

Sistemas Embarcados

Processos de Desenvolvimento Levantamento de requisitos de software e de hardware / Fluxograma

Tiago Piovesan Vendruscolo



Esta licença permite que outros remixem, adaptem e criem a partir do trabalho para fins não comerciais, desde que atribuam o devido crédito aos autores originais. [4.0 international](https://creativecommons.org/licenses/by-nc-nd/4.0/)

- Nem sempre o cliente sabe de forma clara o que ele necessita no seu projeto.
 - *Dificuldade de comunicação entre o usuário e o desenvolvedor.*
- Fazendo um correto levantamento de requisitos desde o início, facilita ao cliente entender se o projeto atenderá as suas expectativas, além de garantir uma maior segurança à equipe de desenvolvimento (reduzirá a probabilidade de ter de refazer parte do projeto, ou até o projeto inteiro).



O que cliente precisa?



Como o cliente explicou



Como o analista de negócios entendeu



Como o analista de sistemas planejou



Como o programador codificou



Como o comercial vendeu



O que foi instalado



Como foi cobrado



Como foi documentado

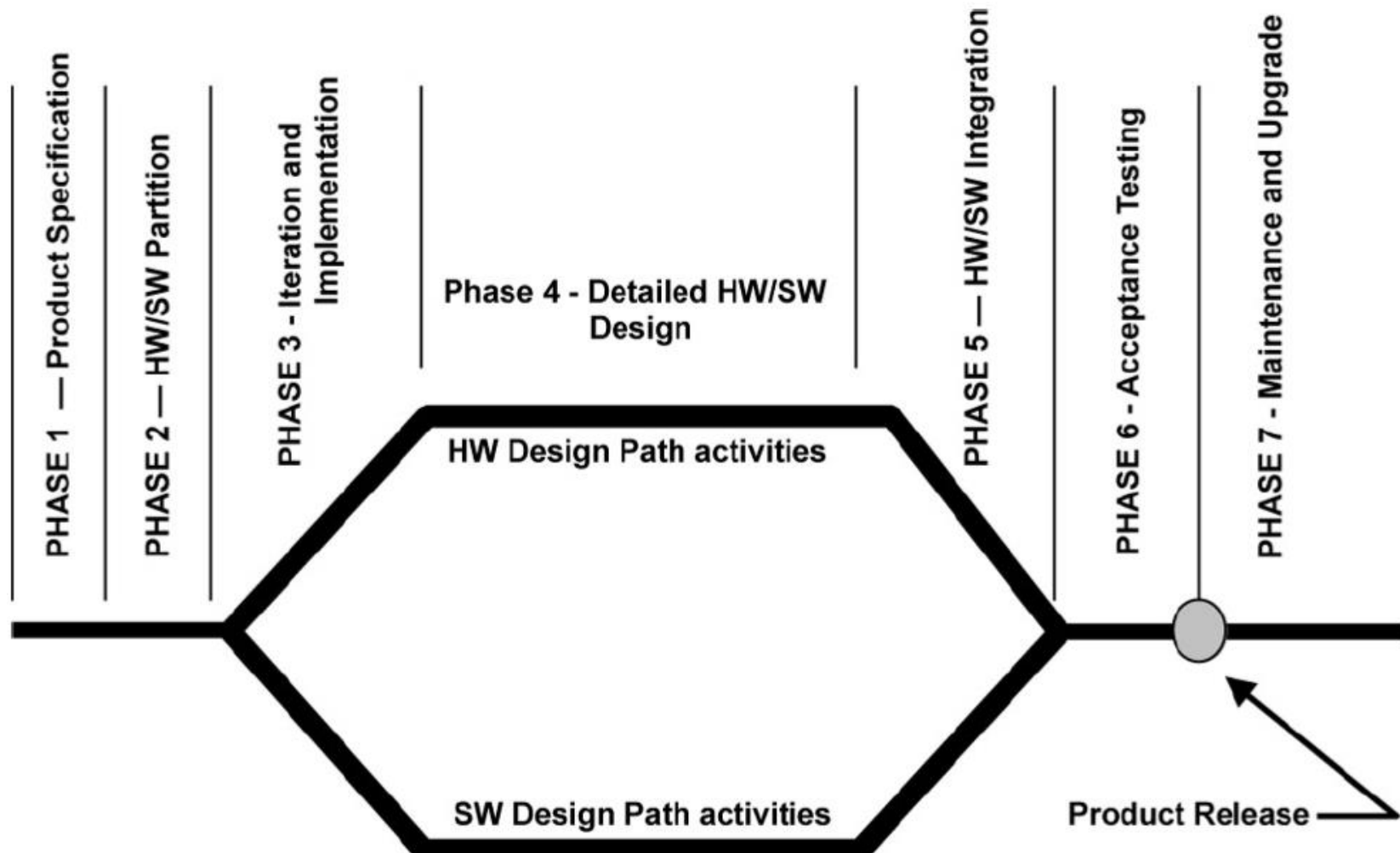


Como foi suportado



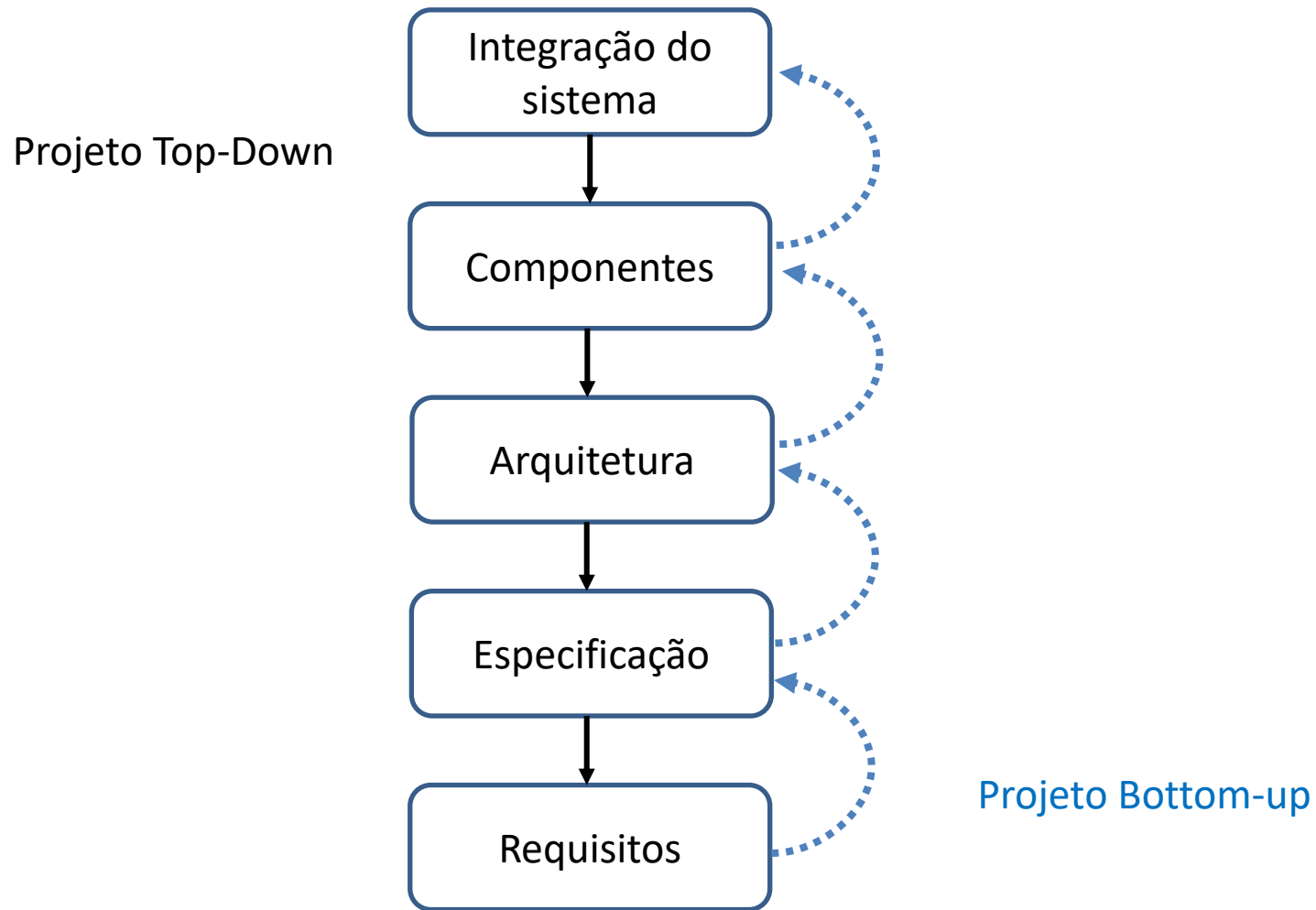
O que o cliente precisava

Fonte:
<https://medium.com/lfdev-blog/como-escrever-requisitos-de-software-de-forma-simples-e-garantir-o-m%C3%ADnimo-de-erros-no-sistema-app-74df2ee241cc>



Fonte: A. S. Berger, 2002

- Níveis de abstração do projeto



Requisitos ideais

- Separar os requisitos das especificações é importante devido à distância entre o que o cliente descreve sobre o sistema desejado, e o que realmente é necessário para projetá-lo.
- Máximo de requisitos preenchidos;
- Claros e objetivos - não ambíguos;
- Verificáveis no final do sistema;
- Consistente: os requisitos não se contradizem.
- Modificável – pode ser atualizado facilmente.
- Saber a função de cada requisito (porque estão lá...).
- Fácil de ser implementado.

Como escolher os requisitos?

- Entrevista com cliente.
- Comparar o projeto com o do concorrente.
- Feedback de vendas.
- Protótipos.

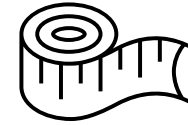
- Requisitos
 - *Funcionais: São aqueles que definem de forma concisa as funcionalidades do sistema.*
 - *Não funcionais: Se referem às restrições do projeto, que podem afetar suas funcionalidades e são fundamentais durante o projeto do sistema.*

- Requisitos não funcionais típicos:
 - Desempenho: Depende da usabilidade do sistema. Pode ser uma combinação de métricas aproximadas em nível de usuário e prazos rígidos (sistemas de tempo real) para operações específicas.
 - Custo: Um dos pontos mais importantes. Dividido pelo menos em duas partes: Custo de produção (importante para equipamentos produzidos em grande escala) e custo de desenvolvimento do sistema/tecnologia (dependendo a situação, pode ser interessante utilizar uma solução pronta – pôr a marca).



- Requisitos não funcionais típicos:

- Tamanho e peso:



- Sistema mobile? VANT?
 - Dispositivos mobile normalmente possuem uma forte restrição quanto ao tamanho e peso, o que pode inviabilizar alguns requisitos exigidos → Importância do levantamento inicial.

- VANTs: Forte restrição em relação ao peso.

- Em ambientes industriais é comum o uso de gabinetes (RACK), que possuem normas específicas quanto ao tamanho do encapsulamento.
 - Será utilizado um design específico?



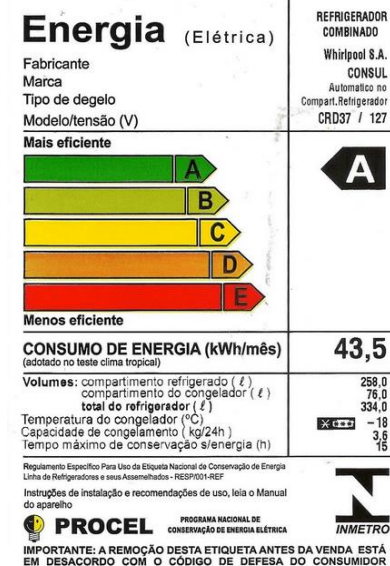
https://www.researchgate.net/figure/Figura-1-Gabinete-de-Control-GES-Foundation-FieldBus_fig1_311617866

■ Requisitos não funcionais típicos:



■ Consumo de energia

- Em sistemas que utilizam baterias, determinam o tempo de funcionamento do sistema enquanto não estiver conectado a tomada. ➔ Deve funcionar enquanto estiver recarregando?
 - Bateria maior ➔ maior peso, custo e tamanho.
 - Bateria menor ➔ maior frequência de carregamento, pode não suprir o requisito de tempo de funcionamento mínimo.
 - Ciclo de vida da bateria.
- Em sistemas permanentemente conectados à rede elétrica, determinam sua classe de eficiência energética.
 - Micro-ondas x geladeira?



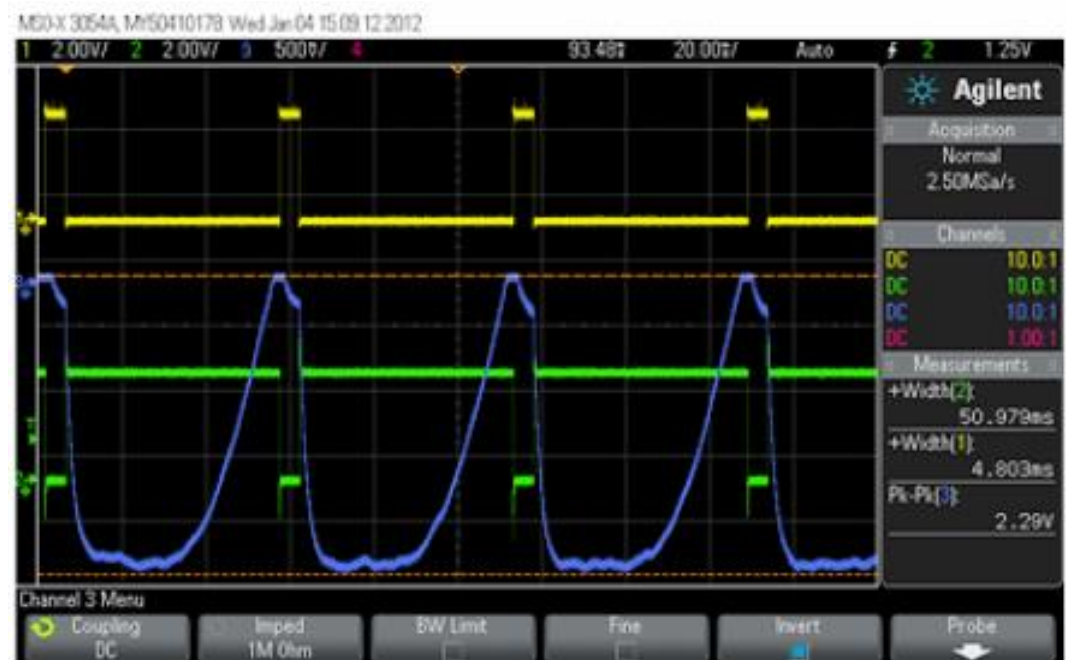
■ Requisitos não funcionais típicos:



- Consumo de energia – Experimento ARM Cortex-M4 vs ARM Cortex-M0 (Jack Ganssle) – processando o mesmo algoritmo.

Legenda:

- ARM Cortex-M4: Amarelo
- ARM Cortex-M0: Verde
- Corrente consumida: Azul (queda de tensão em um resistor de 5 Ω em série com a alimentação)
- ARM Cortex-M4
 - 12 (lógica) a 174 vezes (ponto flutuante) mais rápido.
 - Consumo de 2 a 9 vezes maior.



Fonte: <https://eda360insider.wordpress.com/2012/09/17/a-head-to-head-comparison-of-the-arm-cortex-m4-and-m0-processor-cores-by-jack-ganssle/>

■ Requisitos não funcionais típicos:

- Consumo de energia →
- Temperatura
 - O sistema possui componentes que aquecem?
 - Qual o limite de aquecimento?
 - Limite de hardware/sistema ao qual está inserido ou limite de conforto durante utilização
 - É necessário o uso de dissipadores? Cooler?
 - Peso, custo, tamanho, ruído...
- Ruído
 - Ambiente industrial? Ambiente doméstico? Qual o uso?



■ Requisitos não funcionais típicos:

■ Certificações/homologações

- Certificações de EMI/EMC
- Certificação de proteção de dados
- Certificação de órgãos de segurança
- Certificação prova d'agua e poeira
- Homologação RF (Anatel)
- Homologação ANAC
- Homologação INMETRO







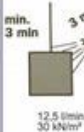
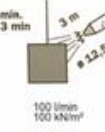
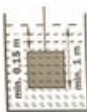

1º NUMERAL CARACTERÍSTICO

Grau de proteção contra pessoas e objetos sólidos

2º NUMERAL CARACTERÍSTICO



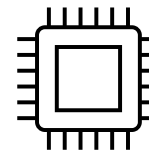
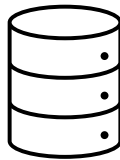
Grau de proteção contra o ingresso prejudicial de água

0	1	2	3	4	5	6	7	8
Não protegido	Protegido contra quedas verticais de gotas d'água	Protegido contra quedas verticais de gotas d'água para uma inclinação máxima de 15°	Protegido contra água aspergida de um ângulo de $\pm 69^\circ$	Protegido contra projeções d'água	Protegido contra jatos d'água	Protegido contra ondas do mar ou jatos potentes	Protegido contra imersão	Protegido contra submersão
								
IP 00	IP 01	IP 02						
IP 10	IP 11	IP 12	IP 13					
IP 20	IP 21	IP 22	IP 23					
IP 30	IP 31	IP 32	IP 33	IP 34				
IP 40	IP 41	IP 42	IP 43	IP 44	IP 45	IP 46		
			IP 53	IP 54	IP 55	IP 56		
					IP 65	IP 66	IP 67	IP 68

<https://www.fibracem.com/o-que-e-grau-de-protecao-ip/>

- Requisitos funcionais típicos:
 - Funcionamento: Descrição sucinta do que o sistema deve fazer.
 - Entradas e saídas:
 - Tipos de dados/sinais: sinais analógicos e/ou digitais (e quantidade), entradas mecânicas, etc.
 - Periodicidade dos dados: Os dados chegam periodicamente ou esporádicos?
 - Tipos de dispositivos de I/O: chaves mecânicas, conversores analógicos/digitais, entradas/saídas de áudio/vídeo, PWM, etc.

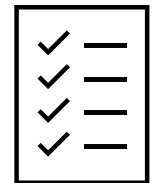
- Particionamento entre *Hardware* e *Software*
 - *Decidir se os módulos (alguns ou todos) serão sintetizados em hardware ou compilados em software.*
 - Às vezes é preferível implementar um canal extra de PWM por software do que trocar o processador.
 - *Escolha dos componentes necessários para a execução do projeto: CPU, DSP, FPGA, etc.*
 - *Módulos de comunicação.*



- Requisitos funcionais típicos:
 - Funcionalidades desejadas: descrição mais detalhada do que o sistema deve fazer.
 - Análise da interação do sistema com o ambiente:
 - O que o sistema deve fazer ao receber uma determinada entrada?
 - Qual o tipo de sinal de saída esperado?
 - Qual a forma de comunicação com sistemas vizinhos (caso exista)?
 - Como as funções interagem entre si?
 - Etc.

- Validação de um conjunto de requisitos
 - Um ambiente de simulação/virtual ou um protótipo pode ser utilizada para demonstrar as funcionalidades do sistema.
 - Implementação de parte do sistema para testes.
 - Protótipos físicos não funcionais podem ser utilizados para demonstrar características de tamanho e peso.

- Fase de especificação do produto
 - *A especificação é a tradução dos requisitos para a linguagem técnica.*
 - *Para a criação de um novo produto pode envolver uma pesquisa de mercado*
 - *Pode ser explicitamente demandado pelo cliente*
 - Entender as necessidades do cliente (requisitos)
 - Transformar os requisitos em especificação
 - Alinhar os requisitos com o cliente



- Prazo para execução de processos
- Muitos processos são executados a todo instante:
 - *Dados de entrada via teclado,*
 - *Sinais de áudio e vídeo,*
 - *Atualização da tela.*
- Prazo de término:
- Requisitos de tempo real:
 - *Soft real time: Habilidade do sistema de respeitar prazos; Utilizado em sistemas onde o tempo de resposta não é crítico.*
 - *Hard real time: Garantir que os prazos serão atendidos; Utilizado em sistemas críticos.*

- Interfaces de usuário
 - Interação do usuário com o sistema
 - Entrada de dados:
 - Teclado, voz, display touch, botões, etc
 - Usabilidade
 - Facilidade de aprender
 - Facilidade de usar
 - Nicho de mercado
 - Design
 - Aceitação comercial

- Processador
 - Quantidade de núcleos
 - Clock necessário para suprir os requisitos de desempenho
 - Específico para tempo real?
 - Números de GPIO
 - Modos de baixo consumo
 - Interfaces/periféricos
 - UART, SPI, I2C, etc
 - Canais PWM, número de timers/contadores, etc
 - A/Ds, D/As, amplificadores, filtros, etc
 - RTC, CAN, ethernet, etc

Especificação por requisitos

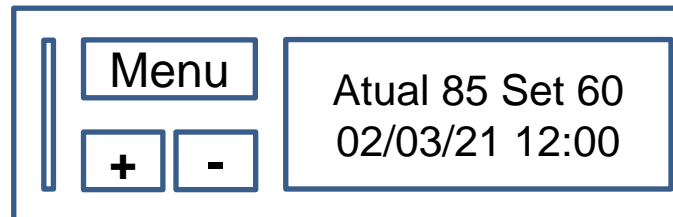
- Memória
 - Quantidade mínima para programa e dados
- Considerações para sistema de tempo real
 - Análise do tempo de resposta para controle, tempo de conversão A/D, processamento, canais DMA, etc
- Ambientes de desenvolvimento
 - Custo
 - Know-how
 - Suporte
 - Desenvolver x comprar pronto

- Canais DMA
 - Possibilita transferências de dados em alta velocidade entre periféricos e a memória sem o uso da CPU.
 - Reduz atrasos no acesso
 - Libera o processador para outras tarefas
 - Podem utilizar DMA
 - USB
 - SPI, UART, etc
 - Conversores A/D e D/A

- Temperatura
 - -40 a 85 graus / estendido → -40 a 105 graus
- Ciclo de vida
 - Ciclo de vida dos componentes e do produto
 - Ciclo de vida relacionado a atualizações

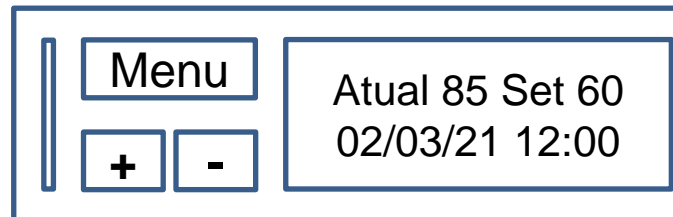
Exemplo de um equipamento

- Exemplo de requisitos: Controle de irrigação.
 - Funcionalidade: O sistema deverá ser capaz de medir a umidade do solo e ligar ou não a bomba de água para iniciar a irrigação sempre que necessário. Também deve salvar em um cartão SD a umidade da terra a cada 1 hora.
 - Interface com o usuário: Deverá conter 3 botões físicos para escolher o limiar de umidade para iniciar a irrigação e para programar a data e hora correta. O nível de umidade atual e escolhido deve aparecer em um display LCD 16x2, além disso, também deve aparecer a data e a hora.

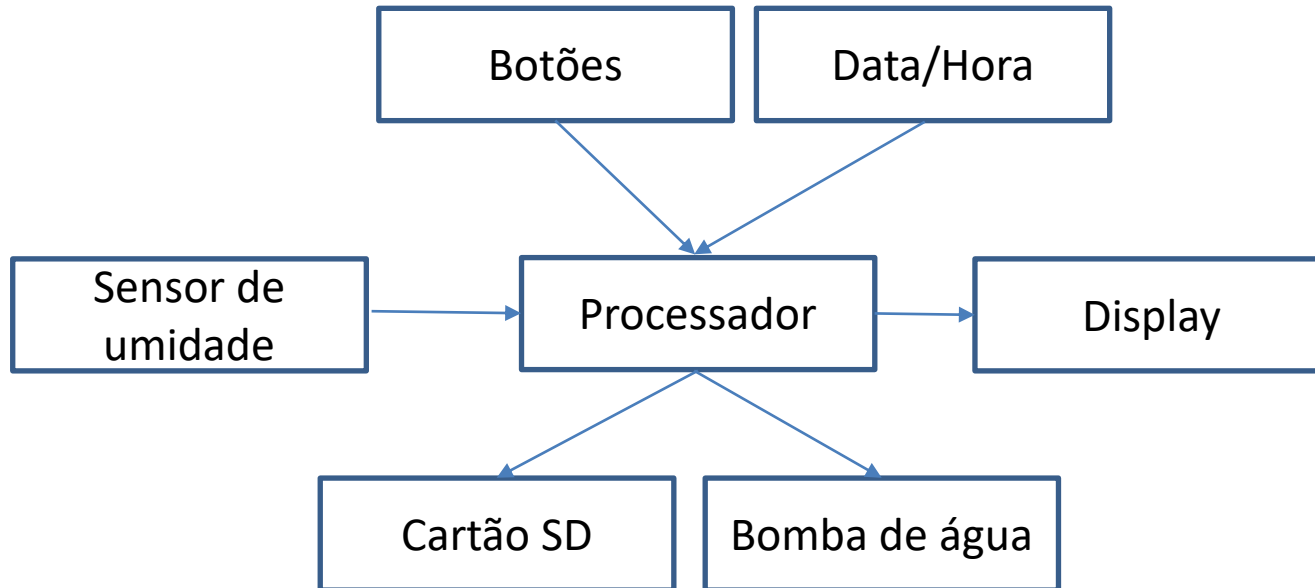


Exemplo de um equipamento

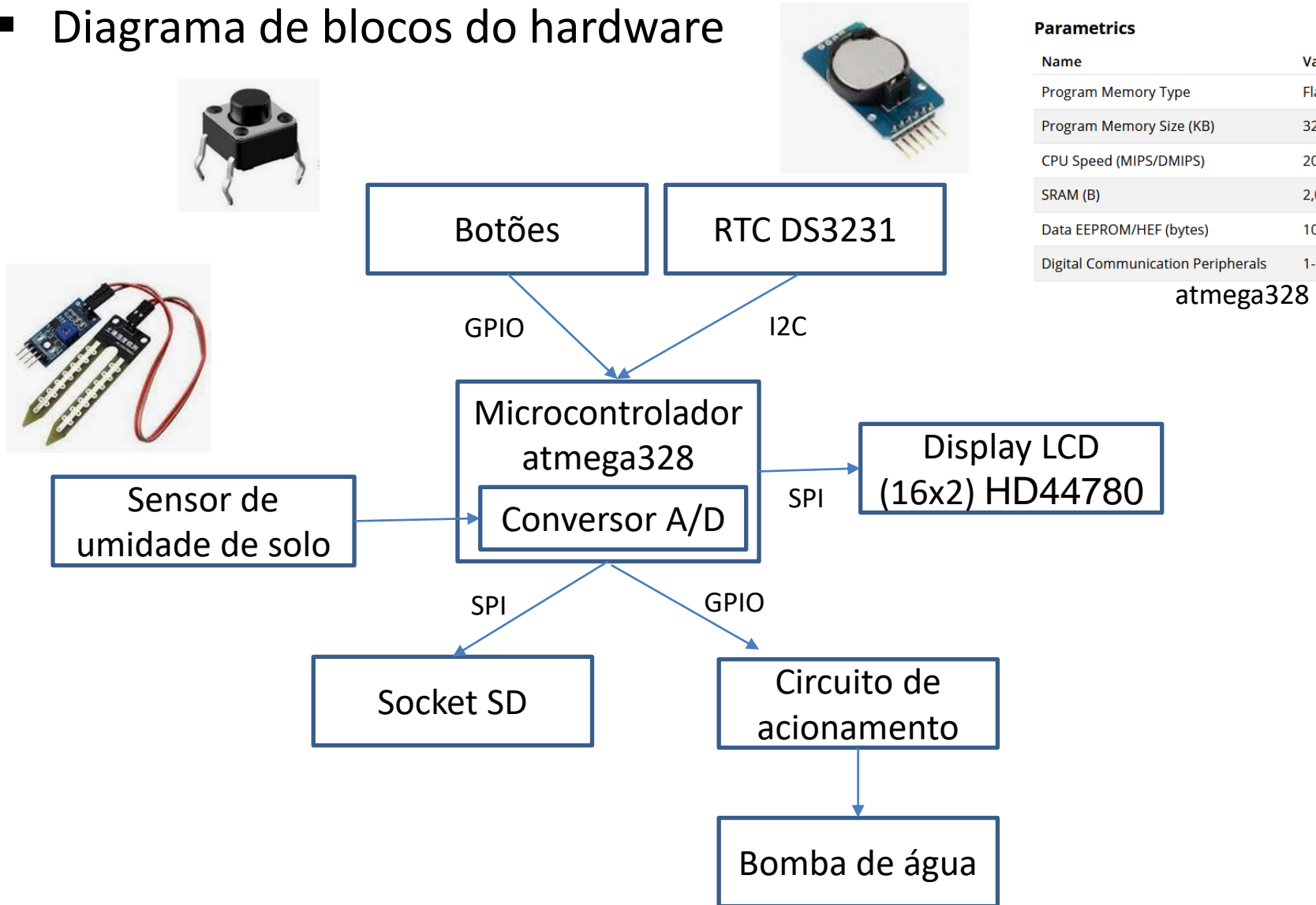
- Exemplo de requisitos: Controle de irrigação.
 - Desempenho: O sistema deve ser capaz de medir a umidade a cada 1 minuto e salvar a umidade média no cartão SD a cada 1 hora.
 - Tamanho e peso: No máximo 15 cm em cada dimensão, sem restrição de peso.
 - Custo de fabricação: Máximo de 100 reais.
 - Consumo: No máximo 3W (controlador). Deve possuir certificação IP65 (proteção contra poeira e jatos de água).



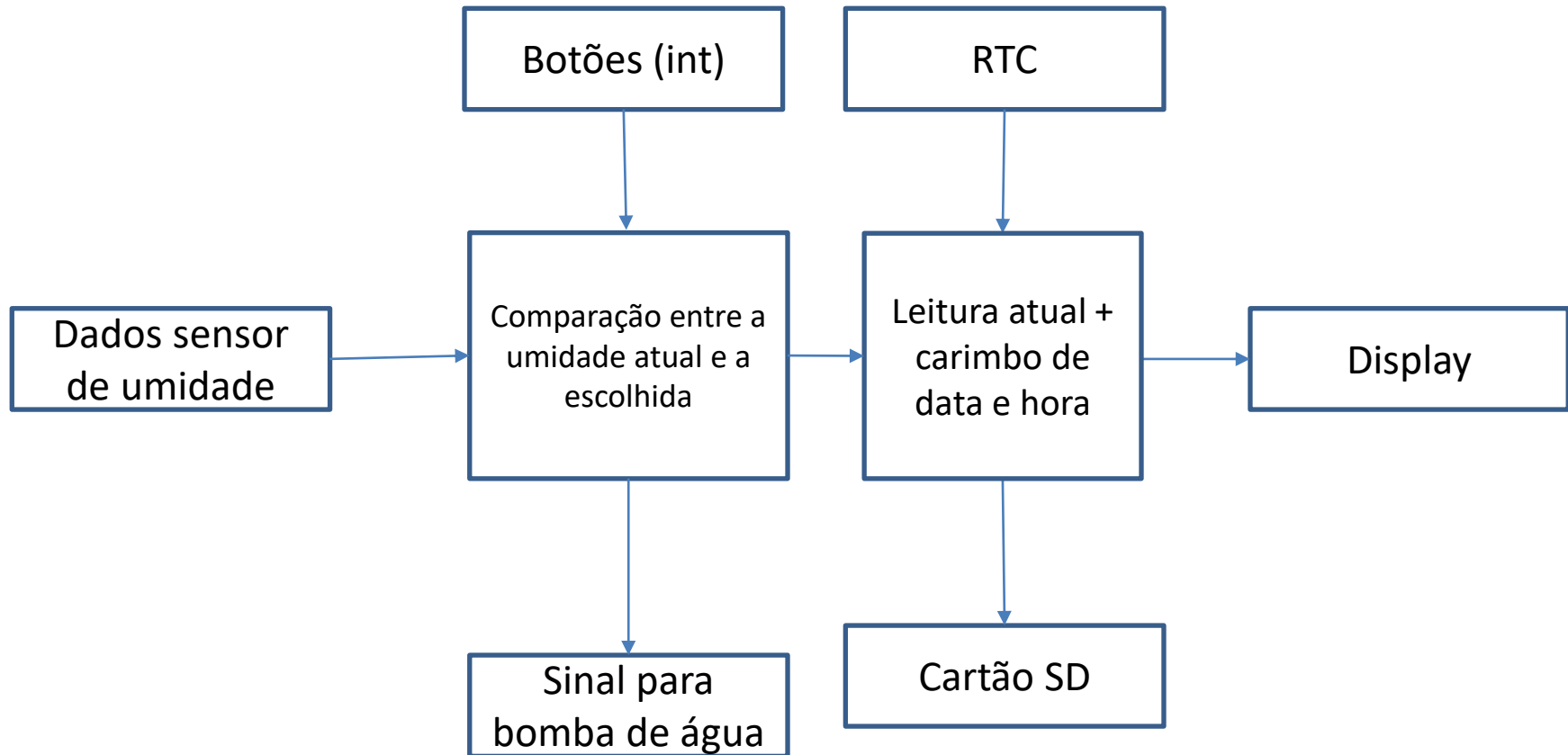
- Diagrama de blocos do sistema de irrigação



■ Diagrama de blocos do hardware



- Diagrama de blocos do software



- Aspectos que não aumentam o custo de engenharia.
- Versão original do produto:
 - *Alteração de características.*
 - *Objetivo principal do sistema não se altera.*
 - *Funcionalidades adicionais.*
- Exemplo: Máquina de lavar roupas.



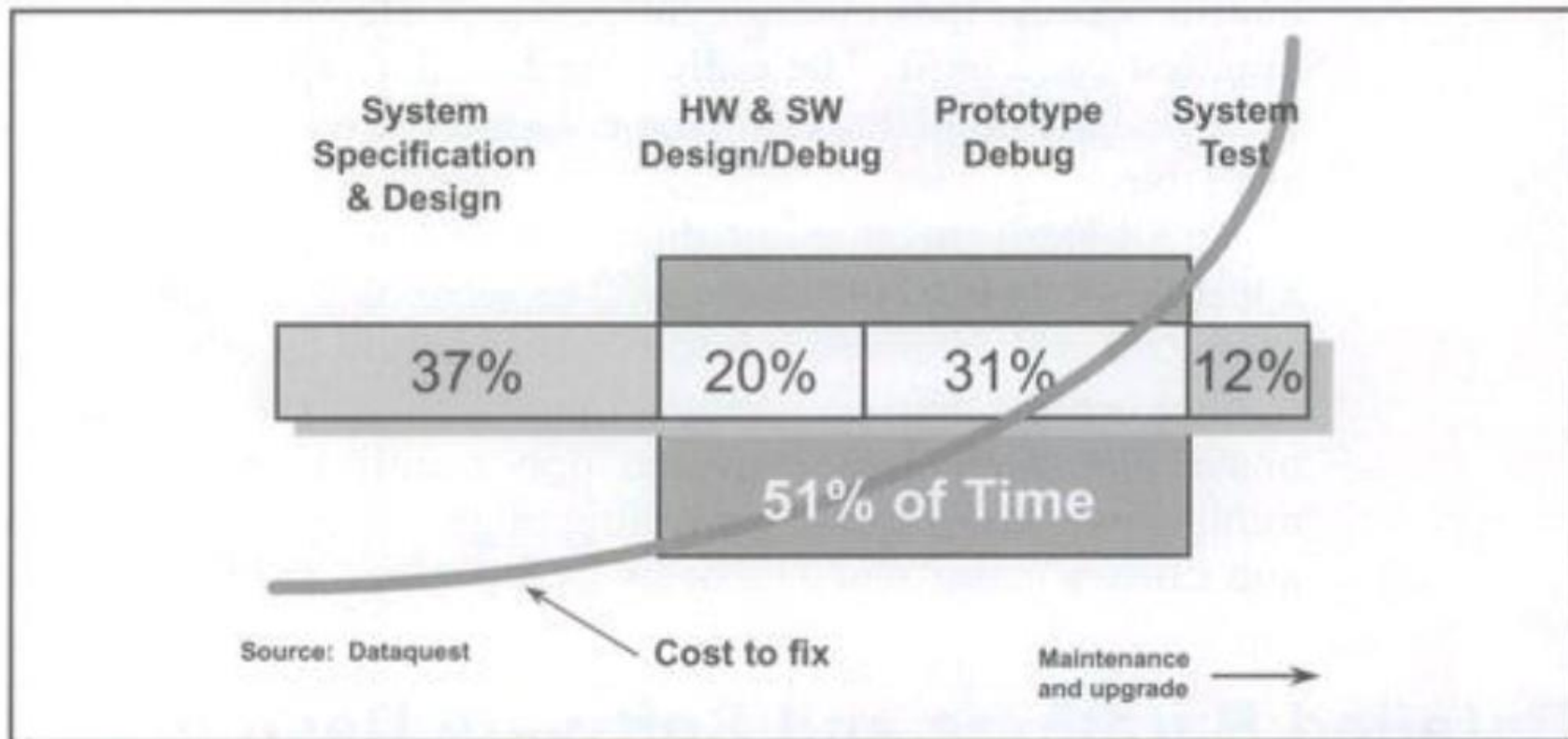
Integração e testes do sistema

- Fase em que todos os componentes do sistema são unidos e os testes do equipamento começam ser executados
 - Geralmente é nessa fase que surgem os maiores problemas. Se a fase de projeto/especificação não foi bem feita, dependendo do tipo de problema encontrado, é necessário “reiniciar” o projeto. Ex. GPIO/memória insuficientes, consumo/tamanho/peso acima do requisitado, EMI não esperadas, oscilação de tensão/harmônicas geradas por motores e chaveamentos, etc.
- O ideal é integrar o sistemas em partes (módulos) e executar testes específicos para cada parte, assim fica mais fácil identificar os problemas
 - Em muitos casos, uma falha total é gerada por um bug simples de resolver.



<https://labdegaragem.com/profiles/blogs/tutorial-raqueando-o-carrinho-de-controle-remoto>

Tempo no desenvolvimento



The percentage of project time spent in each phase of the embedded design life cycle. The curve shows the cost associated with fixing a defect at each stage of the process.

Fonte: A. S. Berger, 2002

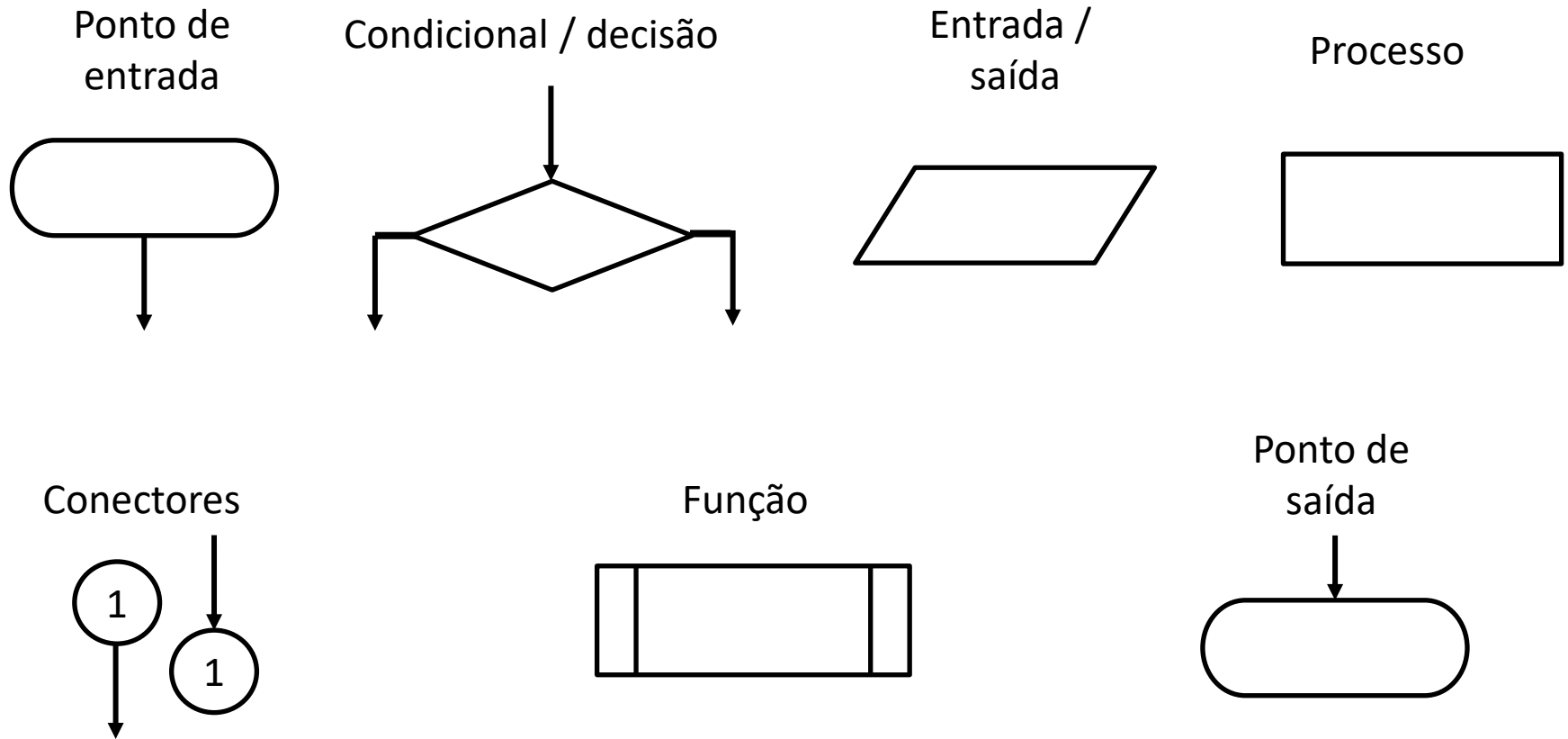
- A dificuldade em analisar os detalhes em um sistema embarcado é maior do que em sistemas de propósito geral
- O uso de ferramentas de depuração auxilia na fase integração
- No entanto, a fase de integração geralmente é mais complexa
 - Experiência é importante nessa fase.

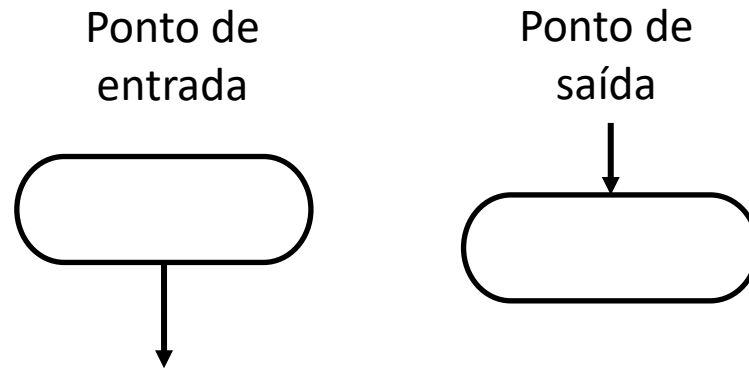
Fase de manutenção e atualizações

- Acontece após o lançamento do produto
 - Nessa fase, é comum identificar bugs que passaram despercebidos na fase de testes, mas que pode afetar a “imagem” do produto.
 - O custo da correção nessa fase é mais alto do que durante o processo de desenvolvimento do produto
 - Também é possível que o cliente peça para incluir novas funcionalidades no produtos, porém essa flexibilidade já deve ser prevista na fase de projeto, pois pode requerer “sobra” de hardware para a atualização.

Fluxograma do projeto

Fluxograma do projeto





- Ponto de entrada e saída: Utilizados como início e final do software, função ou subrotina. Softwares para sistemas embarcados costumam rodar continuamente (infinitamente) e por isso, geralmente não possuem ponto de saída.

Fluxograma do projeto

- Processo: Utilizados para operações específicas.
- Entrada / saída: Utilizado para ligações com periféricos externos, sensores e atuadores.
- Em muitos casos, utiliza-se o paralelogramo e o retângulo para as mesmas funções, mas como entrada / saída é bastante comum em sistemas embarcados, costuma-se separar.

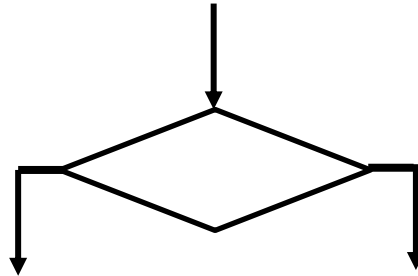
Processo



Entrada / saída

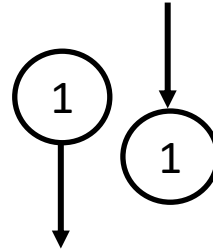


Condicional / decisão



- Condicional: Dentro da forma é colocado o que esta sendo testado.
 1. A condição para cada saída deve ser mutuamente exclusiva. Ex: Se $X = 10$ vá para esquerda, se $X > 2$ vá para a direita. Quando for 10 fica inconclusivo.
 2. Também deve abordar todas as situações. Ex: Se $X < 10$ vá para a esquerda e $X > 20$ vá para a direita. Entre 10 e 20 não tem solução.

Conectores



- Conectores: devem ser numerados. seta saindo significa uma label ou ponto do algoritmo. Com seta entrando é um jump ou comando goto. Quando a execução atinge um conector goto, ele pula para a posição especificada pelo numero do conector.

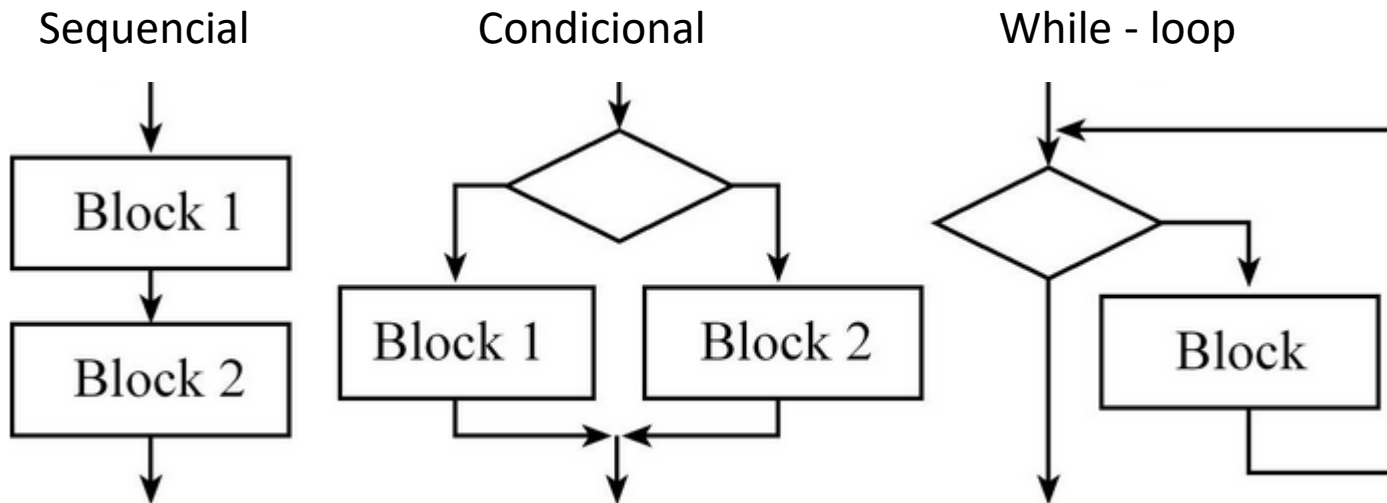
Função



- Leva a execução para uma função específica.

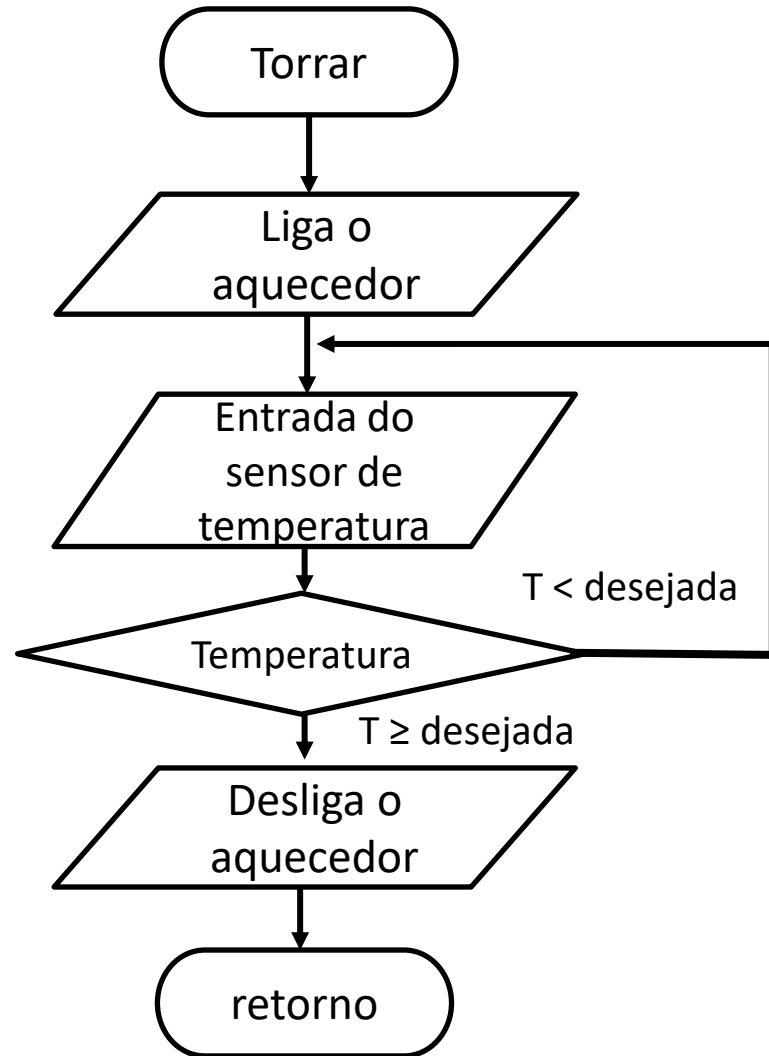
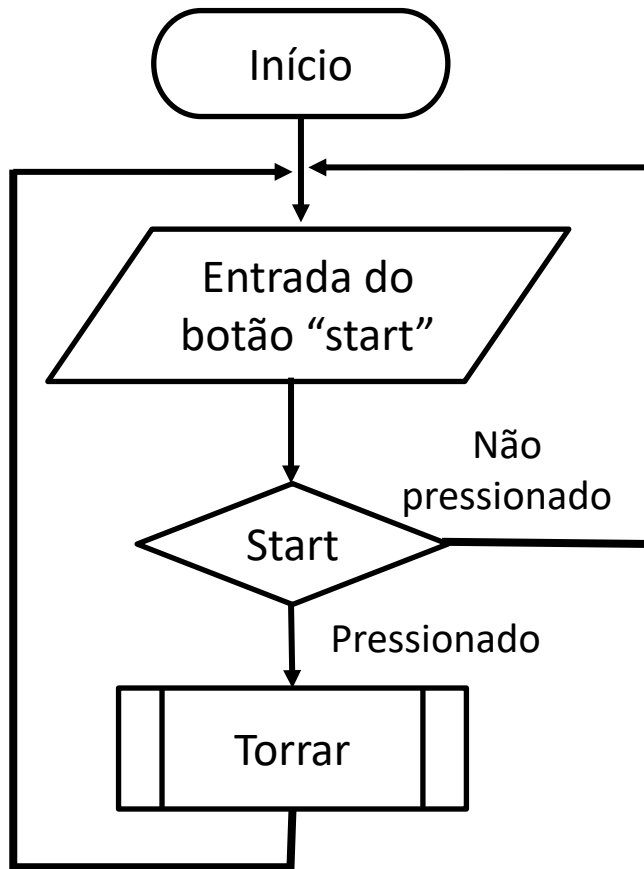
Fluxograma do projeto

- Um software é estruturado de três formas:
 - Sequencial: As operações são executadas uma após a outra.
 - Condicional: Uma condição é testada para decidir qual operação executar.
 - Iterativa: Repete uma operação “infinitamente” até que uma condição se torne verdadeira.



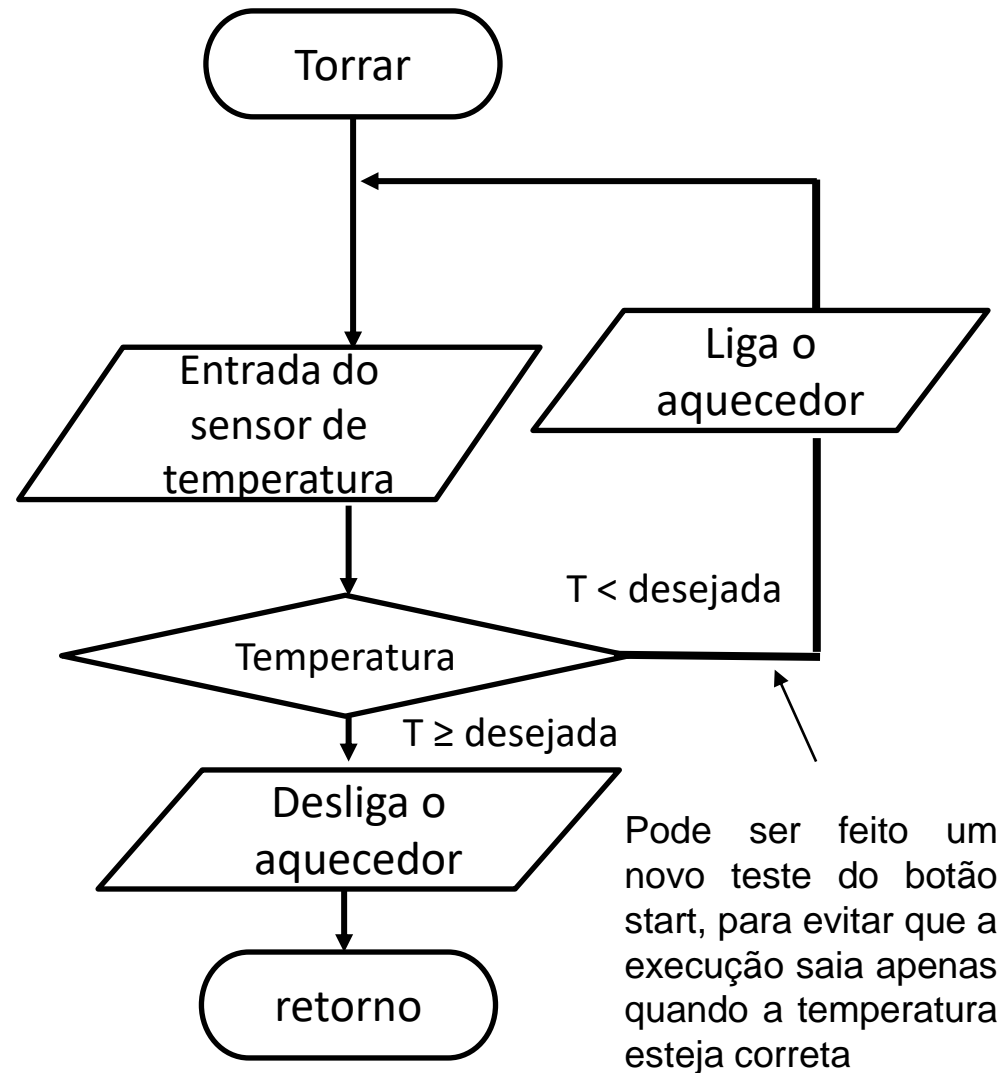
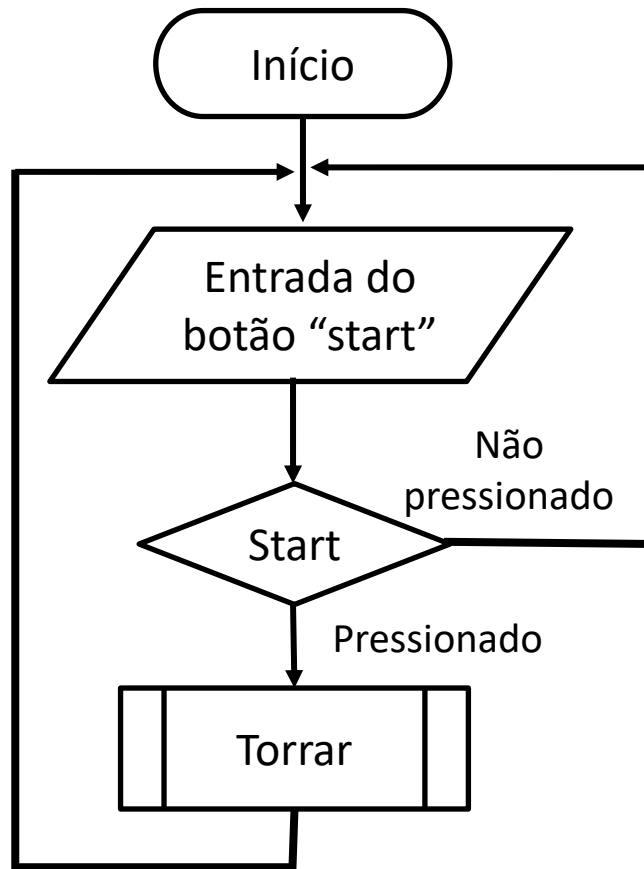
Fluxograma do projeto

- Exercício 1: Faça o fluxograma de uma torradeira de pão com controle de temperatura (temperatura fixa). O controle de temperatura fica ativo enquanto o botão “Start” estiver pressionado.



Fluxograma do projeto

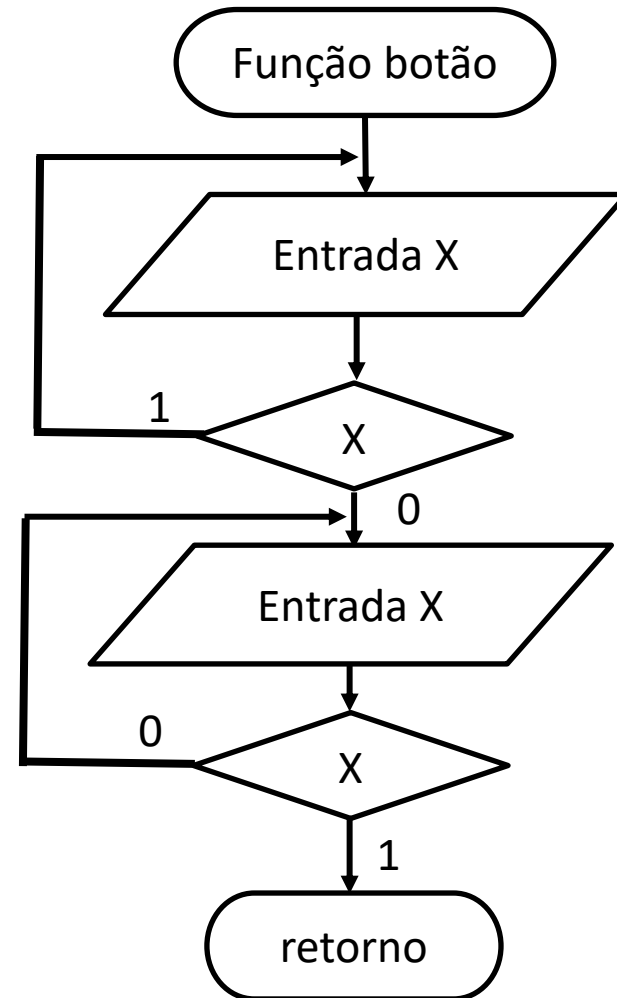
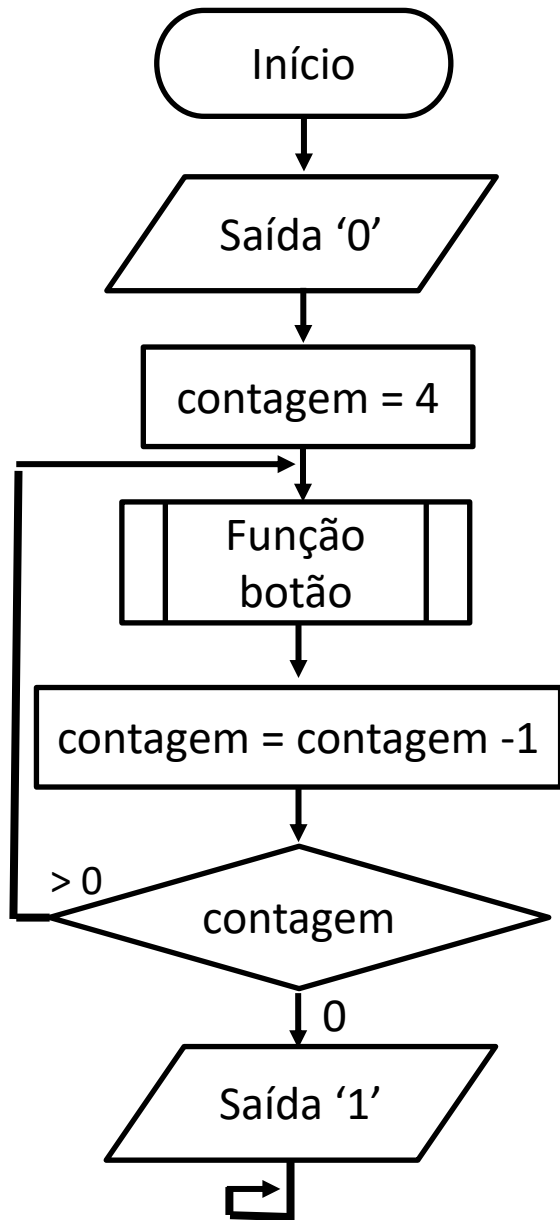
- Exercício 1: Faça o fluxograma de uma torradeira de pão com controle de temperatura (temperatura fixa). O controle de temperatura fica ativo enquanto o botão “Start” estiver pressionado.



Fluxograma do projeto

- Exercício 2: Faça o fluxograma de um sistema com uma entrada X (botão) e uma saída (LED). Uma função deve reconhecer quando a entrada vai de 1 para 0 e depois retorna para 1. Após 4 cliques no botão, a saída deve ir para 1. A saída inicia em zero e a entrada em 1.

Fluxograma do projeto

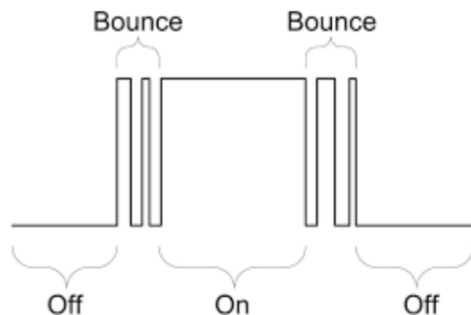


Exemplo

- Faça um software relativo ao fluxograma anterior.

GPIO Pin	Dispositivo
7	LED
3	BOTAO

Quais problemas foram encontrados?



```
#define LED 7
#define BOTAO 3

int contagem = 4;

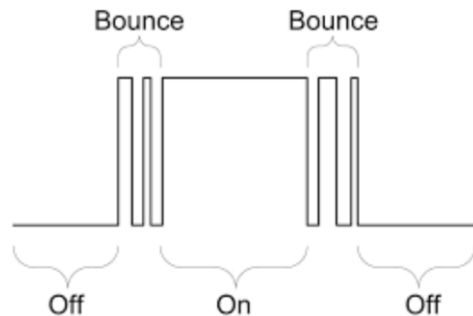
void setup() {
  pinMode (LED, OUTPUT);
  pinMode (BOTAO, INPUT_PULLUP);
}

void loop() {
  if(digitalRead(BOTAO)==LOW) {
    delay(100);
    contagem = contagem - 1;
  }
  while(contagem<1) {
    digitalWrite(LED, HIGH);
  }
}
```

Exemplo

- Faça um software relativo ao fluxograma anterior.

GPIO Pin	Dispositivo
7	LED
3	BOTAO



“filtro” HIGH - LOW →

“filtro” LOW - HIGH →

```
#define LED 7
#define BOTAO 3
int contagem = 4;

void setup() {
  pinMode (LED, OUTPUT);
  pinMode (BOTAO, INPUT_PULLUP);
}

void loop() {
  if (digitalRead (BOTAO) == LOW) {
    delay (50);
    while (digitalRead (BOTAO) == LOW) {}
    delay (100);
    contagem = contagem - 1;
  }
  while (contagem < 1) {
    digitalWrite (LED, HIGH);
  }
}
```


- As duas formas anteriores NÃO são recomendadas de serem utilizadas em códigos mais complexos. Existem duas formas corretas:
 - Debounce por software;
 - Debounce por hardware;

Metodologia de Projeto de Sistemas Embarcados / Git

- DENARDIN, G. W. Notas de aula de sistemas embarcados. UTFPR-PB.
- BACURAU, R.M. Notas de aulas de projeto de sistemas embarcados. UNICAMP, 2020. Disponível em: <https://sites.google.com/site/rodrigobacurau/cursos-2020-1/es670---projeto-de-sistemas-embarcados>
- A. S. Berger, Embedded Systems Design An Introduction to Processes, Tools, and Techniques CMP Books, 2002.