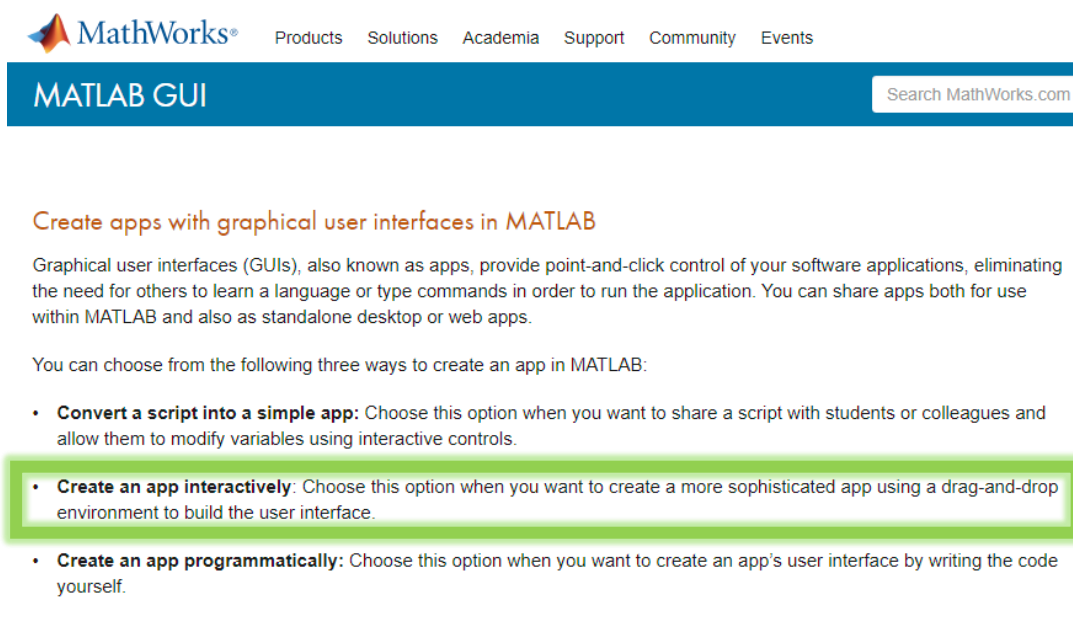


- ⇒ Il y aura un rendu le 14/01 avec un tag sur GitLab.
- ⇒ Il y aura une petite présentation orale le 05/01 (**mettre à jour l'installation de toutes les Raspberry et distribuer le matériel au même moment**) sur :
 - Ce qu'on va présenter
 - Le découpage des tâches jusqu'au 14/01
- 25/11 :
 - Inscriptions à un sujet
 - Appropriation du sujet / Lecture de documents
 - Téléchargement des outils ou récupération du matériel si besoin
- 04/12 :
 - Prise en main de l'environnement de travail
 - Voir si besoin de composants à commander
 - Organisation **obligatoire** de votre [Board Issues Gitlab](#)
 - Premier commit **obligatoire** à pousser sur GitLab
- 11/12 :
 - Dernier délai pour passer d'éventuelles commandes
- 14/01 : Rendu intermédiaire sous forme de **Tag** nommé *vIntermediate*
- 21/01 : Soutenances. Fin des commit sur GitLab à 16h (**fin des rendus**) et livraison sous forme de **Tag** nommé *v1.0*
- 22/01 : Travail complémentaire si exigé par les profs :
 - Fin des commit sur GitLab à 23h59 (**fin des rendus**) et livraison sous forme de **Tag** nommé *v2.0*

Tutos IHM Matlab :

- http://developr6.dr6.cnrs.fr/_media/manifestations/004-ihm/gigot.pdf?id=manifestations%3A004-ihm%3Agigot&cache=cache
- <https://www.mathworks.com/discovery/matlab-gui.html>



MathWorks® Products Solutions Academia Support Community Events

MATLAB GUI

Search MathWorks.com

Create apps with graphical user interfaces in MATLAB

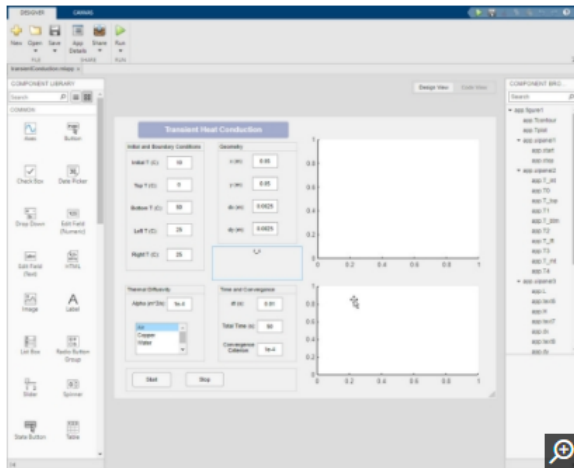
Graphical user interfaces (GUIs), also known as apps, provide point-and-click control of your software applications, eliminating the need for others to learn a language or type commands in order to run the application. You can share apps both for use within MATLAB and also as standalone desktop or web apps.

You can choose from the following three ways to create an app in MATLAB:

- **Convert a script into a simple app:** Choose this option when you want to share a script with students or colleagues and allow them to modify variables using interactive controls.
- **Create an app interactively:** Choose this option when you want to create a more sophisticated app using a drag-and-drop environment to build the user interface.
- **Create an app programmatically:** Choose this option when you want to create an app's user interface by writing the code yourself.

Create an App Interactively

App Designer is an interactive environment that integrates the two primary tasks of app building: laying out the visual components and programming the app's behavior. It allows you to quickly move between visual design in the canvas and developing code in the MATLAB editor.



The App Designer interface.

You can share your app with others to use in MATLAB on the desktop or in a web browser using MATLAB Online. App Designer apps can also be packaged for installation into the MATLAB Apps tab. To share with non-MATLAB users, you can compile apps into standalone desktop and web apps using MATLAB Compiler.

App Designer is good for interactively designing your layout and programming its behavior in one environment. If you prefer, you can program the entire app yourself, including the user interface.

Learn More

- [MATLAB App Designer - Overview](#)
- [Getting Started with App Designer \(4:49\) - Video](#)
- [Component Gallery - Overview](#)
- [Share and Collaborate Using MATLAB Drive - Overview](#)
- [Package Apps in App Designer - Documentation](#)
- [Getting Started: Standalone Applications Using MATLAB Compiler \(3:58\) - Video](#)
- [Sharing with MATLAB Web App Server](#)

<https://www.mathworks.com/help/stateflow/ug/sf4ml-lamp-example.html>

[Products](#)
[Solutions](#)
[Academia](#)
[Support](#)
[Community](#)
[Events](#)

[Get MATLAB](#)

[Help Center](#)

[Support](#)

CONTENTS

[Documentation Home](#)

[Stateflow](#)

[Design Patterns and Applications](#)

[App Design](#)

[Stateflow](#)

[Execution in MATLAB](#)

[Design Human-Machine Interface Logic by Using Stateflow Charts](#)

[ON THIS PAGE](#)

[Control an App Designer User Interface](#)

[Execute Standalone Chart by Using Events](#)

[Connect Standalone Chart to User Interface](#)

[See Also](#)

[Related Topics](#)

[Documentation](#)
[Examples](#)
[Functions](#)
[Blocks](#)
[Videos](#)
[Answers](#)

[Trial Software](#)
[Product Updates](#)

[Design Human-Machine Interface Logic by Using Stateflow Charts](#)
R2020b

[View MATLAB Command](#)

This example shows how to model the logic of a graphical user interface in a standalone Stateflow® chart. Standalone charts implement classic chart semantics with MATLAB® as the action language. You can program the chart by using the full functionality of MATLAB, including those functions that are restricted for code generation in Simulink®. For more information, see [Create Stateflow Charts for Execution as MATLAB Objects](#).

You can execute a standalone Stateflow chart by invoking its input events and using temporal operators. The event- and timer-driven execution workflow is suitable for designing the logic underlying human-machine interfaces (HMIs) and graphical user interfaces (UIs).

- When you use the MATLAB App Designer, callback functions from the interface widgets invoke events in the chart.
- In the Stateflow chart, temporal operators and local data control the properties of the user interface.

For more information on how to use MATLAB to create graphical user interfaces, see [Develop Apps Using App Designer](#).

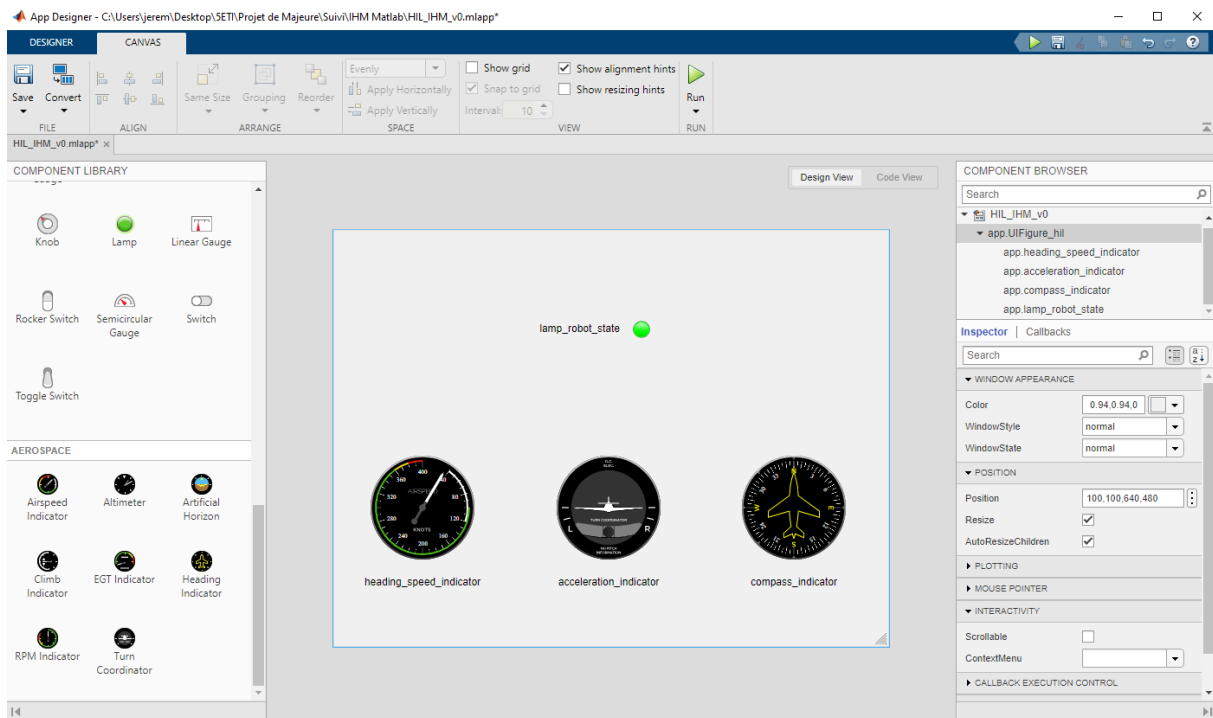
Control an App Designer User Interface

This user interface contains an On-Off switch that controls a lamp. When the switch is in the On position, the lamp lights up in one of two modes, solid or blinking, depending on the position of the Mode option button. You control the rate of blinking by moving the Blink Rate slider.

The file `sf_lamp_logic.sfx` defines a standalone Stateflow chart that implements the logic for the user interface. The chart has input events (`ON`, `OFF`, `BLINKING`, and `SOLID`) and local data (`de1ay` and `app`). The actions in the chart control which widgets are accessible from each state. For instance, the actions in the `OFF` state cause the Lamp widget, the Mode option buttons, and the Blink Rate slider in the user interface to appear dimmed.

Premiers essais :

1) Positionnement d'indicateurs simples



2) Ajout d'actionneurs de test simulés



```

1  classdef HIL_IHM_v0_1 < matlab.apps.AppBase
2
3      % Properties that correspond to app components
4      properties (Access = public)
5          UIFigure_hil          matlab.ui.Figure
6          lamp_robot_stateLabel  matlab.ui.control.Label
7          lamp_robot_state      matlab.ui.control.Lamp
8          compass_indicatorHeadingIndicatorLabel  matlab.ui.control.Label
9          compass_indicator      Aero.ui.control.HeadingIndicator
10         acceleration_indicatorTurnCoordinatorLabel  matlab.ui.control.Label
11         acceleration_indicator  Aero.ui.control.TurnCoordinator
12         heading_speed_indicatorAirspeedIndicatorLabel  matlab.ui.control.Label
13         heading_speed_indicator  Aero.ui.control.AirspeedIndicator
14         test_lamp_robot_stateCheckBox  matlab.ui.control.CheckBox
15         test_heading_speed_indicatorSliderLabel  matlab.ui.control.Label
16         test_heading_speed_indicatorSlider  matlab.ui.control.Slider
17         test_acceleration_indicatorSliderLabel  matlab.ui.control.Label
18         test_acceleration_indicatorSlider  matlab.ui.control.Slider
19         test_compass_indicatorSliderLabel  matlab.ui.control.Label
20         test_compass_indicatorSlider  matlab.ui.control.Slider
21     end
22
23     % Callbacks that handle component events
24     methods (Access = private)
25
26         % Value changed function: test_lamp_robot_stateCheckBox
27         function test_lamp_robot_stateCheckBoxValueChanged(app, event)
28             lamp_robot_state_value = app.test_lamp_robot_stateCheckBox.Value;
29
30         end
31     end

```

3) Test des actionneurs simulés avec les indicateurs positionnés

L'idée est dans un premier temps d'allumer la lampe quand on coche la case de test à laquelle on l'associe et de l'éteindre si elle est décochée.

- ⇒ Quand le système sera complet, ce voyant sera utilisé pour simuler en Hardware In the Loop la LED du robot obligatoire indiquant si le robot est toujours en mission ou s'il a fini sa mission.

```

23     % Callbacks that handle component events
24     methods (Access = private)
25
26         % Value changed function: test_lamp_robot_busy_stateCheckBox
27         function test_lamp_robot_busy_stateCheckBoxValueChanged(app, event)
28             lamp_robot_state_value = app.test_lamp_robot_busy_stateCheckBox.Value;
29             %% Light the light when the test_lamp_robot_state is checked :
30             %% when the robot is busy : light red, otherwise light green.
31             if lamp_robot_state_value == 1
32                 app.lamp_robot_busy_state.Color = 'r';
33             else
34                 app.lamp_robot_busy_state.Color = 'g';
35             end
36         end
37     end

```

➤ https://youtu.be/ydKVamk_Elc

Dans un second temps, on met en œuvre un indicateur de la vitesse en ligne droite qui sera détectée à terme par l'accéléromètre. En attendant la complétion du système Hardware In the Loop, on le simule par un curseur glissant.

⇒ On aura ainsi un suivi temps réel en vitesse du robot.

```
% Value changed function: test_heading_speed_indicatorSlider
function test_heading_speed_indicatorSliderValueChanged(app, event)
    heading_speed_indicator_value = app.test_heading_speed_indicatorSlider.Value;
    %% Update the heading_speed_indicator value with the slider's.
    app.heading_speed_indicator.Airspeed = heading_speed_indicator_value;
end
```

➤ <https://youtu.be/JgDiGK13zkY>

Dans un troisième temps, on choisit d'indiquer la rotation par son accélération et sa vitesse simulées par 2 curseurs glissants respectivement.

⇒ On aura ainsi un suivi temps réel des rotations en vitesse et position.

```
47 % Value changed function: test_turn_speed_indicatorSlider
48 function test_turn_speed_indicatorSliderValueChanged(app, event)
49     turn_indicator_acceleration_value = app.test_turn_speed_indicatorSlider.Value;
50     %% Update the turn_indicator acceleration value with the slider's.
51     app.turn_indicator.Slip = -turn_indicator_acceleration_value;
52 end
53
54 % Value changed function:
55 % test_turn_acceleration_indicatorSlider
56 function test_turn_acceleration_indicatorSliderValueChanged(app, event)
57     turn_indicator_speed_value = app.test_turn_acceleration_indicatorSlider.Value;
58     %% Update the turn_indicator speed value with the slider's.
59     app.turn_indicator.Turn = turn_indicator_speed_value;
60 end
```

On fait de même avec la boussole pour donner un cap simulé par un curseur glissant :

⇒ On aura ainsi un suivi temps réel du cap par la boussole.

```
62 % Value changed function: test_compass_indicatorSlider
63 function test_compass_indicatorSliderValueChanged(app, event)
64     compass_indicator_value = app.test_compass_indicatorSlider.Value;
65     app.compass_indicator.Value = compass_indicator_value;
66 end
```

Voici la vidéo de démonstration complète avec une simulation d'un trajet du robot :

➤ <https://youtu.be/htNL9ZEw4rA>

4) Tutoriel pour relier l'application ainsi créée avec Simulink

<https://www.mathworks.com/matlabcentral/answers/481469-app-designer-linked-to-simulink>