E-bike GUI Design by AppWizard

Example Code Introduction for 32-bit NuMicro® Family

Document Information

Application	This example code utilizes the SEGGER emWin AppWizard tool to create a graphical user interface (GUI) for an electric bicycle (e-bike). It is designed to be compatible with the M467 series development platform to achieve a user-friendly interface for controlling the electric bicycle.
BSP Version	M460_Series_BSP_CMSIS_V3.00.002 emWin Software Package for M460 V3.0
Hardware	NuMaker-HMI-M467

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1. Overview

This example code demonstrates how to use the SEGGER emWin AppWizard tool in conjunction with the NuMaker-HMI-M467 development platform to design the user interface and implement interactivity for an electric bicycle (e-bike) project.

1.1 Principle

1.1.1 Block Diagram

This example code directly utilizes the NuMaker-HMI-M467 evaluation board from Nuvoton's GUI development platform. The required components for development include a 4.3-inch TFT-LCD capacitive touch panel with 480x272 resolution, NuMicro M467HJHAN microcontroller on the main board, and a 4 Mbytes SPI Flash for external storage. The system block diagram is depicted in Figure 1-2.

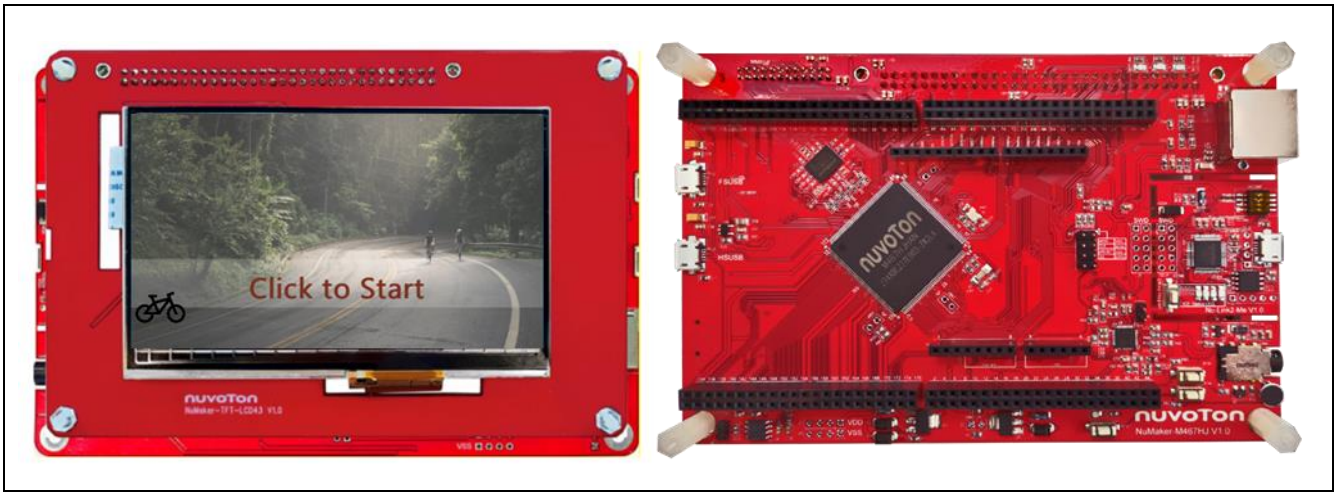


Figure 1-1 NuMaker-HMI-M467

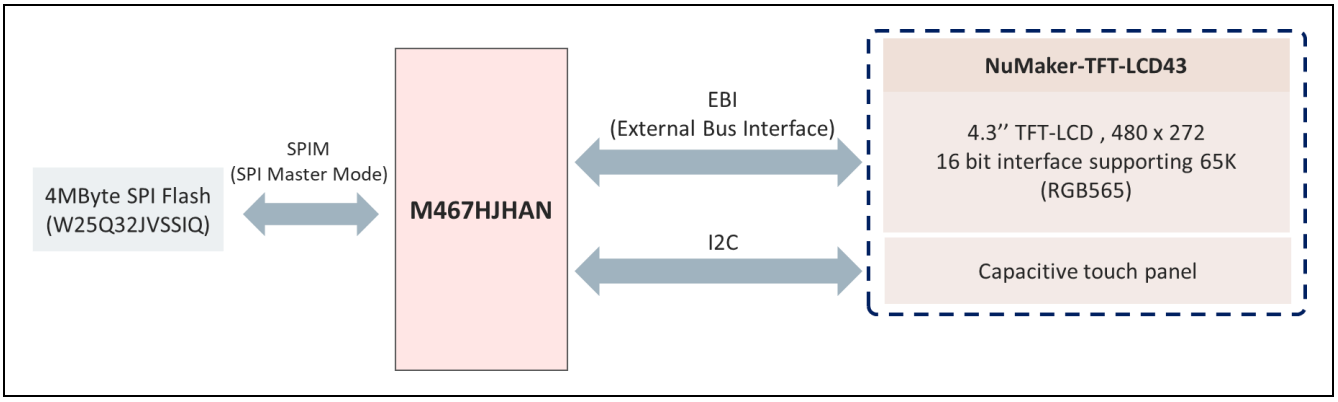


Figure 1-2 Block Diagram

1.1.2 Electric Bicycle Screen

The desired design for the screen in this example is as follows.

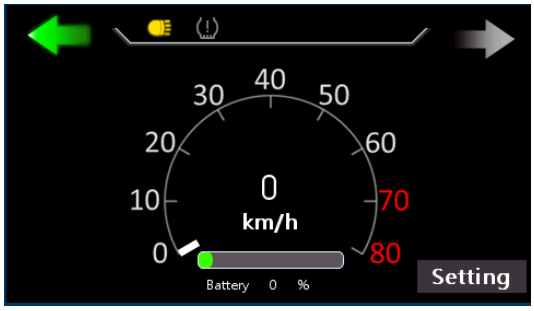
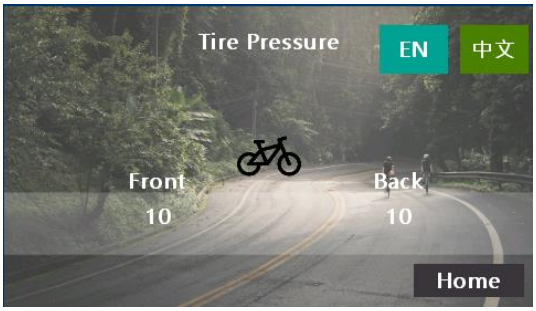
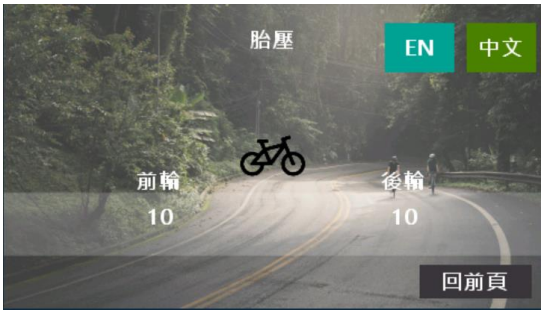
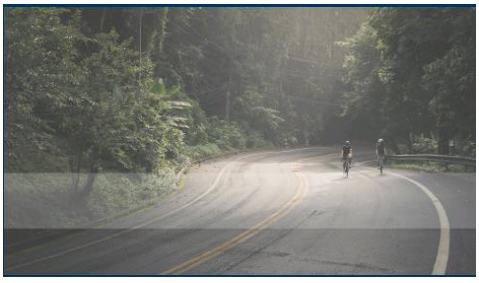
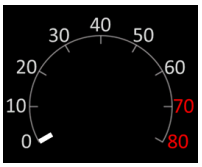
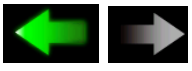
	Switch language to EN	Switch language to 中文(Chinese)
Screen 1		
Screen 2		

Table 1-1 Electric Bicycle Screen

1.1.3 Elements Used in the Electric Bicycle Screen

The elements used in the screen are as follows.

E-bike App Description	E-bike App GUI	Object
A screen divider		Image
Background image in 480x272		
Bike image		

Black background in 480x272		Box
Speedometer		Rotary
Speed number and unit	English interface 	Text
	Chinese interface 	
Battery name, number, and unit	English interface 	
	Chinese interface 	
Tire pressure name and number	English interface 	
	Chinese interface 	
Battery level bar		Progbar
Turn signal		Button
Headlight		
Tire pressure warning light		

Screen switching	English interface 	
	Chinese interface 	
Language switching		

Table 1-2 Elements Used

1.1.4 Supporting Resources

- SEGGER emWin Forum: Segger emWin Forum
- Nuvoton Forum: Nuvoton emWin Forum
- User manual of SEGGER emWin AppWizard: AppWizard User Manual

1.2 Execution Steps

The process of completing a GUI project using the AppWizard can be divided into four stages. In the first stage, hardware and software preparation are required. The second stage involves designing the screens, setting up objects, defining interactions, and configuring animations within the AppWizard interface. In the third stage, the generated code from AppWizard is exported, and custom code can be added to enhance additional functionalities. The fourth stage is programming the board. Each of the stages are described in Section 1.2.1 to 1.2.4.

1.2.1 Hardware and Software Preparation

Please refer to Schematic and User Manual for more related information about the NuMaker-HMI-M467 evaluation board.

To download the AppWizard software tool, go to the "emWin Software Package for M460" download page and click the "emWin Software Package for M460" item to initiate the download. Prior to downloading, registration and login to the Nuvoton official website are required.

After the download is complete, extract the files and open the "M460_emWin_NonOS" folder. According to the path in Figure 1-3 AppWizard – Installation File Path, double-click "*AppWizard_V134a_630a_Install.exe*" to install the AppWizard tool. Please note that if you have an older version of AppWizard, you need to completely uninstall the old version before installing the new version.

Figure 1-3 AppWizard – Installation File Path

1.2.2 GUI Design by AppWizard

1. Open the AppWizard tool using the desktop shortcut.

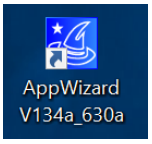


Figure 1-4 AppWizard Desktop Shortcut

2. Click” Create new project” to create a new project.

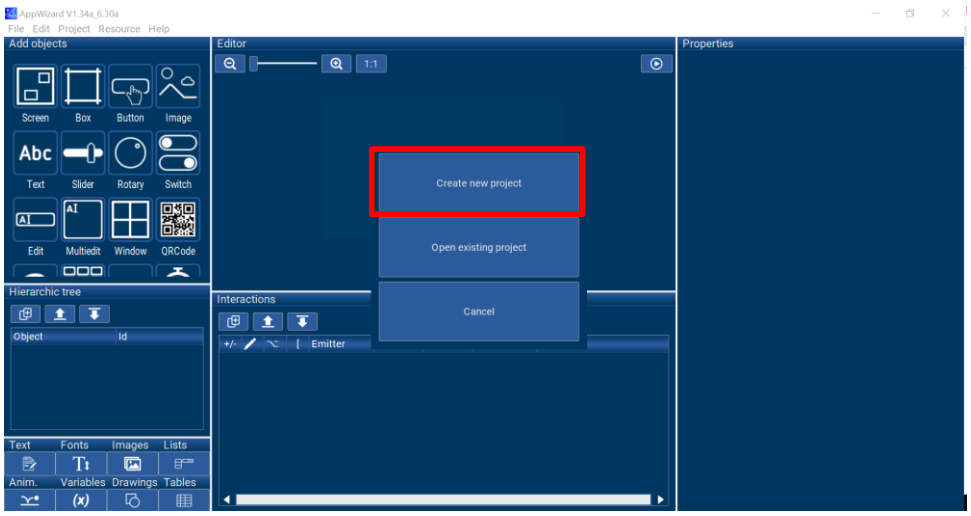


Figure 1-5 Create New Project

3. Edit the project path and project name. Ensure that the color format should match the screen's specification. According to Figure 1-6, choose "16 Bit, GUICC_M565", and check "Enable Multibuffering".

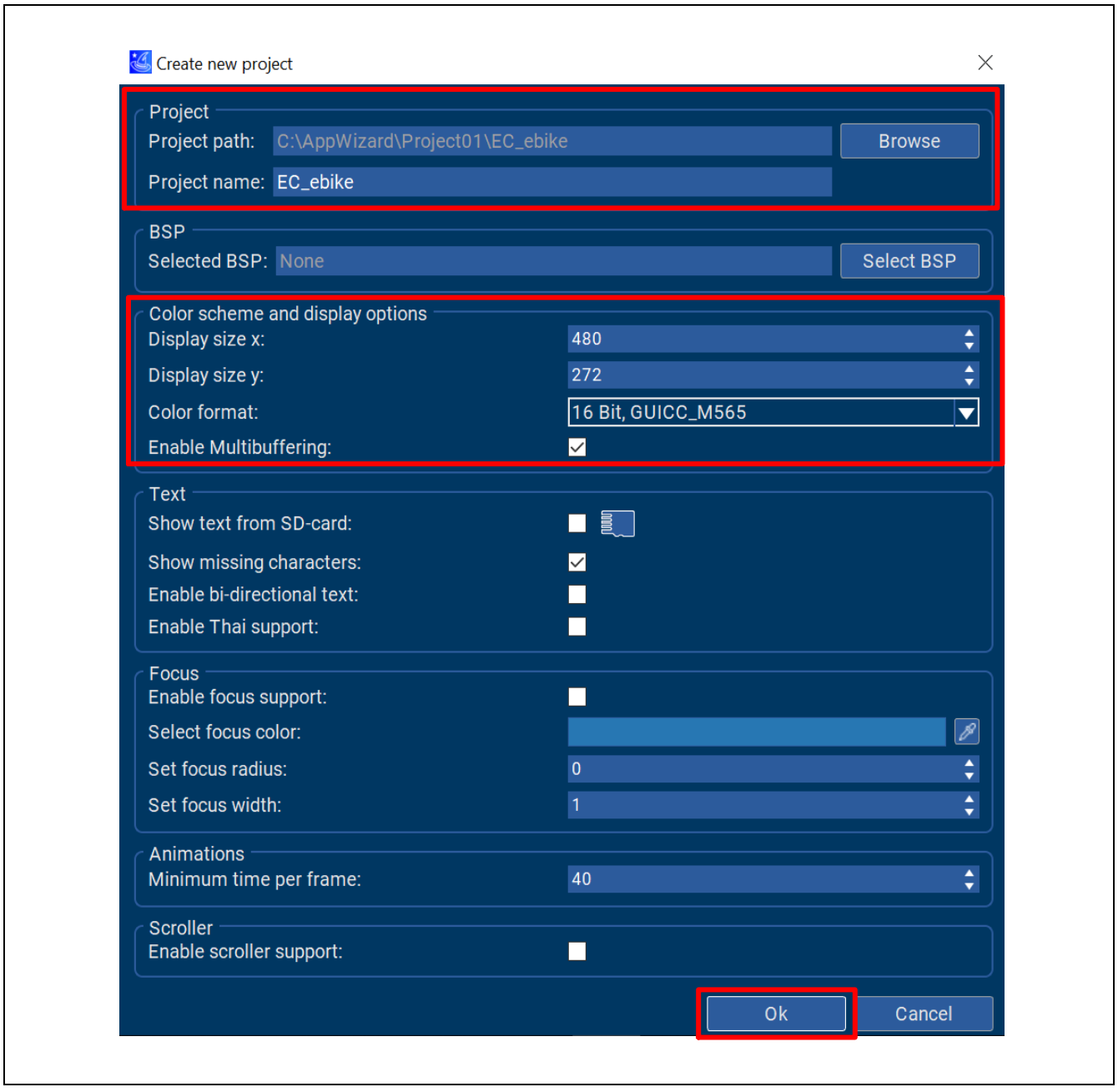


Figure 1-6 New Project Setting

4. Click "Screen" twice under "Add objects" to create two screens. In the Hierarchic tree, name the screens as "ID_SCREEN_Speed" and "ID_SCREEN_Tier" respectively for easier identification.

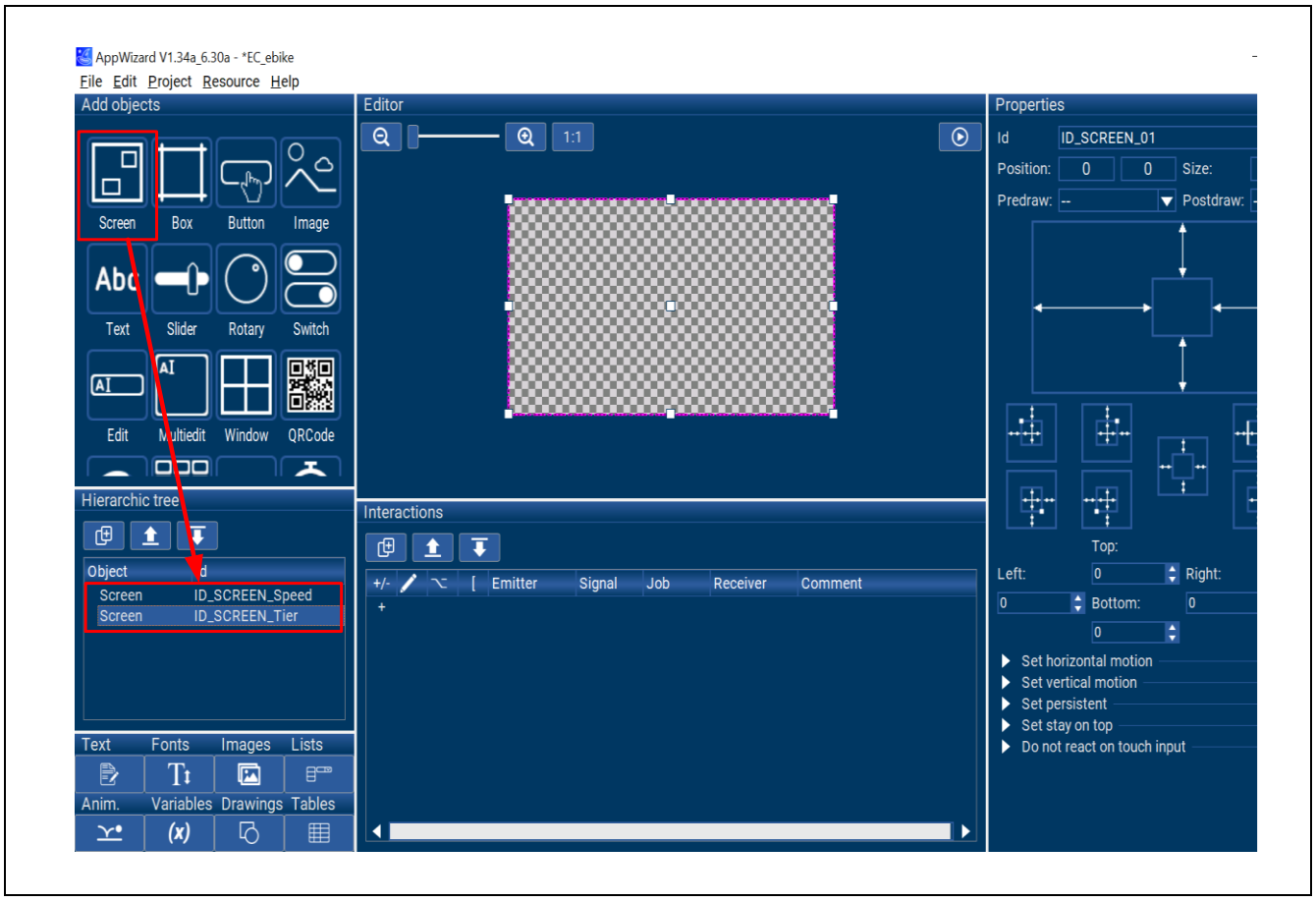


Figure 1-7 Add Screens

5. Click on "ID_SCREEN_Speed" to edit the first screen. Under "Add objects," select "Box" to create a solid color background.

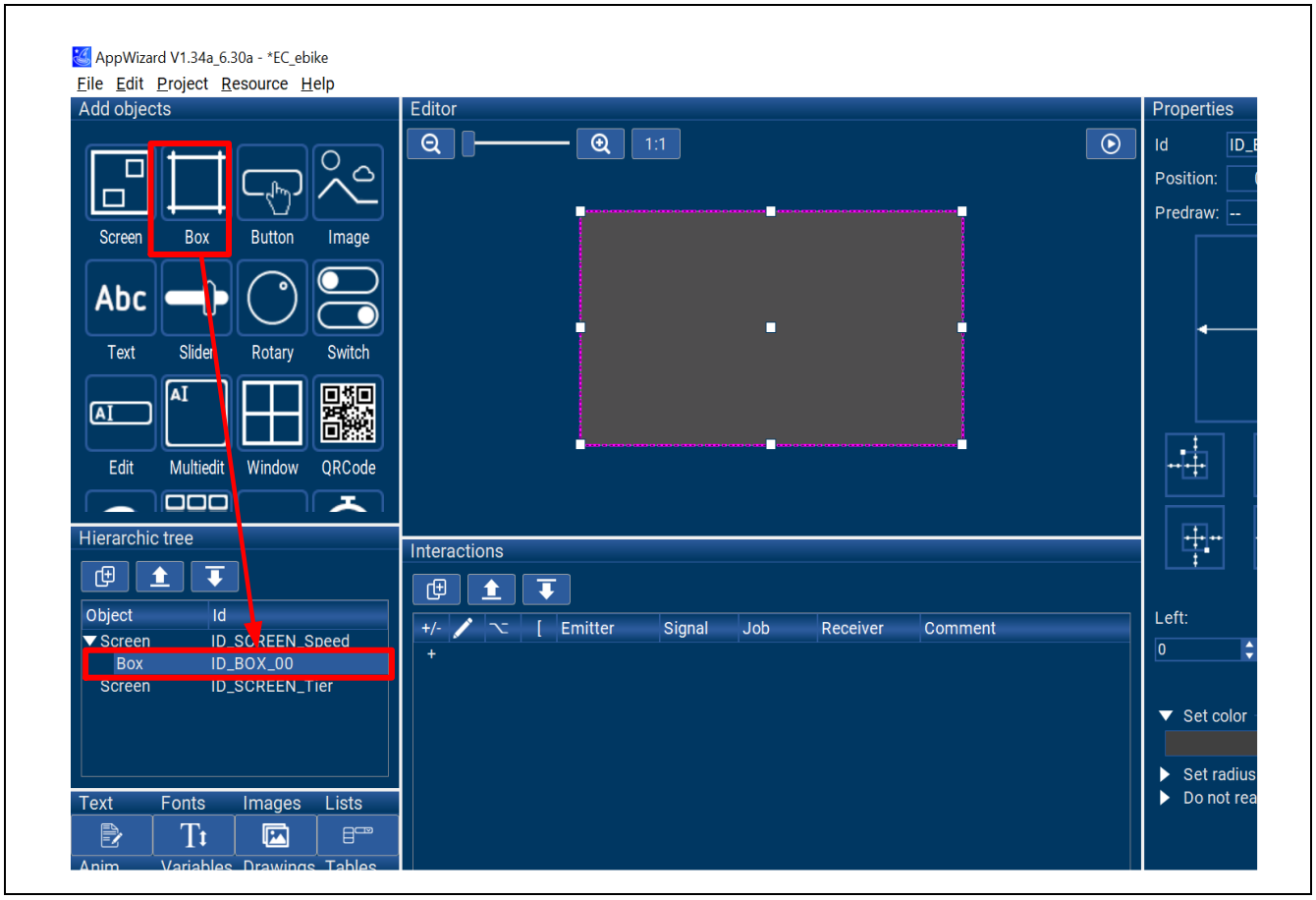


Figure 1-8 Using Box Object to Create A Solid Color Background

6. Configure the relevant properties of the Box widget in the Properties section. Click on "Set color" to set the color of the Box to black.

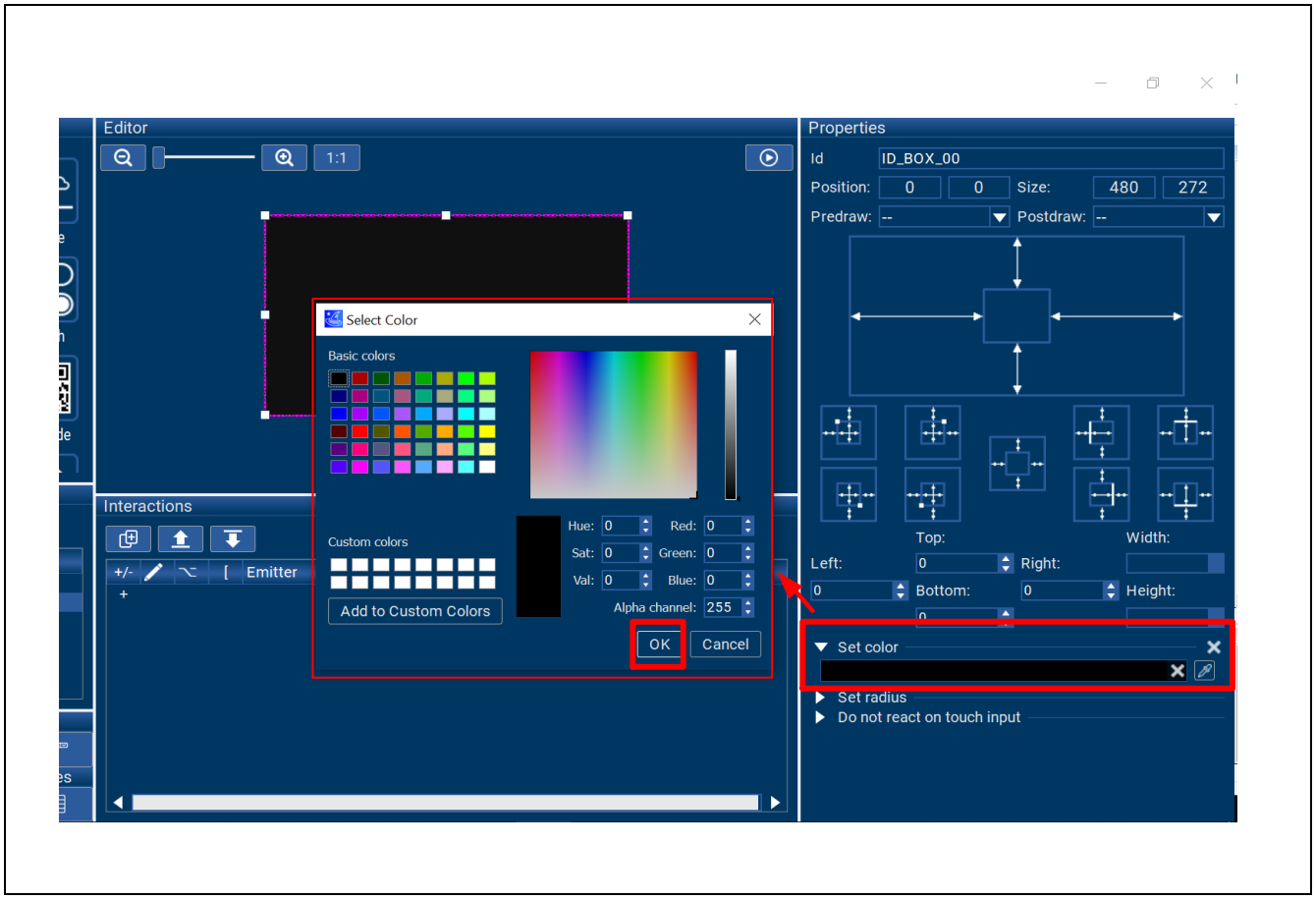


Figure 1-9 Box - Properties

7. Click on "Rotary" in "Add objects" to create a speedometer, and rename it as "ID_ROTARY_Speed" in the Hierarchic tree for better identification.

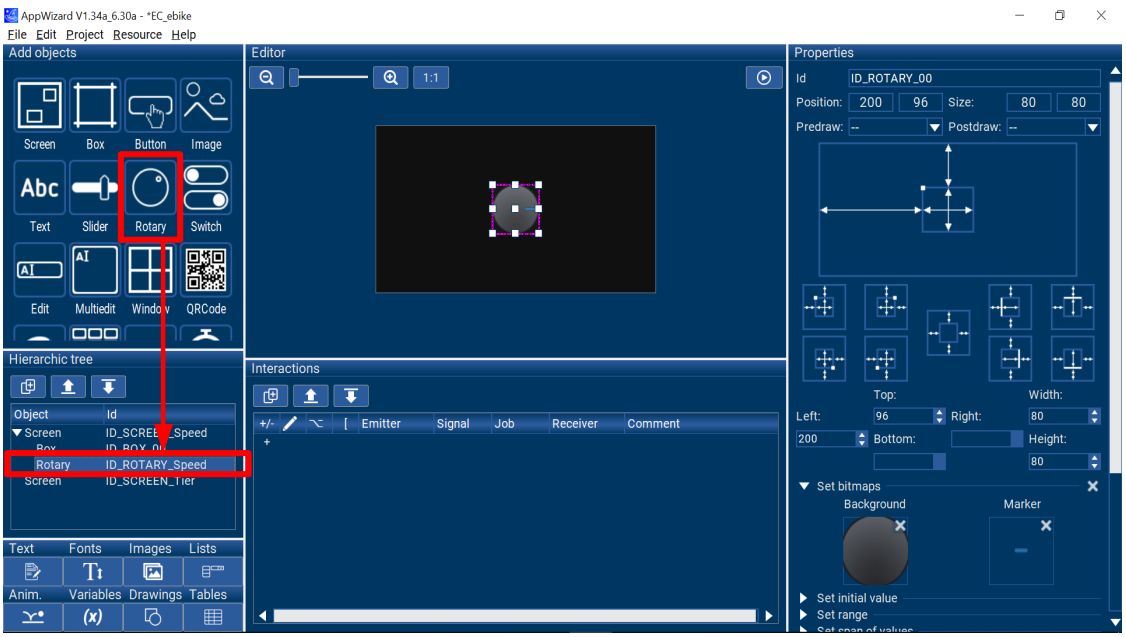


Figure 1-10 Using Rotary Object to Create Speedometer

8. Configure the relevant properties of the Rotary object in the Properties section.

    1) Follow the steps shown in Figure 1-11. Click on "Set bitmaps" and configure the custom image for Background by selecting "Import to project" and choosing the image resources that will be used in the project at once. The images used in this project can be referred to in the directory information of Figure 4-1. Select the PNG image files from the following path: *EC_M467_AppWizard_Ebike_V1.00\EC_ebike\Resource*

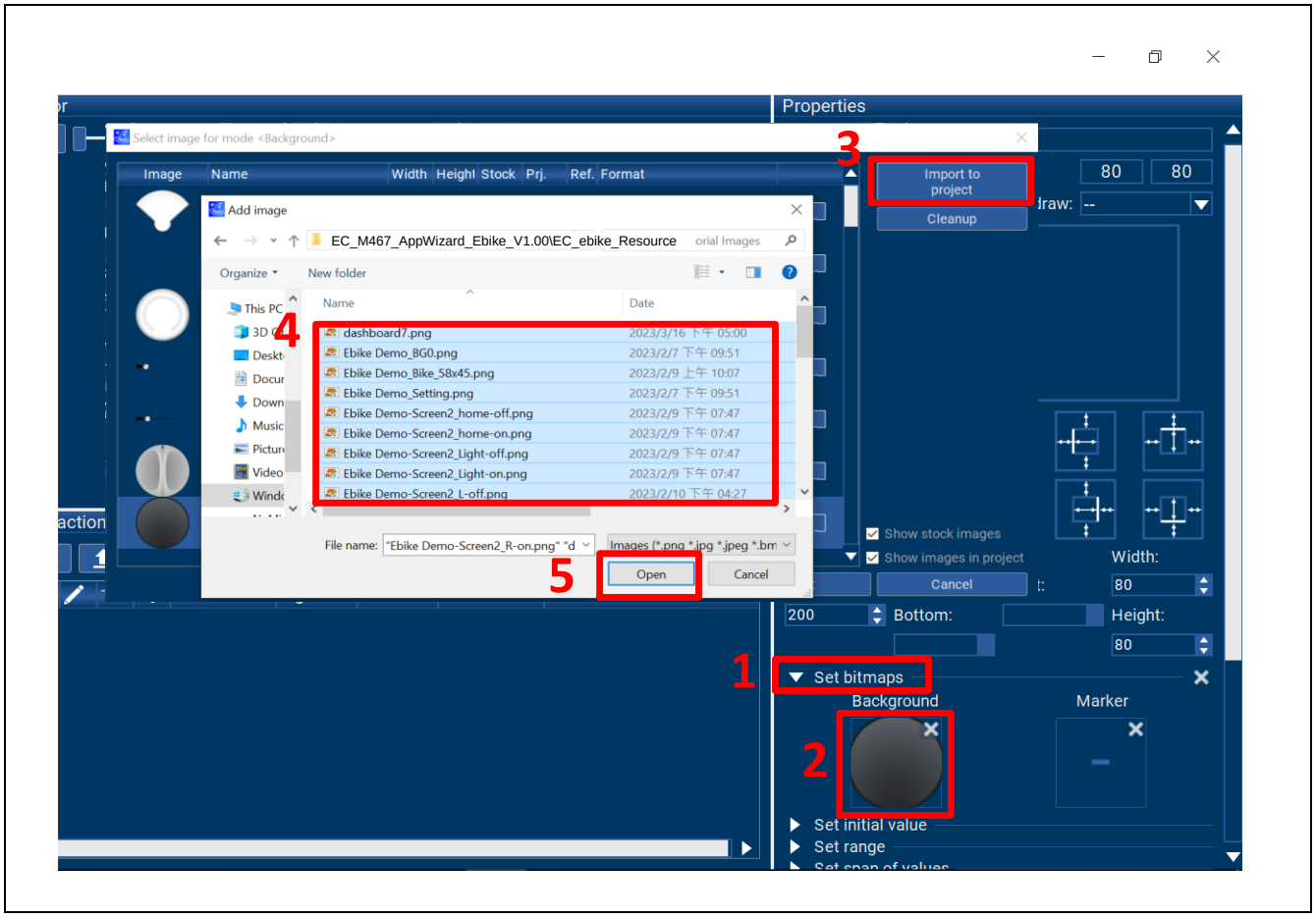


Figure 1-11 Rotary - Properties - Bitmap

2) Follow the steps shown in Figure 1-12. After importing the images, select image  from the available options after importing the images. Click “select”.

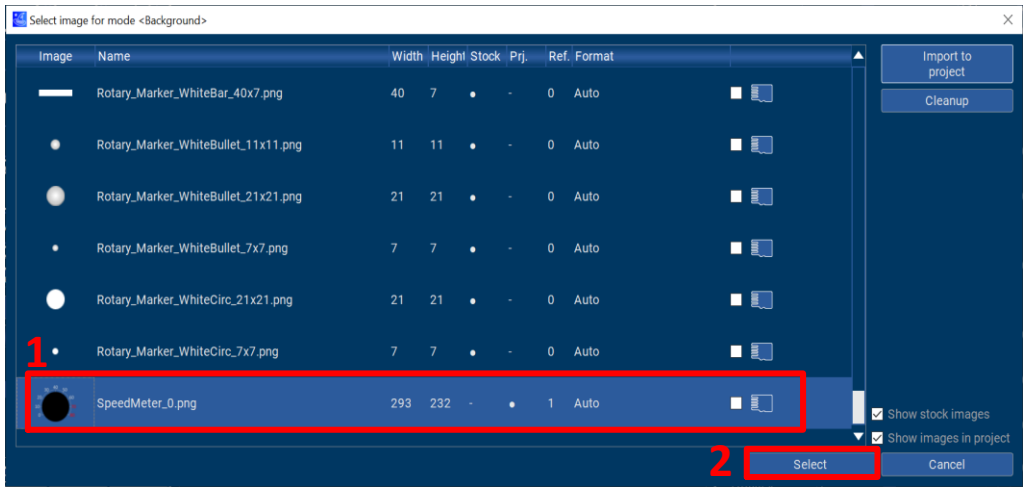


Figure 1-12 Rotary - Properties - Bitmap

3) Follow the steps shown in Figure 1-13. Repeat the same action for the Marker by setting the bitmap and selecting the desired image.  (Rotary_Marker_WhiteBar_20x7.png)

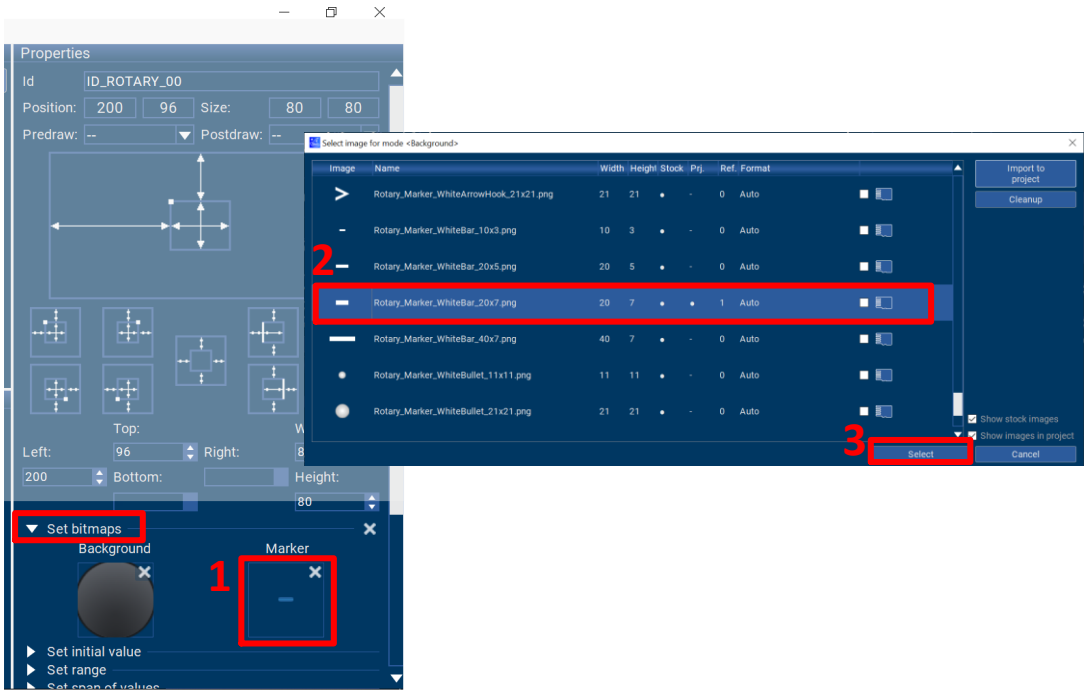


Figure 1-13 Rotary - Properties - Bitmap

4) The system-defined initial angle of 0° is positioned as shown in Figure 1-14. To align with the rotation angle and values of image  , refer to Table 1-3 for instructions on setting the initial value, range, span of values, and offset. The actual configuration items are shown in Figure 1-15.

Figure 1-14 Rotary – Angle Setting

Properties	Description	Setting Value
Initial Value	Starting speed value	0
Span of values	Range of speed value	Min:0 Max:80
Range	Range of rotation angle and direction	Positive:0 Negative:2400 (measured in 10th of degrees)
Offset	Starting angle	2100 (measured in 10th of degrees)

Table 1-3 Rotary – Angle Setting

5) To align the marker's rotation position, refer to Table 1-4 for instructions on setting the marker alignment and radius. The actual configuration items are shown in Figure 1-15.

Properties	Description	Setting Value
Marker alignment	Position of marker	Offset X:1 Offset Y:23 
Radius	Radius of rotary	86

Table 1-4 Rotary – Marker Setting

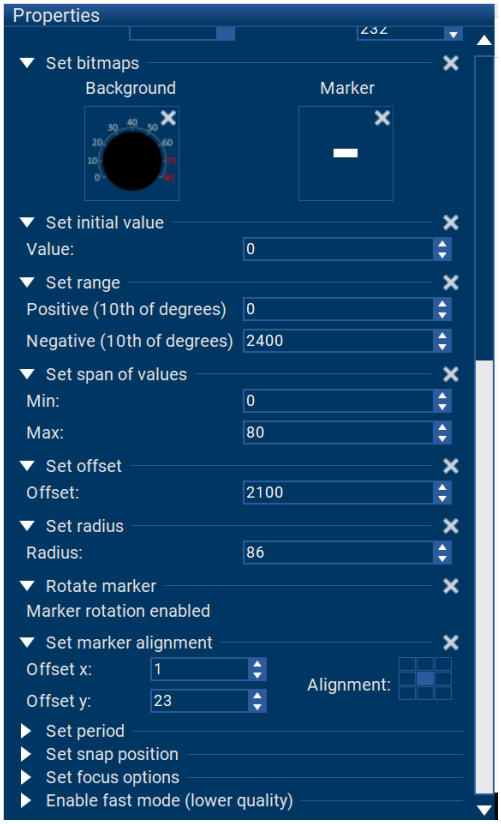


Figure 1-15 Rotary - Properties

6) Use the Rotary object to create the speedometer. Once the configuration is completed, it should appear as shown in Figure 1-16.

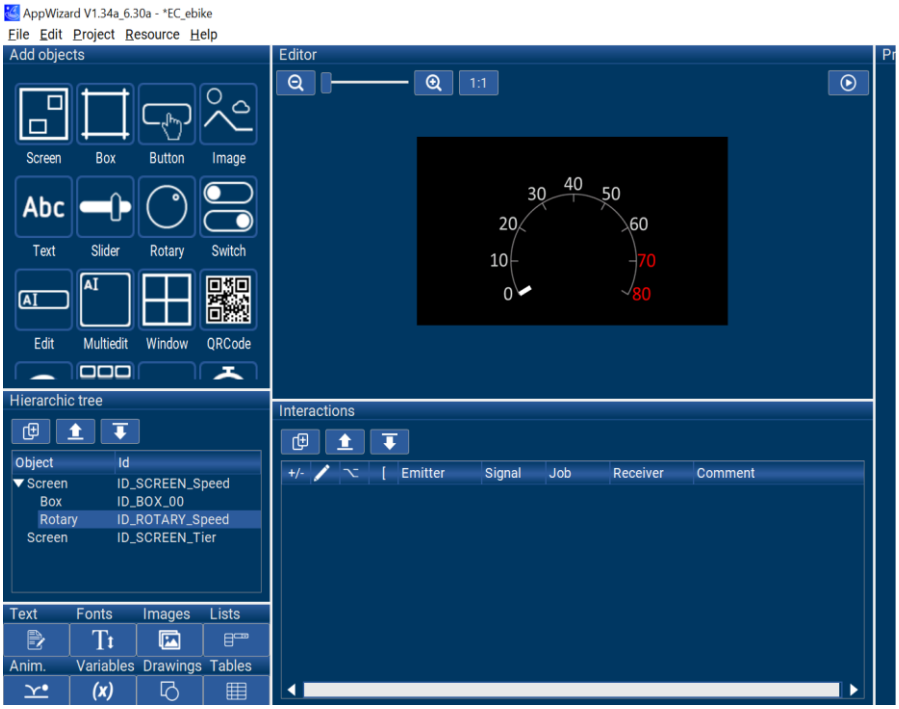


Figure 1-16 Using Rotary Object to Create Speedometer

9. Click on "Text" in "Add objects" to add a numerical display for showing the speed value, and rename it as "ID_TEXT_Speed" in the Hierarchic tree for easier identification.

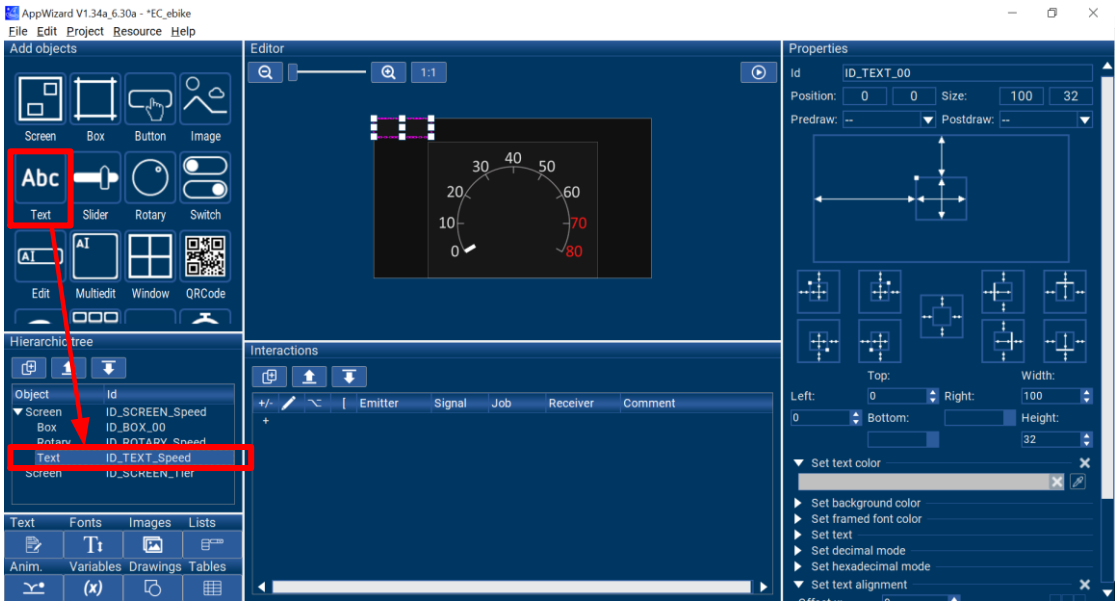


Figure 1-17 Using Text Object to Create Speed Number

10. Configure the relevant properties of the Text object in the Properties section.

    1). Configure the color, decimal mode, text alignment, and value display range for the Text object. The actual configuration items are shown in Figure 1-19.

    2) Set the font for the Text object following the steps shown in Figure 1-18.

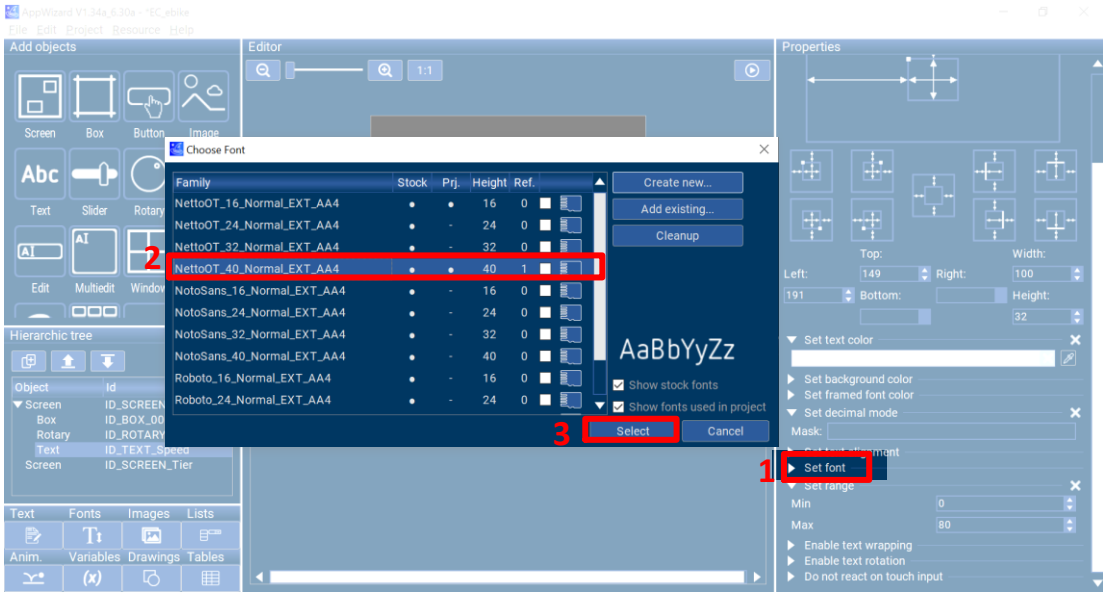


Figure 1-18 Text Object – Properties - Font

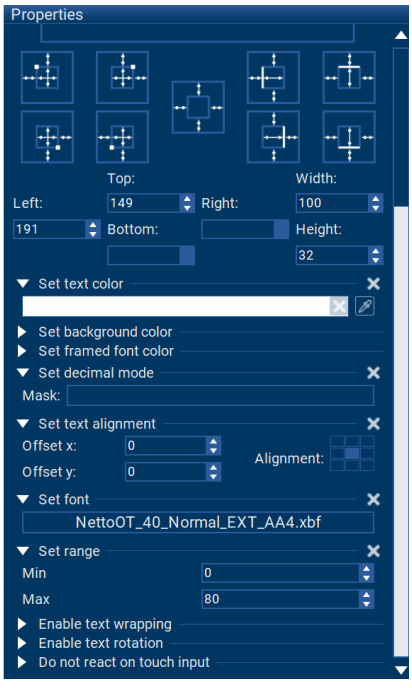


Figure 1-19 Text Object - Properties

3) Use the Text object to display the speed value. Once the configuration is completed, it should appear as shown in Figure 1-20.

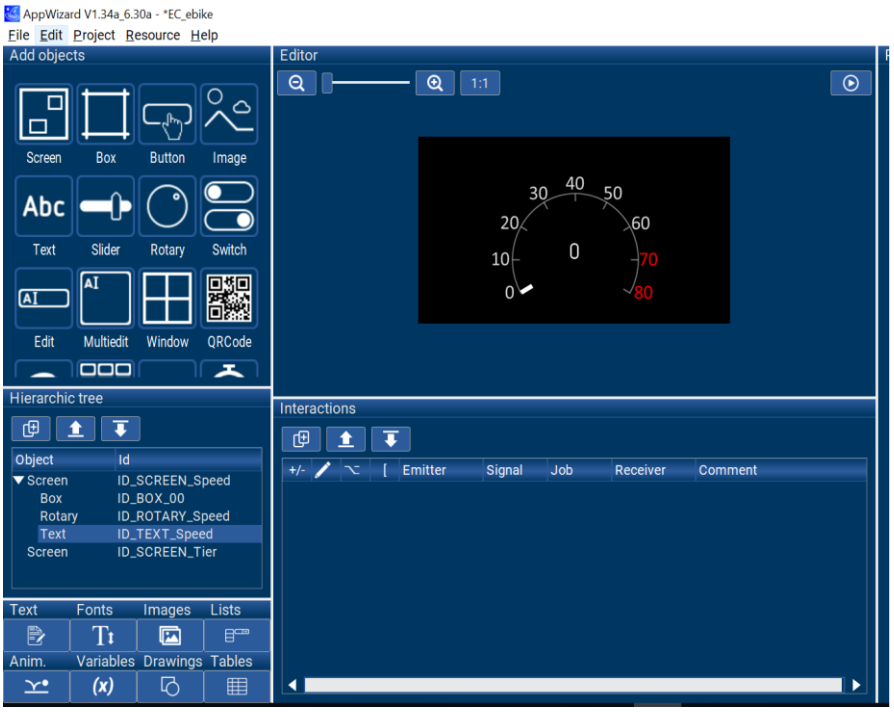


Figure 1-20 Using Text Object to Display Speed Number

11. Click on "Text" in "Add objects" to add text for displaying value names, units, titles, etc.,

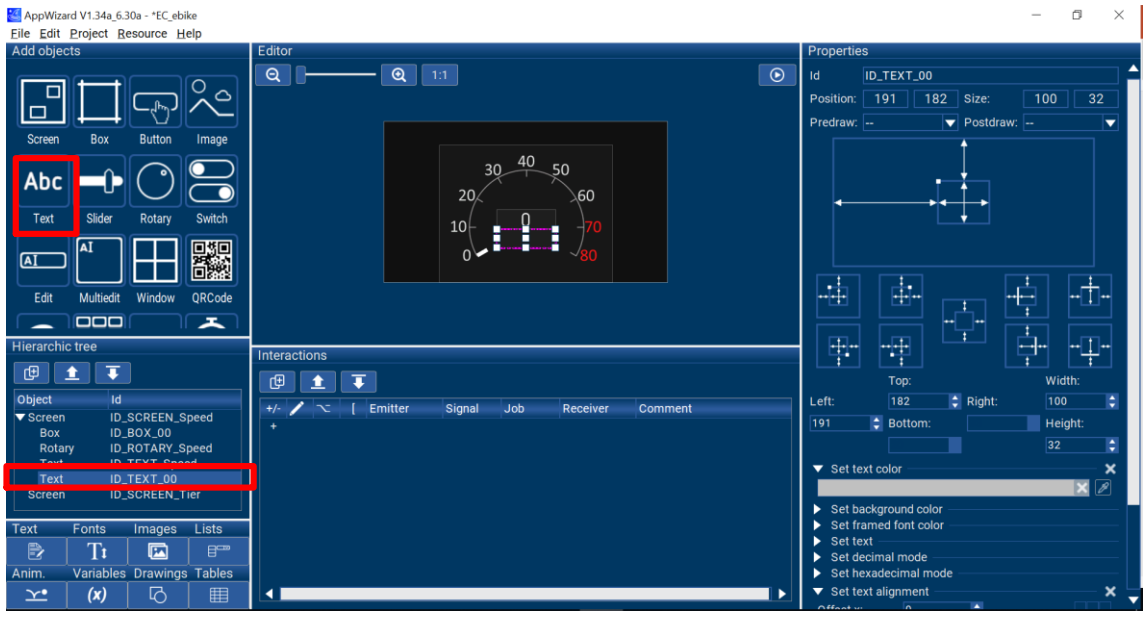


Figure 1-21 Using Text Object to Display Speed Unit

1) Set the color and text alignment for the Text object.

2) Follow the steps shown in Figure 1-22 to Figure 1-26 to configure the displayed text and show both Chinese and English languages simultaneously.

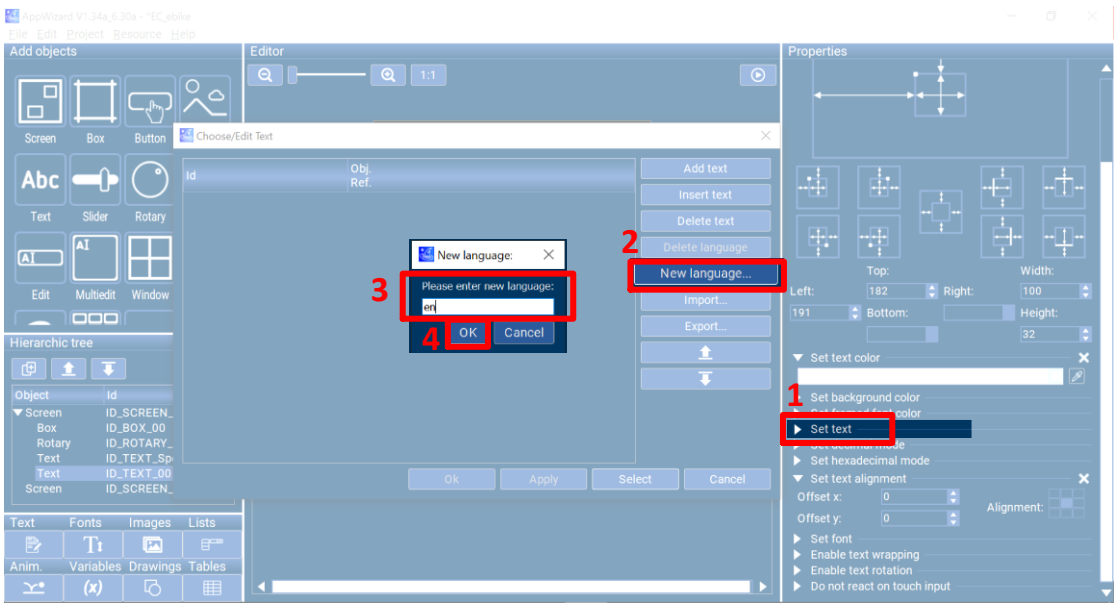


Figure 1-22 Text Object – Properties – Create 1st Text Language (English)

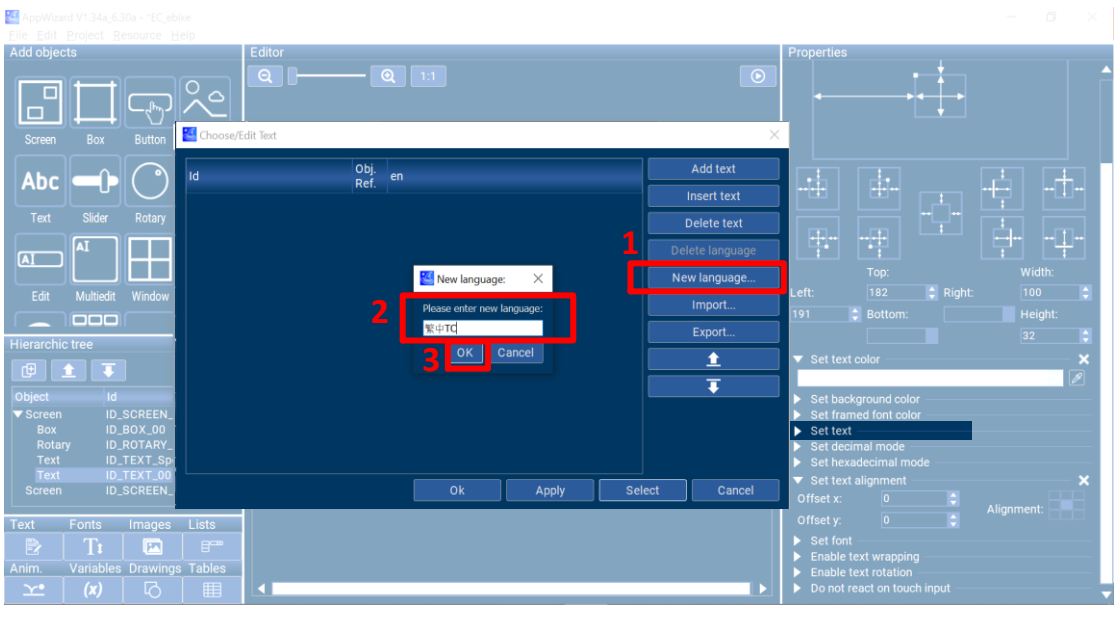


Figure 1-23 Text Object – Properties – Create 2nd Text Language (Chinese)

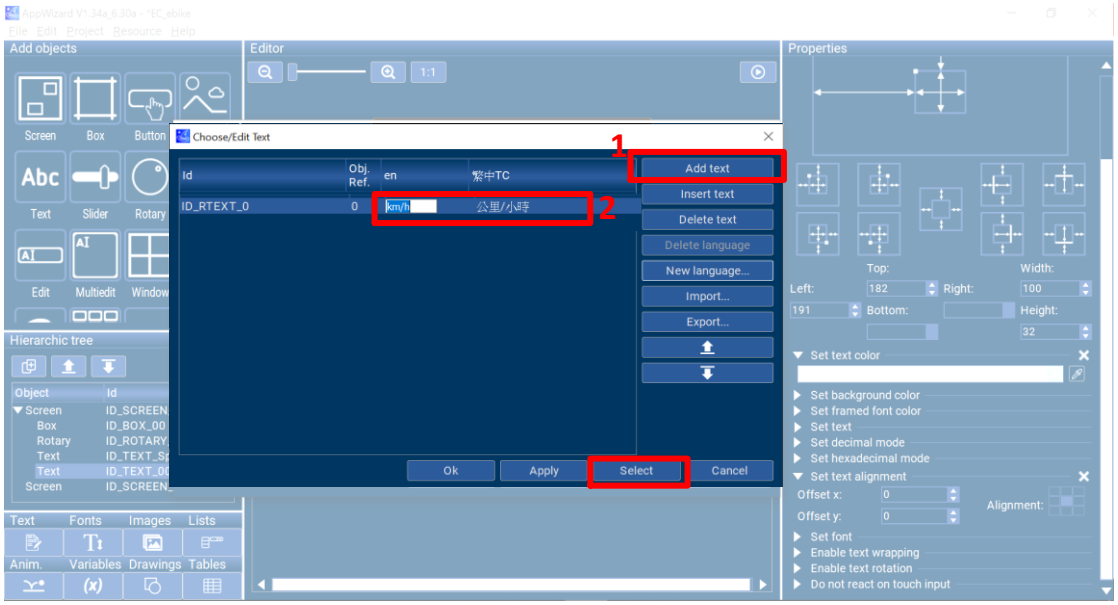


Figure 1-24 Text Object – Properties – Edit Text Contents for 2 Languages Respectively

3) Add and select a font that can display both English and Chinese simultaneously.

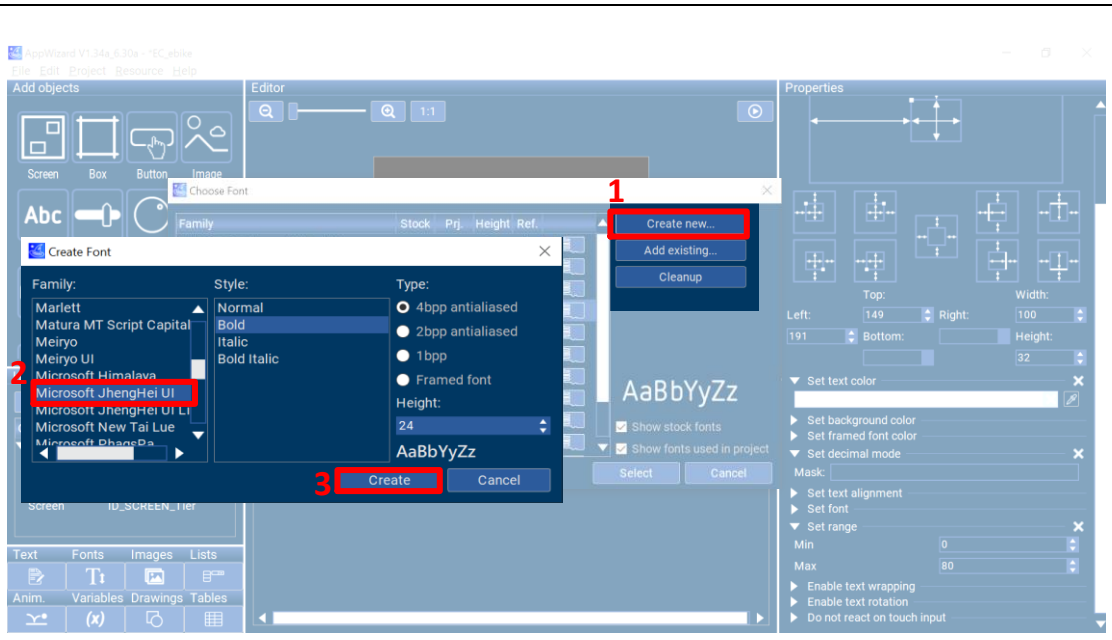


Figure 1-25 Text Object – Properties - Font

4) Set the codepoint range for the font. When the "Define codepoints to be used" window appears, select "Project text only" to choose only the character range that will be used in the project.

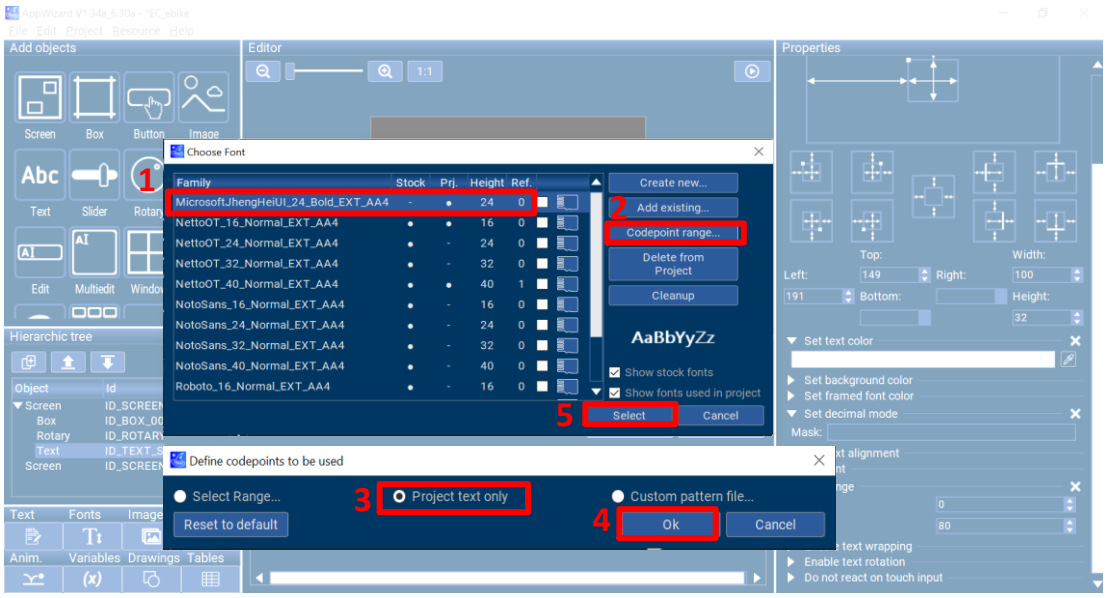


Figure 1-26 Text Object – Properties – Font and Codepoint Range

5) Use the Text object to display English and Chinese text. Once the configuration is complete, it should appear as shown in Figure 1-27.

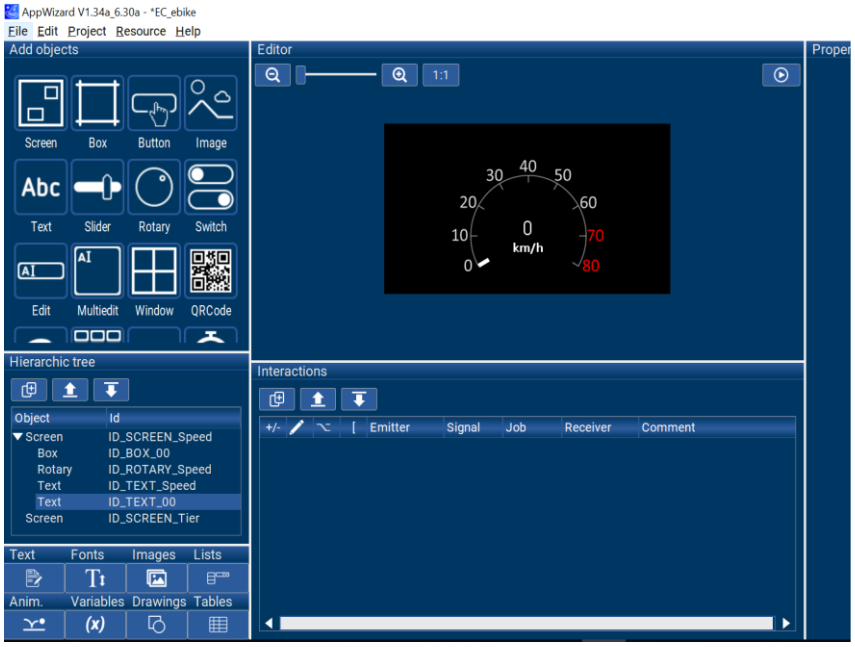


Figure 1-27 Using Text Object to Display Speed Unit

12. Use the Image object to add a separator line to the screen. Click on "Image" in "Add objects."

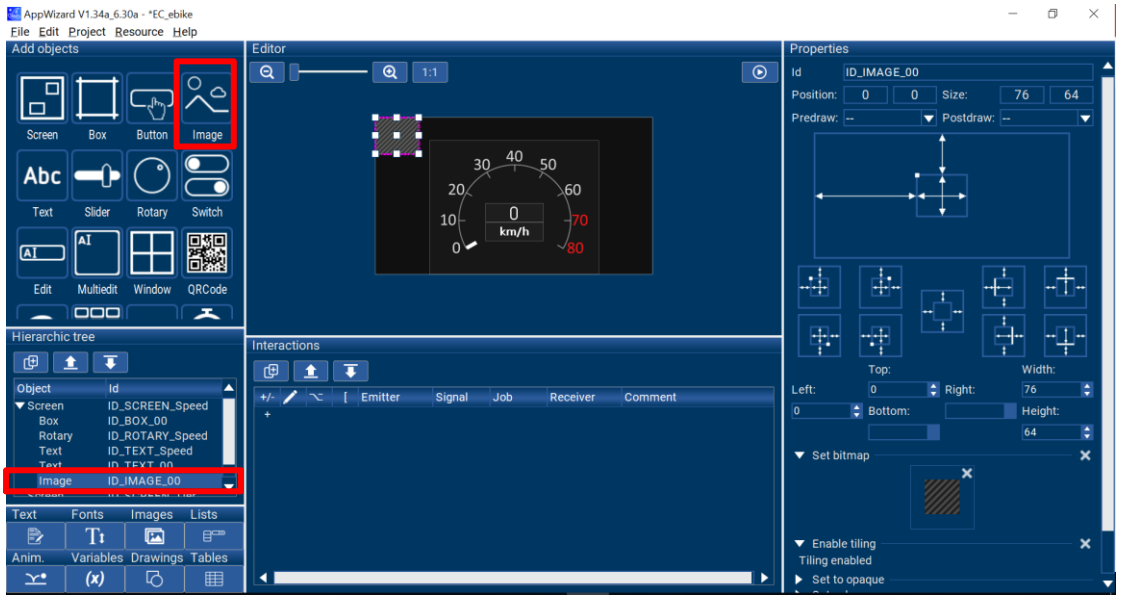


Figure 1-28 Using Image Object to Create a Screen Divider

13. Configure the relevant properties of the Image object in the Properties section.

    1) Follow the steps shown in Figure 1-29. Click on "Set bitmap" and configure the custom image. Disable the "**Enable tiling**" option to prevent infinite extension.

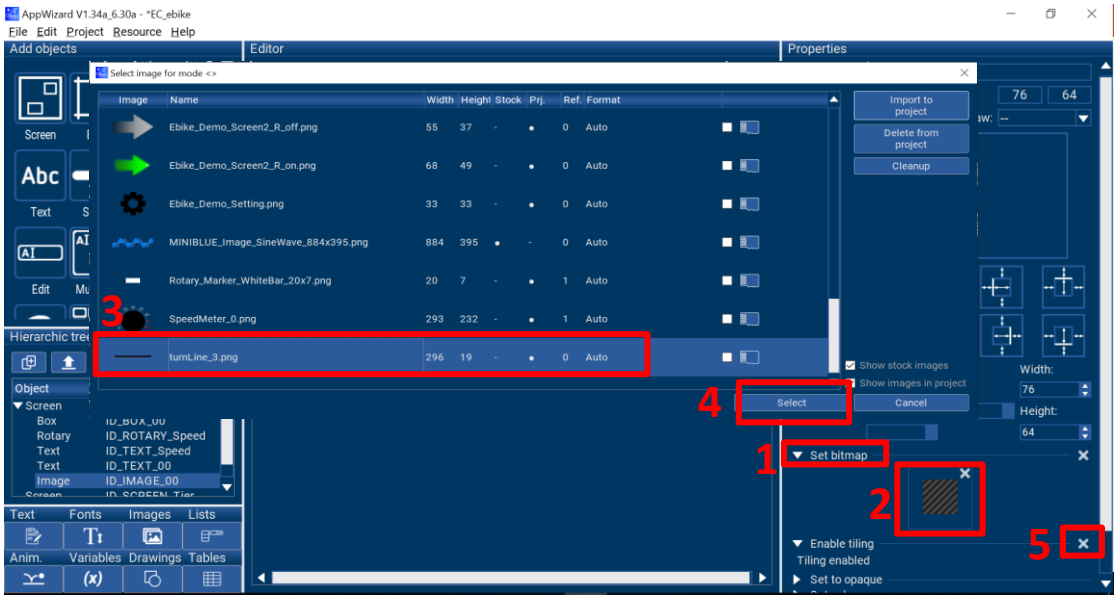


Figure 1-29 Image - Properties - Bitmap

2) Once the configuration is complete, it should appear as shown in Figure 1-30.

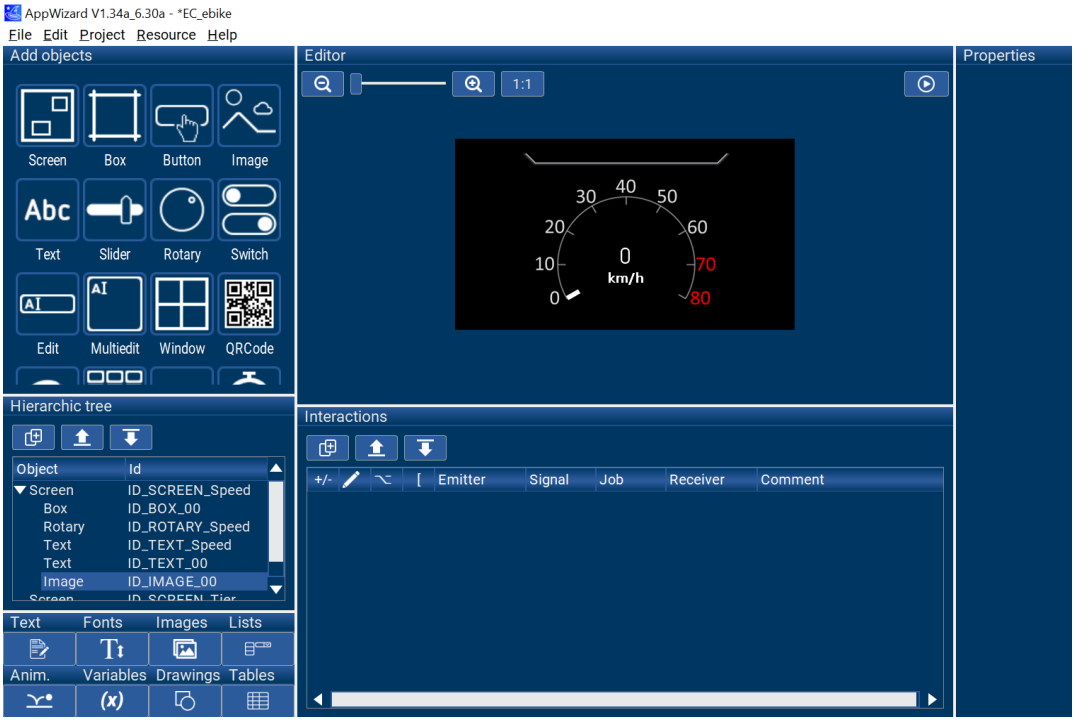


Figure 1-30 Using Image Object to Create A Screen Divider

14. Use the Button object to represent the turn signal functionality. Click on "Button" in "Add objects," and rename it as "ID_BUTTON_Left" in the Hierarchic tree for easier identification.

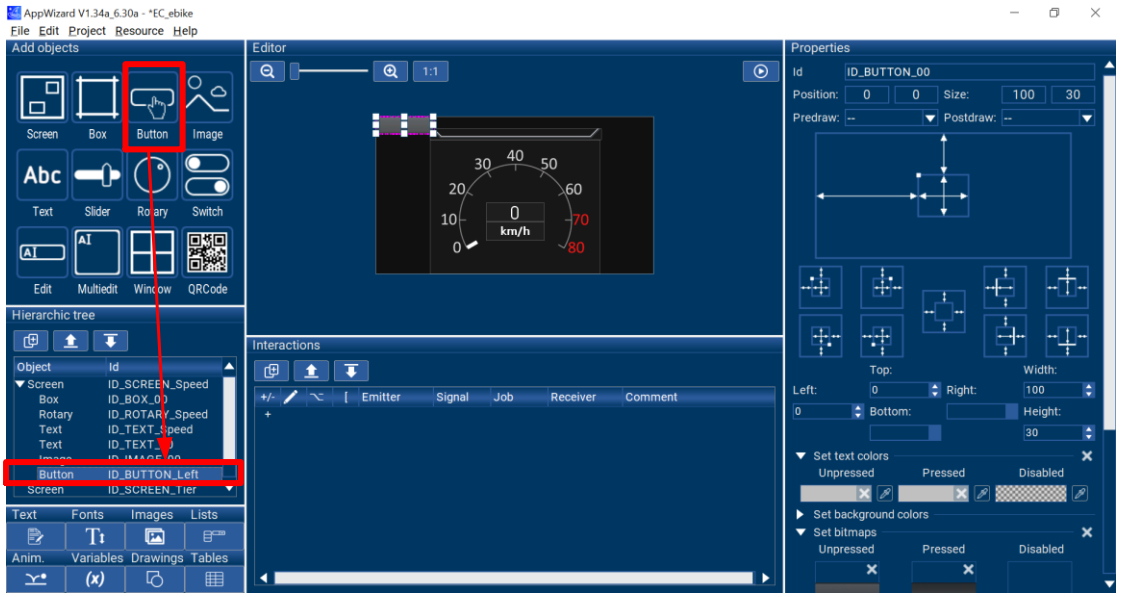
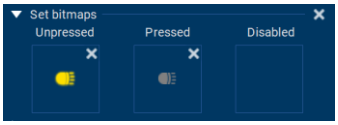


Figure 1-31 Using Button Object to Create Turn Signal

15. In the Properties section, configure the relevant properties of the Button widget.

    1) In this example, the functionality will be achieved by setting different bitmaps for different button states such as "Unpressed" and "Pressed." The selection of bitmaps can be done following the step 13. For this example, you need to set corresponding images for the left/right turn signal, headlights and warning lights.

BUTTON ID	Bitmap
ID_BUTTON_Left	
ID_BUTTON_Right	
ID_BUTTON_Light	

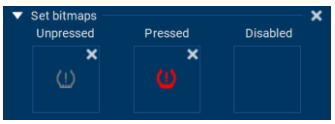
ID_BUTTON_Warn	

Table 1-5 Bitmaps

2) Adjust the positions accordingly as shown in Figure 1-32 once the settings are completed.

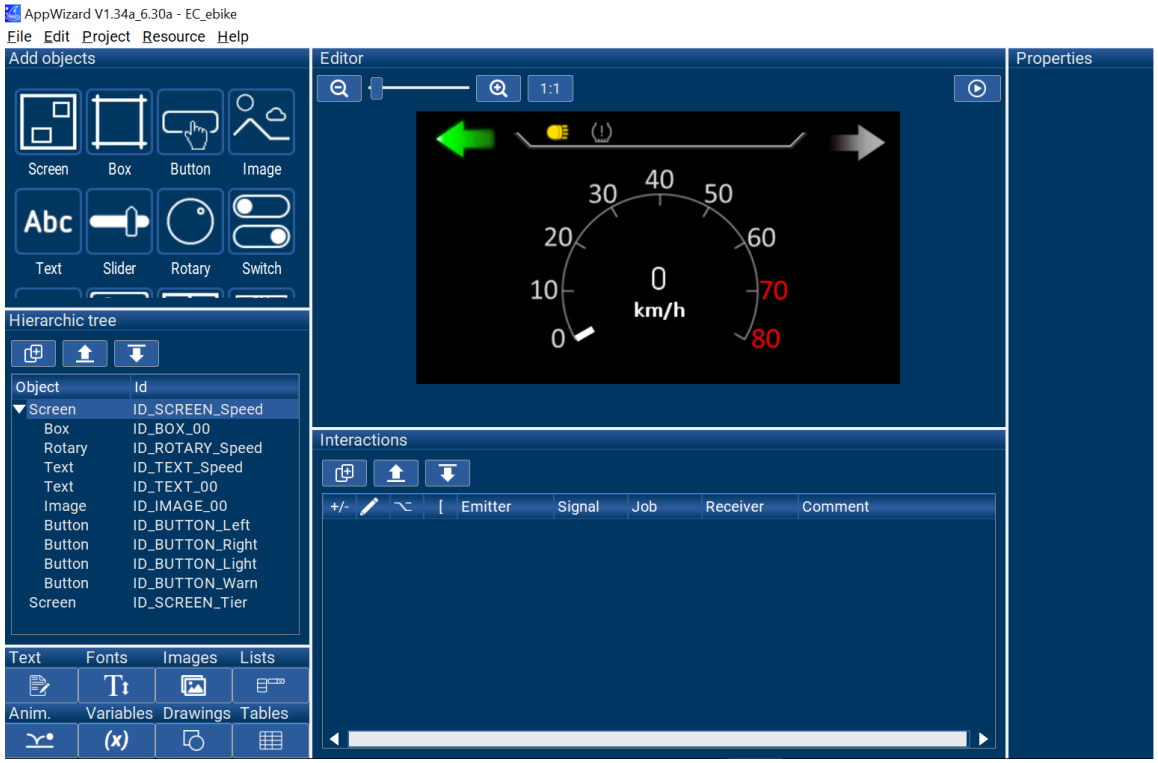


Figure 1-32 Using Button Object to Create Headlight, Warning Light, and Turn Signal

16. Add a "Go to Settings Page" button. Unlike step 14, which used a bitmap as the button interface, this button will not use any image resources. Click on "Button" in "Add objects" and rename it as "ID_BUTTON_Setting" in the Hierarchic tree for easier identification.

    1) Set the text color, background color, remove the bitmap setting, and configure the text and font as shown in Figure 1-33.

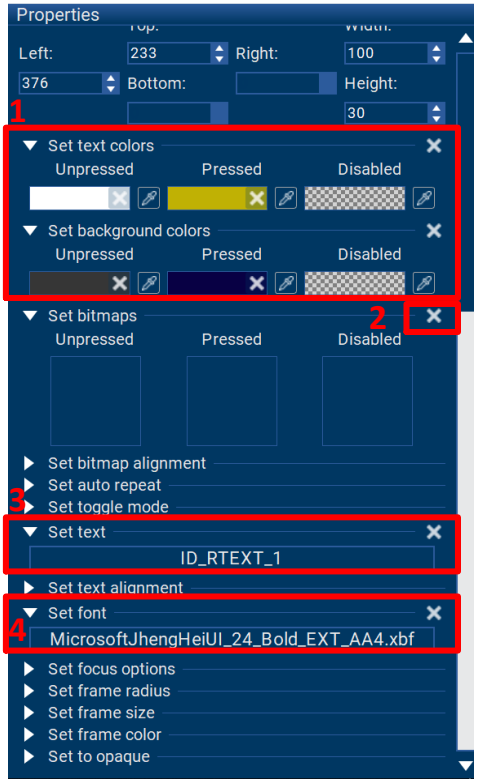


Figure 1-33 Button - Properties

17. Adjust the positions accordingly as shown in Figure 1-34 once the settings are completed.

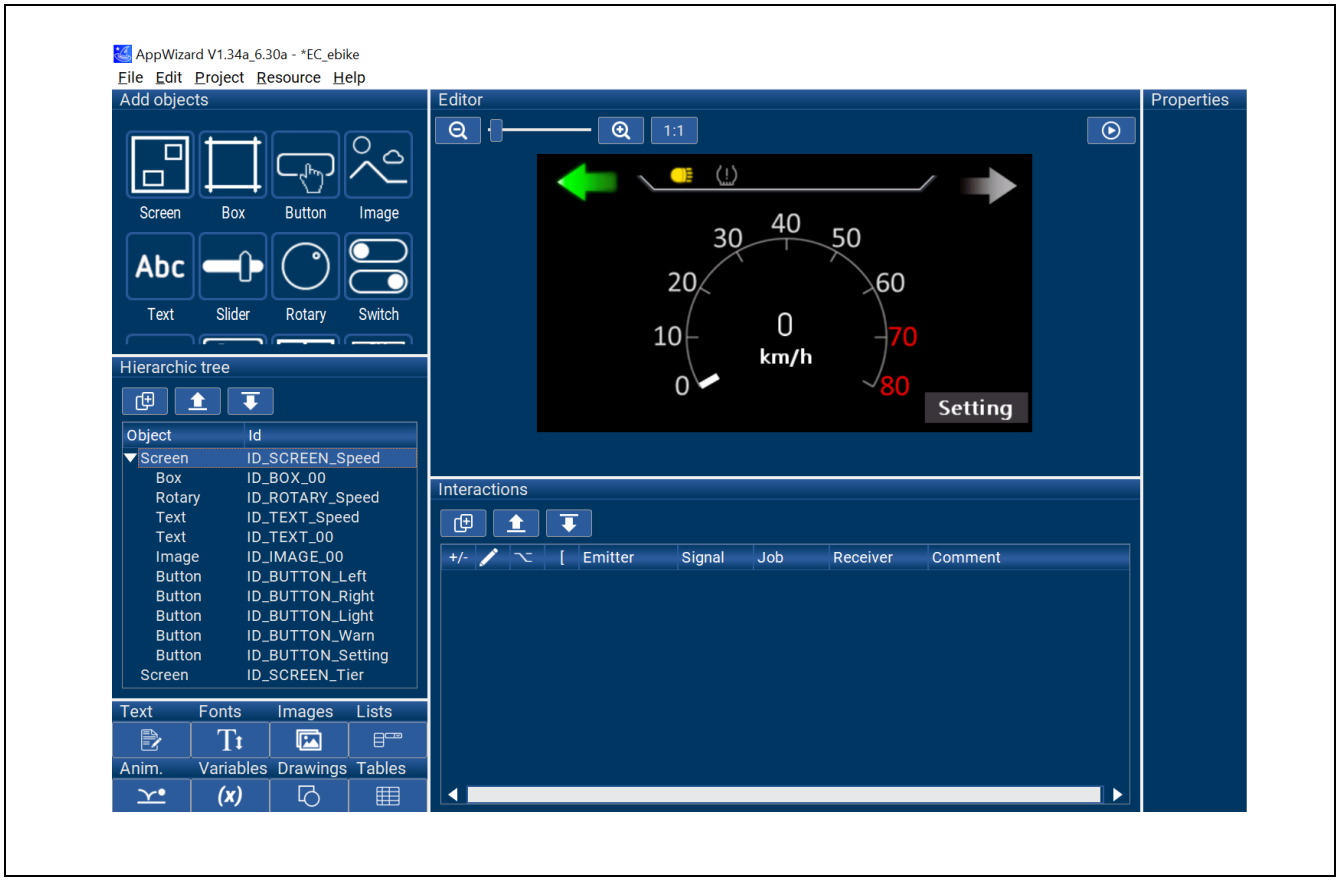


Figure 1-34 Using Button Object to Implement Screen Switching

18. Select the “Progbar” object from “Add objects” to use it as a battery level bar. Rename it as ID_Progbar_Battery in the Hierarchic tree for easier identification.

Figure 1-35 Using Progbar Object to Implement Battery Level Bar

19. Configure the relevant properties of the Progbar object in the Properties section.

    1) Follow the order and settings shown in Figure 1-36 to set the initial value, value range, color, disable bitmap, disable tiling, set the radius, set the frame size, and set the frame color.

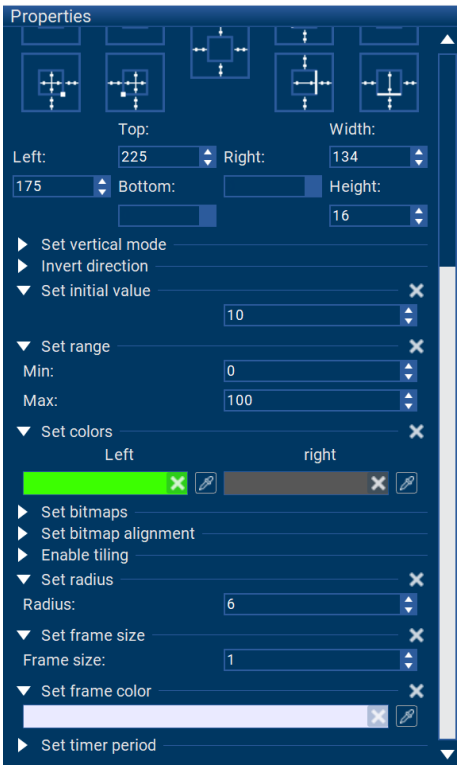


Figure 1-36 Progbar - Properties

2) Adjust the positions accordingly as shown in Figure 1-37 once the settings are completed.

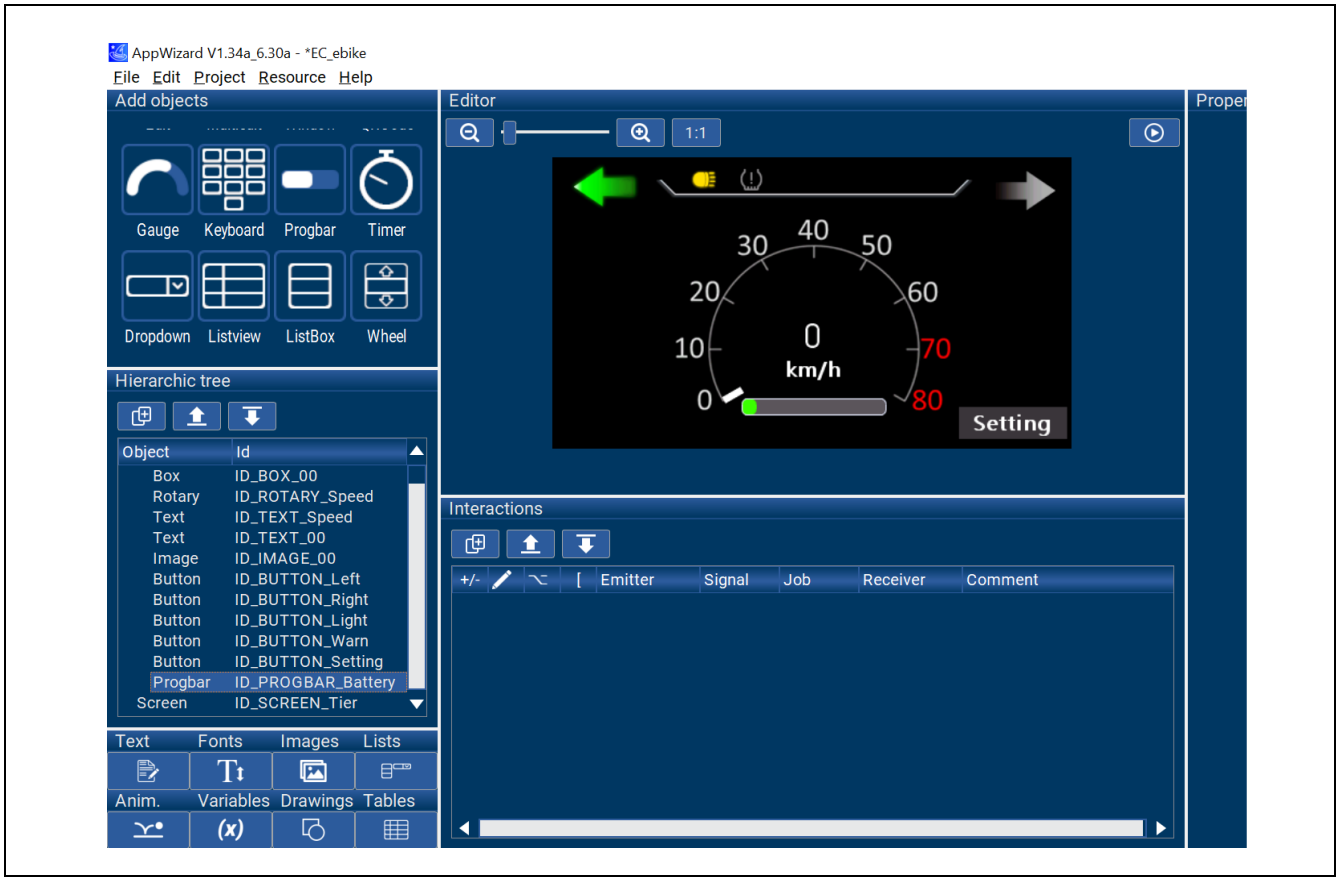


Figure 1-37 Using Progbar Object to Implement Battery Level Bar

20. Use Text object to add battery description text and unit. Please follow the step 11 to set up. For the related settings in this example, refer to Figure 1-38. After completion, it should look like Figure 1-39.

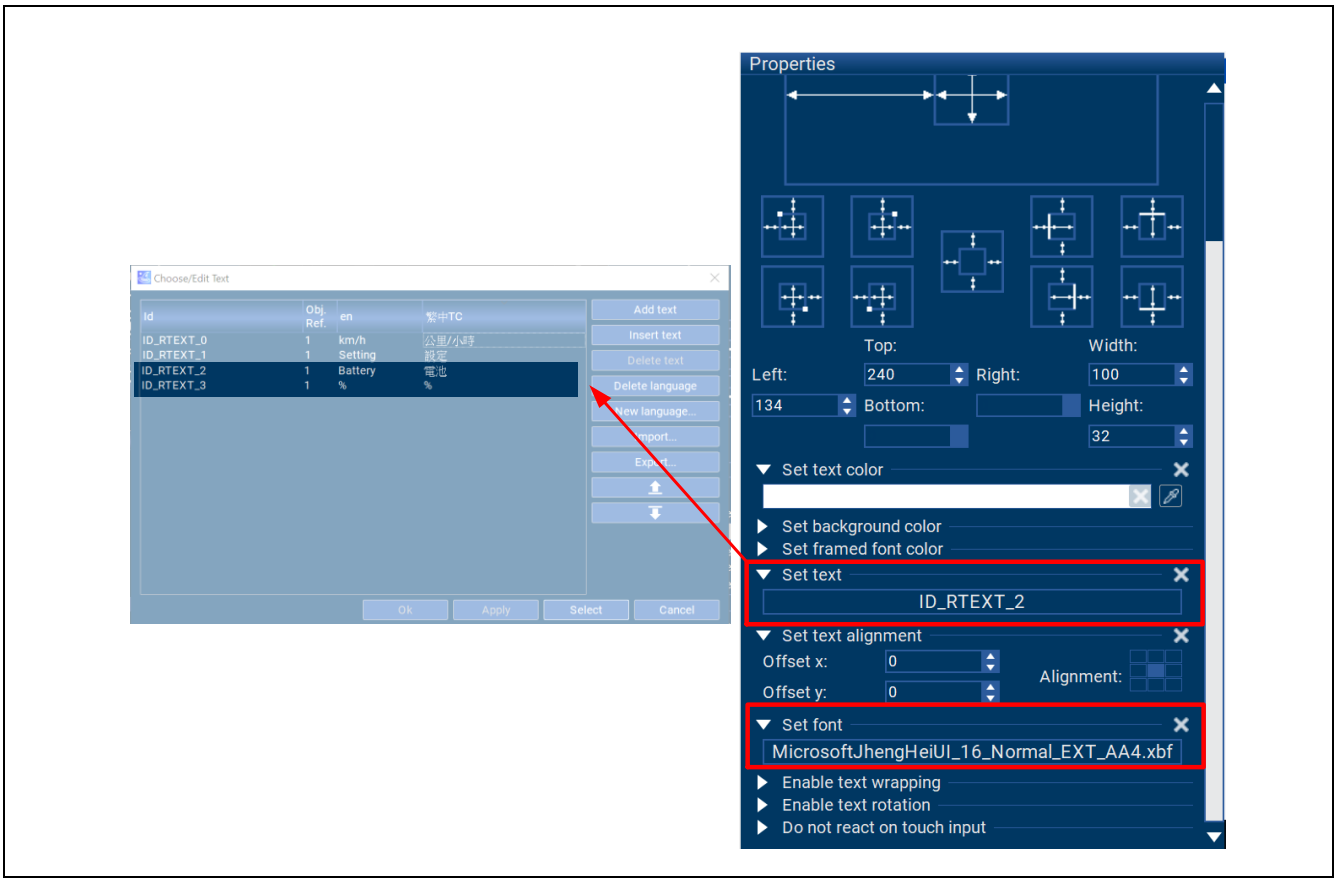


Figure 1-38 Text Object - Properties

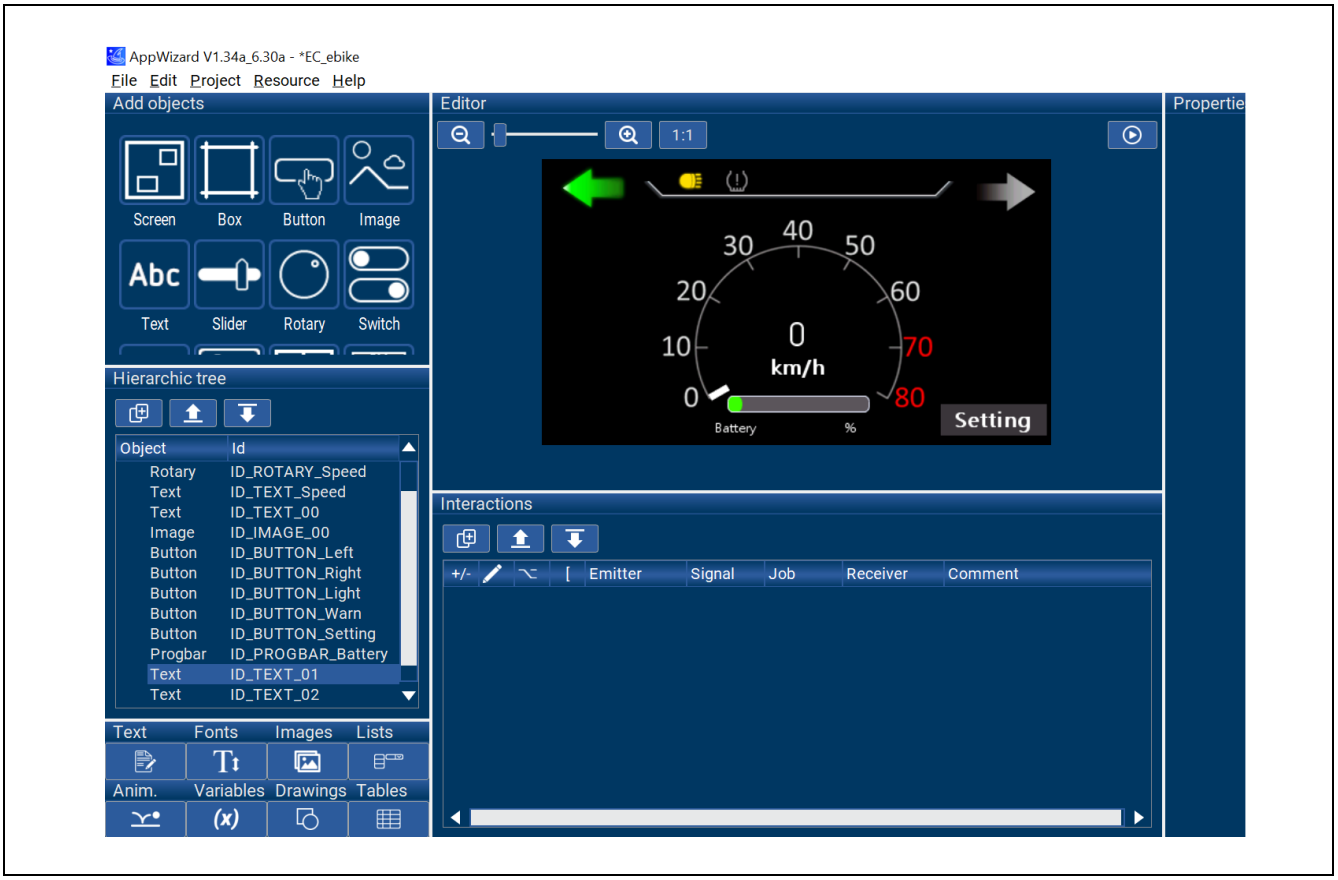


Figure 1-39 Using Text Object to Display Battery Name and Unit

21. Use Text object to add battery level value display. Please follow the step 9 to 10 to set up. For the related settings in this example, refer to Figure 1-40. After completion, it should look like Figure 1-41.

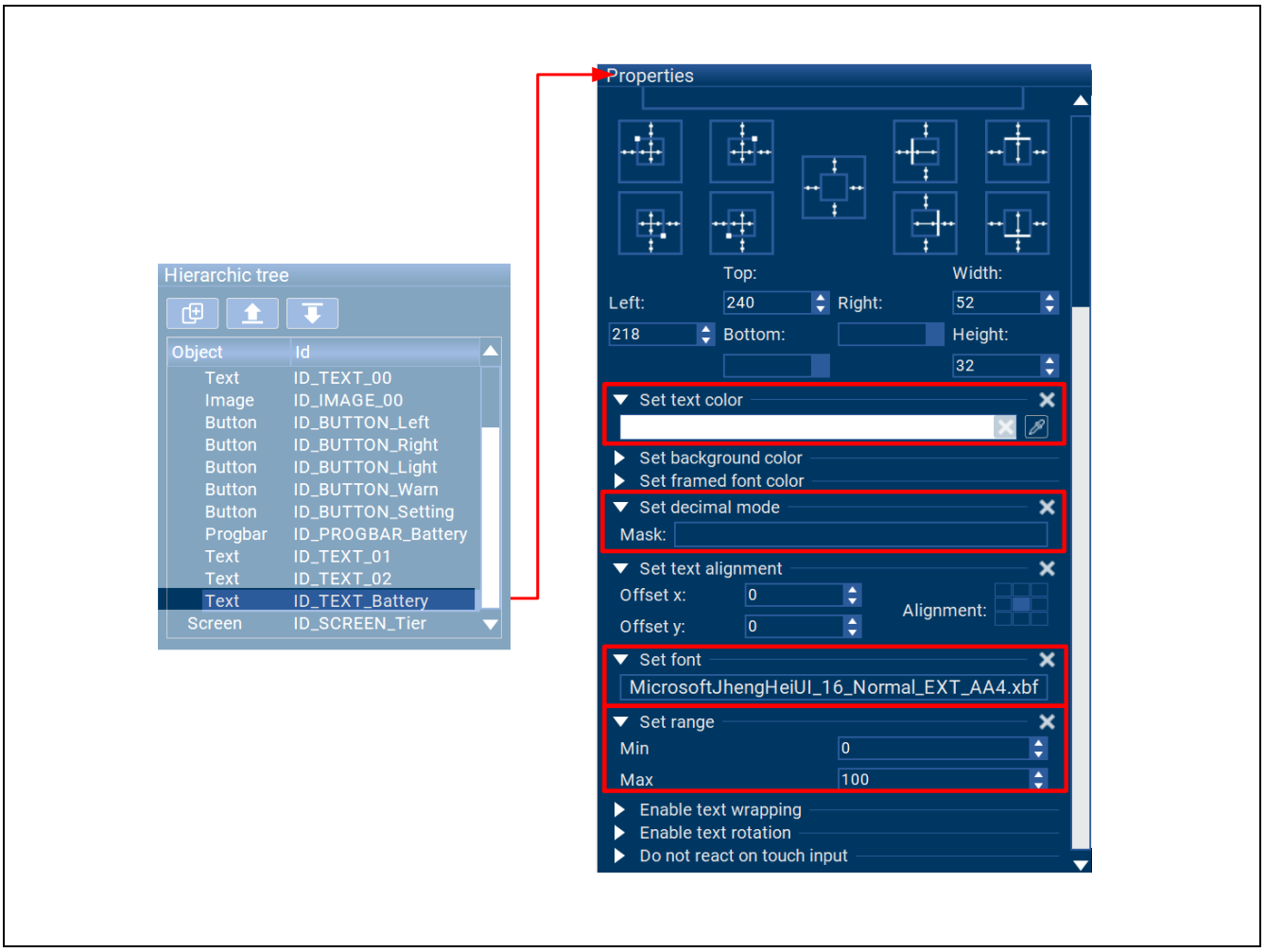


Figure 1-40 Text Object - Properties

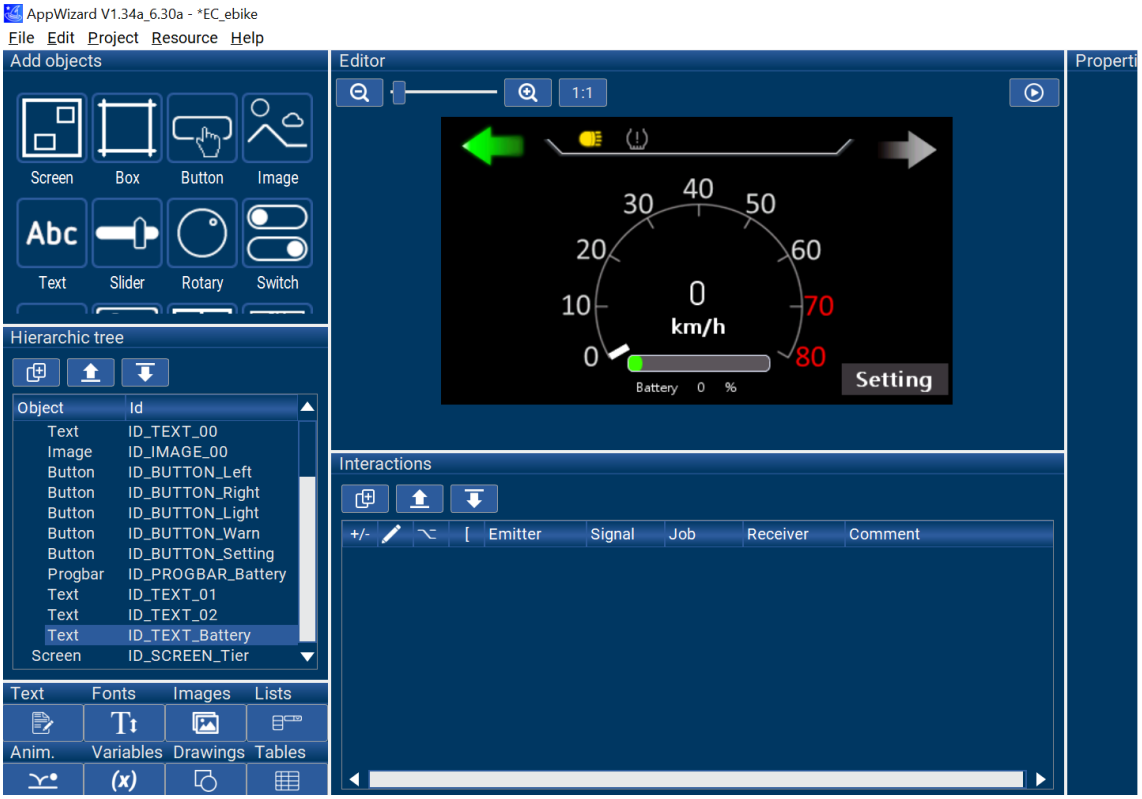


Figure 1-41 Using Text Object to Display Battery Number

22. Start editing the tire pressure page by selecting “ID_SCREEN_Tire” in the Hierarchic tree.

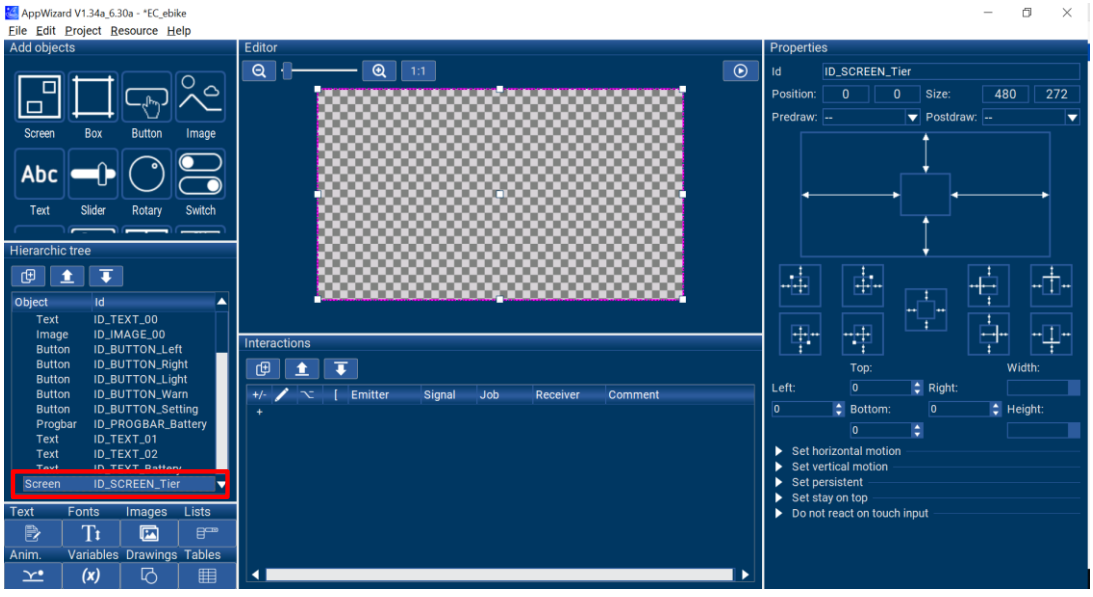


Figure 1-42 Click ID_SCREEN_TIRE to Edit The Tire Screen

23. Use the Image object to import an image with the size of 480x272 as the background. Click "Image" in the “Add objects” menu.

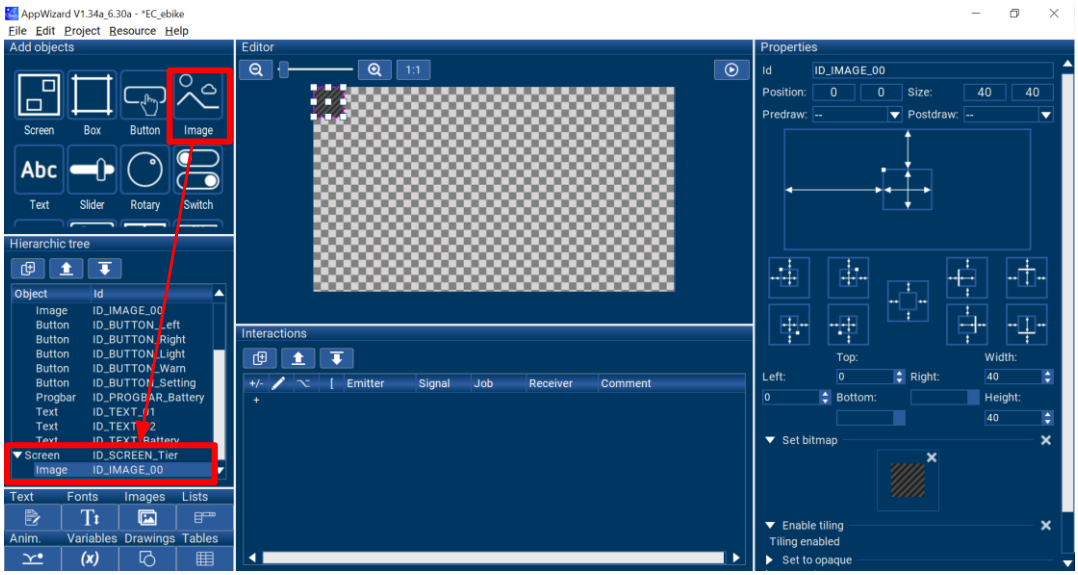


Figure 1-43 Using Image Object to Import Background Image

24. Follow step 12 to 13 to select the background image. After completing the steps, the result should be shown as Figure 1-44.

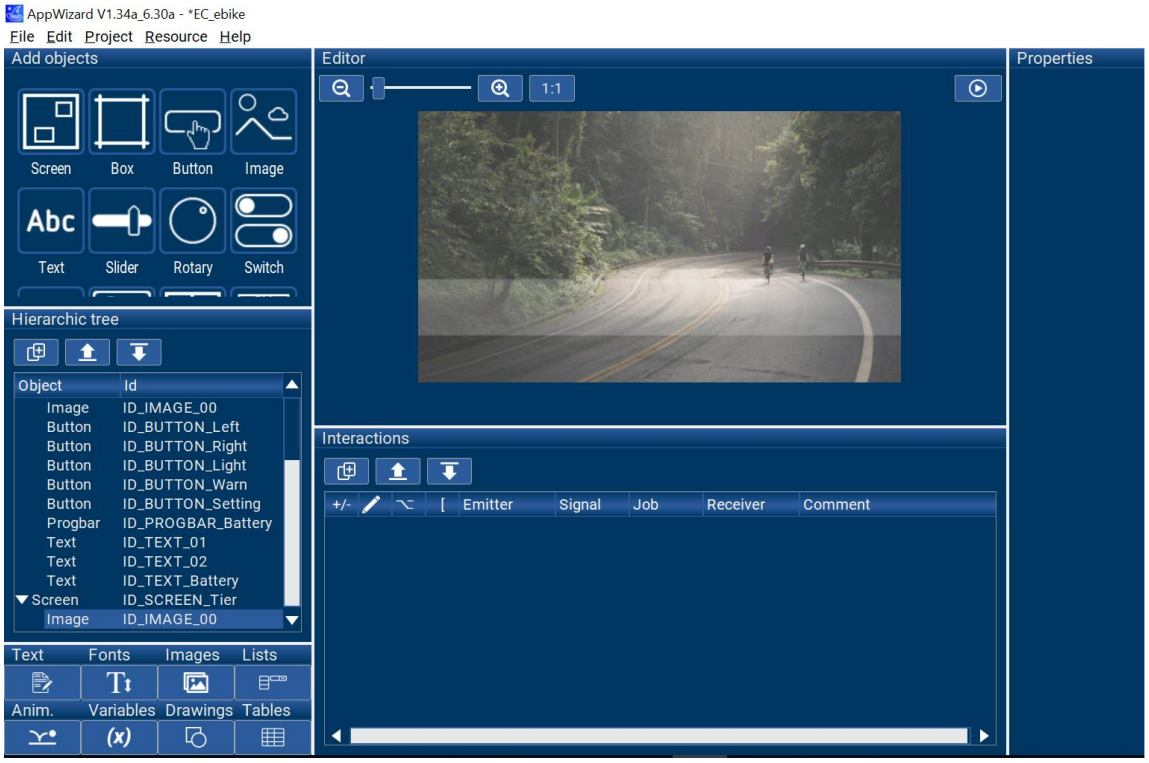


Figure 1-44 Using Image Object to Import Background Image

25. Follow step 12 to 13 to select the bike image. After completing the steps, the result should be shown as Figure 1-45.

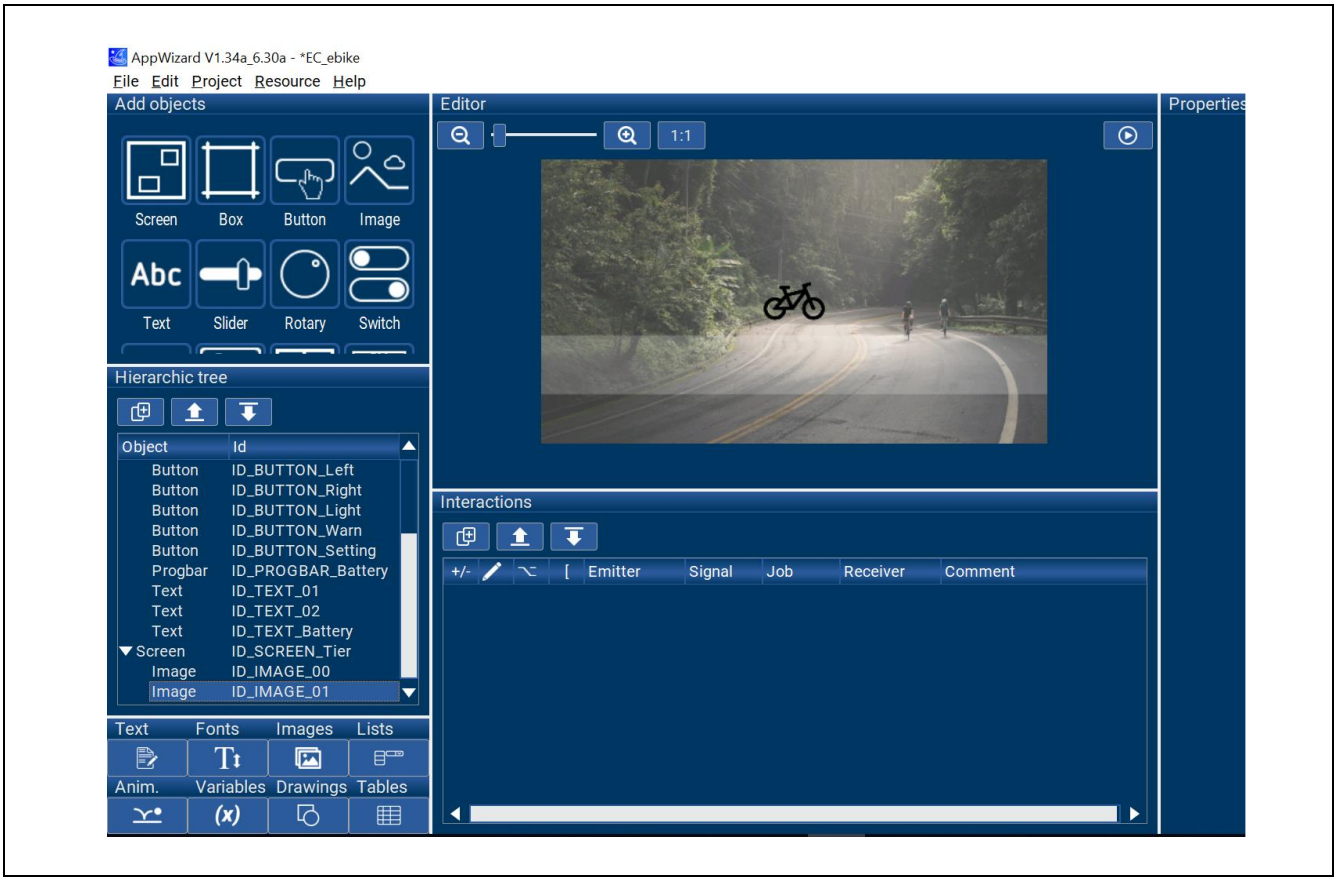


Figure 1-45 Using Image Object to Import Bike Image

26. Use Text object to add a title, tire pressure text, and unit. Follow the step 11. The settings for this example can be referenced in Figure 1-46. After completion, the result should be shown as Figure 1-47. The tire pressure unit, such as psi, kPa, bar, etc., can also be displayed on the screen using the same method.

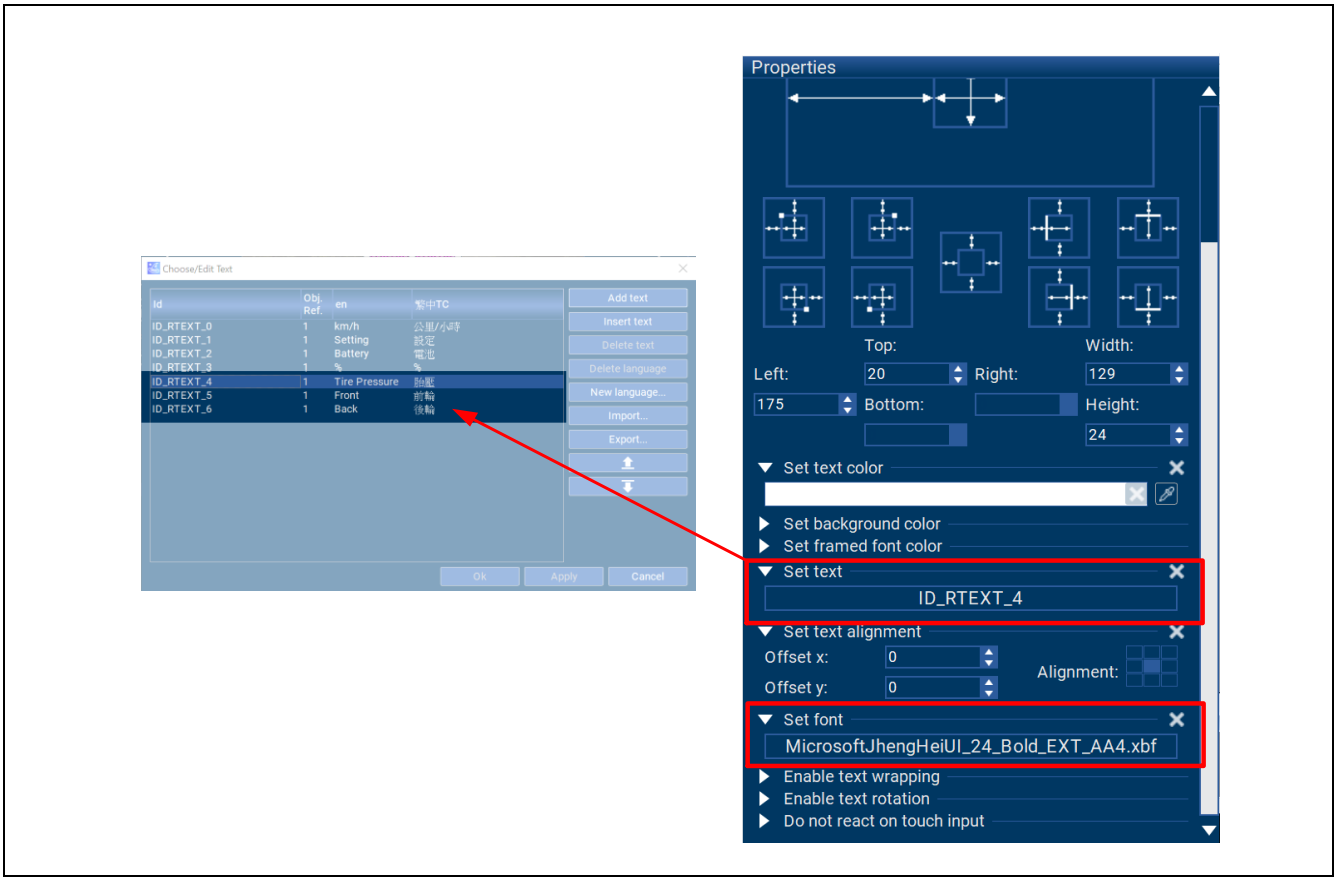


Figure 1-46 Text Object - Properties

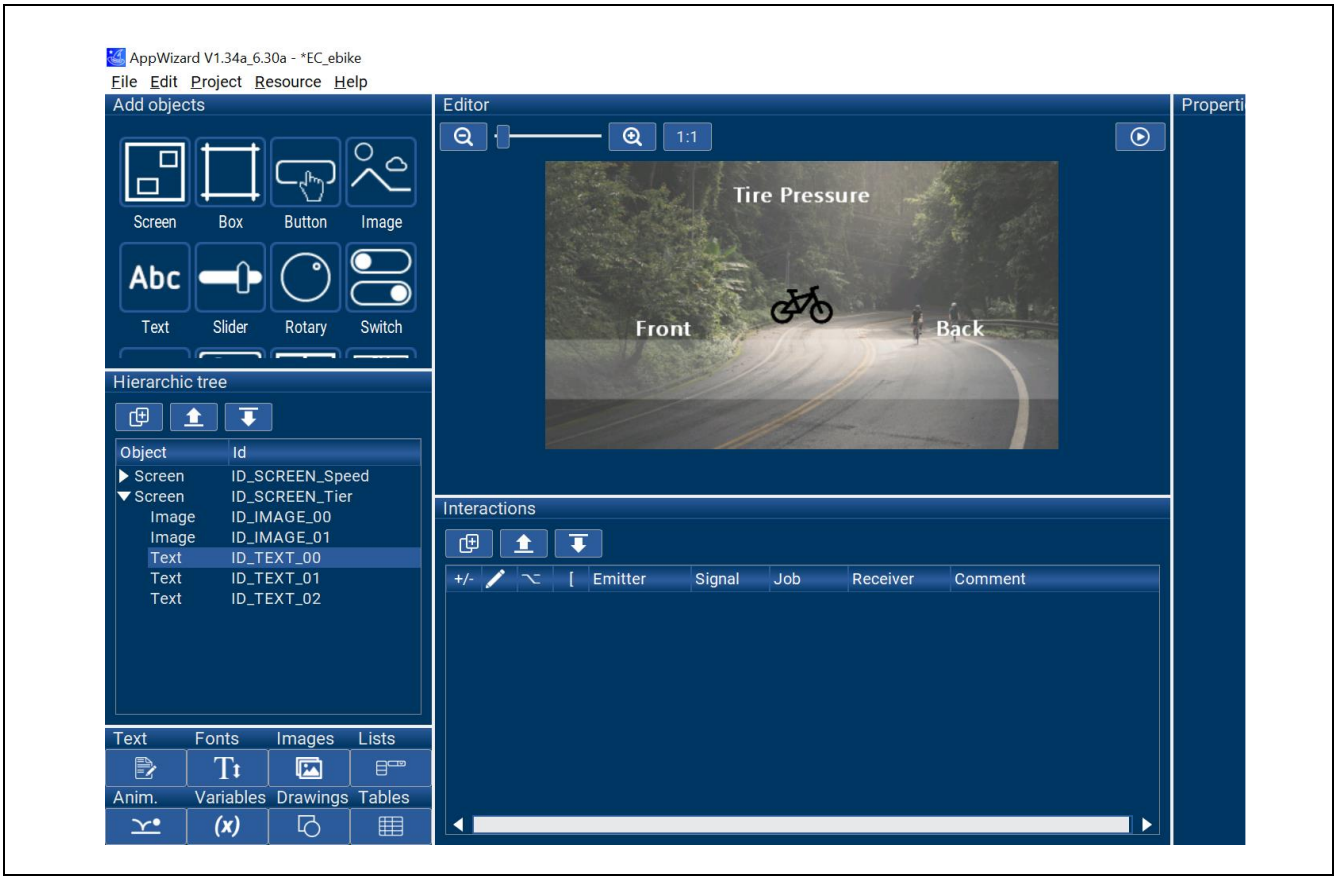


Figure 1-47 Using Text Object to Display Tire Name

27. Use Text object to add tire pressure value display. Follow the step 9 to 10 to complete this task. The settings for this example can be referenced in Figure 1-48. After completion, the result should be shown as Figure 1-49.

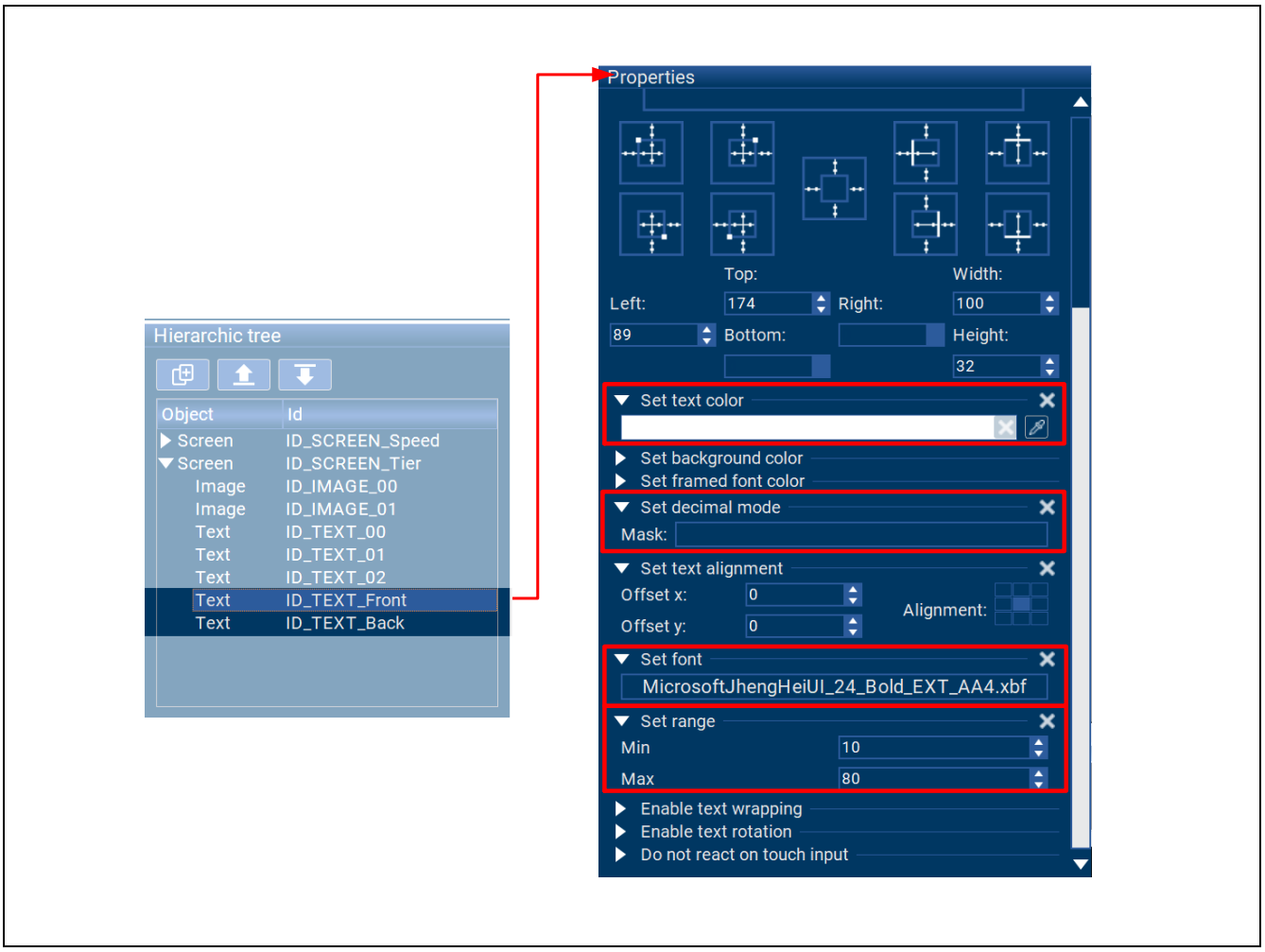


Figure 1-48 Text Object - Properties

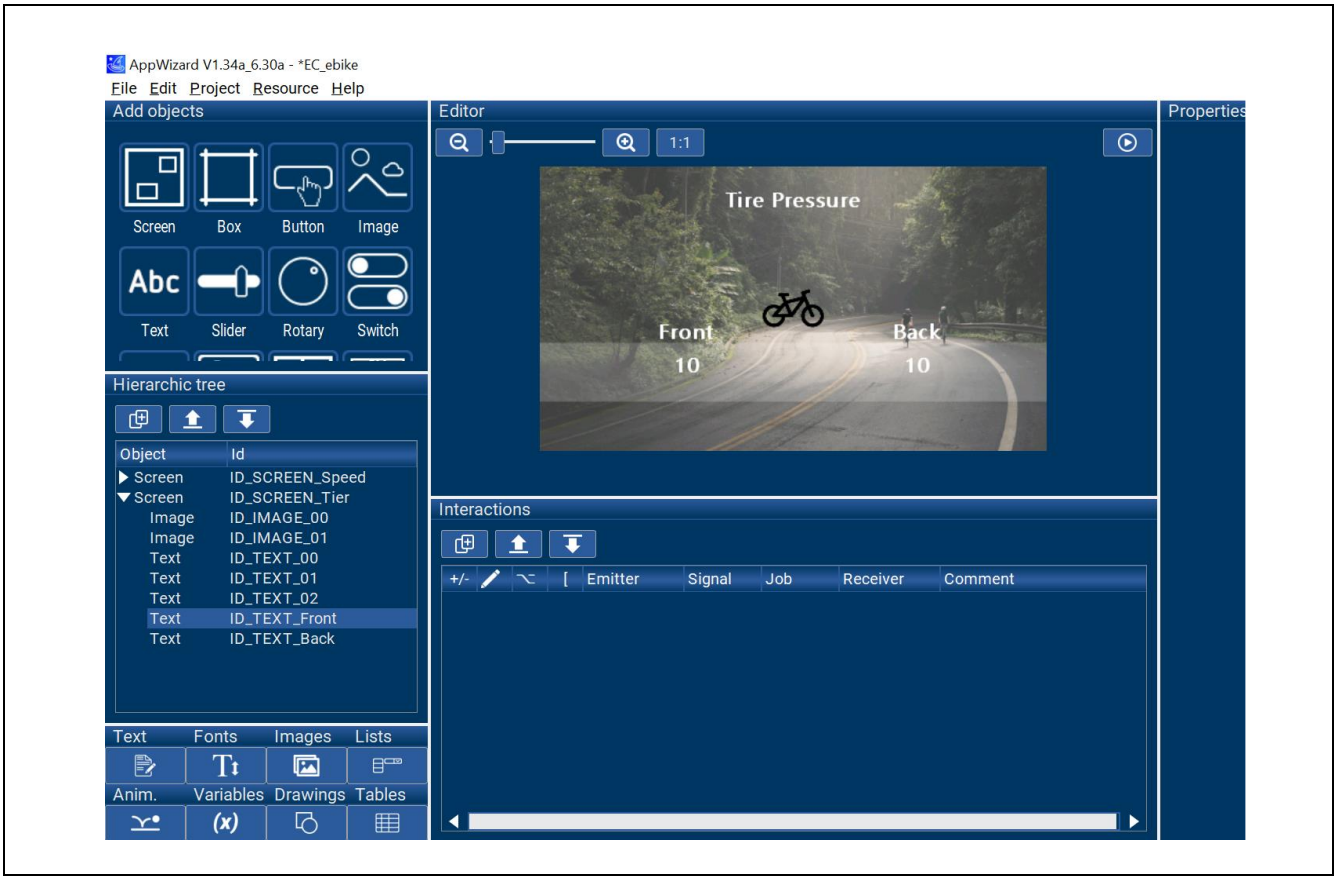


Figure 1-49 Using Text Object to Display Tire Number

28. Follow the step 16 to add a "Home" button. After completion, the result should be shown as Figure 1-50.

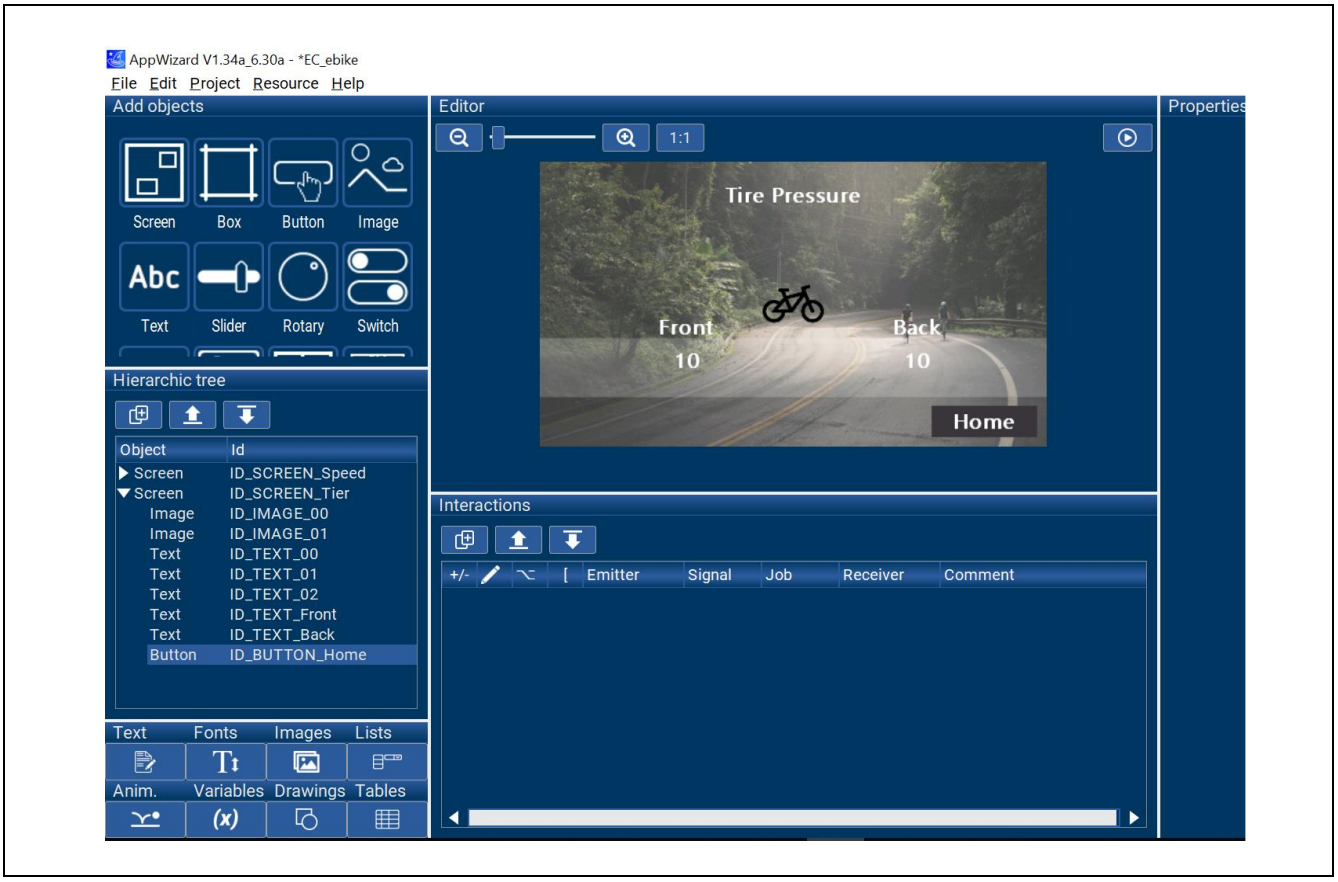


Figure 1-50 Implement a Screen Switching Button

29. Follow the step 16 to add language switch buttons for "中文" (Chinese) and "EN" (English). After completion, the result should be shown as Figure 1-51.

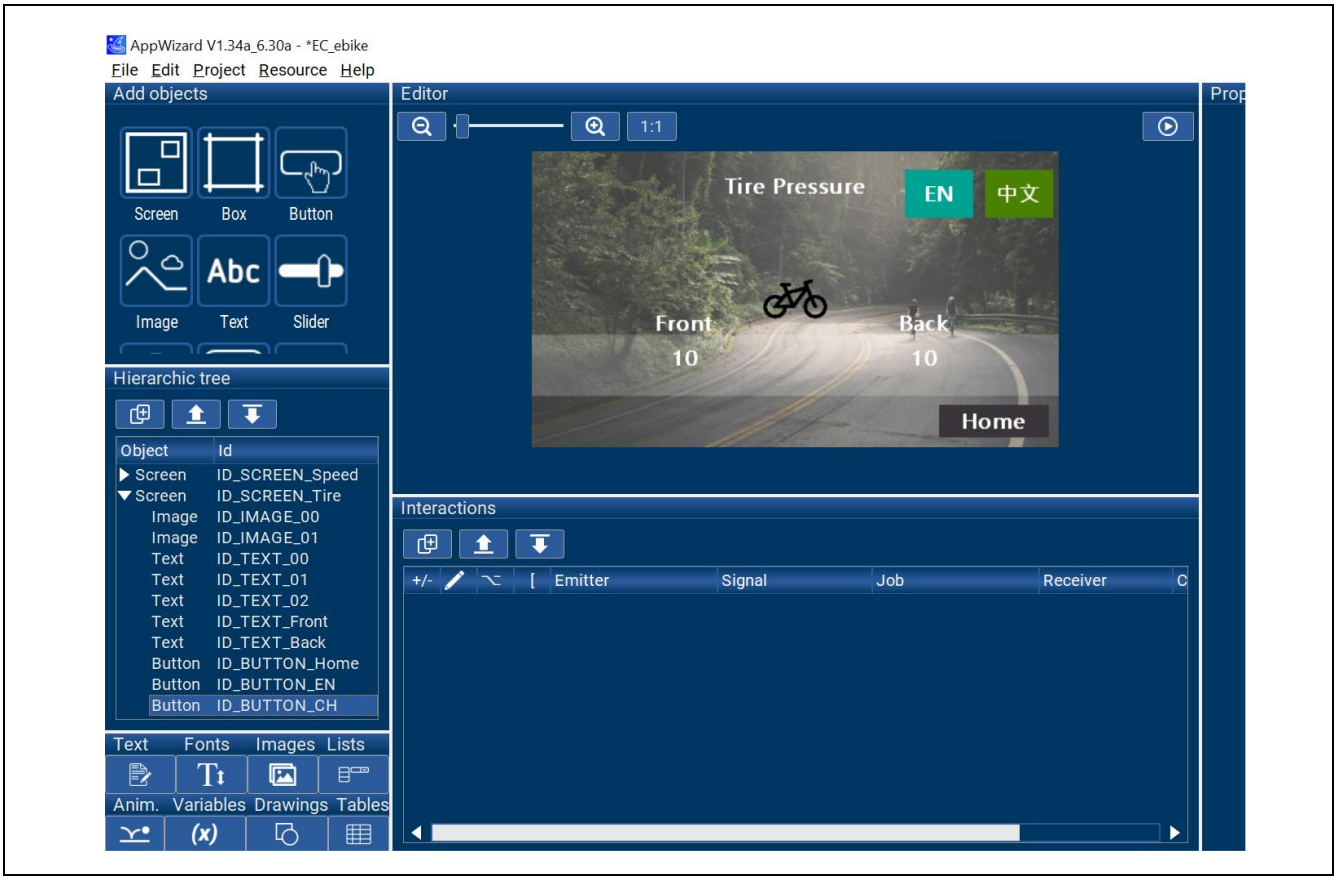


Figure 1-51 Using Button Object to Add a Language Switch Button

30. Begin setting up the interactions.

    1) Use buttons for page navigation and switching. Click on ID_BUTTON_Home under ID_SCREEN_Tire and set the Interactions section as shown in Figure 1-52. This completes the setup for switching from the Tire page to the Speed page. Similarly, set up the button on the Speed page to switch back to the Tire page, as shown in Figure 1-53.

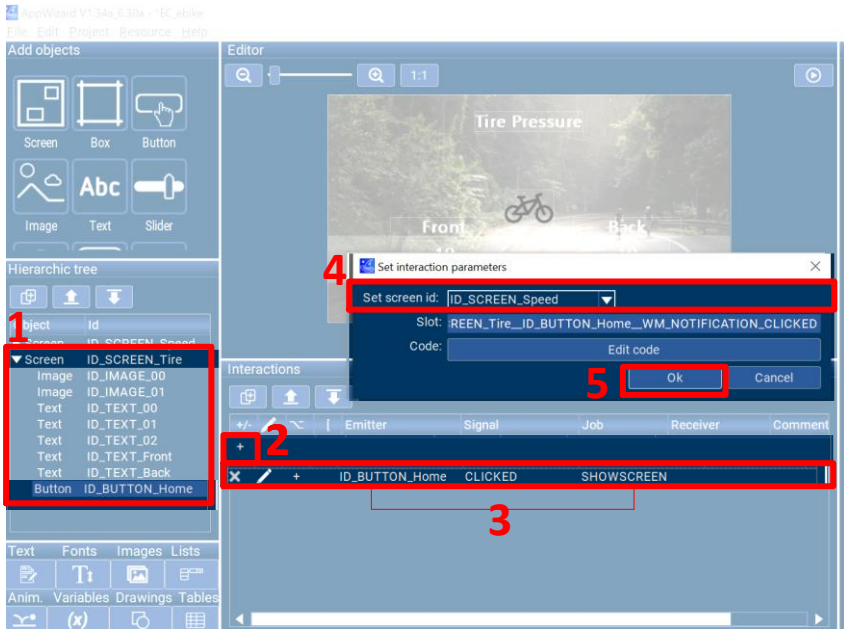


Figure 1-52 Interaction – Screen Switching

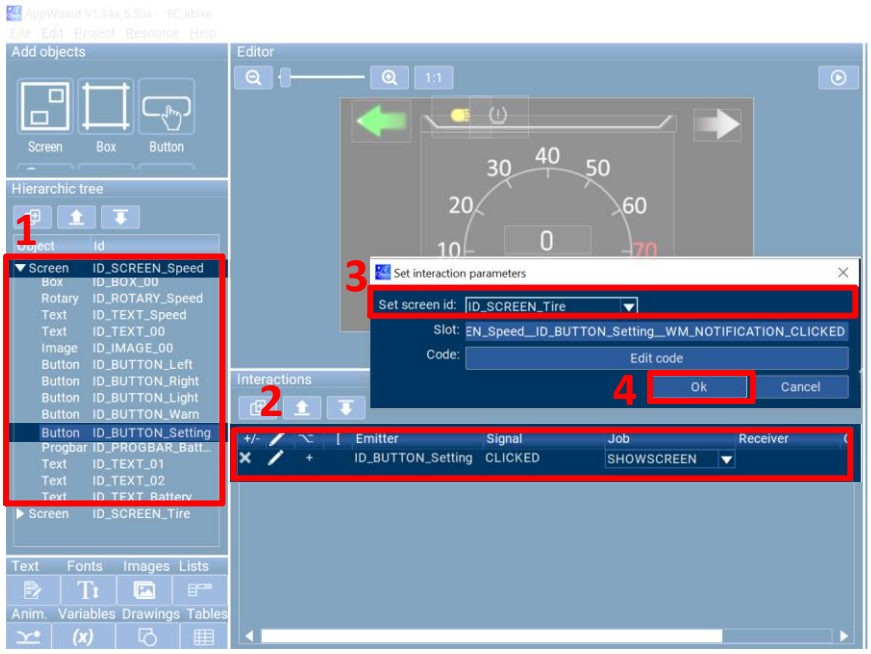


Figure 1-53 Interaction – Screen Switching

2) Use buttons for language switching. Click on ID_BUTTON_EN under ID_SCREEN_Tire and follow the steps as shown in Figure 1-54. This completes the setup for displaying English when the "EN" button is pressed. Similarly, set up the button to switch back to display Chinese, as shown in Figure 1-55.

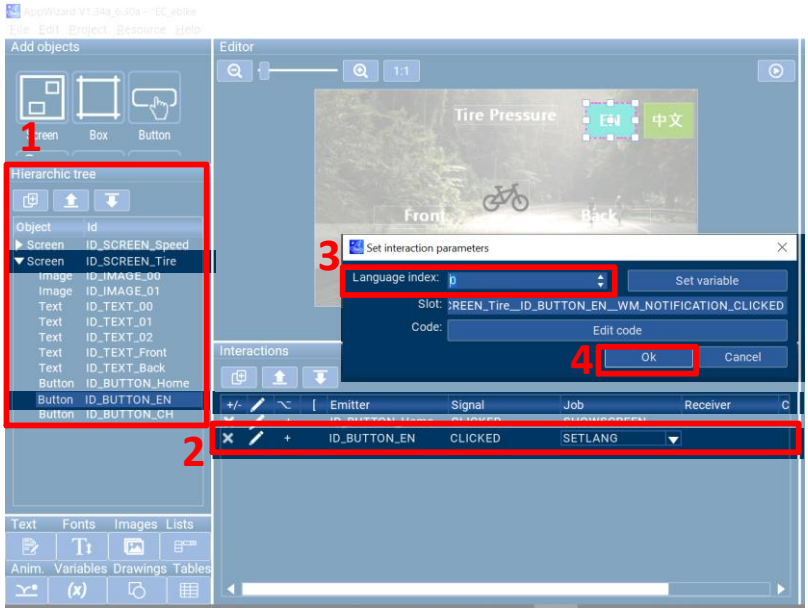


Figure 1-54 Interaction – Language Switching

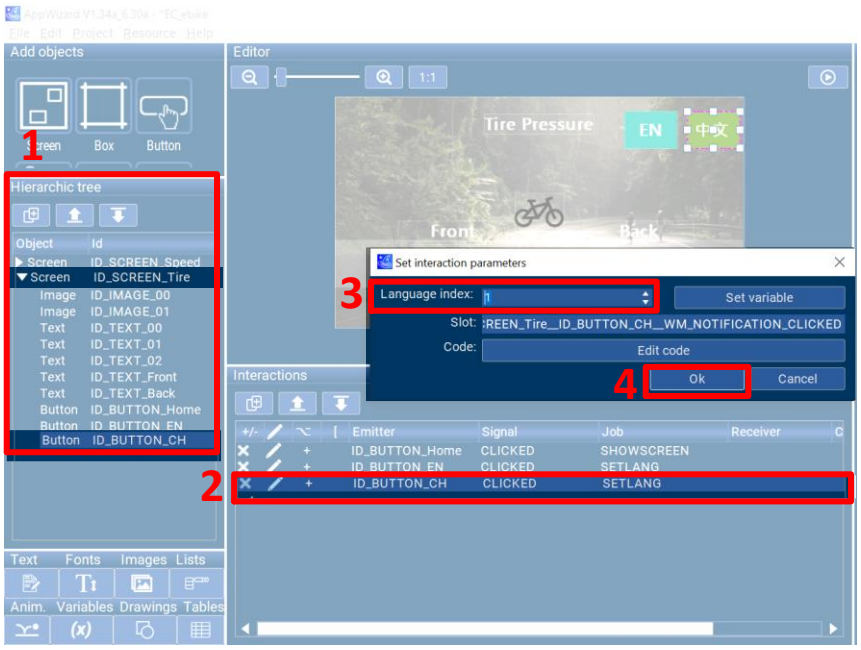


Figure 1-55 Interaction – Language Switching

3) Use the Variable feature to set the values for battery level and tire pressure after calculating them in the MCU. Assign the calculated values to variables in the code, and then use the Interaction feature to assign the variable values to the respective Text objects for display. The process is illustrated in Figure 1-56, and the actual steps can be followed as shown in Figure 1-57 to Figure 1-59. For the code steps, refer to the code explanation in Chapter 2 Code Description.

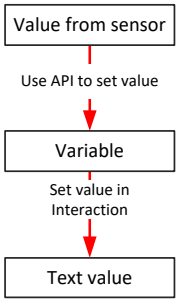


Figure 1-56 Process of Displaying MCU-Calculated Values on the Screen

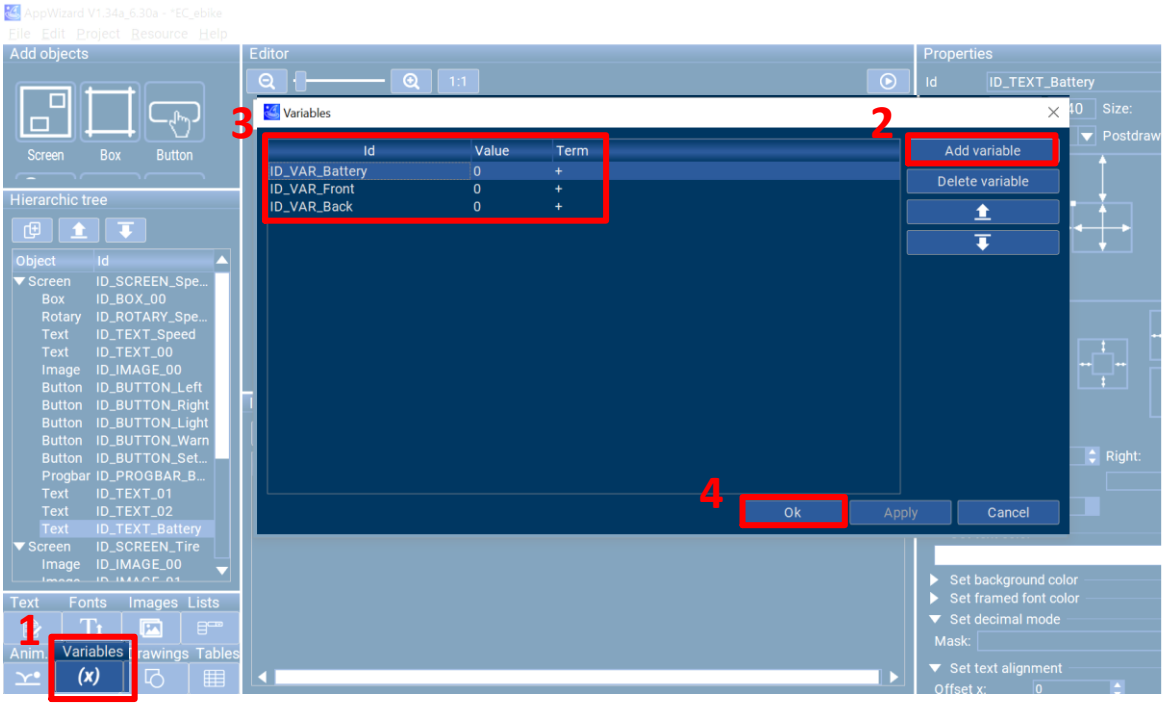


Figure 1-57 Variable Setting

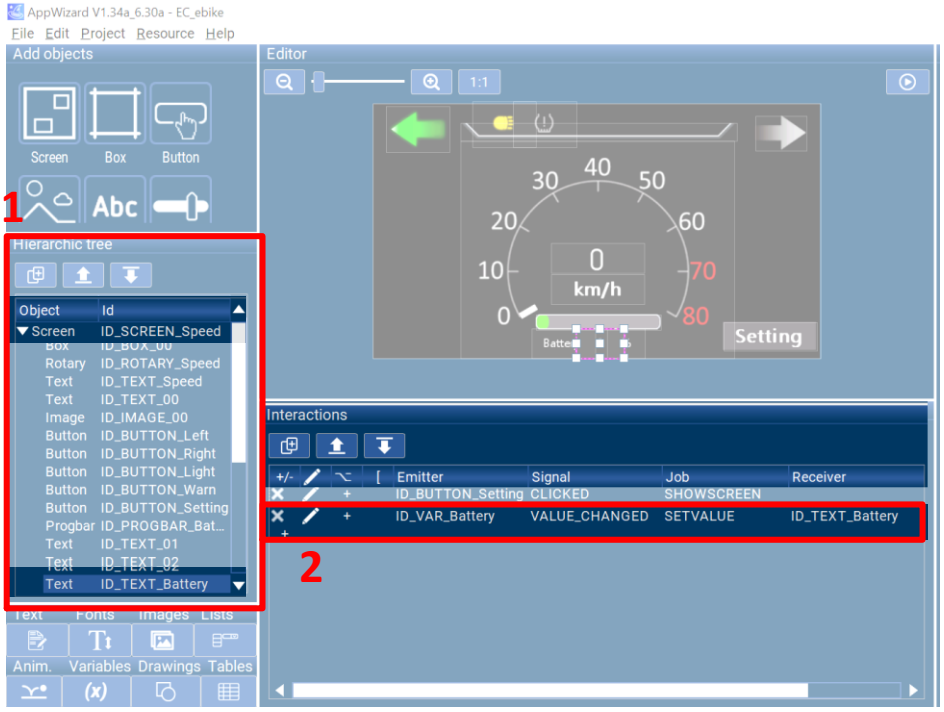


Figure 1-58 Interaction - Variable

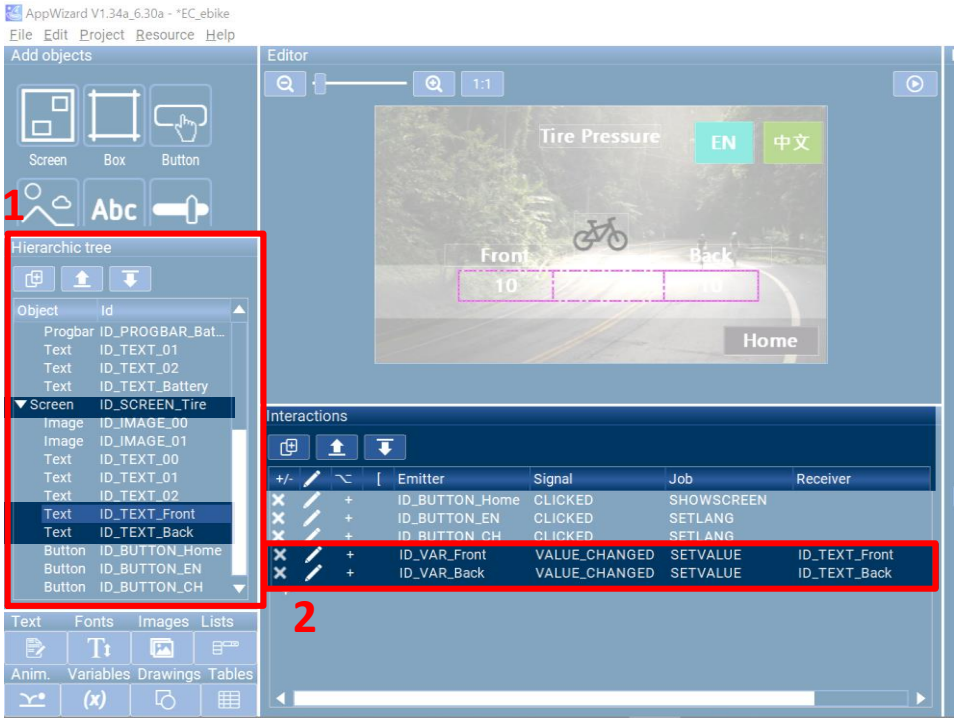


Figure 1-59 Interaction - Variable

4) Use the Animation feature to simulate the changes in the speedometer. Follow the steps shown in Figure 1-60 to Figure 1-63 to set up the automatic change of the speed value. Then, refer to Figure 1-64 to configure the rotation of the speedometer according to the speed value.

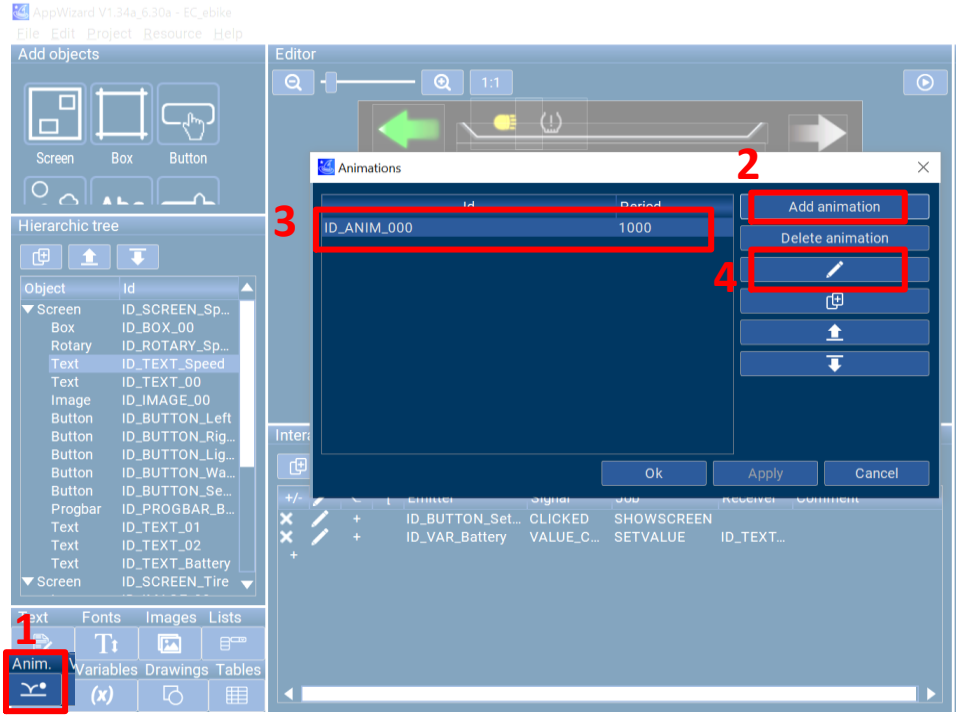


Figure 1-60 Animation Setting

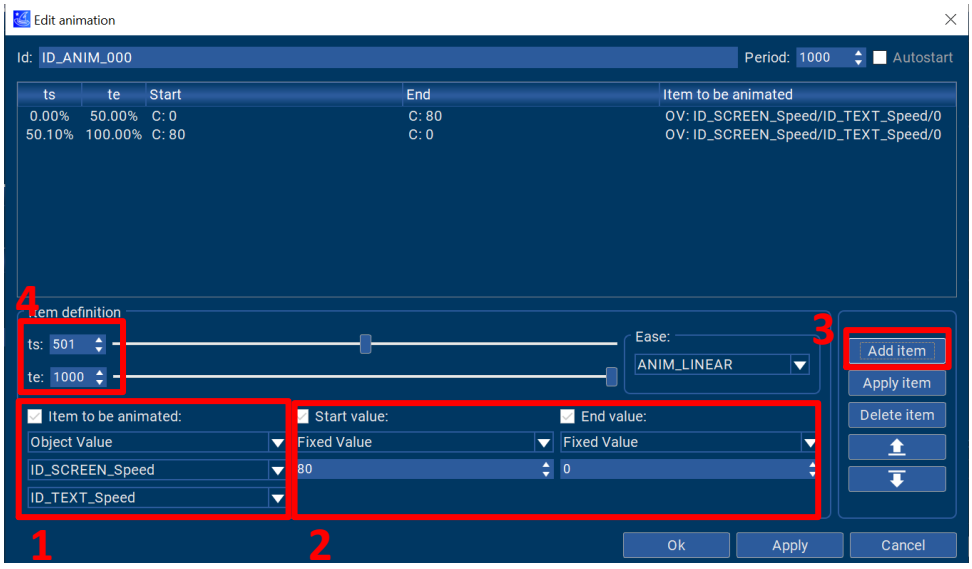


Figure 1-61 Animation Setting

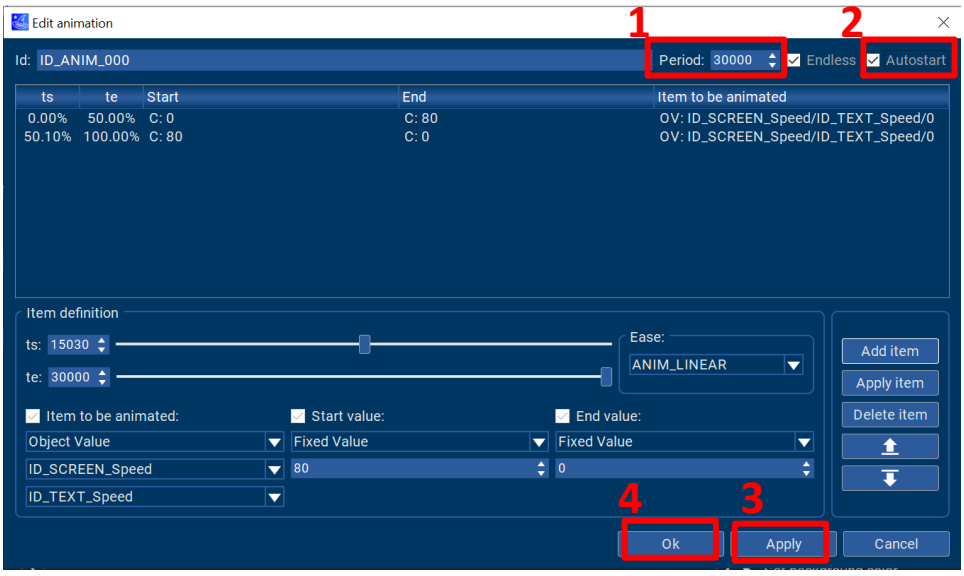


Figure 1-62 Animation Setting

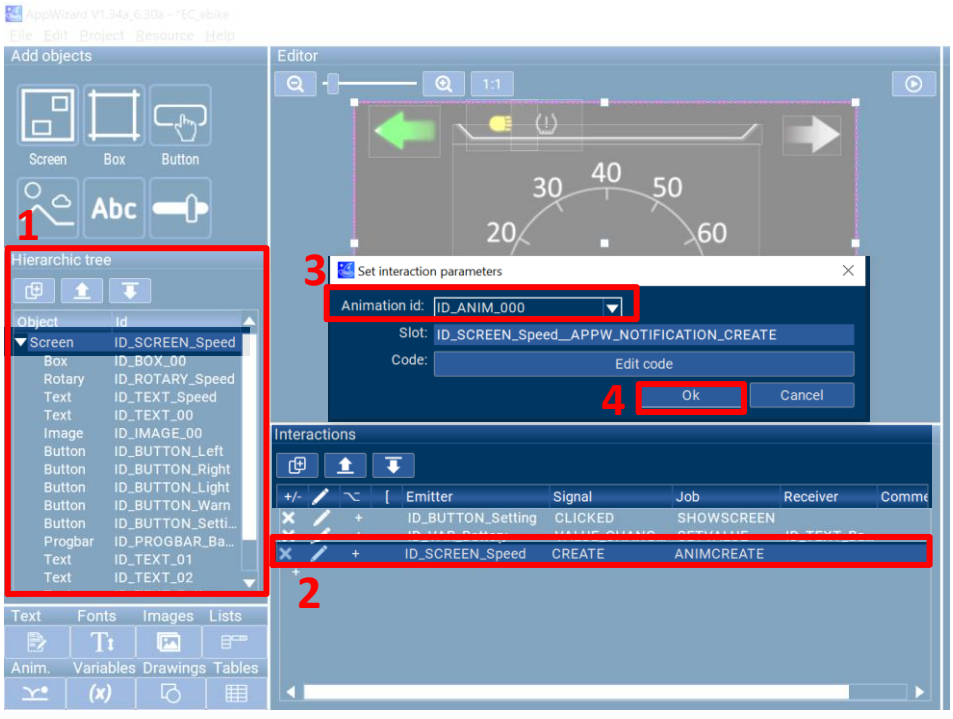


Figure 1-63 Interaction - Animation

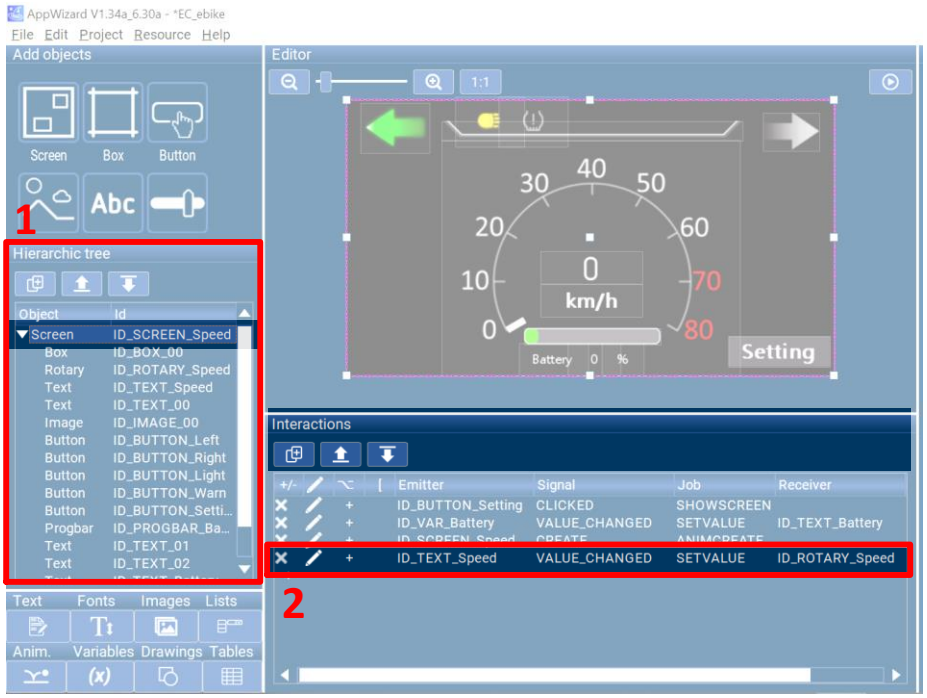


Figure 1-64 Interaction – Link Different Objects

5) Set up the linkage between the battery value and the battery display bar. Follow the steps shown in Figure 1-65 to configure the interaction.

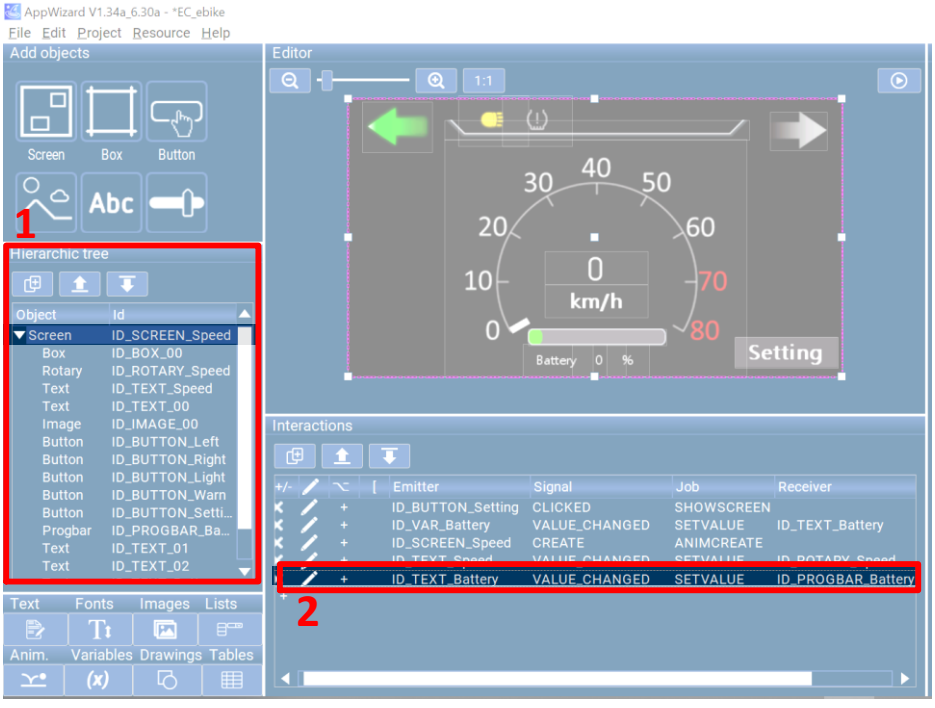


Figure 1-65 Interaction - Link Different Objects

31. Set up page switching using sliding motion. First, click on the Screen ID of the screen you want to switch in the Hierarchic tree. Then, in the properties section, set the motion to either horizontal or vertical. Follow the steps shown in Figure 1-66 to complete the operation.

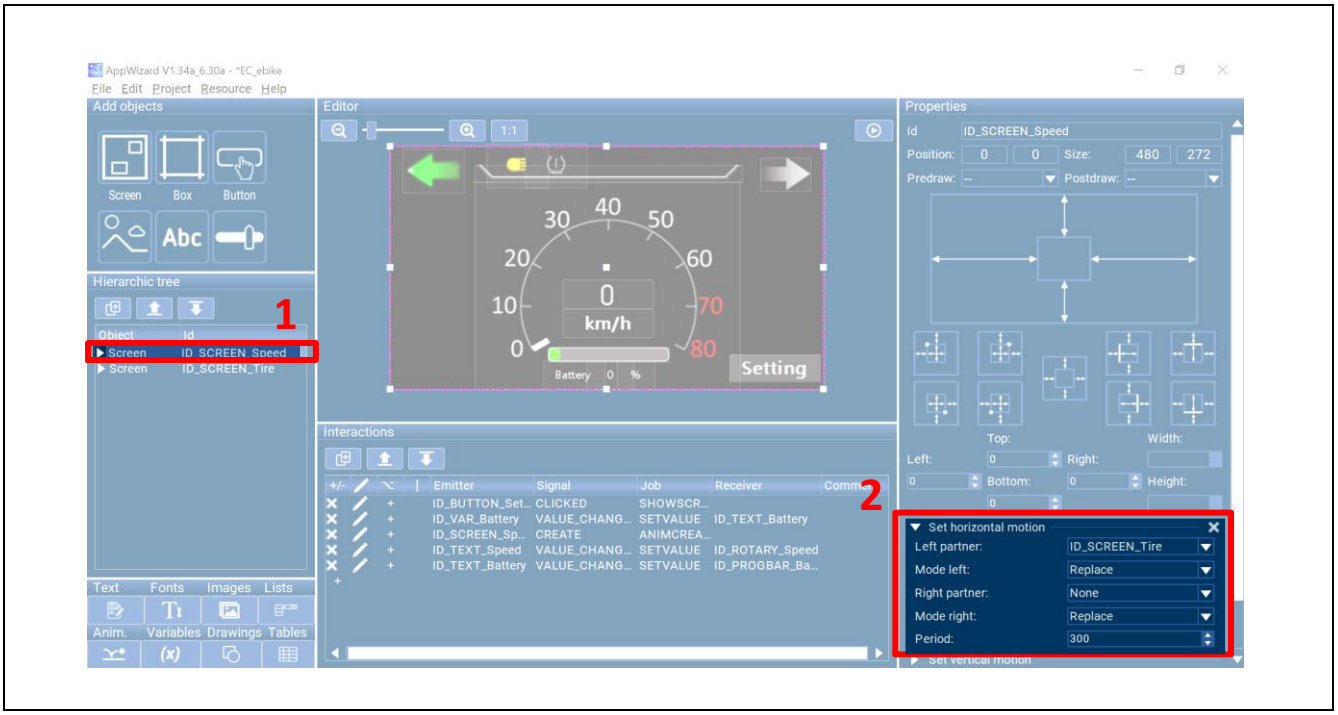


Figure 1-66 Screen - Properties

1.2.3 Coding

1. Once the design is completed, you can export the code through "**File** -> **Export & Save**", as shown in Figure 1-67. The project file used in this example is named as *EC_ebike.zip*. You can refer to Figure 4-1 for the file path. After extracting the file, you can directly open the project file using AppWizard.

Figure 1-67 Export Code

2. Open the AppWizard project folder through “**Project** -> **Open containing folder**”. Copy the "Resource" and "Source" folders. Navigate to the path "*EC_M467_AppWizard_Ebike_V1.00\SampleCode\ExampleCode\M467_AppWizard_Ebike\Application*" and replace the existing files with the copied folders. Follow the steps shown in Figure 1-68 to Figure 1-69.

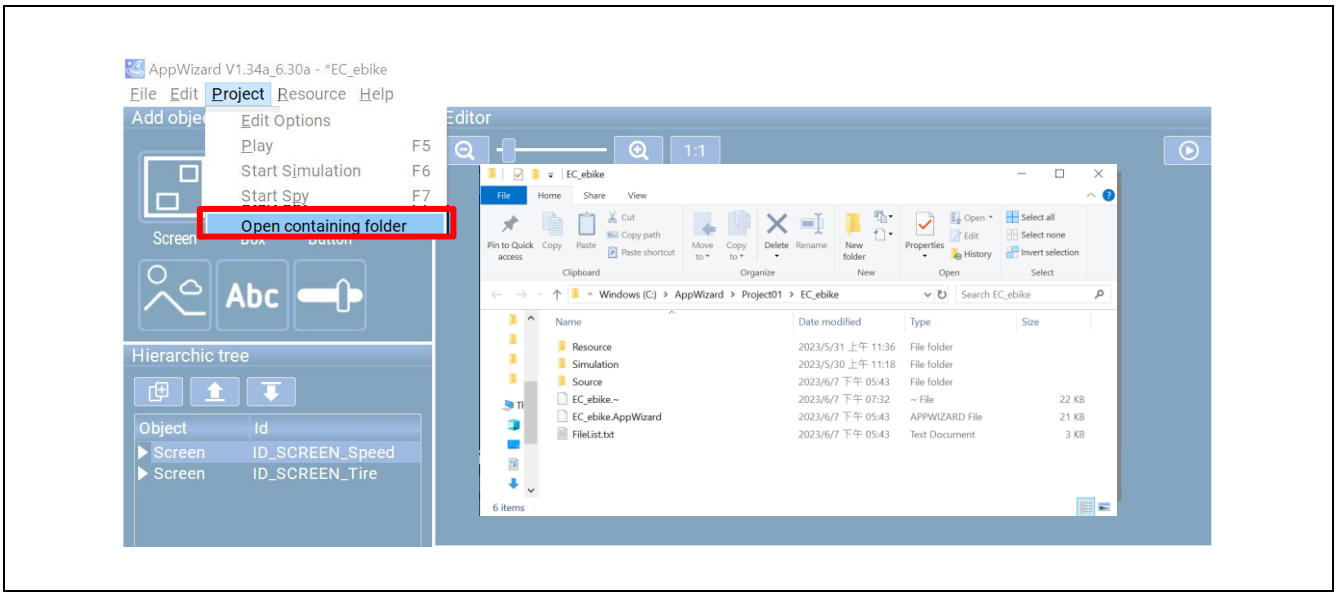


Figure 1-68 Open Project Folder

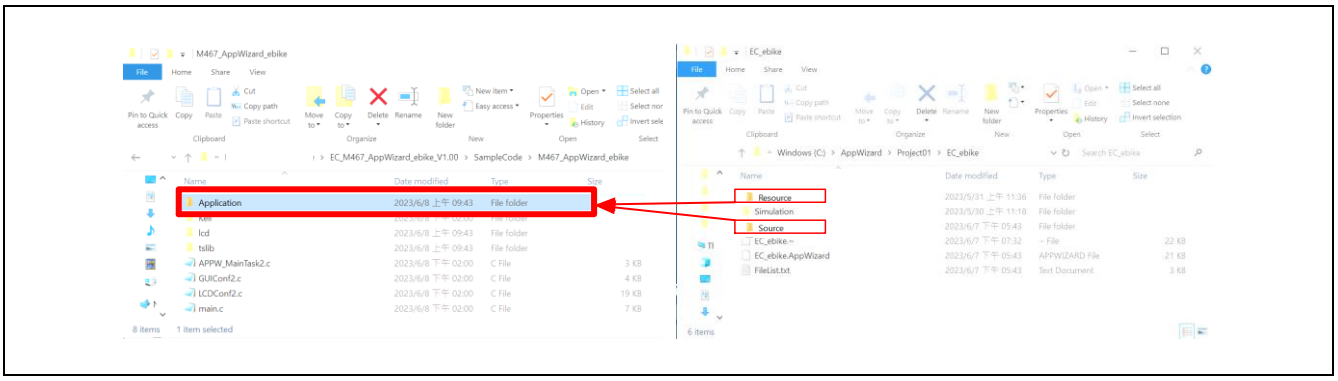


Figure 1-69 Porting to the Code Project

For the coding steps, refer to the code explanation in Chapter 2 Code Description.

1.2.4 Programming the Board

After running on the device, you can still make adjustments to optimize the overall performance. The following are the optimization settings and techniques you can apply.

- In AppWizard, the image format used should be set to match the screen format. In this example, the setting is "High color with alpha (565)."

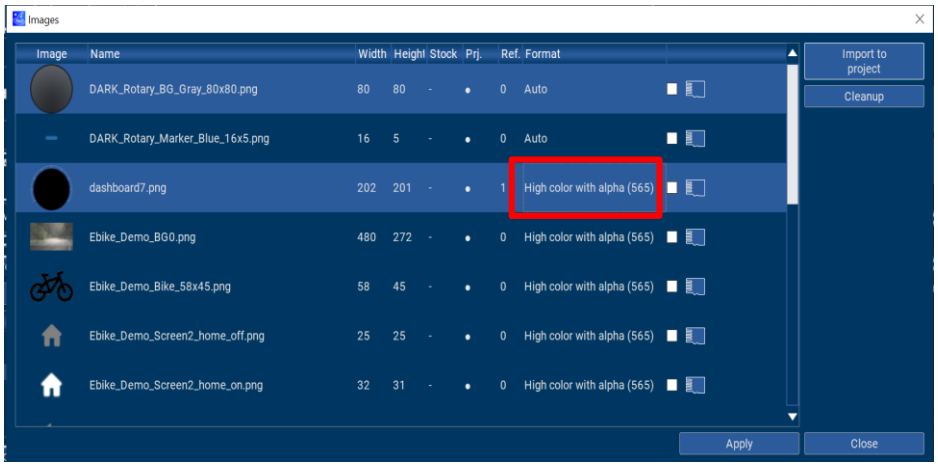


Figure 1-70 Image Setting

1.3 Demo Result

The results of compiling and flashing the code in this example are shown in Figure 1-71.

Figure 1-71 Demo Result

2. Code Description

The main program is entered through main.c and consists of three parts as follows:

- Clock initialization and pin definition: sys_init
- Peripheral initialization and open: UART, SPIM, Timer, Touch, etc.
- Application function: MainTask2()

```
int main(void)
{
    // Initialize system clock
    sys_init();
    // Initialize debug port
    debug_port_init();
    SPIM_init();
    g_pu8FrameBufPtr  = (U8 *)((U32)g_u8FrameBuf);
    g_enable_Touch = 0;
    // Initialize timer
    tmr_init();
    // Startup CTP
    st1633i_low_level_init();
    st1633i_startup_sequence();

    printf("\r\n");
    printf("CPU @ %d Hz\r\n", SystemCoreClock);
    printf("+------------------------------------------------------------+\r\n");
    printf("|               emWin Simple Demo                            |\r\n");
    printf("+------------------------------------------------------------+\r\n");
    printf("\r\n");

    g_enable_Touch = 1;
    // Enable memory device before GUI_Init() to avoid flickering
    printf("Main Task2 -> \n");
    // Main Loop
    // MainTask2 for anti-flickering
    MainTask2();

    return 0;
}
```

The SPIM interface is used to store larger-sized image and text file resources in an external SPI Flash memory. The SPI Synchronous Serial Interface Controller (SPI Master Mode) is utilized in Direct Memory Mapping (DMM) mode, where the SPI Flash can be treated as a ROM module. This allows direct reading of the SPI Flash from specified addresses. With the assistance of the built-in cache and by quad read mode(4-wire), the overall read efficiency is improved.

The initialization code for SPIM is as follows. It includes setting the SPIM speed, SPI flash detection, enabling SPIM cache mode, enabling SPIM DMM mode, and enabling fast quad read mode. Please refer to the W25Q32JVTBIQ documentation for the specific parameter settings.

```
void SPIM_init(void){
    uint8_t     idBuf[3]:
    SYS_UnlockReg();            /* Unlock protected registers */
    SPIM_SET_CLOCK_DIVIDER(2);    /* Set SPIM clock as HCLK divided by 2, 200MHz/2 = 100*/

    SPIM_SET_RXCLKDLY_RDDLYSEL(0);    /* Insert 0 delay cycle. Adjust the sampling clock
of received data to latch the correct data. */
    SPIM_SET_RXCLKDLY_RDEDGE();       /* Use SPI input clock rising edge to sample
received data. */

    SPIM_SET_DCNUM(4);          /* 4 is dummy cycle for command 0xEB(fast quad read).*/

    if (SPIM_InitFlash(1) != 0)        /* Initialized SPI flash */
    {
        printf("SPIM flash initialize failed!\n");
    }

    SPIM_ReadJedecId(idBuf, sizeof (idBuf), 1);
    printf("SPIM get JEDEC ID=0x%02X, 0x%02X, 0x%02X\n", idBuf[0], idBuf[1], idBuf[2]);

    SPIM_DISABLE_CCM();

    SPIM_ENABLE_CACHE();

    if (SPIM_CIPHER_ON)
        SPIM_ENABLE_CIPHER();
    else
        SPIM_DISABLE_CIPHER();

    if (SPIM_Enable_4Bytes_Mode(USE_4_BYTES_MODE, 1) != 0)
    {
        printf("SPIM_Enable_4Bytes_Mode failed!\n");
    }

    SPIM->CTL1 |= SPIM_CTL1_CDINVAL_Msk;        // invalid cache
    SPIM_SetQuadEnable(1, 1);   //set quad mode bit
    SPIM_EnterDirectMapMode(USE_4_BYTES_MODE, CMD_DMA_FAST_QUAD_READ, 0);

}
```

In addition to SPIM initialization, you also need to edit the Scatter file to let the compiler know the location and content of the SPI Flash. In the Keil environment, you can follow the path “Options for Target -> Linker -> Scatter File -> Edit” to open the .sct file and edit its contents as follows:

```
;APROM
LOAD_ROM_1  0x0
{
    APROM.bin  0x0
    {
     startup_M460.o (RESET, +FIRST)
     *(+RO)
    }


    SRAM  0x20000000 0x80000
    {
     * (+RW, +ZI)
    }
}
```

```
;SPI flash
LOAD_ROM_2  0x100000
{
    SPIM.bin  0x100000
    {
     microsoftjhengheiui_16_normal_ext_aa4.o (+RO)
     microsoftjhengheiui_24_bold_ext_aa4.o (+RO)
     nettoot_40_normal_ext_aa4.o (+RO)
     speedmeter_0.o (+RO)
     ebike_demo_screen2_l_off.o (+RO)
     ebike_demo_screen2_l_on.o (+RO)
     ebike_demo_screen2_light_off.o (+RO)
     ebike_demo_screen2_light_on.o (+RO)
     ebike_demo_screen2_pressure_on.o (+RO)
     ebike_demo_screen2_r_off.o (+RO)
     ebike_demo_screen2_r_on.o (+RO)
     ebike_demo_screen2_pressure_off.o (+RO)
     rotary_marker_whitebar_20x7.o (+RO)
     turnline_3.o (+RO)
     Ebike_Demo_BG0.o (+RO)
     ebike_demo_bike_58x45.o (+RO)
    }
}
```

After the program enters the main application function MainTask2(), it will execute the relevant AppWizard functions. Inside the while loop, there will be actions to draw graphics on the screen. In this example, an additional function APPW_SetVarData() is added to pass values from the MCU to the Variable in AppWizard. By following the steps described in 3), the values will be displayed on the screen.

```
void MainTask2(void)
{
    // Setup configuration dependent pointers
    APPW_X_Setup();
    // Initialize AppWizard
    APPW_Init(APPW_PROJECT_PATH);
    // Create all persistent screens except initial screen
    APPW_CreatePersistentScreens();
    // Create initial screen...
    APPW_CreateRoot(APPW_INITIAL_SCREEN, WM_HBKWIN);
    // ...and keep it alive
    while (1)
    {
      // Drawing a frame
      GUI_Exec1();
      APPW_Exec();
      NVT_Draw1();
      APPW_SetVarData(ID_VAR_Battery, 88);
      APPW_SetVarData(ID_VAR_Back, 31);
      APPW_SetVarData(ID_VAR_Front, 40);
    }
}
```

The drawing content executed by AppWizard is read from ID_Screen_xxx_Slots.c. You can try adding printf("Button Click\n") to the code when the button is pressed in step 16. In the actual

Speed screen, if you press the Setting button to navigate to the Tire page, the UART will print the message "Button Click" on the PC.

```
/*
File        : ID_SCREEN_Speed_Slots.c
Purpose     : AppWizard managed file, function content could be changed
---------------------------END-OF-HEADER------------------------------
*/

#include "Application.h"
#include "../Generated/Resource.h"
#include "../Generated/ID_SCREEN_Speed.h"

/*** Begin of user code area ***/
/*** End of user code area ***/

/*********************************************************************
*
*       Public code
*
**********************************************************************
*/
/*********************************************************************
*
*       cbID_SCREEN_Speed
*/
void cbID_SCREEN_Speed(WM_MESSAGE * pMsg) {
  GUI_USE_PARA(pMsg);
}

/*********************************************************************
*
*       ID_SCREEN_Speed__ID_BUTTON_Setting__WM_NOTIFICATION_CLICKED
*/
void ID_SCREEN_Speed__ID_BUTTON_Setting__WM_NOTIFICATION_CLICKED(APPW_ACTION_ITEM *
pAction, WM_HWIN hScreen, WM_MESSAGE * pMsg, int * pResult) {
  GUI_USE_PARA(pAction);
  GUI_USE_PARA(hScreen);
  GUI_USE_PARA(pMsg);
  GUI_USE_PARA(pResult);
  printf("Button Click\n");
}
```

3. Software and Hardware Requirements

3.1 Software Requirements

- BSP version
  - M460_Series_BSP_CMSIS_V3.00.002
- emWin Software Package for M460 version
  - emWin Software Package for M460 V3.0
- IDE version
  - Keil uVersion 5.38.0.0
- AppWizard version
  - AppWizard V1.34a_6.30a

3.2 Hardware Requirements

- Circuit components
  - NuMaker-HMI-M467
- Connect NuMaker-HMI-M467 and PC
  - Connect the Nu-Link2-Me and the daughter board's screen USB connector to the PC using two USB cables, as shown in Figure 3-1.

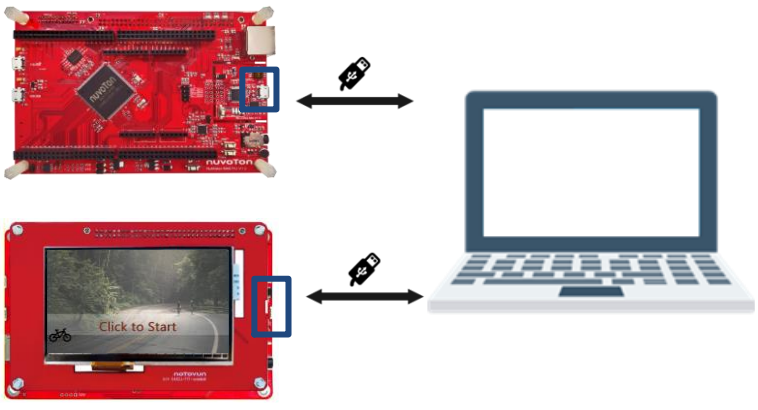


Figure 3-1 Connect NuMaker-HMI-M467 and PC

4. Directory Information

The directory structure is shown below.

EC_M467_AppWizard_Ebike_V1.00	
Library	Sample code header and source files
CMSIS	Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
Device	CMSIS compliant device header file
StdDriver	All peripheral driver header and source files
SampleCode	
ExampleCode	Source file of example code
ThirdParty	Header files for emWin project
EC_ebike.zip	Ebike sample project of AppWizard. Please open by AppWizard.exe.

Figure 4-1 Directory Structure

5. Example Code Execution

1. Browse the sample code folder as described in the Directory Information section and double-click *M467_AppWizard_Ebike.uvprojx*.
2. Enter Keil compile mode.
   - Build
   - Download
   - Start/Stop debug session
3. Enter debug mode.
   - Run

6. Revision History

Date	Revision	Description
2023.05.31	1.00	Initial version.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.