



**POLITÉCNICO
DE LEIRIA**

iModDom



Tutorial – Utilização dos Blocos

Elaborado por:

Marco Pereira - 2190516

Orientado por:

Luís Bento

Carlos Neves

Índice

1. INTRODUÇÃO.....	1
1.1. OBJETIVOS.....	1
2. UTILIZAÇÃO DOS BLOCOS.....	2
2.1. UTILIZAÇÃO DOS BLOCOS <i>BEGINNER</i>	4
2.1.1. <i>Binary Sensor</i>	4
2.1.2. <i>Control a Light</i>	8
2.1.3. <i>Control a Cover</i>	10
2.1.4. <i>Temperature and Humidity Control With DHT</i>	13
2.2. UTILIZAÇÃO DOS BLOCOS <i>ADVANCED</i>	15
2.2.1. <i>Blocos Base</i>	15
2.2.1.1. <i>ESPHome</i>	15
2.2.1.2. <i>Wifi</i>	16
2.2.1.3. <i>IP</i>	16
2.2.1.4. <i>Ethernet</i>	16
2.2.1.5. <i>API</i>	17
2.2.1.6. <i>OTA</i>	17
2.2.1.7. <i>Logger</i>	18
2.2.1.8. <i>I2C</i>	19
2.2.1.9. <i>Web Server</i>	19
2.2.1.10. <i>PCF8574</i>	19
2.2.2. <i>Blocos Sensor</i>	20
2.2.2.1. <i>Binary Sensor</i>	20
2.2.2.2. <i>Binary Sensor (componente)</i>	21
2.2.2.3. <i>Binary Sensor PCF8574</i>	21
2.2.2.4. <i>PIR Sensor</i>	22
2.2.2.5. <i>PIR Sensor PCF8574</i>	23
2.2.2.6. <i>Automations</i>	23
2.2.2.7. <i>On Click</i>	24
2.2.2.8. <i>On Double Click</i>	24
2.2.2.9. <i>On Multi Click</i>	25
2.2.2.10. <i>Binary Sensor Action</i>	26
2.2.2.11. <i>CAN bus Send Action</i>	27

2.2.2.12.	Sensor	28
2.2.2.13.	DHT Sensor	28
2.2.2.14.	SI7021 Sensor	29
2.2.2.15.	BMP280 Sensor.....	29
2.2.2.16.	MQ-7	30
2.2.2.17.	Template Sensor	30
2.2.3.	Blocos Switch.....	32
2.2.3.1.	Switch	32
2.2.3.2.	Switch (componente).....	33
2.2.3.3.	Switch PCF8574	34
2.2.3.4.	Restart Switch.....	35
2.2.3.5.	Shutdown Switch	35
2.2.4.	Blocos Output.....	35
2.2.4.1.	Output	36
2.2.4.2.	Output (componente)	36
2.2.4.3.	Output PCF8574	36
2.2.5.	Blocos Actuators.....	37
2.2.5.1.	Light.....	37
2.2.5.2.	Light (componente)	37
2.2.5.3.	Fan	38
2.2.5.4.	Binary Fan.....	38
2.2.5.5.	Cover	39
2.2.5.6.	Cover (componente)	39
2.2.5.7.	Endstop Cover.....	40
2.2.5.8.	Cover Action	41
2.2.6.	Blocos Communication.....	41
2.2.6.1.	SPI.....	41
2.2.6.2.	CAN Bus	42
2.2.6.3.	CAN bus Receive Action	43

1. Introdução

Para facilitar a criação de código a todos os utilizadores, foram criados blocos que geram esse mesmo código de uma forma bastante intuitiva. Haverá pequenos passos que não serão explicados neste tutorial por já terem sido explicados em outros tutoriais.

1.1. Objetivos

O objetivo com este tutorial é ensinar a utilizar esses mesmos blocos e como tirar partido do código gerado.

2. Utilização dos Blocos

Para utilizar os vários blocos do projeto e assim gerar o seu próprio código deverá seguir os seguintes passos.

1. Aceda à página do *Home Assistant* e de seguida ao separador “iModDom” de modo a ter acesso a página dos blocos (Figura 1).

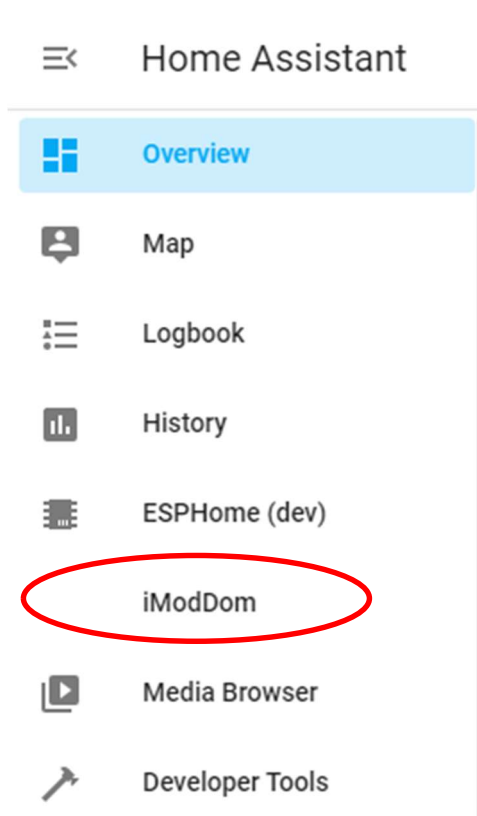


Figura 1 - Separador "iModDom"

2. Na página aberta, começando pelo canto superior esquerdo, encontram-se vários logótipos, entre eles o do *GitHub*, se clicar sobre o mesmo será direcionado para o repositório do projeto, o logótipo do INESC Coimbra, em que se clicar sobre ele será direcionado para a respetiva página, o logótipo do Instituto Politécnico de Leiria, em que se clicar sobre o mesmo irá ser direcionado para a página da Escola Superior de Tecnologia e Gestão, por último existe o logótipo do *Blockly* que o direcionará para o repositório do mesmo. Ainda no canto superior esquerdo existem vários botões, sendo que o primeiro permite fazer o *download* do *playground*, de modo a salvar todo o trabalho realizado, o segundo botão permite fazer o *upload* do *playground* para que seja possível editar trabalhos salvos anteriormente, de seguida existe o botão que

permite guardar o código gerado num ficheiro *yaml*, o quarto botão tem o objetivo de gerar o código através dos blocos adicionados ao *playground*, através do quinto botão é possível copiar todo o conteúdo contido na caixa de texto e o ultimo botão permite apagar todo o conteúdo da caixa de texto e do *playground*. Abaixo dos botões existe uma caixa de texto onde será apresentado o código gerado e do lado direito da caixa de texto existem as várias abas que contêm os blocos para gerar código, estando estes divididos em “*Beginner*” e “*Advanced*”, tal como o nome indica, a primeira aba contém blocos básicos e de fácil utilização e a segunda já contém blocos mais avançados e completos. Do lado direito das abas dos blocos existe o *playground*, local onde colocará os blocos, no canto inferior direito existe um ícone de caixote do lixo, através do qual poderá apagar os seus blocos, arrastando-os para lá, existem ainda três ícones, sendo que o primeiro tem o objetivo de centrar os blocos na página, o segundo tem a função de *zoom in* e o terceiro a função de *zoom out*.

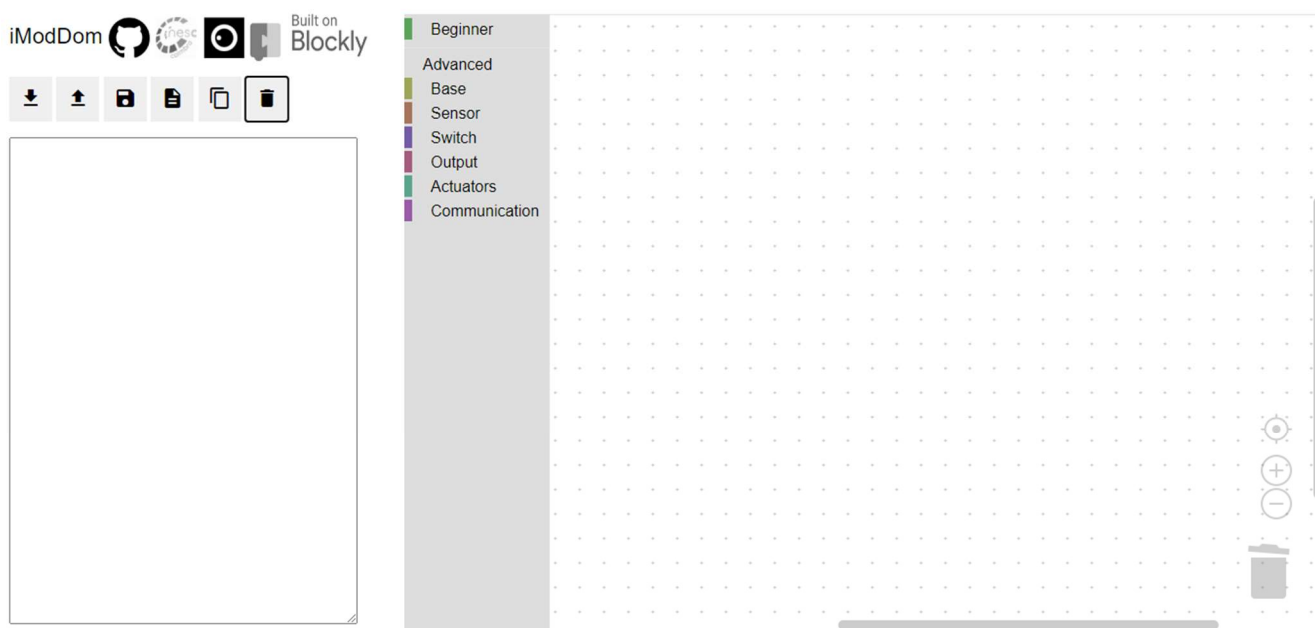


Figura 2 – Página “iModDom”

2.1. Utilização dos Blocos *Beginner*

Estes blocos permitem que o utilizador, de uma forma bastante fácil e intuitiva, utilize pela primeira vez blocos para gerar código. Os blocos desta secção só podem ser utilizados individualmente, isto é, aquando da utilização de um bloco deste tipo não se pode utilizar nenhum outro em simultâneo.

2.1.1. *Binary Sensor*

Este bloco permite gerar código para a utilização de um sensor binário, como é o caso de um botão por exemplo.

1. Comece por aceder ao separador “*Beginner*” e de seguida arraste o bloco para o *workspace* (Figura 3).

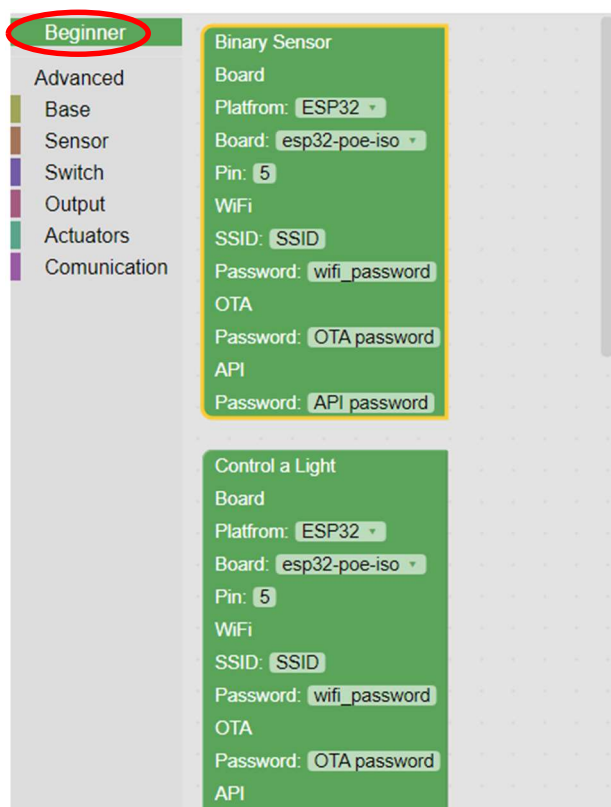


Figura 3 - Seleção do bloco

2. Preencha todos os campos do bloco, sendo que no primeiro e segundo deve seleccionar o seu *ESP*, no terceiro campo (“*Pin*”), introduzida o número do pino onde irá ligar o seu botão, no campo “*SSID*” deverá introduzir o nome da rede WiFi à qual se deseja conectar e no campo “*Password*” a palavra-

chave dessa mesma rede. De seguida deve colocar uma *password* para as atualizações por WiFi e por último uma *password* para poder integrar o código no *Home Assistant* (Figura 4).



Figura 4 - Aspeto final do bloco

3. Clique no ícone do ficheiro para gerar o código a ser utilizado e de seguida clique no ícone de copiar para copiar todo o código (Figura 5).

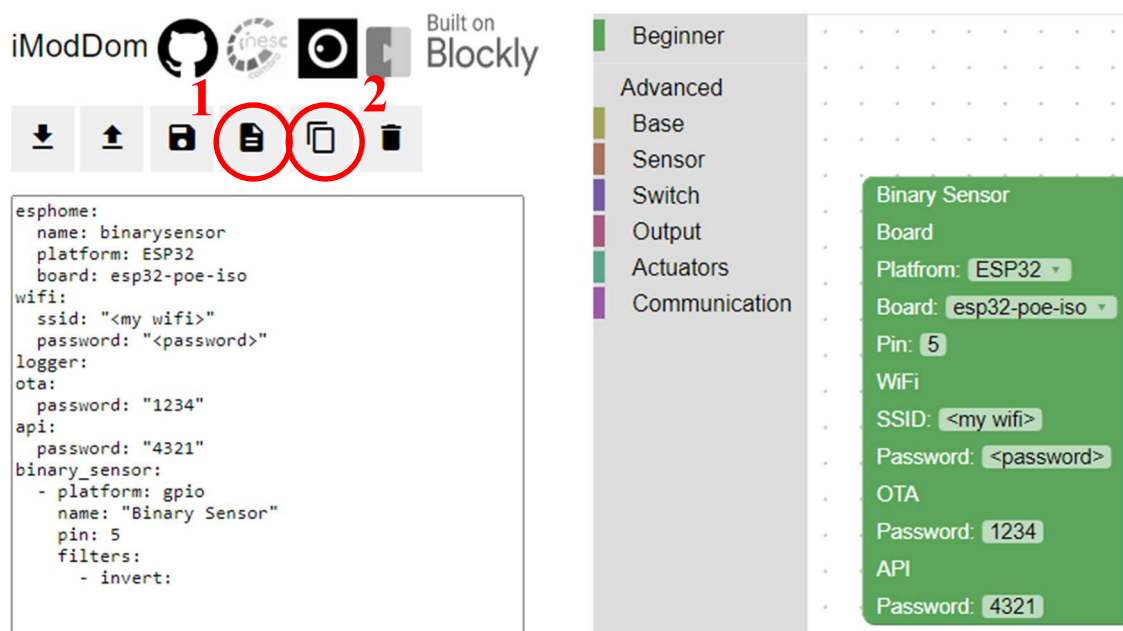


Figura 5 - Gerar e Copiar o Código

4. Com o código gerado e copiado, vá até ao *ESPHome* e clique em “*EDIT*” no ficheiro criado no tutorial de instalação do *ESPHome* no *Home Assistant* (Figura 6).

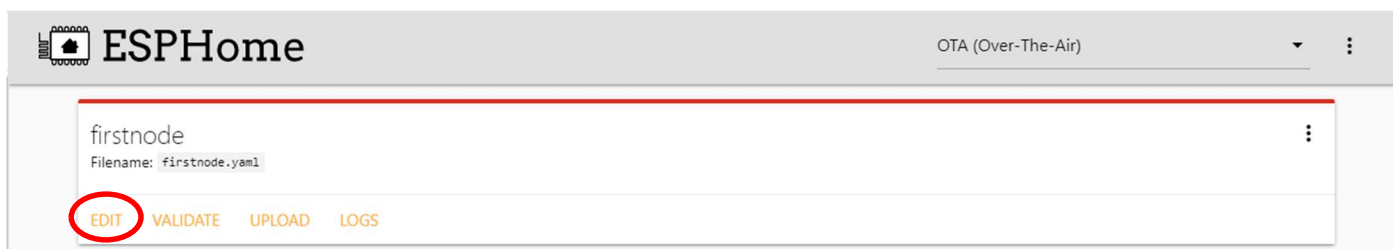


Figura 6 - Editar o Ficheiro Criado

5. Na janela que lhe surge, elimine o conteúdo do ficheiro e cole o código que copiou anteriormente e clique em “*SAVE*” e de seguida em “*CLOSE*” (Figura 7).



Figura 7 - Aspeto do Ficheiro

6. Agora, conecte a sua placa ao *Raspberry Pi*, selecione a porta no canto superior direito e clique em “*UPLOAD*” (Figura 8).



Figura 8 - Upload do Ficheiro

7. Quando o *upload* estiver concluído, adicione a nova integração e de seguida vá até ao painel principal em “*Overview*” e adicione um novo cartão com o sensor binário que está presente no código (Figura 9).

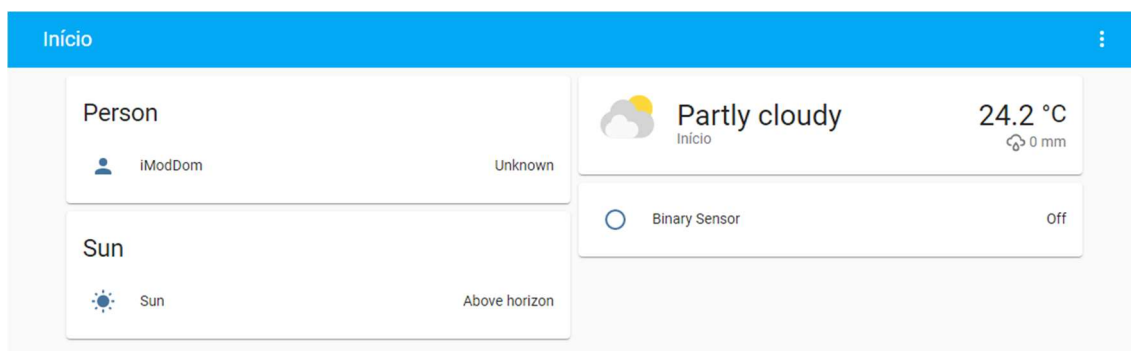


Figura 9 - Aspeto Final do Painel Principal

8. Faça as ligações no seu *ESP* de acordo com a Figura 10 e com o pino que definiu anteriormente.

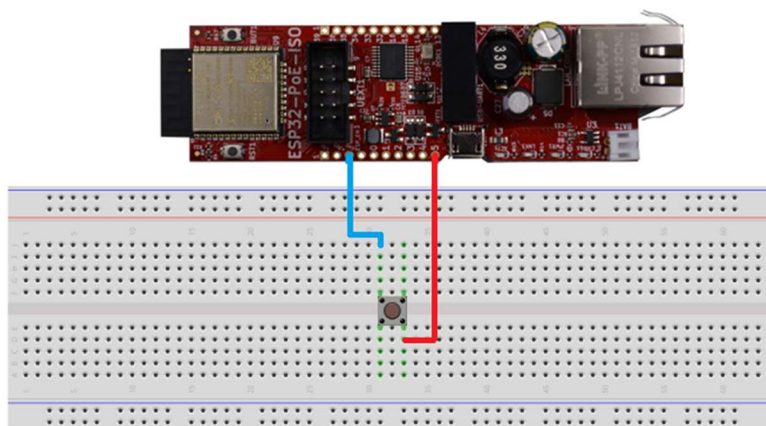


Figura 10 - Esquema de Ligação do Botão

9. Verifique que ao clicar no botão o estado do sensor binário no *Home Assistant* altera (Figura 11).

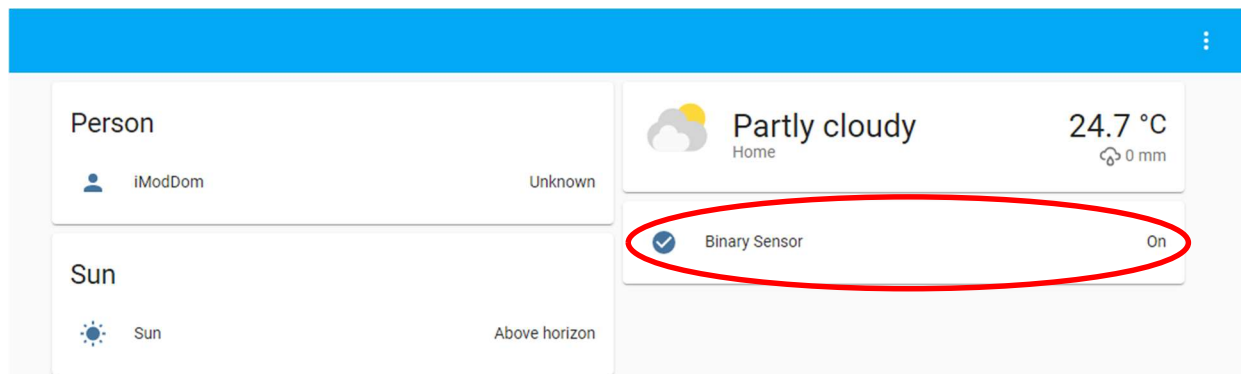


Figura 11 - Botão Acionado

2.1.2. *Control a Light*

Este bloco gera o código que permite controlar um LED.

1. Comece por arrastar o bloco “*Control a Light*” para o *workspace* e preencha todos os campos do bloco com as mesmas informações do bloco anterior, sendo que neste bloco, o pino definido será para ligar o LED (Figura 12).

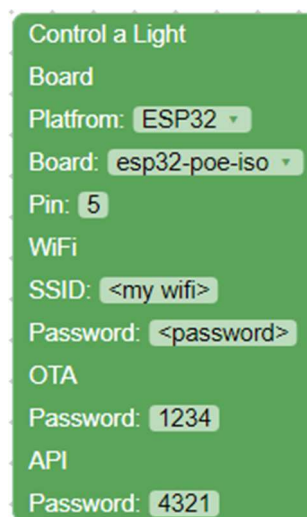


Figura 12 - Aspeto do Bloco

2. Clique no ícone para gerar o código e note que o código gerado apresenta diferenças face ao anterior, uma vez que têm funções diferente (Figura 13).

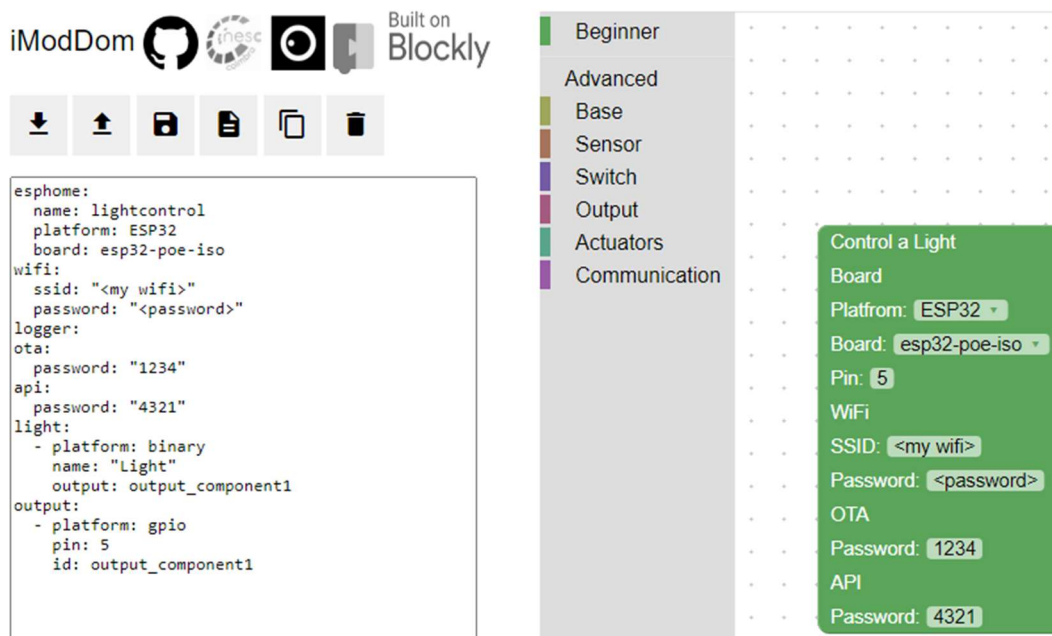


Figura 13 - Código Gerado

3. Copie o código, altere o conteúdo do seu *node* e repita os passos definidos para o bloco anterior até ter o cartão do LED no seu painel principal.
4. Siga as ligações da Figura 14 em que o valor da resistência deverá estar compreendido entre 230Ω e 500Ω.

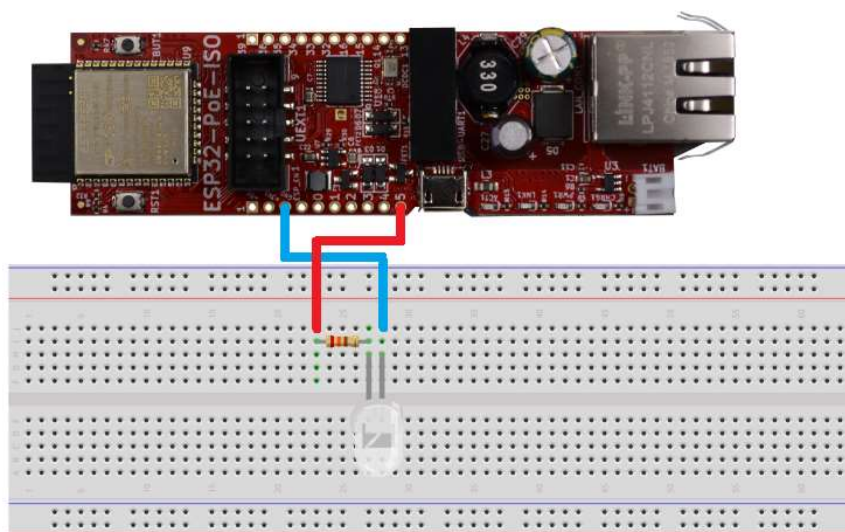


Figura 14 - Esquema de Ligação do LED

5. Como pode verificar, ao clicar no cartão do LED o estado do mesmo irá alterar (Figura 15).

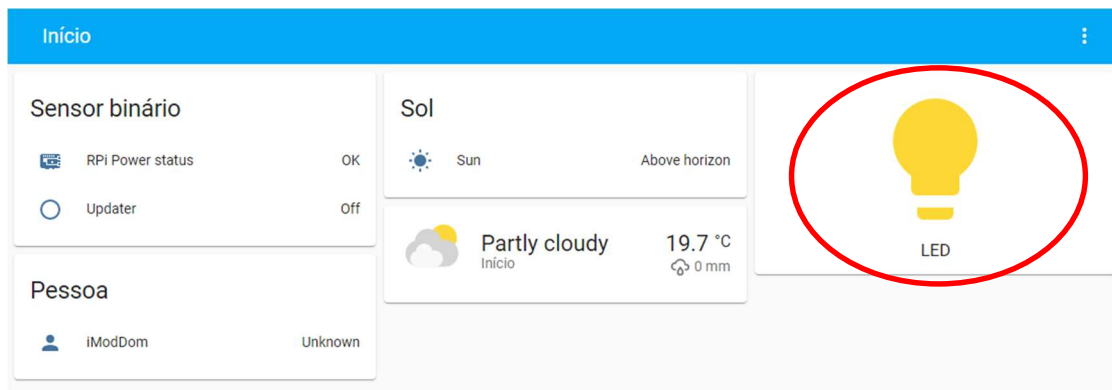


Figura 15 - Estado do LED

2.1.3. *Control a Cover*

O bloco “*Control a Cover*”, tal como o nome indica, permite controlar uns estores elétricos, desde a sua abertura, paragem e encerramento. Neste bloco em específico os estores têm uma duração de abertura de 5 segundos e outros 5 segundos para o seu encerramento.

1. Arraste o bloco para o *workspace* e preencha todos os campos, tal como nos blocos anteriores, à exceção de que neste bloco existem cinco pinos a definir. O primeiro pino “*Pin to open*” será o pino no qual irá ligar o botão que irá abrir os estores, o segundo “*Pin to close*” o pino onde ligará o botão que permite fechar os estores, o terceiro “*Pin to stop*” onde se ligará o botão para parar os estores, no quarto pino “*Open switch*” ligará a saída que permite abrir os estores e no último “*Close switch*”, a saída que irá fechar os estores (Figura 16).

Control a Cover

Board

Platform: ESP32

Board: esp32-poe-iso

Pin to open: 0

Pin to close: 1

Pin to stop: 2

Open switch: 3

Close switch: 4

WiFi

SSID: <my wifi>

Password: <password>

OTA




Password: 1234

API

Password: 4321

Figura 16 - Aspeto do Bloco

2. Agora com o bloco pronto, pode gerar o código, que será mais extenso e complexo face aos anteriores, e repita todos os passos até ter adicionado os três botões e os dois *switch* ao painel principal (Figura 17).

iModDom    Built on Blockly

Download Upload Save Copy Delete

```

esphome:
  name: covercontrol
  platform: ESP32
  board: esp32-poe-iso
wifi:
  ssid: "<my wifi>"
  password: "<password>"
logger:
ota:
  password: "1234"
api:
  password: "4321"
binary_sensor:
  - platform: gpio
    name: "Open Cover"
    pin:
      number: 3
      inverted: True
      mode: INPUT_PULLUP
    on_click:
      min_length: 0s
      max_length: 0.5s
      then:
        - cover.open: first_cover
  - platform: gpio
    name: "Close Cover"
    pin:
      number: 4
      inverted: True
      mode: INPUT_PULLUP
    on_click:
      min_length: 0s
      max_length: 0.5s
  
```

Beginner

Advanced

Base

Sensor

Switch

Output

Actuators

Communication

Control a Cover

Board

Platform: ESP32

Board: esp32-poe-iso

Pin to open: 3

Pin to close: 4

Pin to stop: 5

Open switch: 13

Close switch: 14

WiFi

SSID: <my wifi>

Password: <password>

OTA

Password: 1234

API

Password: 4321

Figura 17 - Código Gerado

6. Faça as seguintes ligações para poder simular o controlo dos estores elétricos, onde os LEDs simulam a abertura e fecho dos mesmos (Figura 18). As duas resistências devem ter um valor compreendido entre 230Ω e 500Ω .

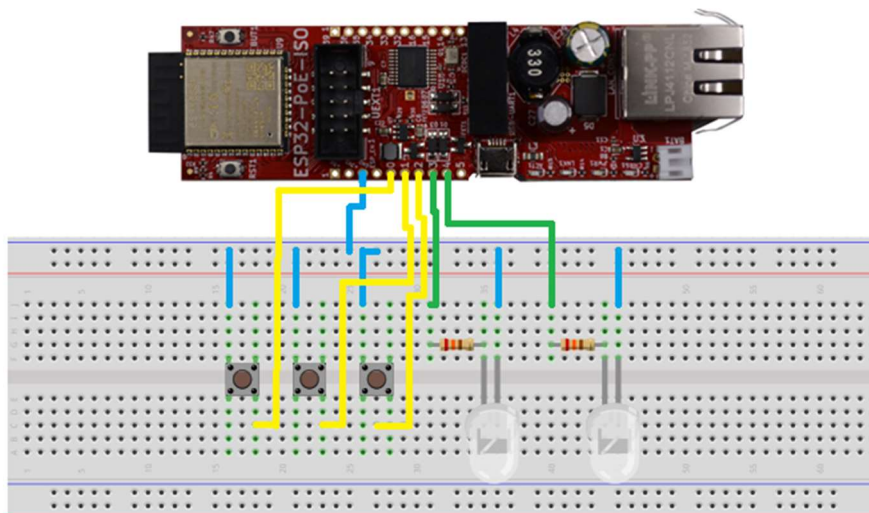


Figura 18 - Esquema de Ligações

3. Repare que agora, tanto pode controlar os estores elétricos clicando nos botões que ligou ao seu *ESP* como clicando nos botões apresentados no *Home Assistant* (Figura 19). Note que só pode clicar nos botões no máximo durante 0.5 segundos para que as ações de abertura, encerramento ou paragem sejam efetuadas. Repare que se clicar no botão para abrir e de seguida para fechar ou vice versa, o que prevalece será sempre a última ordem para evitar que as duas sejam efetuadas em simultâneo, o que resultaria numa avaria do motor num caso prático.

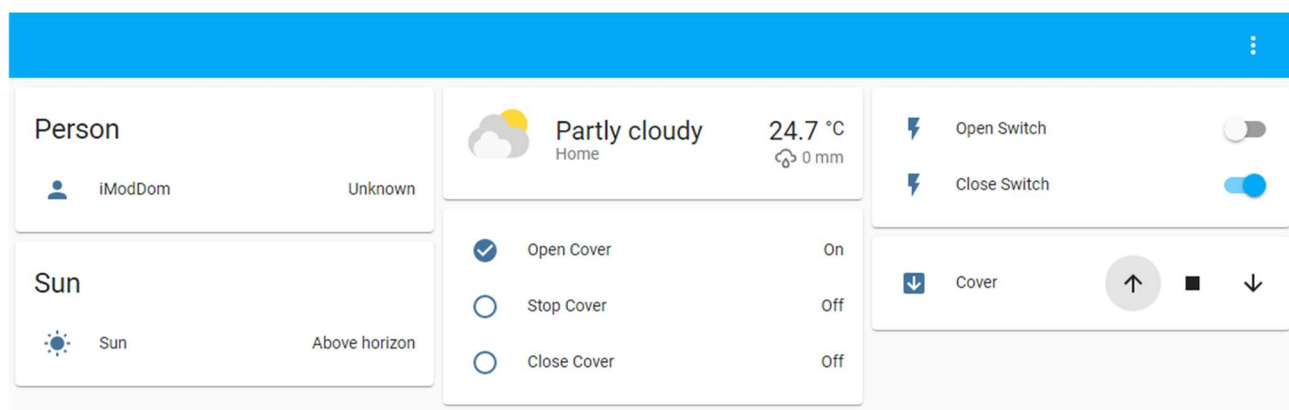


Figura 19 - Funcionamento dos Estores Elétricos

2.1.4. Temperature and Humidity Control With DHT

Através deste bloco é possível fazer leituras da temperatura e humidade com o sensor DHT, tanto o DHT11 como o DHT22, e visualizar os valores no *Home Assistant*.

1. Comece por arrastar o bloco para o *workspace* e de seguida preencha os campos do mesmo tal como fez nos primeiros blocos, no campo “Pin” deve definir o pino ao qual vai ligar a saída do seu sensor DHT (Figura 20).

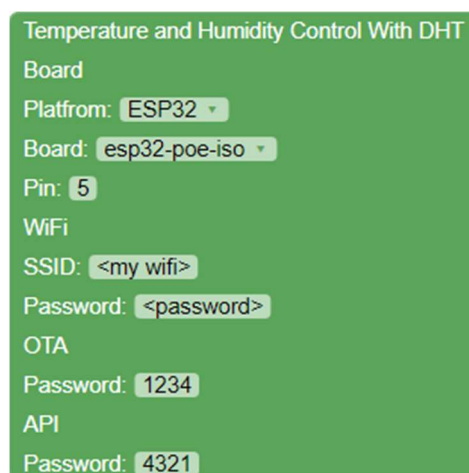


Figura 20 - Aspeto do Bloco

2. Gere o código e execute todos os passos até ter o cartão da humidade e da temperatura apresentados no painel principal (Figura 21).

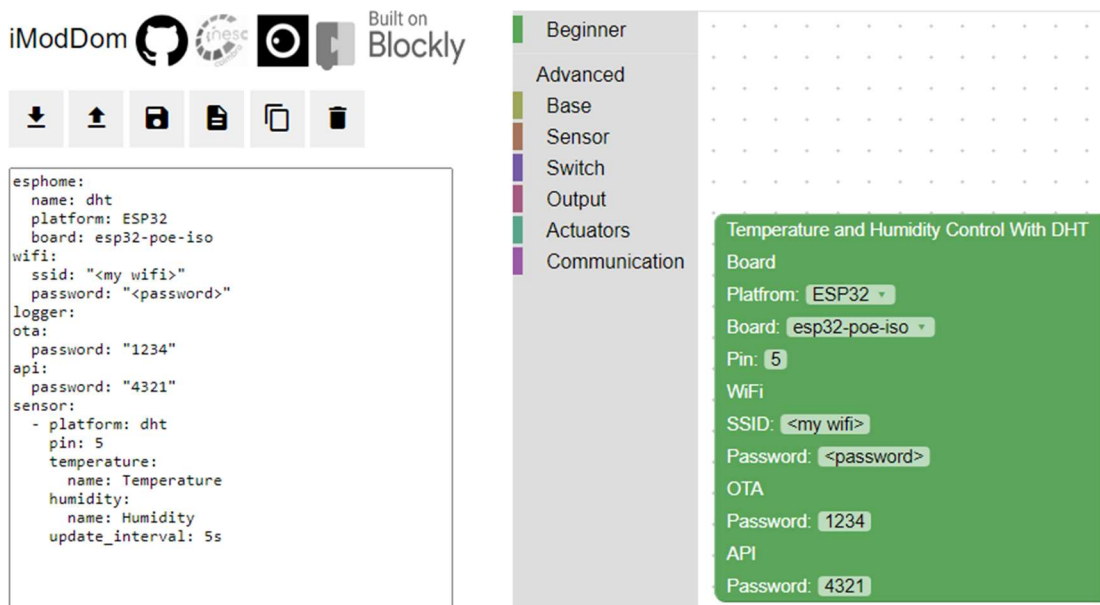


Figura 21 - Código Gerado

3. Ligue o sensor seguindo o esquema de ligações da Figura 22.

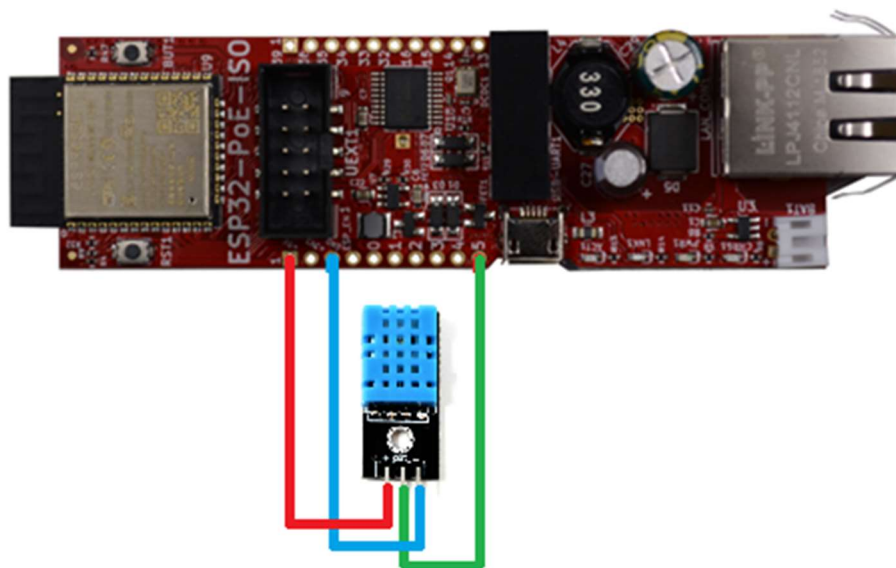


Figura 22 - Esquema de Ligações

4. Como pode confirmar, acendendo ao *Home Assistant*, pode visualizar os valores de temperatura e humidade fornecidos pelo sensor (Figura 23).

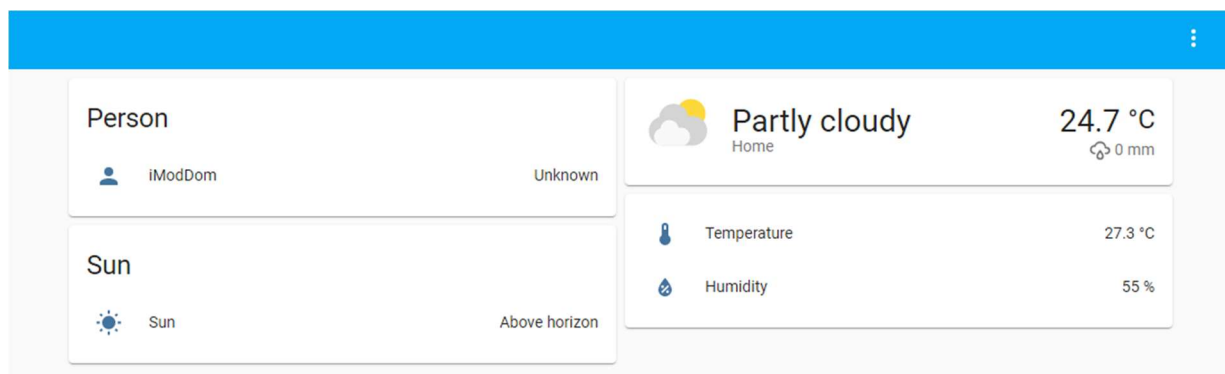


Figura 23 - Visualização dos Valores

2.2. Utilização dos Blocos *Advanced*

Todos os blocos que estão incluídos na secção “*Advanced*”, já são mais complexos face à secção apresentada anteriormente e permitem gerar código mais completo.

2.2.1. Blocos *Base*

Na secção “*Base*” estão todos os blocos que geralmente são utilizados em todos os *nodes* ou aqueles que são mais gerais e não se incluem numa secção específica. Todos os blocos desta secção só podem ser utilizados uma única vez e são do tipo “principais”, excetuando o bloco “*I2C*” que pode ser utilizado várias vezes.

2.2.1.1. *ESPHome*

Este bloco tem a função de atribuir um nome ao *node*, definir qual a plataforma a placa a utilizar (Figura 24). Note que a escolha da placa de desenvolvimento irá afetar a designação dos pinos, pelo que deve colocar a designação dos mesmos conforme se encontra na placa.

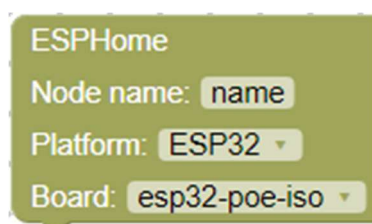


Figura 24 – *ESPHome*

2.2.1.2. Wifi

O bloco “*Wifi*” permite realizar a conexão ao WiFi, preenchendo no primeiro campo (“*SSID*”) com o nome da rede WiFi a conectar, e no último campo (“*Password*”) a *password* da rede. Este bloco tem ainda um encaixe lateral que permite, através de um outro bloco, definir o IP manualmente. Este bloco não pode ser utilizado se o bloco “*Ethernet*” estiver a ser utilizado.

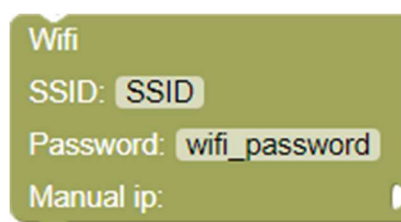


Figura 25 – *Wifi*

2.2.1.3. IP

Através deste bloco é possível configurar o IP do *node*, colocando no primeiro campo (“*Static ip*”) o IP que deseja definir, no campo “*Gateway*” o *gateway* da sua rede e no último campo (“*Subnet*”), a *subnet* da sua rede. Este bloco apenas pode ser encaixado e utilizado com o bloco “*Wifi*” ou com o bloco “*Ethernet*”.

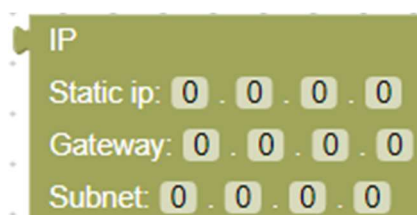


Figura 26 – *IP*

2.2.1.4. Ethernet

O bloco “*Ethernet*” tem a função de permitir uma ligação à rede via *ethernet*, onde no primeiro campo se define o *chip* de *ethernet*, no segundo campo (“*MDC pin*”) define-se o pino *MDC*, no terceiro campo (“*MDIO pin*”) é definido o pino *MDIO*, no quarto campo (“*Clock mode*”) selecciona-se o modo do *clock* e no campo “*Power pin*” define-se o pino através do qual controla a alimentação da placa. Este bloco contém

ainda um encaixe lateral onde se pode encaixar o bloco “IP”, para configurar o IP manualmente. Este bloco não pode ser utilizado se o bloco “Wifi” estiver a ser utilizado.

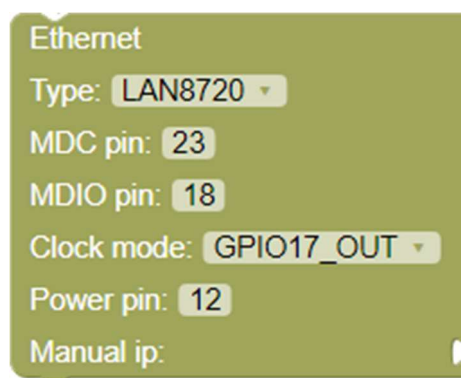


Figura 27 - Ethernet

2.2.1.5. API

O bloco “API”, tem a função de possibilitar a integração com o *Home Assistant*, sendo que para isso basta introduzir uma *password* no campo “API password” do bloco. Este bloco requer a utilização do bloco “Wifi” ou do bloco “Ethernet” e é do tipo “principal”, pelo que só poderá ser utilizado uma vez e encaixado em blocos do mesmo tipo.

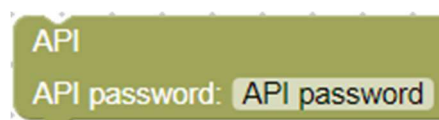


Figura 28 - API

2.2.1.6. OTA

Este bloco permite fazer o *upload* do código via WiFi, sem ser necessário recorrer a um cabo USB, tendo em conta que o primeiro *upload* tem de ser feito sempre através do cabo USB e para que seja possível realizar o *upload* via WiFi é necessário que o nome do *node* seja o mesmo face ao programa que está no ESP. No campo “Safe mode” deste bloco é possível seleccionar entre “True” e “False”, se estiver seleccionado “True”, o *safe mode* é ativo e caso o ESPHome não consiga inicializar ao fim de 10 tentativas, irá inicializar em modo de segurança, desabilitando todos os componentes, e apenas a porta serial, o WiFi e o OTA serão inicializados, de modo que seja possível fazer o *upload* de um novo código. No campo “Password” define-se a *password* a

utilizar para realizar as atualizações via *OTA*. Este bloco requer a utilização do bloco “Wifi” ou “Ethernet”.

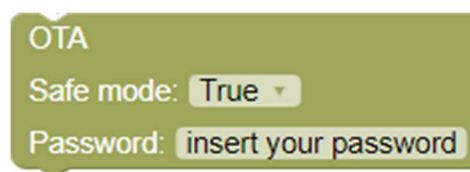


Figura 29 – OTA

2.2.1.7. Logger

O bloco “Logger” permite ativar as mensagens *log*, sendo que se pode seleccionar uma das seguintes opções:

- *NONE*: Nenhuma mensagem é mostrada.
- *ERROR*: Apenas os erros são mostrados. Os erros são problemas que impedem o *ESP* de funcionar corretamente. São mostrados a vermelho.
- *WARN*: Os erros e os avisos são mostrados. Os avisos são problemas como leituras inválidas de sensores dos quais o *ESP* consegue recuperar, são mostrados a amarelo.
- *INFO*: Os erros e avisos são mostrados e também todas as mensagens de informações, são mostradas a verde.
- *DEBUG*: Todas as mensagens descritas anteriormente são apresentadas acrescentando ainda os valores das leituras de sensores e mensagens de estados, é mostrado a ciano
- *VERBOSE*: Semelhante ao “*DEBUG*”, acrescentando mensagens que geralmente são consideradas como *spam*, apresentadas a cinzento.
- *VERY_VERBOSE*: Todas as mensagens internas são apresentadas, incluindo os dados que são transmitidos nos barramentos de dados (I2C, SPI ou UART), são apresentadas a branco. Poderá fazer com que o dispositivo fique lento.



Figura 30 – Logger

2.2.1.8. I2C

Este bloco permite definir o barramento I2C para que seja possível utilizar dispositivos que utilizem esse tipo de comunicação. No primeiro campo define-se o pino para a transmissão de dados, no segundo campo “*SCL pin*” define-se o pino da linha do *clock*, no campo “*Scan*” é possível escolher entre “*true*” e “*false*”, caso seja selecionada a primeira opção, o *ESPHome* irá fazer uma pesquisa de todos os endereços no barramento I2C. No último campo (“*Id*”) define-se o id do barramento I2C de forma a distinguir os vários barramentos caso haja mais do que um.



Figura 31 - I2C

2.2.1.9. Web Server

O bloco “*Web Server*” cria um *web server* tal como o nome indica, no campo “*Port*” deve definir em que porta é que o *web server* irá ser aberto. Para aceder ao mesmo pode utilizar o ip do *node* ou através de “<nome_do_node>.local/”. Note que a criação de um *web server* requer bastante memória do seu *ESP* e poderá gerar problemas, especialmente no *ESP8266*.

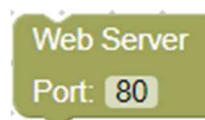


Figura 32 - Web Server

2.2.1.10. PCF8574

Este bloco permite utilizar o *PCF8574*, é um pequeno dispositivo que tem a função de adicionar entradas e saídas digitais. O *PCF8574* em concreto, permite

expandir em até oito entradas ou saídas por dispositivo, num máximo de oito dispositivos, sendo que cada um deles necessita de um endereço que nunca pode ser repetido. Neste bloco já são apresentados o máximo de *PCF8574* que podem ser utilizados em simultâneo e o utilizador deverá seleccionar quantos deseja utilizar clicando na *checkbox* de cada um, sendo que o endereço é fixo para cada um e não pode ser alterado, o id de cada um também já está pré-definido, mas pode ser alterado desde que não seja repetido.

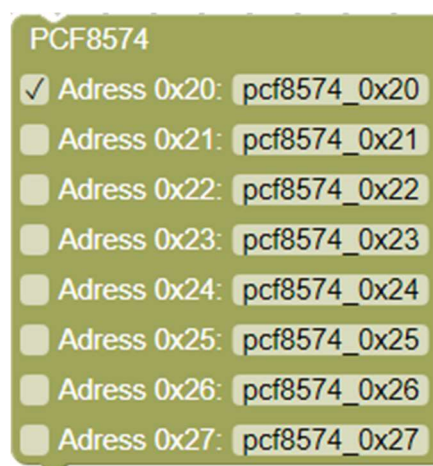


Figura 33 - *PCF8574*

2.2.2. Blocos *Sensor*

A secção “*Sensor*”, contém todos os blocos que estão relacionados a sensores, sejam os próprios sensores ou ações que possam ser geradas pelos mesmos. Nesta secção, apenas os blocos de cor mais escura, nomeadamente o bloco “*Binary Sensor*” e “*Sensor*”, é que são blocos do tipo “principais” e por isso apenas esses podem ser conectados a outros blocos principais.

2.2.2.1. *Binary Sensor*

Este bloco, apenas tem a função de inicializar o componente “*Binary Sensor*” (sensor binário), como é o caso de um botão ou de um sensor PIR. Nos encaixes exteriores podem ser encaixados todos os blocos “principais” e no encaixe interno, podem ser encaixados os blocos “*Binary Sensor*”, “*Binary Sensor PCF8574*”, “*PIR Sensor*” e “*PIR Sensor PCF8574*”. Este bloco só pode ser utilizado uma vez.

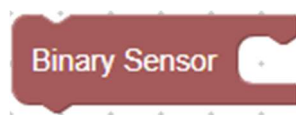


Figura 34 - *Binary Sensor*

2.2.2.2. *Binary Sensor (componente)*

Este bloco é utilizado para integrar o componente “*Binary Sensor*” no código, sendo que o encaixe superior e inferior apenas permite encaixar no interior do bloco anterior e nos blocos que representam sensores binários. Tem ainda um encaixe lateral onde poderá ser encaixado um bloco para realizar pequenas automações, através do bloco “*Automations*”. O primeiro campo tem a finalidade de definir o nome do componente, no segundo campo (“*Id*”) deverá atribuir um id que tem de ser único, no terceiro campo é onde se define o pino através do qual se vai ler o estado do sensor binário, no campo “*Mode*” poderá selecionar “*input*” que será uma entrada normal e os valores poderão sofrer flutuações, tem também a opção “*input pullup*” e “*input pulldown*” para evitar essas mesmas flutuações, sendo que o *ESP* faz uma *pullup* ou *pulldown* internamente (a opção “*input pulldown*” apenas pode ser utilizada no *ESP32*). Já no campo “*Inverted*” é possível escolher entre “*True*” e “*False*”, sendo que com a primeira opção irá inverter o valor lido no pino definido.

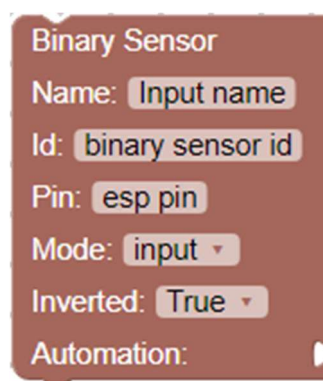


Figura 35 - *Binary Sensor*

2.2.2.3. *Binary Sensor PCF8574*

Este bloco tem a mesma função e características do anterior, à exceção de que neste se utiliza os pinos do *I/O Expander* e no anterior utilizam-se os pinos do *ESP*. Tendo isso em conta, os campos em comum têm as mesmas funções e características,

sendo apenas o campo “PCF8574” diferente, onde é necessário introduzir o id do PCF8574 previamente definido e o pino onde deseja ler o sensor binário, podendo ser um de oito pinos (de 0 a 7). Este bloco requer a utilização do bloco “PCF8574”.

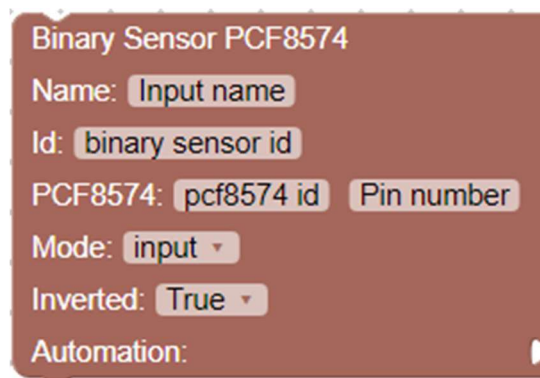


Figura 36 - Binary Sensor PCF8574

2.2.2.4. PIR Sensor

Através do bloco “PIR Sensor” é possível integrar no código um sensor PIR (*Passive Infrared Sensor*), ou seja, um sensor de movimento. Para utilizar este bloco necessita de atribuir um nome ao sensor no primeiro campo, no campo “Id” definir um id único para o mesmo e no terceiro campo definir o pino do ESP através do qual irá fazer a leitura do sensor. Este bloco possui dois campos para realizar ações aquando da deteção e da não deteção de movimento, sendo que as ações apenas serão acionadas apenas no instante em que é detetado movimento, no caso do primeiro encaixe interior, e no instante em que o sensor deixou de detetar movimento, no caso do segundo encaixe interior. Nestes encaixes interiores apenas é possível encaixar o bloco “Binary Sensor Action” e o bloco “CAN bus Send Action”. Note que só é possível encaixar este bloco em blocos da mesma cor e que representem sensores binários ou no interior do bloco “Binary Sensor”.

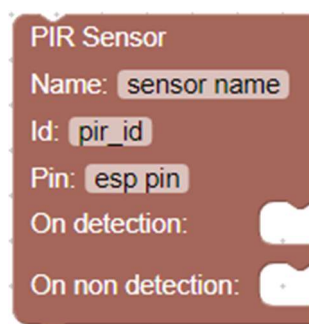


Figura 37 - PIR Sensor

2.2.2.5. PIR Sensor PCF8574

Este bloco é igual ao anterior à exceção de que neste utiliza-se o *PCF8574* para fazer a leitura do sensor PIR, definindo no campo “*PCF8574*” o id do *I/O Expander* e o número do pino. Este bloco requer a utilização do bloco “*PCF8574*”.

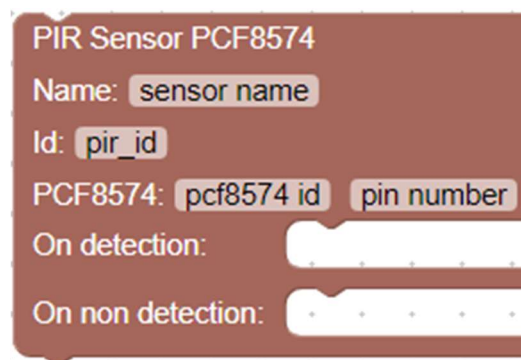


Figura 38 - PIR Sensor PCF8574

2.2.2.6. Automations

Este bloco tem a finalidade de incluir algumas automações nos sensores binários, nomeadamente no bloco “*Binary Sensor*” e “*Binary Sensor PCF8574*”, pois são os únicos blocos onde este bloco pode ser encaixado. Apresenta vários campos onde se podem introduzir várias ações, sendo que as ações que estiverem no campo “*On Click*” serão executadas quando, o botão é clicado, supondo que o sensor binário é um botão, no campo “*On Double Click*” quando existe um clique duplo no botão, no campo “*On Multi Click*” aquando o botão sofre um duplo clique ou um clique longo ou curto, no campo “*On Press*” as ações serão executadas quando um botão for pressionado e no campo “*On Release*” quando o botão deixar de ser pressionado.



Figura 39 – Automations

2.2.2.7. On Click

O bloco “On Click”, só pode ser encaixado no campo “On Click” do bloco “Automations”. Tem a finalidade de definir o tempo mínimo e máximo do clique necessário para que a ação, que é adicionada no encaixe interior seja executada. No encaixe interior deste bloco apenas é possível encaixar o bloco “Binary Sensor Action” e “CAN bus Send Action”.



Figura 40 - On Click

2.2.2.8. On Double Click

Este bloco é bastante semelhante ao anterior, no entanto tem o objetivo de definir a duração mínima e máxima dos duplos cliques para que a ação seja executada, ou seja, sempre que existir um clique duplo em que cada um tenha uma duração compreendida entre a duração mínima e máxima definida. Este bloco apenas pode ser encaixado no campo “On Double Click” do bloco “Automations” e o seu encaixe interior apenas aceita o bloco “Binary Sensor Action” e “CAN bus Send Action”.

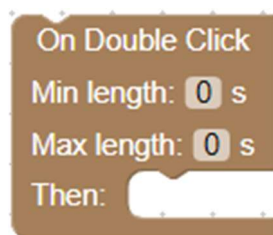


Figura 41 - *On Double Click*

2.2.2.9. *On Multi Click*

O bloco “*On Multi Click*”, tem como objetivo executar várias ações para determinados tipos de cliques, sendo que na zona “*Double click*” deve definir o tempo máximo em que o botão, no primeiro clique, pode estar no estado *ON*, o tempo máximo que pode estar a *OFF*, o tempo máximo em que no segundo clique pode estar a *ON* e o tempo mínimo que necessita de estar a *OFF* para que execute a ação do primeiro encaixe do bloco. Nas zonas “*Single long click*” e “*Single short click*” é necessário definir o intervalo de tempo em que deseja que o botão esteja no estado *ON* e o tempo mínimo que esteja no estado *OFF* para que execute a ação correspondente. Nos encaixes interiores deste bloco só é possível encaixar os blocos “*Binary Sensor Action*” e “*CAN bus Send Action*” e o encaixe superior só aceita o encaixe do campo “*On Multi Click*” do bloco “*Automations*”.

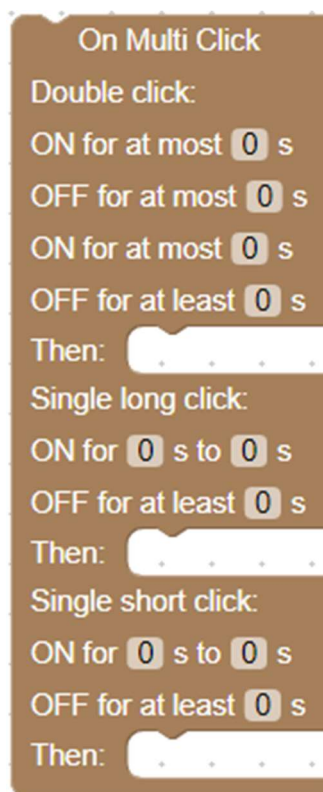


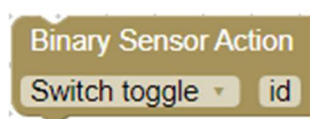
Figura 42 - *On Multi Click*

2.2.2.10. Binary Sensor Action

Este bloco é o responsável pela escolha das ações a executar, sendo que só pode ser encaixado nos campos “On Press” e “On Release” do bloco “Automations” e nos encaixes interiores dos blocos “On Click”, “On Double Click”, “On Multi Click”, “PIR Sensor”, “PIR Sensor PCF8574” em outros blocos “Binary Sensor Action” ou “CAN bus Send Action”. Só tem dois campos, um deles é para selecionar a ação a executar e o outro para definir o id do componente dessa mesma ação, sendo as opções de ações as seguintes:

- *Switch toggle*: Inverte o estado do componente *switch*, independentemente do estado atual.
- *Switch turn on*: Coloca o componente *switch* em questão no estado “ligado”.
- *Switch turn off*: Coloca o componente *switch* em questão no estado “desligado”.
- *Cover open*: Abre os estores elétricos definidos no campo “id”.
- *Cover close*: Fecha os estores elétricos definidos no campo “id”.

- *Cover stop*: Caso estes estejam em movimento, para os estores elétricos definidos no campo “*id*”.
- *Light toggle*: Inverte o estado do componente *light*.
- *Light turn on*: Liga o componente *light* em questão.
- *Light turn off*: Desliga o componente *light* definido no campo “*id*”.
- *Fan toggle*: Inverte o estado do componente *fan*.
- *Fan turn on*: Liga o componente *fan* definido no campo “*id*”.
- *Fan turn off*: Desliga o componente *fan*.
- *Logger*: Apresenta a mensagem escrita no campo “*id*” na consola.


Figura 43 – *Binary Sensor Action*

2.2.2.11. CAN bus Send Action

Este bloco é bastante semelhante ao anterior, sendo que as suas características de encaixe são iguais, com a diferença de que este permite enviar dados através do protocolo de comunicação CANbus. Este bloco requer a utilização do bloco “*CAN Bus*”. No primeiro campo deste bloco é necessário introduzir o id do componente CANbus, no campo “*Can id*” é necessário colocar o id do dispositivo para o qual enviará os dados e no último campo deverá colocar os dados a transmitir, sendo que terão de ser introduzidos num dos seguintes formatos:

- ‘mensagem’: Para enviar uma mensagem escrita.
- [0x01]: Para enviar um *byte*
- [0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08]: Para enviar vários *bytes*, até num máximo de 8.

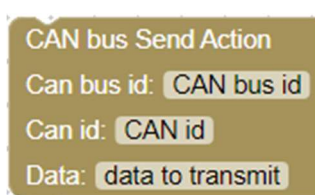


Figura 44 - CAN bus Send Action

2.2.2.12. *Sensor*

O bloco “*Sensor*” é do tipo “principal” e por isso pode ser encaixado em outros blocos do mesmo tipo, tem a função de inicializar o componente “*Sensor*” para que seja possível utilizar os vários sensores disponíveis. No seu encaixe interior apenas podem ser encaixados blocos de cor azul clara que representem sensores. Este bloco apenas pode ser utilizado uma vez, não se podendo repetir.

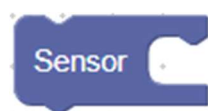


Figura 45 – *Sensor*

2.2.2.13. *DHT Sensor*

O bloco “*DHT Sensor*” permite integrar o sensor DHT, tanto o DHT11 como o DHT22, sendo que no primeiro campo se deve definir o pino que irá receber a leitura do sensor, no campo “*Temperature name*” deve introduzir o nome a apresentar na componente de temperatura, no campo “*Humidity name*” o nome a apresentar na componente de humidade e no último campo o intervalo de tempo, em segundos, que o *ESPHome* deverá atualizar os valores. Caso pretenda enviar os valores do sensor através do protocolo de comunicação CANbus, deverá selecionar a *checkbox* do valor que pretende enviar, tal como o CANbus id do dispositivo CAN e o CAN id da mensagem a enviar, o CAN id deverá ser diferente e único. Este bloco apenas pode ser encaixado no encaixe interior do bloco “*Sensor*” e em blocos da mesma cor e caso selecione alguma das *checkbox* será necessário utilizar o bloco “*CAN Bus*”

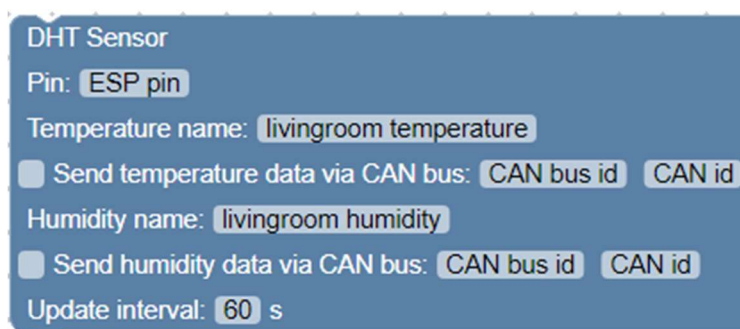


Figura 46 - *DHT Sensor*

2.2.2.14. SI7021 Sensor

Este bloco é bastante semelhante ao anterior, alterando o facto de este integrar o sensor *SI7021*, também ele de temperatura e humidade, no entanto neste bloco não é necessário definir o pino de leitura, já que o *ESPHome* consegue detetar automaticamente o endereço do sensor, sendo necessário, para a utilização deste bloco, o componente I2C que pode ser integrado utilizando o bloco “I2C”. Todas as características dos campos e conexões do bloco são iguais às do bloco anterior.

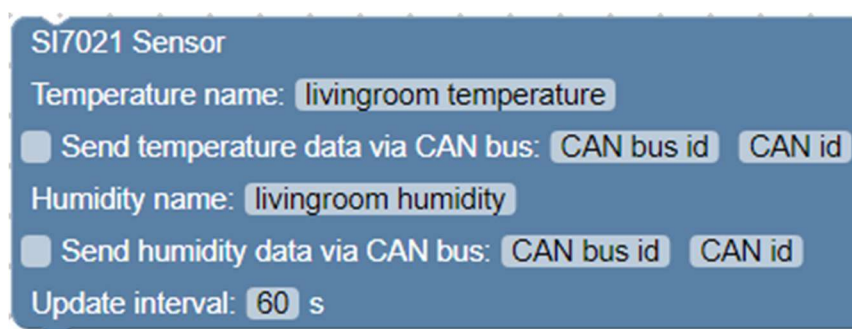


Figura 47 - *SI7021 Sensor*

2.2.2.15. BMP280 Sensor

Este bloco integra o sensor *BMP280*, é um sensor de temperatura e pressão, comunica via I2C e como tal é necessário adicionar o componente I2C através do bloco “I2C”. No primeiro campo define-se o nome a apresentar para o valor de temperatura, no campo “*Pressure name*” define-se o nome a apresentar para o valor de pressão, no campo “*Address*” deve seleccionar o endereço do sensor, podendo variar entre 0x77 e 0x76 (geralmente é 0x77), no último campo define-se a frequência com que o *ESPHome* deve atualizar os valores. Este bloco apenas pode ser encaixado no bloco “*Sensor*” ou em blocos da mesma cor e as características dos campos de envio de dados via CANbus são iguais às do bloco anterior.

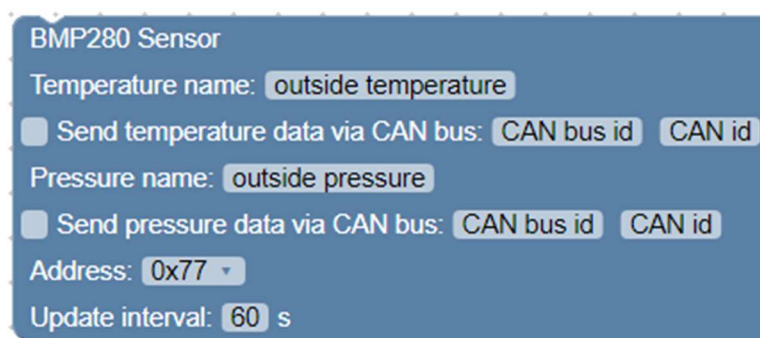


Figura 48 – BMP280 Sensor

2.2.2.16. MQ-7

O bloco “MQ-7” integra o sensor MQ-7. Este sensor mede os níveis de monóxido de carbono presente no ar. No campo “Pin” deste bloco é necessário definir o pino do *ESP* através do qual se irão receber as leituras do sensor, de notar que o pino terá de ser capaz de receber valores analógicos, no segundo campo deverá definir o id do sensor que deve ser único, e no último campo a frequência com que o *ESPHome* atualiza os valores das leituras. As características dos encaixes bem como de envio de dados via CANbus são iguais às do bloco anterior. Este sensor necessita de calibração para que os valores do mesmo sejam coerentes.

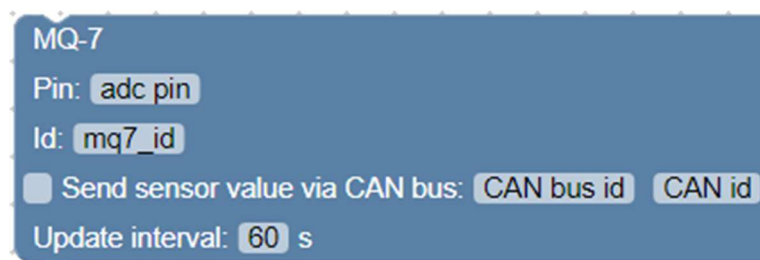


Figura 49 - MQ-7

2.2.2.17. Template Sensor

A função deste bloco é criar um *template* de um sensor onde os valores do mesmo serão fornecidos através do bloco “CAN bus Receive Action”, ou seja, permite visualizar os valores de um sensor, que são recebidos através do protocolo de comunicação CANbus, na forma de sensor no *Home Assistant*. Este bloco apenas poderá ser encaixado em blocos da mesma cor e terá de ser utilizado em conjunto com o bloco “CAN bus Receive Action”. No seu primeiro campo, deverá atribuir um nome ao valor

que irá apresentar, no campo “*Id*” deve definir um id para o *template* do sensor, no campo “*Unit of measurement*” terá de colocar a unidade do valor que o *template* apresentará, sendo que pode selecionar um das seguintes opções:

- °C: Graus Celsius.
- °F: Graus Fahrenheit.
- %: Percentagem.
- *hPa*: Hectopascal.
- *mbar*: Milibar.
- *lx*: Lux.
- *lm*: Lúmen.
- *CO*: Monóxido de carbono.
- *CO2*: Dióxido de carbono.
- *V*: Diferença de potencial.
- *A*: Corrente elétrica.
- *Wh*: Watt-hora.
- *kWh*: Quilowatt-hora.
- *W*: Watt.
- *kW*: Quilowatt.

No campo “*Device class*” deve selecionar a classe dos valores, para que no *Home Assistant* seja mostrado o ícone certo, pode selecionar uma das seguinte opções:

- *Temperature*: Para valores de temperatura.
- *Humidity*: Para valores de humidade.
- *Pressure*: Para valores de pressão.
- *Illuminance*: Para valores de iluminância.
- *Carbon monoxide*: Para valores de monóxido de carbono.
- *Carbon dioxide*: Para valores de dióxido de carbono.
- *Battery*: Para níveis de bateria.
- *Voltage*: Para valores de diferença de potencial.
- *Current*: Para valores de corrente elétrica.
- *Energy*: Para valores de energia.
- *Power*: Para valores de potência.

No campo “*Update interval*”, deve introduzir o intervalo de tempo, em segundos, que deseja que os valores sejam atualizados. Note que a seleção das unidades de medida não altera os valores do sensor em questão, apenas tem o objetivos de mostrar as unidades de medida no *Home Assistant*, cabe ao utilizador verificar a unidade dos valores.

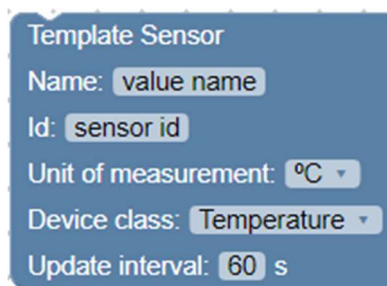


Figura 50 - *Template Sensor*

2.2.3. Blocos *Switch*

A secção de blocos “*Switch*” permite utilizar os pinos do *ESP* para acionar alguns dispositivos, seja com recurso a relés ou ligando diretamente ao *ESP*. Nesta secção apenas o bloco “*Switch*” é do tipo “principal” e por isso apenas esse pode ser conectado a outros do mesmo tipo.

2.2.3.1. *Switch*

O bloco “*Switch*”, permite inicializar o componente *switch* possibilitando assim a utilização de todos os blocos desta secção. Este bloco é do tipo “principal” e só pode ser utilizado uma vez.

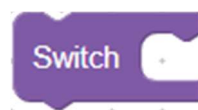


Figura 51 – *Switch*

2.2.3.2. *Switch (componente)*

Este bloco permite definir um pino do *ESP* como uma saída e apenas pode ser encaixado no interior do bloco “*Switch*” ou em blocos da mesma cor. No primeiro campo do bloco deve atribuir um nome para o *switch*, no campo “*Pin*” deverá definir o pino que será controlado, caso pretenda inverter o estado da sua saída, pode fazê-lo selecionando a opção “*True*” no campo “*Inverted*”. No campo “*Id*” deve introduzir o id que pretende atribuir ao componente e este tem de ser único. Este bloco permite realizar o *interlock* entre dois componentes *switch*, o *interlock* é útil em casos onde existem duas saídas que não podem ser acionadas em simultâneo, o *interlock* impedirá que isso aconteça. De modo a acionar essa função, basta selecionar a *checkbox* “*Interlock*”, atribuir um nome ao *interlock* que deseja fazer e no terceiro e quarto campo dessa linha, introduzir o id de cada *switch* que deseja que não sejam acionados em simultâneo. Deverá fazer este procedimento em ambos os blocos “*Switch*”, sendo que os nomes dos *interlock* têm de ser diferentes, veja-se o exemplo da Figura 53. Através deste bloco, é também possível ativar o *restore mode*, que permite ao *ESPHome*, em caso de falha, recuperar o estado do *switch* ou colocar o estado do mesmo a ON ou OFF quando o *ESP* ligar, para isso basta selecionar a *checkbox* do último campo e selecionar uma das várias opções da lista:

- *Always OFF*: Sempre que o *ESP* ligar o estado do *switch* será OFF.
- *Always ON*: Sempre que o *ESP* ligar o estado do *switch* será ON.
- *Restore default OFF*: Ao iniciar, o *ESPHome* irá tentar recuperar o estado anterior do *switch*, caso não seja possível a recuperação o mesmo iniciará com o estado OFF.
- *Restore default ON*: Ao iniciar, o *ESPHome* irá tentar recuperar o estado anterior do *switch*, caso não seja possível a recuperação o mesmo iniciará com o estado ON.

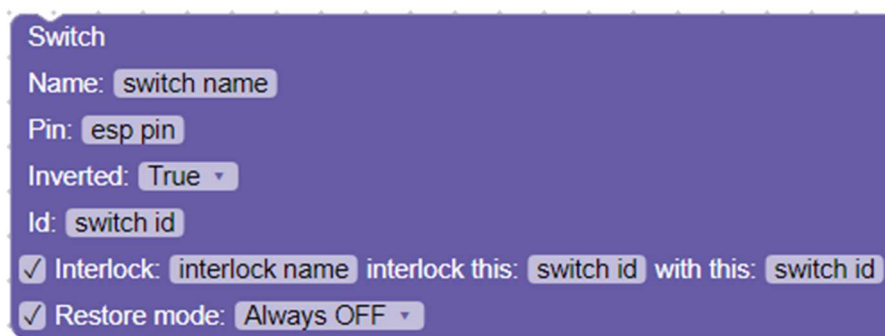


Figura 52 – Switch (componente)



Figura 53 - Exemplo de Interlock

2.2.3.3. Switch PCF8574

Este bloco é idêntico ao anterior em todos os aspetos, à exceção de que neste o pino a controlar não será do *ESP*, mas sim do *PCF8574*, como tal no campo “*PCF8574*” deverá introduzir o id do *PCF8574* e o pino do mesmo ao qual vai ligar a sua saída. Este bloco requer a utilização do bloco “*PCF8574*”.

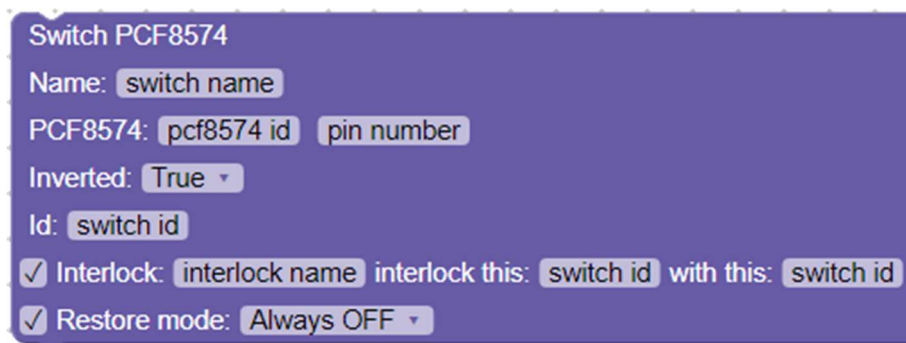


Figura 54 - Switch PCF8574

2.2.3.4. Restart Switch

O bloco “Restart Switch” tem o objetivo de criar um *switch* que permite reinicializar o *ESP*. O bloco apenas tem um campo, onde deve definir um nome para o *switch* e este bloco pode ser encaixado no interior do bloco “Switch” ou em qualquer um da mesma cor.

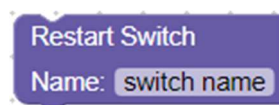


Figura 55 - Restart Switch

2.2.3.5. Shutdown Switch

Este bloco permite desligar o *node* através de um *switch*, colocando-o em *deep sleep*. Após isso a única maneira de iniciar o *node* é clicando no botão de *reset* do *ESP* ou desconectar o *ESP* da alimentação e voltar a conectá-lo à mesma. Este bloco tem as mesmas características de encaixes do anterior.



Figura 56 - Shutdown Switch

2.2.4. Blocos Output

A secção de blocos “Output”, permite utilizar pinos do *ESP* como saída, no entanto, ao contrário dos blocos da secção anterior, estes não são apresentados no *Home*

Assistant nem no *web server*. Estes blocos são apenas blocos intermediários que podem ser utilizados com blocos “*Light*” por exemplo.

2.2.4.1. Output

O bloco “*Output*” é do tipo “principal”, e como tal apenas pode ser encaixado em outros blocos do mesmo tipo e utilizado uma única vez. Tem o objetivo de inicializar o componente *output*, de modo a possibilitar a utilização de blocos desta secção.



Figura 57 – *Output*

2.2.4.2. Output (componente)

Este bloco permite controlar um pino do *ESP* como saída, de forma intermediária. No seu primeiro campo deve definir o pino do *ESP* a controlar, no campo “*Inverted*” pode inverter o estado da saída escolhendo a opção “*True*”, e no último campo deve atribuir, ao componente, um id que tem de ser único. Este bloco apenas pode ser encaixado no interior do bloco “*Output*” e nos blocos da mesma cor.

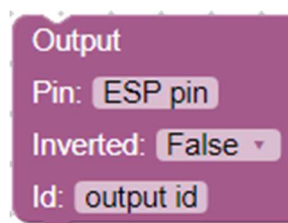


Figura 58 - *Output (componente)*

2.2.4.3. Output PCF8574

O bloco “*Output PCF8574*” possui as mesmas características face ao anterior, alterando o facto de que neste bloco o pino a ser controlado é do *PCF8574*, e por isso, no primeiro campo deve introduzir o id do mesmo e o número do pino a controlar. Este bloco requer a utilização do bloco “*PCF8574*”.

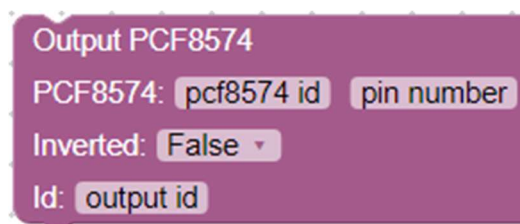


Figura 59 - *Output PCF8574*

2.2.5. Blocos *Actuators*

Esta secção de blocos inclui todos os blocos que permitem acionar saídas de componentes específicos. Os blocos “*Light*” e “*Fan*” desta secção têm de ser utilizados em conjunto com o bloco “*Output*”, para que exista uma saída possível de acionar, já os blocos “*Cover*” e “*Endstop Cover*” terão de ser utilizados juntamente com o bloco “*Switch*” e o bloco “*Endstop Cover*” necessita também de dois blocos “*Binary Sensor*”.

2.2.5.1. *Light*

Este bloco permite inicializar o componente *light* e é do tipo “principal”, pelo que só poderá ser encaixado em blocos desse tipo e apenas pode ser utilizado uma vez.



Figura 60 – *Light*

2.2.5.2. *Light (componente)*

Através deste bloco pode integrar o componente *light* no seu *node* e assim controlar uma fonte de luz. Este bloco apenas pode ser encaixado no interior do bloco “*Light*”. No primeiro campo do bloco deve atribuir um nome ao componente, no campo “*Output*” deve colocar o id do componente *output* ao qual estará ligada a fonte de luz, no campo “*Id*” necessita de atribuir um id único ao componente *light* e no último campo pode seleccionar a *checkbox* de modo que o *ESPHome*, em caso de falha, tente recuperar o estado do componente em questão ou colocar o estado do mesmo a ON ou OFF quando o *ESP* ligar, para isso basta seleccionar uma das várias opções da lista:

- *Always OFF*: Sempre que o *ESP* ligar o estado da *light* será OFF.

- *Always ON*: Sempre que o *ESP* ligar o estado da *light* será ON.
- *Restore default OFF*: Ao iniciar, o *ESPHome* irá tentar recuperar o estado anterior da *light*, caso não seja possível a recuperação o mesmo iniciará com o estado OFF.
- *Restore default ON*: Ao iniciar, o *ESPHome* irá tentar recuperar o estado anterior da *light*, caso não seja possível a recuperação o mesmo iniciará com o estado ON.

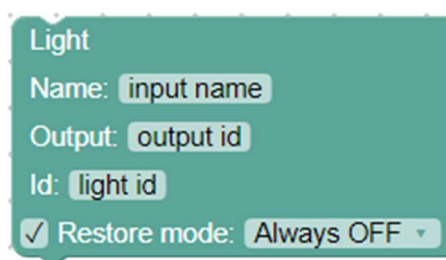


Figura 61 - *Light* (componente)

2.2.5.3. *Fan*

O bloco “*Fan*” possibilita a integração do componente *fan*, é um bloco do tipo “principal”, o que faz com que apenas seja possível utilizá-lo uma vez e encaixá-lo em outros blocos do mesmo tipo.



Figura 62 – *Fan*

2.2.5.4. Binary Fan

Este bloco permite controlar o estado de uma ventoinha. No campo “*Name*” é pedido que atribua um nome ao componente, no segundo campo deve atribuir um id que seja único e no último campo deve introduzir o id do componente *output* onde a ventoinha estará ligada. Este bloco só pode ser encaixado no interior do bloco “*Fan*”.

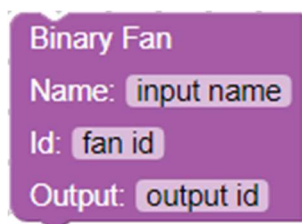


Figura 63 - *Binary Fan*

2.2.5.5. Cover

O bloco “*Cover*” possibilita a integração do componente *cover* e a utilização dos seus blocos. É um bloco do tipo “principal”, ou seja, apenas poderá ser encaixado em blocos do mesmo tipo e só pode ser utilizado uma única vez.

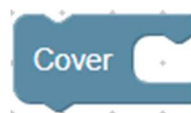


Figura 64 – Cover

2.2.5.6. Cover (componente)

Este bloco permite controlar uns estores elétricos, sendo que no primeiro campo deve atribuir um nome ao componente e no segundo campo um id único ao mesmo. No campo “*Open action*” deverá colocar as ações que pretende realizar quando é dada ordem de abertura, o mesmo se passa nos campos “*Close action*” e “*Stop Action*”, para as ordens de fecho e paragem, respetivamente. No campo “*Open duration*” deverá introduzir o intervalo de tempo, em segundos, que os estores elétricos demoram a abrir e no campo “*Close duration*”, o intervalo de tempo que demoram a fechar, note que é de extrema importância que este valor seja certo já que os estores apenas irão parar de abrir e/ou fechar passado esse tempo (caso não seja bem definido o motor poderá entrar em esforço). Os encaixes interiores deste bloco apenas permitem encaixar o bloco “*Cover Action*” desta secção e os encaixes superior e inferior só permitem que o bloco seja encaixado no bloco “*Cover*” e em blocos da mesma cor. Este bloco requer a implementação de duas saídas através do bloco “*Switch (componente)*” e que os seus encaixes interiores estejam preenchidos.

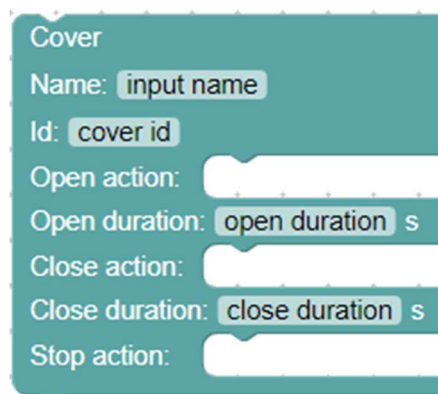


Figura 65 - Cover (componente)

2.2.5.7. Endstop Cover

Este bloco permite controlar uns estores elétricos tal como o bloco anterior, no entanto, neste bloco a abertura e fecho dos estores será controlada através de sensores binários, ou seja, os estores só irão parar de fechar ou abrir quando o respetivo sensor for acionado, contudo, é necessário preencher os campos “*Open duration*” e “*Close duration*”. Todas as configurações deste bloco são iguais ao anterior, sendo que neste é necessário introduzir o id do sensor que irá detetar a abertura total e o fecho total, nos campos “*Open endstop*” e “*Close endstop*”, respetivamente. Este bloco requer a implementação de duas saídas através do bloco “*Switch (componente)*” e que os seus encaixes interiores estejam preenchidos, tal como a implementação de dois sensores binários através do bloco “*Binary Sensor*”.



Figura 66 - Endstop Cover

2.2.5.8. Cover Action

O bloco “Cover Action” tem a função de alterar o estado dos componentes *switch* quando dada ordem de abertura, fecho ou paragem dos estores elétricos, sendo que para isso, deve seleccionar o estado que deseja e inserir o id do *switch* do qual pretende alterar o estado. Este bloco apenas pode ser encaixado em blocos iguais ou nos encaixes interiores dos blocos “Cover” e “Endstop Cover”.



Figura 67 - Cover Action

2.2.6. Blocos Communication

Esta secção de blocos inclui todos os blocos que permitem realizar a comunicação entre dispositivos via CANbus.

2.2.6.1. SPI

O bloco “SPI”, permite inicializar o componente SPI e desse modo adicionar outros blocos que dependam desse mesmo componente. Este bloco é do tipo “principal” e por isso, só pode ser encaixado noutros blocos do mesmo tipo. No primeiro campo deve definir o pino a utilizar para a linha do *clock*, no campo “*Mosi pin*” é onde se define o pino da linha *mosi*, através da qual o dispositivo *master* irá enviar os dados para os dispositivos *slave*, no campo “*Miso pin*” deverá definir o pino a utilizar para a linha *miso*, linha essa onde os dispositivos *slave* enviam os dados para os dispositivos *master*. No último campo, é necessário definir um id único para o barramento SPI.



Figura 68 – SPI

2.2.6.2. CAN Bus

O bloco “CAN Bus” possibilita a comunicação entre dispositivos via CANbus, que é um protocolo de comunicação que utiliza apenas duas linhas para a sua comunicação. Este bloco é do tipo “principal”, o que faz com que só possa ser encaixado em blocos do mesmo tipo e utilizado uma vez. Este bloco requer a utilização do bloco “SPP”. No primeiro campo deste bloco deve definir um id para o componente CANbus e no segundo campo, um id para identificar o dispositivo em questão, no campo “Use extended id” ao selecionar a opção “True” o componente irá suportar um espaço de 29 bits para o Can id, caso selecione “False” suportará um espaço de 11 bits. No campo “CS pin” deve definir o pino da linha *chip select* (CS), no campo “Bit rate” deverá selecionar uma das várias velocidades de comunicação, no campo “clock” deverá selecionar a frequência do *clock* utilizada no seu CANbus *transceiver*, no encaixe interior do bloco, deve colocar as ações que deseja executar através do bloco “CAN bus Receive Action” (só este bloco é suportado no encaixe), no campo “Mode” é necessário selecionar uma das seguintes opções de funcionamento:

- *NORMAL*: Irá ter o funcionamento normal, recebe e envia dados.
- *LOOPBACK*: Pode ser utilizado para testar as ligações do componente SPI, pelo que irá enviar mensagens e recebê-las de volta.
- *LISTENONLY*: Neste modo, o CANbus apenas irá receber mensagens.

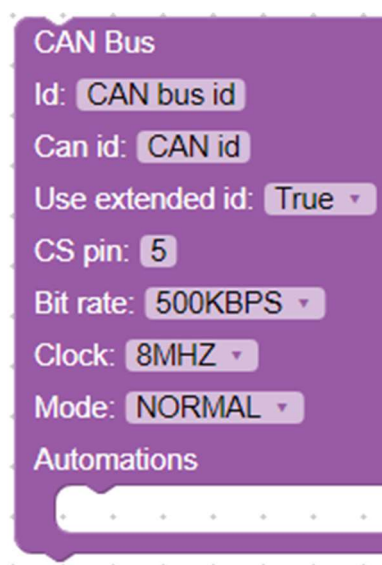


Figura 69 - CAN Bus

2.2.6.3. *CAN bus Receive Action*

Este bloco apenas pode ser encaixado no encaixe interior do bloco “*CAN Bus*” e em blocos da mesma cor. O bloco “*CAN bus Receive Action*”, permite executar determinadas ações quando se recebe uma mensagem de determinado dispositivo, sendo que para isso basta introduzir o id da mensagem CANbus, do qual irá receber a mensagem, no primeiro campo. No último campo deverá introduzir o id do componente da ação a executar e selecionar uma ação da lista:

- *Switch toggle*: Inverte o estado do componente *switch*, independentemente do estado atual.
- *Switch turn on*: Coloca o componente *switch* em questão no estado “ligado”.
- *Switch turn off*: Coloca o componente *switch* em questão no estado “desligado”.
- *Cover open*: Abre os estores elétricos definidos no campo “id”.
- *Cover close*: Fecha os estores elétricos definidos no campo “id”.
- *Cover stop*: Caso estes estejam em movimento, para os estores elétricos definidos no campo “id”.
- *Light toggle*: Inverte o estado do componente *light*.
- *Light turn on*: Liga o componente *light* em questão.
- *Light turn off*: Desliga o componente *light* definido no campo “id”.
- *Fan toggle*: Inverte o estado do componente *fan*.
- *Fan turn on*: Liga o componente *fan* definido no campo “id”.
- *Fan turn off*: Desliga o componente *fan*.
- *Receive sensor data*: Coloca os valores recebidos no id definido no campo “Can id” no *template* definido através do seu id no campo “id”.
- *Logger*: Apresenta a mensagem escrita no campo “id” na consola.

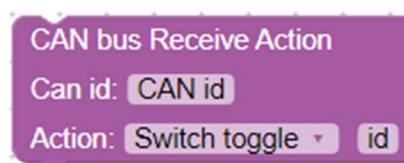


Figura 70 – *CAN bus Receive Action*