



PROJECT RESULT 2

Block Programming Tool Development for Low-cost IoT Electronics: Facilino

Abstract

This document describes project result 2 of EcoThings project where we have developed block programming tools aimed to be used for low-cost IoT Electronics. In particular, in this document, we will describe activities related to the development of Facilino, a block programming tool based on Blockly that is used to generate code for low-cost electronics such as Arduino, Raspberry Pi and ESP32.

Leopoldo Armesto Ángel
Universitat Politècnica de Valencia
larmesto@idf.upv.es



Contents

1. Introduction	2
a. Background and Motivation	2
b. Highlights	2
2. Main Results	4
a. Old Facilino version versus Facilino OTA	4
b. Open source and free software	4
c. Front-end and Back-end.....	4
d. User's account.....	5
e. Project's dashboard.....	8
f. Facilino OTA Server	12
g. Block simplification and feature extension	19
h. Block filters.....	20
i. Tutorial and Project-related Exercises	21
j. Translation tool.....	26
k. Over-the-air (OTA)	27
l. Documentation	27
References	28



1. Introduction

Block programming tools are one of the most powerful tools that educators have in their hands to teach programming skills to pupils. Based on their simplicity for connecting blocks together, they can cope with complex code generation. Blockly [1] is, probably, the most widely used tool to develop block programming software. It is basically divided into two aspects, the block shape generator and the code generator. In this sense, Facilino is a software that uses Blockly to generate code for low-cost electronics such as Arduino, Raspberry Pi and ESP32.

a. Background and Motivation

Facilino [2], was originally developed by Leopoldo Armesto (UPV) in a collaboration with a local SME, Robotica Fácil [3], to provide solutions for schools so they can integrate different kinds of STEAM activities within their curricula. From the beginning, Facilino has shown interest in their community. Originally, the software was planned as a free-ware software, as part of the agreement for development. Therefore, many blocks were free to use, but others, require a license.

Taking Facilino as a basis and considering that the old version of Facilino was not maintained anymore (last published release is dated on end of 2019), we consider that EcoThings project brings a great opportunity to retake the development under a completely different approach. For obvious reasons, we have kept the original software name, trademark and logo, but now its development is fully maintained by the UPV. Indeed, this approach has shown to have an impact on the community because some people already use Facilino and thus the new version sounds familiar to them and thus showed interest in learning more about this new version of Facilino. Due to the previous collaboration with Robótica Fácil, we could reach a sector that was precisely of interest of this project. When presenting the new version we referred to EcoThings project.

b. Highlights

During the project execution, we have made a big effort in readapting many aspects:

- **Hosting:** The new version is hosted at the UPV servers, and it will be maintained by the UPV.
- **Totally free software:** All blocks can be now used for free.
- **Front-end & back-end development:** Facilino works now with a front-end based on HTML and Javascript (in the client side), while it runs PHP and MySQL in the server side.
- **User's account:** We have included a simple account log in, registration, password recovery and user's profile.
- **Project's dashboard:** Users can manage their own projects, once logged in.



- **Facilino OTA Server:** This tool allows Facilino to compile code and upload it to the electronics. The main advantage, compare to the previous version is that now, all dependencies of the code generated by Facilino are now integrated in a single tool.
- **Block simplification and feature extension:** Some original blocks of Facilino have been simplified and in some cases. Also, we have developed new blocks that have considered relevant to boost code development those related with EcoThings project. Blockly includes the possibility to generate shadow blocks, which is a great utility to generate default inputs for a specific block instruction, so that users can quickly find out how to use a instruction without the need of reading documentation about it.
- **Block filters:** The set of block instructions of Facilino has become large, which might be confusing for non-experienced users. For that reason, we have included block filters that will show blocks that can be typically used for a specific project type. In particular, EcoThings project has its own block filter, which means that most of the blocks that one can expect to use within EcoThings proposals, will be shown by default and the rest will be hidden.
- **Tutorial and Project-related exercises:** Based on previous Tutorial exercises, the new version of Facilino includes a set of tutorials to start coding. In the majority of these tutorials and exercises, we propose connection diagrams and alternative approaches that can be used too. We provide a problem description, hints, Facilino code and a ThinkerCAD project embedded in the tutorial so that the proposed exercise can be executed in a simulated environment, without need of physical electronics. Similarly, we have just started documenting some specific exercises that are related with some projects such a LED race or low-cost robotics platforms. We have also started documenting the kind of exercises that can be used within the EcoThings project.
- **Translation tool:** The new version of Facilino includes an easy-to-use translation tool that aims to translate Facilino into multiple languages. So far, we have translated Facilino into Spanish and other languages such as Catalanian, German, Italian, French and Portuguese have been Google translated and are currently under review.
- **Over-the-Air (OTA):** The new version of Facilino, includes a feature that allows to program micro controller such as ESP8266 or ESP32 using an over-the-air feature. In combination of Facilino OTA Server, a user can compile and upload code to a specific device via WiFi, without need of physical USB cable. This represents a great advantage for schools using tables or iPADS in their lab sessions.
- **Documentation:** This is a working progress feature, and it will be implemented in a mid-term period (by the end of the project, if possible). The idea is to show specific examples of how to use instructions. In PR4, we will particularly generate graphic documentation on how to code with Facilino within EcoThings project.



2. Main Results

a. Old Facilino version versus Facilino OTA

Facilino is hosted at:

<https://facilino.webs.upv.es/>

This host will be maintained by the UPV even when the project is finished, with the purpose of providing access to anyone who wants to use this tool.

b. Open source and free software

Facilino, also known as Facilino OTA, has been published as an open source software under Apache 2.0 license. The new version of facilino is totally free, which means that we have reimplemented the underlying code for those blocks so that now all of them generated code at no cost.

The code has been published on GitHub (using the original account, but on a new repository):

https://github.com/roboticafacil/facilino_ota

Together with Facilino OTA, users must install Facilino OTA Server, another open source tool that is hosted at:

https://github.com/roboticafacil/facilino_ota_server

We have created a first release, with binary files for Windows and Ubuntu (also works in LliureX).

c. Front-end and Back-end

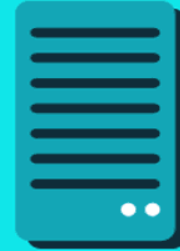
Facilino uses a frontend/backend architecture, which means that when the user loads Facilino web page, this page is served by a server running PHP. The server generated HTML and javascript code that the client (frontend) renders into a web page. The server make queries to a database using MySQL.

FRONTEND



HTML & Javascript

BACKEND



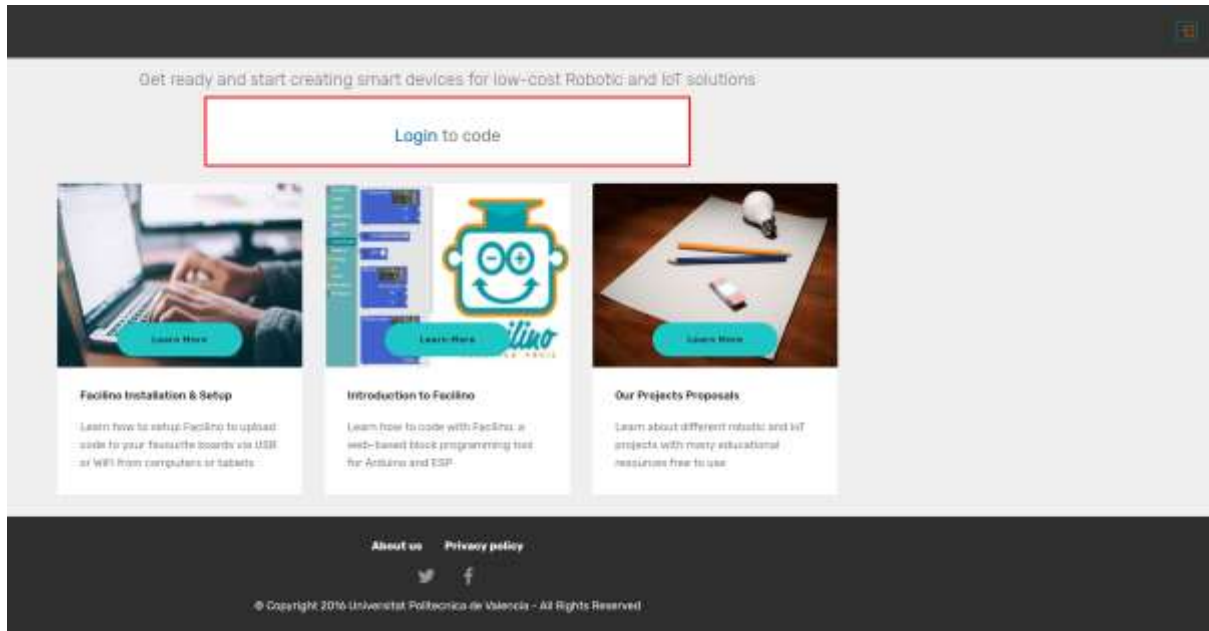
PHP & MySQL

In addition to this, it is interesting to understand how Facilino generates code to be uploaded on electronics.

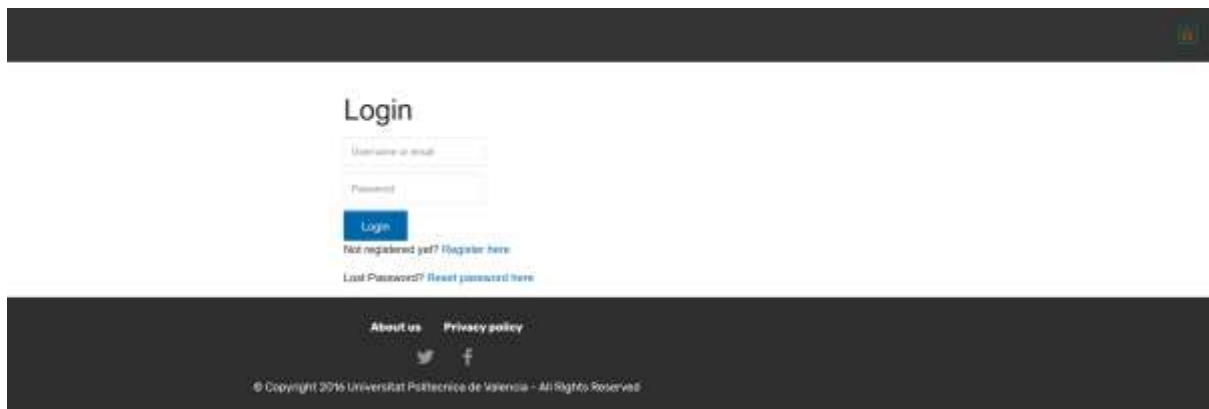


d. User's account

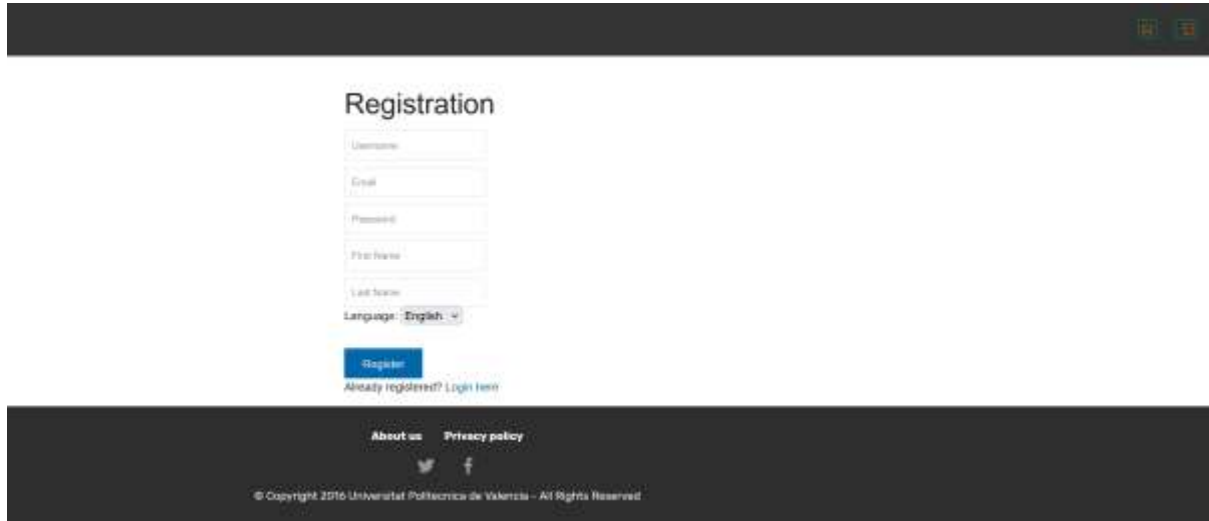
In order to use Facilino, users must create an account. When not logged in, a red box informs the user that needs to login in order to code. Login can be done by clicking on the top right icon at the menu bar.



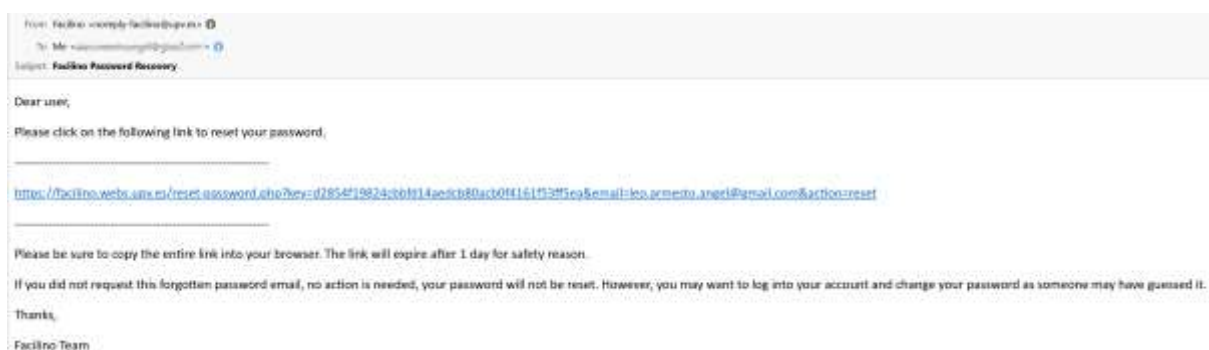
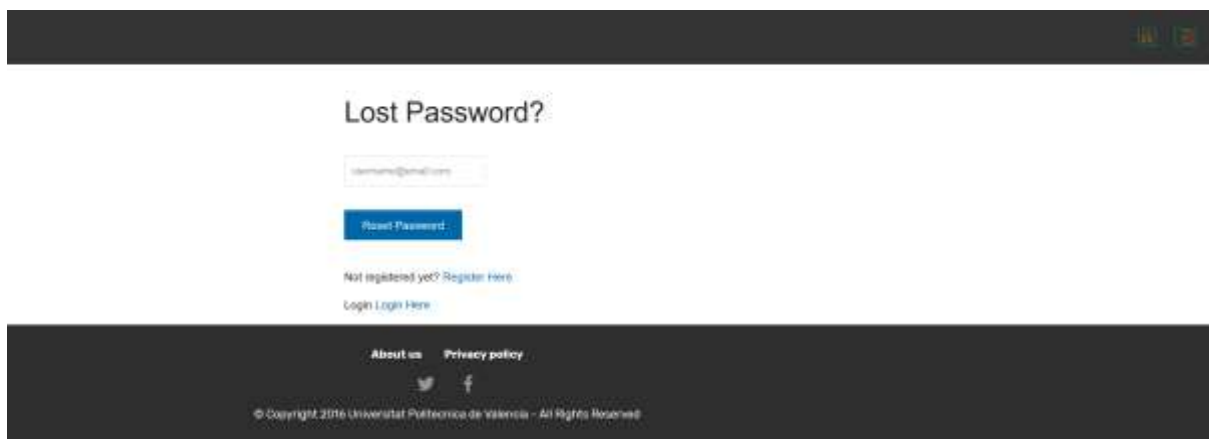
If a user has an account, it can simply log in to start coding, otherwise, the user will need to create an account to as indicated in the login page.



Registration page, ask for few personal data and language preference (currently only english and spanish):

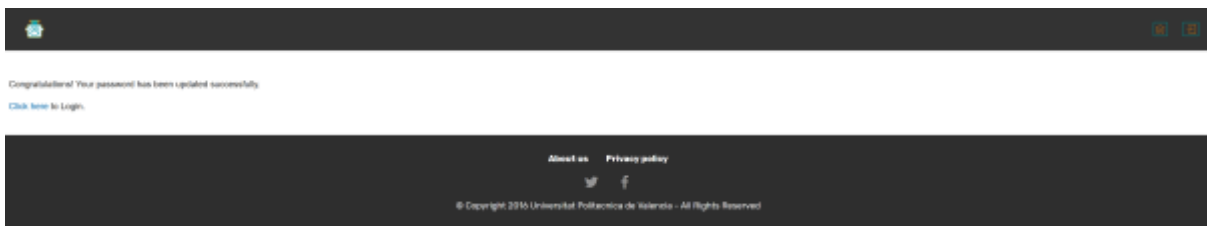


Users can recover their password if they do not remember and they will receive an email with a link to reset their password:





Once clicking on the link, they just need to set a new password and the user's account password will be updated.

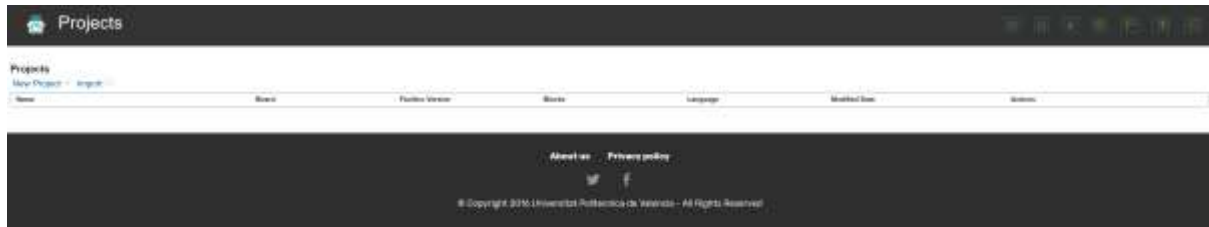


Once logged in, they can access to their user profile to update their name or language preferences. User's can also apply for an account upgrade in which they can participate in the Translation program or Academy program. These two programs are described below so that users can contribute to the development of Facilino.



e. Project's dashboard

User's can create projects that will be included in their project's dashboard. So far, this is a simple dashboard where all users' projects are listed and they can duplicate a project, delete, edit, download code, etc... We are considering to include a search/filtering tool so that when users start have a big amount of projects, can be easily found.













To create a project, users must click on “New Project” link and a form with all possible options will be shown.

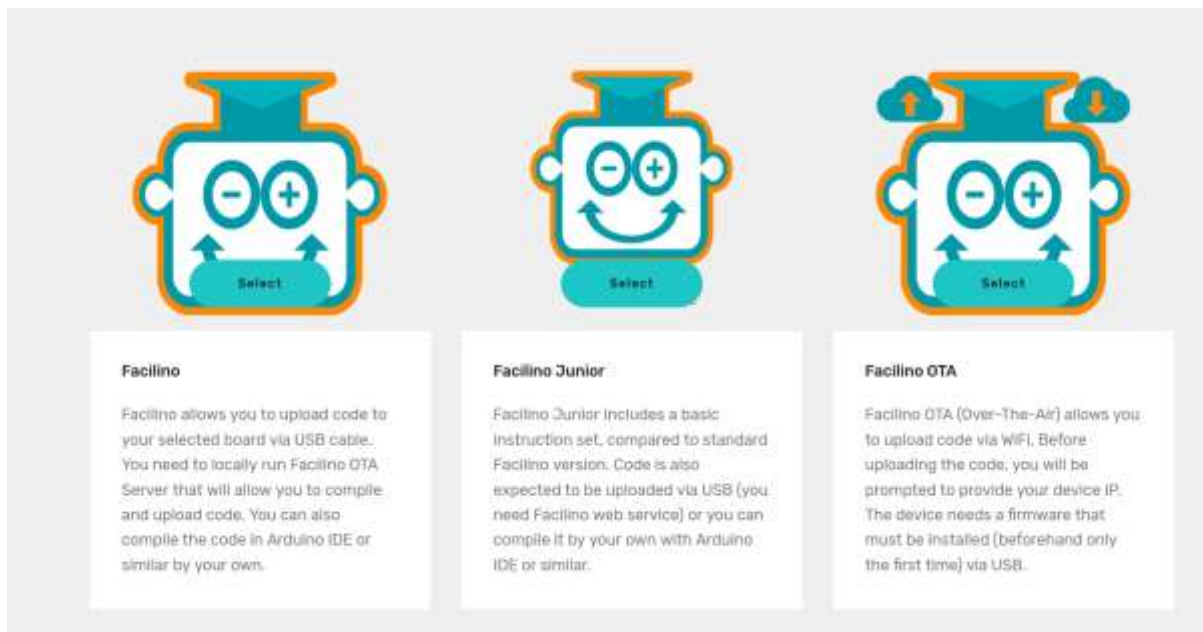


Users must provide a project’s name, select the board they plan to use, Facilino’s version, the block instruction set as well as the language. All these options can be modified later once the project is created. They can also select the Server IP and the Device IP for Facilino OTA settings.

Facilino generates Arduino code for multiple type of boards. It works with several Arduino processors such as Arduino Uno; Espressif boards such as ESP8266 and ESP32 and Raspberry Pi. Facilino blocks instructions are identical in most cases, while the generate code adapted to the specific board, which greatly abstracts many issues related to hardware abstraction compared to classic text coding. So far, we have included a subset of boards, but this can be extended in the future to include new ones:

 <p>Select</p>	 <p>Select</p>	 <p>Select</p>
<p>Arduino Nano Arduino Nano (Atmega328p)</p>	<p>Arduino Uno Arduino Uno board (Atmega328p)</p>	<p>Arduino Mega 2560 Arduino Mega (AtMega2560)</p>
 <p>Select</p>	 <p>Select</p>	 <p>Select</p>
<p>ESP01 ESP-01 (ESP8266)</p>	<p>NodeMCU v3.0 NodeMCU v3.0 (ESP8266)</p>	<p>WeMos D1R2 Wemos D1R2 (ESP8266)</p>
 <p>Select</p>	 <p>Select</p>	 <p>Select</p>
<p>ESP32 Generic ESP32 board</p>	<p>Wemos D1R32 (with Arduino Shield) Wemos D1R32 with Arduino Shield (ESP32)</p>	<p>ESP32 - DevKitC ESP32- DevKitC (ESP32)</p>
 <p>Select</p>		
<p>TTGO Watch 2020 TTGO Smart watch</p>		

While Facilino is a code generation tool, the actual code will be uploaded to using Facilino OTA Server (see next section). This will will upload code either through USB, selecting **Facilino** version, or over the air selecting **Facilino OTA**. We have also created a simplified version of Facilino (using USB), named as **Facilino Junior** where most of the blocks have been adapted to a simpler way of use, while limiting obviously the full capacity of the tool if advanced instructions were used instead.

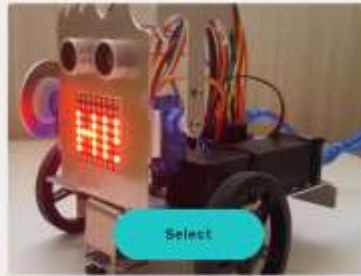


Finally, the user must select a Block Instruction set. The purpose of this selection is to filter some blocks that the user will not use depending on the type of project he/she intends to do. In that sense, the option by default is a generic project with the full set of instructions, while other projects such as robotic projects or EcoThings project will require a different kind of instructions. Thus, by selecting a specific project, some instructions will not be shown by default and therefore, it's easier for non-experienced users to find the instructions they need to user for their actual needs.



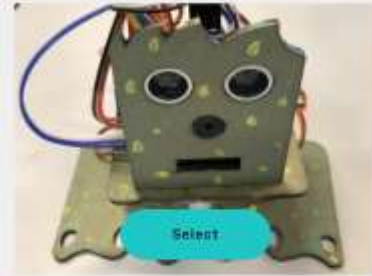
Generic Project

Complete instruction set with for all type of projects; Make your dreams become true with the full instruction set.



DYOR Robot

Block instruction set for low-cost Arduino-based wheeled robots to generate movements, sounds and expressions.



bPED Robot

Block instruction set for hipless walking robots like bPED to generate movements, sounds and expressions.



meArm Robot

Block instruction set for a robot arm with three servos and a gripper. It includes instructions to controlling the robot in several modes.



Multisensor Board

Block instruction set for using the multisensor shield. It includes instructions to learn how to read sensors or activate outputs with the multisensor board.



DIY Home Automation Kit

Block instruction set for environmental sensors used in home automation, intelligent greenhouse, weather station projects.



LED Race

Block instruction set for a LED race project. It includes the instructions focused on LED strips and push-buttons.

f. Facilino OTA Server

You can download here Facilino OTA Server. Select the appropriate version for your OS.

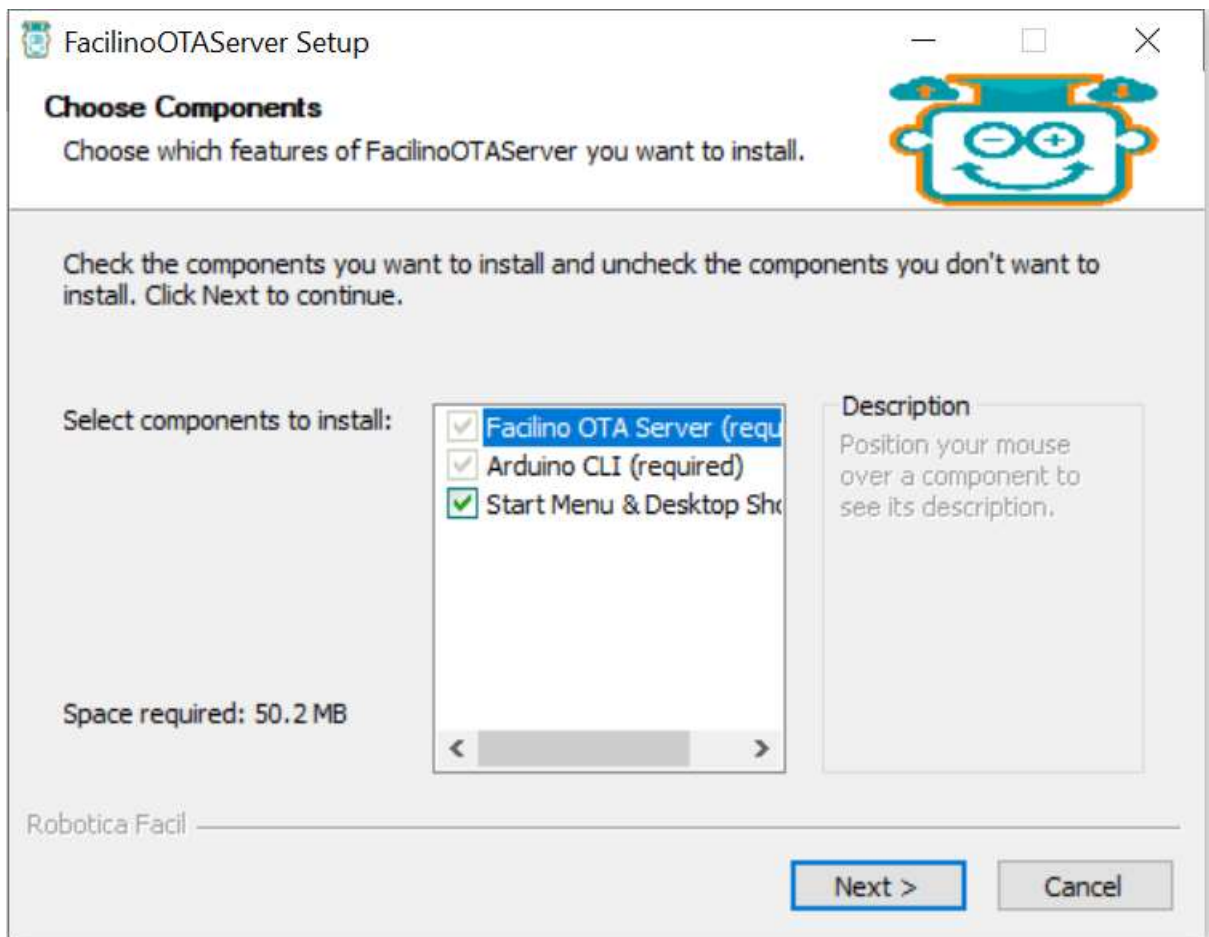
1. Download Files

First, download Facilino OTA Server. Be advised that these downloads have been taken from Robotica Facil's GitHub, where you can find the source code of the application and old releases.

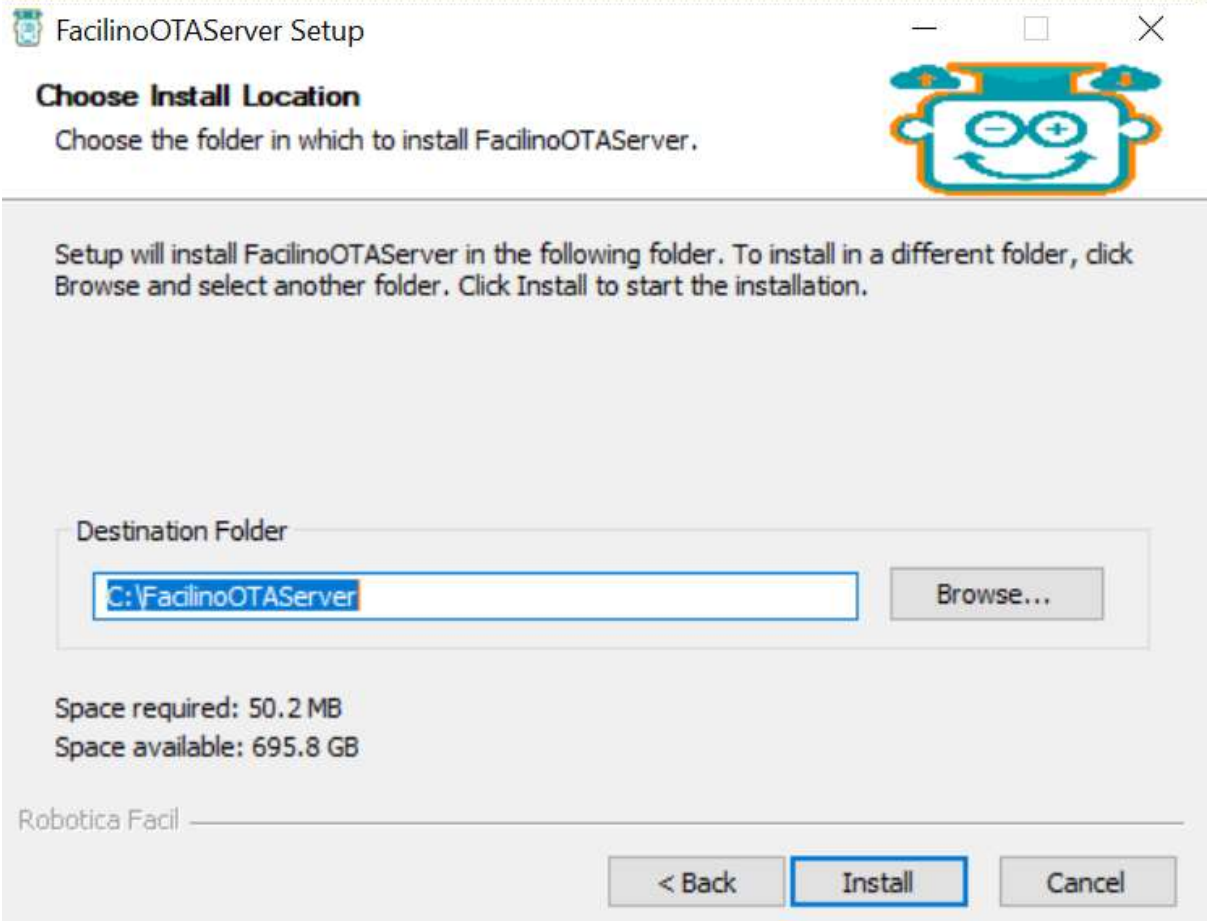
2. Facilino OTA Server Windows Installer

Click on the installer to execute it. Your computer might give a warning as the Installation comes from an unknown source. Simply, give permission under the extended tab option. The Facilino OTA Server is completely safe and does not pose a threat to your computer. Indeed, Facilino OTA Server actually is a port to the Facilino web page and Arduino-CLI. The web page generates code based on blocks, while Arduino CLI is the software that compiles your code so it is highly recommended to check that has been properly installed after the installation process finishes, including all boards and libraries required by Facilino, otherwise, compiling and uploading code might fail.

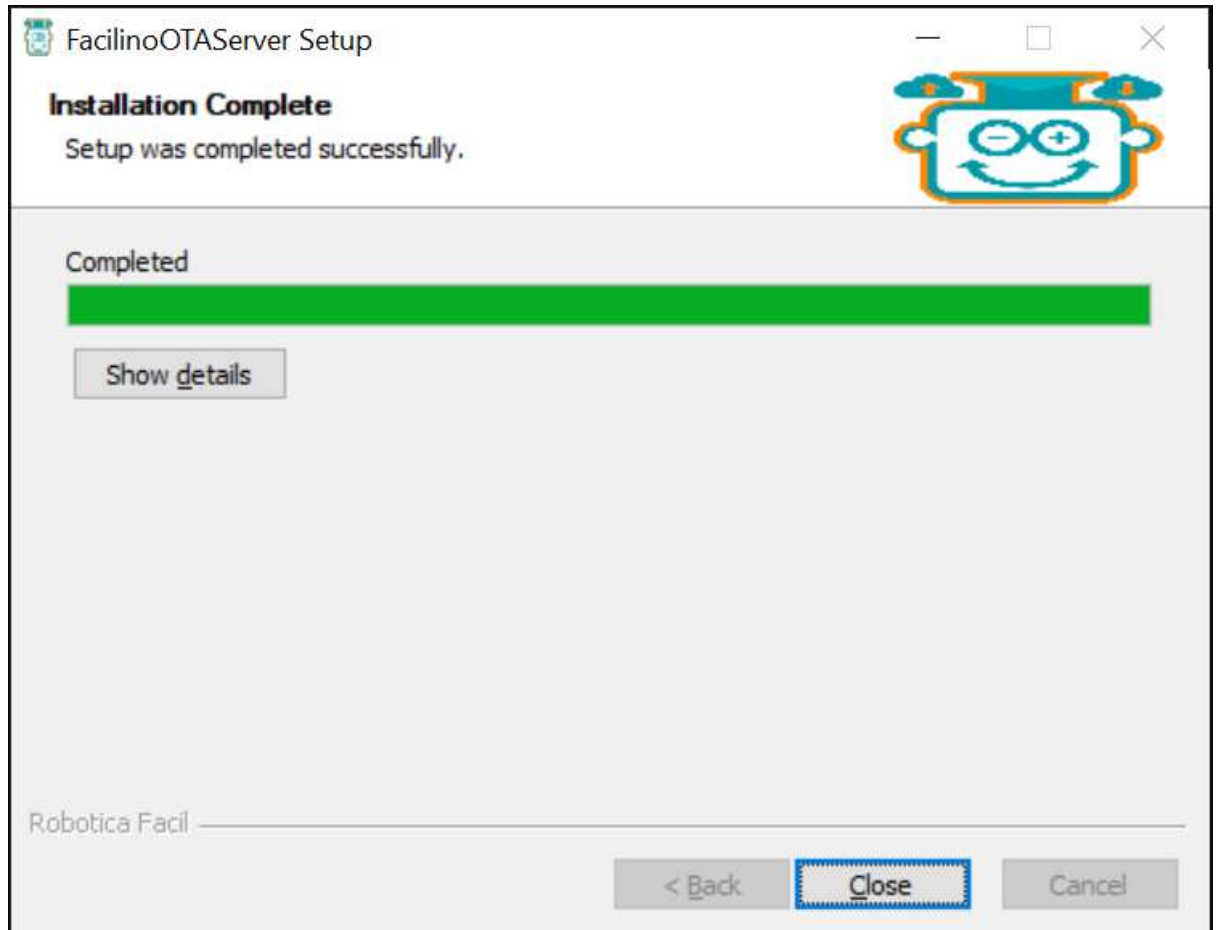
When executing Facilino OTA Server Installer, you should see the following window:



Select the installation location, this is automatically set to your folder `C:\FacilinoOTAServer`. We recommend you not to change this path.



Then installation should start immediately and a progress bar should be visible. This installation might take a few minutes, particularly when installing Arduino libraries, because they will be first downloaded and then installed.



3. Facilino OTA Server Windows ZIP (for non-admin users)

Unzip Facilino OTA Server and `cd` to the unzipped folder and type `config.bat` in the command shell. You should see on the shell output all `arduino-cli` commands required by Facilino to be able to compile and upload code to the supported boards and libraries.

4. Facilino OTA Server on Ubuntu

Untar Facilino OTA Server and `cd` to the uncompressed folder and bash `config.sh` in the console. You should see on the shell output all `arduino-cli` commands required by Facilino to be able to compile and upload code to the supported boards and libraries.

5. Arduino CLI

Open the command shell and move to Facilino OTA Server directory (i.e.: `C:\FacilinoOTAServer`). Then, `cd` to `arduino-cli` folder and type `arduino-cli.exe core list` (on Windows) or `./arduino-cli core list` (on Ubuntu). You should see a list of supported boards:


```
C:\FacilinoOTAServer\arduino-cli>arduino-cli.exe core list
ID                Installed Latest Name
arduino:avr       1.8.6      1.8.6  Arduino AVR Boards
arduino:mbed_rp2040 3.5.4      3.5.4  Arduino Mbed OS RP2040 Boards
arduino:megaavr   1.8.8      1.8.8  Arduino megaAVR Boards
esp32:esp32       1.0.4      1.0.4  esp32
esp8266:esp8266   3.1.1      3.1.1  esp8266
```

If you type: *arduino-cli.exe lib list* you should see a list of installed libraries:

```
C:\FacilinoOTAServer\arduino-cli>arduino-cli.exe lib list
Name                Installed Available Location Description
Adafruit_BMP085_Library 1.2.2 - LIBRARY_LOCATION_USER -
Adafruit_BusIO        1.14.1 - LIBRARY_LOCATION_USER -
Adafruit_GFX_Library  1.11.5 - LIBRARY_LOCATION_USER -
Adafruit_NeoPixel     1.11.0 - LIBRARY_LOCATION_USER -
Adafruit_SSD1306     2.5.7 - LIBRARY_LOCATION_USER -
Adafruit_Unified_Sensor 1.1.7 - LIBRARY_LOCATION_USER -
ArduinoHttpClient     0.4.0 - LIBRARY_LOCATION_USER -
ArduinoJson           6.20.0 - LIBRARY_LOCATION_USER -
ArduinoSTL            1.3.3 - LIBRARY_LOCATION_USER -
AsyncTCP              1.1.1 1.1.4 LIBRARY_LOCATION_USER Async TCP Library for ESP32
DallasTemperature     3.9.0 - LIBRARY_LOCATION_USER -
DFRobotDFPlayerMini  1.0.5 - LIBRARY_LOCATION_USER -
DHT_sensor_library    1.4.4 - LIBRARY_LOCATION_USER -
DHT_sensor_library_for_ESPx 1.18 - LIBRARY_LOCATION_USER -
Dimmable_Light_for_Arduino 1.5.0 - LIBRARY_LOCATION_USER -
Dynamixel2Arduino    0.6.1 0.6.2 LIBRARY_LOCATION_USER DYNAMIXEL protocol Library for Arduino
ESP32_Servo           1.0 - LIBRARY_LOCATION_USER -
ESP8266               1.0.0 - LIBRARY_LOCATION_USER -
ESPAsyncTCP           1.2.2 1.2.4 LIBRARY_LOCATION_USER Async TCP Library for ESP8266 and ESP318
ESPAsyncWebServer     1.2.3 - LIBRARY_LOCATION_USER -
ESPUI                 2.2.1 - LIBRARY_LOCATION_USER -
FauxmoESP             3.4 - LIBRARY_LOCATION_USER -
IRremote              4.0.0 - LIBRARY_LOCATION_USER -
LiquidCrystal         1.0.7 - LIBRARY_LOCATION_USER -
LiquidCrystal_I2C     1.1.2 - LIBRARY_LOCATION_USER -
LittleFS_esp32        1.0.6 - LIBRARY_LOCATION_USER -
OneWire               2.3.7 - LIBRARY_LOCATION_USER -
PubSubClient          2.8 - LIBRARY_LOCATION_USER -
ThingsBoard           0.9.0 - LIBRARY_LOCATION_USER -
```

If any of the previous steps fails or during compilation of a program there's a missing library, you can install them manually (see *arduino-cli*'s help by typing *arduino-cli.exe -h* on Windows or *./arduino-cli -h* on Ubuntu).

6. Running Facilino OTA Server

You can run Facilino OTA Server from the command shell by typing *FacilinoOTAServer.exe -e* on Windows or *./FacilinoOTAServer -e* on Ubuntu (you need to *cd* to Facilino installation folder).

```
C:\FacilinoOTAServer>FacilinoOTAServer.exe -e
Using config file C:/FacilinoOTAServer/etc/FacilinoOTAServer.ini
ServiceHelper: Starting service
HttpListener: Listening on port 4000
Arduino-cli path: C:/FacilinoOTAServer/arduino-cli/arduino-cli.exe
Logging to C:/FacilinoOTAServer/logs/FacilinoOTAServer.log
```

Every time your computer restarts or you close the command shell, you will need to manually execute Facilino OTA server so it can compile or upload code. This is OK if you are the only user using the computer, however, if more users are about to use Facilino (i.e. in a Lab room), in order to avoid copies of the same libraries for each user, the recommended method is to install a service that runs automatically at boot up.

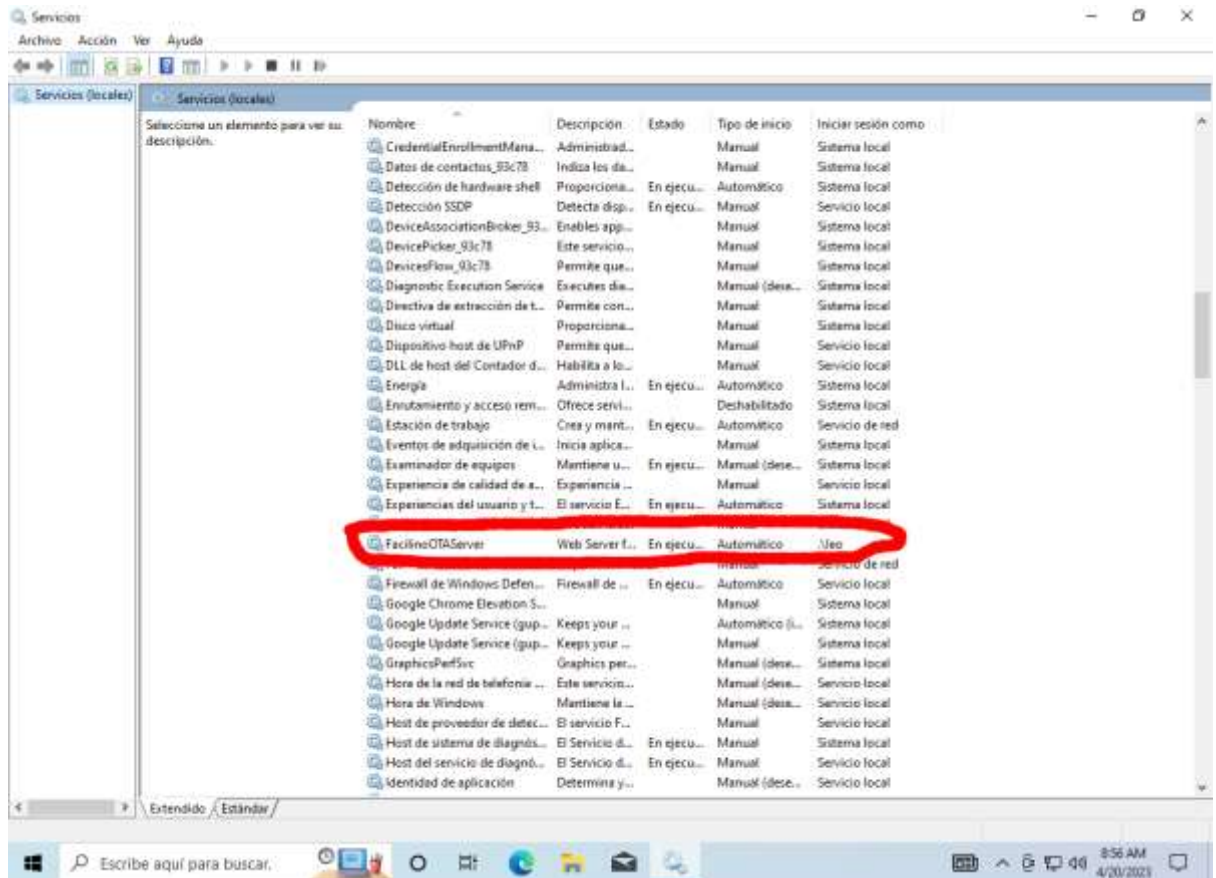
IMPORANT: In order to run Facilino OTA Server as a service, it must be installed by the administrator of the machine and the service must be installed also by the same user. In that case, open a command shell with administrative permission and *cd* to Facilino OTA Server and type *FacilinoOTAServer.exe -i <user> <password>* on Windows or *sudo ./FacilinoOTAServer -i <user> <password>* on Ubuntu, where *<user>* and *<password>* denotes the system user and password of the computer with administrative permissions.

```
C:\FacilinoOTAServer>FacilinoOTAServer.exe -i  
The service FacilinoOTAServer has been installed under: C:\FacilinoOTAServer\FacilinoOTAServer.exe
```

Now, start the service simply by executing the command *FacilinoOTAServer.exe* on Windows or *./FacilinoOTAServer* on Ubuntu. To stop the service, you can execute the command *FacilinoOTAServer.exe -t* on Windows or *./FacilinoOTAServer -t* on Ubuntu.

```
C:\FacilinoOTAServer>FacilinoOTAServer.exe
C:\FacilinoOTAServer>FacilinoOTAServer.exe -t
```

It is recommended to check the status of the service through the Windows service utility.



In Ubuntu, you can check if the service is running by executing the command `lsuf -i:4000`.

```
COMMAND  PID USER  FD  TYPE DEVICE SIZE/OFF NODE NAME
Facilino0 37024 leo    5u  IPv6 141622      0t0  TCP *:4000 (LISTEN)
```

7. Compile & Upload

Here, we assume that you have registered an account on Facilino web page, if not, please go to Register and complete the registration and log in.

Here, we assume that you have registered an account on Facilino web page, if not, please go to Register and complete the registration and log in.

To check if Facilino OTA can compile and upload code, create a blank project. Goto to the dashboard page Dashboard, click on New Project and select the desired processor among the list of available ones; select Facilino as *Facilino Version* and Generic Project *Block Instruction Set* and press *Create*.

Then, connect your board to the USB port and click on *Compile & Upload*. A window will show up with the generated code (so far since the project is empty, only setup and loop functions should appear). Select the board port in the dropdown list (if necessary press the *Refresh*) and board:



```

/***/ Global variables ***/
/***/ Function declaration ***/
/***/ Class declaration ***/
/***/ Task declaration ***/
/***/ ISE function declaration ***/
/***/ Additional Global variables ***/
void setup()
{
}

```

Console output:

Then, verify that the code compiles by click on *Verify* and then if succeed upload the code by clicking on *Upload*.



```

/***/ Global variables ***/
/***/ Function declaration ***/
/***/ Class declaration ***/
/***/ Task declaration ***/
/***/ ISE function declaration ***/
/***/ Additional Global variables ***/
void setup()
{
}

```

Console output:

Building...
C:\Facilino\OTA\Server\arduino-cli.exe compile --fqbn esp32:esp32:iot32 C:\Facilino\OTA\Server\temp\temp-iss-arduino\temp-iss-arduino.ino
Finished with code: 0



```

/***/ Global variables ***/
/***/ Function declaration ***/
/***/ Class declaration ***/
/***/ Task declaration ***/
/***/ ISE function declaration ***/
/***/ Additional Global variables ***/
void setup()
{
}

```

Console output:

Building...
C:\Facilino\OTA\Server\arduino-cli.exe upload --fqbn esp32:esp32:iot32 g:COAG C:\Facilino\OTA\Server\temp\temp-iss-arduino\temp-iss-arduino.ino
Finished with code: 0

Remark: Every time you change the code, you need to compile first and then upload, otherwise the latest compiled code will be uploaded.

g. Block simplification and feature extension

While the previous version of Facilino included many blocks, in this new version, we have included new blocks, updated existing ones and add shadow blocks to provide a hint on the type of input that a block expects.

In particular, from the toolbox, we can see now all blocks variations so the user can directly select the one that he/she is interested in (previously, only the default block aspect was shown, but now we show all of them). For instance, the arithmetic block operation that



included summation, subtraction, multiplication and division in a one single block, now it is shown as if they were four different blocks in the toolbox.

Also, on each input, we have added shadow blocks, which are default input values so that the user does not need to add by himself/herself. Of course, the user can change this default value to something which is more convenient for him/her, but in most of the cases, the default value serves as a hint to the user to know which kind of input is expected, but also as quick code generation, just by dragging few blocks.

We have updated, both the aspect and the code generated of some blocks, particularly those related with Bluetooth, because we have developed an App Inventor extension that is compatible with those blocks. Bluetooth-related blocks have also additional improvements. Among them, we can highlight the fact that now we can transmit and receive data between two ESP32 devices as long as one of them acts as master and the other one as slave. We have also created new blocks, such as HTTP REST API blocks to communicate WiFi devices with a mobile device via HTTP protocol. Another important block that has been modified that is widely used are blocks related to GLOBAL variables. Now, this block is simply added to the workspace, but not necessary in the setup as before. Since they are global variables, their definition is not link to either to “setup” or “loop” sections. In addition to this, PWM-related blocks have their own subcategory in the toolbox, which is easier to find them (before they were included in the Analog section).

We have also included many new blocks that were not included in the old version, such as servo motor attach/detach, ignore output, meArm-related blocks, HTU21D temperature/humidity sensor block,

h. Block filters

When creating a project, we can select the type of project we intend to work with, which affects to the block instruction set. For instance, a generic project will have all blocks available by default, while other type of projects will show by default only some categories. The following table summarizes the categories and subcategories shown for each type of project:

Category	Subcategory	Project type
Functions	-	All
Control	Flow Control	All
	Programming	Generic
	Interrupts	Generic
	State machine	Generic
Logic	-	All
	Bitwise	Generic
Math	-	All
	Array	Generic
	Curve	Generic
Variables	-	All
	Array	Generic
	EEPROM	Generic



Text	-	All
Basic I/O	Analog	All
	Digital	All
	PWM	All
	Button	Generic, Multisensor, Home Automation, LED race
	Bus	Generic
Display	LCD 16x2	Generic, Multisensor, Home Automation
	LED Matrix 8x8	Generic, DYOR
	RGB LEDs	Generic, DYOR, bPED, HomeAutomation
	OLED 128x32	Generic, DYOR, bPED
Communication	USB	All
	Bluetooth	All
	WiFi	All
Light	Infrared	All
	Colour	Generic, DYOR, Multisensor, Home Automation
	LDR	Generic, DYOR, Multisensor, Home Automation
	Dimmer	Generic, Multisensor, Home Automation
Distance	-	All
Sound	Buzzer	All
	Music	Generic, DYOR, bPED
	MP3/WAV	Generic, DYOR, bPED, Multisensor, Home Automation
Movement	Motors	All
	Robot base	Generic, DYOR
	Robot accessories	Generic, DYOR
	Robot walk	Generic, bPED
	Robot arm	Generic, mArm
System	Controller	Generic
	Filtering	Generic
Environment	Temperature	Generic, Multisensor, Home Automation
	Humidity	Generic, Multisensor, Home Automation
	Rain	Generic, Multisensor, Home Automation
	Gas	Generic, Multisensor, Home Automation
	Miscellaneous	Generic, Multisensor, Home Automation

i. Tutorial and Project-related Exercises

We have included a set of tutorials. The basic ones, have been fully integrated in the new version of Facilino, which includes now TinkerCAD simulations to reproduce the proposed exercises even without the need of programming the physical device. This basic exercises, have been designed to understand how to use specific block instructions. On every tutorial, it is explained which are the worked instructions and which are additional ones that will be used in the exercises. Every exercise includes a description and a hint on how to solve it and also includes the solution using Facilino code.



[Learn More](#)

USB Serial Communication

Learn how to print to the console and read from the console with the USB serial interface.



[Learn More](#)

Blinking LEDs

Learn how to work with digital inputs and outputs with Facilino to make LEDs blink.



[Learn More](#)

Logic

Learn how to work with logic instructions with Facilino, including Boolean and bitwise operations.



[Learn More](#)

Flow Control

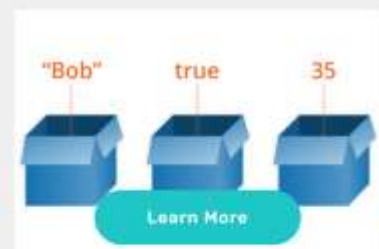
Learn how to make decisions with 'if' statements and repeat blocks of instructions with 'while' loops.



[Learn More](#)

Functions and Procedures

Learn how to implement functions and procedures to reuse part of your code and call it at any part of your program.



[Learn More](#)

Variables

Learn how to work with local and global variables and their types. Also learn how to work with array variables.



[Learn More](#)

Maths

Learn how to use math instructions to perform arithmetic operations, minimum and maximum and even to generate a sinusoidal wave.



[Learn More](#)

Text

Learn how to search, crop, append and compare text with Facilino.



Coming Soon

RGB LEDs

Learn how to set the color of a RGB LED and dimm the light using PWM (analog write).



Coming Soon

Digital Temperature and Humidity

Learn how to read temperature and humidity data from Facilino and set an alarm if temperatures or humidities are too high or too low.



Coming Soon

Sound Buzzer

Learn how to generate predefined sounds, melodies and even melodies that run in a background task.



Coming Soon

Infrared (IR) Receive

Learn how to receive IR codes from your IR remote controller to command 'orders' to your device.



Coming Soon

Liquid Cristal Display (LCD)

Learn how to print and scroll text to an LCD (I2C) screen with Facilino.



Coming Soon

Light Dependent Resistor (LDR)

Learn how to read raw and calibrated values from an LDR sensor.



Coming Soon

Gas Sensor

Learn how to detect smoke and alcohol from a MQ2 gas sensor.



Coming Soon

Relay and Servo Actuators

Learn how to control with a relay and move a standard servo.










Coming Soon

RGB LED Strips

Learn how to set colors of RGB LED strips individually or in group. Also learn how to set the LEDs brightness.

Also, we have created a set of exercises for specific projects.

 <p>Learn More</p>	 <p>Comming Soon</p>	 <p>Comming Soon</p>
<p>LED Race</p> <p>Compete to win a LED race while having fun and learn programming.</p>	<p>DYOR</p> <p>Learn how to program a complete DIY robot with wheels.</p>	<p>bPED</p> <p>Learn how to program a complete DIY walking robot.</p>
 <p>Comming Soon</p>	 <p>Learn More</p>	 <p>Comming Soon</p>
<p>meArm</p> <p>Learn how to program a complete DIY robot arm.</p>	<p>EcoThings</p> <p>EcoThings project: Everything you need to get you started instantly to control lights and temperature in home automation projects!</p>	<p>IoT Starter Kit Intelligent Green House</p> <p>Everything to get you started instantly to control temperature and humidity!</p>
 <p>Comming Soon</p>		
<p>IoT Starter Kit Weather Station</p> <p>Everything to get you started instantly to sense temperature, humidity, gas and rain!</p>		

As we can see, this is still a work-in-progress project and we hope to complete most of them by the end of 2023.

Here we can see a TinkerCAD circuit simulation which includes code generated by Facilino to reproduce a specific behaviour.

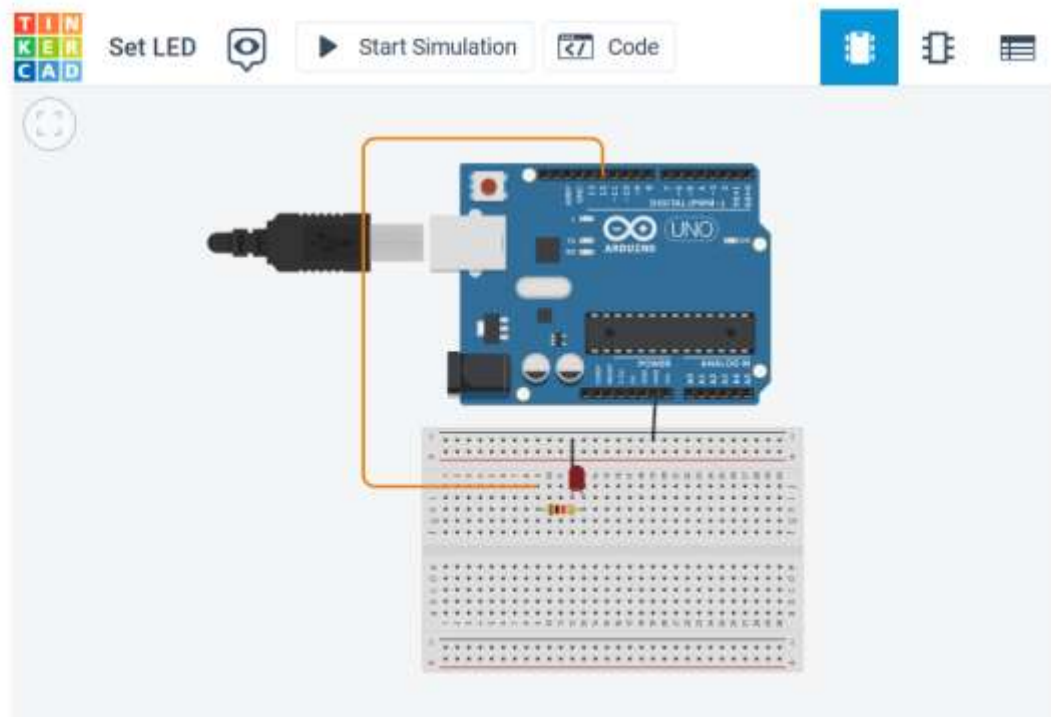
1

Set LED

At start up, set the LED connected to pin D12 HIGH (yes, we know, this is not very impressive, but its the beginning).

Hint:

Use the 'DigitalWrite' instruction and set the state to HIGH.



Here is a list with the number of examples created so far:

	Basic Exercises	Intermediate Exercises
USB Serial Communication	5	2
Blinking LEDs	5	3
Logic	4	3
Flow Control	3	3
Functions and Procedures	4	3
Variables	4	3
Maths	4	3
Text	3	3



j. Translation tool

In order to provide a localization feature when using Facilino, we have included a translation tool that we hope it will help to translate Facilino into multiple languages. The basic idea of this tool is that a user applies for being part of the translation program, which means that he/she will be able to provide translations of specific texts, words and sentences that are used in Facilino and in their examples.

Based on previous contributors and an automated tool using Google translate services, Facilino has been already translated into other languages, but it needs to be revised. Thus, the translation tool asks a user to review/translate some text and he/she can modify it. Those translations will be reviewed by a second reviewer (and possibly more), and then confirmed added to the database.

To apply to the program, users must fill the following form:

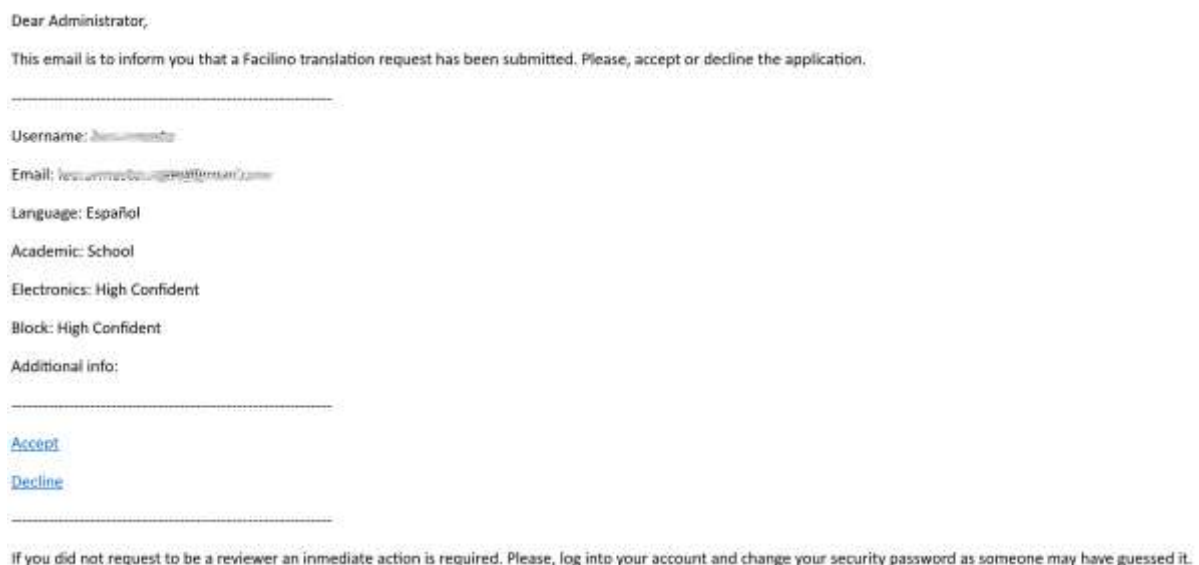


The screenshot shows a web browser window titled "Facilino Translate". The page contains a form with the following fields: "Username", "Academic job", "Electronic expertise", "High confident", "Block Programming", "High confident", and "Additional info". There are "Apply" and "Cancel" buttons at the bottom of the form.



The screenshot shows an email confirmation message with the following text: "Facilino Translate", "An e-mail has been sent to the administrators with your application. Thank you for your interest in collaborating with the translation of Facilino", and a "Continue" button.

Once filled and they apply, the administrator receives an email with the form data:



The screenshot shows an email received by the administrator with the following content: "Dear Administrator,", "This email is to inform you that a Facilino translation request has been submitted. Please, accept or decline the application.", "Username: [redacted]", "Email: [redacted]", "Language: Español", "Academic: School", "Electronics: High Confident", "Block: High Confident", "Additional info:", "Accept", "Decline", and "If you did not request to be a reviewer an immediate action is required. Please, log into your account and change your security password as someone may have guessed it."



User Data

An e-mail has been sent to the user with your response.

[Continue](#)

Dear Ingeborg,

This email is to inform you that your Facilino translation request has been accepted. We thank you in advance for your kind interest. Now, you can translate Facilino into your native language or review pending translations from other contributors. We will, eventually, send you pending reviews, please, accept or decline the tool to review. If you detect an inappropriate translation, please, do not hesitate to report it.

If you did not request to be part of the Facilino Translation Team, an immediate action is required. Please, log into your account and change your security password as someone may have guessed it.

Thanks,

Here, you can see an example of a key to be translated/reviewed:



k. Over-the-air (OTA)

Facilino OTA can be used to program over-the-air devices such as ESP32 and ESP8266. This is a feature that allows programming a device without USB cable.

When creating a new project, we must select Facilino OTA in order to be able to use this feature:



I. Documentation

To be done.



3. Dissemination and Impact

Since Facilino has suffered from lots of changes, we decided to have an operational version of it before disseminating with users that had used the previous version. However, during development, we have performed several dissemination activities for small groups of users:

- 1) Santiago Apostol school has used the tool, during the academic year 2022/2023, with their students to start learning the basics of coding with simple exercises such as turning on a LED, both in a simulated environment using TinkerCAD and with actual electronics using Facilino OTA. As a consequence, the tool has been tested on kids aged 10-12 years old.
- 2) We have collaborated with a secondary school in Valencia, CE Marni, which is not part of this association, but they were willing to use this new version for some of their academic activities, because they were active users of the old version and decided to implement during the academic year 2022/2023. As a consequence, the tool has been tested on kids aged at 12-15 years old.
- 3) Leopoldo Armesto, has encouraged their university students to use this tool to provide code templates as part of an academic work assignment consisting of building a robot within a robotic course of the Industrial Electronics and Automation degree at the UPV. As a consequence, the tool has been tested on adults too, with the main purpose of generating code which they will adapt to their specific needs later on.
- 4) Sara Blanch and Leopoldo Armesto have co-tutorized two vocational students (Computer Science) performing an internship at the UPV, which have been also using this tool to generate code in order to complete their tasks assignments.

References

- [1] Blockly <https://developers.google.com/blockly>
- [2] Facilino (old version) <https://roboticafacil.es/facilino/blockly/Facilino.html>
- [3] Robótica Fácil <https://roboticafacil.es>