Desafio IoT

Manual do jogo

Versão: 1.0

Autora:

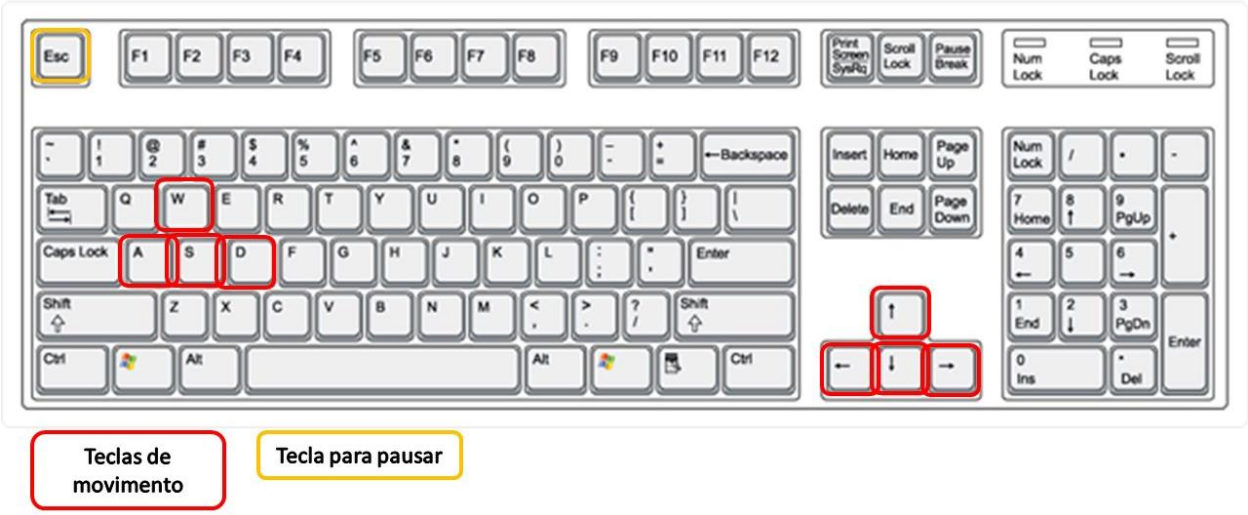
Cleidiana Reis dos Santos

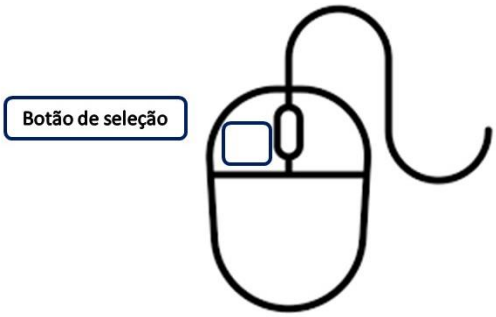
julho de 2022

Índice

1. Controles

Os comandos disponíveis no jogo são apresentados a seguir:





2. Jogabilidade

Cada desafio pode ser resumido em quatro passos, mostrados na Figura 1:

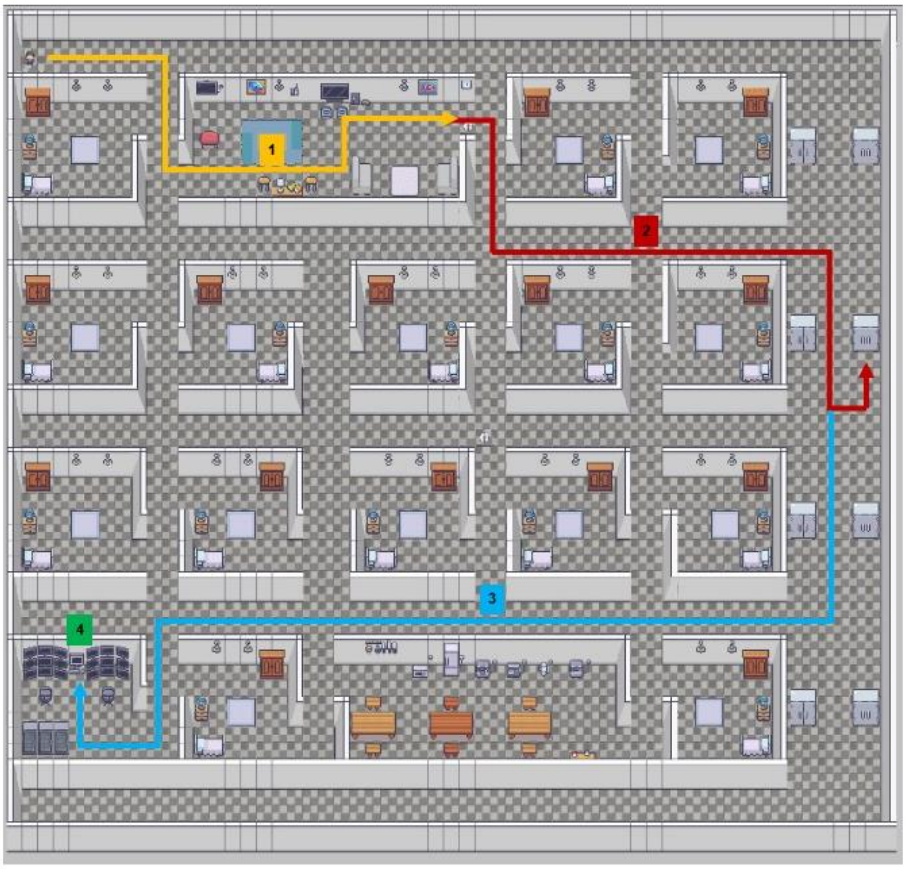


Figura 1 - Passos dos desafios.

1. Ativar desafio – Quando não há nenhum desafio na tela, o jogador deve encontrar um marcador de exclamação no mapa para iniciá-lo;
2. Pegar componentes – Com o desafio definido, o jogador deve escolher (com auxílio das informações da seção 3), se necessário, os componentes que serão usados na solução IoT no armário. Todos os componentes estão em APENAS um armário, que está apontado na Figura 1;
3. Levar até o computador – Com os itens escolhidos, o jogador deve ir até o computador e clicar sobre eles, para adicionar ao projeto.

*Os passos 2 e 3 devem ser realizados dentro de um tempo específico; caso não chegue a tempo no PC, os itens e o desafio são resetados.

4. Programar solução IoT – A tela do computador abre automaticamente (Figura 2). Usando as informações da seção 4, o jogador deve preencher os espaços vazios no código. O martelo é utilizado para *build* e gravação do projeto.

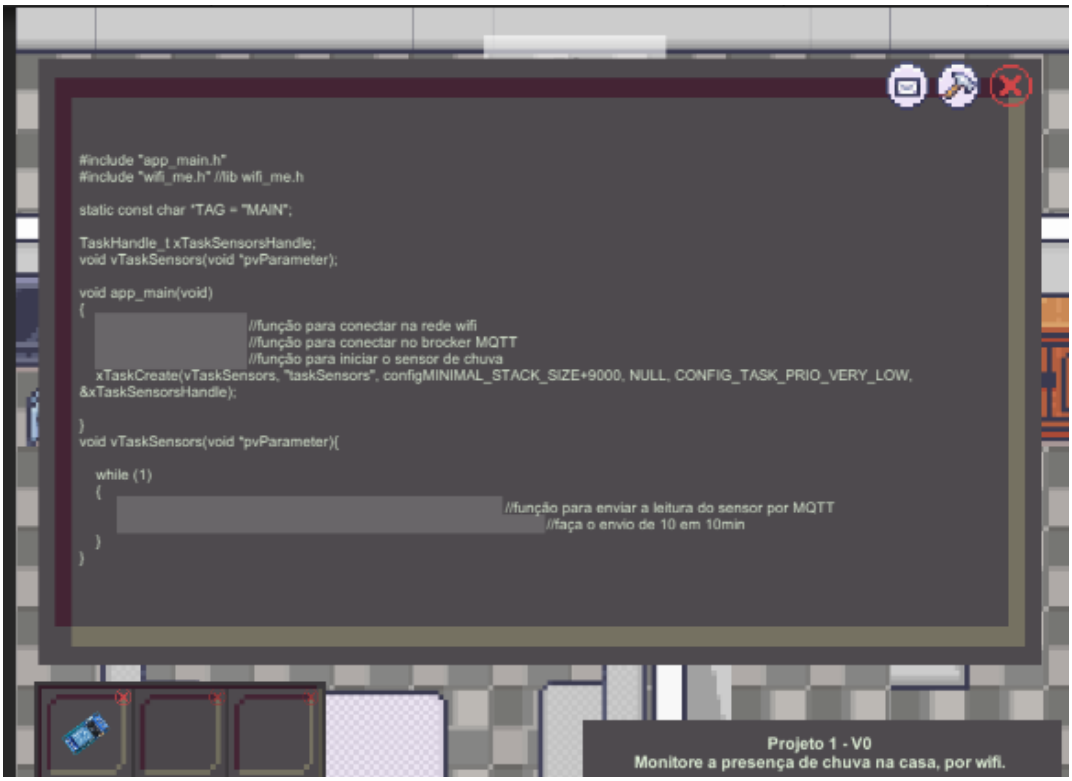


Figura 2 – Computador.

Após concluído o objetivo, deve-se procurar outro desafio para ser ativado, até não existir mais desafios e o jogo ser completado. A Figura 3 apresenta o ciclo do jogo.

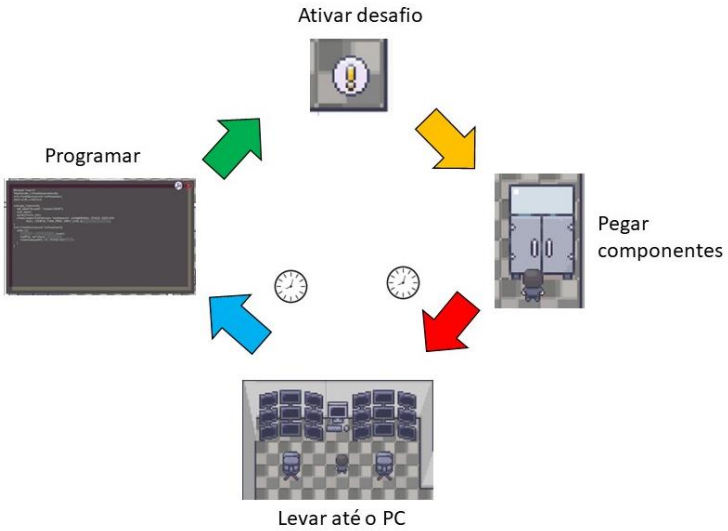


Figura 3 - Ciclo de cada desafio.

3. Componentes/*Hardwares*

O armário de componentes contém os itens que estão listados na Tabela 1, na mesma ordem em que aparecem na Figura 4.

Figura 4 – Inventário.

Tabela 1 - Componentes do inventário

Componente	Ex. Utilização
Relé 3V	Pode ser usado para acionar cargas por microcontroladores. O microcontrolador deve ter a saída de 3.3V.
Sensor de Chuva	Pode ser usado para monitorar se está chovendo ou não em determinada região.
ESP32	Microcontrolador de baixo custo que pode ser usado em projetos de IoT e possui comunicação Bluetooth e Wi-Fi. Sua saída de acionamento é de 3.3V.
Motor	Motor para acionamento de trilhos, como, por exemplo, janelas, portões etc.

DHT22	Pode ser usado para medir a temperatura e umidade do ambiente, para realizar alguma automação.
Buzzer	Emite um alerta sonoro, podendo ser usado como alarme, despertador etc.
Microfone	Pode ser usado como escuta no ambiente. Pode ser utilizado para controle por voz.
Bateria	Pode ser usada para alimentar equipamentos que estejam sem acesso a rede elétrica.
Medidor de corrente elétrica	Realiza a medição da corrente que está passando pelo fio, de forma não invasiva. Pode ser usado para verificar o consumo de determinado equipamento.
Sensor de umidade no solo	Pode detectar a umidade de solo. Pode ser utilizado para monitorar um jardim.
Bomba para irrigação	A bomba consegue impulsionar água para ser usada em irrigação, robôs bombeiros etc.
Relé 5V	Pode ser usado para acionar cargas por microcontroladores. O microcontrolador deve ter a saída de 5V.
RTC Externo	O RTC externo não precisa de nenhum loop associado para incremento. Pode ser usado como relógio para agendamento de tarefas.

Além dos componentes do inventário, será necessária a interação com eletrônicos que já estão na casa, a saber: alimentador elétrico de cachorro, cafeteira de preparo automático, ar-condicionado de baixa corrente e lâmpadas. O alimentador de cachorro precisa permanecer ligado por cinco segundos para a ração deslizar do reservatório para o comedouro do animal; após esse período, DEVE ser desligado. A cafeteira demora cinco minutos para preparar o café; após esse período, DEVE ser desligada. A forma como os relés serão ligados nos pinos do microcontrolador é mostrada na Tabela 2:

Tabela 2 - Pinos utilizados do microcontrolador

Componentes que podem ser controlados por relé	**Pino de saída**
Ar-condicionado	1
Alimentador de cachorro	2
Cafeteira	3
Lâmpadas	4

4. Programação/*Firmware*

Para simplificar, durante o jogo, não é necessário digitar a função completa, sendo preciso apenas usá-la no lugar adequado. O código fonte está incluído no projeto pelo include “main.h”.

4.1 Funções do FreeRTOS[1]:

```
/*
   Handle e protótipo de função do FreeRTOS
   que são necessários para criar a tarefa em FreeRTOS.
*/
TaskHandle_t xTaskSensorsHandle;
void vTaskSensors(void *pvParameter);

/* Função que criar tarefa com FreeRTOS, onde:

   pvTaskCode: Ponteiro para a função a ser executada nesta tarefa;
   pcName: Nome (string) a ser associado a task;
   usStackDepth: Tamanho da stack (em words) reservada para essa tarefa;
   Obs: O tamanho de uma word é do tamanho de bits do microcontrolador;
   uxPriority: Prioridade da tarefa;
         pxCreatedTask: Handle para a tarefa. Este parâmetro é usado para
   suspender e reiniciar a tarefa durante a execução.
*/
BaseType_t xTaskCreate( TaskFunction_t pvTaskCode,
const char * const pcName,
configSTACK_DEPTH_TYPE usStackDepth,
void *pvParameters,
UBaseType_t uxPriority,
TaskHandle_t *pxCreatedTask);

/*
   Função que adiciona delay com FreeRTOS, onde:
   xTicksToDelay: Numero de ticks do microcontrolador que será feito
atraso;

*/
vTaskDelay(const TickType_t xTicksToDelay);
```

4.2 Funções para uso com sensores:

- Sensor de chuva

```
/* Função que inicializa o sensor de chuva */
void sensorChuva_init();
/*
      Função que retorna a medida do sensor de chuva
      retorna 1 está chovendo, e 0 não está.

*/
```

[1] FreeRTOS: https://www.freertos.org/

```
uint8_t sensorChuva_meas();
```

- Sensor de umidade e temperatura

```
/* Função que inicializa o sensor de temperatura e umidade */
void sensorTempUmd_init();
/* Função que retorna a medida do sensor de temperatura ambiente*/
uint8_t sensorTempUmd_measTemp();
/* Função que retorna a medida do sensor de umidade ambiente */
uint8_t sensorTempUmd_measUmd();
```

- Sensor de umidade do solo

```
/* Função que inicializa o sensor de umidade do solo*/
void sensorUmd_init();
/* Função que retorna a medida do sensor de umidade do solo */
uint8_t sensorUmid_meas();
```

- Medidor de corrente elétrica

```
/* Função que inicializa o medidor de corrente */
void sensorCor_init();
/* Função que retorna a medida de corrente elétrica*/
uint8_t sensorCor_meas();
```

4.3 Funções para comunicação e rede:

- Wi-Fi

```
/*
      Função que inicializa o Wi-Fi
      como parâmetro recebe o nome e senha do Wi-fi
      para estabelecer conexão.

*/
void wifi_start(char nome[], char senha[]);
```

- Bluetooth/BLE

```
/* Função que inicializa a comunicação através do Bluetooth Clássico */
void bth_start();
/*
      Função que envia mensagem por Bluetooth Clássico,
      como parâmetro recebe a mensagem e o valor da medida.

*/
void bthPub_val(char msg[], uint8_t val);
/*
      Função que inicializa a comunicação através do Bluetooth Low
      Energy (BLE)
*/
```

```
void ble_start();
/*
      Função que envia mensagem por BLE,
      como parâmetro recebe a mensagem e o valor da medida.

*/
void blePub_val(char msg[], uint8_t val);
```

- MQTT

```
/* Função que inicializa os recursos MQTT*/
void mqtt_start();
/*
      Função que envia mensagem por Wi-Fi, via MQTT,
      como parâmetro recebe a mensagem e o valor da medida.

*/
void mqttPub_val(char msg[], uint8_t val);
```

4.4 Funções para componentes de saída:

- Relé

```
/* Função que inicializa todos os pinos de saída que podem ser usados
para relé */
void rele_init();
/*
      Função que retorna o valor do estado do pino passado como
parâmetro,
      1 caso esteja com valor alto e 0 caso contrario
*/
uint8_t releGet_state(int pino);
/*
      Função para configurar o estado do pino passado como parâmetro.
      val: 1 para ligado e 0 para desligado.
      timer: período transitório (em segundos), caso a configuração
      deva permanecer deve ser 0.
*/
void releSet_state(int pino, int val, int timer);
```

- Buzzer

```
/* Função que inicializa o buzzer */
void buzzer_init();
/*
      Função para ligar e desligar o buzzer.
      val: 1 para ligado e 0 para desligado.
      timer: período transitório (em segundos), caso a configuração
      deva permanecer deve ser 0.
*/
void buzzerSet_state(int val, int timer);
```

- Motor

```
/* Função que inicializa o motor */
void motor_init();
/*
      Função para ligar e desligar o motor.
      val: 1 para ligado e 0 para desligado.
*/
void motorSet_rot(uint8_t val);
```

- Bomba de irrigação

```
/* Função que inicializa a bomba de irrigação */
void irrig_init();
/*
      Função para ligar e desligar a bomba de irrigação.
      val: true para ligado e false para desligado.
*/
void irrigSet_state(bool val);
```

4.5 Funções para o microfone e RTC:

- RTC externo

```
/* Função que inicializa o RTC externo com o horário atual */
void RTC_init();
/* Função que retorna a hora atual*/
uint8_t RTCGet_hr();
/* Função que retorna o minuto atual*/
uint8_t RTCGet_min();
```

- Microfone

```
/* Função que inicializa o microfone*/
void microf_init();
/*
      Função que retorna 1 está foi dito para ligar as lâmpadas,
      e 0 foi para desligar.

*/
uint8_t microLamp_meas();
```

Informações complementares para o projeto:

- A Rede Wi-Fi possui NOME: Ronaldo, SENHA: ronaldo12345;
- A leitura do microfone DEVE ser realizada de um em um segundo, e a dos sensores de 10 em 10 minutos;

- Os componentes de saída DEVEM ser ligados ou desligados em cinco segundos após o evento relacionado aos sensores e microfone acontecer;
- Os componentes de saída DEVEM ser ligados ou desligados em dois minutos após o evento relacionado ao tempo (RTC) acontecer;
- Quando não informado, o projeto será alimentado pela rede elétrica e DEVE usar o Bluetooth clássico;
- O tamanho da memória reservada para tarefas de leitura dos sensores DEVE ser configMINIMAL_STACK_SIZE+256 e para tarefas dos atuadores DEVE ser configMINIMAL_STACK_SIZE+512.
- DEVE ser adicionado o *delay* por FreeRTOS no seguinte formato pdMS_TO_TICKS(1000)*val_segundos, pdMS_TO_TICKS(1000) representa o valor de 1s para o microcontrolador.