

PROGRAMA VISUAL DE EXECUÇÃO DE CÓDIGO PARA LPC11U3X

Aluno: Willian de Avilla Silveira

Orientador: Miguel A. Wisintainer

Roteiro

- Introdução;
- Objetivos;
- Fundamentação teórico;
- Trabalhos correlatos;
- Requisitos;
- Especificação;
- Implementação;
- Resultados;
- Conclusão.

Introdução

- Bruce Eisenhard;
- A Coridium Corp presta consultoria em design de sistemas embarcados desde 2004, ao qual vendem várias placas de microprocessadores embarcadas e ferramentas de software.
- O BASIC é uma linguagem de programação, fácil de se aprender e utilizar

Objetivos

- VLPs (Visual Programming Languages);
- Google Blockly;
- BASICTools;
- LPC11u37.

Fundamentação Teórica

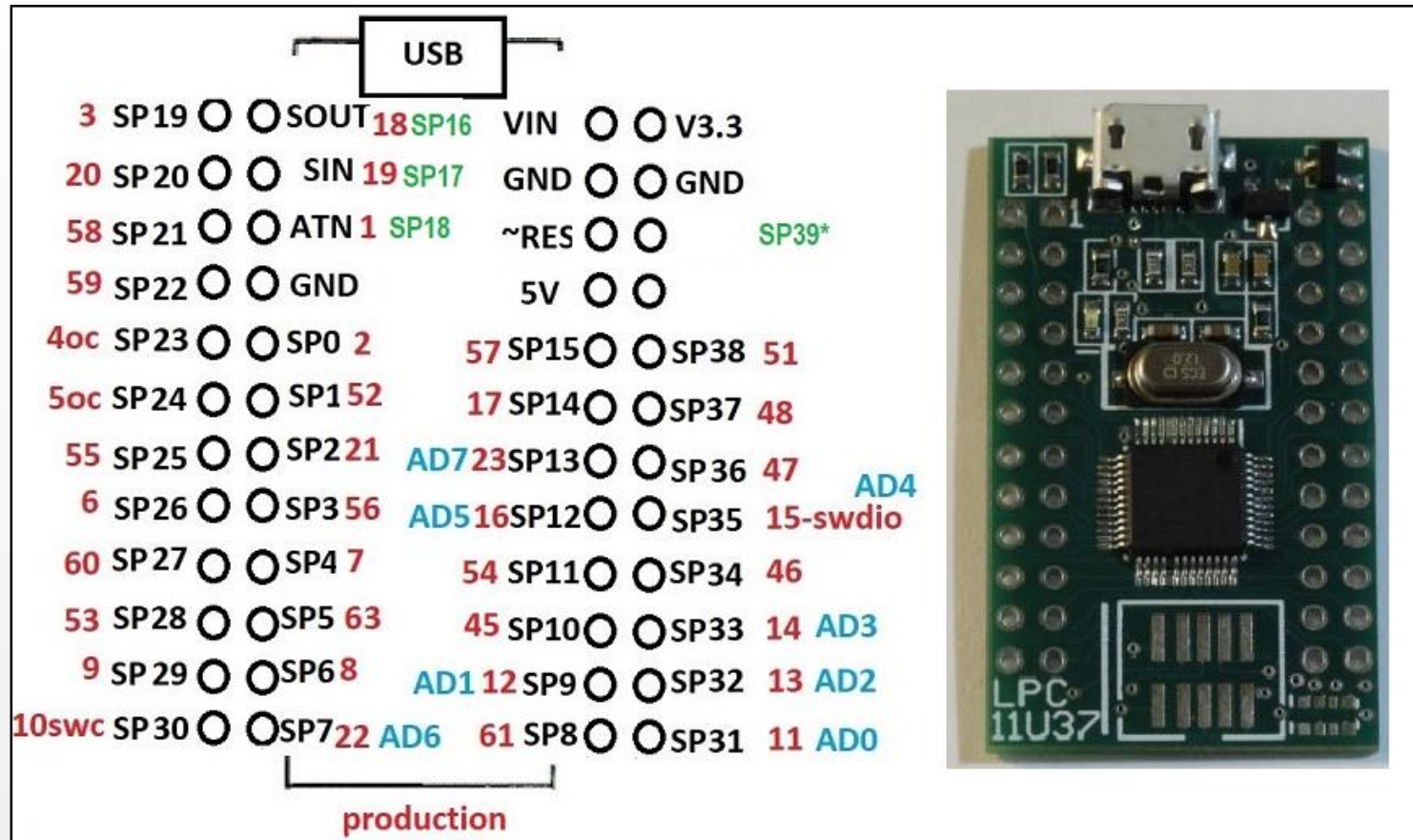
- IoT;
- Compiladores;
- Basic para processadores ARM;
- LPC11u37.

Fundamentação Teórica

```
WHILE X<30  
  
    print X  
    X=X+1  
  
    IO(1) = X and 1      ' IO() sets pin direction and state  
    WAIT(500)  
LOOP
```

Fundamentação Teórica

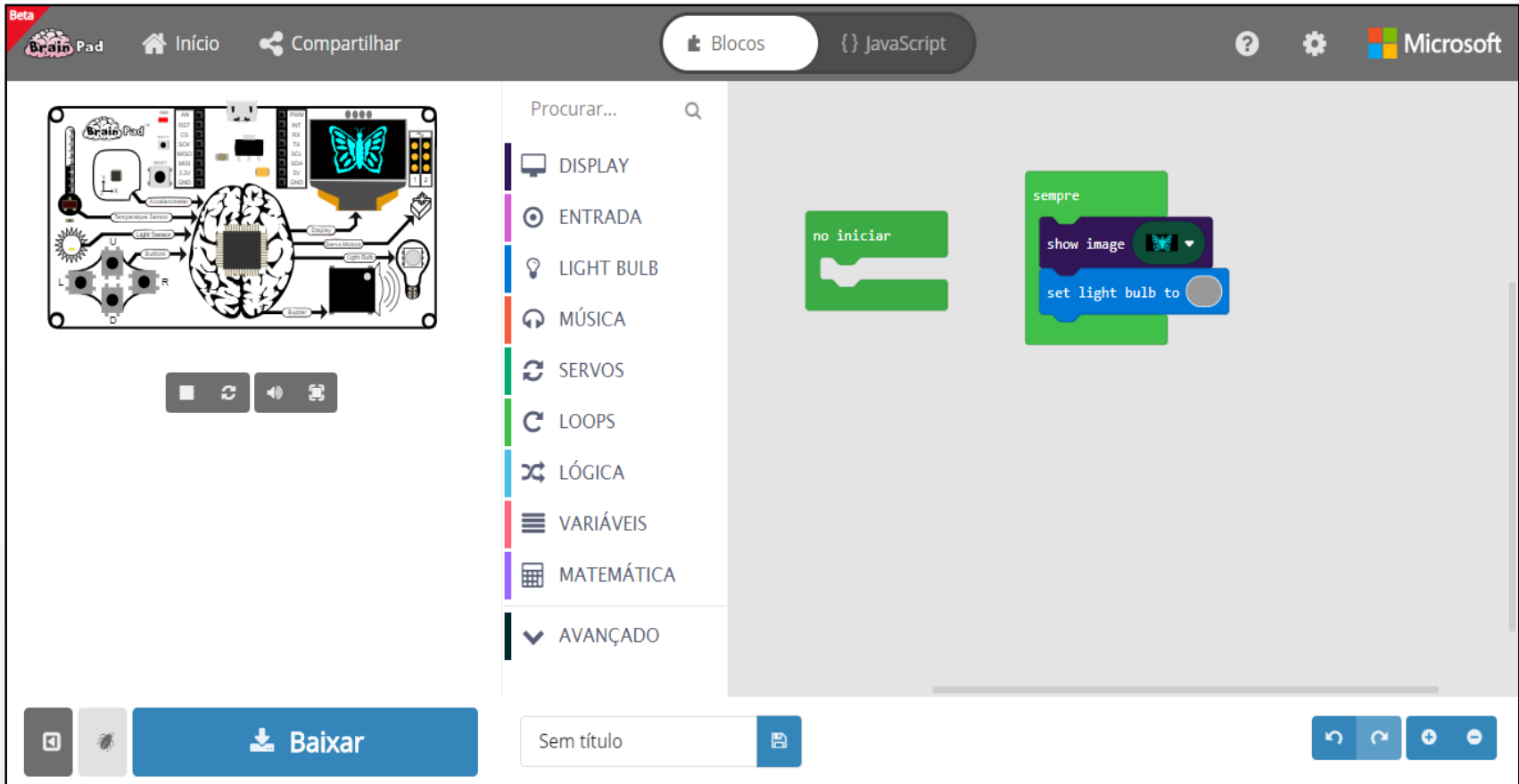
Placa LPC11u37



Fonte: Coridium (2019)

Trabalhos Correlatos

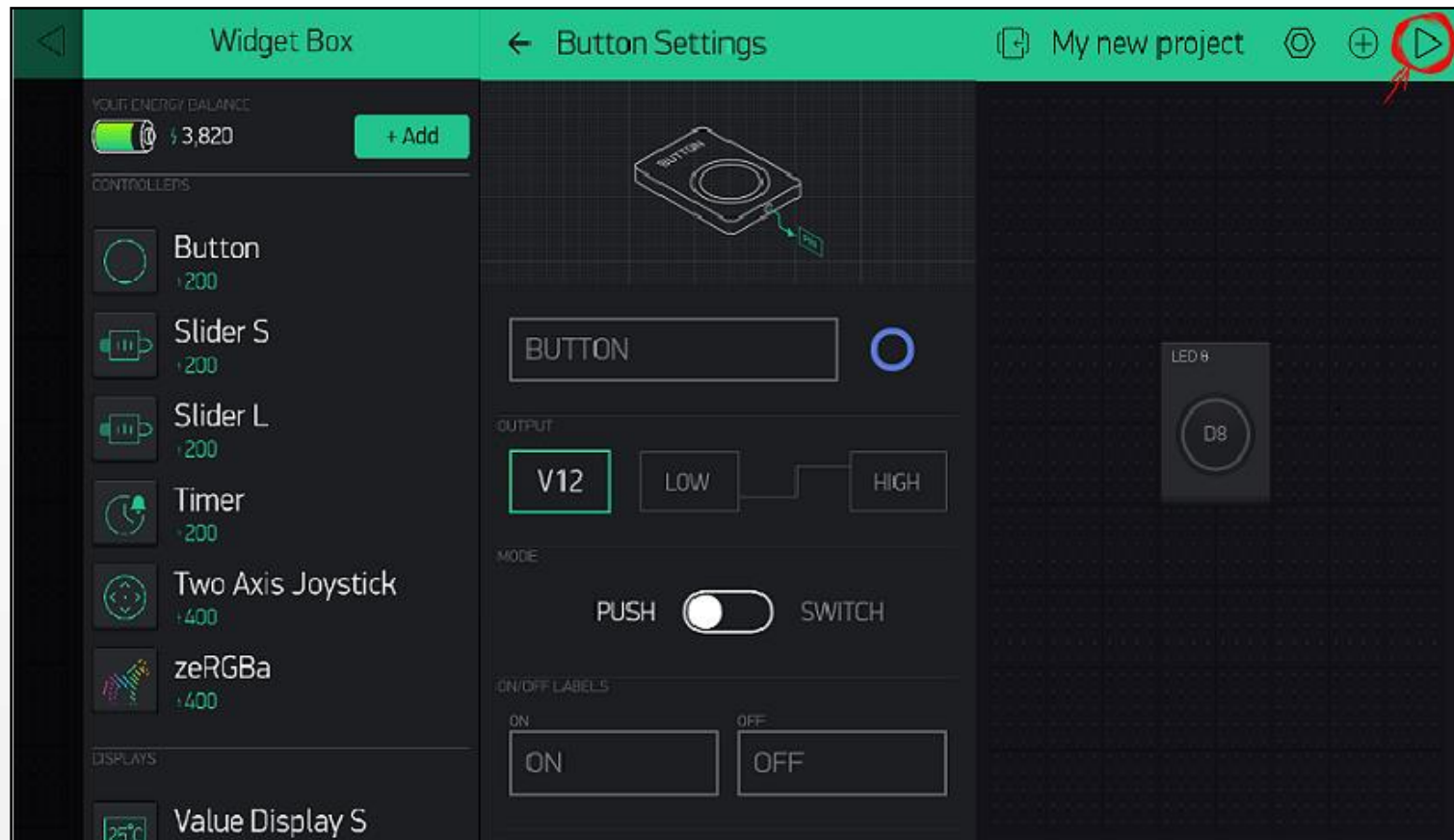
BrainPad



Fonte: BrainPad (2019)

Trabalhos Correlatos

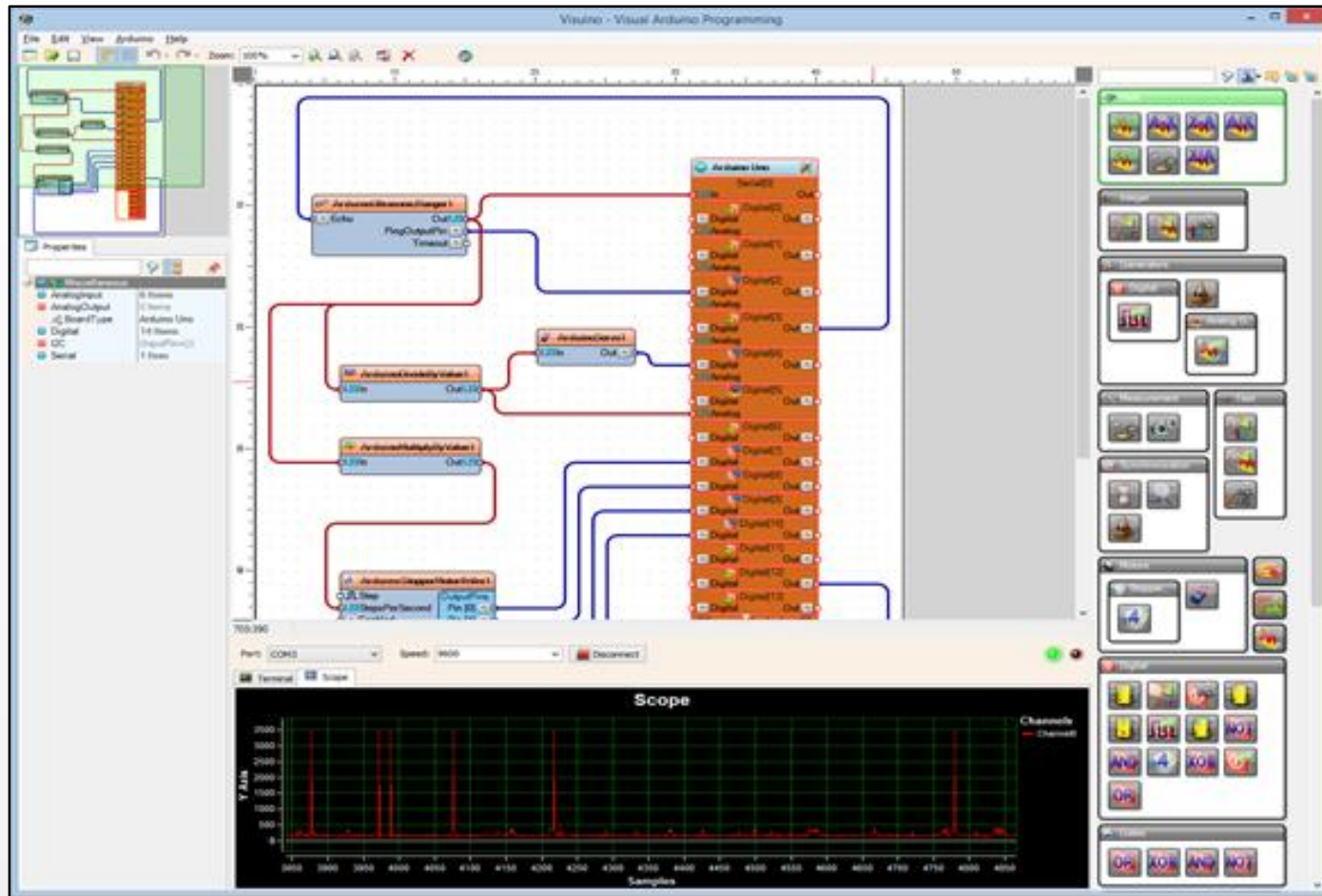
Blynk



Fonte: Baiborodin (2019).

Trabalhos Correlatos

Visuino

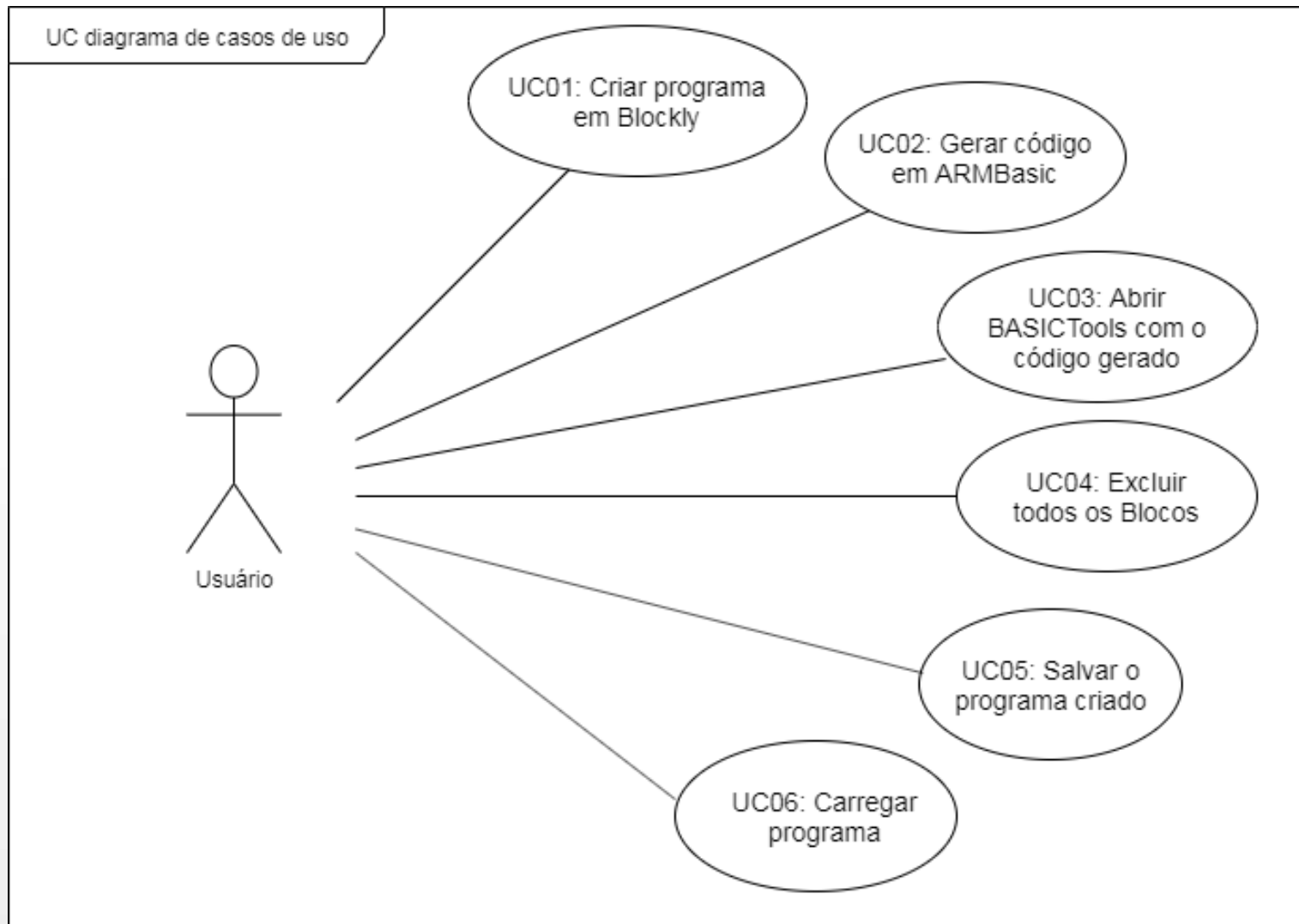


Fonte: Visuino (2017)

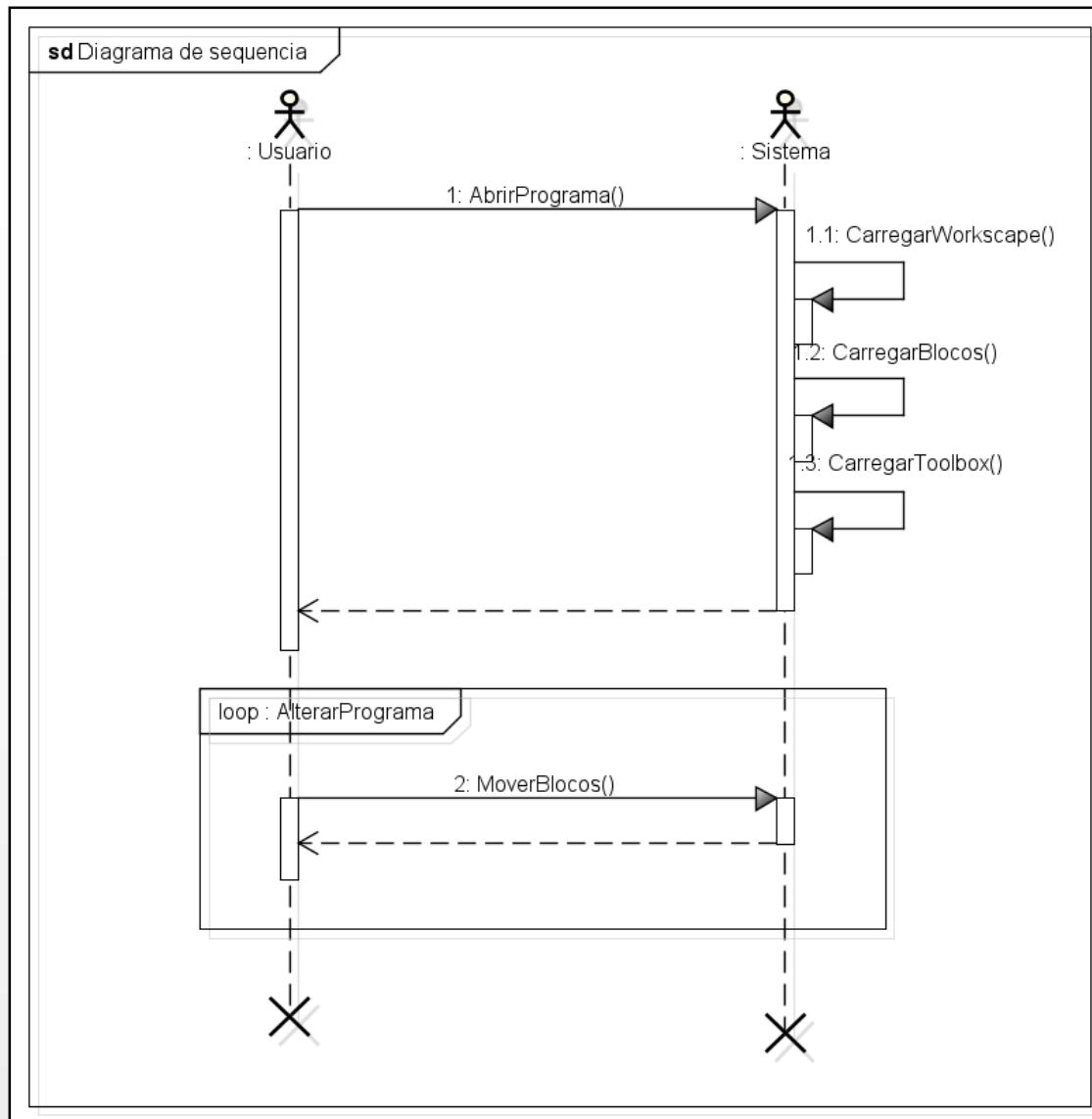
Requisitos

- permitir que o usuário visualize e edite programas escritos em Blockly (Requisito Funcional - RF);
- permitir que o usuário salve e carregue programas escritos em Blockly (RF);
- permitir que o usuário gere e possa visualizar o código na linguagem BASIC (RF);
- permitir que o usuário execute o código BASIC gerado na placa LPC11u37 (RF);
- utilizar a plataforma embarcada LPC11u37 (Requisito não Funcional - RNF);
- utilizar ferramenta BASICTools (RNF);
- utilizar a API Blockly (RNF).

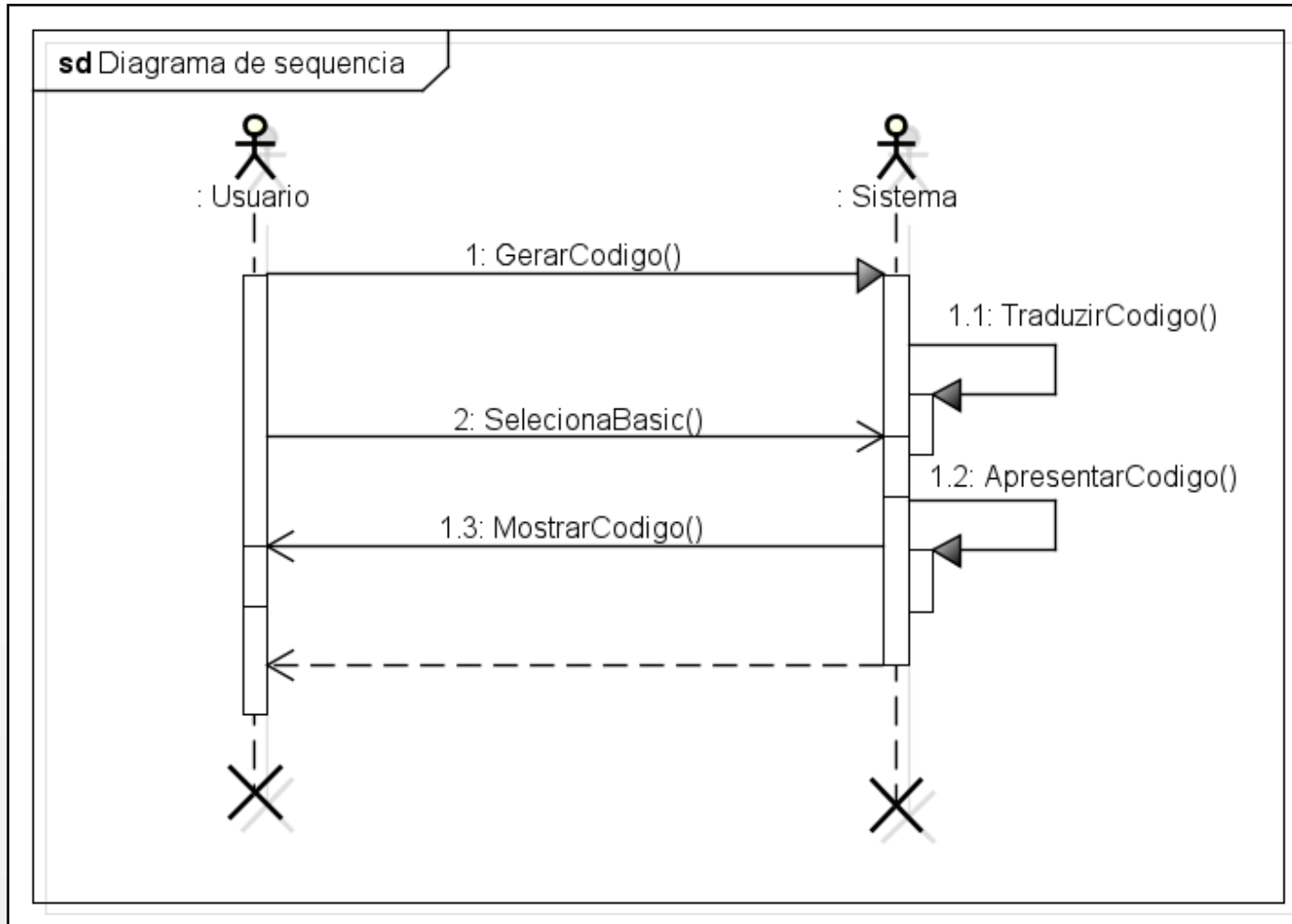
Especificação



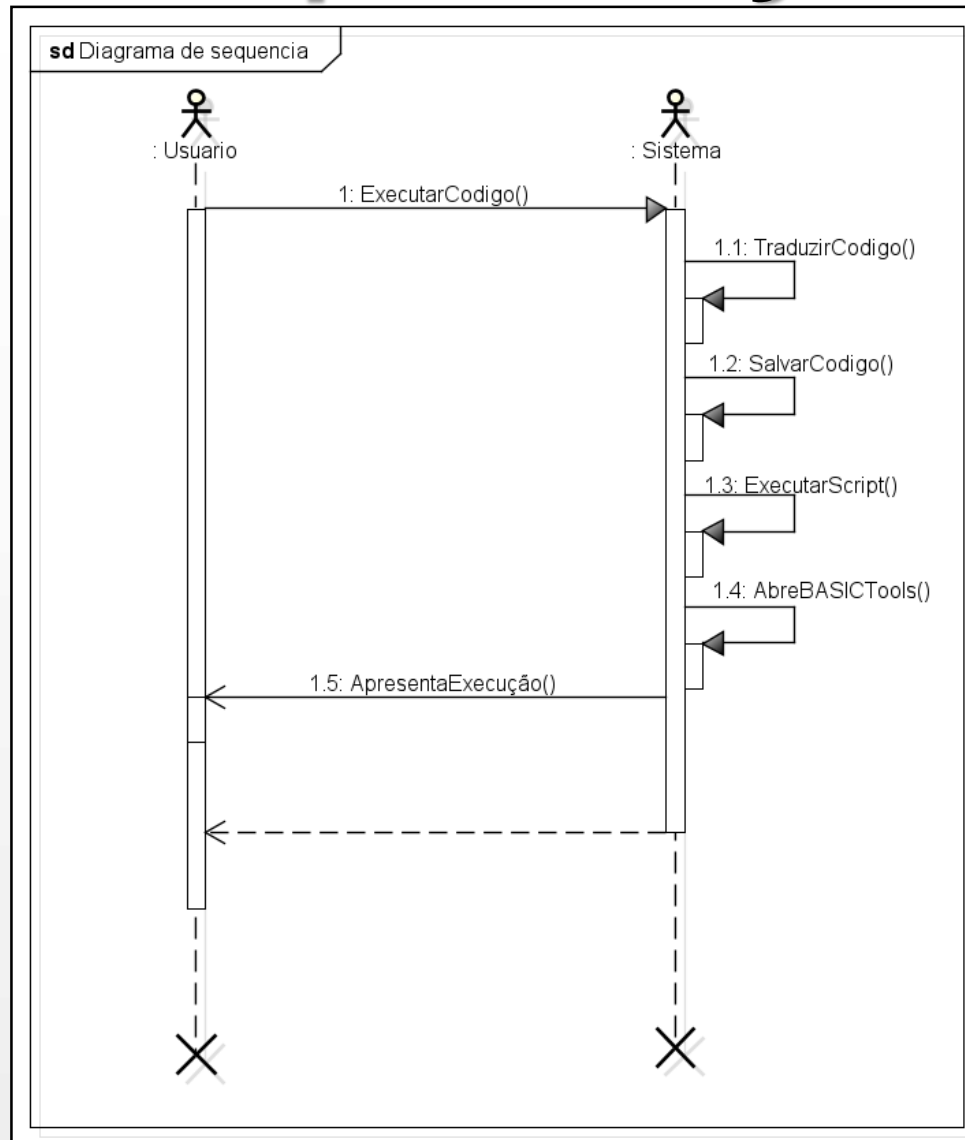
Especificação



Especificação



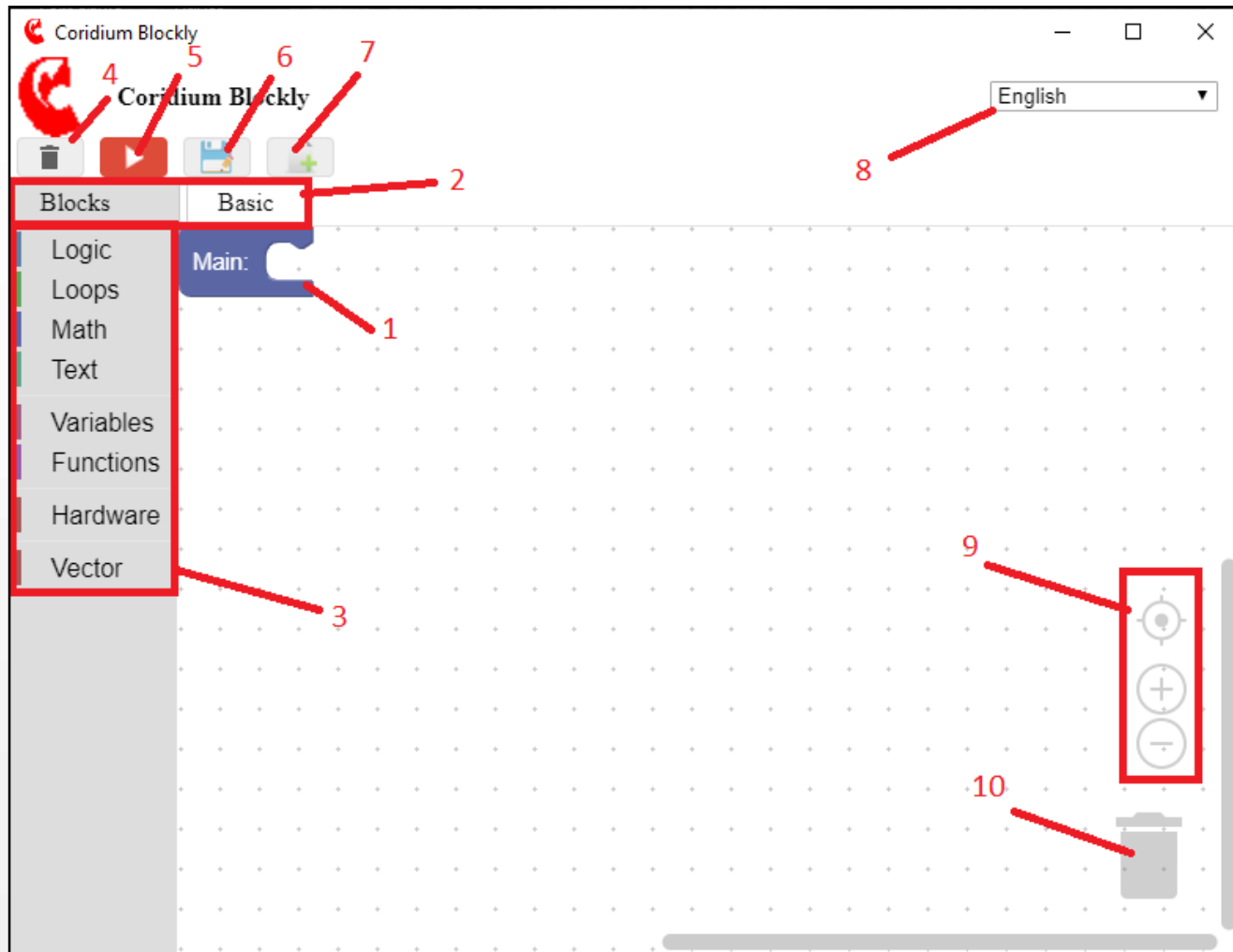
Especificação



Implementação

- Google Blockly;
- BASICtools;
- Electron.

Implementação



Implementação

Blockly > Demos > Blockly Developer Tools

Block Factory | Block Exporter | Workspace Factory

Save "block_type" | Clear Library | Import Block Library

Block Library | Delete "block_type" | Preview: | Download Block Library

Input: name block_type

Field: inputs

Type: automatic inputs

Colour: no connections

tooltip: ""

help url: ""

colour: hue: 230°

Block Definition: JSON

```
{
  "type": "block_type",
  "message0": "",
  "colour": 230,
  "tooltip": ""
}
```

Generator stub: JavaScript

```
Blockly.JavaScript['block_type'] = function(block) {
  // TODO: Assemble JavaScript into code variable
  var code = '...;\n';
}
```

Fonte: Digitalizado pelo autor.

Implementação

```
Blockly.Basic.finish = function(code) {
  // Convert the definitions dictionary into a list.
  var definitions = [];
  var functions = [];
  var teste = [];
  for (var name in Blockly.Basic.definitions_) {
    var teste = [];
    teste = Blockly.Basic.definitions_[name].split(' ');
    if(teste[0] == "function" || teste[0] == "SUB"){
      functions.push(Blockly.Basic.definitions_[name]);
    } else {
      definitions.push(Blockly.Basic.definitions_[name]);
    }
  }
  var defines = [];
  defines.push("const true = 1", "const false = 0", "const HIGH = 1", "const LOW = 0");
  // Clean up temporary data.
  delete Blockly.Basic.definitions_;
  delete Blockly.Basic.functionNames_;
  Blockly.Basic.variableDB_.reset();
  return defines.join('\n')+'\n'+definitions.join('\n')+'\n'+functions.join('\n')+'\n'+code;
};
```

Fonte: Elaborado pelo autor.

Implementação

	Categorias							
	Logic	Loops	Math	Text	Variables	Function	Hardware	Vector
Blocos	IF, else	For	Number	String	Set var	Sub	Wait	DIM Local
	=, !=, <, =<, >, =>	While	Arithmetic	Print text	Get var	Function	AD	DIM Global
	And, OR						Input	Get value in Vector
	Not						Output	Set value in Vector
							IN	
							OUT	
							Pointer	

Implementação

Electron – Estrutura de Arquivos necessárias

```
electron-app/  
├── app  
│   ├── assets  
│   │   └── css  
│   │       └── main.css  
│   ├── main.js  
│   ├── index.html  
│   └── package.json  
├── Gulpfile.js  
└── package.json
```

Fonte: SOSA (2015).

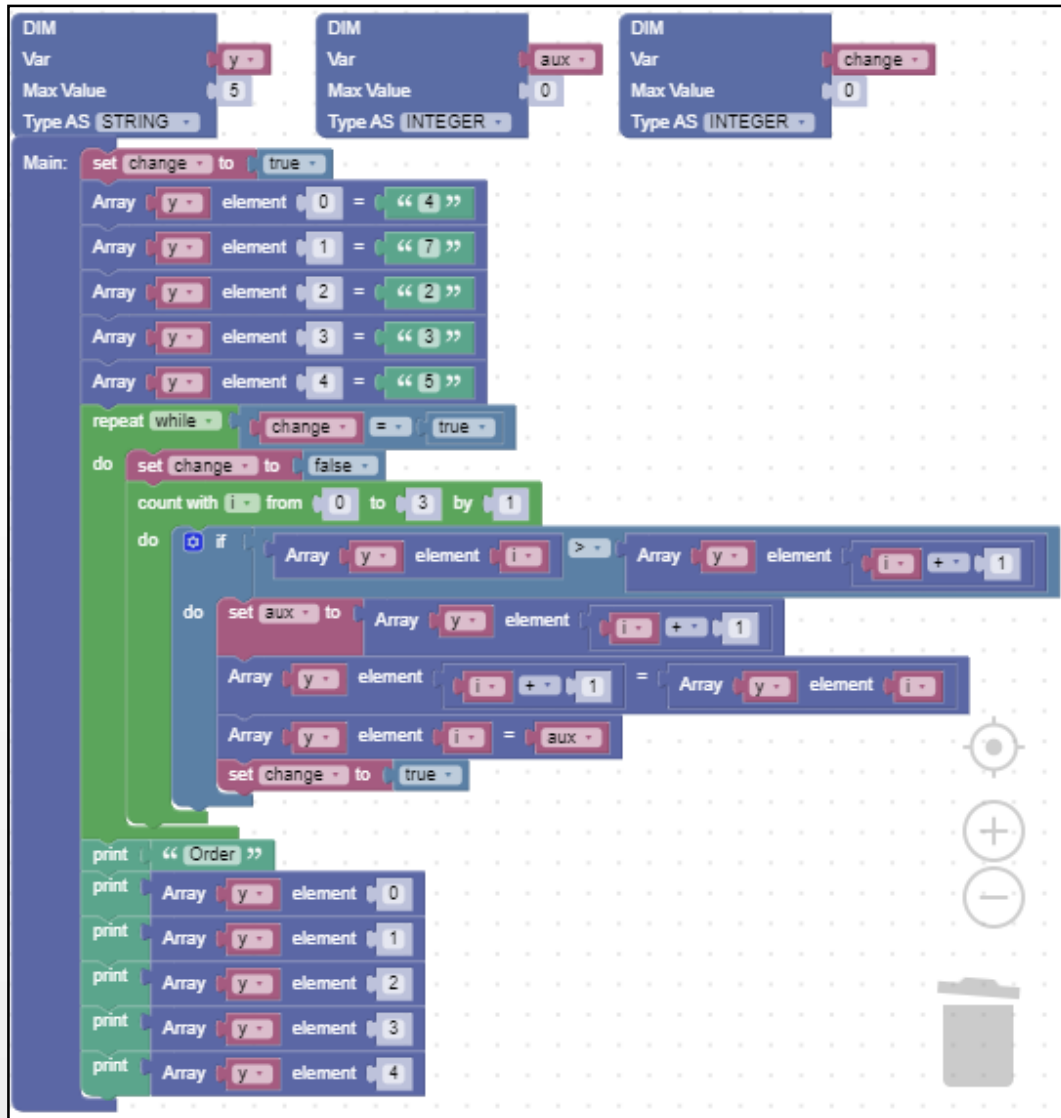
Análise dos Resultados

- Dificuldades;
- 135,476 MB – Renderizador Electron;
- Sistemas operacionais;
- Testes;
- 25 Blocos;
- Teste com 114 Blocos;
- Coridium e sua aprovação.

Análise dos Resultados

correlatos características	Coridium Blockly	Brainpad (2019)	Blynk (2019)	Visuino (2017)
gera código para mais de um dispositivo	X		X	X
tipo de programação	Blockly	Scratch	Gráfica	Gráfica
placas suportadas	Todas com suporte a ARMBasic	Brainpad	Arduino (e Ramificações), ESP8266, Raspberry PI entre outros.	Arduino (e Ramificações), Raspberry PI, ESP8266, entre outros.
plataformas	Web, Android e Desktop	Web, Android e IOs	Android e iOS	Desktop
intuito educacional	X	X	X	
dependente de hardware	X		X	X
linguagem gerada	ARMBasic, JavaScript, Python, PHP, Lua e Dart	Javascript	C++	C++
idiomas	Suporta mais de 40 idiomas	Inglês	Inglês	Inglês

Análise dos Resultados



The image shows a Scratch script for a bubble sort algorithm. It features three DIM blocks at the top for variables 'y', 'aux', and 'change'. The 'y' block is a string array of size 5, 'aux' is an integer of size 0, and 'change' is an integer of size 0. The main script starts with 'set change to true', followed by five 'Array y element' blocks setting values 4, 7, 2, 3, and 5. A 'repeat while' loop with condition 'change == true' contains a 'do' block. Inside the 'do' block, 'set change to false' is followed by a 'count with i from 0 to 3 by 1' loop. Inside this loop, an 'if' block checks 'Array y element i > Array y element i + 1'. If true, it sets 'aux' to 'Array y element i + 1', then 'Array y element i + 1 = Array y element i', and 'Array y element i = aux'. Finally, 'set change to true' is executed. The script ends with six 'print' blocks: 'Order', and elements 0 through 4 of array 'y'.

2 ms

0,37K

93 Blocos

5 Categorias:

- Texto
- Logico
- Loop
- Vetor
- Variaveis

Conclusões e Sugestões

- Cumpriu com os objetivos e requisitos;
- Objetivos: gerar código coerente, carregar um programa na memória do Hardware, processar as entradas e produzir saídas conforme o esperado;
- Complexidade.

Conclusões e Sugestões

- Adicionar novas linguagens de programação na geração de código;
- Desenvolver uma versão para smartphones do Blockly, permitindo assim que a aplicação seja roda em dispositivos moveis e através da rede;
- Implementar mais blocos relacionados a linguagem;
- Utilizar um Web servisse para que o Blockly possa ser manipulado através de um navegador;

Conclusões e Sugestões

- Estruturar documentação: a documentação desenvolvida explica em bastante detalhe a implementação, mas poderia ser melhorada e estruturada para ser usada em sala de aula;
- Uso de softwares *open-source* para melhorar a interface, garantir segurança e integridade dos arquivos.