar do lado esquerdo de um entalhe em forma de meia-lua em uma das extremidades do microcontrolador.

Esse Microcontrolador tem três portas de I/O[9] de propósito geral que são chamadas de **Port B**, **Port C** e **Port D**. Cada uma dessas portas pode ter até o máximo de oito pinos de I/O. Esses pinos são genericamente chamados de *General-Purpose I/O* (GPIO). Por exemplo: a Port C tem 7 pinos, chamados de PC0 até PC6, que estão conectados aos pinos físicos 1 e 23–28 do microcontrolador (pinos de cor amarela na Figura 26). Note também que cada um desses pinos pode exercer mais de uma função no microcontrolador, por exemplo: o pino PC5 pode exercer três funções diferentes, ADC5, SCL, PCINT13 (identificadas na pinagem do microcontrolador, logo ao lado do nome do pino). Essa nomenclatura do microcontrolador é mascarada pelos nomes utilizados no Arduino, de forma que o pino PC5 do microcon-

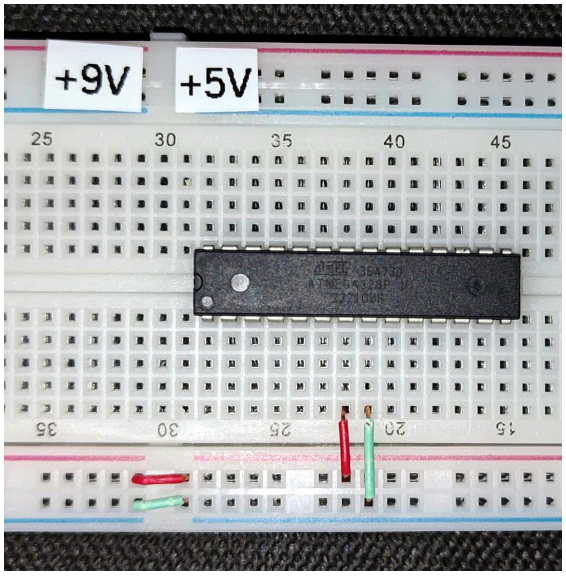
[9] I/O: *input*/*output*; entrada/saída.

trolador é conhecido por pino A5 no Arduino; o pino PD5 do microcontrolador é conhecido por pino D5 no Arduino; o pino PB4 do microcontrolador é conhecido por pino D12 no Arduino; e assim por diante. A Figura 26 identifica todo o mapeamento dos pinos reais do ATmega328 com os nomes utilizados pelo Arduino. Note também que mesmo na nomenclatura do Arduino os pinos podem exercer mais de uma função, por exemplo: o pino A5 do Arduino também pode ser utilizado como pino D19 ou como pino SCL[10].

O estudo detalhado de cada um desses pinos e das funções que eles podem exercer está fora do escopo deste artigo mas é necessário que você tenha a Figura 26 por perto ao montar circuitos de Arduino Standalone para que você consiga substituir as conexões feitas diretamente no Arduino pelas conexões a serem feitas diretamente no microcontrolador.

Para começar a montagem do microcontrolador, coloque-o na protoboard, identifique o pino 1 e faça a ligação do VCC (que fornece 5 V ao ATmega328P) no pino 7, e do GND no pino 8, conforme demonstrado na Figura 27. Tenha **muito cuidado** para não errar as conexões aos pinos do microcontrolador!

Figura 27: VCC e GND

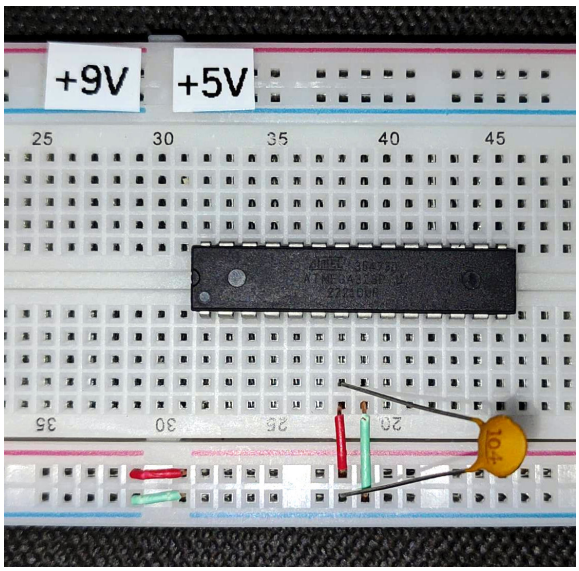


Agora coloque um capacitor cerâmico de 100 nF entre o pino 7 (VCC) e o GND, conforme a Figura 28. Isso é um **capacitor de desacoplamento** e é muito importante para a estabilidade do circuito. Ele realiza duas funções importantes para que o ATmega328P não trave ou reinicialize sozinho: a) reservatório local de energia para evitar quedas de tensão quando a comutação dos transístores do microcontrolador exigir picos rápidos de corrente; e b) filtragem de ruídos de alta freqüência

[10] Um dos pinos utilizados em comunicação *Inter-Integrated Circuit* (I^2C)

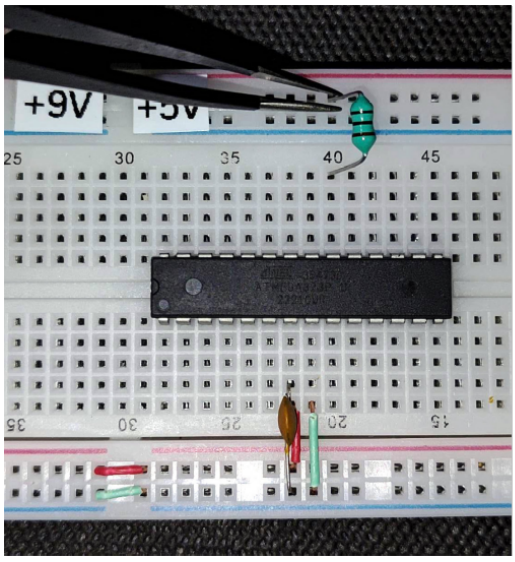
gerados por motores, relés ou outros componentes, pois o capacitor oferece um caminho fácil para o ruído ir direto para GND, desacoplando o ruído da alimentação principal do microcontrolador.

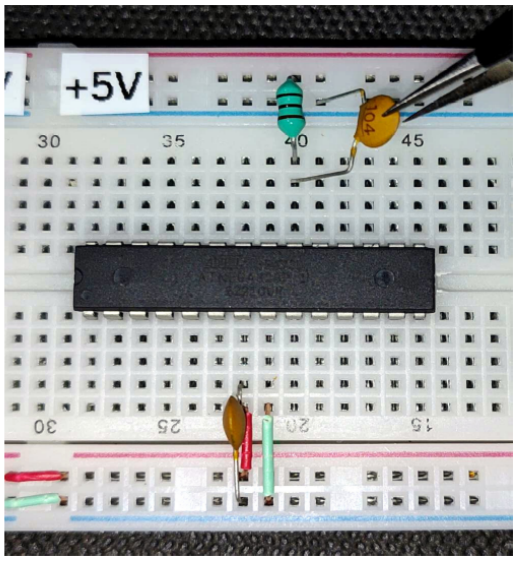
Figura 28: Capacitor entre VCC e GND



O próximo passo é colocar um indutor axial de $10\,\mu H$ entre a linha de $5\,V$ e o pino AVCC (pino 20), e um capacitor cerâmico de $100\,nF$ entre o pino AVCC (pino 20) e o GND, conforme a Figura 29:

Figura 29: Conexão do AVCC

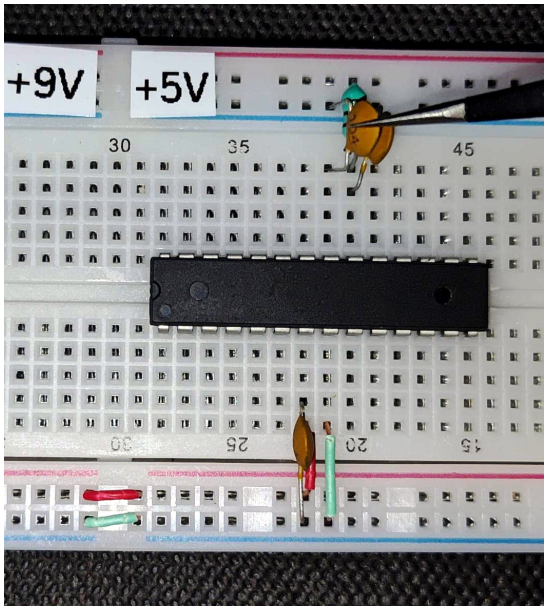




Por que esse conjunto de indutor mais capacitor é necessário? O pino AVCC é a fonte de tensão para o conversor analógico-digital (ADC) do ATmega328 (que lê tensões analógicas nos pinos PC0 até PC5), e esse conversor precisa de uma tensão de referência perfeitamente estável para fazer as leituras e comparar os sinais. O conjunto indutor mais capacitor forma um **Filtro LC** (Filtro Indutor-Capacitor) cuja função é proteger a exatidão das leituras analógicas. O indutor atua como um filtro passa-baixa e age como uma entrada seletiva: para corrente contínua o indutor é praticamente um fio comum, e deixa passar os $5\,\mathrm{V}$ para alimentar o ADC; para ruídos de alta freqüência o indutor apresenta alta impedância (resistência) e "bloqueia" as oscilações rápidas e ruídos gerados por chaveamento digital ou outras fontes. O capacitor ajuda o indutor e "engole" qualquer resíduo que o indutor deixar passar. O resultado é que a tensão no pino AVCC é uma energia "limpa" e estável.

Agora faça a conexão de um outro capacitor de $100\,\mathrm{nF}$ entre o pino AREF (pino 21) e o GND, conforme exibido na Figura 30:

Figura 30: Capacitor para AREF



Esse capacitor no AREF merece uma explicação um pouco mais detalhada. O pino AREF é a referência de tensão analógica para as leituras do conversor analógico-digital do microcontrolador. Essa tensão de referência é como uma "régua" a partir da qual as tensões analógicas são medidas: o ADC funciona comparando a tensão de entrada em um pino analógico com a tensão de referência em AREF. Essa tensão de referência pode ser configurada para:

- **AREF Externo**: nessa configuração o microcontrolador exige que você conecte ao pino AREF uma fonte de tensão de comparação externa. Se você

não conectar uma tensão de referência externa as leituras analógicas serão incorretas. Você usaria essa configuração se, por exemplo, você está utilizando um sensor que trabalha em $3{,}3\,\text{V}$: nesse caso você deve conectar uma tensão de $3{,}3\,\text{V}$ em AREF para que essa tensão seja a referência de comparação do ADC com as leituras feitas a partir do sensor. **Atenção:** o uso de AREF externo implica em certos cuidados no código, em especial você não pode fazer nenhuma leitura analógica sem antes informar e configurar o microcontrolador para utilizar corretamente essa tensão de referência externa. Note que mesmo que você utilize uma referência externa de tensão o capacitor de $100\,\text{nF}$ ainda é necessário entre AREF e GND.

- **AREF Interno em** $1{,}1\,\text{V}$: nessa configuração você não precisa conectar nenhuma tensão de referência no pino AREF, o microcontrolador utilizará uma tensão interna de $1{,}1\,\text{V}$ como referência analógica. Essa é uma situação não muito comum pois a maioria dos sensores analógicos utilizados trabalha com tensões de $5\,\text{V}$ ou $3{,}3\,\text{V}$. O capacitor entre AREF e GND continua sendo necessário.
- **AREF em AVCC**: nessa configuração estamos dizendo ao microcontrolador para usar como tensão de referência analógica a tensão de entrada no pino AVCC (que conectamos aos $5\,\text{V}$ filtrados e limpos com o Filtro LC). O capacitor entre AREF e GND continua sendo necessário.

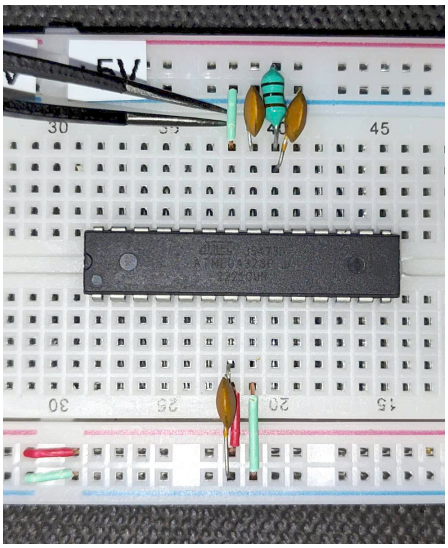
Por padrão o ATmega328P vem configurado de fábrica para usar "AREF Externo", ou seja, nós somos obrigados a conectar uma tensão de referência no pino AREF. Mas por que não fizemos isso e só conectamos um capacitor cerâmico de $100\,\text{nF}$ nesse pino? Esse é um outro exemplo de como o Arduino esconde as complexidades do hardware e do microcontrolador para simplificar um pouco as coisas para usuários iniciantes: ao compilar um código pela IDE do Arduino, usando as funções e bibliotecas do próprio Arduino, o Arduino automaticamente altera as configurações do ATmega328P para usar "AREF em AVCC" e ajusta outras coisas para que o ADC utilize como tensão de referência os $5\,\text{V}$ filtrados e limpos conectados ao pino AVCC[11]. O capacitor conectado entre AREF e GND ajuda a estabilizar ainda mais a tensão de referência para o conversor analógico-digital.

Ao usar "AREF em AVCC" não devemos conectar mais nenhuma outra fonte de tensão no pino AREF (21), apenas o capacitor de filtragem, pois a tensão de comparação será fornecida apenas por AVCC.

O próximo passo na montagem do microcontrolador é conectar o pino GND (22) à linha de GND da protoboard, conforme a Figura 31. O pino de GND (22) é conectado internamento ao pino de GND (8), mas ele fica disponível para o ADC e, às vezes, é encontrado com a identificação AGND.

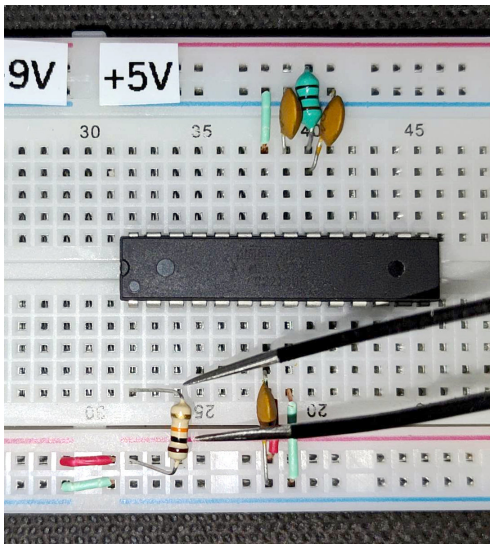
[11] Se você esta programando o microcontrolador em C puro (AVR C), sem usar as bibliotecas do Arduino, você é obrigado a ajustar manualmente em seu código a configuração do AREF desejada e a fazer alguns outros ajustes para que o conversor analógico-digital funcione corretamente. Isso é feito alterando-se a configuração de alguns registradores do microcontrolador, mas isso foge ao escopo deste artigo.

Figura 31: GND para ADC



O pino RESET (1) do microcontrolador deve ser conectado, através de um resistor de $10\,\text{k}\Omega$, à trilha de $5\,\text{V}$ da protoboard, conforme a Figura 32. Esse pino funciona da seguinte maneira: quando ele está em nível lógico "HIGH" ($5\,\text{V}$) o microcontrolador está funcionando normalmente; mas se ele for colocado em nível lógico "LOW" ($0\,\text{V}$) o microcontrolador é reiniciado.

Figura 32: Pino de RESET

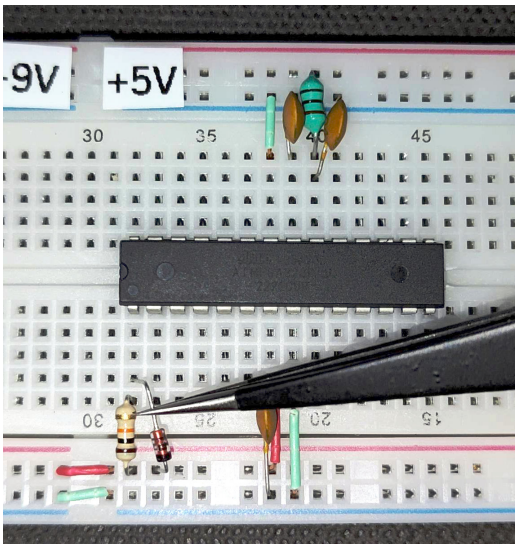


Nesse momento já temos todas as conexões básicas de alimentação do microcontrolador prontas. O próximo passo agora é montar um circuito de reset para o

microcontrolador pois para podermos gravar um novo programa o microcontrolador deve ser reiniciado. Vamos montar o circuito de RESET.

Coloque um diodo 1N4148 conectado entre o pino RESET (1) e a linha de 5 V da protoboard da seguinte maneira: o cátodo do diodo deve ficar conectado aos 5 V e o ânodo do diodo deve ficar conectado ao pino de RESET (1), ou seja: o diodo está "invertido" e não deixa passar corrente da alimentação para o pino de RESET (a corrente obrigatoriamente passa pelo resistor). Veja a Figura 33 para entender como esse diodo deve ser conectado.

Figura 33: Circuito de reset: diodo



Qual o propósito desse diodo? Bem, nesse momento ele não tem função praticamente nenhuma, ele está tecnicamente "inútil"[12] e poderia até ser removido do circuito sem grandes prejuízos. Mesmo assim é uma boa prática deixar esse diodo já conectado. O diodo 1N4148 é conhecido como **diodo de sinal** ou diodo de chaveamento. Isso significa que ele é extremamente veloz ao passar do estado "bloqueando" para o estado "conduzindo" energia. Ele está aí por um motivo de proteção futura se resolvermos usar um adaptador FTDI/USB-Serial para a programação do microcontrolador.

Para programar o microcontrolador, em muitos projetos de Arduino Standalone, nós usamos um adaptador FTDI/USB-Serial. Para que a IDE do Arduino consiga enviar o código automaticamente, ela precisa reiniciar o microcontrolador. Isso é feito ligando-se o adaptador ao pino RESET através de um capacitor de 100 nF (que não está na protoboard, mas costuma ser adicionado depois se realmente formos usar o adaptador FTDI/USB-Serial).

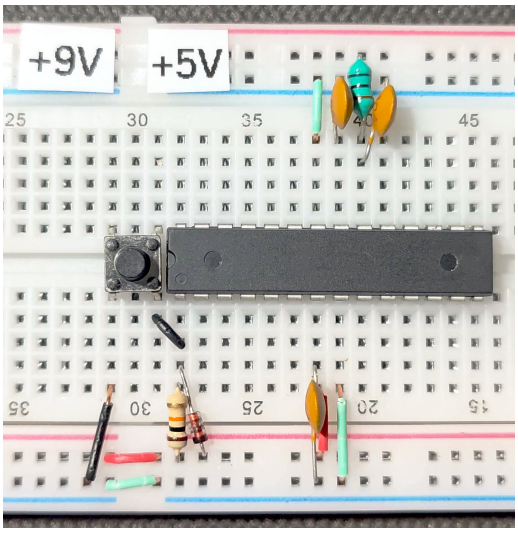
[12] Exceto, talvez, por uma pequena proteção contra descarga eletrostática se alguém tocar no botão com o dedo carregado de eletricidade estática: nesse caso o diodo ajuda a desviar essa energia para a fonte ao invés de danificar a porta de RESET do microcontrolador.

Mas um problema que ocorre ao usar esse adaptador é que ele força a tensão no pino de RESET ir para 0 V, reiniciando o microcontrolador, mas, quando a tensão volta para os 5 V da linha de tensão da protoboard, a tensão armazenada no capacitor se soma à tensão da linha de alimentação. Isso pode gerar um pico de tensão momentâneo no pino RESET que pode chegar até 10 V, causando efeitos inesperados ou prejudiciais no ATmega328P. O diodo 1N4148 age como uma válvula de escape (chamado de **clamping diode**) e funciona do seguinte modo: se a tensão no pino RESET subir acima de aproximadamente $5{,}7$ V, ele começa a conduzir e desvia esse excesso de tensão para a linha de 5 V da protoboard, mantendo o pino se RESET seguro.

Mesmo que o diodo 1N4148 não exerça função praticamente nenhuma em nosso projeto no momento, se adicionarmos um adaptador FTDI/USB-Serial para a programação do microcontrolador no futuro o circuito de RESET já estará protegido e, por isso, é melhor deixar esse diodo já conectado.

Para finalizarmos o circuito de reset temos que colocar um botão na protoboard que, ao ser pressionado, coloca o pino de RESET (1) em "LOW" e faz o microcontrolador se reiniciar. Para isso coloque um botão ao lado do microcontrolador, e ligue o botão no pino de RESET (1) e ao GND, conforme a Figura 34. Esse é o circuito de reset.

Figura 34: Circuito de reset: botão

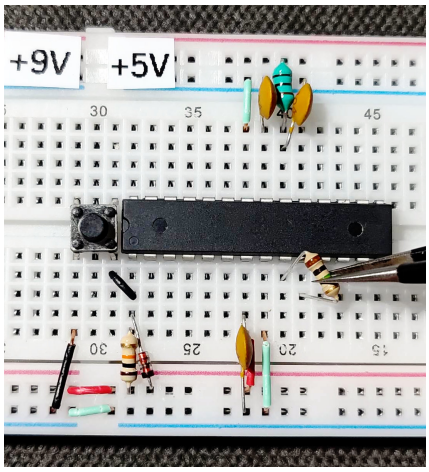


Nesse momento nosso microcontrolador já está totalmente montado e pronto: se quiséssemos já poderíamos usar o Arduino Standalone sem problema algum. Entretanto, nessa montagem, somos forçados a utilizar o clock interno do microcontrolador, que não é tão veloz quando comparado a um clock externo. Vamos portanto acrescentar um circuito de clock externo ao nosso Arduino Standalone.

3.3 Clock externo

Acrescentar um clock externo ao nosso Arduino Standalone não é difícil. Em primeiro lugar vamos colocar um resistor de $1\,\mathrm{M}\Omega$ entre os pinos 9 e 10 do microcontrolador (pinos XTAL1 e XTAL2), conforme a Figura 35.

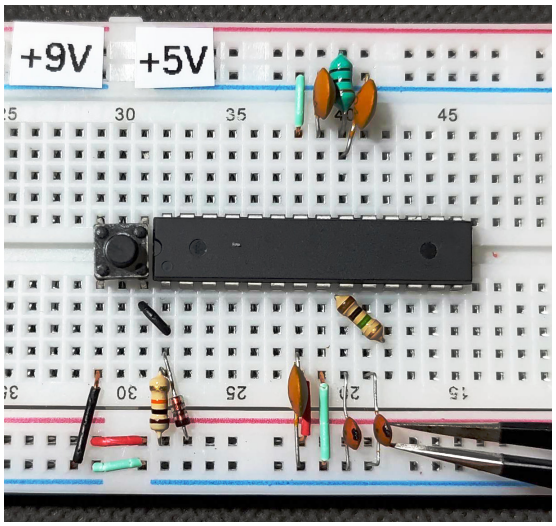
Figura 35: Clock externo: resistor



De modo simplificado esse resistor ajuda o cristal oscilador a começar a funcionar. A conexão é difícil de fazer na protoboard pois o resistor não cabe entre os pinos 9 e 10: a solução é dobrar os pinos de contato e deixar o resistor na diagonal.

Coloque dois capacitores cerâmicos de $22\,\mathrm{pF}$ para conectar os pinos 9 e 10 ao GND da protoboard, conforme a Figura 36.

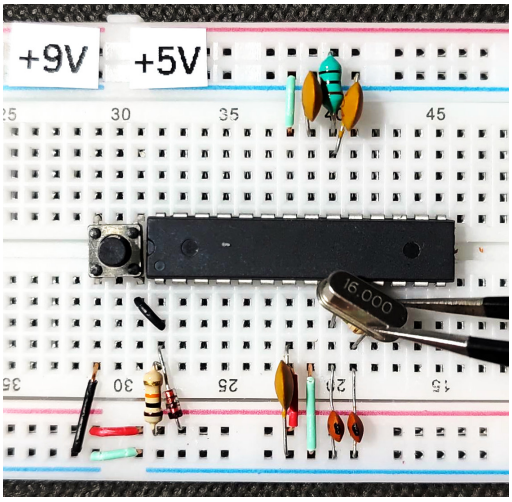
Figura 36: Clock externo: capacitores



Também de modo simplificado a função desses capacitores é filtrar ruídos indesejados e garantir que a oscilação do cristal gere uma onda senoidal limpa e na freqüência correta.

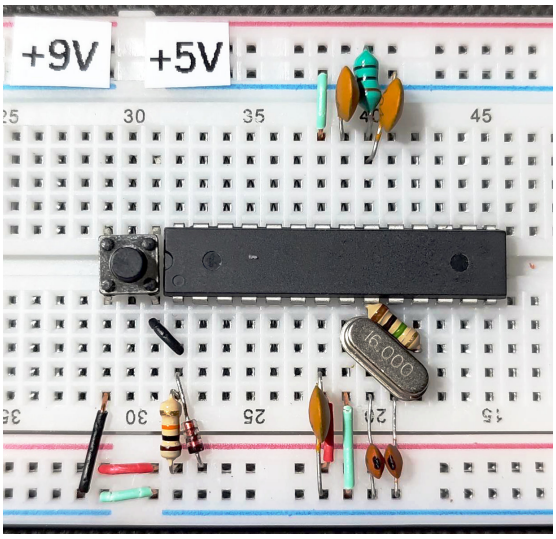
Por fim, coloque um cristal oscilador de 16 MHz entre os pinos 9 e 10 (XTAL1 e XTAL2), conforme a Figura 37. A colocação do cristal também é difícil pois ele não cabe adequadamente na protoboard e precisa ser colocado de lado.

Figura 37: Clock externo: cristal



Agora sim, de fato, terminamos a montagem do microcontrolador, com o circuito de reset e o clock externo. Você deve ter algo semelhante à Figura

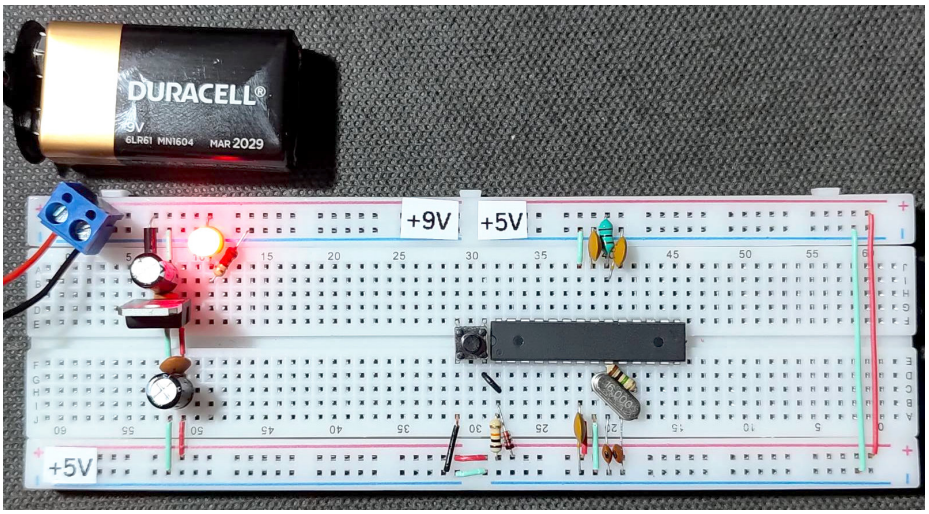
Figura 38: Microcontrolador, reset e clock



3.4 Montagem completa

Se você seguiu à risca as instruções de montagem, seu Arduino Standalone deve se parecer com a Figura 39, abaixo:

Figura 39: Arduino Standalone completo



Para referência, as figuras abaixo trazem todos os diagramas esquemáticos de nosso Arduino Standalone.

Figura 40: Esquemático: regulador de tensão

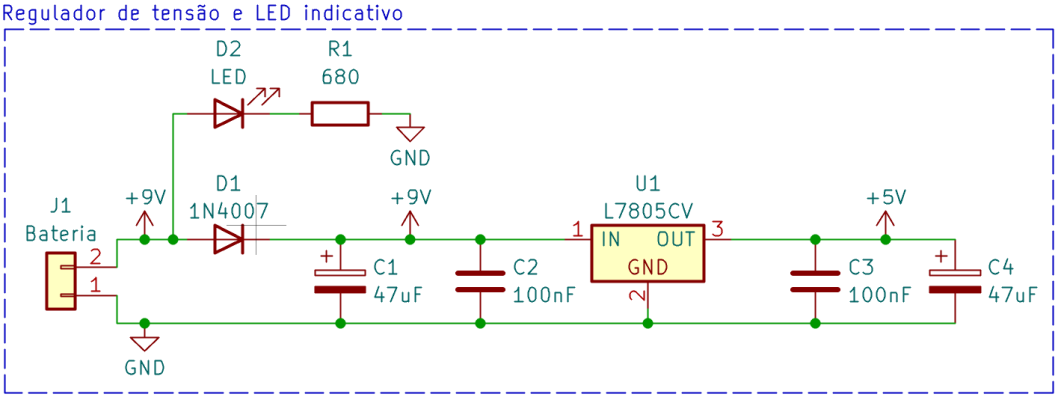


Figura 41: Esquemático: microcontrolador

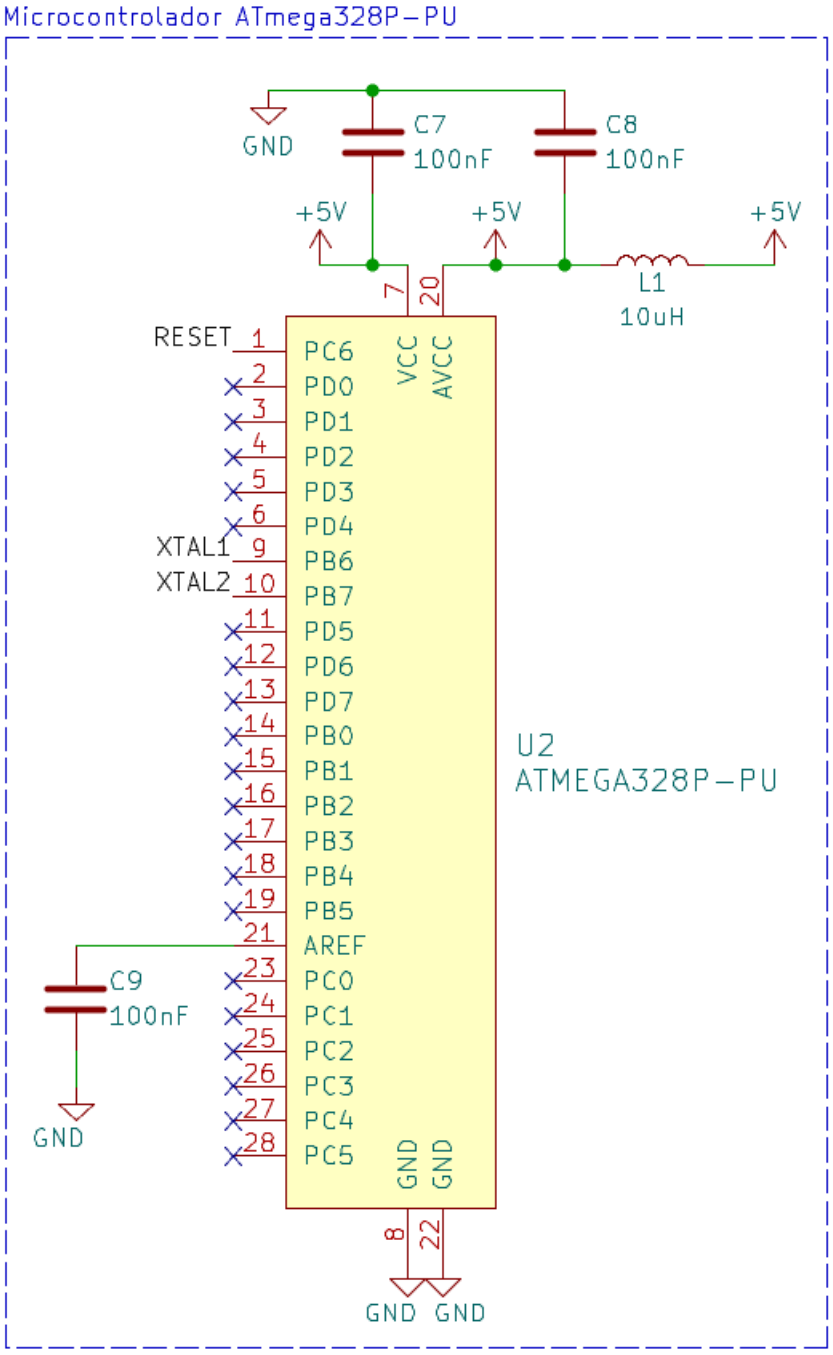


Figura 42: Esquemático: circuito de reset

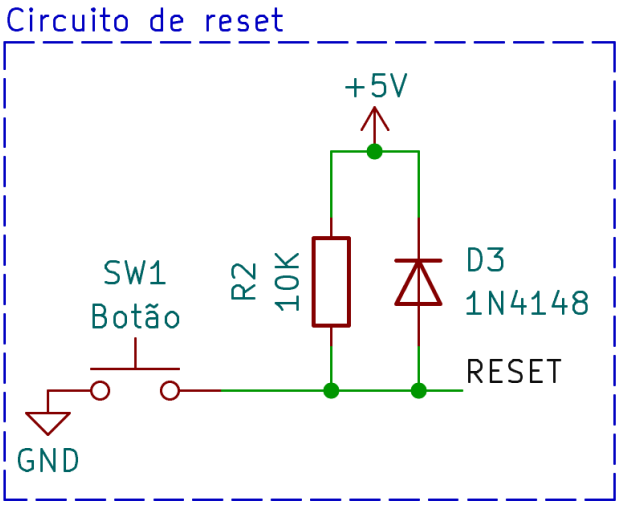
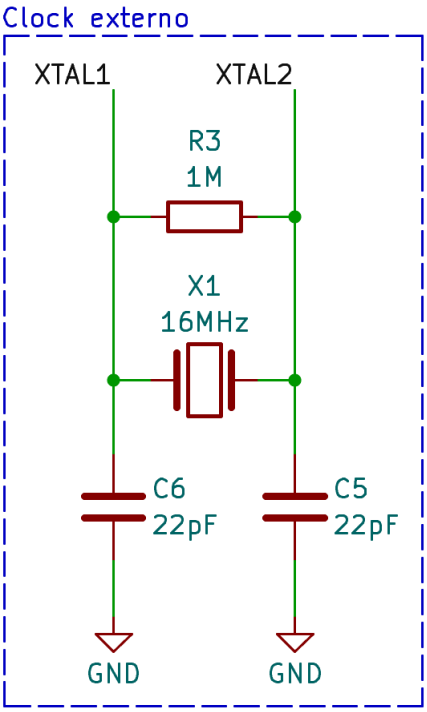


Figura 43: Esquemático: clock externo



4 Próximos passos

O que podemos fazer com nosso Arduino Standalone? No momento praticamente nada… antes de podermos utilizar o microcontrolador na protoboard para nossos projetos de sistemas embarcados, precisamos aprender a programá-lo diretamente na protoboard.

Se você está com pressa em ver seu Arduino Standalone funcionando, pode utilizar o seguinte artifício: utilize o Arduino normalmente para o seu projeto, fazendo o protótipo conforme você já está acostumado. Ao terminar, **remova cuidadosamente** o microcontrolador do Arduino e coloque na protoboard fazendo exatamente as conexões que mostramos aqui. Depois reproduza na protoboard todas as conexões que você tinha quando estava usando a placa do Arduino (lembre-se de que os pinos do Arduino não correspondem exatamente aos pinos do microcontrolador — utilize a Figura 26 para se orientar). Agora você tem um microcontrolador já programado para seu projeto. Claro, essa não é a melhor solução: ficar removendo e recolocando o pode acabar danificando o microcontrolador.

As melhores soluções para a programação de seu Arduino Standalone são usar um adaptador FTDI/USB-Serial ou um programador AVR dedicado. Falarei sobre isso em artigos futuros.