Pimp My Arduino

Book · December 2016

CITATIONS
0

READS
46

3 authors, including:

Michael Barney Galindo Júnior
Federal University of Pernambuco
2 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Otacilio Saraiva Maia Neto
Universidade de Pernambuco
1 PUBLICATION **0** CITATIONS
SEE PROFILE

Pimp My Arduino

Criando Arduinos Sob Medida.

Sumário

1 - Prefácio

O ***Arduino*** revolucionou o mundo. Essa afirmação aparentemente exagerada se torna cada vez mais real na medida em que se analisa os impactos que essa placa de prototipação de circuitos eletrônicos acarretou no mundo *Maker.* Sua história de origem apresenta uma de suas maiores conquistas: a eletrônica e a programação disponível para todos. Criada na Itália por um grupo de estudantes de *design*, o *Arduino* teve como objetivo principal a facilitação do desenvolvimento de qualquer aparelho eletrônico de maneira eficaz porém simples para qualquer "inventor". Incrivelmente, a placa atingiu muito mais do que seu objetivo inicial, se tornando um fenômeno utilizado mundialmente que catalisou o desenvolvimento exponencial da comunidade *Maker,* onde pessoas de qualquer idade ou profissionalização têm a possibilidade de expressar a sua criatividade criando o que vier em suas mentes.

Após o sucesso mundial, começaram a surgir novas versões do *Arduino. Nano, Mega, Mini, Esplora….* A marca *Arduino* atualmente possui inúmeras adaptações feitas para cada tipo de inventor ou projeto. Porém, o maior meio de desenvolvimento de novas maneiras de se reinventar a placa veio da gigantesca comunidade que cerca a plataforma. Por onde se procura, é possível encontrar cada vez mais adaptações criadas por pessoas comuns que tenham como interesse criar uma placa Compatível com *Arduino* feito sob medida, customizadas para atingir os objetivos que ela e seus projetos necessitam.

O presente livro tem como objetivo detalhar da melhor maneira possível todas as informações necessárias para o desenvolvimento de uma placa Arduino própria. O funcionamento de um Arduino é muito simples e intuitivo, porém é necessário saber de algumas informações pertinentes ao substituir e adicionar componentes de um projeto já conhecido. Além disso, o universo Arduino não se limita apenas ao *hardware* (Eletrônica), podendo também abranger diversas questões de *software* (Programação).

A Banana Digital convida você, leitor, para explorar o mundo das possibilidades de uma simples placa.

2 - Mergulhando de Cabeça no UNO

De maneira geral, a versão UNO da placa *Arduino* (Figura 1) é a mais conhecida, simplesmente porque ela contém uma boa quantidade de portas de entrada e saída, memória interna, velocidade de processamento para projetos de pequeno porte. Enfim, é ideal para qualquer pessoa iniciando a sua aventura no mundo da eletrônica e se encaixa na grande maioria dos projetos. Neste capítulo do livro, vamos detalhar de maneira simplificada algumas características dessa placa.

Primeiramente, devemos observar e entender algumas das especificações dela. Observe a tabela abaixo:

Microcontrolador	ATmega328P
Tensão de Operação	5 Volts
Tensão de Entrada (Recomendada)	7-12V
Tensão de Entrada (Limites)	6-20V
Pinos I/O	14 (6 com PWM)
Pinos Analógicos	6
Corrente por pino I/O	40 mA
Corrente no pino 3.3v	50 mA
Memória Flash	32 kB
SRAM	2 kB
EEPROM	1 kB
Velocidade de Clock	16 MHz

Tabela 1 - Especificações do *Arduino* UNO

Se você já possui conhecimento sobre todas essas informações e sabe o que elas significam, sinta-se livre para ir para o próximo capítulo, caso contrário, iremos analisar esses dados.

Figura 1 - Arduino UNO

1. Primeiramente, o **Microcontrolador**, isso é, o *chip* que controla o *Arduino*. É nele onde é realizada toda a matemática, onde o código é armazenado e onde as instruções são realizadas, ou seja, ele é o “cérebro” da placa. No caso da UNO, o Microcontrolador utilizado é o Atmega328P, desenvolvido pela *Atmel.*

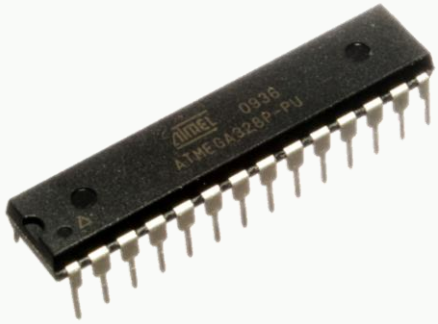


Figura 2 - ATMega328P

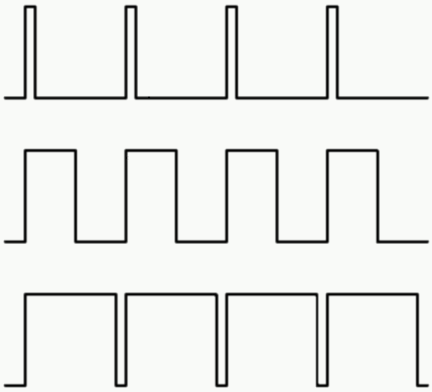
2. A **Tensão de Operação** não apenas é a tensão necessária para o funcionamento da placa, mas também é a utilizada nos pinos I/O (Entrada e Saída). Portanto em todas as saídas e entradas digitais da placa, deverá “entrar” ou “sair” 5 volts.

3. Tanto a **Tensão de Entrada (Recomendada)** quanto a **Tensão de Entrada (Limite)** representam a tensão necessária na alimentação do dispositivo para o seu funcionamento. Esses valores decorrem do Regulador de Tensão da placa, sendo ele o componente que converte uma tensão mais elevada para uma tensão de referência fixa, nesse caso os 5V de Tensão de Operação. Portanto, o Regulador de Tensão do dispositivo (NCP1117) necessita no mínimo uma tensão de entrada de 6V para que na sua saída se tenha 5V. Além disso, só é permitido ter na sua entrada a tensão máxima de 20V, devido à potência máxima do componente, sendo essas as tensões limites (6V - 20V). Porém, para

que a vida útil da placa seja a melhor possível, é recomendado a aplicação de uma tensão entre 7V e 12V.

4. Os **Pinos I/O,** também chamados de portas digitais, são as entradas e saídas digitais da placa. Através do código, é possível configurar cada uma como entrada ou saída de tensão (*In* ou *Out*). Se for configurada como saída, pode-se programá-la para o estado ligado (chamado de "*HIGH"* ou 1), onde a tensão na porta é de 5V, ou para o estado desligado (chamado de "*LOW"* ou 0), onde a tensão na porta é de 0V. Caso esteja configurado como entrada, é possível ler o estado da tensão aplicada sobre ela de maneira binária, ou seja, se está com 5V (1) ou com 0v (0). No caso da Arduino UNO, são 14 portas digitais para serem usufruídas.

5. Alguns dos Pinos I/O podem também ser configurados para emitirem sinais ***PWM*** *(*Modulação de Largura de Pulso*).* Esses sinais são utilizados muitas vezes para simular variações de tensões através do chaveamento entre os estados 1 e 0. Uma das aplicações por exemplo é para variar a velocidade de um motor de corrente continua controlado por um *Arduino*. 6 das 14 portas digitais do Arduino UNO podem ser utilizadas como emissoras de sinais *PWM*, sendo elas identificadas com um traço em cima do número do pino.



6. Já os **Pinos Analógicos**, não apenas podem atuar como portas digitais, mas também podem fazer uma leitura mais detalhada da tensão aplicada sobre ela (em contraste com a leitura binária das portas digitais). A tensão de entrada pode variar entre qualquer valor de 0 a 5 volts, porém ao ler o resultado observa-se que os valores obtidos são entre 0 e 1023, isso se deve à maneira que o *Arduino* processa as informações analógicas. Na *Arduino* UNO, estão disponíveis 6 portas analógicas.

7. A **Corrente por Pino I/O** equivale à corrente máxima que uma porta digital pode fornecer no estado *HIGH* (Ligada). No *Arduino* UNO essa corrente equivale a 40mA, e como a tensão de saída é de 5v, ao aplicar a fórmula: **Potência = Tensão x Corrente**, é possível analisar também que a potência máxima fornecida por uma porta digital é de 200 mW. É importante notar também que a corrente máxima total sendo fornecida por todas as portas digitais de uma placa Arduino UNO não pode ultrapassar 200mA, portanto uma potência total de 1 Watt.

8. A **Corrente no Pino 3.3v** é a corrente máxima que a alimentação de 3.3V do Arduino pode atingir sem ocasionar problemas. A corrente é baixa devido ao Regulador de Tensão utilizado para produzir esta tensão de saída. Como na Arduino UNO a corrente no pino 3.3V equivale a 50mA, isso significa que a potência máxima dela é de 165 mW.

9. A **Memória Flash** é a memória não volátil (que não perde as informações ao ser desligada), ela é utilizada para armazenar tanto o *Bootloader* (Programação interna que permite a utilização do Programador Arduino) da placa quanta o código carregado para a placa. O Arduino UNO conta com 32kB de Memória Flash, porém 512 bytes são reservados para o *Bootloader*.

10. Já a **SRAM** é a memória volátil da placa. É nela que são armazenadas as informações temporárias que serão apagadas ao desligar a placa, como variáveis. Percebe-se que a memória RAM da Arduino UNO é muito baixa (2kB). Para ter uma ideia, em uma variável do tipo *String* (que armazena informações do tipo de texto), cada caractere equivale a um *byte*, então uma mensagem como "*Hello World*" que equivale a Dez (10) *bytes* pode não parecer muito, mas rapidamente pode atingir o valor máximo de 2048 *bytes.* Na medida em que a memória SRAM começa a atingir seu valor máximo, a placa começa a cada vez mais a dar problemas imprevisíveis como corromper o programa desenvolvido, gerando erros em seus processos.

11. A última memória do Arduino UNO, a **EEPROM**, é um espaço designado para o desenvolvedor armazenar dados de longa duração. Como a Memória Flash, ela é não volátil, porém utiliza uma tecnologia diferente para escrever e apagar os dados. Na EEPROM, serão armazenadas informações de acordo com cada projeto, podendo por exemplo armazenar horários para acordar em um projeto de despertador. Na Arduino UNO, 1kB (1024 *bytes*) de memória EEPROM estão disponíveis.

12. Finalmente, a **Velocidade do *Clock*** é a frequência em que cada instrução física do processador opera por segundo, logo os 16MHz do Arduino UNO são equivalentes a 16 milhões de instruções por segundo. Porém, cada comando no código do Arduino equivale a várias instruções físicas, variando o tempo de execução de comando para comando.

3 - Dissecando o UNO

Devido ao caráter *Open-Source* do Arduino, a grande maioria dos dados do Arduino UNO foram divulgados, inclusive o seu esquemático (Figura 2). Na primeira vista, a figura é bastante intimidadora, porém após uma análise cuidadosa, se percebe toda a simplicidade do dispositivo. Nesse capítulo vamos descrever de maneira generalizada o circuito interno dessa placa.

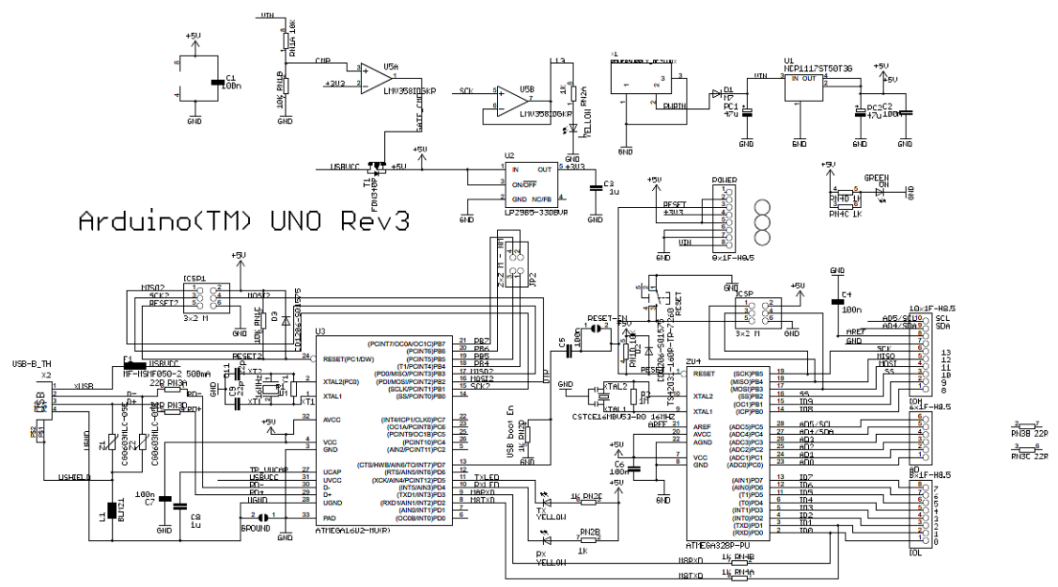


Figura 3 - Esquemático do Arduino Uno

Alimentação:

O Arduino permite duas fontes de alimentação, uma fonte USB e uma fonte externa, como representado na seguinte figura:

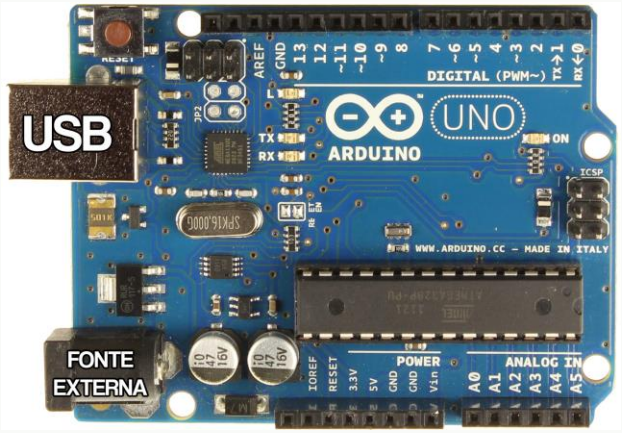


Figura 4 - Alimentação *Arduino* UNO

O *ATmega328P* (Microcontrolador do *Arduino* UNO) possui uma tensão de alimentação de 5V, que é a mesma tensão fornecida pelas portas USB de um computador, permitindo que o Arduino possa ser facilmente alimentado e programado em um computador qualquer. Para garantir a segurança da placa e do computador, foi adicionado um circuito de proteção à conexão USB, para que mesmo que o *Arduino* esteja conectado a um circuito que venha a agir de forma inesperada, ele não venha a queimar as portas USB do computador do usuário (normalmente limitado a 500mA).

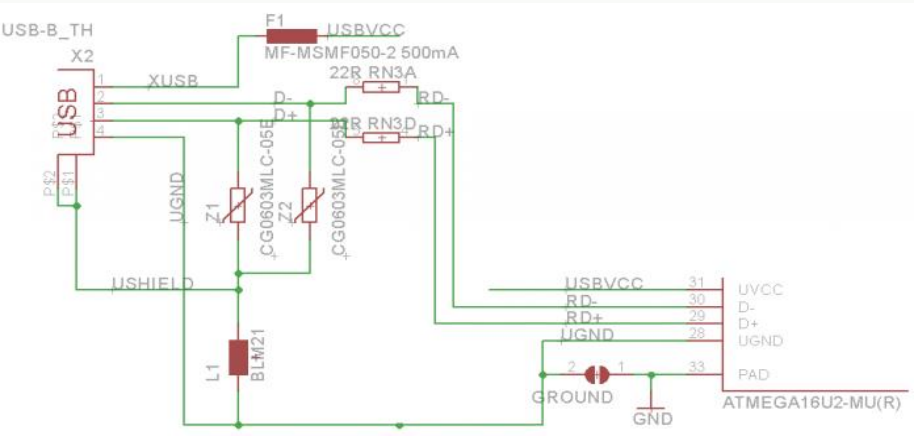


Figura 5 - Circuito de alimentação por USB

O problema de uma alimentação USB é a necessidade de manter um computador conectado ao *Arduino*, o que impede de usá-lo em soluções de maneira independente.. Para resolver esse problema, utiliza-se um circuito de regulação de tensão, para que possam ser usadas fontes externas (como baterias) para alimentar a placa. No caso do *Arduino* UNO o regulador utilizado é o NCP1117. Reguladores de Tensão, contudo, costumam liberar na forma de calor parte da potência fornecida ao circuito. No *Arduino* UNO esse gasto de energia é mínimo, devido ao seu regulador de tensão ser projetado para isso, garantindo uma maior efetividade.

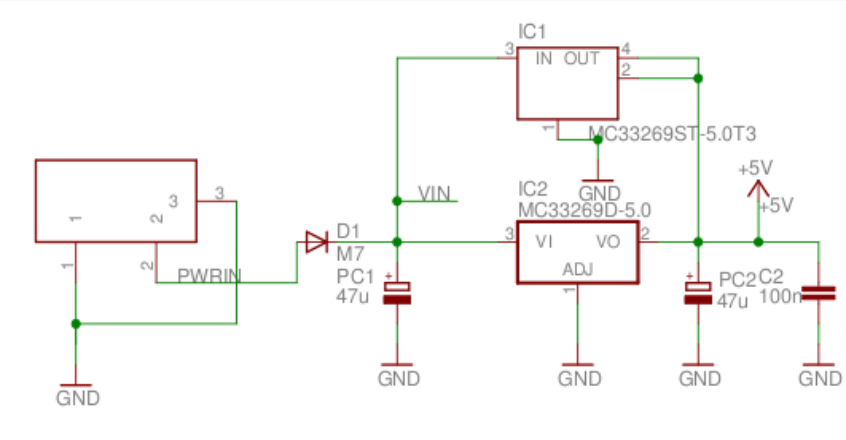


Figura 6 - Regulador de tensão

GPIO:

O que torna o Arduino algo tão útil é a sua capacidade de se comunicar com o mundo, e isso é feito através da GPIO (Entradas e Saídas de Uso Geral).

A GPIO pode ser dividida basicamente em duas partes: Sinal Digital e Sinal Analógico.

Os pinos de Sinal Digital são mais simples, eles detectam níveis de tensão de uma forma binária (SIM ou NÃO; 1 ou 0; 5v ou 1v), isso pode ser utilizado para LIGAR e DESLIGAR uma lâmpada, por exemplo.

Já os pinos analógicos apresentam uma certa diferença que os torna um pouco mais complicados, eles representam sinais que variam com muito mais possibilidades (temperatura, luminosidade, distância...). Isso os torna mais difíceis de se trabalhar pois o mundo da eletrônica é feito através apenas de informações digitais. Para se realizar a leitura de um sinal analógico é necessário transformá-lo em um sinal digital, isso é feito por um conversor implementado internamente ao circuito do Microcontrolador do Arduino.

Programação:

Para programar o *Arduino* é necessário que se faça uma comunicação entre as portas USB presente no computador e o Microcontrolador do Arduino. Contudo, eles não apresentam o mesmo protocolo de comunicação, é como se uma falasse em português e a outra em mandarim, o que torna necessária a atuação de um sistema de tradução de sinal para que seja realizada essa comunicação, no *Arduino* UNO essa tradução é mediada por um Microcontrolador denominado ATmega8U2.

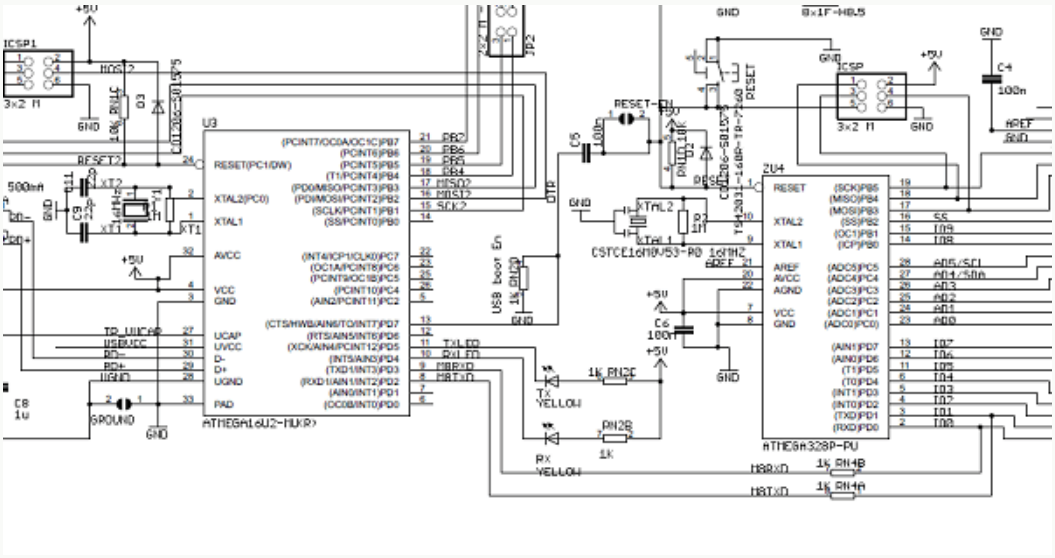


Figura 8 - Circuito de Comunicação

Um conhecimento pouco conhecido…

Alguns usuários do *Arduino* não possuem conhecimento de uma das técnicas mais úteis dele: o uso de interrupções de Microcontrolador. Estas interrupções permitem que o usuário realize operações a partir de eventos. Por exemplo, em um determinado projeto, quando um sensor de presença detectar que uma pessoa chegou ao ambiente, é preciso mostrar em um display uma mensagem de boas-vindas para a pessoa. Porém, é necessário durante isso realizar outra tarefa, como tocar uma música. Para isso, pode-se utilizar uma interrupção para só realizar a tarefa de exibir a mensagem no display no momento que a pessoa chega.

As interrupções podem ser de tempo, para realizar tarefas baseadas em tempo (como apitar um alarme de meia em meia hora), ou externas, para tarefas acionadas por algum agente fora da Arduino (como o exemplo dado do sensor de presença).

4 - Dividir para Conquistar

Agora que entendemos bastante do funcionamento da placa Arduino mais conhecida, podemos perceber que esses dispositivos podem facilmente ser divididos em vários blocos, e assim como diria o Imperador César, “vamos dividir para poder conquistar”.

Os grandes blocos são:

- O Microcontrolador
- A Alimentação
- A Programação
- A Área de Trabalho
- Componentes Extras
- Funcionais

Cada bloco representa uma grande parte do que significa ser uma placa Arduino. A modificação de cada um deles de certa maneira pode construir qualquer placa compatível com Arduino que já exista ou que possa existir. Por exemplo, se for trocado o Microcontrolador por um com mais pinos digitais, permitir que a alimentação dele seja feita por uma bateria recarregável, permitir a programação do Arduino via *Bluetooth* e montá-lo sobre uma Placa de Circuito Impresso própria (onde o circuito é construído e seus componentes são posicionados e soldados), a placa desenvolvida é completamente diferente do *Arduino* UNO original.

Para entender melhor esse conceito do Arduino dividido em blocos, vamos analisar uma versão da placa feita pela própria empresa Arduino e outra criada pela comunidade, sempre levando o Arduino UNO como um referencial.

1. **O Arduino Pro Mini 5V**

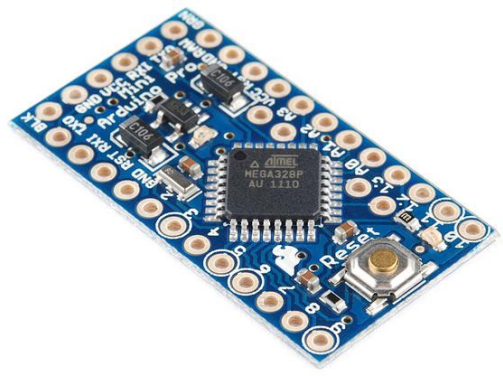


Figura 9 - Arduino Pro Mini

Primeiramente, deve-se entender que a finalidade dessa versão produzida pela Arduino foi construir uma placa extremamente pequena e de extremo baixo custo sem comprometer com a funcionalidade.

- O **Microcontrolador** praticamente não sofreu nenhuma mudança, sendo ainda o Atmega328, porém agora em uma versão SMD (Montado na Superfície), tornando-o menor. Ou seja, foi mantido o número de pinos digitais e analógicos, a frequência de *clock*, os valores de todas as memórias e as correntes e tensões.

- Na **Alimentação** percebe-se que foi removido o conector J2 de fontes de tensão e também a entrada USB. Sendo agora permitida apenas a alimentação feita diretamente pelo pino intitulado "*Raw".*

- A **Programação** foi altamente modificada, visto que a placa não possui mais um programador interno nem uma entrada USB. A programação agora é feita apenas através de um programador FTDI externo (que realiza a tradução da informação do computador para que o *Arduino* entenda) conectado na placa através de pinos localizados nas laterais da placa.

- A **Área de Trabalho**, ou seja, o local em que todo o circuito foi montado, foi alterado para que se a placa fosse menor tamanho possível. Portanto a Placa de Circuito Impresso desenvolvida com várias camadas permitiu compactar ao máximo os componentes na placa, sendo agora alguns soldados até na parte inferior da placa.

- Os **Componentes Extras** são dispositivos, sensores ou componentes que não influenciam diretamente na funcionalidade da placa Arduino, mas garantem funcionalidades extras que se deseja adicionar à ela. Na placa Arduino Pro Mini, não foi adicionado nenhum componente extra.

- Os **Funcionais** são todas as peças necessárias para que os blocos funcionem corretamente. Eles variam de acordo com as modificações feitas. Um Microcontrolador que trabalhe à uma frequência de 8MHz necessitará a modificação do Cristal Oscilador para um de 8MHz, por exemplo. No caso do Arduino Pro Mini, a única grande alteração dos componentes extras foi a preferência por suas versões SMD, para que o tamanho da placa fosse o menor possível.

2. O *Arduboy*

O *Arduboy* é uma placa compatível com *Arduino* feito pela comunidade com o objetivo de ser um console de *videogame* portátil de baixo custo e com jogos programáveis no Arduino ou baixados da comunidade.

Figura 10 - Arduboy

- O **Microcontrolador** usado nele é o ATMega32U4, que será explicado mais detalhadamente no próximo capítulo. Em suma, ele possui um pouco mais de pinos digitais, pinos PWM, e memória (comparado ao ATMega328 do Arduino UNO) e dispensa o uso do chip FTDI para realizar a tradução de protocolo na sua programação.

- Sua **Alimentação** não só pode ser feita através de um cabo *micro*-USB, mas também através de uma bateria recarregável, garantindo que ele fique ligado mesmo sem usar algum cabo.

- Já na **Programação** pouco se modificou em relação ao Arduino UNO, sendo ainda programado através de um cabo USB. As mudanças são apenas no fato da entrada de informação ser menor (*micro*-USB) e do Microcontrolador já possuir um programador interno.

- Com o objetivo do dispositivo ser do tamanho de um cartão de crédito, a **Área de Trabalho** foi feita com uma Placa de Circuito Impresso de várias camadas, nota-se também o uso da cor branca nela por uma questão visual.

- Os **Componentes Extras** são o grande diferencial dessa placa que conta com 6 botões, um *Speaker* Piezo Elétrico (emissor de áudio), e um Display OLED para que juntos garantam que a placa funcione exatamente como um console de video-game portátil.

- Os **Funcionais** dessa placa mudaram especificamente para acomodar os Componentes Extras adicionados e a bateria com seu circuito para carregá-la.

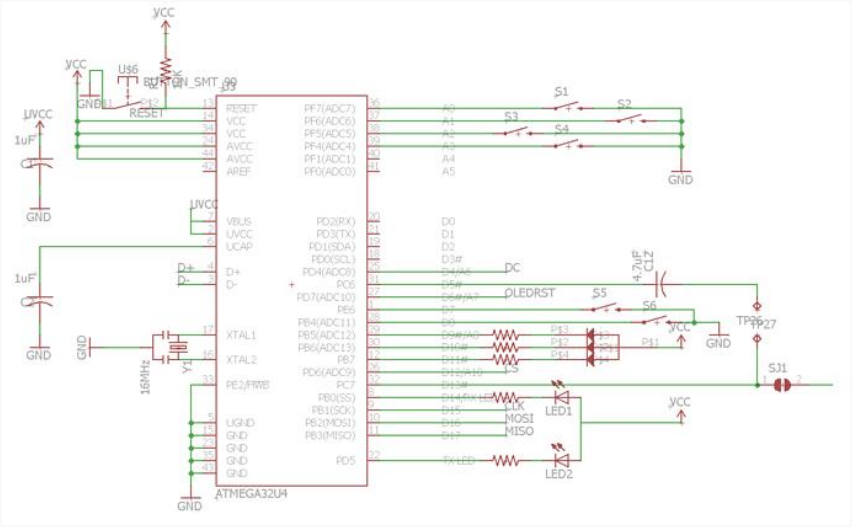


Figura 11 - Circuito Arduboy

5 - Passeando Pelos Blocos

Agora que entendemos o que é cada bloco, podemos passear por eles observando possíveis mudanças e possibilidades, analisando as suas principais diferenças e verificando as maneiras dessas modificações serem implementadas.

1. **O Bloco Microcontrolador.**

   Boa parte das placas Arduino utilizam Microcontroladores AVR da *Atmel,* como no caso da Arduino Uno que utiliza um **ATmega328P.** Neste tópico serão abordados os Microcontroladores utilizados por outras placas Arduino, bem como Microcontroladores menos conhecidos que podem também ser inseridos.

   - **ATmega32U4**

     É o Microcontrolador utilizado em exemplos como o ***Arduino Micro*** e o ***Arduboy***. Como grande vantagem em relação ao ATmega328 é que dispensa a utilização de um chip FTDI ou de algum outro Microcontrolador programado como tradutor USB para Serial, possuindo a possibilidade de realizar gravação nativa no chip. Sua arquitetura é semelhante ao ATmega328, possuindo um *Clock* máximo de 16MHz, contra os 20 MHz máximo do ATmega328.

     Como vantagem o ATmega32U4 possui mais pinos de I/O (26 pinos contra os 23 do ATmega328), porém possui menos interrupções externas (13 comparados aos 24 apresentados pelo ATmega328). O ATmega32U4 também apresenta uma memória SRAM maior que o ATmega328 (2.5kB contra apenas 2kB) além de possuir 8 pinos com suporte PWM contra somente 6 suportados pelo ATMega328.

     Em suma o **ATmega32U4** se mostra uma escolha interessante na hora de modificar o seu Arduino, dispensando a necessidade de um gravador e apresentando alguns pinos a mais que o ATmega328.

   - **ATmega2560**

     É o Microcontrolador utilizado na ***Arduino Mega* 2560**, ele já vem com um gravador programado de fábrica, o que permite o upload de novos programas sem precisar de um CHIP FTDI ou outro Microcontrolador.

     Porém ele não é capaz de criar uma porta COM virtual (necessária para ser reconhecida pelo sistema operacional Windows). Enquanto que o OSX e o Linux reconhecem a conexão como uma COM automaticamente, a solução adotada pelo time Arduino para que a placa MEGA 2560 pudesse ser reconhecida no Windows foi utilizar um

Microcontrolador mais simples, o ATmega16u2, responsável apenas por criar a virtualização da porta COM.

O ATmega2560 possui 256kB de memória flash, o que permite programas mais pesados que os Microcontroladores apresentados anteriormente, tendo 86 pinos de I/O, 32 interrupções externas. Possui também 16 pinos analógicos e 15 pinos PWM e uma memória SRAM de incríveis 8kB.

É uma escolha interessante para projetos de alta demanda de periféricos, sendo um Microcontrolador com uma grande quantidade de pinos, porém pode se mostrar um desnecessariamente complexo para determinados projetos.

- **Série *ATtiny***

  É o Microcontrolador utilizado na ***Arduino Gemma***, possuindo apenas 8 pinos, sendo 6 deles pinos de I/O, tendo suporte para PWM nos 6 pinos e suporte para leitura analógica em 4 deles. 8kB de memória flash e suporte para cristal de até 20 MHz. É uma excelente escolha para aplicações de baixa complexidade que não necessitem mais do que seis pinos.

São diversas possibilidades para a substituição do Microcontrolador, a tabela a seguir apresenta as possibilidades mais interessantes para substituições e fontes ensinando como realizar essas substituições.

Microcontrolador	Característica	Como usar como Arduino
ATtiny25/45/85/24/44/84	Menores e mais simples.	https://github.com/damellis/attiny/
ATmega1284/P, ATmega644/P/PA/A, ATmega324P/PA/A, ATmega164P/PA/A, ATmega32, ATmega16, ATmega8535	Diferentes opções de *ATmega* com diversas configurações distintas.	https://github.com/MCUdude/MightyCore
ESP8266	Microcontrolador com Wi-Fi incluso.	https://github.com/esp8266/Arduino

ATmega 32U4	Elevada performance e programador interno.	http://electronut.in/bootloader-atmega32u4/
AtXMega	Baixo consumo e alta performance.	https://github.com/XMegaForArduino

2. O Bloco da Alimentação

A alimentação nas placas *Arduino* utiliza fontes externas de alimentação para 5V. Como dito antes, é utilizado um regulador de tensão (NCP1117) para garantir essa tensão nos pinos de alimentação do Microcontrolador no Arduino UNO. Aqui serão mostradas algumas opções de substituição para o Bloco de Alimentação.

- **LM7805**

  Esse regulador de tensão costuma ser o mais discutido no início da aprendizagem de eletrônica devido ao fato de ser um regulador de tensão que fornece os 5V necessários para a placa Arduino e devido ao seu fácil acesso.

- **MAX756 DC-DC *StepUp***

  Com o MAX756 é possível realizar um processo inverso aos reguladores apresentados acima, partindo de uma tensão de menor valor e elevá-la para um valor maior. Com ele é possível alimentar a sua placa Arduino utilizando uma tensão de 1,5 Volts, provenientes de uma única pilha AA ou de uma pequena célula fotovoltaica (para alimentar a placa através de energia solar!).

- **Não usar um!**

  Outra possibilidade é a de não utilizar um regulador de tensão, desde que não seja necessário realizar conversões de tensão, permitindo assim que a placa seja alimentada exclusivamente pela tensão de alimentação suportada pelo Microcontrolador escolhido. Esta é a opção escolhida pelo ***Arduino Lilypad***.

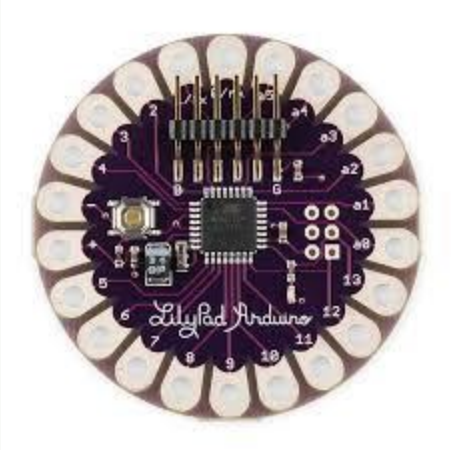


Figura 12 - *Arduino Lilypad*

3. O Bloco de Programação

A programação realizada no *Arduino* depende usualmente de um circuito que realize a conversão do sinal USB para um sinal UART (recebido pelo *Arduino*). Porém é possível fazer a programação de uma placa de diversas maneiras e com diversos tradutores, vamos ver alguns exemplos:

- **Bluetooth**: é possível programar o Arduino sem nenhum contato direto entre o computador e a placa através de uma comunicação Bluetooth, obtida facilmente através de um módulo como o HC-06, dispositivo que facilita o uso dessa tecnologia.

- ***Zigbee*:** ele um protocolo de comunicação por RF(Rádio Frequência) que é capaz de alcançar grandes distâncias, devido a sua capacidade de repetir sinais através de cada módulo da rede de comunicação, além de apresentar um baixo consumo de energia. A distância entre cada módulo pode ser de até uma centena de metros. É possível realizar a programação de forma similar à do Bluetooth, onde os dados trafegam através da rede *Zigbee* e ao final da linha haverá um programador como o utilizado no Arduino UNO para traduzir o protocolo *Zigbee* no padrão reconhecido pelo Microcontrolador.

- **WiFi:** É também possível programar por meio de WiFi, necessitando também de um módulo de recepção de sinal WiFi e tradução para o protocolo UART compreendido pelos Microcontroladores.

4. O Bloco de Componentes Extras

Agora que já descrevemos os blocos que são efetivamente necessários para as placas Arduino, vamos dedicar um tempo explicando como reinventá-las e personalizá-las para seus próprios objetivos.

É possível realizar algumas mudanças no projeto do *Arduino* para adicionar um componente qualquer (Display, LED's, Botões...) conectado alguns dos terminais digitais do Microcontrolador, garantindo novas funcionalidades já inclusas na placa, veja alguns circuitos que podem ser adicionados:

- **Comunicação por Rádio Frequência.**

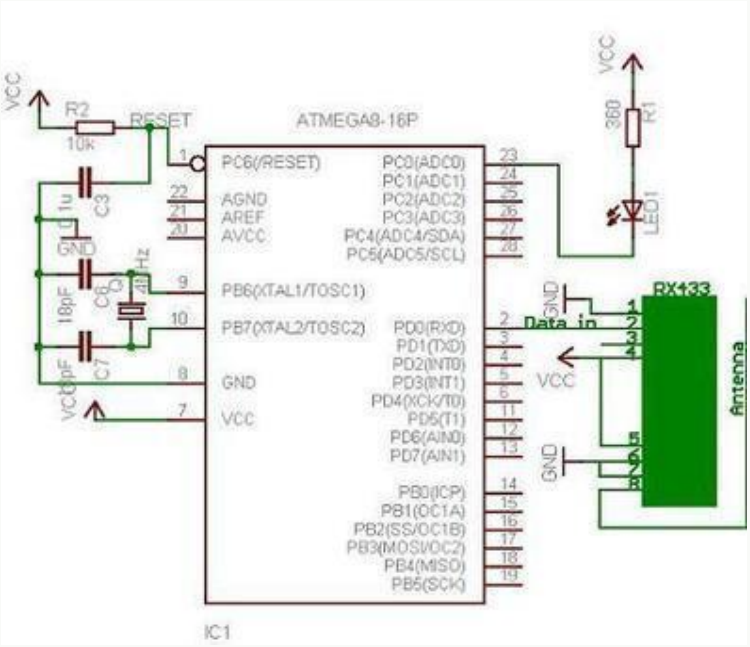


Figura 13 - Exemplo de circuito para comunicação via Rádio Frequência.

A comunicação por rádio frequência se dá pela emissão de sinais digitais codificados em ondas portadoras que são construídas por um circuito de interface, no caso do exemplo acima é uma placa externa que é responsável por essa codificação para que as ondas possam ser emitidas.

- **Controle de motores por meio de uma Ponte H.**

  Ponte H (ou *Half Bridge*) é um circuito de interface responsável por controlar motores DC que têm o seu sentido de rotação alterado com base na polaridade da tensão presente em seus terminais.

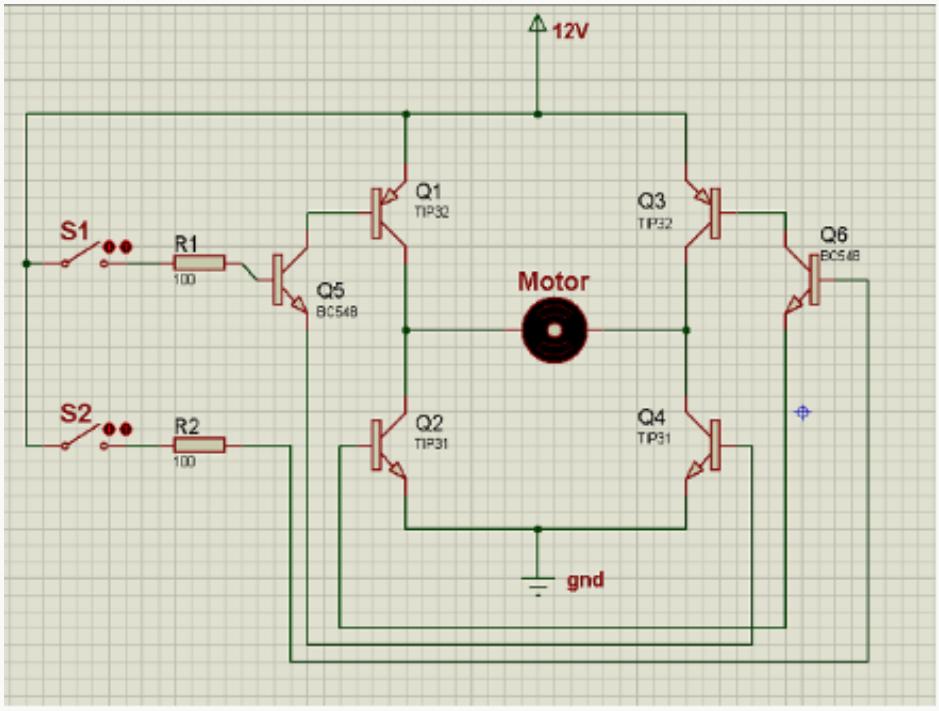


Figura 14 - Circuito Ponte H.

- **Nível de luminosidade do ambiente por LDR.**

  O *nome LDR é uma sigla para Light Dependent Resistor que significa Resistor dependente de luz, ou seja, a resistência dele depende da luminosidade que está presente na sua superfície, por isso, ao colocá-lo em série com outro resistor (Rl) de resistência constante, divisor de tensão, é possível ler o nível de tensão nesse Rl que vai variar de acordo com a luminosidade do ambiente.*

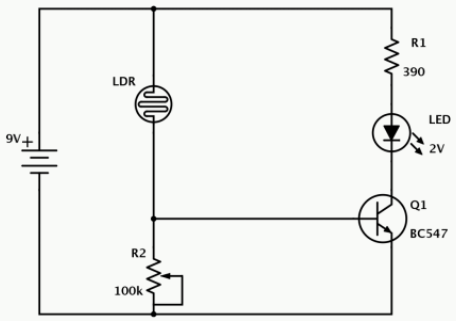


Figura 15 - Controle de LED a partir da luz no ambiente, quanto menos luz no ambiente, mais forte o LED brilha.

- **Interface Ethernet.**

  O protocolo Ethernet é o padrão utilizado em redes de alta velocidade e também utilizado para fazer a comunicação com a internet por meio de rede cabeada, é extremamente útil para ser utilizada em aplicações de IoT (Internet das Coisas).

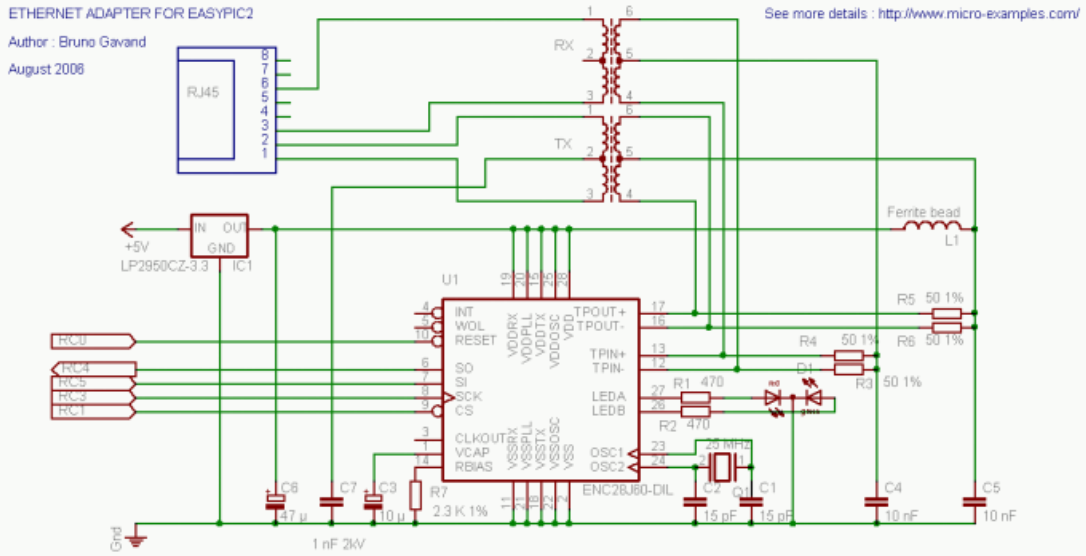


Figura 16 - Circuito de interface Ethernet.

5. O Bloco de Funcionais

Ao realizar estas mudanças e criar efetivamente uma nova placa Arduino, torna-se necessário durante o processo a adição e substituição de componentes funcionais (Capacitores, Resistores, Transistores, Diodos...) de acordo com o circuito de cada componente. Portanto torna-se sempre necessário a pesquisa do circuito de cada modificação para que ela funcione adequadamente.

6. Área de Trabalho

A área de trabalho é o local onde todos os componentes de cada bloco serão montados. Na maioria das placas eletrônicas, engenheiros e hobbistas constroem seus circuitos em placas de cobre onde são impressas ligações entre os componentes que serão soldados na placa, as chamadas Placas de Circuito Impresso (PCI).

O Design da placa de circuito impresso, que torna o Arduino o que ele é, deve ser realizado pelo projetista do circuito. Esse processo de design da placa é também chamado de Roteamento pelos engenheiros eletrônicos, pois, é um processo de definir rotas para os sinais eletrônicos que circulam no circuito projetado e que será impresso em uma placa real.

Para projetar as **PCBs** são utilizados programas chamados de *CAD softwares* (*CAD - Computer Aided-Design*), onde se desenha o circuito e se realiza o procedimento de roteamento da placa de circuito impresso.

Recomendamos o download de um desses softwares CAD além de começar a desenvolver os circuitos e seus esquemáticos.

Os projetistas da comunidade Arduino/Genuino utilizam programa Fritzing para desenvolver o layout de seus projetos, podendo ainda realizar a simulação de circuitos utilizando os códigos do *Arduino* no próprio programa.

Outro software recomendado é o *KiCAD*, CAD mais profissional do que o Fritzing, contudo, a curva de aprendizado dele é um pouco maior. Ambos os programas citados são gratuitos, o que facilita muito para um iniciante fazer o uso.

Alguns exemplos de CADs utilizados por profissionais são o *Eagle* e o *Proteus*, que são largamente utilizados por pessoas que necessitam realizar projetos mais elaborados de PCB. A plataforma Arduino utiliza o *Eagle* para desenvolvimento de seus projetos, tendo os seus arquivos de esquemáticos liberados para que toda a comunidade possa visualizar. A seguir uma imagem do layout da placa Arduino UNO no *Eagle*.

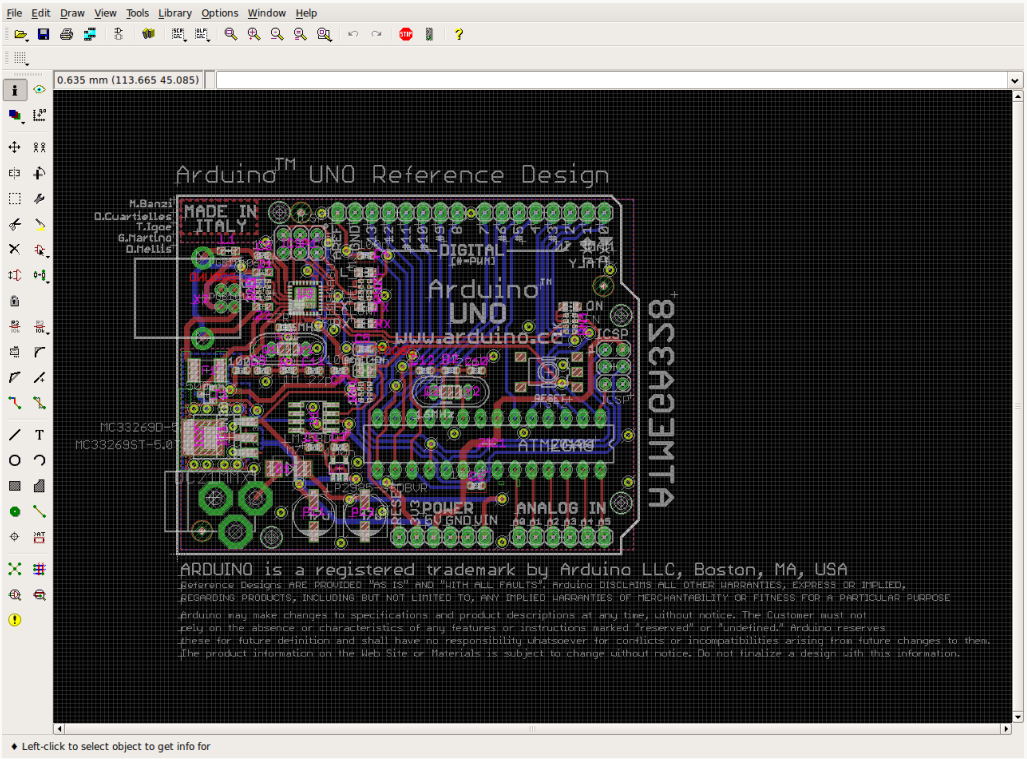


Figura 16 - Esquemático do arduino no Eagle

6- A Comunidade *Maker* e o *Creative Commons*

Ao concluir um projeto, deseja-se compartilha-lo com a comunidade e mostrar o resultado do seu trabalho. Porém, pode surgir a situação de alguém compartilhar o projeto que você fez como se fosse dela. Essa com certeza não é uma situação agradável. Pensando nisso, algumas pessoas fundaram licenças para uso público. A instituição que iremos abordar aqui é a organização denominada *Creative Commons*, ela é não governamental e sem fins lucrativos, fundada em 2001 por Lawrence Lessig, que é professor de direito em Harvard. Essa instituição está presente em diversos países do mundo, inclusive no Brasil, tendo seu site disponível aqui.

A licença *Creative Commons* pode apresentar algumas variações de acordo com a liberdade de uso que o desenvolvedor queira dar às demais pessoas quanto a seu trabalho.

Os símbolos presentes na licença são:

Creative Commons: Toda projeto com a licença *Creative Commons* deve apresentar esse símbolo.

Attribution License: Sempre que alguém for utilizar ou compartilhar o projeto com essa licença, essa pessoa deve dar os créditos ao autor original e cumprir as atribuições requeridas por ele. Por exemplo, o autor pode definir que sempre que compartilhar um projeto dele é necessário que haja um link para o seu *website.*

No Derivate License: Projetos com essa licença não permitem pessoas possam realizar alterações nele, ou seja, o trabalho só poderá ser utilizado e/ou compartilhado como o original, sem qualquer adaptação ou alteração.

***Share Allike License**:* Qualquer trabalho que seja criado a partir de um trabalho com essa licença deverá ser compartilhado com a mesma licença do trabalho original, por exemplo, quando você construir seu próprio *Arduino*, ele deve reconhecer a autoria da plataforma original e deverá ter o seu design divulgado para a comunidade com as mesmas licenças do *Arduino* original.

Non-Commercial License: O projeto que for protegido por essa licença e seus derivados não poderá ser utilizado para fins comerciais, ou seja, qualquer pessoa poderá distribuir e compartilhar o projeto, mas nunca poderá vender.

No site da organização CC existe uma área que serve para que as pessoas possam definir qual licença deverá ser utilizada em seus produtos que pode ser acessada clicando aqui, há também uma área para que as pessoas possam ler as licenças, que pode ser acessada clicando aqui. Um exemplo de uma Licença:

Figura 17 - Licença **CC BY-NC-ND**

7 - A Bananino

Agora que entendemos bem o funcionamento de um Arduino e a sua divisão em blocos, que tal desenvolver a nossa própria placa? Nessa parte do livro vamos detalhar o desenvolvimento da placa **Bananino**, desenvolvida pela Banana Digital.

Primeiramente, devemos definir o objetivo da placa a ser desenvolvida. Neste caso, queremos construir uma que seja ideal para qualquer pessoa iniciando o aprendizado de eletrônica e programação. Então o objetivo é facilitar tanto o processo de programação da placa quanto ao uso de componentes eletrônicos.

Agora, para atingir esse objetivo, o que é necessário?

Se for desenvolver uma placa que já possua nela o circuitos e componentes conhecidos, como sensores e motores, observa-se que irá ser alterado bastante o bloco de **Componentes Extras**. No caso, queremos que a alimentação da placa possa ser feita por uma bateria recarregável e que a programação seja feita por *Bluetooth*, para que a placa seja completamente sem fio.

Portanto, essas são as mudanças nos Blocos:

- **O Microcontrolador**: Permanece o Mesmo do Arduino UNO, o Atmega328p.

- **A Alimentação**: Feita por uma Bateria LIPO recarregável.

- **A Programação**: Programação feita por *Bluetooth* (pelo computador ou por um smartphone).

- **A Área de Trabalho**: Será desenvolvida uma Placa de Circuito Impresso utilizando o método de transfer com uma única camada para facilitar a produção artesanal.

- **Componentes Extras**: Serão adicionados vários sensores e componentes que possuem caráter educacional, que são muito utilizados na comunidade.

- **Funcionais**: Serão adicionados circuitos para os sensores e componentes adicionados além do circuito de alimentação.

1. Desenvolvendo a PCB

Para desenvolver o circuito da forma mais profissional possível e mantendo o fácil acesso, será utilizado o software CAD *OpenSource* KiCAD. O primeiro de todos os passos será definir os componentes a serem utilizados.

Serão adicionados:

- Sensor de temperatura
- Sensor de Luz
- LED's
- Potenciômetro
- Servo Motor

Esses componentes foram escolhidos com base no uso mais intenso deles ao longo de nossa experiência com o Arduino.

O primeiro passo a ser seguido no desenvolvimento de uma nova placa é analisar o circuito base do Arduino que mais se encaixa ao que se está fazendo. Como o objetivo é uma placa para usuários iniciantes que farão uso de alguns dos componentes mais utilizados, a base que escolhemos foi o circuito da placa Arduino UNO, que está disponível para download aqui. Se o objetivo fosse construir uma placa *Arduino* para ser utilizada como *Wearable* (Dispositivo feito para se vestir) teríamos escolhido algum *Arduino* que já existe com esse objetivo, como a *Arduino Lilypad* mencionado anteriormente.

2. Desenvolvendo os blocos

Conforme a metodologia de análise adotada ao longo do livro para analisar o Arduino UNO, teremos que dissecar nosso projeto em pequenas partes a partir do circuito original. Uma boa prática é sempre começar com o bloco do Microcontrolador, pois, caso esse bloco seja modificado, todo o resto do circuito também será, afinal, o sistema inteiro é controlado por esse pequeno bloco, que é o coração das placas baseadas em Arduino. Em nosso caso, não iremos alterar o Microcontrolador, o que reflete na escolha da placa fundamental de toda a cultura baseada em Arduino, pois, o Atmega328P cumpre bem a necessidade que teremos.

Em seguida temos que analisar os demais blocos, o que iremos modificar é a retirada da GPIO para o usuário, visto que o objetivo da placa é ser completamente autossuficiente, ou seja, sem a necessidade de conectar nenhum equipamento extra a ela para que sejam realizados os testes.

Seguindo em frente, agora que realizamos as alterações básicas que estarão presentes na nossa nova placa podemos partir para a parte principal, os blocos de funcionalidades extras, de acordo com os componentes escolhidos para serem adicionados. Dividimos os extras da seguinte forma:

- Entradas (Botões)
- Saídas (LED's)
- Sensores
- Motores

Essas três divisões serão adicionadas com base nas bibliotecas presentes para o *Arduino* que fazem uso desses componentes e terão pinos dedicados a estas funcionalidades.

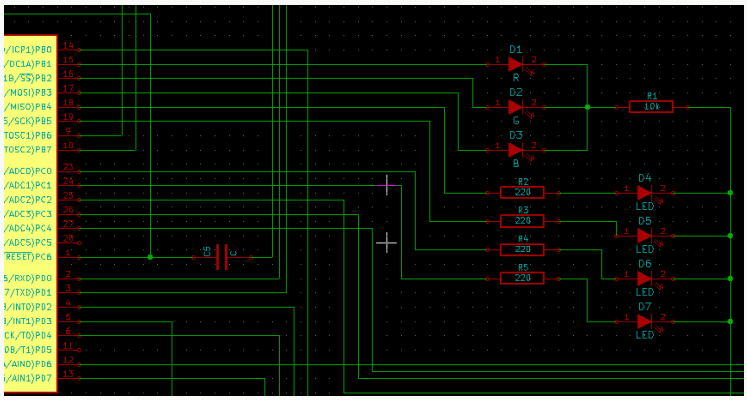


Figura 18 - Saídas com LED's.

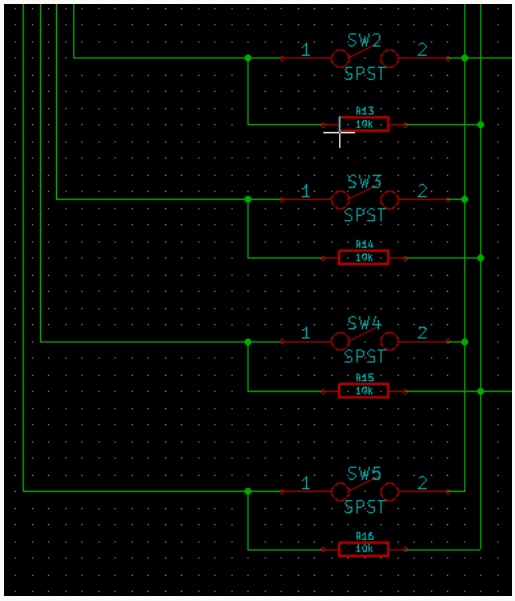


Figura 19 - Botões de entrada.

3. Desenvolvendo o layout

Para realizar o desenvolvimento do layout da placa é necessário um procedimento de design de circuito impresso, contudo, não é do escopo deste livro ensinar a desenvolver placas de circuito impresso. Mas é necessário que seja compreendido que o circuito que foi desenvolvido para a Bananino segue um grupo de regras e que foi desenvolvido visando usar uma placa de fenolite de 10cm x 10cm, pois, é um material que pode ser facilmente encontrado em meio ao comércio de eletrônicos. Além de ser utilizado uma única camada para roteamento das trilhas, devido a uma maior facilidade de realizar a construção de uma placa de face única.

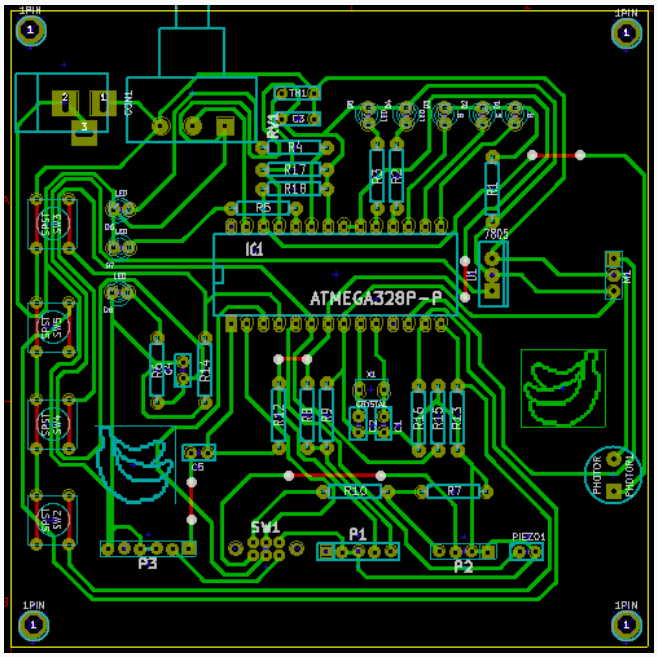


Imagem 19 - Layout desenvolvido para a Bananino.

4. Modelo 3D

O software utilizado foi escolhido também por sua capacidade de gerar um arquivo tridimensional mostrando como ficaria a placa pronta sem ser necessário gastar tempo e dinheiro para que seja testado isso. Após essa etapa a única coisa que resta é comprar os materiais e montar sua própria Bananino!

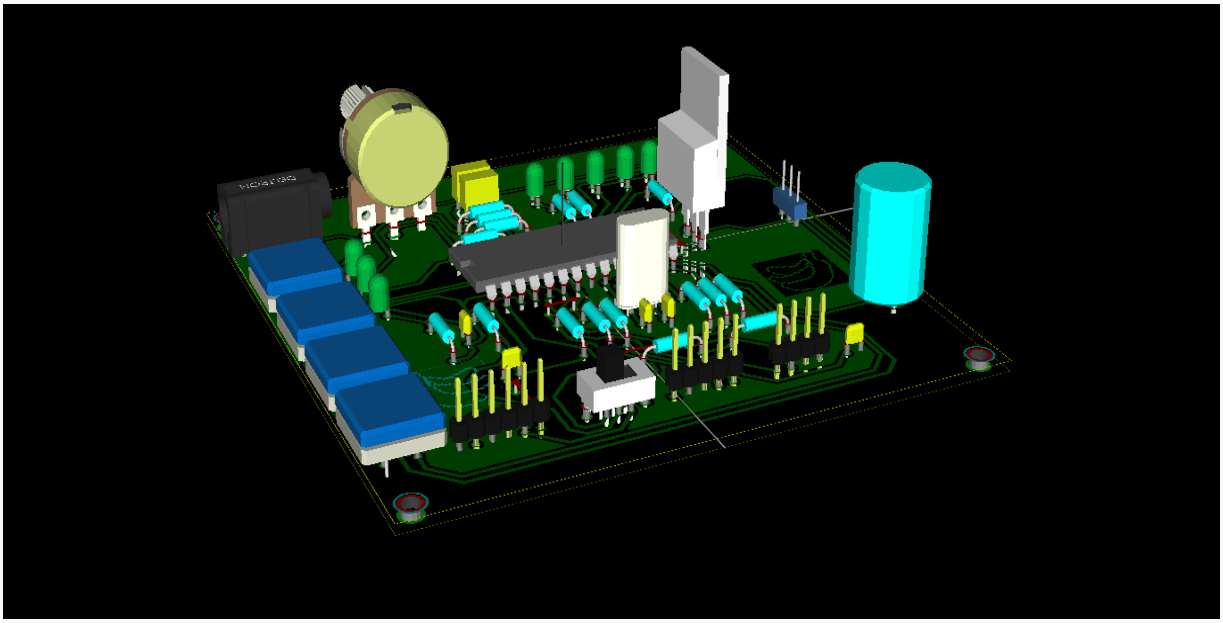


Imagem 20 - Simulação tridimensional da Bananino.

8 - Considerações Finais

Após a leitura desse livro é possível visualizar que o desenvolvimento de uma placa *Arduino* pode ser muito simples e caracteriza uma maneira muito eficaz de aprender o seu funcionamento interno. Seu desenvolvimento pode ser realizado por qualquer pessoa, principalmente pelo fato de ser uma plataforma *Open Source* e poder contar com a ajuda dos demais membros da comunidade para resolver diversas dificuldades que possa vir a ter, facilitando muito o desenvolvimento de um novo modelo que pode ajudar outras pessoas com seus objetivos.

Esperamos também que tenha sanado ao menos as dúvidas necessárias para começar a desenvolver um modelo diferenciado de *Arduino* para seus projetos no futuro ou até mesmo Shields (que seguem os mesmos conceitos de desenvolvimento). Embora a questão de desenvolvimento não tenha sido abordada extensivamente por não fazer parte do enfoque do livro, esperamos que as referências demonstradas sirvam no momento do desenvolvimento dos seus próximos esquemáticos e circuitos.

Nós da Banana Digital agradecemos à sua atenção para com nosso livro e esperamos poder ajudá-lo na medida do possível.

Você **APRENDEU**. Agora **FAÇA** e **MUDE O MUNDO!**