SMART PARKING DESIGN ACTIVITY

For the STM Nucleo-WB55RG laboratory and Keil Studio Cloud

Author: Luis Rodríguez Gil <luis@labsland.com>

Updated: 20 May 2025

Table of Contents

Introduction

In this activity, you are tasked with designing the logic for a "smart parking" system, to be implemented on an STM Nucleo-WB55RG board and using Mbed OS 6 and Keil Studio Cloud. This exercise is conducted using the LabsLand's STM Nucleo-WB55RG remote laboratory. Here, you can run your program on actual hardware and remotely interact with it through a live video stream. The parking system is represented by a 3D virtual model, which operates in tandem with the physical STM device, allowing for bidirectional interaction.

Laboratory and materials

To carry out this activity, you'll use LabsLand's STM Nucleo-WB55RG laboratory, a remote facility that provides access to a real STM Nucleo-WB55RG development board. This particular lab version allows you to upload your own compiled file for the WB55RG board, so you may use any toolset you want.

This activity is designed specifically for ARM's Keil Studio Cloud toolset and workflow and to use the Mbed OS 6 API. Keil Studio Cloud is a free online IDE that is powerful yet easy to use and easy to get started with. The Mbed OS 6 API is a C++ OS and API that is

clean, user-friendly, well-designed, and common for many different microprocessor platforms.

To know how to access the lab you should ask your instructor or see the "Accessing and using the remote lab" section.

The lab should be set up with the Smart Parking virtual model, for which you will be creating the logic. This Smart Parking model is a dynamic simulation featuring 3D visualization, capable of interacting with the actual physical board.

The Smart Parking

The Smart Parking is what you will need to control by designing logic for the STM Nucleo-WB55RG board.

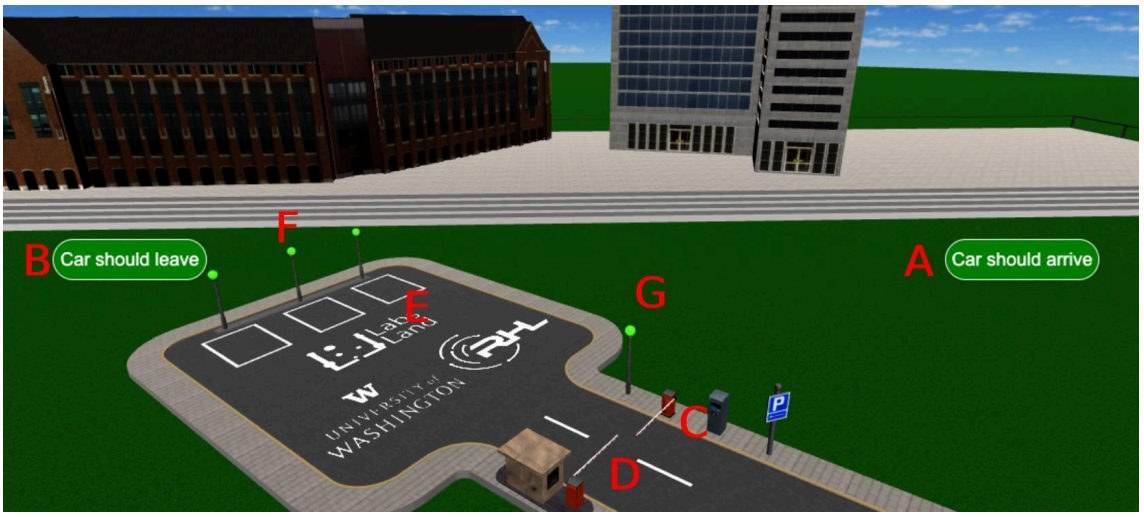


Figure 1. The Smart Parking.

Figure 1 shows the Smart Parking: the virtual model that you will control using the real physical board. Its components are the following:

	Component	Description
A	Car-should-arrive button	Will make a new car arrive to the parking, so that you can test if your Smart Parking logic is working as expected.
B	Car-should-leave button	Will make an existing car leave the parking, so that you can test if your Smart Parking logic is working as expected.
C	Entry barrier	You can control whether it opens. There is a presence sensor in front of the barrier.
D	Exit barrier	You can control whether it opens. There is a presence sensor in front of the barrier.
E	Parking slots	The three parking slots which cars will have available. They have occupancy sensors.
F	Availability indicators	You control these three lights, which should be used to indicate whether a spot is available or occupied.
G	Full indicator	You control this light, which should be used to indicate whether the parking still has slots available or is full.

Hardware & controller

You will control the Smart Parking using a single STM Nucleo-WB55RG board. This is a quite powerful development board by STMicroelectronics which is often used for IoT applications.

To design your logic you will need to know how this board is connected to the sensors and actuators (in this case, the Smart Parking model). Most of them are connected to single pins in the microprocessor. The connections are described in the following table. Note that the names we use are the standard ones for the STM32 pins; and note that the type is from the microprocontroller's perspective (an output is an output for the microcontroller; e.g. an LED).

Component	Pin	Type	Description
Entry barrier presence sensor	PB8	Input	Placed just in front of the barrier. Will be 1 when a car is present in front of the entry barrier, 0 otherwise.
Exit barrier presence sensor	PC12	Input	Placed just in front of the barrier. Will be 1 when a car is present in front of the exit barrier, 0 otherwise.
Parking 1 presence sensor	PC13	Input	1 if a car is present in parking slot 1, 0 otherwise.
Parking 2 presence sensor	PA6	Input	1 if a car is present in parking slot 2, 0 otherwise.
Parking 3 presence sensor	PB9	Input	1 if a car is present in parking slot 3, 0 otherwise.
Entry barrier actuator	PB1	Output	If the signal switches from 0 to 1 the barrier will briefly open so that a car may pass, and close again on its own after a while.
Exit barrier actuator	PB15	Output	If the signal switches from 0 to 1 the barrier will briefly open so that a car may pass, and close again on its own after a while.
Parking full light indicator	PB0	Output	If the signal is 1 then the parking-full indicator will be lit in red, if 0 it will be lit in green.
Parking Slot 1 indicator light	PC4	Output	If the signal is 1 then the parking slot 1 indicator will be lit in green, if 0 it will be lit in red.
Parking Slot 2 indicator light	PD0	Output	If the signal is 2 then the parking slot 2 indicator will be lit in green, if 0 it will be lit in red.
Parking Slot 3 indicator light	PD1	Output	If the signal is 3 then the parking slot 3 indicator will be lit in green, if 0 it will be lit in red.

For the activity you will also use some of the physical peripherals that are connected to the board. These peripherals are described in the Fritzing diagram that is available from the laboratory UI itself, but we list the specific ones here for convenience:

Component	Pin	Type	Description
Green LED1	PB13	Output	Green LED connected to the development board.
Green LED2	PB14	Output	Green LED connected to the development board.
Green LED3	PB15	Output	Green LED connected to the development board. Warning: Pin is shared with the exit barrier actuator (see note). Do not use this LED in this assignment.
Potentiometer1	PC0	Input	Potentiometer 1 (out of the 8 potentiometers of the board)
Switch 1	PB2	Input	Switch 1 (rightmost one in the UI)
Button 1	PC5	Input	Button 1
Button 2	PC6	Input	Button 2
Display SPI CLK	PA5	Output	LCD Display SPI CLK pin
Display SPI MOSI	PA7	Output	LCD Display SPI MOSI pin
Display LCD RES	PE4	Output	LCD Display LCD RES pin
Display SPI CS	PA4	Output	LCD Display SPI CS pin
Servo PWM	PA15	Output	PWM control pin of the servo motor

The servo motor is a SG90 servo. To control it through PWM we recommend you use a min pulse width of 0.5 ms, a max of 2.5 ms, and a period of 20 ms (50 Hz).

Note: The exit barrier actuator is connected to the PB15 physical pin, as depicted in the table. The third external LED is also connected to the PB15 physical pin. This is unusual but done because there is a limited number of pins. Since we will be using the PB15 for the exit barrier actuator, we will not explicitly involve the third green LED in this activity, but know that it will be lit whenever you send a signal to the exit barrier actuator. This is expected and not an error on your part or on the hardware.

Challenge & requirements

You need to design the logic and implement it in the C++ programming language for the STM board. It should be tested with the Smart Parking virtual model in LabsLand and work reliably when cars enter and leave the parking.

The requirements for the logic of the Smart Parking are the following:

- If a car arrives and there are parking slots available, the barrier should open to let the car enter and park.
- If a car wishes to leave the parking, the barrier should open and let it exit.
- Barriers should not open unnecessarily, and if no parking slots are available the entry barrier should not open for more cars.
- If a car parks into one of the three designated parking lot spaces, then its light indicator should be illuminated red to indicate that it is occupied. Otherwise, if free, it should be lit green.
- If the parking lot is full the full parking indicator should be lit red. Otherwise, it should be lit green.
- If the parking lot is NOT full then the physical green LED1 of the board should be lit. If it is full (the 3 parking slots are occupied) then it should be off.
- Every time a car enters the parking a message should be printed to the serial terminal saying “A car has entered. X slots available.”, X being the number of slots that are left.
- Every time a car leaves the parking a message should be printed to the serial terminal saying “A car has left. X slots available.”, X being the number of slots that are available.
- Button 1 (entry barrier override) should force the entry barrier to open regardless of the state.
- Button 2 (exit barrier override) should force the exit barrier to open regardless of the state.
- Switch 1 (emergency stop) should enable an emergency mode. When an emergency mode is enabled the serial terminal should display in uppercase “EMERGENCY MODE” and the entry and exit barriers should not open automatically, they should only

open through the overrides. The mode can be disabled by turning off the switch, and then the system should return to normal operation.

- When the barriers are ordered to open, the servo should also move from 0 to 180 degrees and back to 0.

Accessing and using the remote laboratory and the tools

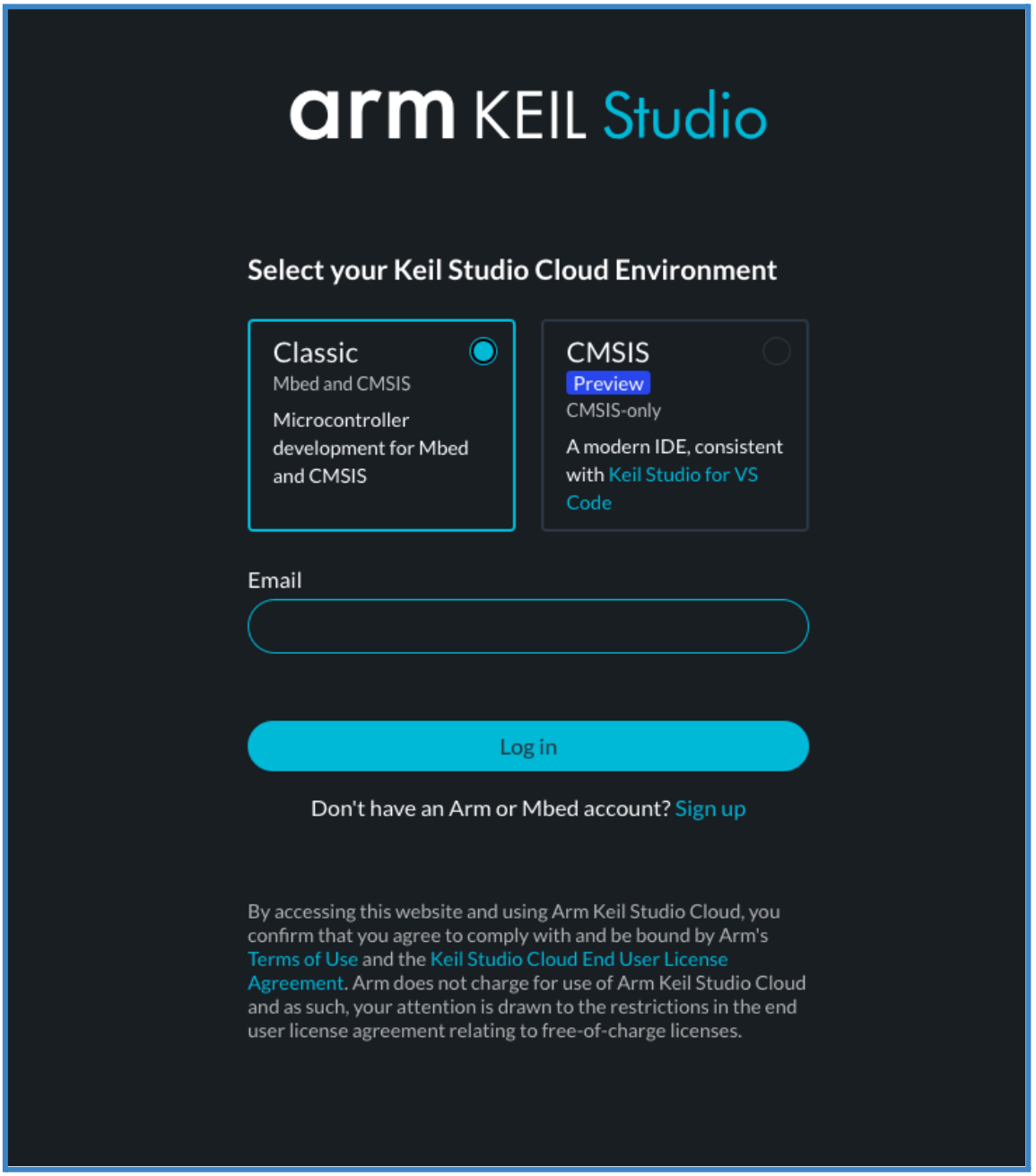
Keil Studio Cloud

Keil Studio Cloud is the IDE that this activity is designed for. It is a free and powerful online IDE that is nonetheless very convenient and relatively straightforward to use. It allows you to compile for specific target boards and to download the binary program. Once you have downloaded the binary program, you can upload it into the LabsLand laboratory, which will be described later, to test it.

Though this section will explain Keil Studio Cloud it is noteworthy that it is also possible to use any other toolset in a similar way, including offline IDEs.

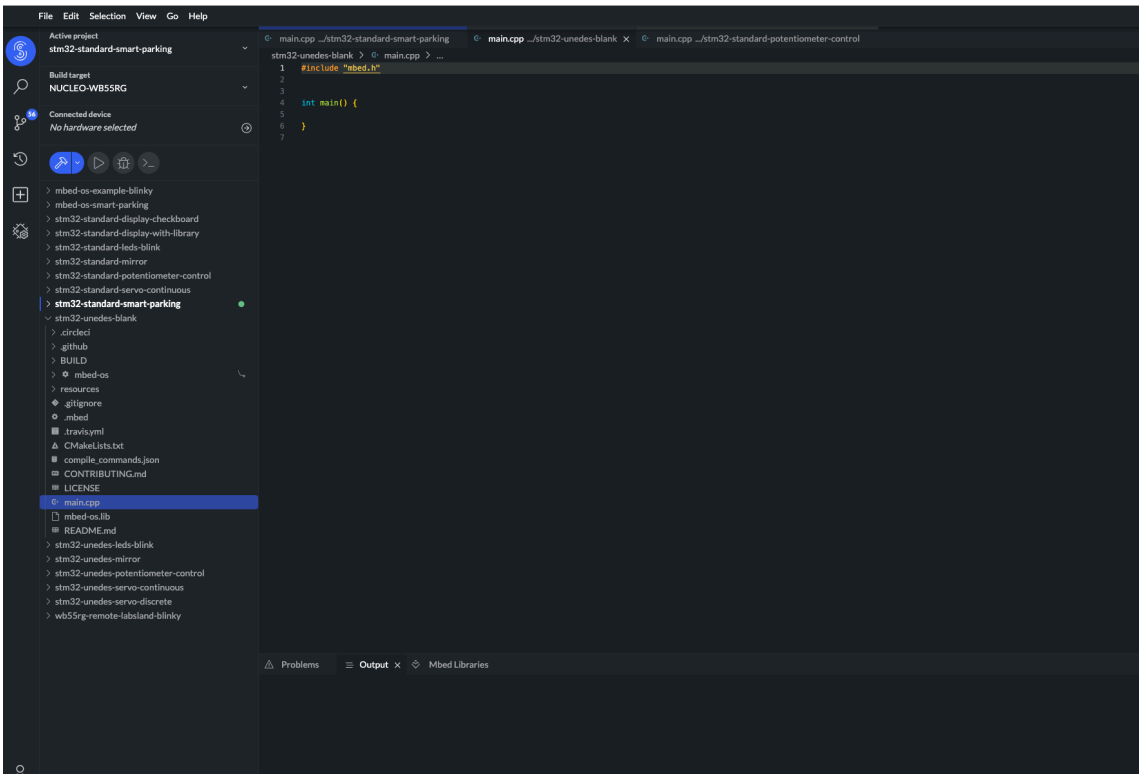
To first access Keil Studio Cloud you should go to:
https://studio.keil.arm.com/

As of now two options are offered:



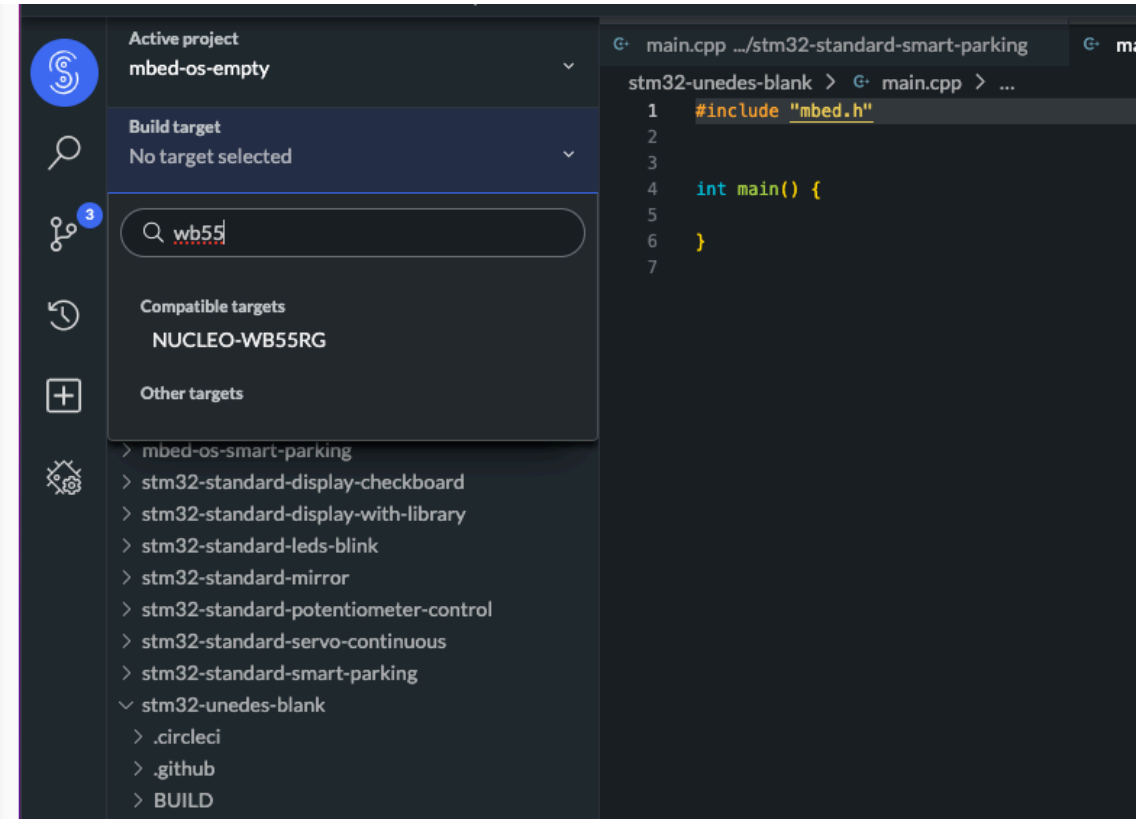
You should pick the Classic version and create an account (click on "Sign up" and follow the instructions).

Once you have created an account and logged in, you should see something similar to this:



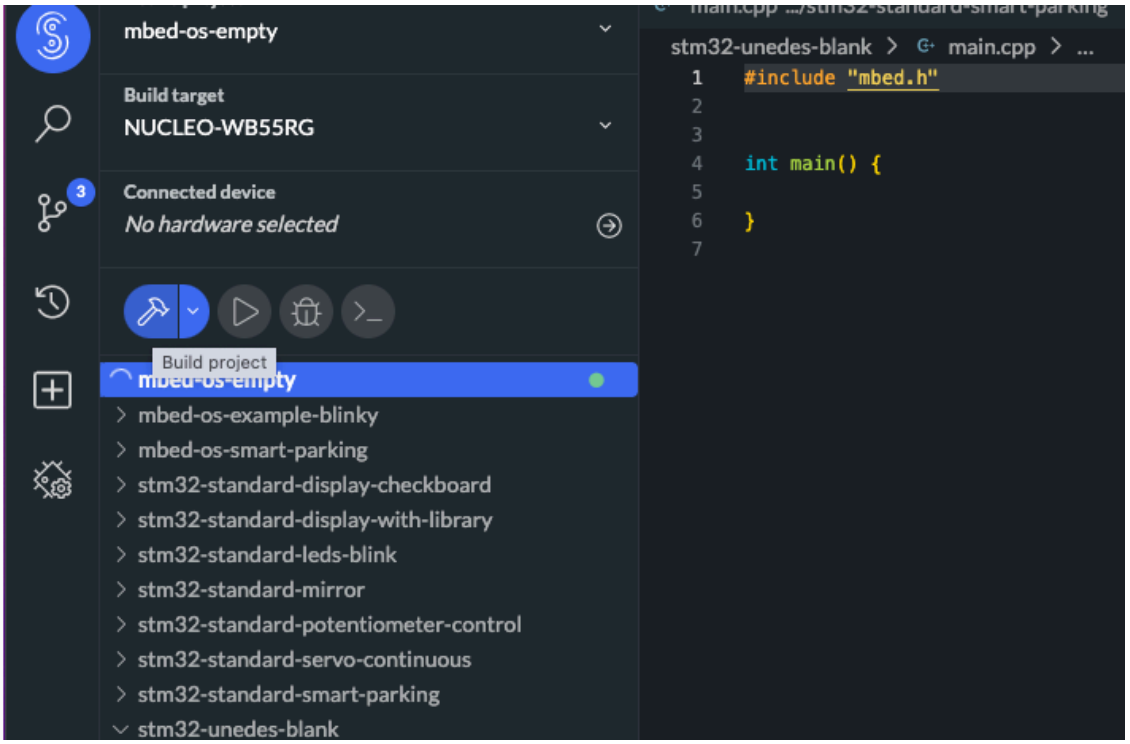
To get started, you can create a new project by clicking on the menu's File -> New -> Mbed Project. You can choose an Empty Mbed project or start from an example such as Blinky.

Next, it is important to configure the right board so that you can compile for it. In our case, you should configure the project for the Nucleo WB55RG board. On the upper left side of the screen, click on "Build target", and select the following:



Then, you can just use the IDE to create files and write your code. The IDE uses the Mbed OS 6 API. Mbed OS 6 is an Open Source OS oriented towards IOT which provides a common framework for many different processors and boards. You can find out more about it here:
https://os.mbed.com/docs/mbed-os/v6.16/introduction/index.html

When your program is ready, to compile it you need to click on the “Build” button, which is the hammer icon in the upper left corner:

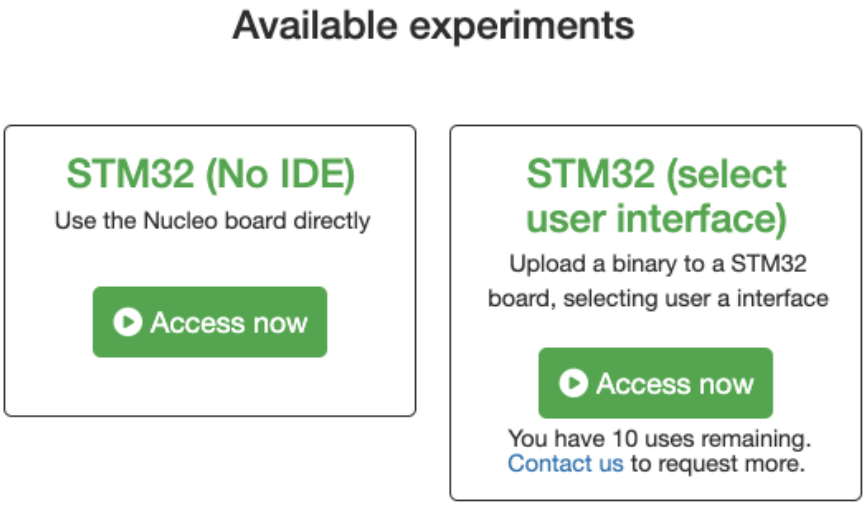


If there is no compilation error, your "active project" will be built and your compiled file will be downloaded. You will then be ready to switch to the LabsLand laboratory to upload and test your compiled file.

LabsLand STM32 Laboratory

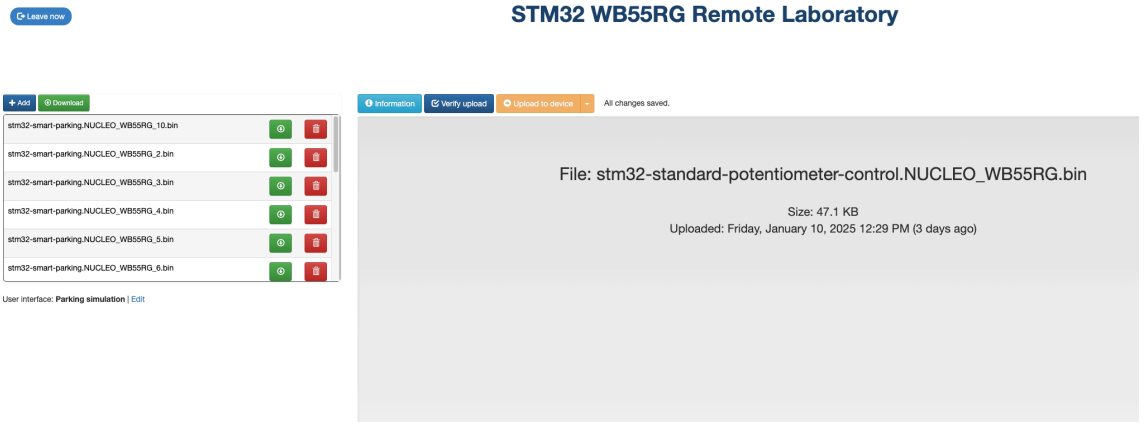
Now that you have your binary file ready, you can upload it into the LabsLand STM32 laboratory (no IDE version). The version of the laboratory that we should use in this activity is called "No-IDE" because you can upload an already-compiled file.

To enter the laboratory, you should follow your instructor's instructions. Depending on the access method, you may have to choose:

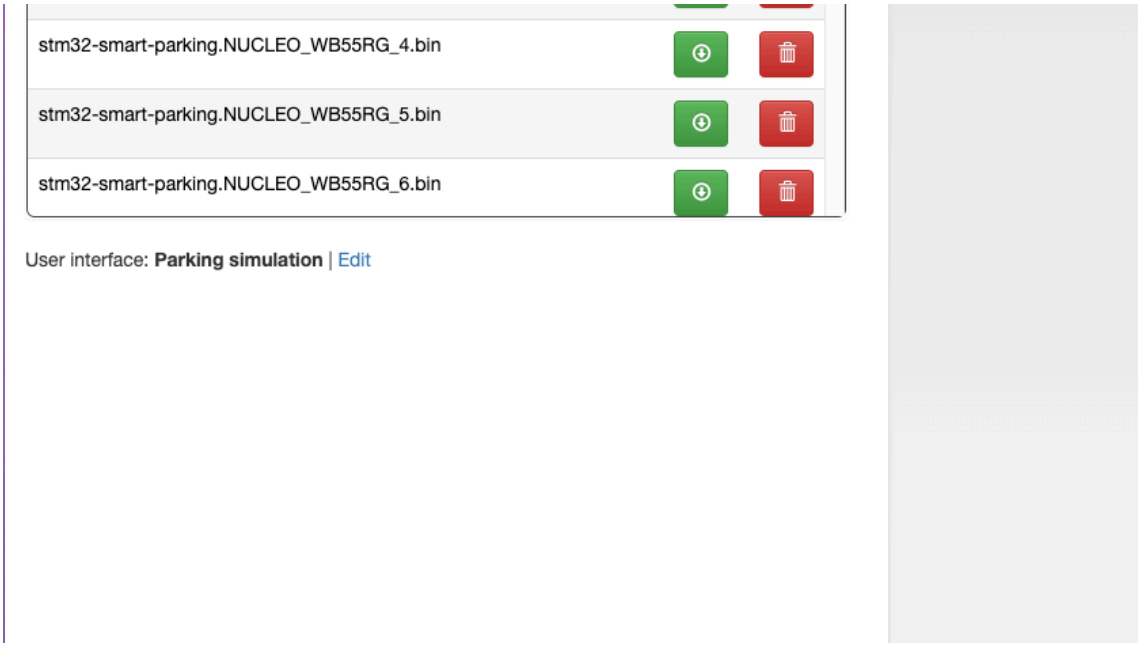


If these options are presented, you should use the "Select user interface" version, since that will allow you to select the "Smart Parking" UI.

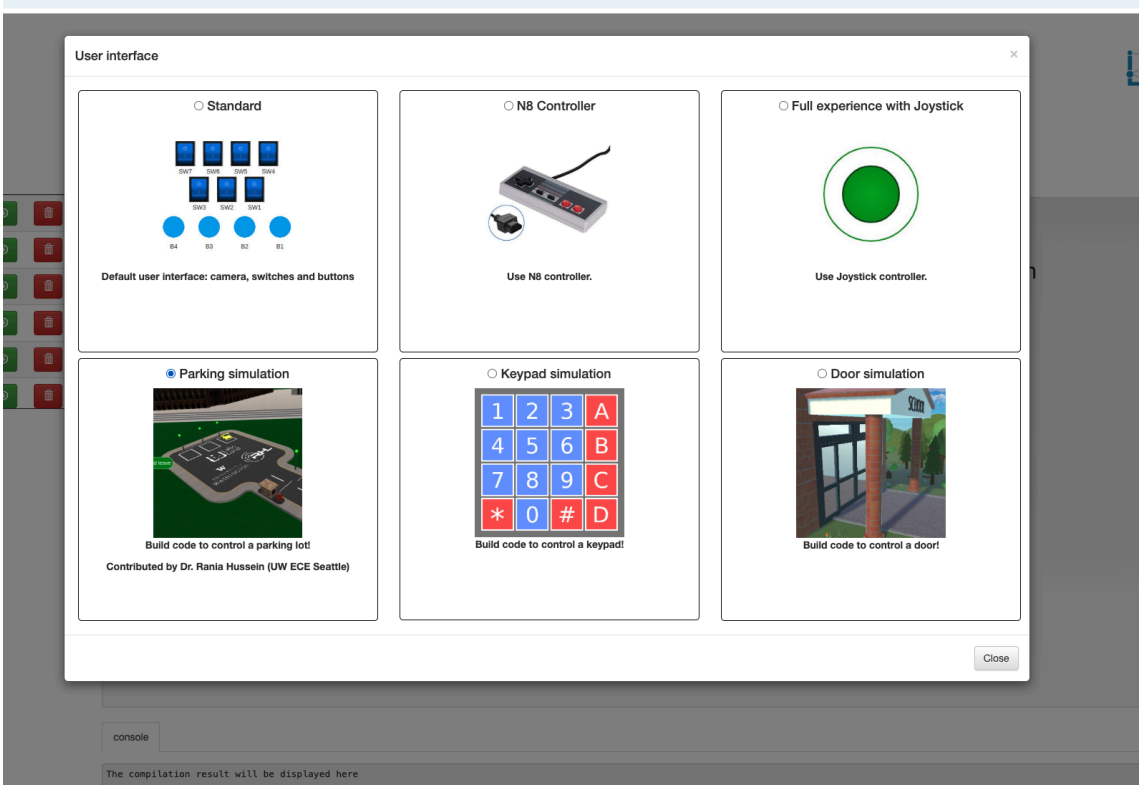
When you enter you should see this interface:



The first thing you should do is to select the right UI. For this "smart parking" activity, you will need to select the "smart parking" UI, on the lower left side of the screen:



Click on Edit, and then choose the Parking in this screen:



Now, to get started, upload your compiled file into the control on the left side. It is important to make sure that the file you upload is the compiled file by Keil Studio Cloud (or another IDE9, and **not** a source file. This version of the lab does **not** support directly providing the source file.

The environment allows you to upload different programs. Now that you have uploaded the current one, you must select it on the list, click on "Verify upload" (the blue button), and if all goes smoothly, you can click on "Upload to device" to upload your program into the board and to move into the actual lab.

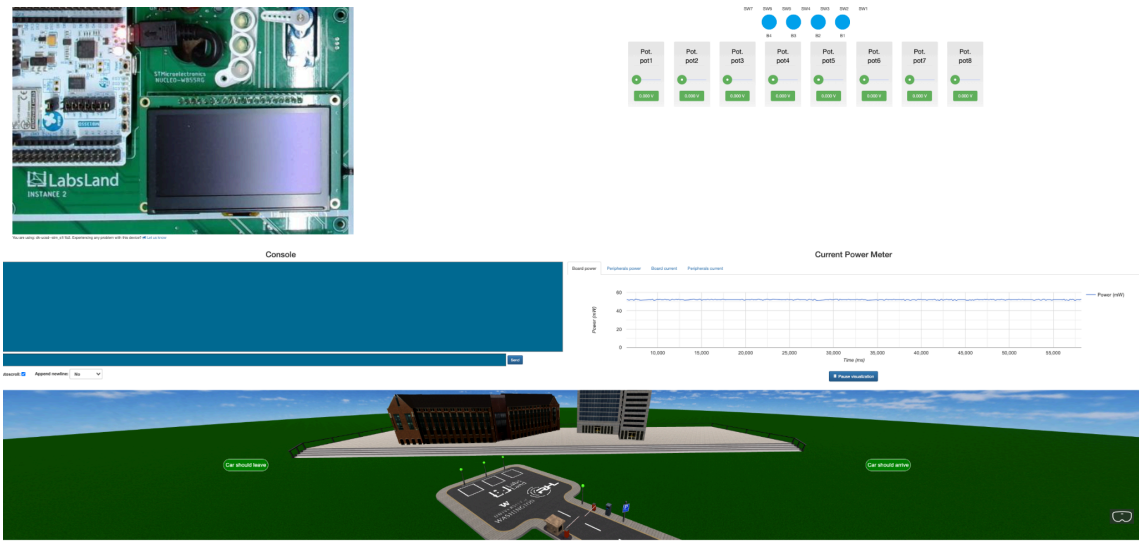


Figure 3. The laboratory UI and the smart parking 3D environment.

Figure 3 shows the UI to interact with the development board that will be running your logic. You can see various UI components, including a live-stream of the physical board, the console terminal, various buttons and switches, and the Smart Parking 3D environment itself.

Your program should now be running on the real board and controlling the virtual hardware, so you should now be able to test whether the program you wrote is working.