Our development idea is as follows：

Step one：you install QT and PyCharm on your PC .

Step two: you develop and debug in your pc

Step three: you copy the code on 410C board and run it (python + the path of maintable’s file)

First. the environment to build:

https://draapho.github.io/2016/11/15/1616-python-ble/

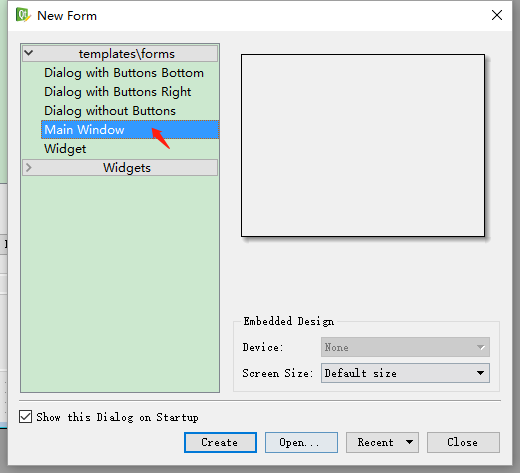
1. Pygatt download

Open the terminal and enter pip install pygatt

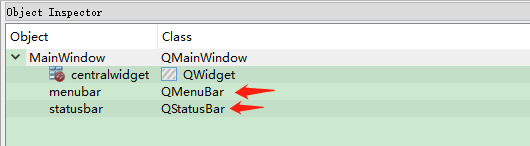
Second. use Qt Designer to develop UI:

Create the project directory SmartLighting and create the libs folder in the SmartLighting directory.

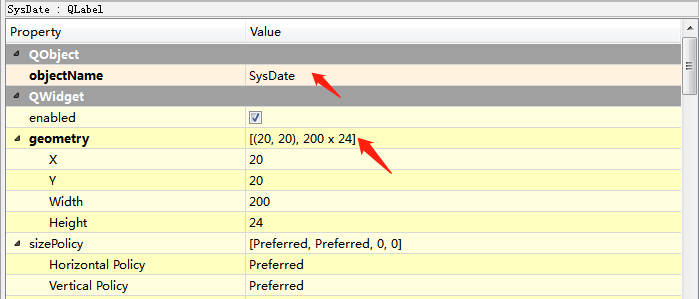
Open the QtDesigner tool, click File->New, select the Main Window option in the pop-up panel, click the Creat button, create a UI file, named SmartLightUI.ui, save it to the project directory SmartLighting/libs path.



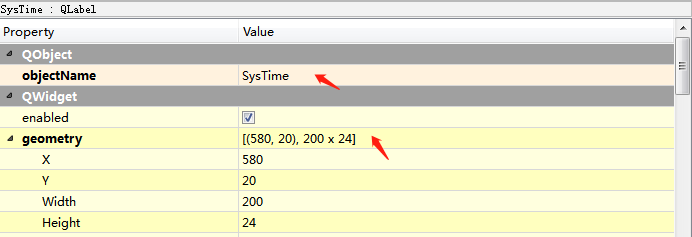
After creating the ui file, right-click on the menubar and statusbar in the ObjectInspector panel and click on the corresponding remove to delete the two objects.



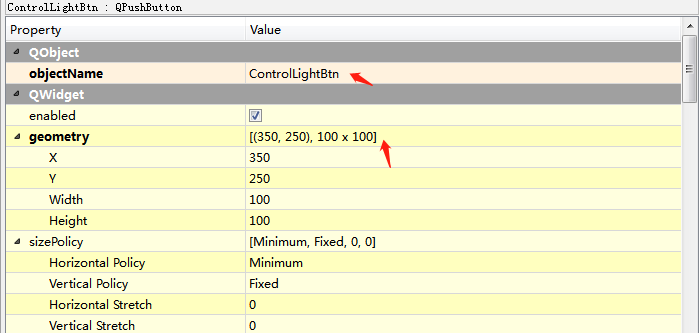
Locate the Label component in the WidgetBox panel, drag it to the ui interface, select the created Label component in the Object Inspector, and in the PropertyEidtor panel, modify the Label name to SysDate and modify the geometry as shown below.



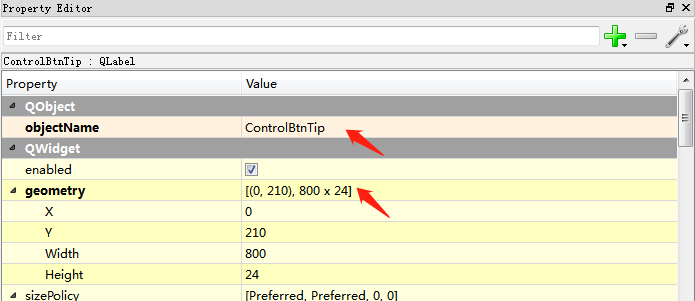
Locate the Label component in the WidgetBox panel, drag it to the ui interface, select the created Label component in the Object Inspector, and in the PropertyEidtor panel, modify the Label name to SysTime and modify the geometry as shown below.



Find the Push Button component in the Widget box panel, drag and drop it into the Light panel, and set its properties as follows:



Locate the Label component in the Widget Box panel, drag it into the UI panel, and set its properties as follows:



Third. use PyCharm to develop functional modules:

1. BleDeviceManager module

Open PyCharm and import the SmartLighting project directory, right-click on the SmartLIghting/libs directory, select New->Python File, and create the BleDeviceManager.py file.

BleDeviceManager handles the Ble communication related function logic in the Light module.

2. LightManager module

Right-click on the SmartLIghting/libs directory and select New->Python File to create the LightManager.py file.

The LightManager module is primarily responsible for managing Light core logic code.

3. SmartLightData module

Right-click on the SmartLIghting/libs directory and select New->Python File to create the SmartLightData.py file.

The SmartLightData module is primarily responsible for saving Light related data.

4. TimeManager module

Right-click on the SmartLIghting/libs directory and select New->Python File to create the TimeManager.py file.

The TimeManager module is mainly responsible for saving the relevant logic code of the Time module.

5. Main module

Right-click on the SmartLIghting directory and select New->Python File to create the Main.py file.

The Main module is mainly responsible for providing program exits, handling UI response events, and managing functions such as modules.

Fourth, the API:

1. BleDeviceManager module

|  |  |
| --- | --- |
| Attributes | Effect |
| Self.device | Current ble device |
| self.adapter | Current Ble device adapter |
|  | |
| Method | Effect |
| stop(self) | Stop adapter |
| scan(self, timeout=5) | Scan the current Ble device |
| connect\_name(self, name, devices=None) | Connect a ble device named name |
| connect(self, address) | Connect the device according to the Ble device address |
| discover\_characteristics(self, device=None) | Find the attribute value of the device Ble communication |
| read\_characteristics(self, uuid, device=None) | Read Ble device attribute values according to uuid |
| write\_characteristics(self, str, uuid, device=None) | Write device attribute values based on uuid |

1. LightManager module

|  |  |
| --- | --- |
| Attributes | Effect |
| self.lightIsOpen | Whether the current light is on |
| self.lightStateCallBack | Light state change callback |
| self.\_\_controlValueUrl | Wilddog saves light state value node |
| self.\_\_myApp | Main module MyApp instance |
| self.\_\_wilddogMgr | Wilddog Module instance |
|  | |
| Method | Effect |
| WilddogCallBack(self,wilddogInfo) | Handling Wilddog module callback information |
| ChargeLightControllerState(self) | Switch light status |
| InitLight(self) | Initialize the light state |
| OpenLight(self) | Turn on the light |
| CloseLight(self) | Turn off the light |

1. SmartLightData module

|  |  |
| --- | --- |
| Attributes | Effect |
| self.DataFormat | System date display format |
| self.TimeFormat | System tine display format |
| self.DataTimeFormat | System time and date display format |
| self.OpenLightBtnTip | Light button open status prompt text |
| self.CloseLightBtnTip | Light button close status prompt text |

1. TimeManager module

|  |  |
| --- | --- |
| Method | Effect |
| GetSystemDataTime(self) | Get current system time |
| GetSystemDataTimeStamp(self) | Get the current system timestamp |
| StartUpdateDataTime(self,callBackFunc) | Start updating system time, execute callBackFunc callback method every second |
| StopUpdateDataTime(self) | Stop updating system time |
| updateTimerCallBackFunc(self) | Callback method for performing update time |

1. Main module

|  |  |
| --- | --- |
| Attributes | Effect |
| Self.\_smartLightData | SmartLIghtData Module instance |
| Self.\_lightControler | LightManager Module instance |
| Self.\_timeMgr | TimeManager Module instance |
|  | |
| Method | Effect |
| OnLightBtnClick(self) | UI Light button click event |
| OnLightChanged(self,lightIsOpen,tipCode) | Light state change callback |
| UpdateDataTimeLabel(self, dataTime) | Update time and date prompt text |
| if \_\_name\_\_ == "\_\_main\_\_": | Main program entry |