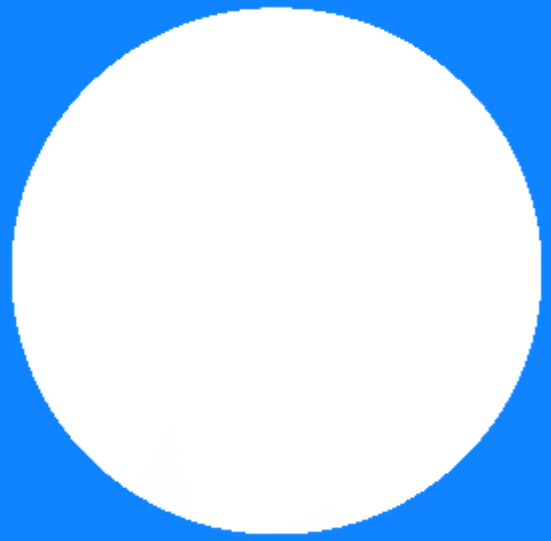


campus party

Como Fazer Seu Próprio Gameboy

Rafael de Moura Moreira

Durante o lero-lero inicial, baixem a IDE do Arduino!

<https://www.arduino.cc/en/Main/Software>

(ou digite “arduino ide” no Google)

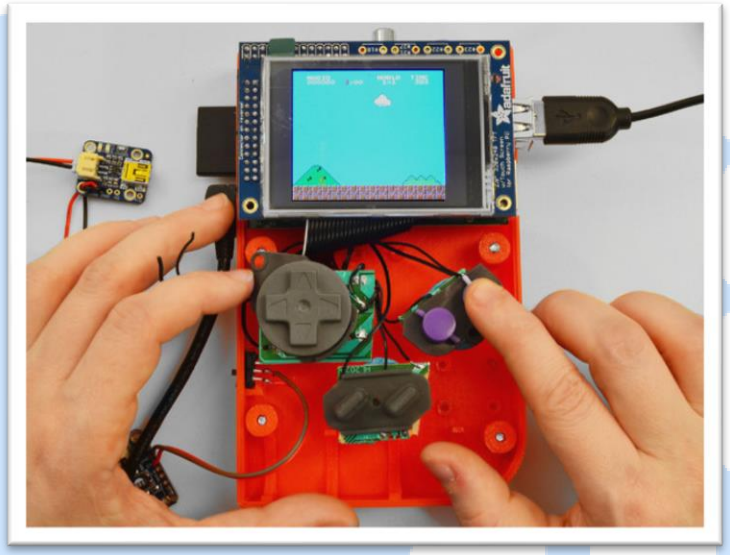
@author

Rafael de Moura Moreira <rafaelmmoreira@gmail.com>

Professor substituto na Universidade Federal de Itajubá (UNIFEI)
(Quase) Mestre em Ciência e Tecnologia da Computação pela UNIFEI
Engenheiro de Computação pela UNIFEI

@rafaelmmoreira

O que não veremos



Como fazer seu Gameboy com Raspberry Pi + impressão 3D

<https://www.thingiverse.com/thing:382485>

Como fazer seu Gameboy com Raspberry Pi + Gameboy quebrado

<https://superpiboy.wordpress.com>

O que não veremos

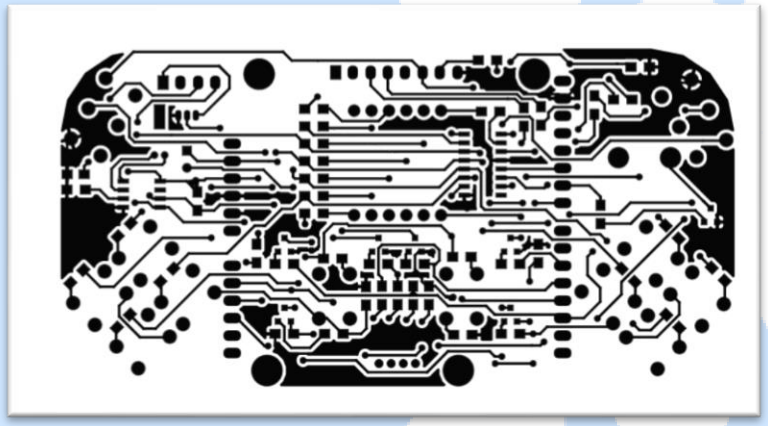
Como fazer seu Gameboy com Raspberry Pi + impressão 3D

<https://www.thingiverse.com/thing:382485>

Como fazer seu Gameboy com Raspberry Pi + Gameboy quebrado

<https://superpiboy.wordpress.com>

O que veremos

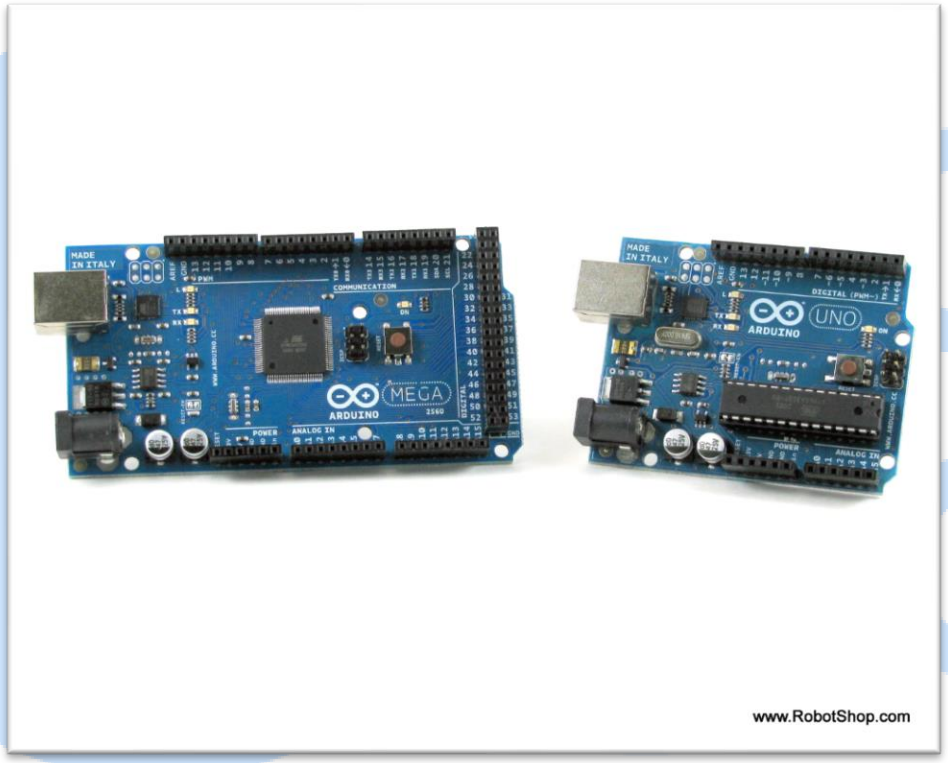


```
void loopUpdate(){
  if (key != mov){
    switch(key){
      case 'U':
        mov = key;
        if (linha > 0)
          linha--;
        break;
      case 'D':
        mov = key;
        if (linha < 3)
          linha++;
        break;
      case 'L':
        mov = key;
        if (coluna > 0)
          coluna--;
        break;
      case 'R':
        mov = key;
        if (coluna < 15)
          coluna++;
        break;
```

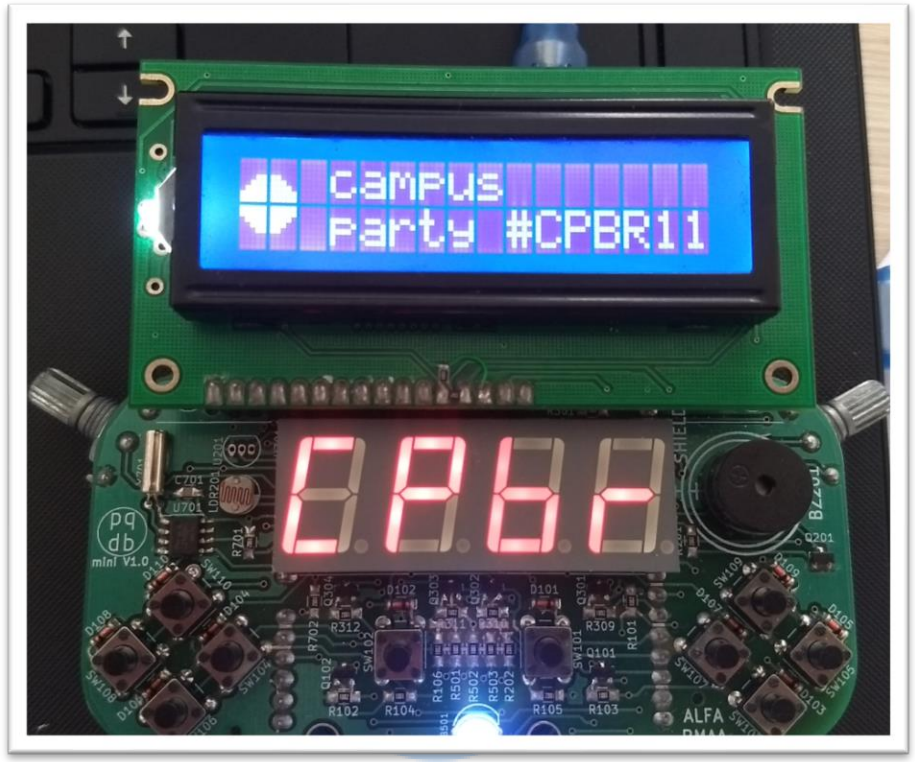
projeto de hardware

projeto de software

Conhecendo nosso equipamento



Arduino (Uno ou Mega)



Pão de Queijo Development Board (PQDB)

Arduino (Uno/Mega)

- Microcontrolador Atmega 16MHz, 2kB (Uno) / 8kB (Mega) SRAM
- Pinos de entrada e saída digital e analógica
- Hardware aberto e livre
- IDE grátis
- Barato ($)

Pão de Queijo Development Board (PQDB)

- Shield didático para estudo de embarcados
- Compatível com pinos do Arduino Uno (qualquer microcontrolador)
- Contém saída digital (LED RGB), analógica (buzzer), entradas digitais (botões), entradas analógicas (potenciômetro e sensores), LCD e display de 7 segmentos
- Hardware aberto e livre
- Desenvolvido em Minas Gerais, uai!

Projeto de hardware

O hardware da maioria dos sistemas computacionais pode ser subdividido da seguinte maneira:

ENTRADA	PROCESSAMENTO	SAÍDA
Dispositivos através dos quais fornecemos informações para o computador	CPU + memória Realiza as operações	Dispositivos através dos quais obtemos informações do computador

Projeto de hardware

Entradas: 8 botões (4 direções, A, B, Start, Select)

Processamento: Processador personalizado baseado no Z80 e Intel 8080 @ 4.19MHz, 8kB SRAM + 8kB VRAM

Saídas: LCD 160x144, auto-falantes

Projeto de hardware

Entradas: 8 botões (4 direções, A, B, Start, Select)

Processamento: Processador personalizado baseado no Z80 e Intel 8080 @ 4.19MHz, 8kB SRAM + 8kB VRAM

Saídas: LCD 160x144, auto-falantes

Projeto de hardware

Objetivos: que o meu console precisa fazer?

Viabilidade técnica: como eu faço meu console fazer o que eu quero?

Custos: até quanto eu posso cobrar pelo meu console?

Projeto de hardware

Objetivos: que o meu console precisa fazer?

É portátil ou doméstico? Precisa de alto desempenho gráfico? Usa internet? Aceita acessórios alternativos (guitarra do Guitar Hero, volante de corrida etc)? Deverá ser compatível com algum console já existente?

Projeto de hardware

Viabilidade técnica: como eu faço meu console fazer o que eu quero?

Qual processador aguentaria rodar o que imaginei? Quanta memória devo usar? Quais portas de entrada e saída usarei? O que o joystick deve ter? Como será o *case* para garantir dissipação adequada de calor? Caso seja portátil, qual o tamanho e formato adequado para o *case*? Caso seja portátil, qual bateria aguenta várias horas sem deixar o equipamento pesado ou superaquecido?

Projeto de hardware

Custos: até quanto eu posso cobrar pelo meu console?

Quanto meus clientes estão dispostos a pagar? Quanto meus concorrentes estão cobrando? Eu consigo ter margem de lucro cobrando esse preço por esse equipamento? Preciso fazer ajustes no projeto para caber no orçamento?

Projeto de hardware

Vamos definir o que precisamos para o nosso curso?

Projeto de hardware

Objetivos:

1) Queremos aprender a fazer algo estilo Gameboy, portanto deverá ser portátil e deverá rodar jogos retrô.
2) Temos pouco tempo de curso, então deverá ser fácil de montar e programar.

Projeto de hardware

Viabilidade técnica:

1) Para ser portátil e fácil de montar, é ideal usar uma plaquinha já pré-montada com microcontrolador.
2) Para ser fácil de programar, deve ser uma plaquinha com bom suporte a software e documentação fácil na internet.
3) Para rodar jogo retrô, não é necessário processador poderoso ou muita memória.

Projeto de hardware

Custo:

O mais barato possível! O projeto não é comercial, muitos de nós somos estudantes e é fácil queimar/danificar componentes :)

Projeto de hardware

Processador: Arduino Uno

- 8 bits, 16MHz, 2kB SRAM – suficiente para jogos leves, com pouco gráfico
- 20~40 reais, fácil de encontrar
- Grande disponibilidade de documentação, bibliotecas e exemplos na internet
- Usado para ensinar programação e eletrônica para iniciantes

Projeto de hardware

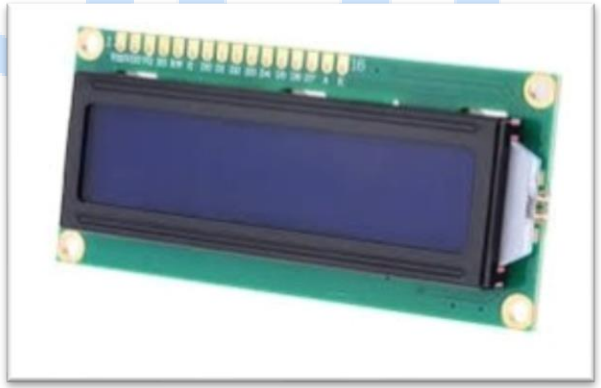
Entradas: 10 botões digitais

- Suficiente para fazer um layout de joystick estilo Super Nintendo
- Fácil de programar, retornam 2 valores diferentes (0 ou 1)
- Preço em centavos

Projeto de hardware

Saída: LCD 16x2

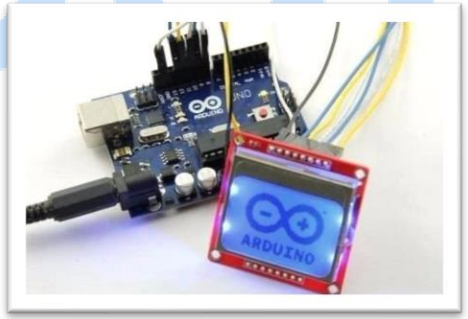
- Fácil de programar, os desenhos são representados como caracteres
- Gráficos estilo "*dot matrix*", como os do Gameboy
- Preço: 5 a 20 reais
- Desvantagem: resolução baixa, gráficos subdivididos em retângulos na tela



Projeto de hardware

Saída alternativa: LCD Nokia 5110 (a tela de celular Nokia antigo!)

- Tela gráfica verdadeira
- Gráficos estilo "*dot matrix*", como os do Gameboy
- Preço: na faixa dos 20 reais
- Relativamente fácil de programar, mas menos do que o LCD 16x2

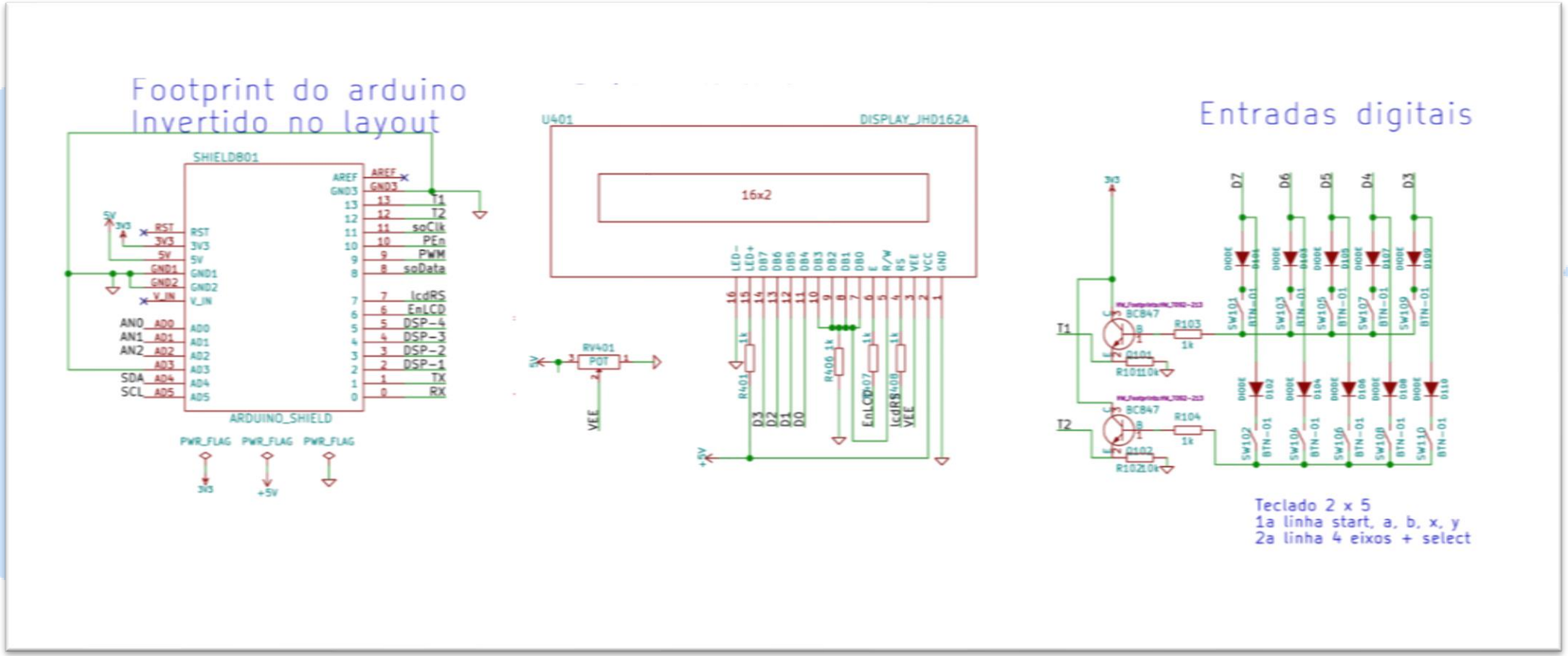


Projeto de hardware

Devido ao tempo limitado, neste curso usaremos o display LCD 16x2.

Ao final desta apresentação há links com documentação para quem quiser montar o circuito com o Nokia 5110 em casa!

Projeto de hardware



Esquemático

Interface Hardware-Software

É importante facilitar a vida dos desenvolvedores dos jogos.

Trabalhar diretamente com o hardware dificulta o desenvolvimento dos jogos, prolonga e encarece o projeto e pode resultar em jogos abaixo do ideal.

Interface Hardware-Software

"Desenvolvedores tiveram problemas para criar jogos para o Saturn. (...) sem experiência avançada em programação de jogos era difícil de desenvolver algo para ele porque seu hardware era muito complexo. Estima-se que 1 em 100 programadores realmente conseguiam usar todo o seu potencial. (...) e como desenvolvedores estavam com dificuldade para fazer os jogos, eles optaram pelo Playstation."

<http://www.dcshooters.co.uk/sega/saturn/saturn.php>

Interface Hardware-Software

Fabricantes de hardware geralmente disponibilizam camadas intermediárias de software que se comunica com o hardware, como sistemas operacionais, drivers de dispositivo e bibliotecas.

Essas camadas intermediárias fornecem aos programadores funções fáceis de utilizar, que não exigem conhecimento avançado do hardware, facilitando tanto o desenvolvimento quanto a portabilidade do código.

Interface Hardware-Software

```
key = kpReadKey();
```

Código para ler qual botão foi pressionado utilizando um driver pronto

```
for(int i = 0; i<5; i++){
  soWrite(1<<(i+3));
  if(digitalRead(KEYPAD_1_PIN)){
    bitSet(newRead,i);
  }
  if(digitalRead(KEYPAD_2_PIN)){
    bitSet(newRead,i+5);
  }
}
if (oldRead == newRead) {
    tempo--;
} else {
    tempo = 4;
    oldRead = newRead;
}
if (tempo == 0) {
    keys = oldRead;
}
```

Trecho de ~50% do código da leitura do botão no driver

Interface Hardware-Software

Como projetar um bom driver ou biblioteca de hardware:

1) Ler o datasheet do dispositivo e entender o que ele faz, qual o formato dos dados que ele recebe e qual o formato dos dados que ele entrega
2) Identificar quais recursos o usuário irá precisar usar
3) As funções deverão exigir e fornecer dados em formato amigável para o usuário – elas são responsáveis por colocar no formato adequado para o hardware, não o usuário!
4) Boa documentação: nomes intuitivos e boas explicações de como usar

Interface Hardware-Software

Para este curso, devido a limitações de tempo, não desenvolveremos os drivers – vamos utilizar os fornecidos pelos fabricantes da PQDB e partir para o desenvolvimento de um joguinho!

Interface Hardware-Software

Como ler qual botão foi pressionado:

A função kpReadKey() retorna um caractere informando qual tecla foi pressionada: (U)p, (D)own, (L)eft, (R)ight, (A), (B), (X), (Y), (S)elect ou (s)tart.

Interface Hardware-Software

Como desenhar na tela:

A função lcdCreateChar(posição, desenho) copia um desenho binário (8x5) em uma certa posição da memória do LCD.

É com ela que criaremos os gráficos do nosso joguinho. Posteriormente utilizaremos lcdChar(numero) para colocar na tela o desenho salvo na posição "numero" da memória.

Criando o nosso primeiro jogo

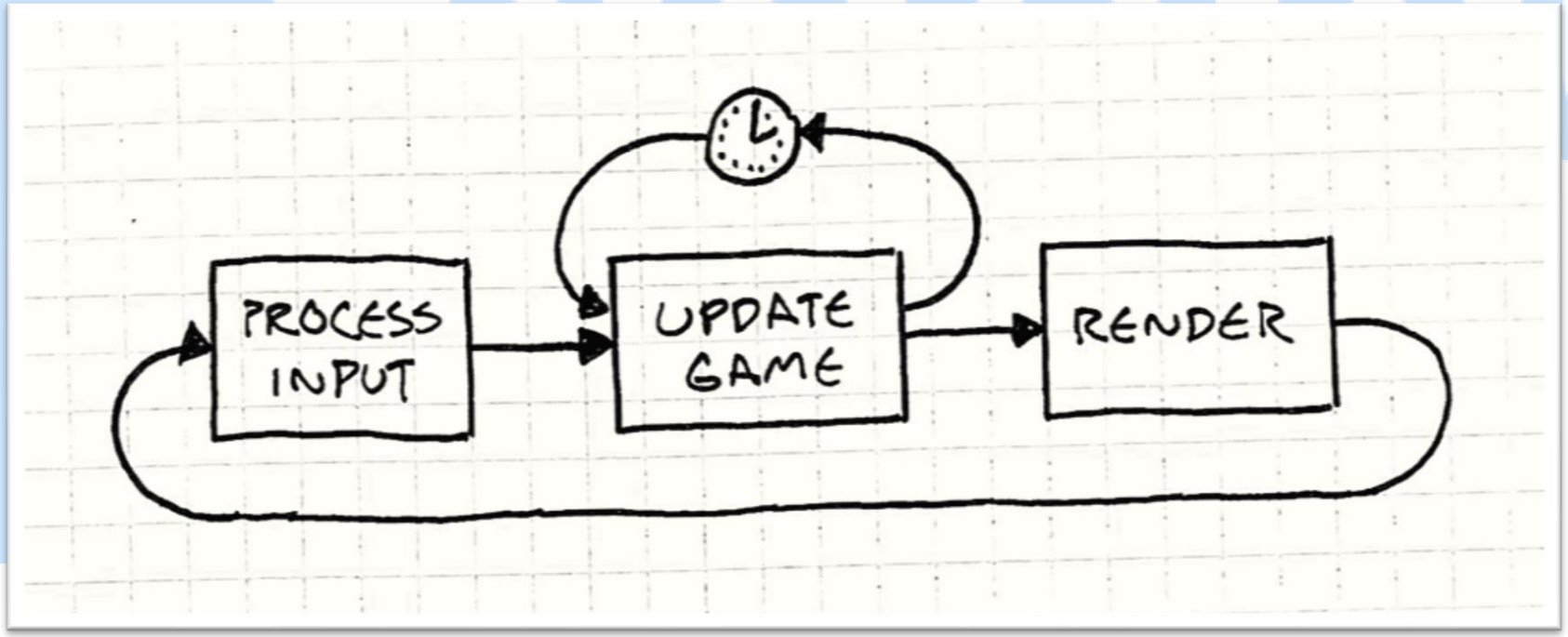
Um bom código não deve apenas funcionar bem – ele deve ser limpo, organizado e o mais legível possível.

Isso facilita o trabalho em grupo e o processo de debug.

Boas técnicas de programação incluem o uso de nomes intuitivos para funções e variáveis, o uso de comentários explicando o que cada trecho faz e o uso de padrões populares para cada tipo de programa.

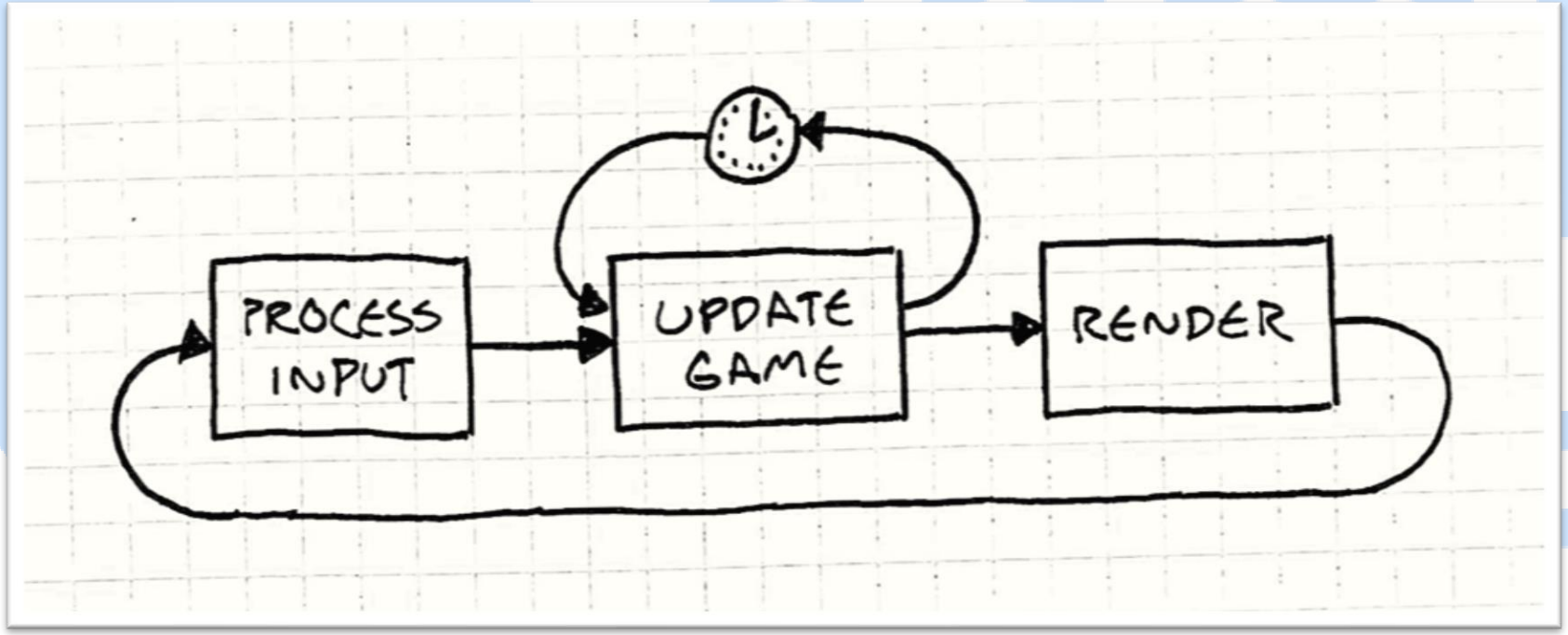
Criando o nosso primeiro jogo

Uma forma de dividir o processamento do jogo é em 3 etapas:



Criando o nosso primeiro jogo

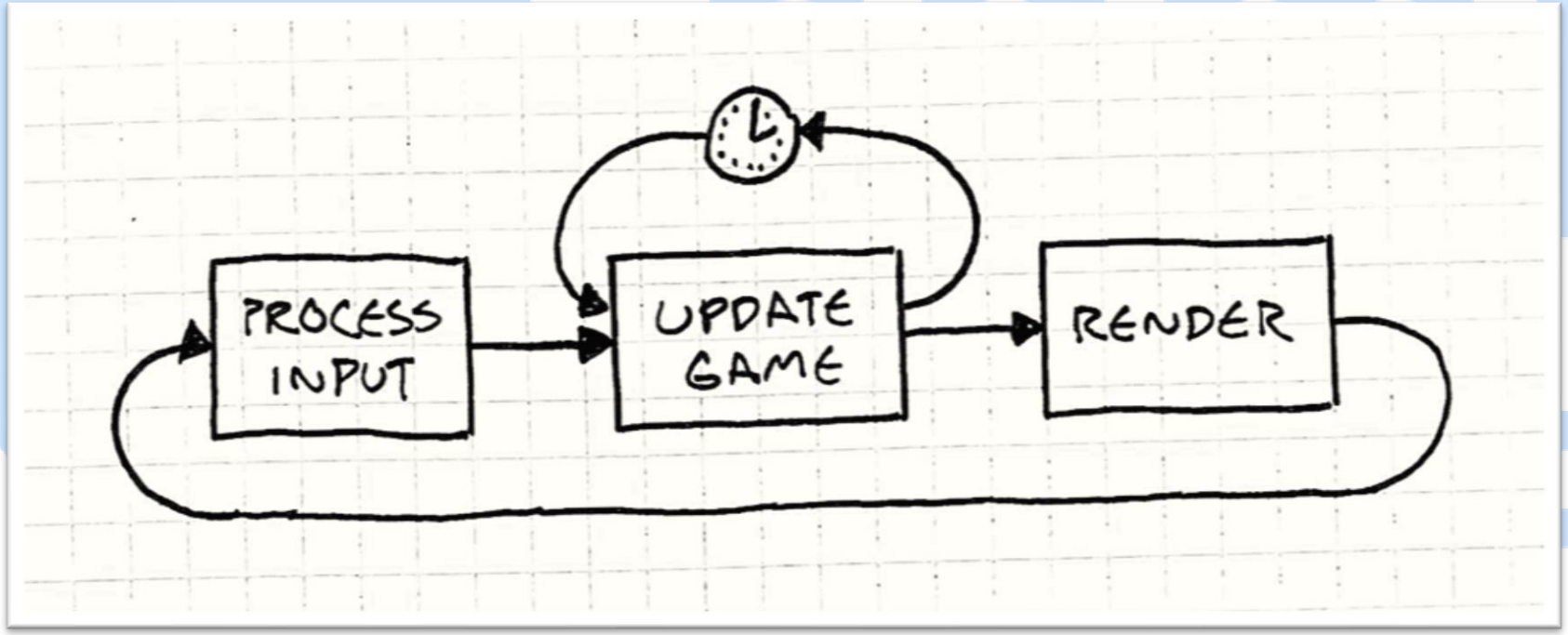
Uma forma de dividir o processamento do jogo é em 3 etapas:



Input: Processar as entradas (ex: usuário apertou direcional para esquerda)

Criando o nosso primeiro jogo

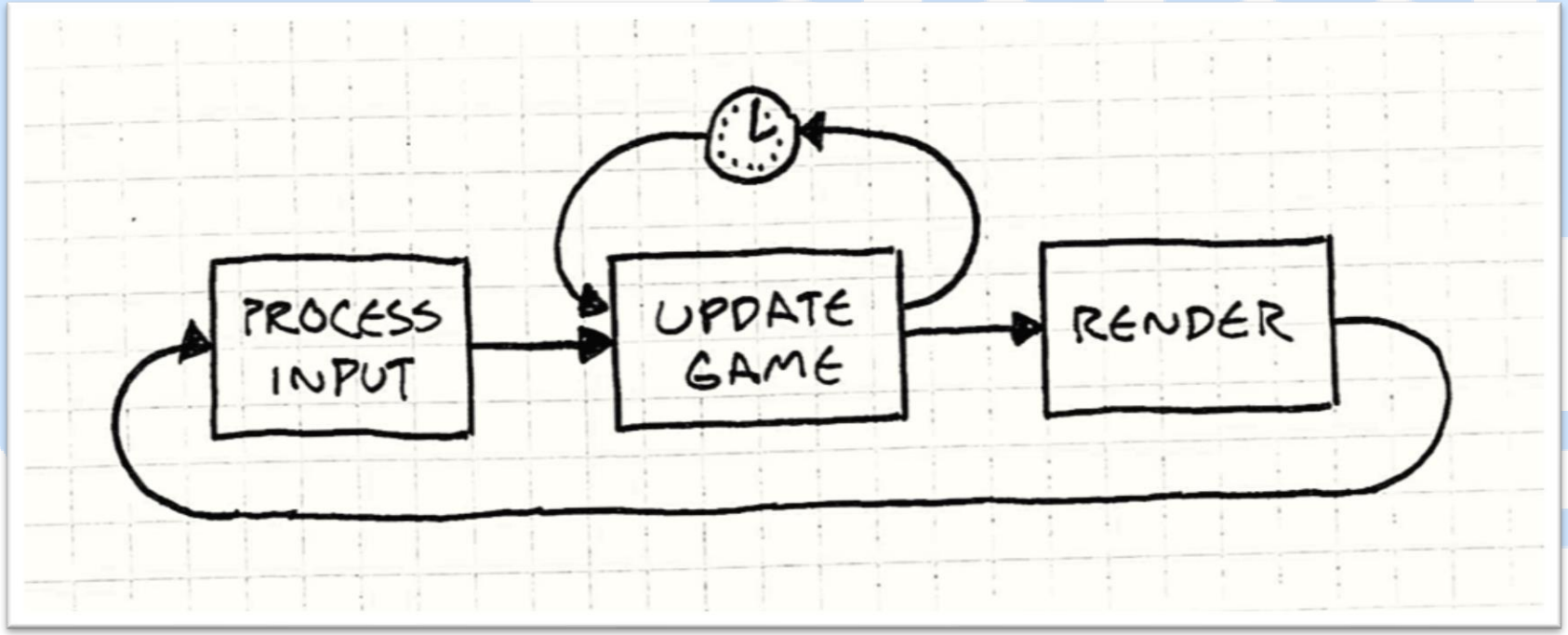
Uma forma de dividir o processamento do jogo é em 3 etapas:



Update: atualizar as informações do jogo (ex: posição do jogador no mapa, testar se houve colisão, gerar inimigos etc)

Criando o nosso primeiro jogo

Uma forma de dividir o processamento do jogo é em 3 etapas:



Render:
Atualizar a tela
do jogo

Criando o nosso primeiro jogo

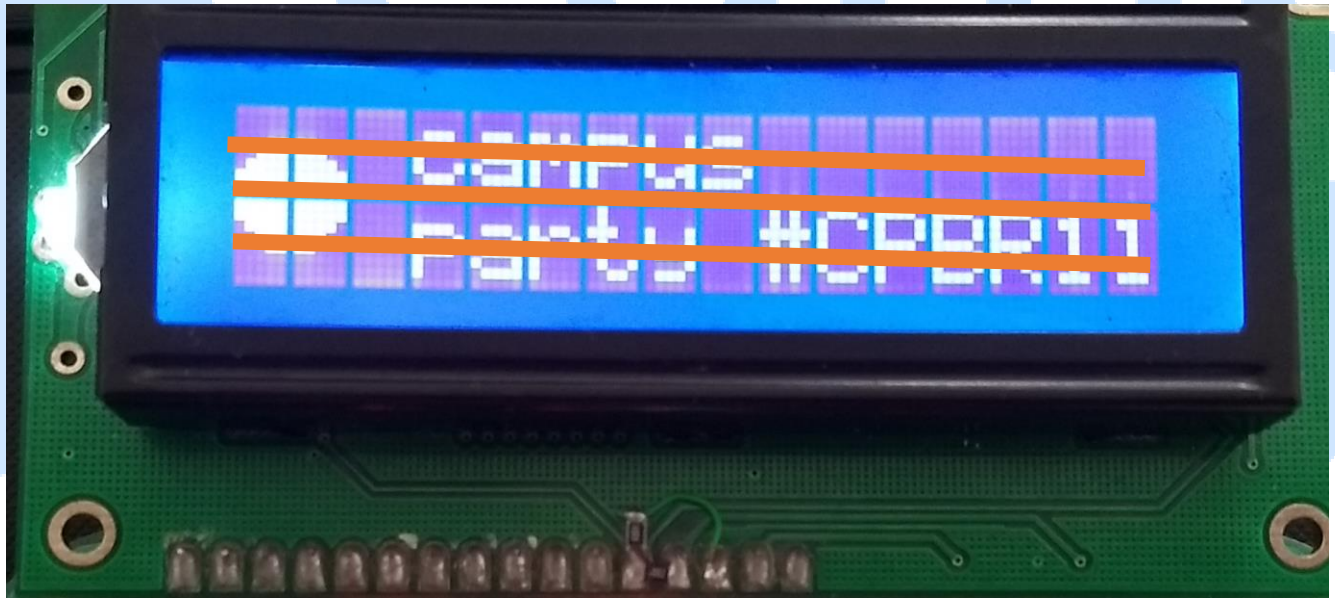
Proposta de joguinho: um jogo de carrinho estilo Enduro (Atari 2600, 1983) onde o jogador deve desviar de obstáculos (no original, outros carrinhos).

Vamos estudar a lógica do joguinho. O código-fonte está no GitHub:

<https://github.com/rafaelmmoreira>

Criando o nosso primeiro jogo

Vamos subdividir o LCD em 4 linhas. Ou seja, cada "caractere" dele corresponderá a 2 linhas. Assim teremos mais espaço para o jogo.

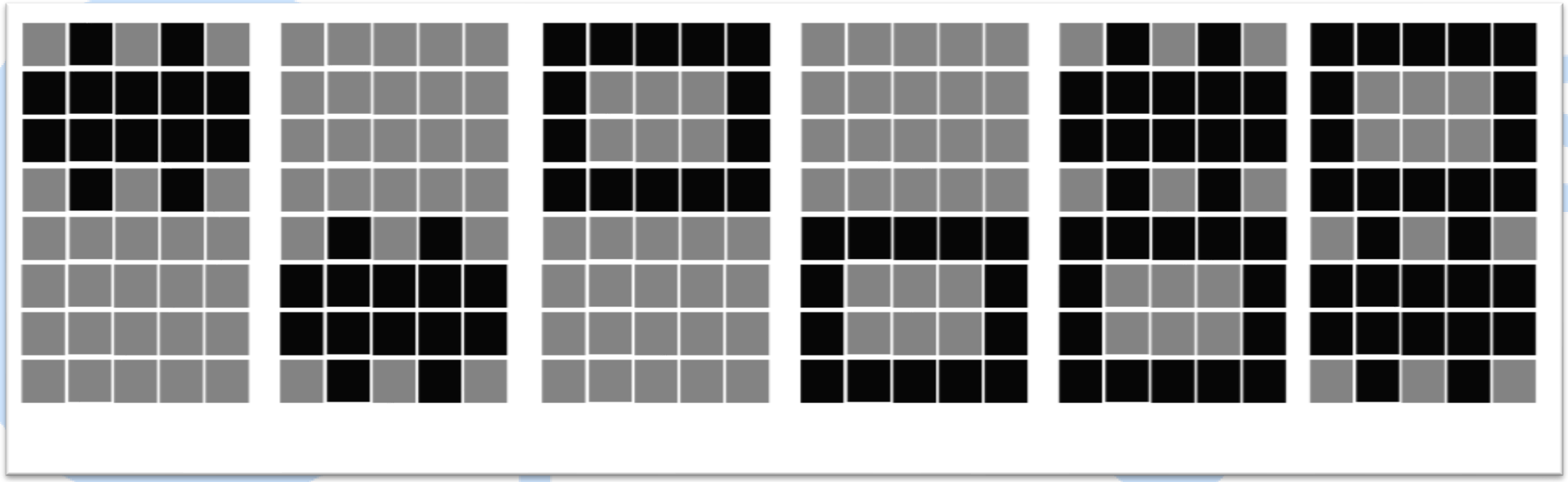


Criando o nosso primeiro jogo

Como cada desenho ocupa um caractere inteiro, e só temos 2 linhas de caracteres, teremos que fazer diferentes combinações de cada desenho:

1) Carrinho em cima do retângulo
2) Carrinho embaixo do retângulo
3) Obstáculo em cima do retângulo
4) Obstáculo embaixo do retângulo
5) Carrinho em cima e obstáculo embaixo
6) Carrinho embaixo e obstáculo em cima

Criando o nosso primeiro jogo



Cada quadradinho pintado corresponde a “1”, enquanto cada quadradinho em branco corresponde a “0”.

Criando o nosso primeiro jogo

Todo código de Arduino possui uma função setup(), que é executada uma vez ao início do programa, e uma função loop(), que é repetida enquanto o dispositivo estiver ligado.

Utilizaremos a setup() para fazer a inicialização do hardware e para salvar nossos desenhos na tela. A função loop() chamará, em sequência, cada uma das 3 funções que vimos antes: loop_input(), loop_update() e loop_render().

Criando o nosso primeiro jogo

loop_input():

Apenas lê qual tecla o usuário pressionou e a armazena. Caso nenhuma tecla tenha sido pressionada, armazena 0 (zero).

Note no código-fonte que a função possui um loop. Por que?

Criando o nosso primeiro jogo

loop_update():

Verifica qual tecla foi apertada e move a posição do carrinho no mapa (aumentar ou diminuir linhas e colunas).

Movimenta o carrinho para frente na estrada (desloca o mapa inteiro 1 casa para trás) e realiza um sorteio para saber se novos obstáculos aparecerão.

Testa se houve Game Over (posição do carrinho no mapa = obstáculo).

Criando o nosso primeiro jogo

loop_render():

Limpa a tela, posiciona o cursor na primeira posição e percorre a matriz desenhando seu conteúdo.

Antes de partir para a atividade prática, vamos dar os devidos créditos aos envolvidos!

Agradecimentos

Aos professores Rodrigo “Max” Almeida e “Carlão” Valério da UNIFEI e ao Thiago Lima do Embarcados pelo convite, pelo apoio e pelo equipamento!

À equipe da Campus Party pelo evento fantástico!

À minha namorada Brenda (essa que está ficando com vergonha agora) por todo o apoio e por me acompanhar na correria dos últimos dias!

E a todos vocês que estão passando sua noite de sábado estudando embarcados com a gente!

Referências e links úteis

- Projeto PQDB no GitHub – inclui drivers e toda a documentação de hardware: <https://github.com/projetopqdb/>
- Portal Embarcados – inclui vários tutoriais com Arduino e ensina a fazer a sua própria PQDB! <https://www.embarcados.com.br/>
- Game Programming Patterns – ótima referência para programação de jogos! <http://gameprogrammingpatterns.com/>
- Livro: Programação de Sistemas Embarcados – R. M. A. Almeida, C. H. V. Moraes, T. F. P. Seraphim – Ed. Elsevier - é daí que surgiu a PQDB. Um curso completo de programação de embarcados que não exige conhecimentos prévios. Parte do mais básico e chega em assuntos avançados.

DESAFIO HANDS-ON

Leia com atenção o código do jogo e entenda o que está acontecendo. Em seguida, implemente algo de novo nele.

DICA: pense em como sua proposta afeta cada uma das 3 partes do loop: input, update e render.

Caso ajude, estes slides estão disponíveis em: <https://www.slideshare.net/rafaelmmoreira>

Exemplos:

- Uma tela de Game Over
- Uma tela de Pause (caso o usuário aperte Start)
- Um míssil que explode uma caixinha
- Um meio de mostrar o placar (1 ponto por caixinha ultrapassada)
- Um meio de aumentar a velocidade conforme o jogo avança
- Modo 2 jogadores (P2 utiliza os botões A, B, X e Y como direcional)
- Outras ideias criativas!