ACTIVATE ARDUINO LIBRARY

Documentation for Altair Activate Library

1. Introduction

Altair Activate, version 2020.1 and above, provides a block library to support Arduino boards.

This document covers Arduino and the Arduino IDE, the Activate Arduino Library and a first tutorial. Beyond this document, the library includes several demos accessible from the Library Browser.

2. ARDUINO

2.1. What is Arduino?

Arduino (www.arduino.cc) provides an environment, boards and software to design and develop electronics-based applications. Initially created for prototyping, Arduino has become a very popular and rich ecosystem.

Activate 2020.1 introduces an Arduino block library to facilitate the modeling, simulation and code generation for Arduino boards.

2.2. Boards Supported by Activate 2020.1

Activate does not require the specification of the board to which it is connected and is therefore flexible to support any Arduino board.

For the Activate 2020 release, the Arduino library has been tested with UNO and MEGA boards.

3. ARDUINO IDE

3.1. Why Arduino IDE?

The Arduino IDE provides a graphical environment for the development of Arduino software. For Activate, Arduino's IDE is used to leverage the Arduino compiler and download compiled code to the target board.

Arduino code (.ino extension) can be opened in the Arduino IDE, compiled and uploaded to Arduino boards. The process is described in detail below (see Section 4).

3.2. Recommended Version

The Arduino Library blocks and features included with Activate 2020.1 have been tested with the Arduino IDE software version 1.8.12.

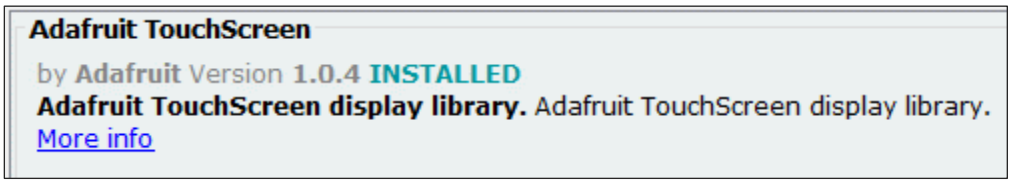
3.3. Where to Download Arduino IDE?

Website: https://www.arduino.cc/en/Main/Software.

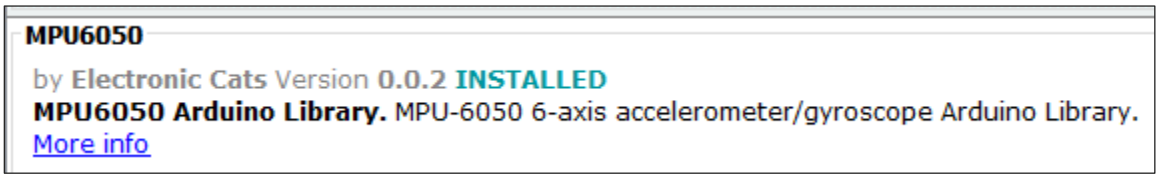
3.4. Required Extensions/Libraries for the Arduino Block Library

The following Arduino libraries are required to support some of the blocks provided in the Activate Arduino block library:

- Adafruit touchscreen



- MPU6050



3.5. Importing Libraries into Arduino IDE

Many libraries can be imported from the Arduino website for use in the Arduino IDE Library Manager. To use the Library Manager for installing Arduino libraries, follow these steps:

1. Run Arduino.exe to start the Arduino IDE.
2. Select **Sketch** > **Include Library** > **Manage Libraries**.
3. In the Library Manager Search box, enter the full or partial name of the library for which you want to search.
4. Select the library. Note that some libraries require that you select a version before you can install them.
5. Click **Install**. By default, the library is downloaded to this location: C:\Users\<user-name>\My_Documents\Arduino\libraries.

4. ARDUINO Library

4.1. Introduction

The Activate Arduino block library can be used in two different modes. In simulation mode, all computations are performed by Activate and the Arduino board is essentially used to interface with the electronic components to which it is connected. This task in Arduino is realized using a generic ino code that should be uploaded once into the board.

In the second mode, Activate generates a custom ino code, which includes all the computations, so that the Arduino board can operate as a stand-alone device without any connections to Activate. The custom ino code in this case is produced by the Activate code generator from an Activate diagram.

4.2. Library Overview

The Activate Arduino block library includes the following blocks:

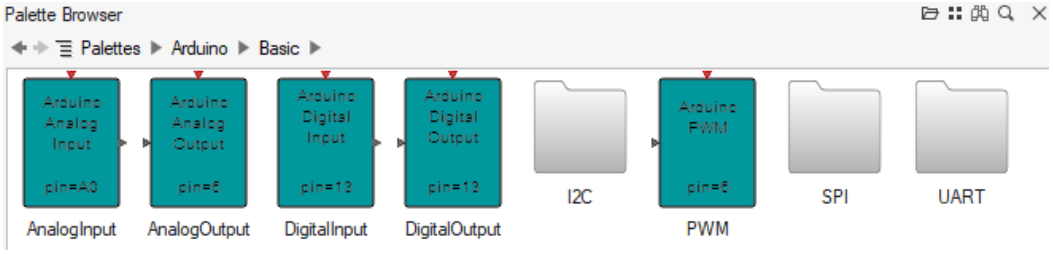
- **Configuration**

The Configuration block is mandatory and must be included and properly parameterized in the Activate diagram.

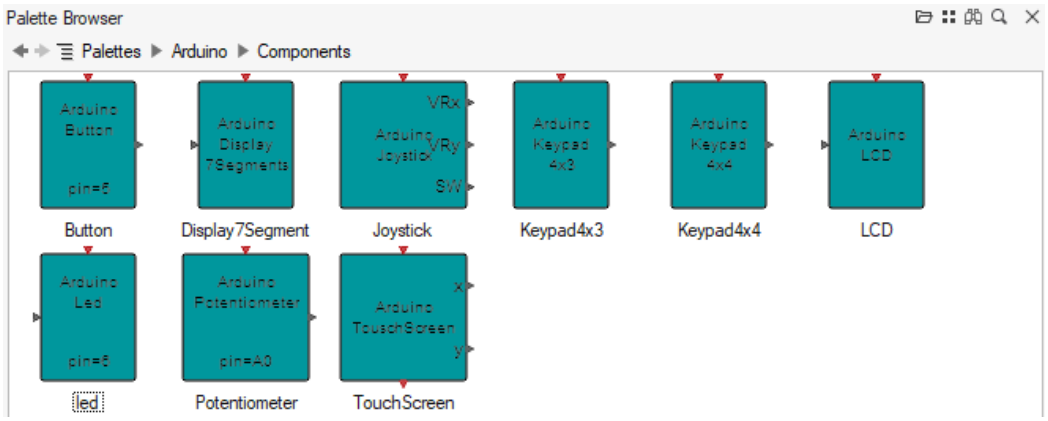
In particular, the COM port is defined on this block. On a PC, the COM port is emulated by the USB port. The Arduino board is connected to the USB port of the computer. The actual COM port assigned by Windows is indicated by the Windows Device Manager. This information is required to define the Configuration block.

Other components are shown below. Refer to the online help or F1 help of the individual blocks for more details.

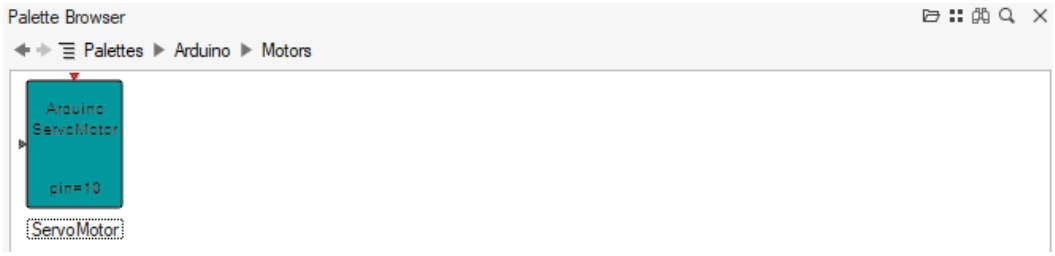
- **Basic**



- **Components**



- **Motors**



4.3. How Does the Library Work?

The Activate Arduino block library achieves two main objectives:

- **Simulation**
  In this mode, the code runs in Activate but a live connection and exchange of data occurs between Activate and the Arduino board.

- **Code Generation**
  Activate generates a code designed to run autonomously on the Arduino board.

4.4. Serial Communication

In Simulation mode, the Activate Arduino block library uses COM information in the Arduino configuration block to open a serial communication with the Arduino board. Activate supports the connection of several Arduino boards simultaneously, in which case multiple Arduino configuration blocks, one for each board, must be placed in the model.

Every Arduino block has two parameters in the Serial tab to specify the following:

- if opened serial communication is to be used.
- which communication port is to be used.

If the model contains only one Arduino Configuration block, select the “Use Opened Serial” check box (the default option). In that case, the serial communication port does not need to be defined in each block.

If the model is connected to more than one Arduino board, clear the “Use Opened Serial” check box and instead define the communication port for each block. In this case, a best practice is to place into a separate super block all of the blocks representing components connected to a board and mask the super block by making the communication port a mask parameter of that super block. This makes the model more modular and thus easier to share.

Notes

- In Code Generation mode, only one board is supported. This restriction should be lifted in the future.
- To generate code for a given super block containing Arduino blocks, the super block should not have any inputs or outputs.
- In Activate, data of type double are stored as float64_t. In Arduino, they can be stored either as float32_t or float64_t depending on the type of the Arduino board.

4.5. Activate Arduino Block Library: *Simulation* Mode

The Activate Arduino block library contains a set of blocks that can communicate with an Arduino board.

These blocks can be simply-defined for use with the standard Arduino libraries, or they can be complex and require the use of external libraries (see above the list of needed libraries).

The Arduino library includes the folder *_bin/Arduino* which contains *ino* files. These files define a generic Arduino code that is used by Activate during simulation to communicate with the Arduino board.

Before running an Activate model containing Arduino blocks, a best practice is to ensure that the “sketch” provided in the library is uploaded to the board. To achieve this, open the *arduino.ino* file in the Arduino IDE, compile, and upload the result to the board.

Once this code is on the board, any Activate model containing Arduino blocks can be used in *simulation* mode.

- **How Does Activate Communicate with That Generic Code?**

Each Arduino block is either a basic block, i.e., has a simulation function, or is a super block containing basic blocks. During model simulation the following occurs:

1. Every time the simulation function of a basic block is called, it sends an array of characters to the Arduino board through the serial port.

2. The generic Arduino code, running in the loop waiting for the message, receives data on its serial port.
3. The received data is then decoded, and the corresponding operation performed in Arduino.

A simple example is used below to illustrate these steps. In this example the Arduino board is used to read a digital value.

1. On the Arduino side, to read data on a digital port, the port should be defined as an input and the function *digitalRead* be used.
2. The simulation function of the Activate block, DigitalInput, sends two coded strings: one for initialization and one every time the block needs to read the value on the digital port.
3. The array for initialization contains the following ascii code: {**2, 0, n, 2**}. It is sent by Activate and received by Arduino.
4. The Arduino code, seeing the code **2** knows that this message concerns a digital port, the following code, **0** is for initialization, and **n** is the number of the pin coded as an ascii character, and the last **2** indicates that the pin should be set as INPUT_PULLUP, so the Arduino code called in that case will be ***pinMode(n,INPUT_PULLUP).***
5. Then, every time the Activate block is called, the simulation function sends the following code {**2, 1, n**} where **2** is for digital, **1** for read and **n** is the pin number (set before as INPUT_PULLUP). The Arduino code called in that case is ***uint8_t dgv=digitalRead(n);***
6. The arduino code sends the value of the digital pin to Activate through the serial port using the code: ***Serial.write(dgv);***
7. The block simulation function waits for the value until it is available on the serial port; it then reads it and copies the value to the output of the Activate block.

4.6. Activate Arduino Block Library Code Generation

The code generation for models using blocks from the Arduino library is in a beta production stage and is not yet exposed as a menu.

To generate embedded code for Arduino:

1. Place all the blocks used in the definition of the embedded code in a superblock.
2. Make sure the superblock does not have any inputs or outputs.
3. Select the superblock in the diagram.
4. Call the function ***vssGenerateInoFile*** from the OML command window; the function returns the generated ino file path.
5. Open the ino file in the Arduino IDE.
6. Compile and upload the code to the Arduino board.

Some known limitations in the current release:

- If the model contains any of the keypad blocks (Keypad4x3 or Keypad4x4), the block should be **inlined** before generating code for the super block. The inlining operation is available through the contextual menu.
- The generated code contains the ino codes corresponding to all Activate Arduino blocks, even if they are not needed.

5. Tutorial on a Simple Model

This tutorial illustrates how to use blocks from the Activate Arduino library to use a button and an LED on an Arduino UNO.

In this example, a button turns on an LED. The LED stays on for 2 seconds, then turns off automatically.

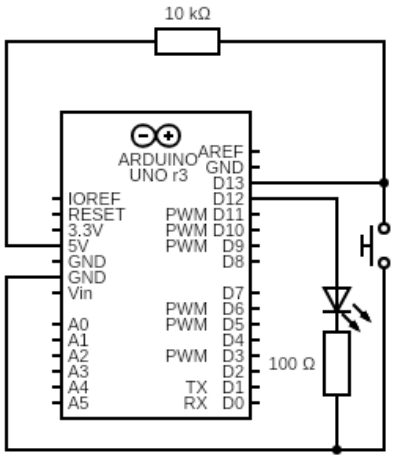
- **Required Hardware**

These components are required to complete the tutorial:

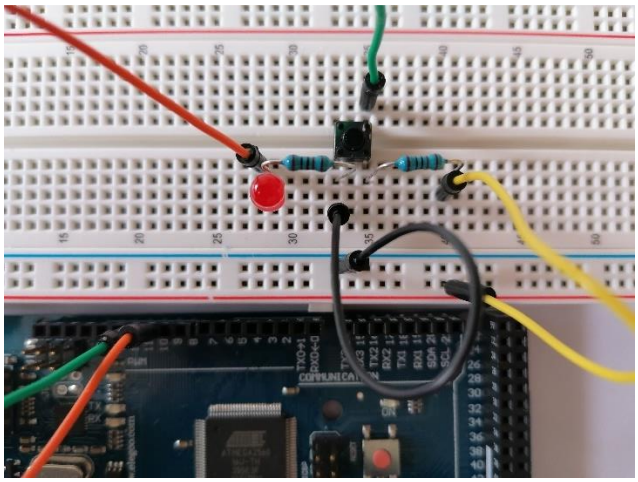
- Arduino Board
- Breadboard + Hook-up wires
- Button, LED
- 10K ohm resistor, 100-ohm resistor

- **Circuit**

The following is an image of the circuit:

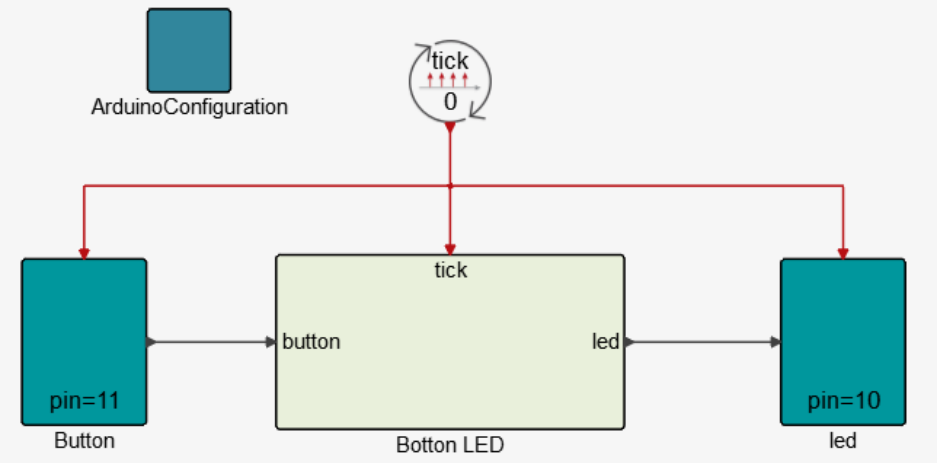


This image illustrates how the components are placed and connected on the breadboard.



- **Activate Model**

The behavior of the button and the LED is defined in the following Activate model:

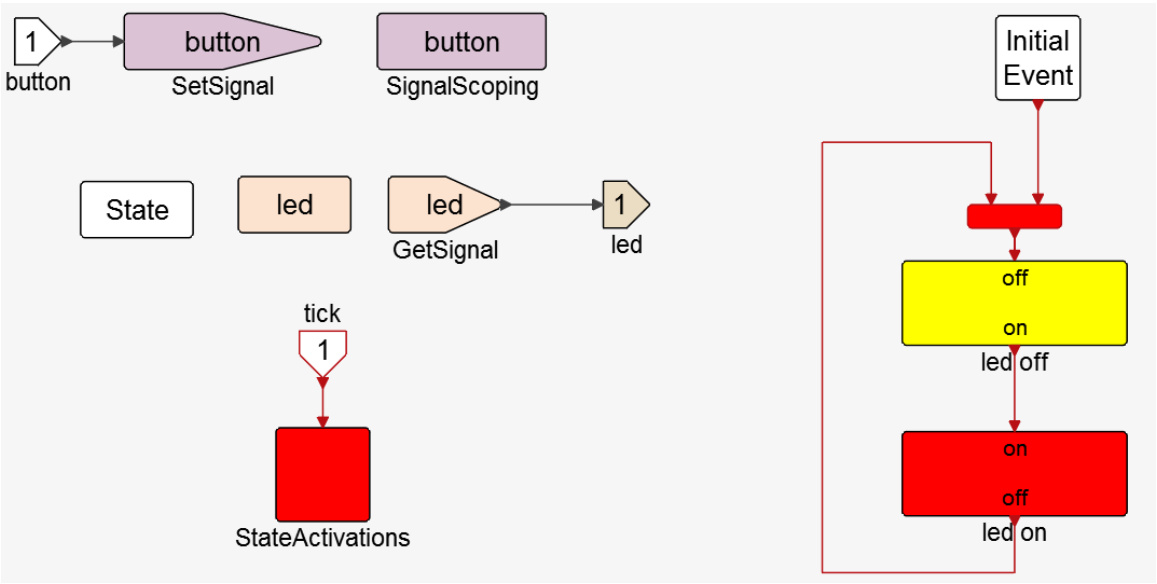


This model contains:

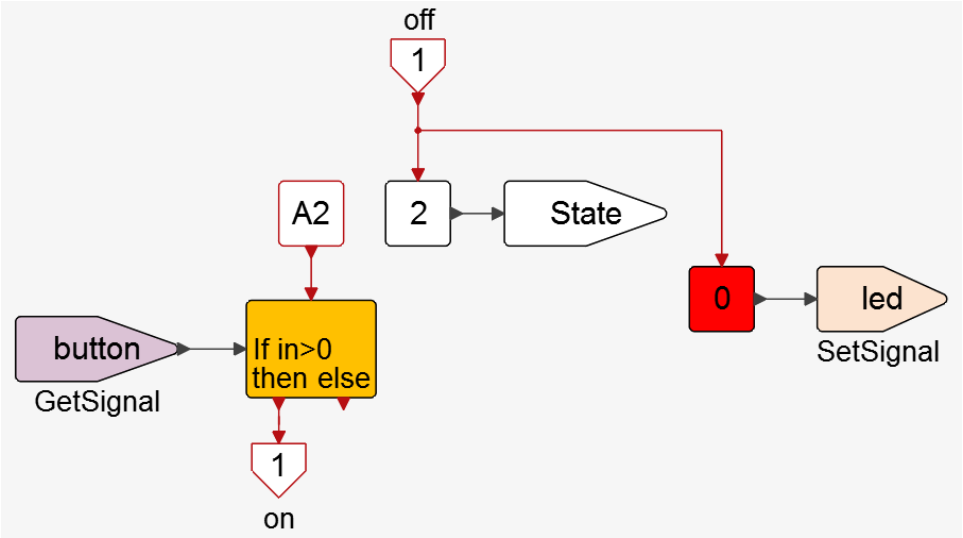
- An Arduino Configuration block used in simulation mode to open the serial communication port with the Arduino board.
- A Button block representing the hardware button connected to pin 11 of the Arduino board.
- An LED button representing the hardware to which the LED is connected to pin 10 of the Arduino board.
- A button super block containing the dynamics of the system. When the button is pressed, the LED switches on for two seconds, then switches off.

The model uses a state machine construction in Activate. This method of constructing state machines is presented in the Activate Extended Definitions .pdf file.

In this example, the model system can be in two different states: one where the LED is on, and another where it is off:

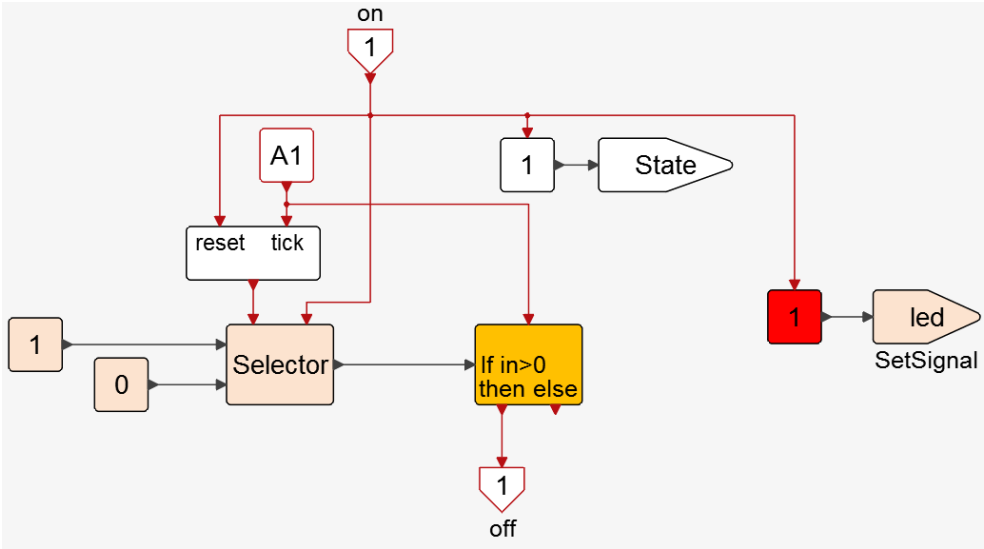


Content of the **LED-off** super block



Initially this super block is activated through its activation input port. This process changes the state number to 2 and sets to zero the **LED** signal. Subsequent activations come through the activation signal A2. No output activation is generated until the **button** signal goes positive. The output activation signals **on**, then enters the **LED-on** super block.

Content of the **LED-on** super block



The **on** signal changes the value of state to 1 (thus deactivating the state represented by the **LED-off** super block and activating the **LED_on** super block state. It also sets the signal **LED** to 1 (turning on the LED) and resets a counter.

The subsequent successive A1 activation signals then increments a counter (through tick input) incrementing the count until the desired value (chosen so that the process takes 2 seconds).

- **Steps for Simulation**
  - Set the real time scale in the simulation setup of the model to 1. This indicates that one unit of Activate simulation time corresponds to 1 second.
  - Connect the Arduino board to a USB port of the PC.

- Double-click on the ArduinoConfiguration block. The connected serial communication ports are listed. Use the port connected to the Arduino board.
- Open the file : <activate_installation>/hwx/databases/activate/system/Arduino/_bin/arduino.ino with the Arduino IDE.
- In the Arduino IDE, if the needed libraries are not yet installed, install them.
- In the Arduino IDE, compile the file arduino.ino and upload the sketch to the Arduino board.
- Return to Activate, then click the Run button. The simulation runs for 30 seconds.
- On the breadboard, click the button and observe how the LED turns on and off.

- **Step for Code Generation**
  - Set the status of the block ArduinoConfiguration to off (to avoid errors if the Arduino board is not connected).
  - At the top level, select all the blocks and place them in a super block (use the Super Block menu).
  - Select the Super block and execute the function ***vssGenerateInoFile.***
  - Open the created ino file with the Arduino IDE. From the Tools menu, select the card and the ports.
  - Compile and upload the sketch to the card.
  - The code now runs standalone on the Arduino board.