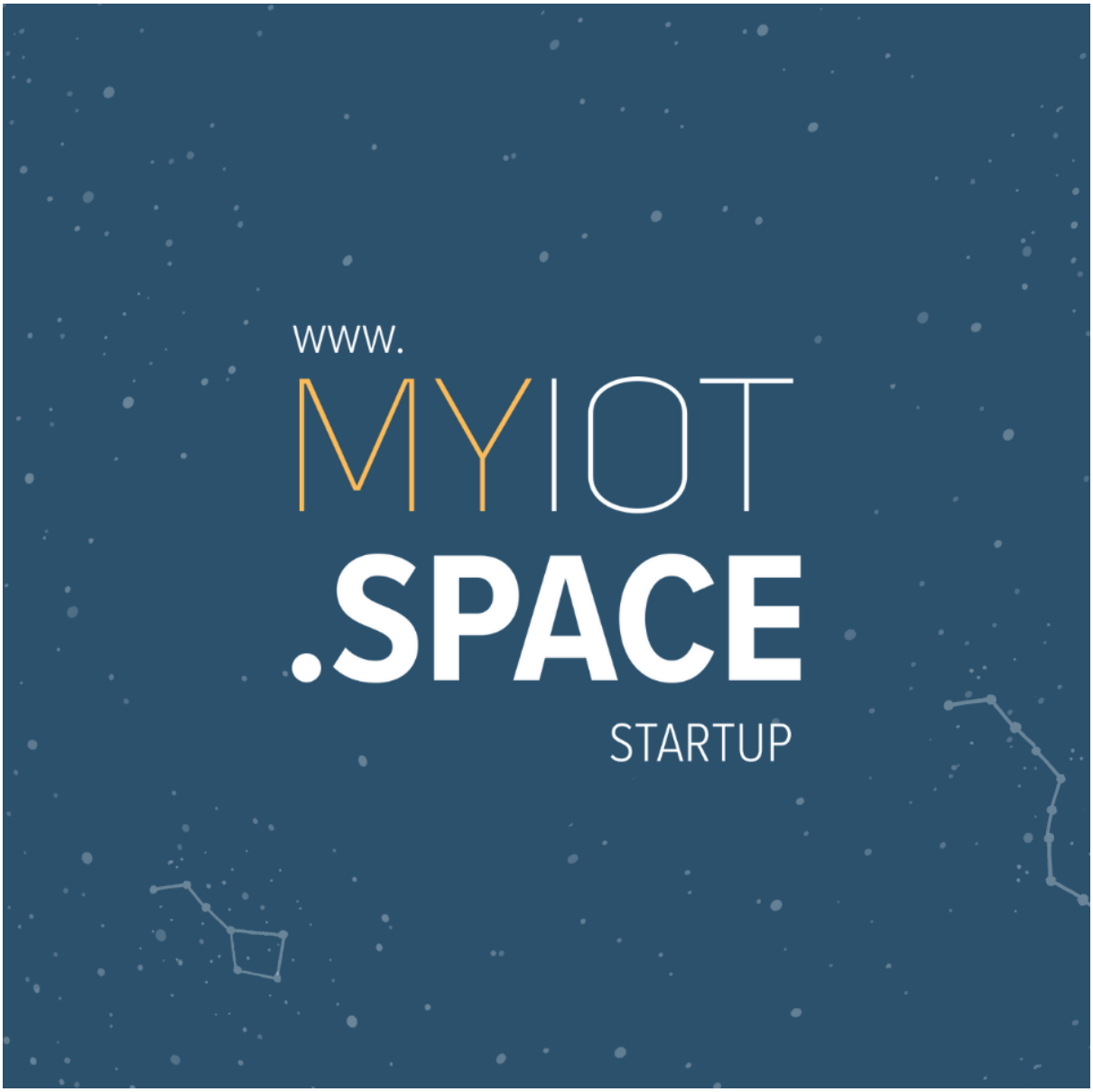


COMPLETE REFERENCE GUIDE

INDEX

1. PROGRAMMING INTERNET OF THINGS DEVICES

1.1. PROGRAMMING MEANS

With IoT, each device to be used (called microcontrollers) must be programmed specifically for the project in question. There are three ways to program them:

- **Programming IDE -**IDE (Integrated Development Environment) is an English term that means “Integrated Development Environment”. The idea is of a platform where the user composes a written code to specify functionalities for a microcontroller according to sensors, actuators or communication with the internet. This way of programming is only suitable for advanced users, as it is necessary to know the C++ programming language.

- **MyIoT Flasher -** MyIoT Flasher is a practical software that can be used by both advanced users and beginners. The program provides a library with ready-made codes, which are automatically inserted and configured in a microcontroller connected to the computer. Each code has a description of how it works, as well as a usage tutorial, with its particularities, possible applications and electrical connections. For more information on this software, access section 2.5 - FLASHER.

➢ **anaconda coding**- Sucuri Coding is an application that seeks to allow users with no programming experience to create custom programs and codes for the microcontrollers Arudino Uno, Arduino Nano, Modelix 3.6, ESP 32 and ESP 8266 and ESP32-CAM. The software works with written programming and also has the possibility of programming from blocks, but in a unique language developed by MyIoT that is easy to understand. For more information about this software, access section 2.6 - SUCURI CODING.

1.2. CREATING AN IOT MICROCONTROLLER

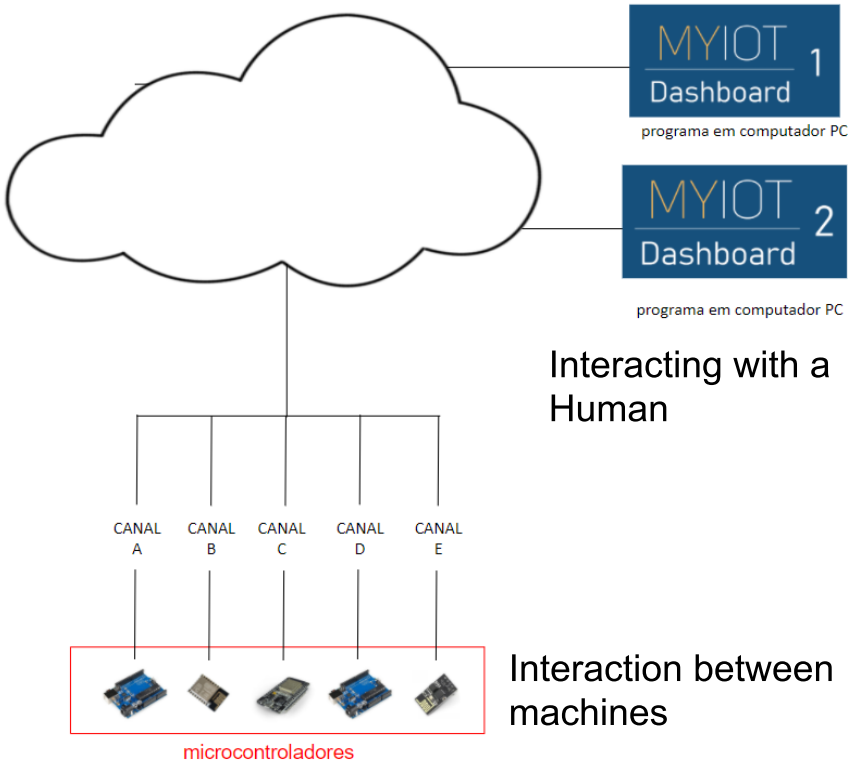
Thinking in the context of Internet of Things programming, there are two steps required to create an “IoT microcontroller”:

- Define the routine of how the microcontroller should behave based on the sensors and actuators physically connected to it;

- Define instructions on how the microcontroller should behave with commands received from the “cloud”, as well as outline what events (thinking about sensors and actuators) will cause commands to be sent to the “cloud”. Such commands are called triggers, which will be explained further in the next section.

An IoT project allows different microcontrollers to communicate over the Internet, but it is necessary to program each one individually so that they interact in the desired way.

Each microcontroller communicates with the internet through “streets”, which we call channels. We can communicate with the channels from a “communication center”, which we call the Dashboard (more information in section 2.2 - Dashboard), as well as through the microcontrollers themselves. In general, the functioning of the system can be visualized as follows:



1.3. CONCEPT OF TRIGGERS

triggers, in English, is trigger. This concept in programming corresponds to a letter or word that triggers an action or sequence of actions in a code. Examples are #output B, #output b, #email, #message.

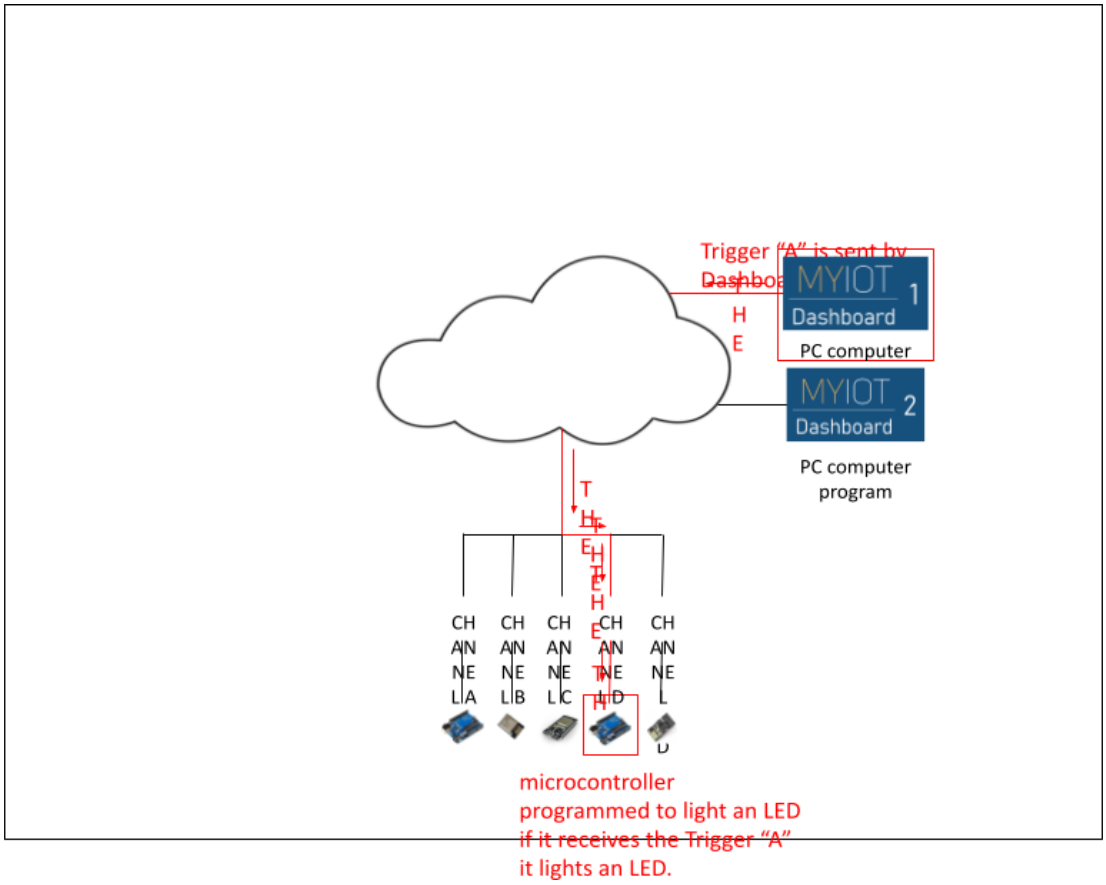
One*trigger*it can originate in the “cloud”, that is, in an application connected to the internet such as Dashboard; or it can originate in a microcontroller, sending the signal to the “cloud”. In both cases, the trigger causes an action or sequence of actions on the Dashboard or on the microcontroller. This tool enables communication between microcontrollers, as well as the creation of highly functional and complex projects.

Therefore, understanding this concept is essential. The mechanism is used in most IoT projects, and the mastery of its operation leads to a great increase in the possibilities of a project.

In summary, the two ways to use triggers are:

- **Sending a trigger directly from the Dashboard -**This option corresponds to the Dashboard's “Send triggers” and “Send to channels” tools. In it, it is possible to send the triggers “A” - “E”, in lowercase and uppercase, as well as any standard system trigger. Triggers will be specified later in this section.

Example:



- **Programming microcontrollers to send triggers -**Different from the first way of using the trigger, sending a trigger directly from a microcontroller involves knowing the trigger types and their possibilities. Since using Flasher it is not necessary to think about the logic of the code, it is not necessary to apply knowledge about triggers in this way of programming.

  Therefore, the following types of*trigger*to be sent originating from microcontrollers:

  - **#email**
  - **#Warning**

- **#message**
- **#voice**
- **#routine**
- **#input**
- **#intranet**
- **#var**
- **#output**

1. #emailnumber

This trigger causes emails to be sent automatically by the Dashboard. Sending the trigger must always be accompanied by a number, which is linked to a specific e-mail pre-configured in the Dashboard by the user (See how to pre-configure an email in section 2.1 of the Dashboard). The content, recipient and source email address must be pre-edited in the Dashboard emails tool. The source email must be a gmail account with the security option enabled.

Example:

- **#email 1**
- Email configured as 1 in Dashboard, with recipientMyIoT@teste.com.brand content “This was an email sent by IoT!”, is sent.

- Every time this trigger with the number 1 is activated, this same email will be sent. This will happen until the user edits it in the Dashboard.

two. #Warningnumber

This trigger causes text warnings to be displayed on the Dashboard. Like the e-mail trigger, the warning trigger must be followed by a number that is linked to a specific pre-configured warning in the Dashboard. The warning text will be displayed in the central window of the "warnings" tool, in the central part of the screen. This will come with the date and time of sending as well as its origin (as notices can be sent between collaborators in different accounts).

Example:

- **# notice 1**
- Notice configured as 1 in the Dashboard, "Good morning user!", is sent.
- Every time this trigger with the number 1 is sent, this same warning will be displayed. This will happen until the user edits it in the Dashboard.

3. #messagetext to be shown on display and Dashboard message

This trigger causes text messages to be sent to the Dashboard message screen and presented by microcontrollers connected to the internet with LED matrices or LCD displays attached. These microcontrollers must use flashes corresponding to the actuator that will be used (LED or LCD Matrix). Flashes can be inserted through the Flasher application.

Example:

- #message Good afternoon!

4. #voicetext that will be read in the sound warning

This trigger causes sound warnings to be played on the Dashboard. Remember to enable these commands on the Dashboard!

Example:

- #voice Hello user!
- If the computer's volume is turned on and the Dashboard is open, the phrase "Hello User!" can be heard.

5. #routinename of the routine to be started

This trigger starts routines configured in the Programar application. Routines can contain a trigger or a sequence of triggers, which seek to perform specific actions.

This function depends on whether your MyIoT package has the "Program Functions" module or not.

Example:

- #morning_warning routine!
- The routine "aviso_matinal" is reproduced. This contains the triggers:

  BR

  # notice 1

#voice A great day for you user!

BR

- Notice 1, configured as “Good morning!” is displayed in the Dashboard bulletin board. Then, the voice message “A great day for you, user!”

6. #inputquestion to be asked

This trigger causes a question to be asked to the Dashboard in the Input field. Just below the question, a fill-in field aims to record the user's response in the cloud. This recorded response can trigger actions if programmed into a microcontroller or in the Program Actions application. For both options, the code must have a reaction to a specific text readout expected for the response.

*Example*BR

- #input Is it dark?
- The user types the answer “yes” in the input field.
- A microcontroller connected to the internet programmed to activate a lamp through a relay if the input answer is yes (if input = yes) activates the lamp.

7. #intranetAE letter

This trigger has the same functionality as the #output trigger, but directly between the microcontrollers. The idea is to be a direct communication between the controllers.

*Example*BR

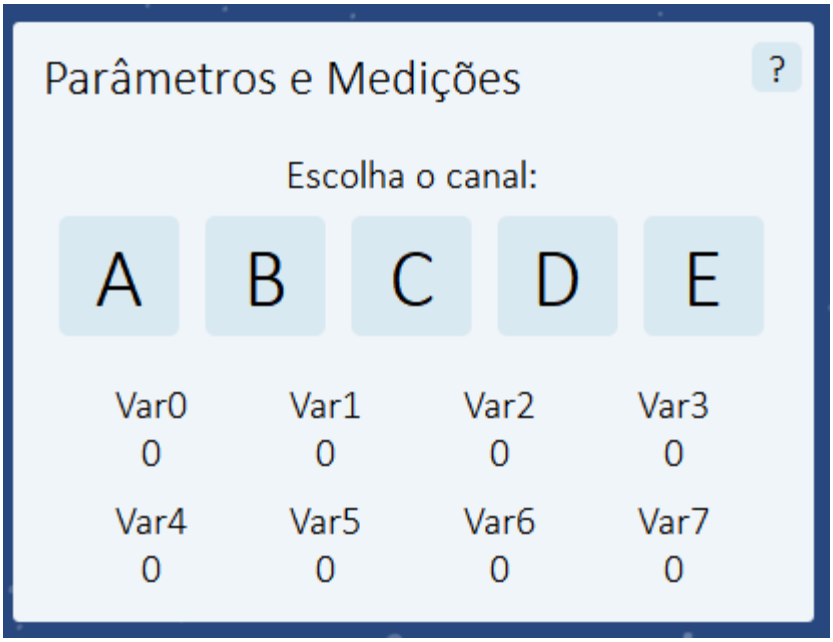
- # intranet C
- A microcontroller connected to the internet programmed to activate a lamp through a relay if the intranet value is “C” (if intranet = “C” activates the lamp).

8. #varvariable_numberchannel_letterBRtext_or_number

This trigger sends anumeric or text valueto one of8 variableslinked toeach channeland consequently to the microcontroller.

*Example*BR

- #var 2A = bright
- A microcontroller connected to the internet on channel A receives “luminous” as the value of its variable 2.
- A code created in the program application defines that if var2A = bright, the trigger d is sent to the channel, this one in which the microcontroller is configured to turn off a lamp connected to a relay upon receipt of this command



BR

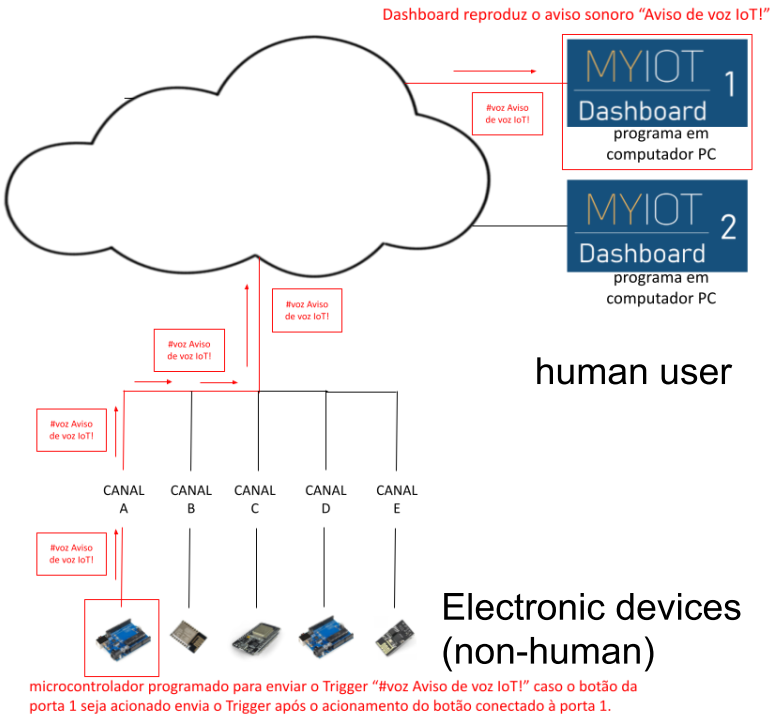
9. #outputchannel letter =AE letter

This trigger sends an AE letter, uppercase or lowercase, to a specific channel linked to a microcontroller. The trigger needs two variables to be specified by the user:which channelandwhich letter to send(AE, uppercase or lowercase)BR

Example:

- #output A = c
- The letter “c” is sent to channel A.
- An internet connected microcontroller configured to start a motor if the letter “c” is sent to it starts the motor.

Example of a trigger system on a microcontroller:



1.4. HOW TO PROGRAM A TRIGGER IN A MICROCONTROLLER?

➢ **Programming a Trigger in C++ by a generic IDE**

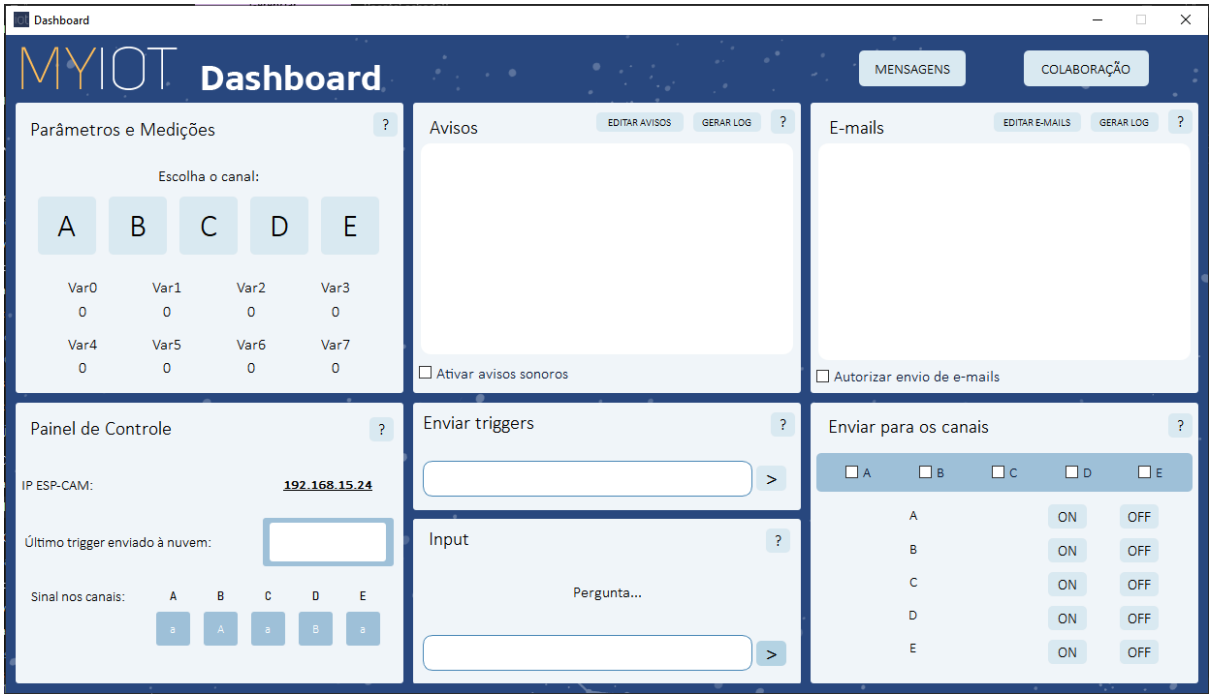
```
println("#voice IoT voice prompt!");
```

2. MYIOT PACKAGE APPLICATIONS

2.1. DASHBOARD

The Dashboard is the communication center of the MyIoT system. The program consists of a computer or cell phone window that allows the user to view various information about IoT devices. In addition, the software enables user interaction with projects by sending triggers and editing parameters in real time. The program comes with several tools already built-in, from which it is possible to create and manage different types of projects. To open the application, click on Dashboard in the main menu of the MyIoT package.

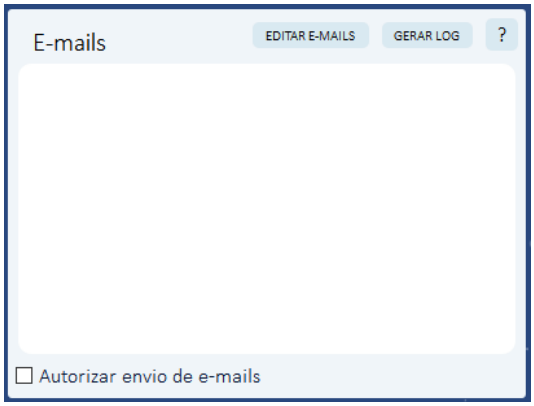
The main application window is:



Your tools are:

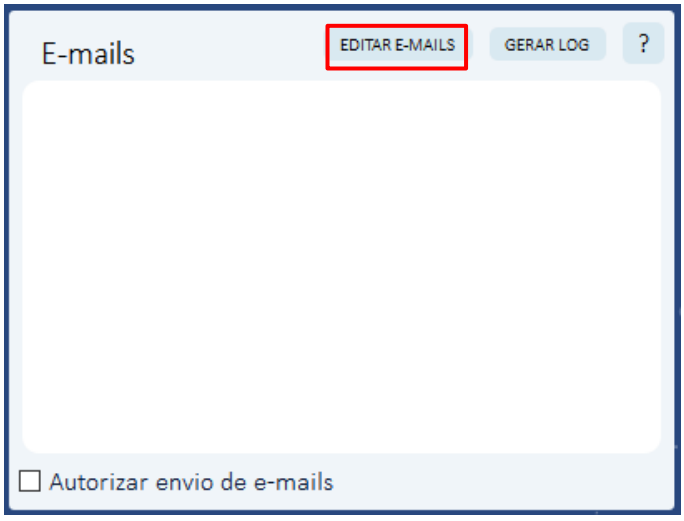
2.1.1. SENDING E-MAILS

The e-mail functionality consists of automatically sending a pre-programmed e-mail if the specific command “#email number” is sent to the Dashboard. The number indicates which email to send, as multiple emails can be pre-programmed.

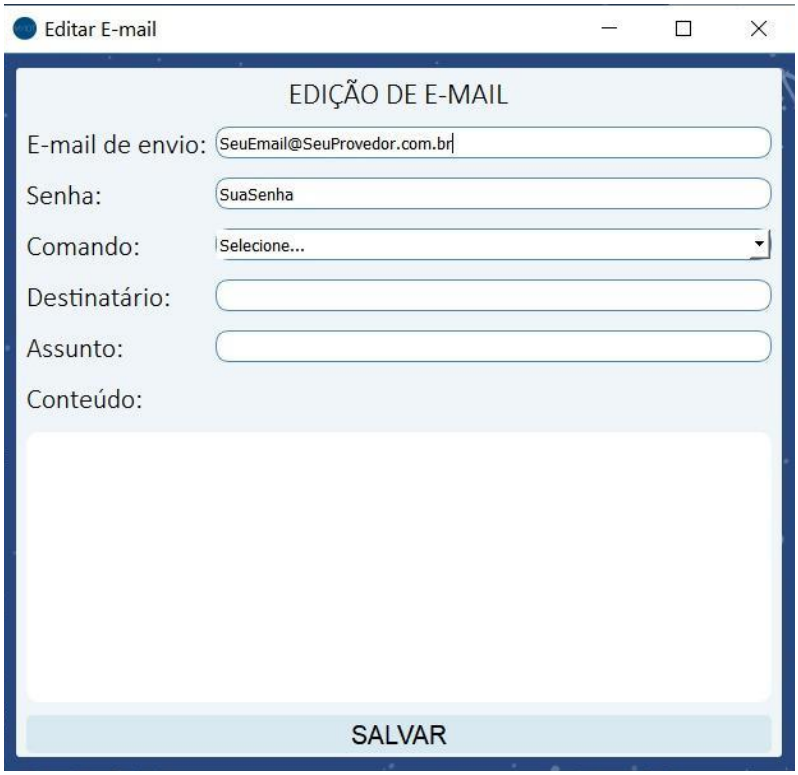


PRE-PROGRAMMING THE EMAILS

Enter email editing mode by clicking on “Edit Email”.

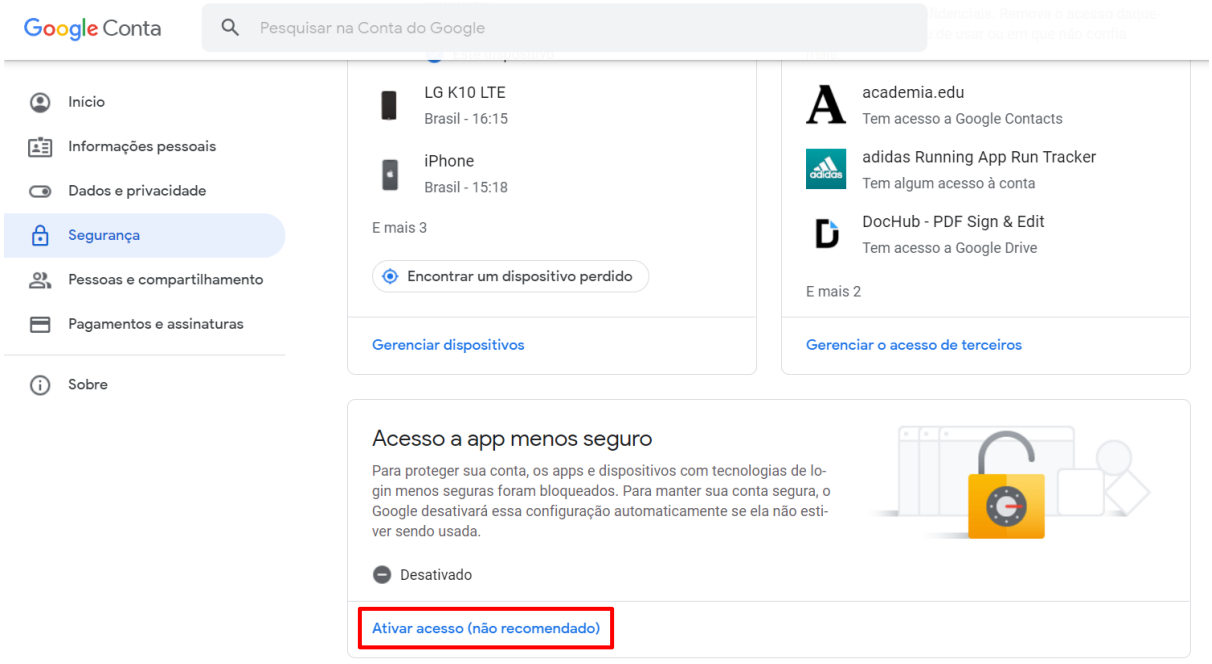


In the window that will open, the variables are the following:

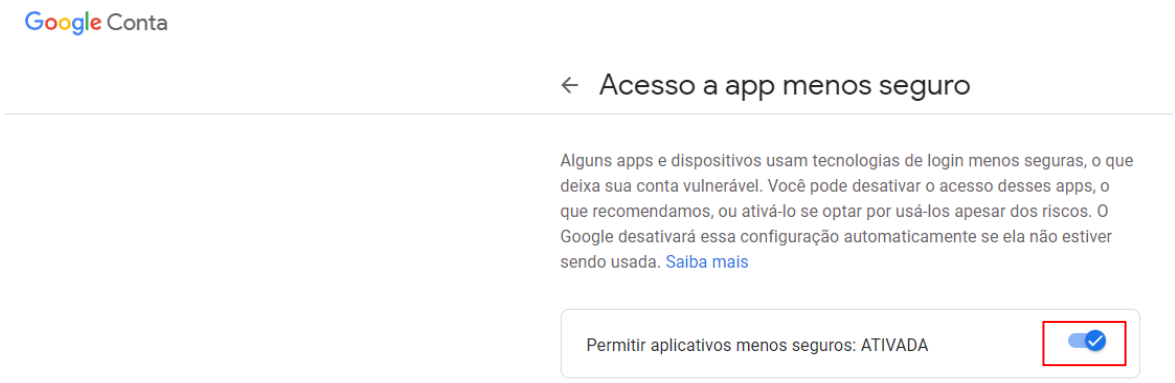


shipping emailBREmail address from which emails will be sent. This needs to be a Gmail account with the security feature enabled. To activate this feature from a Gmail account, follow these steps:

- Log into your Google account (https://account.google.com/BR If you don't have one, click on "create an account" and return to the same link after completing the registration.
- Go to “Security”, scroll down to “Less secure app access” and click “Enable access”.



- In the window that opens, select the option “Allow less secure applications” so that it says “ON”.



PasswordBRPassword for the email address from which emails will be sent.

CommandBRNumber corresponding to the email in question. Click the down arrow and choose one of the listed numbers, then record an email for this specific command. You can have as many recorded emails as you want, with different subjects and recipients. Each email will be recorded with a unique number for that email (1, 2, 3...), which can be called in the trigger as “#email 1”.

Recipient: Email address to which the email will be sent.

Subject: Email subject.

Contents: Email content.

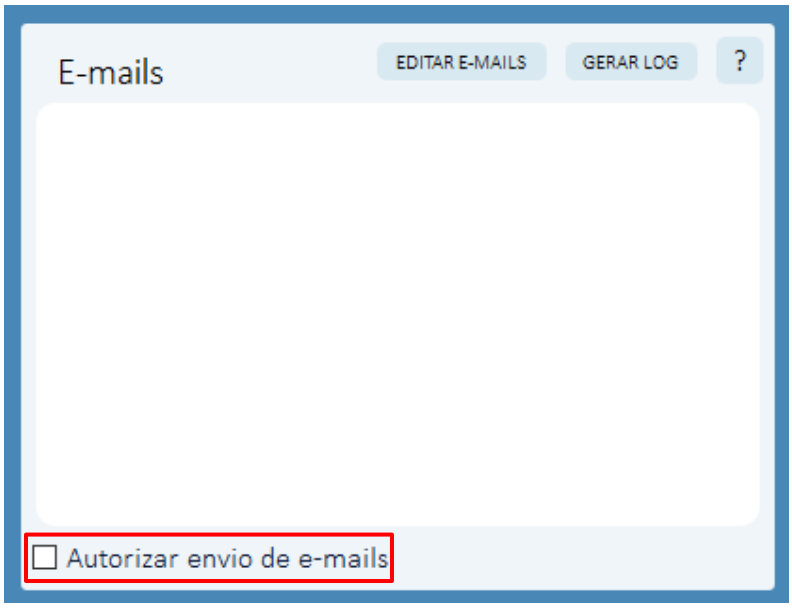
Finishing editing emails, click save and the window will close automatically.

Attention: Be aware that Google limits the amount of emails you can send each day.

Probably to avoid sending SPAM. Avoid sending more than 200 emails during the same day.

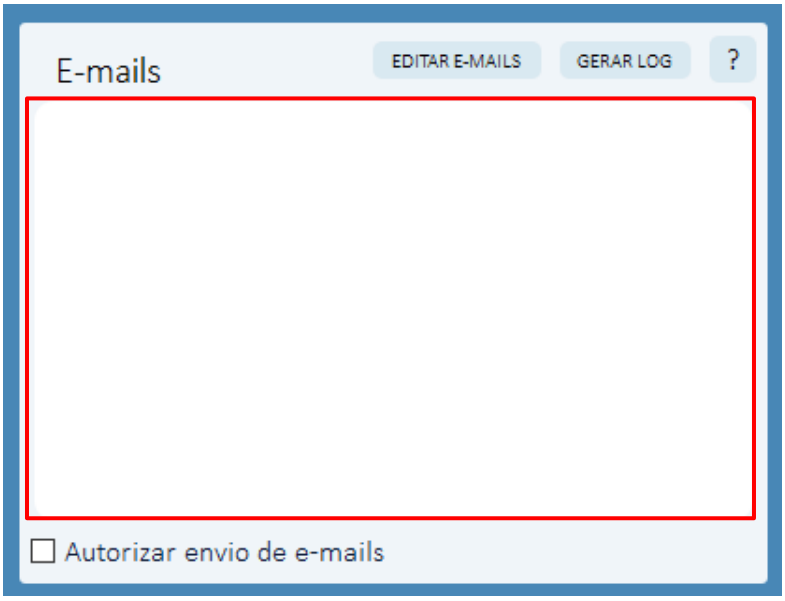
AUTHORIZING THE SEND OF E-MAILS

Check or uncheck the box in the line “Allow sending emails” to allow or block the sending of emails through the Dashboard. This tool is specific to the Dashboard in which it is used, since it is possible to run more than one Dashboard with the same MyIoT account.



VIEWING SENT EMAILS

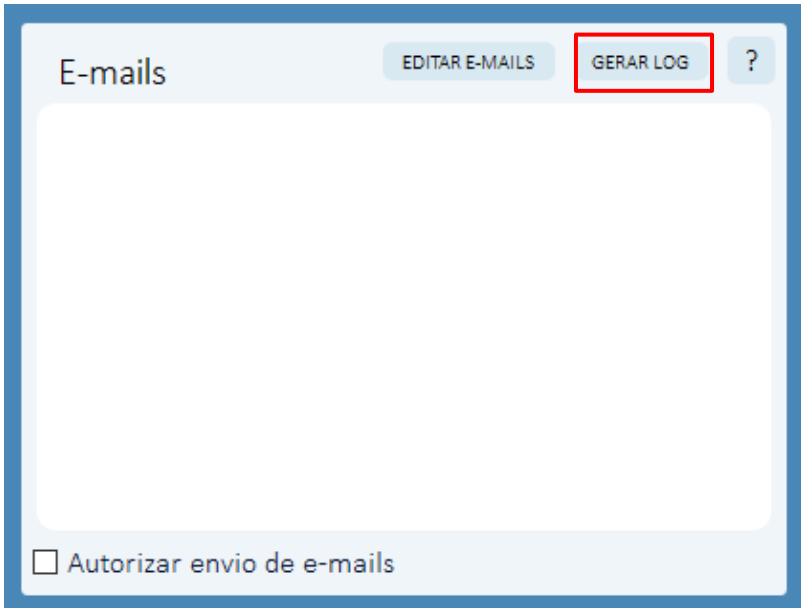
In this window appear the e-mails sent by the open Dashboard. Since it is possible to have multiple Dashboards opened with the same account, the emails sent only appear in the specific Dashboard through which they were configured and then sent.



GENERATE LOG

The Generate Log button creates a text document with all emails sent from the computer in question, as well as their contents.

The Log is saved in the folder where the MyIoT executable is located.



2.1.2. PARAMETERS AND MEASUREMENTS

The parameters and measurements window presents updated data provided by sensors connected to a microcontroller connected to the internet, regardless of location.

The tool presents the real-time status of up to 8 variables in each channel, corresponding to specific microcontroller ports.

2.1.3. NOTICES

The warning window shows warning triggers sent to the Dashboard and plays sound warning triggers (which are text readings with the computer's voice).

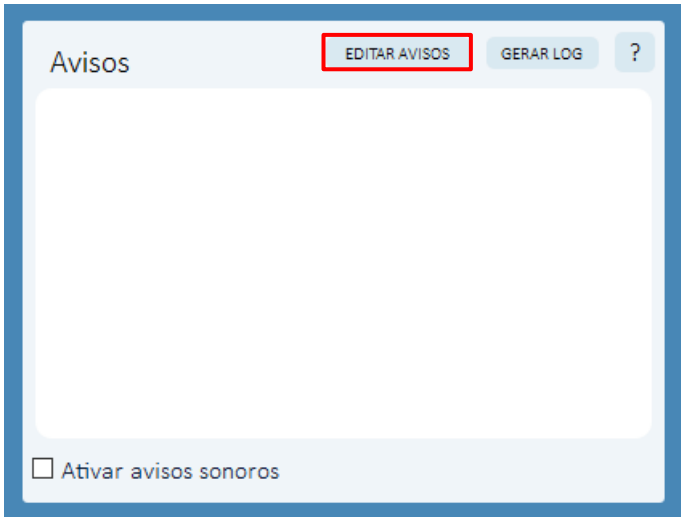
VISUALIZATION OF NOTICES

Shows text warnings based on events programmed into microcontrollers.

*trigger*corresponding: “#text notice”.

EDIT NOTICES

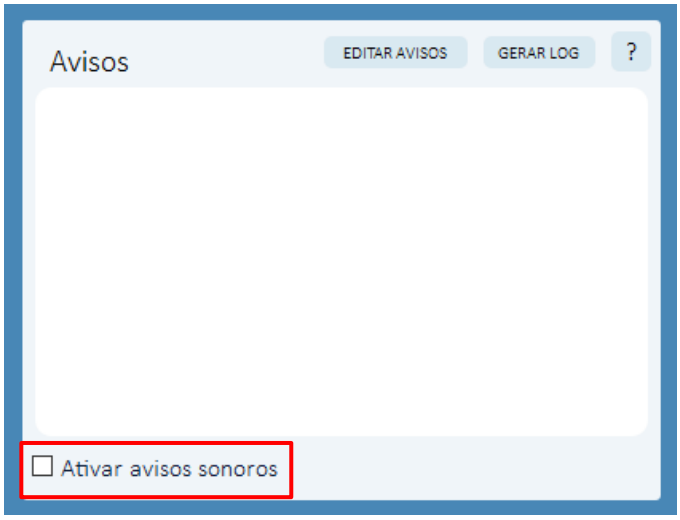
To edit the preconfigured warnings to fire with the #warning trigger, click Edit Warnings.



In “command” choose a number, and write the pre-configured warning you want for that number. Don't forget to click save after each content edit.

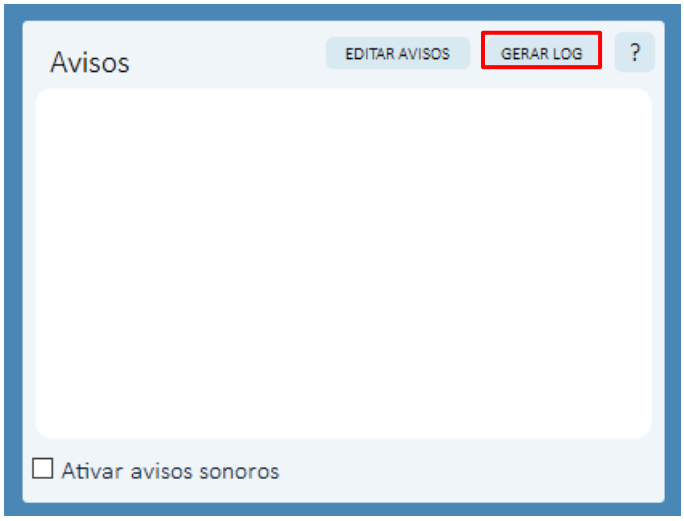
ACTIVATING AUDIBLE WARNINGS

Check and uncheck the box**Turn on sound alerts?**to enable the tool. The Dashboard plays audio prompts if the trigger #voice followed by text is sent to it. Example: “#voice IoT voice prompt!”.



GENERATE LOG

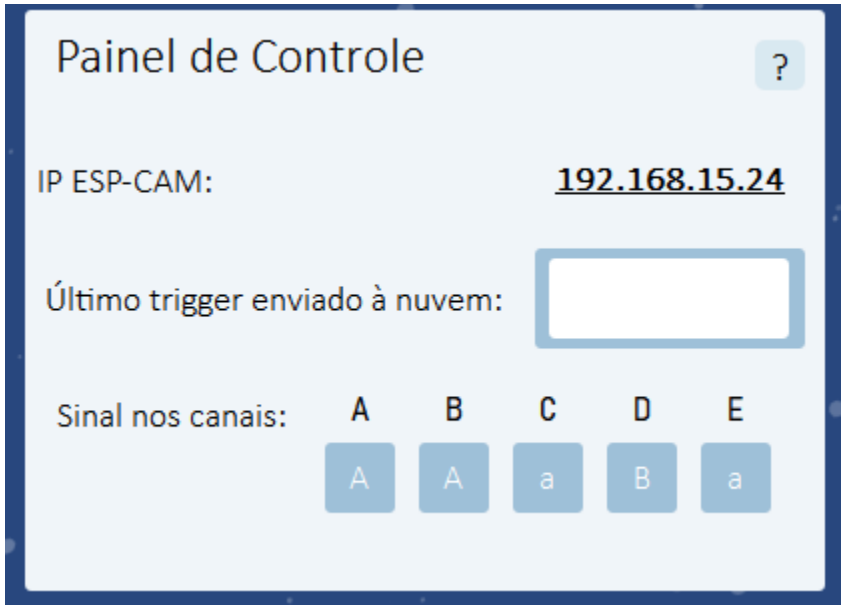
The Generate Log button downloads a text file with the history of recent warnings. The generated file is in the same folder as the MyIoT package executable.



2.1.4. CONTROL PANEL

The Control Panel window shows the*triggers*"running" through the channels and on the Dashboard in question. In addition, you can click on the IP that your

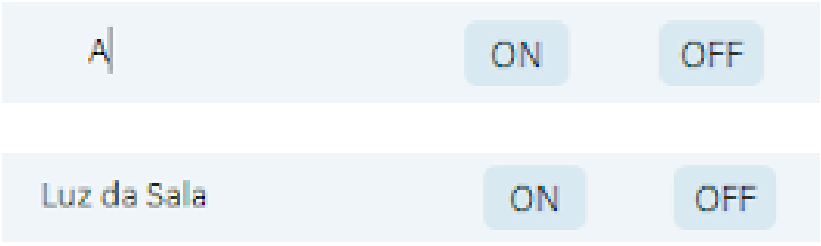
ESP-Cam generates to see what it is showing, a very important function that is extremely necessary for any IoT project.



2.1.5. SEND TO CHANNELS

This window allows*triggers*can be sent to specific channels to be chosen by the user. Since each channel corresponds to a specific microcontroller, this tool allows direct communication with a microcontroller programmed to react to these commands.

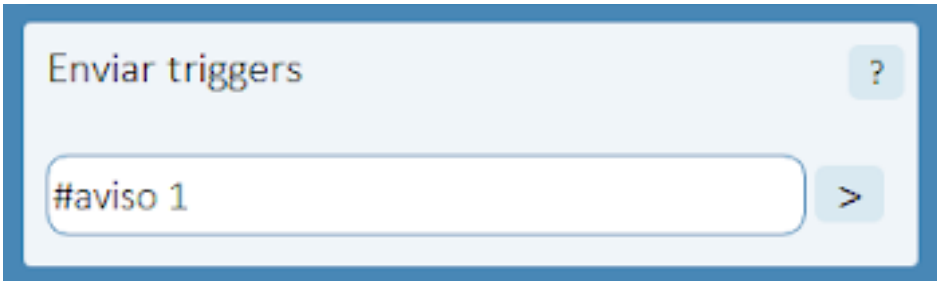
Each button sends the letters (AE [ON]; ae [off]) as a trigger to the microcontroller. By clicking on the letters on the left, you can edit the text to a specific description, such as “Room Light”.



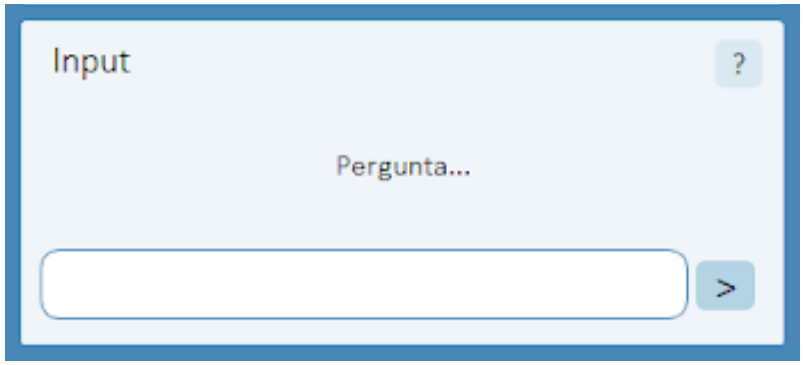
Even changing the description, the trigger sent will be the original letter of that column in uppercase and lowercase. Test and view the value of the buttons on the control panel. In the example above, when clicking “ON” in the Room Light line, the trigger sent will be “A”.

2.1.6. SEND TRIGGERS

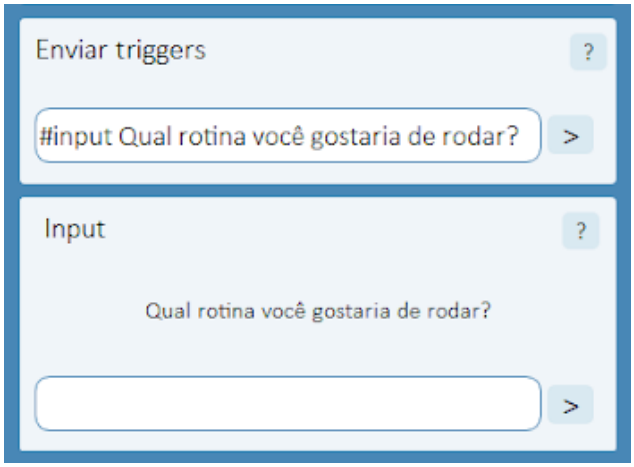
The Send Triggers window allows any trigger to be sent to the system. These can be seen in detail in section 1.3.



2.1.7. INPUT



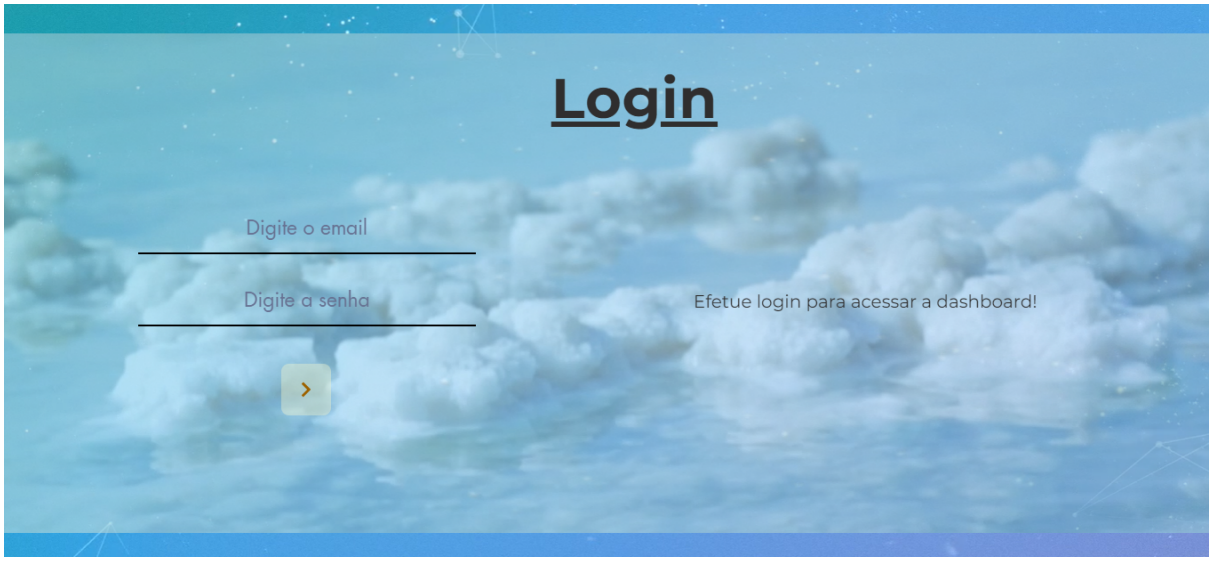
This window functions as a question and answer system, with the purpose of triggering specific actions in a system based on the user's response. The question will appear in**Question…**if the #input query command is sent to the system.



The response will trigger actions if the response word (exactly the same) has been previously programmed in the Schedule application to initiate something.

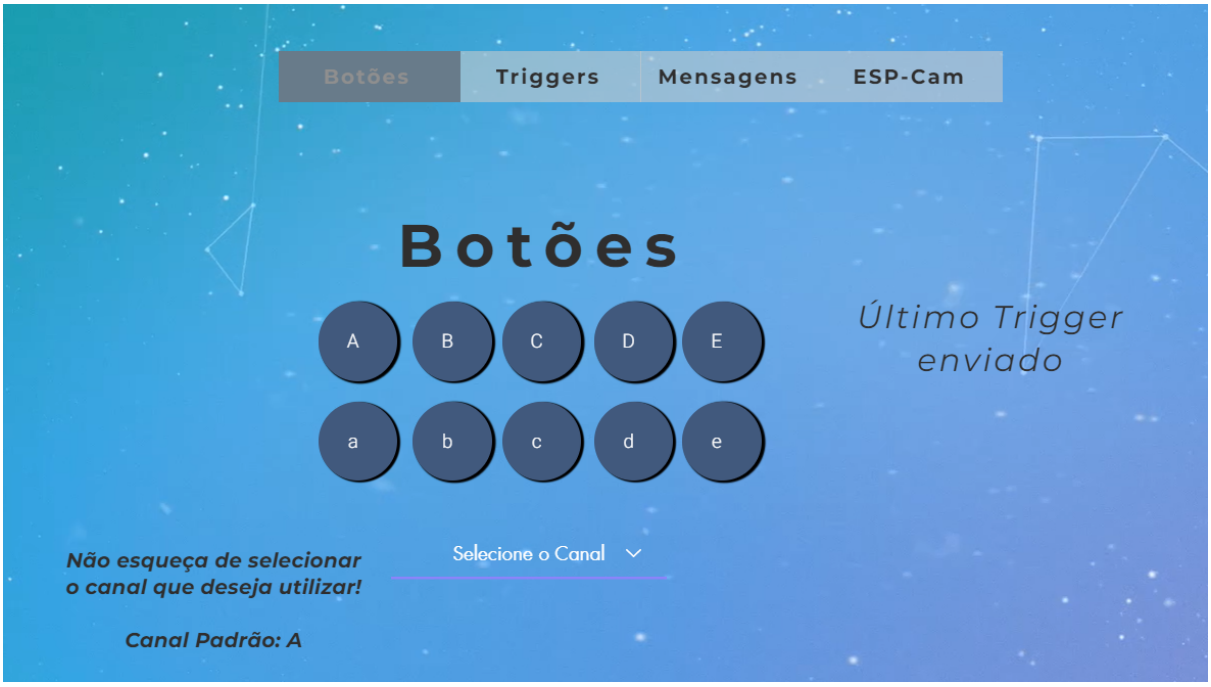
2.1.8. ONLINE DASHBOARD

We also have the Dashboard online, accessible at the linkhttps://myiot.space/dashboardBR When opening the link this screen will appear:



Enter your login account and password to login, if the login is correct then click Login.

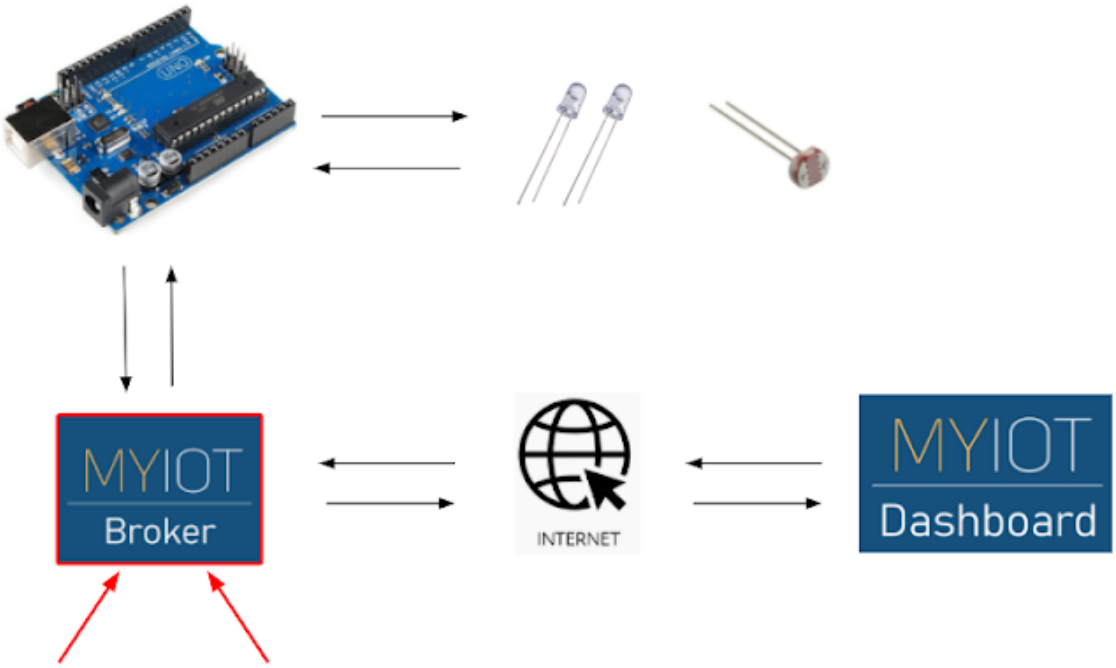
Use the top buttons to navigate through the menus, in the Buttons page you can select a channel and send lyrics to it, in the trigger menu you can select triggers and send to the cloud, in the Messages page you can look at the last Message received by the cloud.


Botões
Triggers
Mensagens
ESP-Cam
Botões
A
B
C
D
E
a
b
c
d
e
Último Trigger enviado
Não esqueça de selecionar o canal que deseja utilizar!
Selecione o Canal
Canal Padrão: A


Botões
Triggers
Mensagens
Envio de triggers
Escolha o Trigger
Digite a mesagem
Enviar

2.2. BROKER

The word “Broker” comes from English, and means “realtor”. The concept revolves around the intermediation practice of this profession, where the professional acquires knowledge of houses and buildings and communicates information about them to someone else. In an IoT project, the “Broker” is a computer program that acquires information from sensors and actuators, and then communicates it to the internet. The application aims to intermediate a microcontroller with the internet, enabling communication with other devices and design elements.

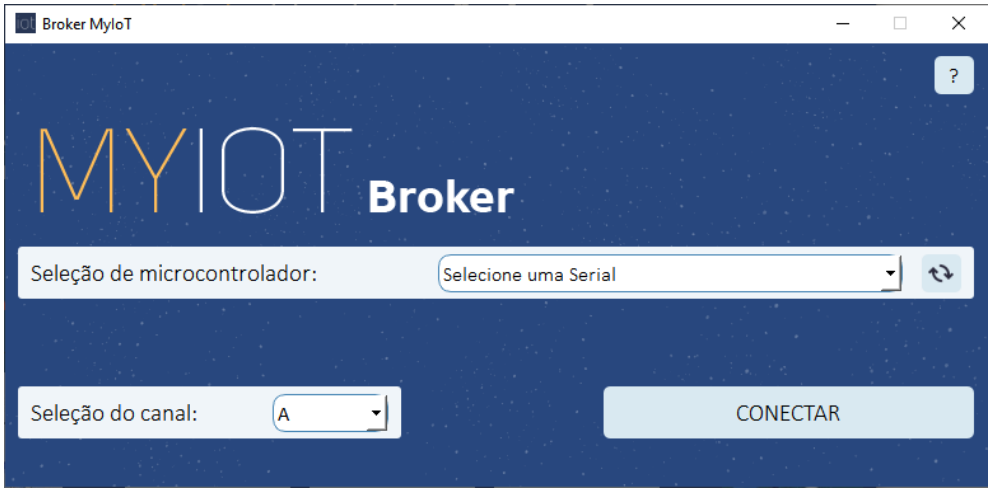


There are two ways to use the “Broker”:

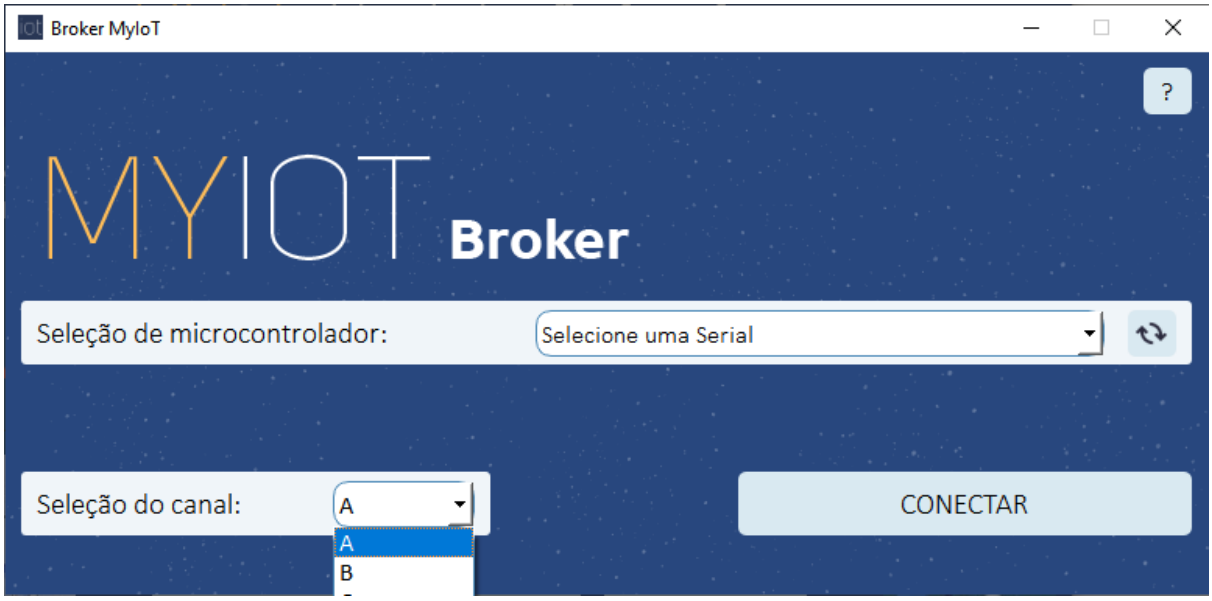
- **Desktop Broker**- The Desktop broker works on a PC computer, requiring the microcontroller to be connected to the machine for as long as it is working (via USB AB cable). To use it:

- In the MyIoT main menu, click on “Broker”.

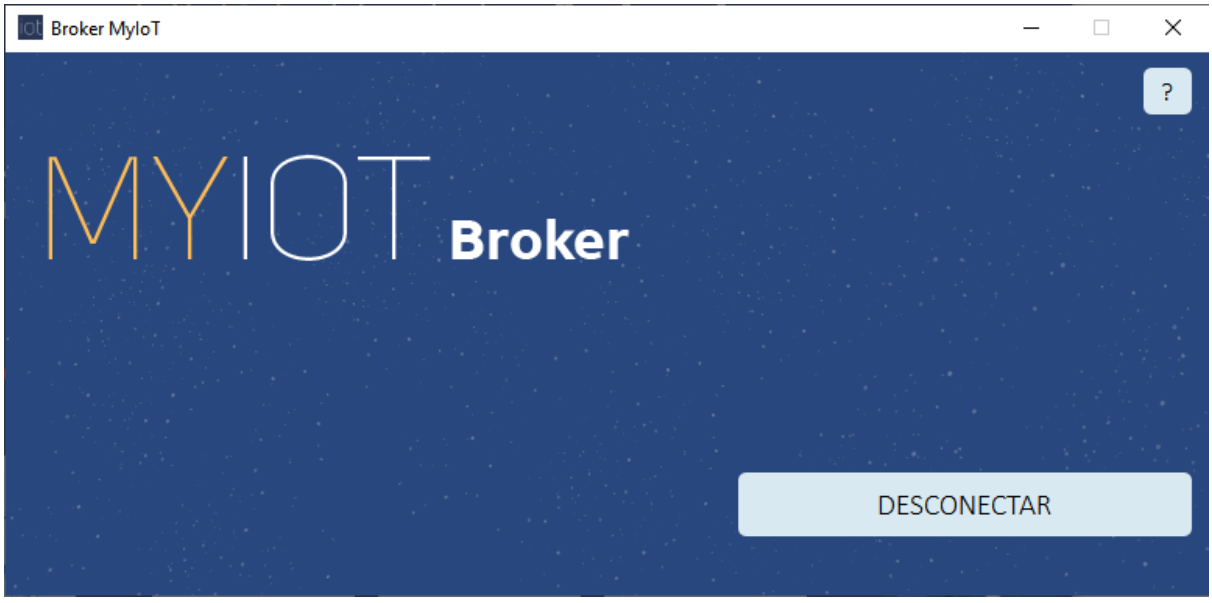
- When opening the application, click on the button  and the program will automatically recognize the microcontroller connected to the machine.



- After the microcontroller is listed, select in the first line the channel that the microcontroller will use. You only need to worry about this if you want to use more than one microcontroller connected to a Broker: two different microcontrollers cannot be on the same channel.



- Click CONNECT. The appearance of the Connected prompt will indicate that the system is connected.



- **Mobile Broker**- The mobile broker is a way to enable wireless connection of microcontrollers to the internet. The code is inserted into an ESP board (microcontroller capable of communicating with the internet via WI-FI), and the original microcontroller only needs to be connected to the “Broker

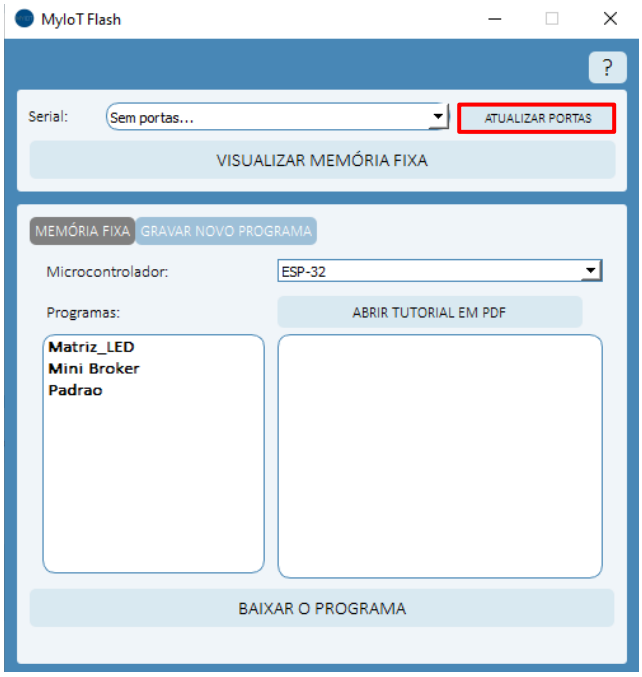
microcontroller” and to a power source. To use it, it is necessary to insert the Mobile Broker code into the ESP board using the MyIoT Flasher application. Mobile Broker code is available for ESP12, ESP32 and ESP01. For more information about MyIoT Flasher, go to section 2.5 - Flasher.

2.3. FLASHER

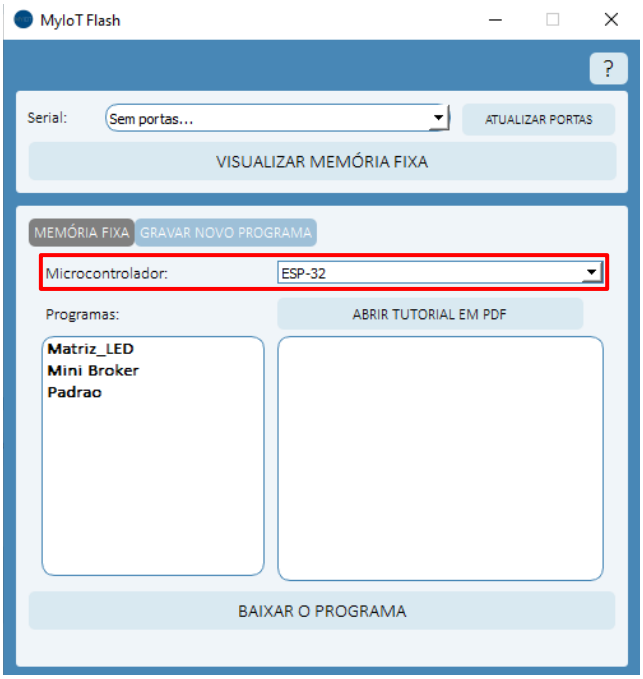
Flasher is an application that automatically inserts ready-made codes into microcontrollers. The software contains a vast library of codes for different controllers, with different utilities and applications. In general, the program can be used for two groups:

FLASHER FOR MICROCONTROLLER WITHOUT WI-FI COMMUNICATION

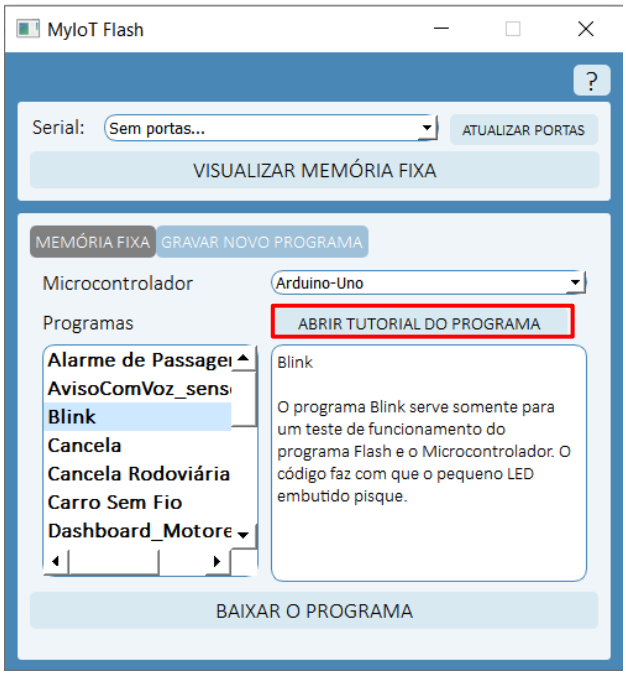
- Click Update Ports. The microcontroller connected to the machine will appear automatically.



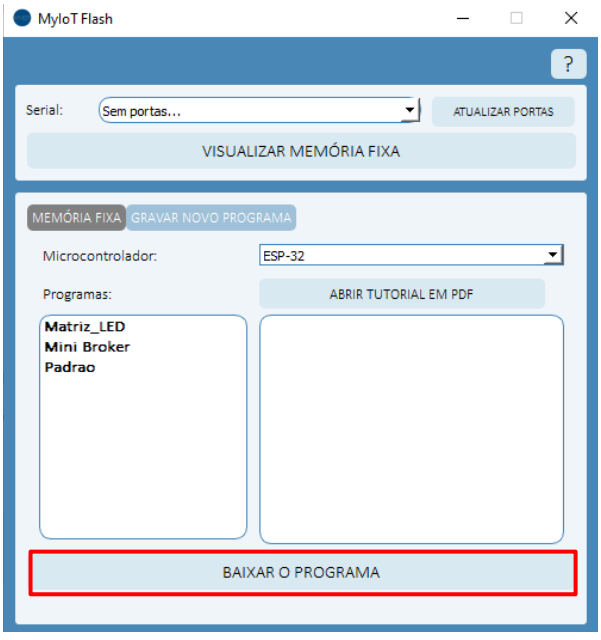
- In Microcontroller, choose the microcontroller corresponding to the one to be used



- Under Programs, choose the code you want for the microcontroller. In the right center of the screen, a short text describes the code. For more information about each program, click Open PDF Tutorial.
    a. Note that not all codes have a PDF available, in cases of very simple codes, just the text description is enough.



- Click Download the program. After loading, a warning will indicate the success of the Download.



FLASHER FOR MICROCONTROLLER WITH WI-FI COMMUNICATION

Unlike microcontrollers without WI-FI communication, now it is necessary to enter the data of the WI-FI network to be used.

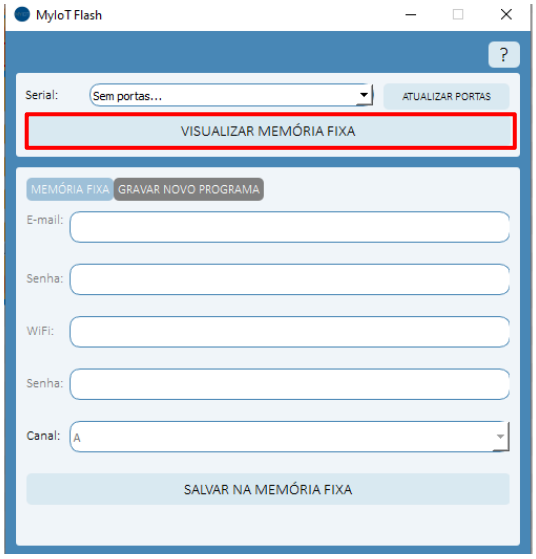
- Click Update Ports. The microcontroller connected to the machine will appear automatically.

- In Microcontroller, choose the microcontroller corresponding to the one to be used.

- Under Programs, choose the code you want for the microcontroller. In the right center of the screen, a short text will describe the code. For more information about each program, click Open PDF Tutorial.

- Click Download the program. After loading, a prompt will indicate Download success.

- Click View Fixed Memory. Another tab will open.



- The email and password data will be filled in automatically, as they correspond to the MyIoT Space account data used to open the MyIoT Package. Do not change this data. Fill in the name of the WI-FI network and the password, taking care not to make typing mistakes.

- Choose the channel that the microcontroller will use. It is only necessary to pay attention to this line if more than one microcontroller will be used in the same MyIoT Package account. For this situation, it is necessary to define a specific channel for each microcontroller.

- Click Save to Fixed Memory. If the button turns gray and then black again, your microcontroller is ready!

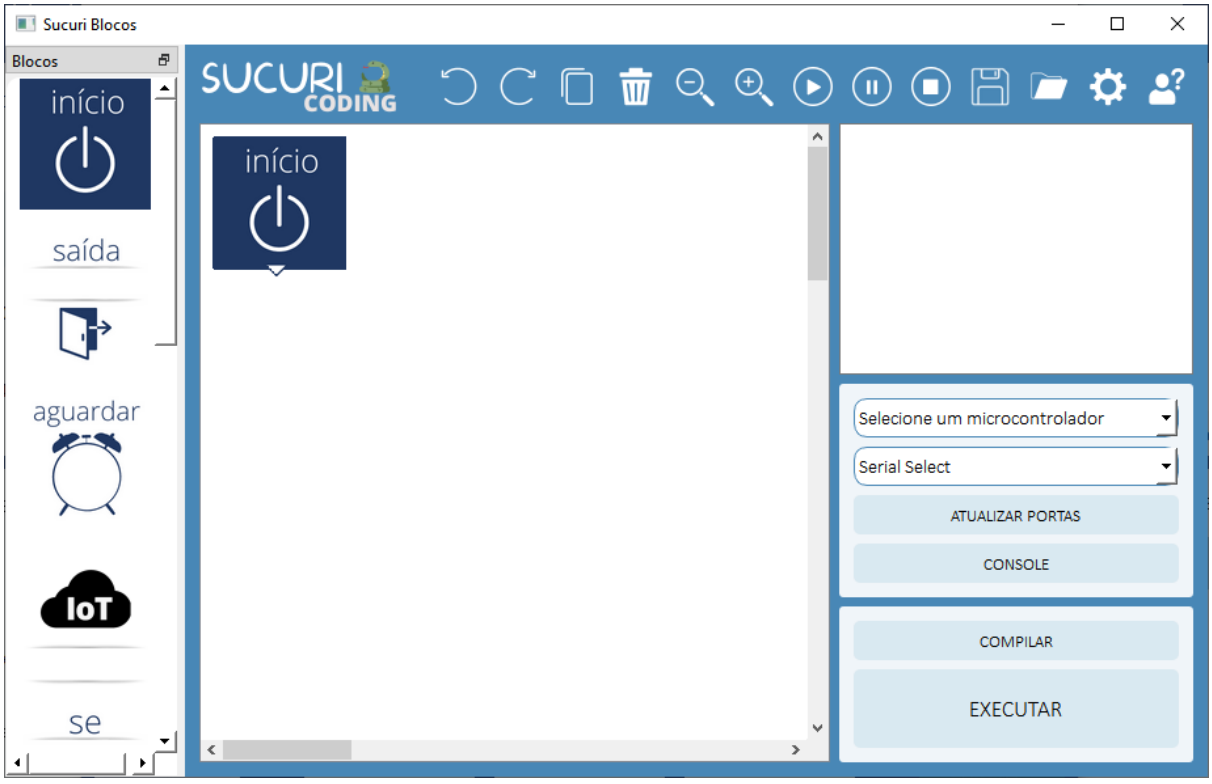
1.1. SUCURI CODING

1.1.1. INTRODUCTION AND INITIAL CONFIGURATION

Unlike the Program Functions application, Sucuri Coding aims to enable accessible programming from a flowchart.

To open the application, click on Sucuri Coding in the MyIoT Package main menu.

The main screen of the application can be seen below:

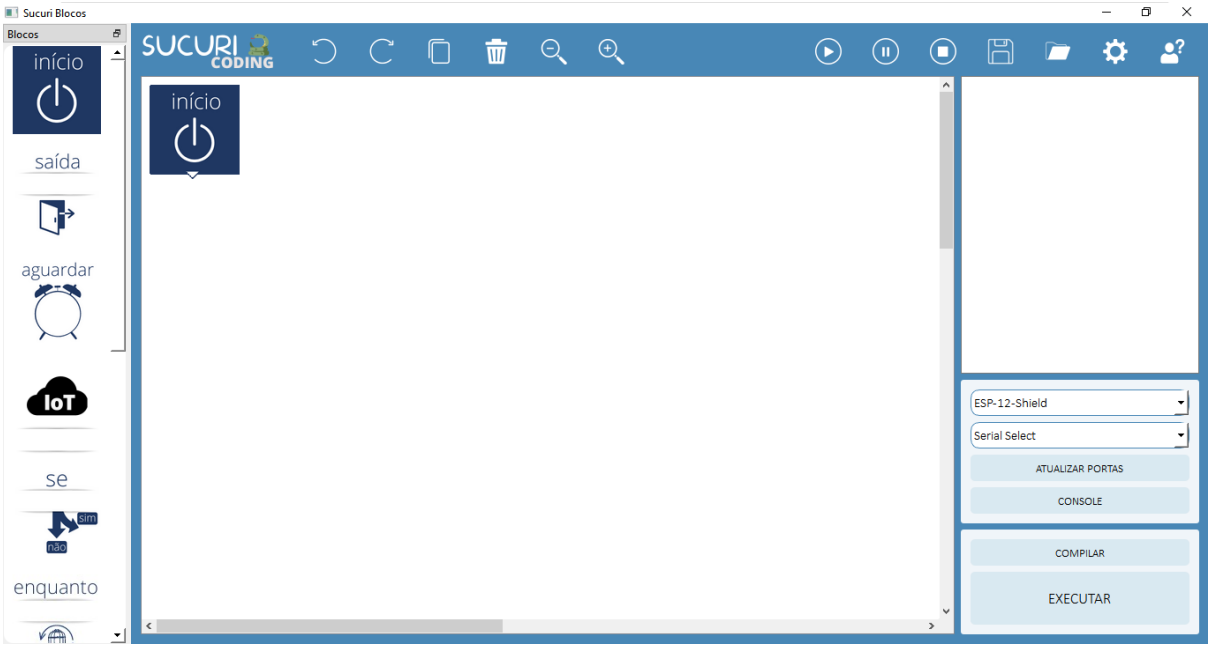


Before starting to create the flowchart, it is necessary to prepare the microcontroller so that it “understands” the block diagram. To do so, connect the microcontroller to the machine, choose the correct type in the side menu and click Update Ports.

When the port appears, click Settings. After loading, your microcontroller will be ready to be used with the application.

1.1.2. USING THE APPLICATION

On the main screen on the left, you can find the main flowchart programming environment.



The block flowchart programming logic is based on the construction of a sequence of events based on sensors and actuators connected to the microcontroller. These events are represented by blocks, which must be placed in the programming environment and connected from lines according to the specific sequence of events that the user wants. The program will run from top to bottom, starting from the Home block.

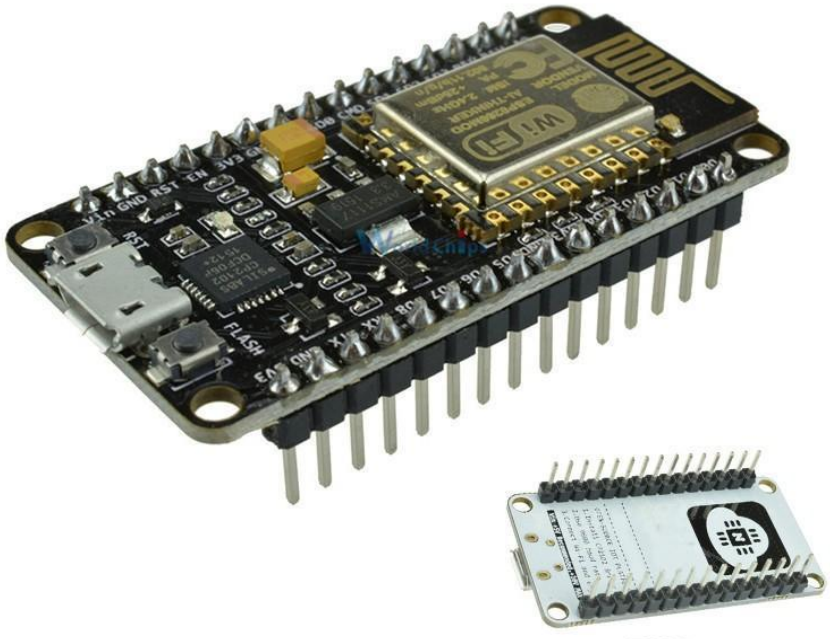
Example:

In this code created, port D7 will be activated and, after 2 seconds, it will be turned off.

The code will run as soon as the microcontroller with the code already downloaded and disconnected from the computer is connected to a power source. To download the program to the microcontroller click on Compile to generate the text code and then click on Run.

1.1.3. DOOR OPERATION LOGIC

The program works by reading and/or energizing ports (“ports” are the pins on the side of the board) specific to the microcontroller.



These ports can be of two types:

- **PROHIBITED**- An input port has the technical name of INPUT. This type of port serves to input information into the microcontroller. This occurs from the connection of sensors, which are nothing more than devices that gather information from the environment and communicate to the microcontroller. This type of door is used for projects where it is necessary to monitor the work environment, communicating the user when something happens or

automatically performing actions based on previous procedures defined by the user.

Example of devices to be connected to this type of port:

- ➢ Light Sensor (LDR)
- ➢ Infrared Sensor
- ➢ Touch Sensor (button)
- ➢ Ultrasonic Distance Sensor

- **EXIT**- An output port has the technical name of OUTPUT. This type of port is used for signal output, that is, for energy output to drive actuators. These devices perform actions, such as turning on a light, turning an engine, or making a sound. As their name describes, they do something to act.

Example of devices to be connected to this type of port:

- ➢ led
- ➢ Electric Motor
- ➢ Buzzer (beep sound)
- ➢ Relay (used to switch on devices with a higher electrical load, such as a light bulb)

There is another important feature about all ports: these can be digital or analog:

DIGITAL PORT -A digital port is a port that sends or understands only two types of information: 0 or 1. 1 corresponds to the on state of the port, while 0 corresponds to the off state. A digital actuator can only be on (1) or off (0), while a digital sensor only communicates that it is reading something (1), or not reading anything (0).

Example of digital devices to be connected to this type of port:

- ➢ Touch Sensor (button)
- ➢ LED actuator

ANALOG PORT -Unlike a digital port, this type of port sends or understands a large number of numbers: from 0 to 1023. It is used with analog sensors for more precise monitoring of some information (luminosity being at 250, for example), or for more specific port power-up (like turning on an LED with only 70% of its capacity, for example). While all actuators can be used on analog OUTPUTs (since any electronic device can be driven with partial power), there are specific actuators designed to be controlled more precisely, like a servo motor.

Example of analog devices to be connected to this type of port:

- ➢ Light Sensor (LDR)
- ➢ Servo motor actuator

When creating the flowchart, the user must specify the sensor or actuator that he wants to control from the port where it is inserted. However, it is necessary to pay attention that there are pre-defined ports for each type of device. This does not occur in classic programming with written codes, but it is a convention adopted by Sucuri Coding to facilitate the user experience. The port map can be seen below:

The port categories are:

APPETIZER

DIGITAL INPUT- There are two DIGITAL INPUT ports (D5 and D6), and these must be used for signal input from digital sensors. If an analog sensor is inserted into this type of port, it will not work.

ANALOG INPUT -There is an ANALOG INPUT input (A0), and this must be used for signal input from an analog sensor. If a digital sensor is inserted in this type of port, it will only be able to report 0 or 1023.

ULTRASONIC SPECIAL CASE -Port mapping includes a special case for the ultrasonic analog sensor. Since this sensor needs 4 ports to work, ports D0 and D5 are ready to be used with it. It is important to point out that while you are using it, these ports can no longer be used as INPUT and OUTPUT.

OUTPUTS

OUTPUT -There are two OUTPUT ports (D0 and D7), and these must be used for signal output to actuators. For motors, there are specific ports marked MINI MOTOR DRIVER.

MINI MOTOR DRIVER (ENGINES)- The MINI MOTOR DRIVER ports correspond to four signal outputs made specifically for motors connected to a MINI MOTOR DRIVER (D1, D2, D3 and D4). The specific connections between each microcontroller port, Mini Driver port and the motors themselves can be seen below.

1.1.4. PHYSICALLY CONNECTING A SENSOR OR ACTUATOR

When connecting your device to the microcontroller, there is another detail to pay attention to in addition to the main port: for actuators, it is necessary to connect the other pole of the device to the GND port. For some sensors, which have 3 wires, it is necessary to connect the VCC and GND port in addition to the signal port. Below are some examples of physical connections.

Example 1 - LDR Analog Light Sensor

Example 2 - Infrared Digital Sensor

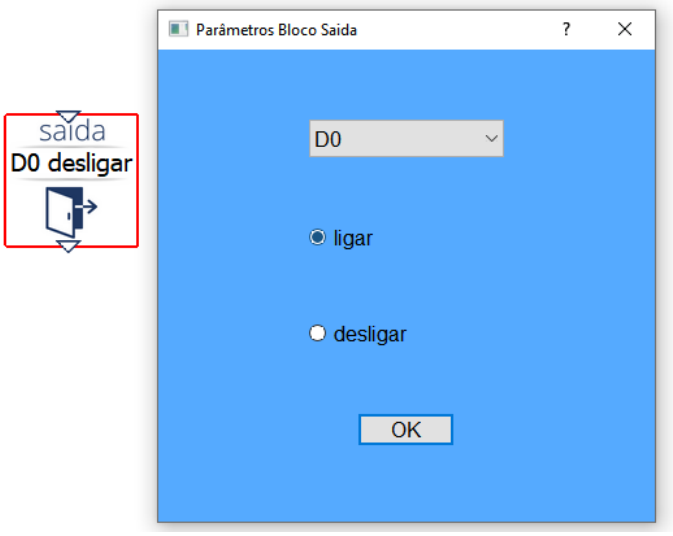
Example 3 - LED actuator

1.1.5. USING AVAILABLE BLOCKS

The following blocks are available for use:

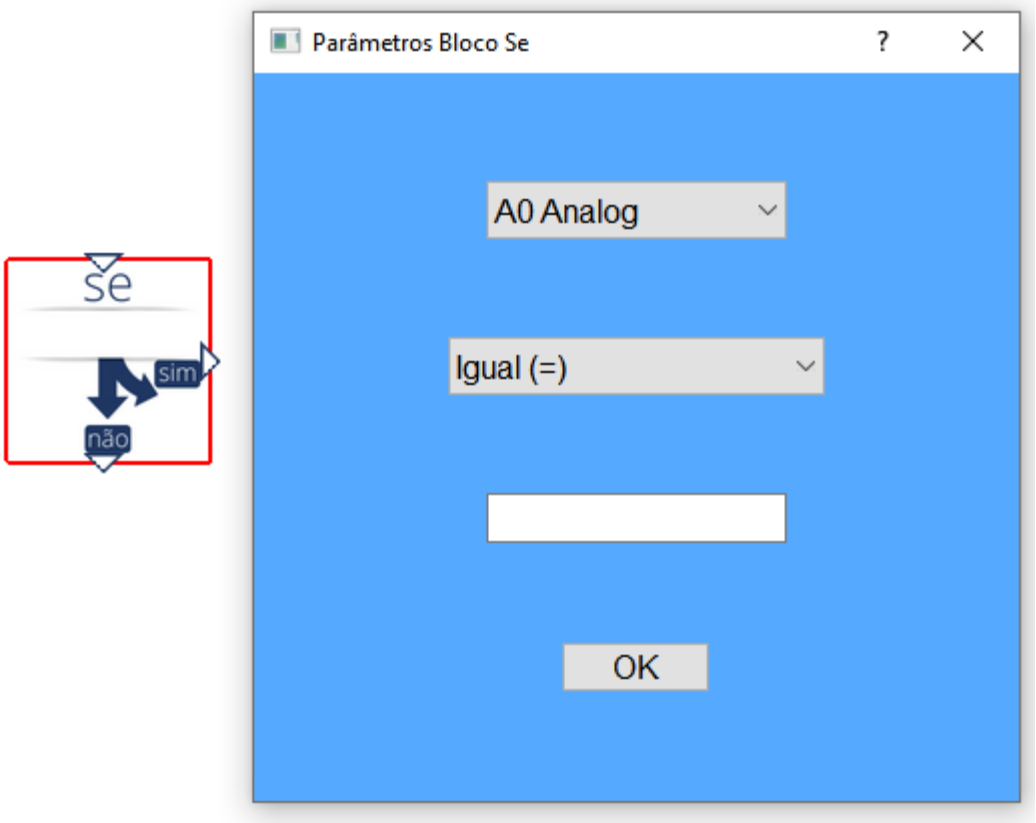
OUTPUT BLOCK - This block turns an output port on or off, that is, OUTPUT ports. When inserting the block, double-click on it and select the action (on or off) that you want this block to perform on the OUTPUT port of your choice.

Ex.



IF BLOCK (INPUT) - Referring to the IF command, in English, this block creates a bifurcation, and serves to create a condition that decides the course of the sequence of events. This condition is based on a reading from a sensor, which must be inserted into INPUT ports. When placing the block, double-click and choose the reading and comparison that the decision is based on. To do so, choose between the commands Equal (=), Less than (<), Greater than (>) or Not equal to (!=). The block output will be YES or NO, referring to the defined event.

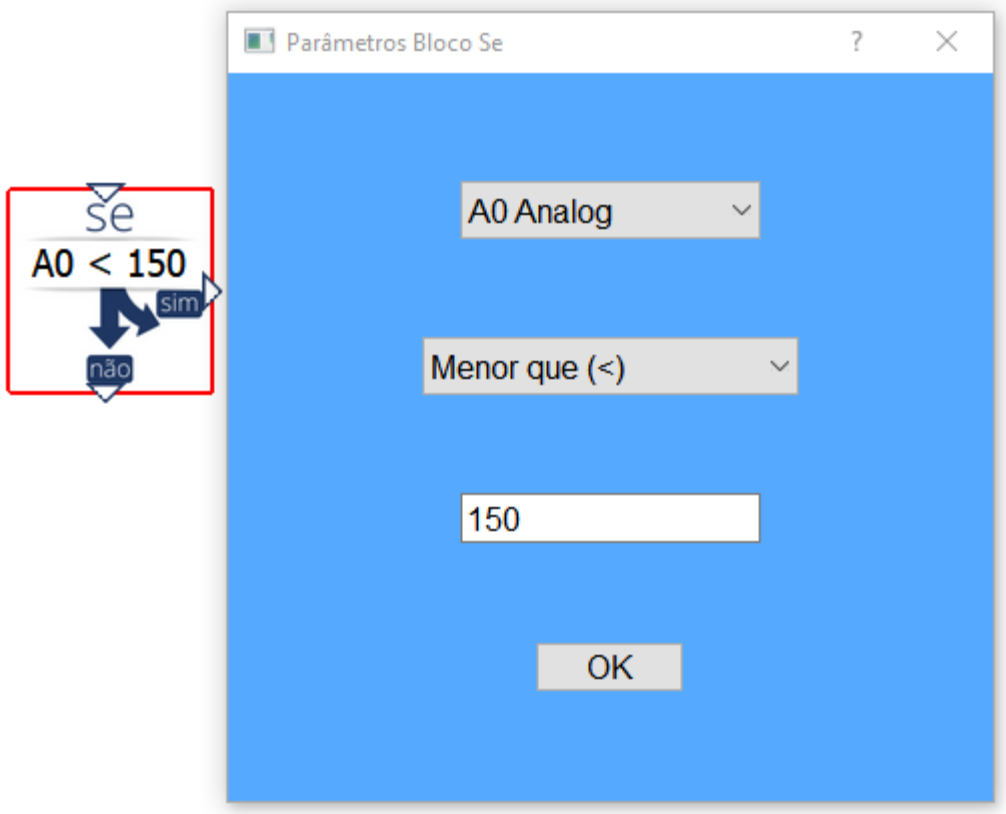
Ex.1



D6 is a DIGITAL INPUT port. If a digital sensor connected to port D6 reads signal 1, OUTPUT port D7 is turned on. If the signal is not 1, port D7 will be turned off.

APPLICATION - This program could describe a digital touch sensor (button) inserted in port D6 that, when pressed (reading = 1, on), turns on an LED connected to port D7.

Ex.2

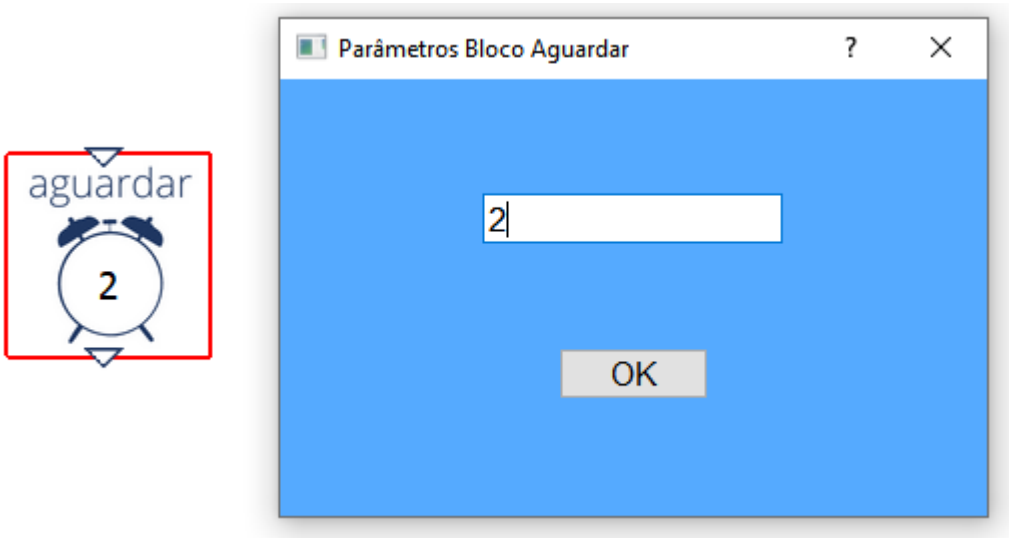


A0 is an ANALOG INPUT port. If an analog sensor connected to port A0 reads a value less than 150, OUTPUT port D7 is activated. If the signal is greater than 150, port D7 will be disabled.

APPLICATION - This program could describe an analog light sensor (LDR), which reads less than 150 when an environment is dark. This is inserted into port A0, and when reading a value lower than 150 (that is, detecting a dark environment), an LED connected to port D7 lights up.

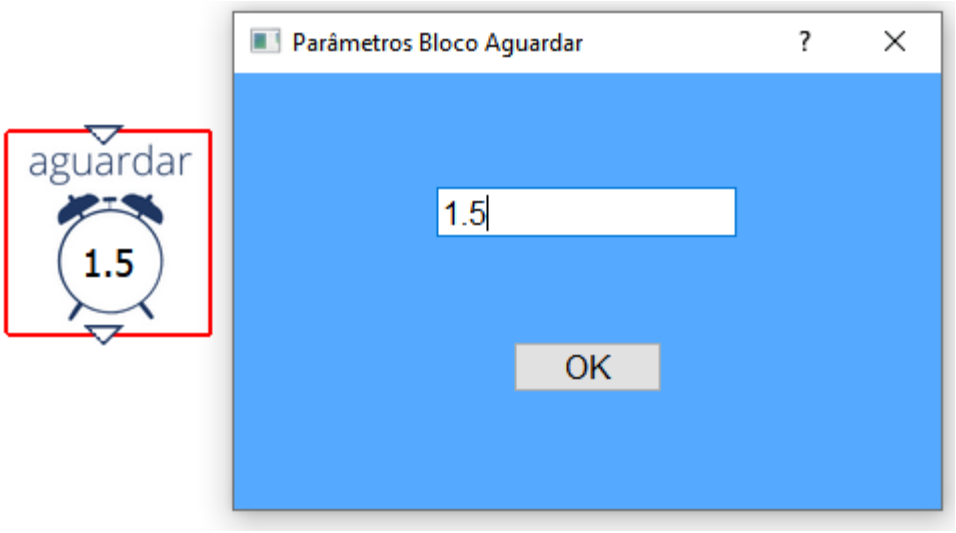
WAIT BLOCK - This block makes the previous event (block above) keep happening for the defined time. Double-click the tile and enter the time in seconds.

Ex 1.



In this example, the Motor A triggered forward event lasts for 3 seconds.

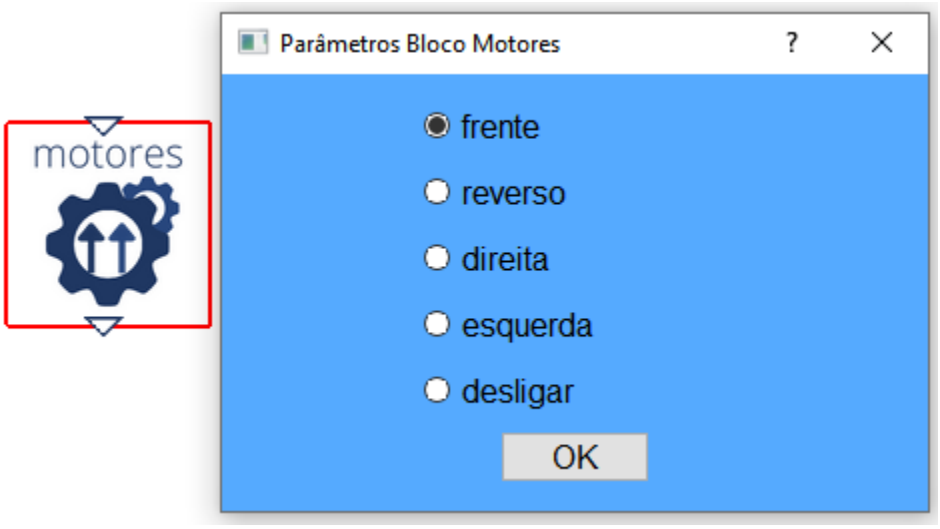
Ex.2



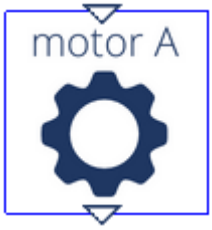
In this example, the Motor A triggered forward event lasts for 1.5 seconds.

MOTOR BLOCK - This block controls two motors connected to the MINI MOTOR DRIVER at the same time. Double click after placing the block and choose what you want the motors to do.

Ex.

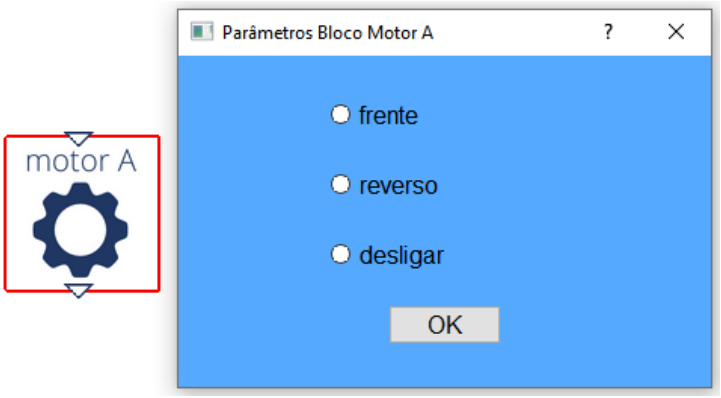


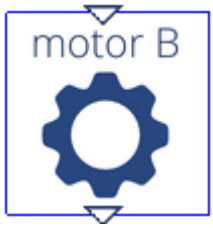
Right and Left options are specific for using two motors in robot/car projects. The command causes the motors to rotate in opposite directions, and cause a turn.



MOTOR BLOCK A - This block controls a motor connected to ports D1 and D3.

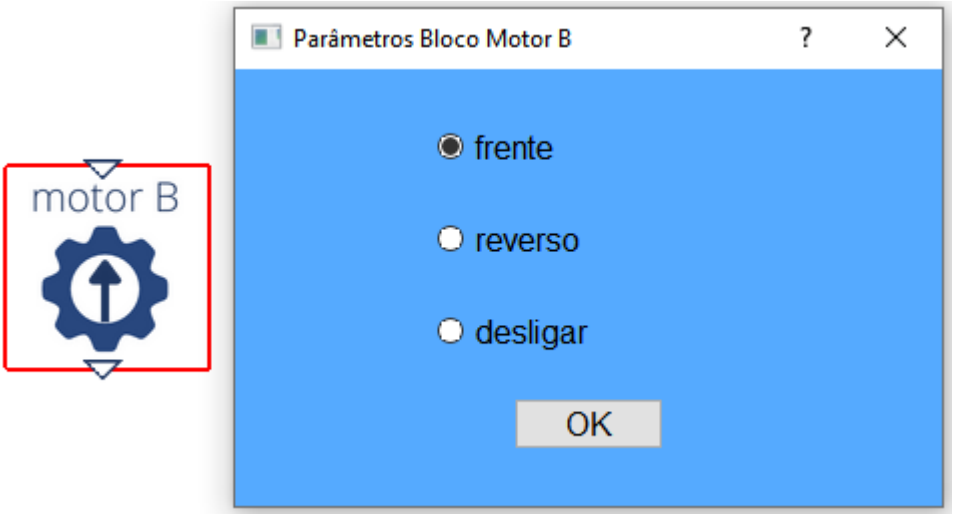
Ex.



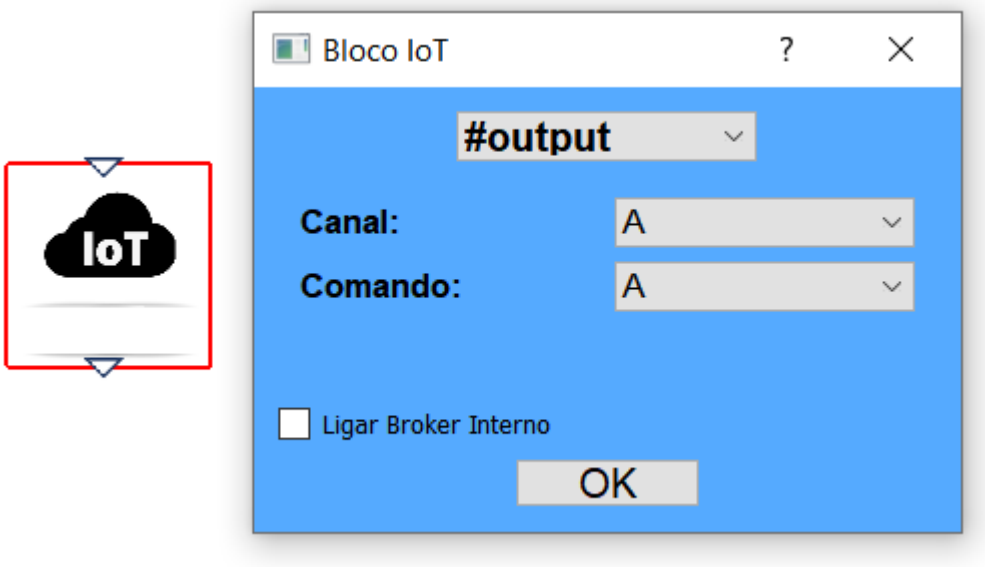


MOTOR BLOCK B - This block controls a motor connected to ports D2 and D4.

Ex.

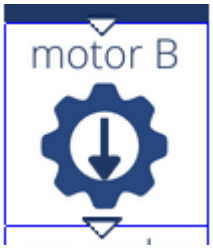


IOT BLOCK - This block serves to send the triggers to the cloud.



1.1.6. TEXT CONVERSION TOOL

When creating a flowchart, an equivalent text code is automatically created. This happens because each block represents a sequence of words, which can be seen within the blocks themselves.



In the example above, where the motor block B was chosen with the Reverse option, the equivalent text code created is #reverse motor B.

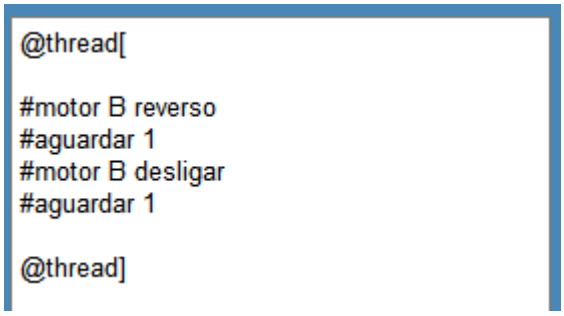
To view the complete flowchart in text format, click Compile in the lower right menu.

The equivalent text will appear in the application's right main window.

Example

Flowchart created:

Equivalent text code:



This tool is important since it is also possible to create a program just by writing command lines which are, in practice, the same thing as positioning blocks. If you want to program in this way, the possibilities of text commands (which are the same as the blocks themselves) that can be used are:

- **#exit**port_to_be_activated- This command powers up the specified port. This port has to be an OUTPUT port.

  Examples:

  - # exit D14 call
  - #output D12 turn on
  - #output D13 turn off

- **#motor**A/B**forward / reverse / off****or #engines****forward / off / left / right / reverse** - These commands turn on and off motors connected to specific ports for motors. The #motors command controls both motors at the same time.

  Examples:

- # engine forward
- # engine B front
- #engines off

➢ **#wait**seconds BR This command causes the program to wait the specified amount of seconds before descending to the next line of code.

Examples:

- # engine forward
- #wait 3
- #engine shutting down
- #wait 1.25

➢ **if**digital/analog_read= value**or else**digital/analog_read= value - This command calls a necessary condition for the code to move to the next line. The condition can be a sensor reading, the state of a port (on or off), or the value of a user-created variable (#var).

Examples:

- if D27 = on
- if D34>300
- if Variable_1 >10
- if Ultrasonic > 40
- otherwise D27 = on
- end if

➢ **#print out**text_to_be_shown_on_display**BR**This command displays texts on the connected OLED display.

Examples:

- #print Good morning!!
- #print The battery is running low!

➢ **#printserial**text_to_be_sent_to_the_system- This command is for WI-FI and bluetooth communication. It is necessary to connect a Mini Broker or Bluetooth module to use it.

Examples:

- #printserial Wireless message!!!
- #printserial #output D27 call

➢ **#var**variable_number = value- This command is intended to enable more complex combinations of events. The user can, using the if command, create a program start condition based on data from a specific variable created by the user.

Examples:

- #var Variable_2 = 45
- #var Variable_3 = (Variable_1 +Variable_2)*2

➢ **#repeat**number_of_repeats**and #endrepeat** - These commands are intended to make it possible to repeat the reading of the lines between them.

Examples:

- #repeat 2
- # engine starting
- #wait 3
- #engine shutting down
- #end repeat

1.1.7. CONNECTION FACILITATING SHIELDS

To simplify the connection of sensors and actuators with each microcontroller, it is possible to use specific shields.

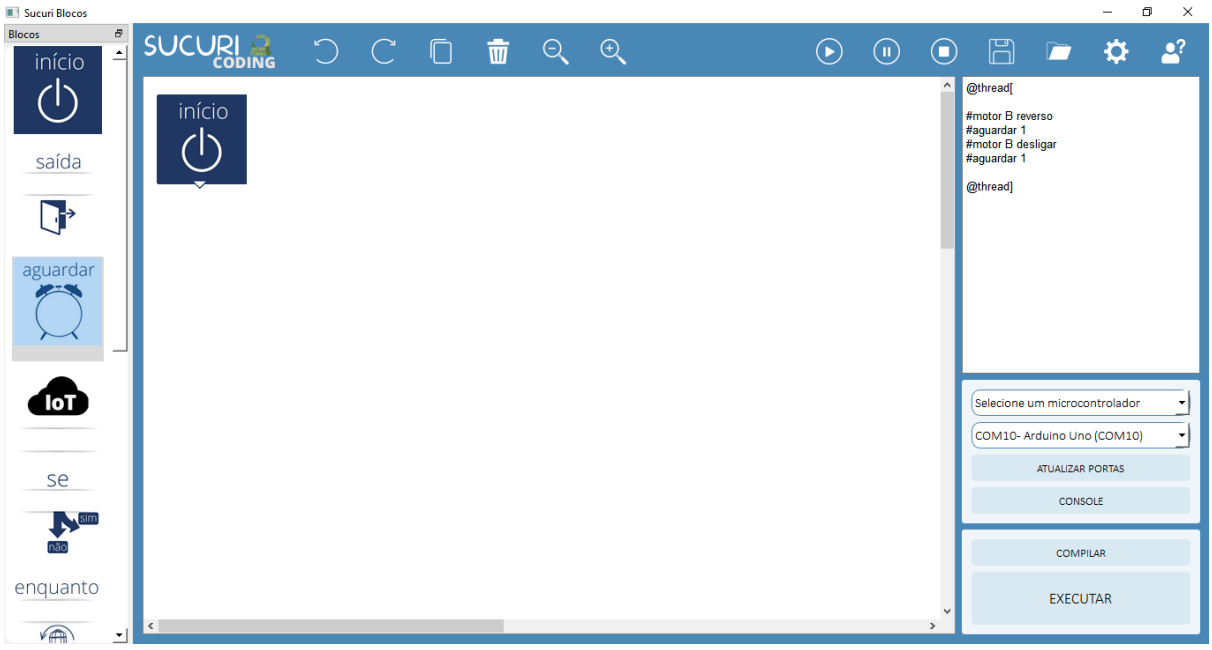
Engine Shield ESP 8266 12

Example 1 of using the Motor Shield

Example 2 of using the Motor Shield

1.1.8. USE OF THE APPLICATION

On the main screen on the left, you can find the main flowchart programming environment.



The block flowchart programming logic is based on the construction of a sequence of events based on sensors and actuators connected to the microcontroller. These events are represented by blocks, which must be placed in the programming environment and connected from lines according to the specific sequence of events that the user wants. The program will run from top to bottom, starting from the Home block.

Example:

In this code created, port D7 will be activated and, after 2 seconds, it will be turned off.

The code will run as soon as the microcontroller with the code already downloaded and disconnected from the computer is connected to a power source. To download the program to the microcontroller, click compile and then Run.

1.1.9. DOOR OPERATION LOGIC

The program works by reading and/or energizing ports (“ports” are the pins on the side of the board) specific to the microcontroller.

These ports can be of two types:

- **PROHIBITED**- An input port has the technical name of INPUT. This type of port serves to input information into the microcontroller. This occurs from the connection of sensors, which are nothing more than devices that gather information from the environment and communicate to the microcontroller. This type of door is used for projects where it is necessary to monitor the work environment, communicating the user when something happens or automatically performing actions based on previous procedures defined by the user.

Example of devices to be connected to this type of port:

- ➢ Light Sensor (LDR)
- ➢ Infrared Sensor
- ➢ Touch Sensor (button)
- ➢ Ultrasonic Distance Sensor

- **EXIT**- An output port has the technical name of OUTPUT. This type of port is used for signal output, that is, for energy output to drive actuators. These devices perform actions, such as turning on a light, turning an engine, or making a sound. As their name describes, they do something to act.

Example of devices to be connected to this type of port:

- ➢ led
- ➢ Electric Motor
- ➢ Buzzer (beep sound)
- ➢ Relay (used to switch on devices with a higher electrical load, such as a light bulb)

There is another important feature about all ports: these can be digital or analog:

DIGITAL PORT -A digital port is a port that sends or understands only two types of information: 0 or 1. 1 corresponds to the on state of the port, while 0 corresponds to the off state. A digital actuator can only be on (1) or off (0), while a digital sensor only communicates that it is reading something (1), or not reading anything (0).

Example of digital devices to be connected to this type of port:

- ➢ Touch Sensor (button)
- ➢ LED actuator

ANALOG PORT -Unlike a digital port, this type of port sends or understands a large number of numbers: from 0 to 1023. It is used with analog sensors for more precise monitoring of some information (luminosity being at 250, for example), or for more specific port power-up (like turning on an LED with only 70% of its capacity, for example). While all actuators can be used on analog OUTPUTs (since any electronic device can be driven with partial power), there are specific actuators designed to be controlled more precisely, like a servo motor.

Example of analog devices to be connected to this type of port:

- ➢ Light Sensor (LDR)

When creating the flowchart, the user must specify the sensor or actuator that he wants to control from the port where it is inserted. However, it is necessary to pay attention that there are pre-defined ports for each type of device. This does not occur in classic programming with written codes, but it is a convention adopted by Sucuri Coding to facilitate the user experience. The port map can be seen below:

The port categories are:

APPETIZER

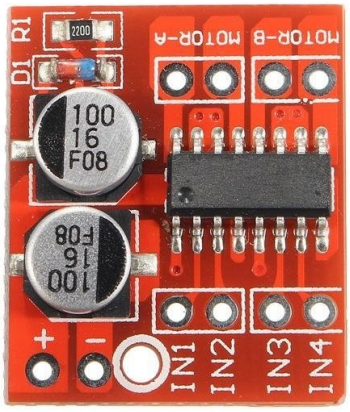
DIGITAL INPUT- There are two DIGITAL INPUT ports (D36 and D39), and these must be used for signal input from digital sensors. If an analog sensor is inserted into this type of port, it will not work.

ANALOG INPUT -There are two ANALOG INPUT inputs (D34 and D35), and these must be used for signal input from an analog sensor. If a digital sensor is inserted in this type of port, it will only be able to report 0 or 1023.

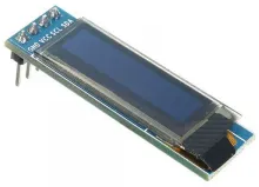
ULTRASONIC SPECIAL CASE -Port mapping includes a special case for the ultrasonic analog sensor. Since this sensor needs 4 ports to work, ports D32 and D35 are prepared to be used with it. See proper sensor connection in the example in the next section.

OUTPUTS

OUTPUT -There are four OUTPUT ports (D12, D13, D14 and D27), and these must be used for signal output to actuators. For motors, there are specific ports marked MINI MOTOR DRIVER.



MINI MOTOR DRIVER (ENGINES)- The MINI MOTOR DRIVER ports correspond to four signal outputs made specifically for motors connected to a MINI MOTOR DRIVER (D4, D5, D15 and D18). The specific connections between each microcontroller port, Mini Driver port and the motors themselves can be seen in the next section.



SPECIAL CASE OLED DISPLAY- Port mapping includes a special case for the DISPLAY OLED actuator. Since this actuator needs 4 ports to function, ports D22 and D21 are prepared to be used with it. See proper connection to the actuator in the example in the next section.

1.1.10. PHYSICALLY CONNECTING A SENSOR OR ACTUATOR

When connecting your device to the microcontroller, there is another detail to pay attention to in addition to the main port: for actuators, it is necessary to connect the other pole of the device to the GND port. For some sensors, which have 3 wires, it is necessary to connect the VCC and GND port in addition to the signal port. Below are some examples of physical connections.

Example 1

OPERATION

In this created code, the warning “Someone is in front of the sensor!” appears on the OLED Display when the ultrasonic sensor senses something less than 30 cm from its reader. Also, a serial message with a*trigger*is sent via Bluetooth. A cell phone connected with the MYIOT app will receive the notification “Someone is in front of the sensor!”

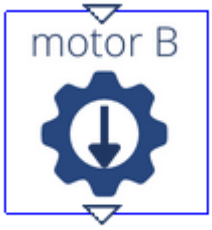
Example 2

OPERATION

In this code the warning “Someone is in front of the sensor!” appears on the OLED Display when the ultrasonic sensor senses something less than 30 cm from its reader. Also, a serial message with a*trigger*is sent to cloud. Any device connected with the MYIOT system will receive the notification “Someone is in front of the sensor!”. Note that this code is very similar to that of example 2. The only difference is the replacement of the bluetooth module for an ESP 01 microcontroller working as a Mini Broker. As much as the final result will be the same, it will not be necessary to connect a cell phone via bluetooth. The serial message (#print serial) will be sent directly to the cloud.

1.1.11. TEXT CONVERSION TOOL

When creating a flowchart, an equivalent text code is automatically created. This happens because each block represents a sequence of words, which can be seen within the blocks themselves.



In the example above, where the motor block B was chosen with the Reverse option, the equivalent text code created is #reverse motor B.

To view the complete flowchart in text format, click Compile in the upper right menu.

The equivalent text will appear in the application's right main window.

This tool is important since it is also possible to create a program just by writing command lines which are, in practice, the same thing as positioning blocks. If you want to program in this way, the possibilities of text commands (which are the same as the blocks themselves) that can be used are:

- **#exit**port_to_be_activated- This command powers up the specified port. This port has to be an OUTPUT port.

  Examples:

  - # exit D14 call
  - #output D12 turn on
  - #output D13 turn off

➢ **#motorA/Bon offor #engineson off** - These commands turn on and off motors connected to specific ports for motors. The #motors command controls both motors at the same time.

Examples:

- # engine starting
- # engine B turn on
- #engines off

➢ **#waitseconds**BR This command causes the program to wait the specified amount of seconds before descending to the next line of code.

Examples:

- # engine starting
- #wait 3
- #engine shutting down
- #wait 1.5

➢ **ifdigital/analog_read= valueor elsedigital/analog_read= value** - This command calls a necessary condition for the code to move to the next line. The condition can be a sensor reading, the state of a port (on or off), or the value of a user-created variable (#var).

Examples:

- if D27 = on
- if D34>300

- if Variable_1 >10
- if Ultrasonic > 40
- otherwise D27 = on
- end if

➢ **#print out**text_to_be_shown_on_display**BR**This command displays texts on the connected OLED display.

Examples:

- #print Good morning!!
- #print The battery is running low!

➢ **#printserial**text_to_be_sent_to_the_system- This command is for WI-FI and bluetooth communication. It is necessary to connect a Mini Broker or Bluetooth module to use it.

Examples:

- #printserial Wireless message!!!
- #printserial #output D27 call

➢ **#var**variable_number = value- This command is intended to enable more complex combinations of events. The user can, using the if command, create a program start condition based on data from a specific variable created by the user.

Examples:

- #var Variable_2 = 45
- #var Variable_3 = (Variable_1 +Variable_2)*2

➢ **#repeatnumber_of_repeatsand #endrepeat** - These commands are intended to make it possible to repeat the reading of the lines between them.

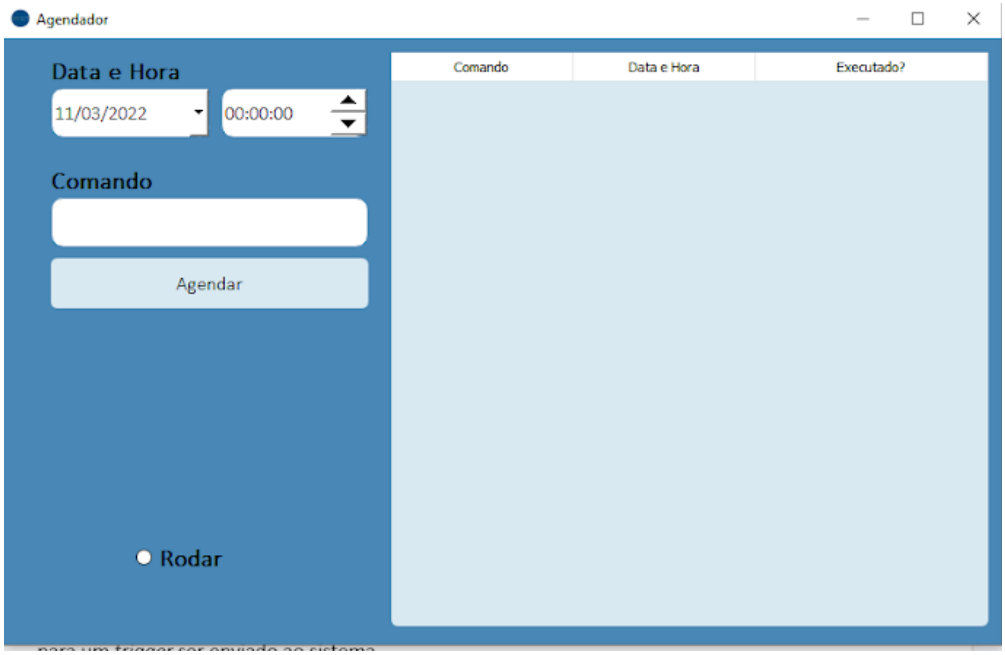
Examples:

- #repeat 2
- # engine starting
- #wait 3
- #engine shutting down
- #endrepeat

1.2. SCHEDULER

To open the application, click on Program in the MyIoT Package main menu.

The scheduler application allows the user to set a specific time and date for a trigger to be sent to the system.



In**Date and time,**choose when you want the trigger to be sent. In Command, type the trigger itself. Check or uncheck Rotate to enable and disable schedules.