Spring 2019

Capstone

Camper Water System & Power Monitoring Unit

Allia Richardson
EASTERN OREGON UNIVERSITY

Table of Contents

1. Proposal

General Description and Purpose:

A software application that monitors the voltage the power supply, battery, gives to the Raspberry Pi, monitors the levels of the fresh water, grey water, and black water tanks, and displays the usage of the tanks and battery over time.

A camper and/or Recreation vehicle (RV) has three types of water tanks, which are fresh water, grey water, and black water. The fresh water tank holds the water that can be consumed, used to wash dishes, and/or used for bathing. The grey water takes holds the water that was used by household activities, such as bath water, dish water, and/or shower water. The black water tank holds the used toilet water. The application will monitor these three tanks, it will insure that the black and grey water do not exceed the capacity of their tanks, that the fresh water does not become empty, and that the power voltage from the battery does not get low enough to shut down the Raspberry Pi. If the black and grey water tanks get too high or the fresh water and battery voltage get too low an alarm will go off to alert the user that something needs to happen.

The tanks will utilize a sensor to monitor the water pressure. If the water pressure changes it alters the voltage signal which the DAQC board read and send to the system. The voltage, which is fed into the DAQC card, is interpreted by the system.

Audience:

The targeted audience for this application will be individuals that own an RV or a camper.

Intended Platforms:

The application will be made with the use of Python3 and PyQt5, and it will run on the Raspberry Pi's operating system platform, Raspbian.

Devices and Hardware:

Raspberry Pi
Raspberry Pi 3.3190G 7" Touchscreen Display
Pi-Plates DAQCplate
Three water pressure sensors
An alarm device
An Arduino
A breadboard

Key Features:

The Sensor tab allows the user the ability to see percent capacity of the tanks. It utilizes three colors. The first is green, which indicates that the tanks and the battery power are in good standing. The second is orange, which indicates that the user should start to consider the action that they would like to take with the tanks and battery (fill up the fresh water, empty the black and/or grey water tanks, or replace battery. The third is red, which indicates that the user needs to take action immediately before it is too late.

The Usage tab allows the user the ability to see the history of the tanks levels and the battery voltage. If the user presses the refresh button it will show the information of the usage history within a graph.

The Power tab allows the user to either power down the Raspberry Pi or it allows them the option of turning off the application. If they choose either one a message window will pop-up and ask if they are sure they would like to close the window or the power down the Raspberry Pi.

The Alarm window will pop-up if at least one or more of these occur, the grey and black water reach 80 percent or higher, the fresh water drops down to 20 percent, or the battery drops down to 45 percent voltage (2.25 volts). If the user closed the window the alarm will turn off, but if another sensor or a voltage drop occurs it will reinstate the alarm.

Glossary of Terms:

- **Raspberry Pi –** A devices that is a fully functional computer, which contains its own operating system.
- **Fresh Water –** Water that is drinkable.
- **Grey Water –** Water that was used by household activities such as bathing and washing dishes.
- **Black Water –** Used toilet water.
- **Python –** A high-level general-purpose and difficult programing Language
- **PyQt5 –** Is a set of comprehensive python bindings for Qt version 5, which allows for the creation of graphical user interfaces (GUIs).
- **DAQC –** (Pronounced: dac-see) Data Acquisition and Communication
- **Pi-Plates DAQCplate –** A DAQC that is designed to attach to a Raspberry Pi.
- **Arduino –** A micro controller.
- **Breadboard –** A board that allows for circuit design for temporary electronic devices, and it does not use solder.

2. **The User Requirements –** A list of requirements that the user should be able to perform on the application.

2.1. **Sensor Tab**

2.1.1. The user must be able to view the progress bar for…

2.1.1.1. …fresh water.
2.1.1.2. …grey water.
2.1.1.3. …black water.
2.1.1.4. …battery.

2.1.2. The user must be able to select the…

2.1.2.1. …Usage tab.
2.1.2.2. …Power tab.

2.2. **Usage Tab**

2.2.1. The user must be able to click a button to refresh the graph.

2.2.2. The user must be able to view a graph that show the last 100 records recorded for…

2.2.2.1. …fresh water tank percentage.
2.2.2.2. …grey water tank percentage.
2.2.2.3. …black water tank percentage.
2.2.2.4. …battery percentage.

2.2.3. The user must be able to select the…

2.2.3.1. …Progress tab.
2.2.3.2. …Power tab.

2.3. **Power Tab**

2.3.1. The user must have the ability to…

2.3.1.1. …exit the application.
2.3.1.2. …power down the Raspberry Pi.

2.3.2. The user must be given a warning window to confirm…

2.3.2.1. …the application is to be shut down.
2.3.2.2. …the Raspberry Pi is to be powered off.

2.4. **Alarm Window**

2.4.1. The user must be informed of why there is a warning.
2.4.2. The user must be able to hit an ‘okay’ button to turn off the alarm.

3. **Functional Requirements** – Requirements that the application(s) should be able to perform to meet the needs of the user.

3.1. **Sensor Tab**

3.1.1. The system must record information into the database for…

3.1.1.1. …fresh water.

3.1.1.2. …grey water.

3.1.1.3. …black water.

3.1.1.4. …battery.

3.1.1.5. …time recorded

3.1.2. The system will set the progress bar's color to green when…

3.1.2.1. …the fresh water tank measurement is between 50% and 100%.

3.1.2.2. …the grey water tank measurement is between 0 and 40%.

3.1.2.3. …the black water tank measurement is between 0 and 40%.

3.1.2.4. …the battery usage is between 80% and 100%

3.1.3. The system will set the progress bar's color to orange when…

3.1.3.1. …the fresh water tank measurement is between 21% and 49%.

3.1.3.2. …the grey water tank measurement is between 41% and 79%.

3.1.3.3. …the black water tank measurement is between 41% and 79%.

3.1.3.4. …the battery usage is between 61% and 79%

3.1.4. The system will set the progress bar's color to red when…

3.1.4.1. …the fresh water tank measurement is less than or equal to 20%.

3.1.4.2. …the grey water tank measurement is greater than or equal to 80%.

3.1.4.3. …the black water tank measurement is greater than or equal to 80%.

3.1.4.4. …the battery usage is less than or equal to 60%.

3.1.5. The system must be able to go back to the…

3.1.5.1. …the Usage tab.

3.1.5.2. …the Power tab.

3.2. **Usage Tab**

3.2.1. The system must get 100 queries from the database for…

3.2.1.1. …the fresh water tank's measurement information.

3.2.1.2. …the grey water tank's measurement information.

3.2.1.3. …the black water tank's measurement information.

3.2.1.4. …the battery usage information.

3.2.1.5. …the times the measurement information was taken.

3.2.2. When the refresh button is clicked this system should

3.2.2.1. …re-query the database again for 100 inputs.

3.2.2.2. …rebuild the graph with the new information.

3.2.2.3.

3.2.3. The system must build a graph that shows the usage of…
   3.2.3.1. …the fresh water tank over time based on queried data.
   3.2.3.2. …the grey water tank over time based on queried data.
   3.2.3.3. …the black water tank over time based on queried data.
   3.2.3.4. …the battery over time based on queried data.

3.2.4. When the refresh button is clicked this system should
   3.2.4.1. …re-query the database again for 100 inputs.
   3.2.4.2. …rebuild the graph with the new information.

3.2.5. The system must be able to go back to the…
   3.2.5.1. …the Sensors tab.
   3.2.5.2. …the Power tab.

3.3. **Power Tab**

3.3.1. The system must be able to…
   3.3.1.1. …exit the application.
   3.3.1.2. …power down the Raspberry Pi.
   3.3.1.3. …pop-up an application exit warning window.
   3.3.1.4. …pop-up a Raspberry shut down warning window.

3.3.2. The system must be able to go to…
   3.3.2.1. …Sensors tab.
   3.3.2.2. …Usage tab.

3.4. **Alarm Window**

3.4.1. The system will give a warning when…
   3.4.1.1. …the fresh water tank is less than or equal to 20%.
   3.4.1.2. …the black water tank is greater than or equal to 80%.
   3.4.1.3. …the grey water tank is greater than or equal to 80%.
   3.4.1.4. …the fresh water tank is less than or equal to 60%.

3.4.2. The system will sound the alarm when…
   3.4.2.1. …the fresh water tank is less than or equal to 20%.
   3.4.2.2. …the black water tank is greater than or equal to 80%.
   3.4.2.3. …the grey water tank is greater than or equal to 80%.
   3.4.2.4. …the fresh water tank is less than or equal to 60%.
   3.4.2.5. …there has been a change in Sensor.
   3.4.2.6. …if the state of sensors changes after the alarm has been turn off.

3.4.3. The system must state why the alarm was activated.

3.4.4. The alarm must repeat until the alarm window is turned off.

3.4.5. The system must have the ability to…
   3.4.5.1. …turn off the alarm.
   3.4.5.2. …turn on the alarm.

3.4.6. The system must be able to go to…
- 3.4.6.1. …Sensors tab.
- 3.4.6.2. …Usage tab.

4. Non-Functional Requirements – Are the constraints on the system that do not fall under user and functional requirements. The list of requirements that are next presented fall under Organizational Requirements.

4.1. Programming Languages
- 4.1.1. Python3
- 4.1.2. PyQt5

4.2. Development Environment
- 4.2.1. Eric IDE
- 4.2.2. Qt Designer

4.3. APIs Used
- 4.3.1. piplates.DAQCplate
- 4.3.2. sys
- 4.3.3. mysql.connector
- 4.3.4. UI.mainWindow
- 4.3.5. subprocess
- 4.3.6. PyQt5
  - 4.3.6.1. QtCore
  - 4.3.6.2. QtGui
  - 4.3.6.3. QtWidgets

4.4. Other Software tools
- 4.4.1. Photoshop

5. **Diagrams –** a visual representation of how the system will work given use case, sequence, and class diagrams.
   5.1. **Extended Entity Relationship Diagram** – Shows a relation between an entity, usageData, and its attributes, DateTime, Fresh_Water, Grey_Water, Black_Water, and Battery. The primary key for the usageData table is the DateTime attribute.

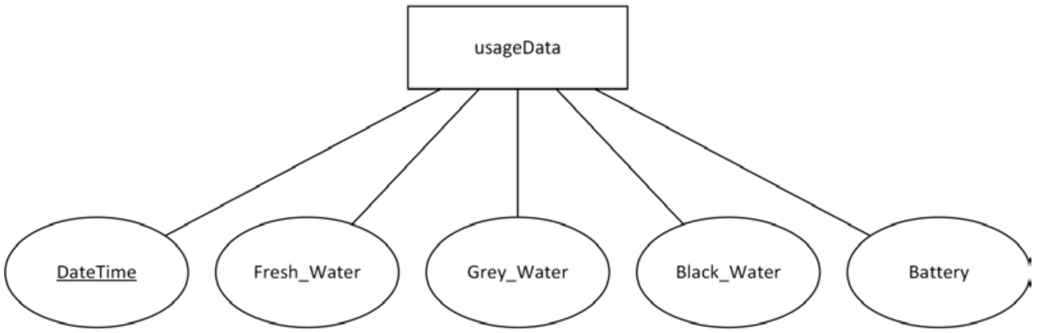


Figure 5.1.1. The EER Diagram of the usageData Entity and the relational model for CamperWaterDB.

   5.2. **Use Case Diagram** - A diagram that shows a possible use case scenario on how the software could be utilized.
      5.2.1. **User Use Case Diagram** – The use case diagram for the system shows how the user interacts with the camper system. The diagram shows what classes are invoked the user interacts with camper system. The database class is invoked when the user presses the refresh button in the usage window. The user can select the power tab, and invoke the sys class (to turn the application off) or invoke the call class (power off the Raspberry Pi).

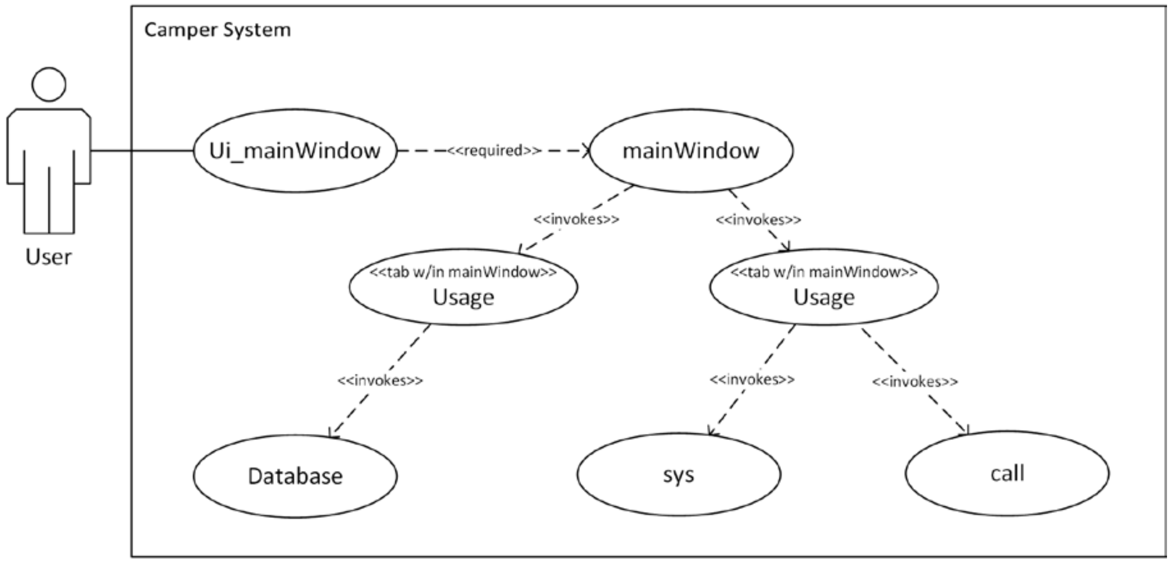


Figure 5.2.1. The use case diagram for the user of the system.

5.2.2. **System Use Case Diagram –** The use case diagram for the system shows that the system first starts with the mainWindow class. The main window class requires that the Ui_mainWindow class to be present to work, since the Ui_mainWindow is the Graphical User Interface (GUI) for the software. The mainWindow first activates a ThreadClass. The ThreadClass requires the use of the classes Alarm, SensorUsageInfo, and Database to be able to perform its necessary functions. The ThreadClass then returns information back to the mainWindow and ups dates the necessary Ui_mainWindow attributes. The mainWindow then accesses information from the Alarm class. It repeats these processes until the software system is turned off.

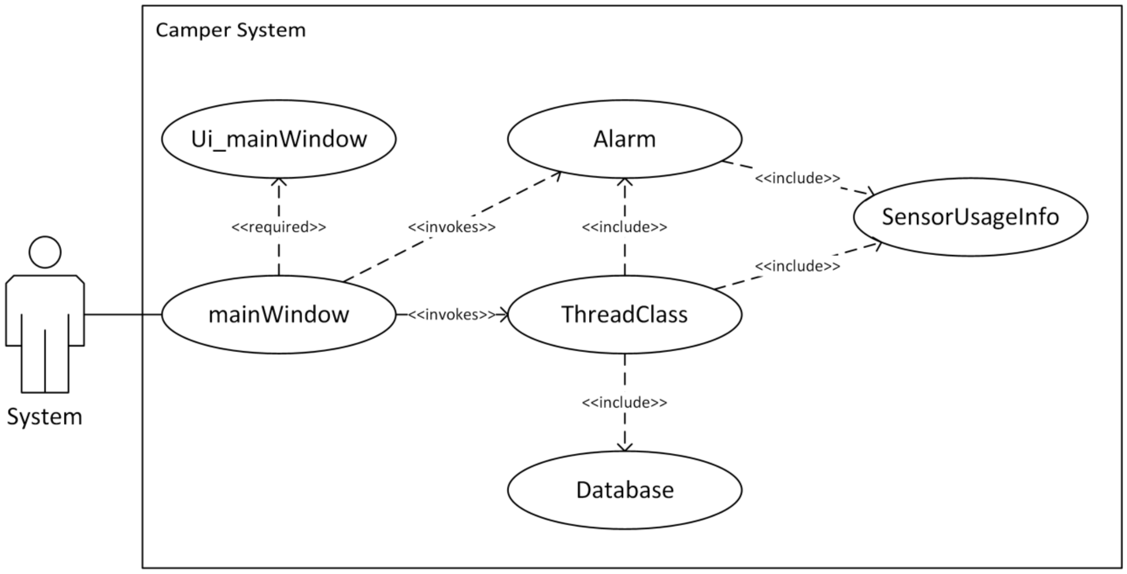


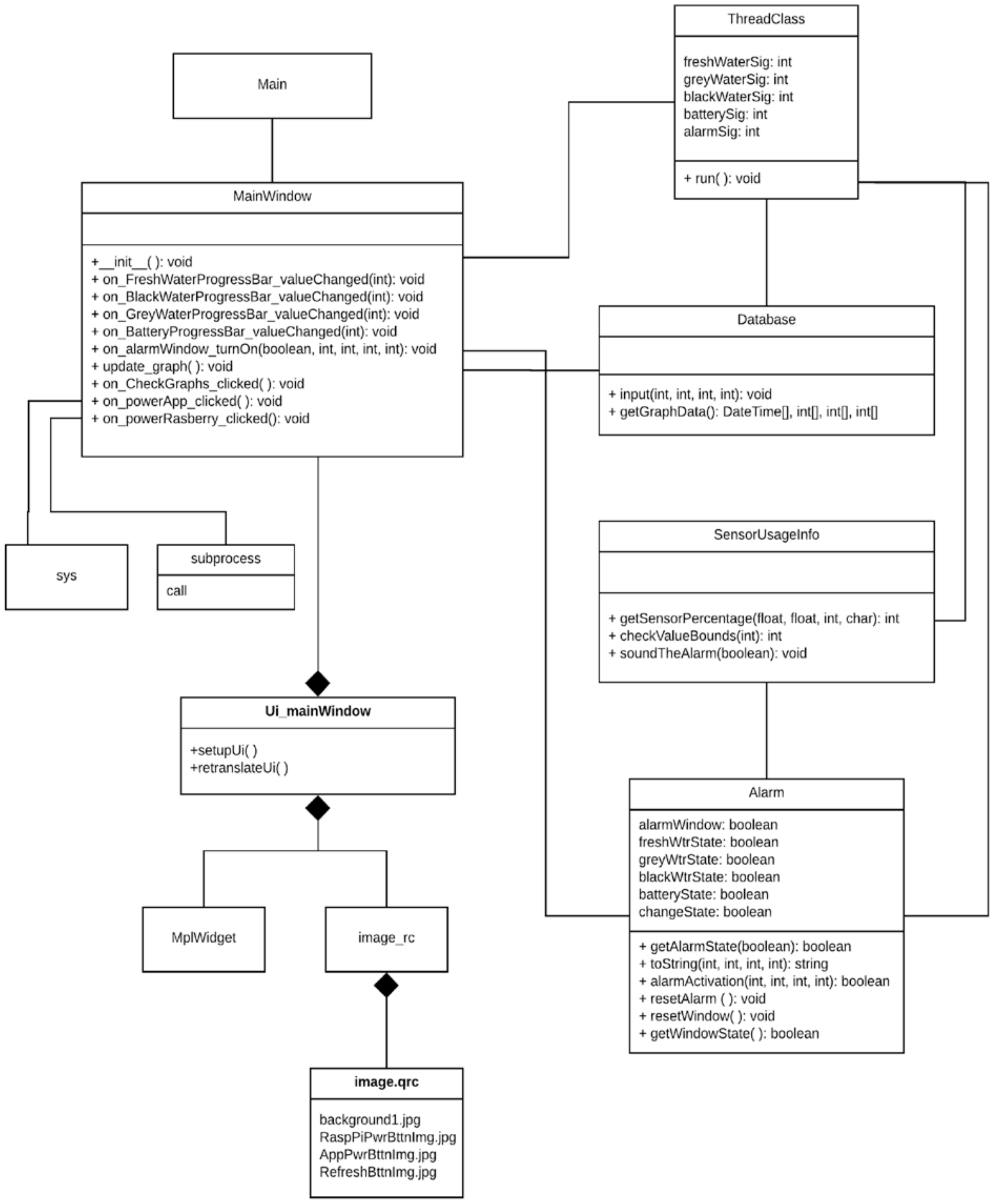
Figure 5.2.2. The use case diagram for system.

5.3. **Sequence Diagram –** A sequence diagram shows the interaction between objects in a class based off of a use case diagram example.

5.3.1. **User Sequence Diagram –** The diagram that shows the interaction between the user and the system as depicted in 5.2.1. can be found as an attached image.

5.3.2. **System Sequence Diagram –** The diagram that shows the interaction between the system and the system objects as depicted in 5.2.2. can be found as an attached image.

5.4. **Class Diagram** – A diagram that shows the relationships between the classes.



Main
MainWindow
+__init__( ): void
+ on_FreshWaterProgressBar_valueChanged(int): void
+ on_BlackWaterProgressBar_valueChanged(int): void
+ on_GreyWaterProgressBar_valueChanged(int): void
+ on_BatteryProgressBar_valueChanged(int): void
+ on_alarmWindow_turnOn(boolean, int, int, int, int): void
+ update_graph( ): void
+ on_CheckGraphs_clicked( ): void
+ on_powerApp_clicked( ): void
+ on_powerRasberry_clicked(): void
ThreadClass
freshWaterSig: int
greyWaterSig: int
blackWaterSig: int
batterySig: int
alarmSig: int
+ run( ): void
Database
+ input(int, int, int, int): void
+ getGraphData(): DateTime[], int[], int[], int[]
SensorUsageInfo
+ getSensorPercentage(float, float, int, char): int
+ checkValueBounds(int): int
+ soundTheAlarm(boolean): void
Alarm
alarmWindow: boolean
freshWtrState: boolean
greyWtrState: boolean
blackWtrState: boolean
batteryState: boolean
changeState: boolean
+ getAlarmState(boolean): boolean
+ toString(int, int, int, int): string
+ alarmActivation(int, int, int, int): boolean
+ resetAlarm ( ): void
+ resetWindow( ): void
+ getWindowState( ): boolean
sys
subprocess
call
Ui_mainWindow
+setupUi( )
+retranslateUi( )
MplWidget
image_rc
image.qrc
background1.jpg
RaspPiPwrBttnImg.jpg
AppPwrBttnImg.jpg
RefreshBttnImg.jpg

Figure 5.4.1. The class diagram of the Camper System.

6. **Visual Prototype –** a visual representation of what the Graphical User Interface (GUI) will look like within the application.

   6.1. **Sensor Tab -** The following is visual images of the prototype Sensor tab. It shows the three water tank progress bars at various stages of fullness, and it also shows the battery life progress bar.

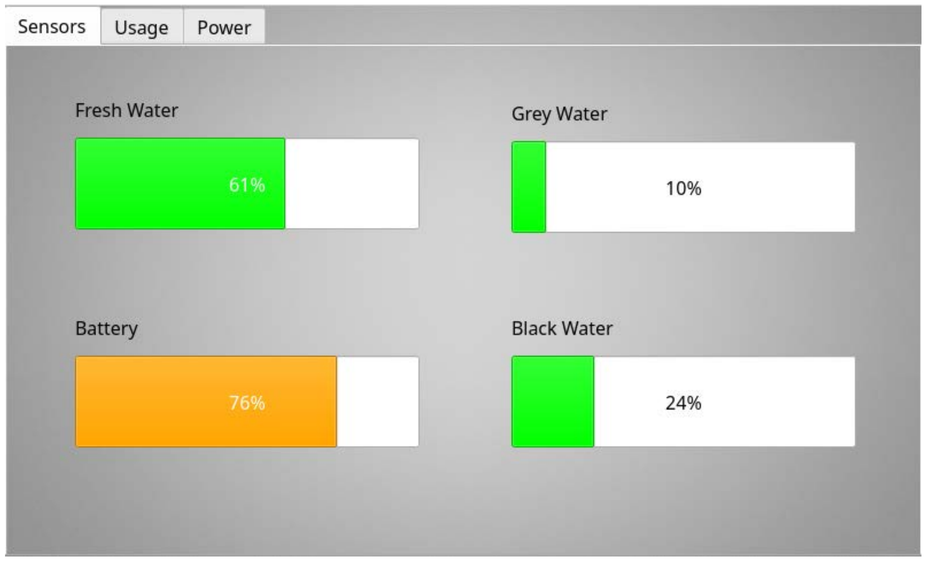


Figure 6.1.1. Shows the various states of the sensors
.

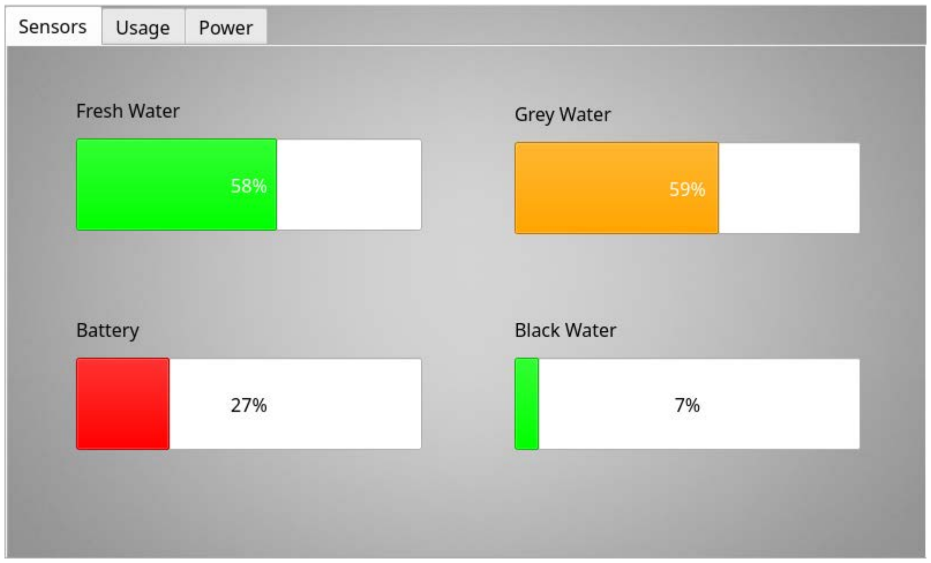
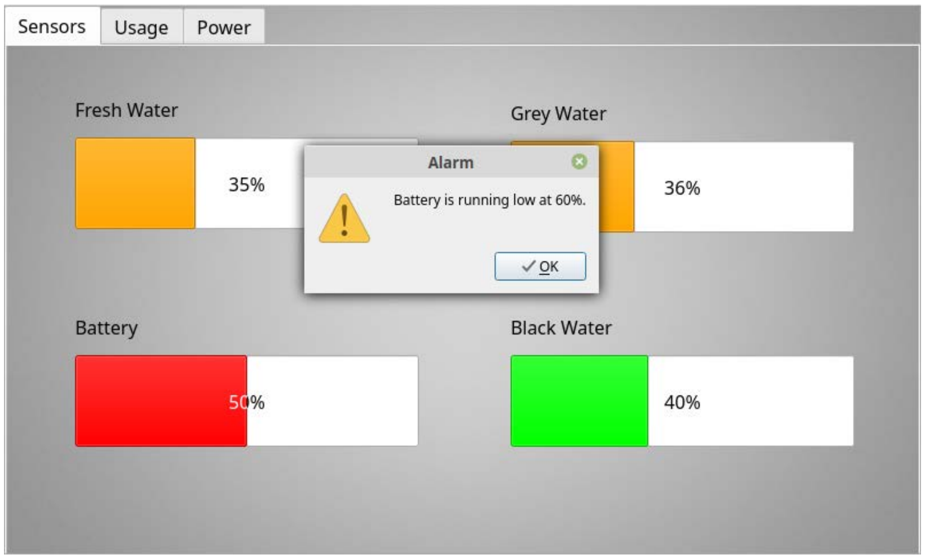


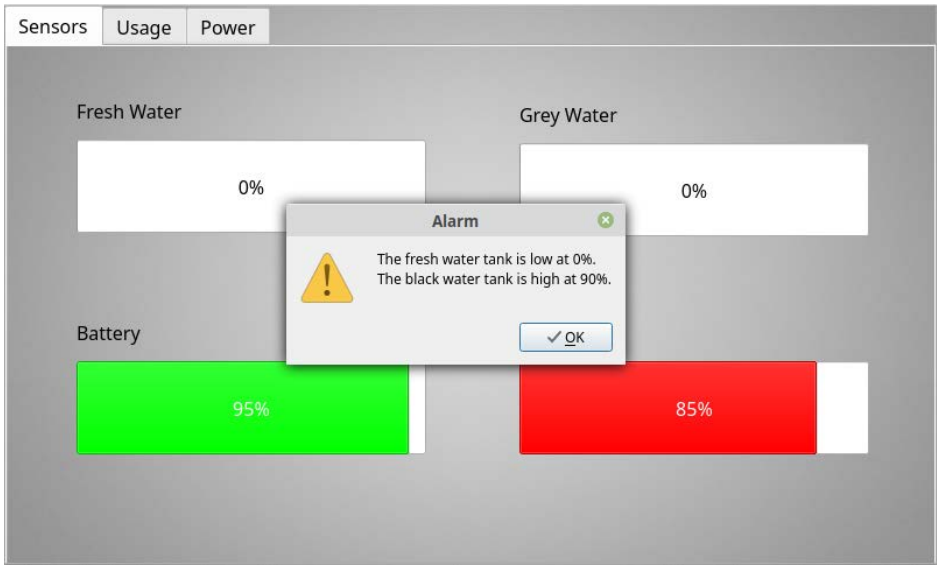
Figure 6.1.2. An example of how the state change affects the color of the progress bars.

6.2. **Alarm Window -** Alarm Window occurs when the grey and black tanks are almost full, the battery is too low or the freshwater tank almost empty. The window displays a string that lets the user know what the issued alarm is for as seen in Figures 3, 4, 5, and 6.



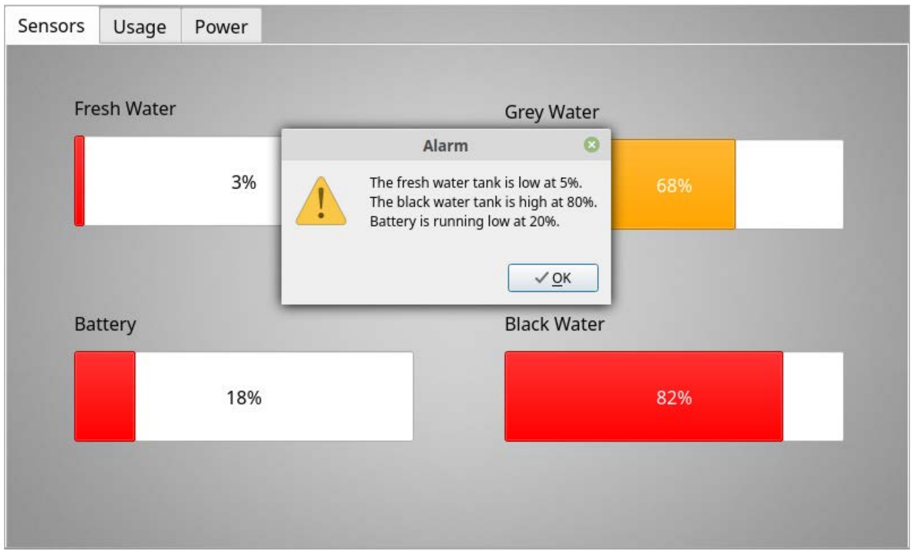
Battery is running low at 60%.

Figure 6.2.1. An alarm warning the user that the batter is running low.



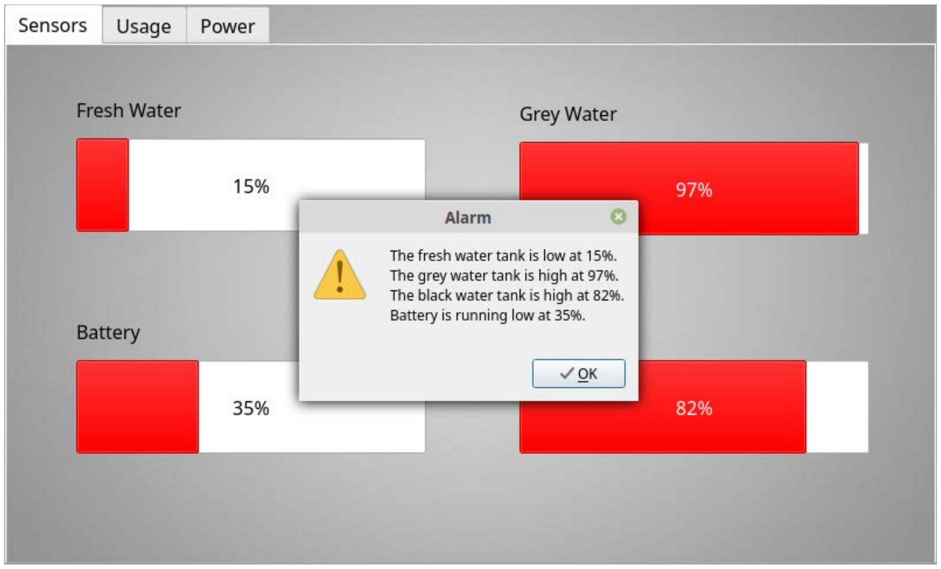
The fresh water tank is low at 0%.
The black water tank is high at 90%.

Figure 6.2.2. An alarm warning the user that the fresh water tank is to low and the black water tank is too high.



The fresh water tank is low at 5%.
The black water tank is high at 80%.
Battery is running low at 20%.

Figure 6.2.3 An alarm warning the user that the fresh water and battery are low, and that they black water tank is too high.



The fresh water tank is low at 15%.
The grey water tank is high at 97%.
The black water tank is high at 82%.
Battery is running low at 35%.

Figure 6.2.4. An alarm warning the user that the fresh water tank and battery are too low, and that the grey and black water tanks are too high.

6.3. **Usage Tab**

The history tab displays a refresh button, which, when clicked, makes a call to the database to gather the last 100 entries in the database and display them on the graph to the left of it.

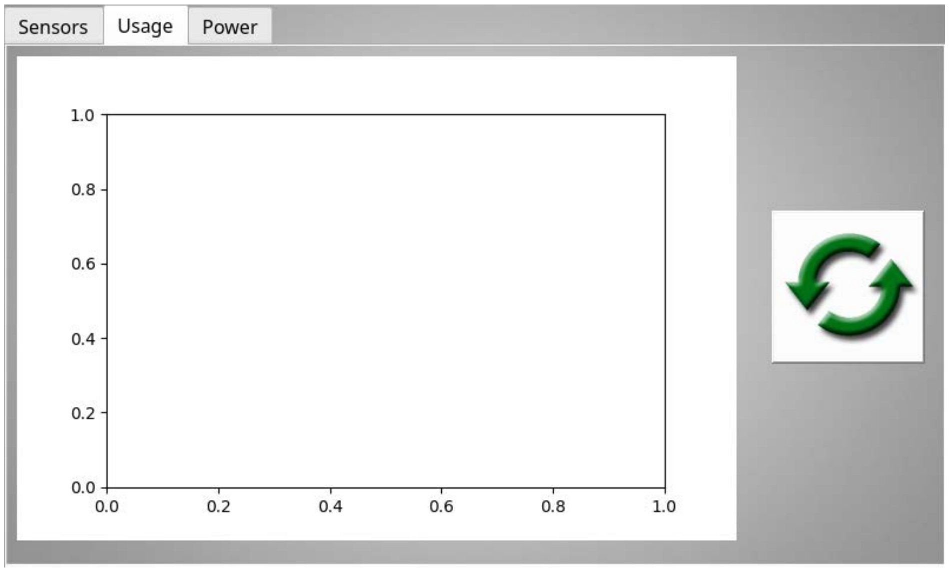


Figure 6.3.1: The graph before it is refreshed

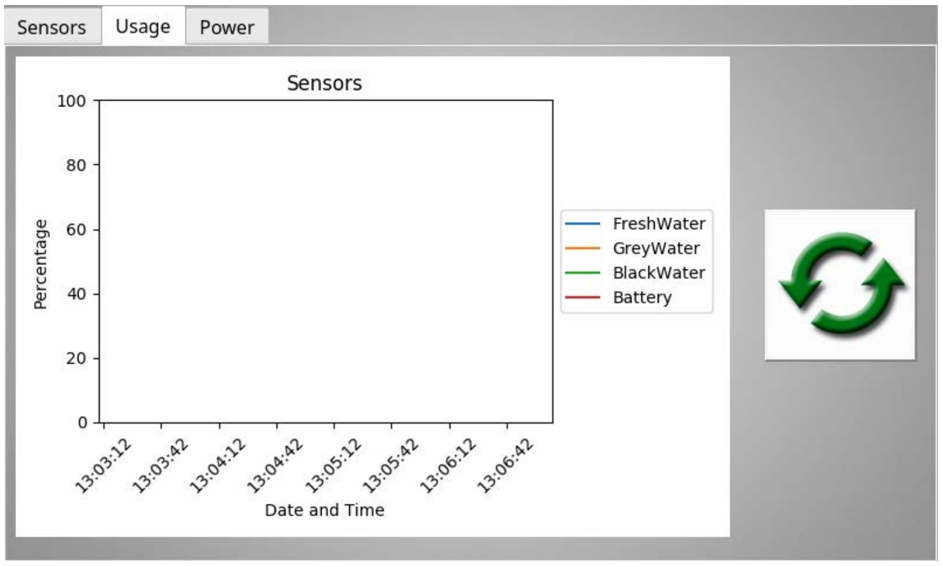


Figure 6.3.2: The graph after a refresh with no information stored in the database

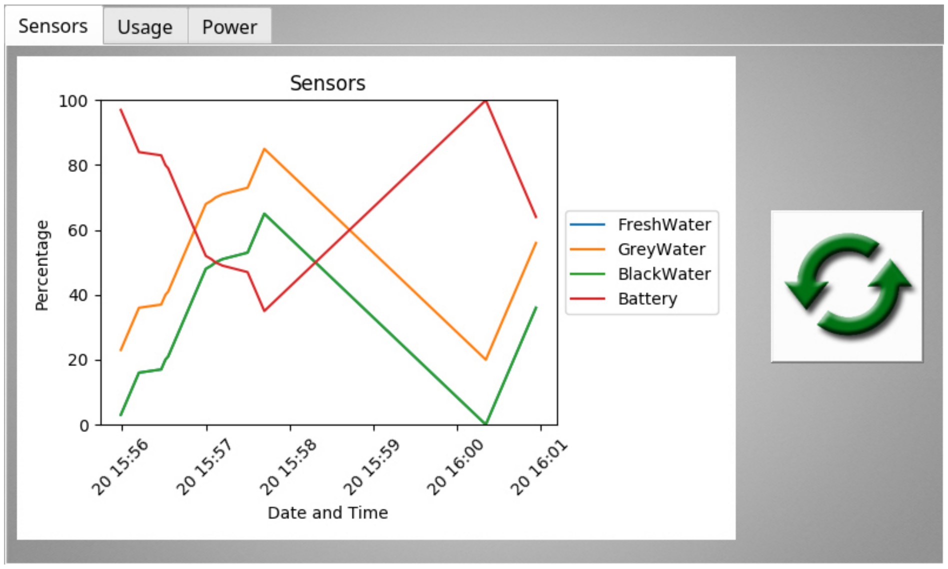


Figure 6.3.3: The graph after it has been refreshed with less than 100 sensor data points from the database

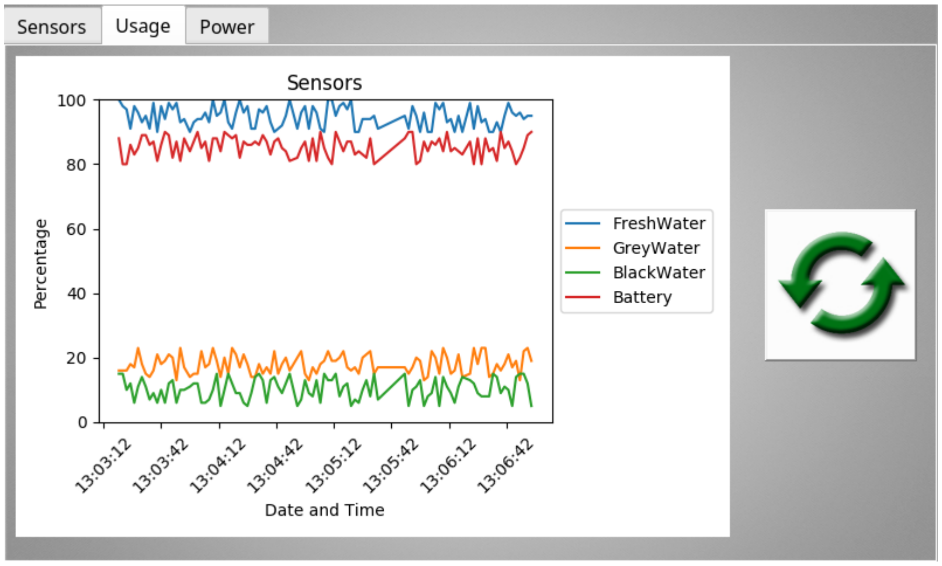


Figure 6.3.4: The graph after it has been refreshed with 100 sensor data points from the database.

6.4. **Power Tab**

The power tab has two buttons, the button on the left closes down the application, and the button on the right shuts down the raspberry pi. Both buttons have their own individual pop-up message, which checks if the user wants to go through with the request.

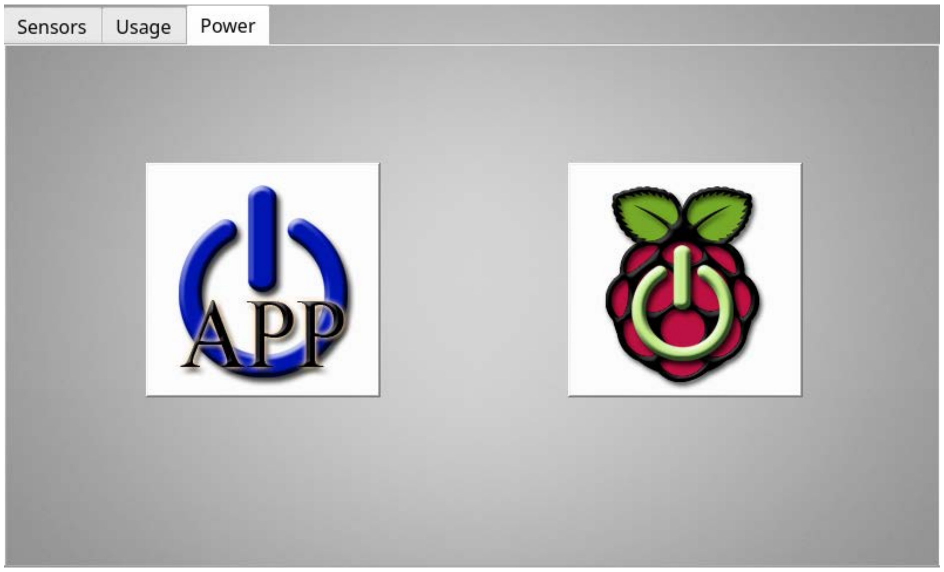
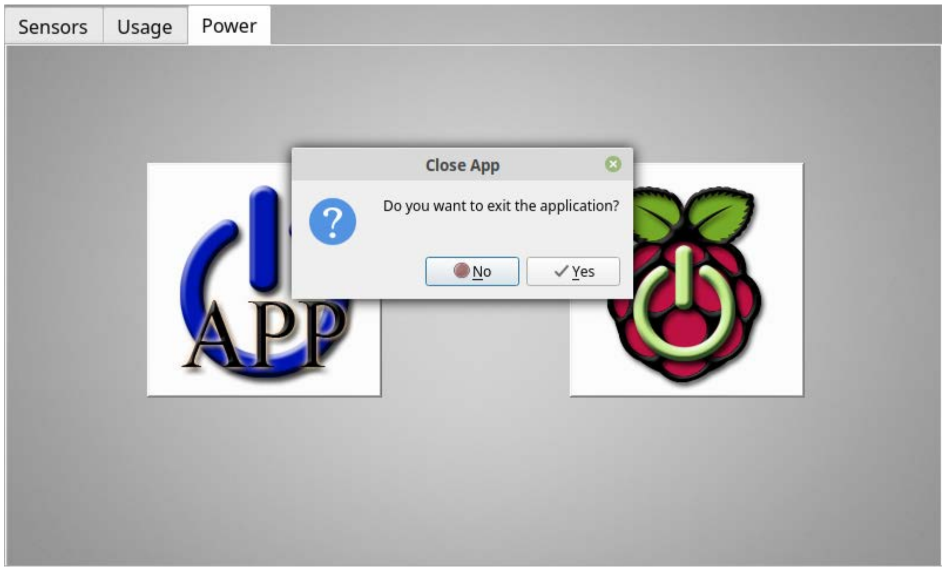
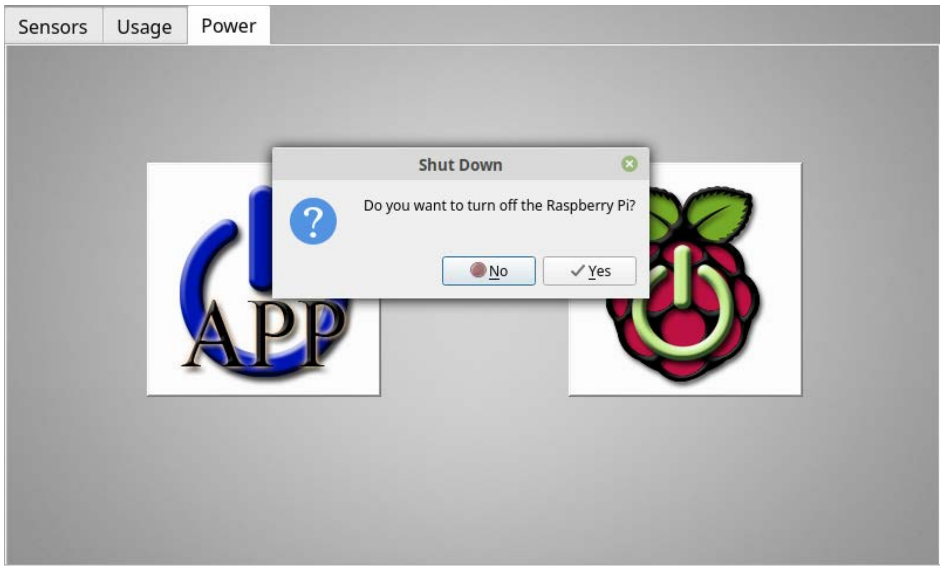


Figure 6.4.1: The power tab, which has two buttons one for turning of the application, and the other for turning off the Raspberry Pi



Do you want to exit the application?

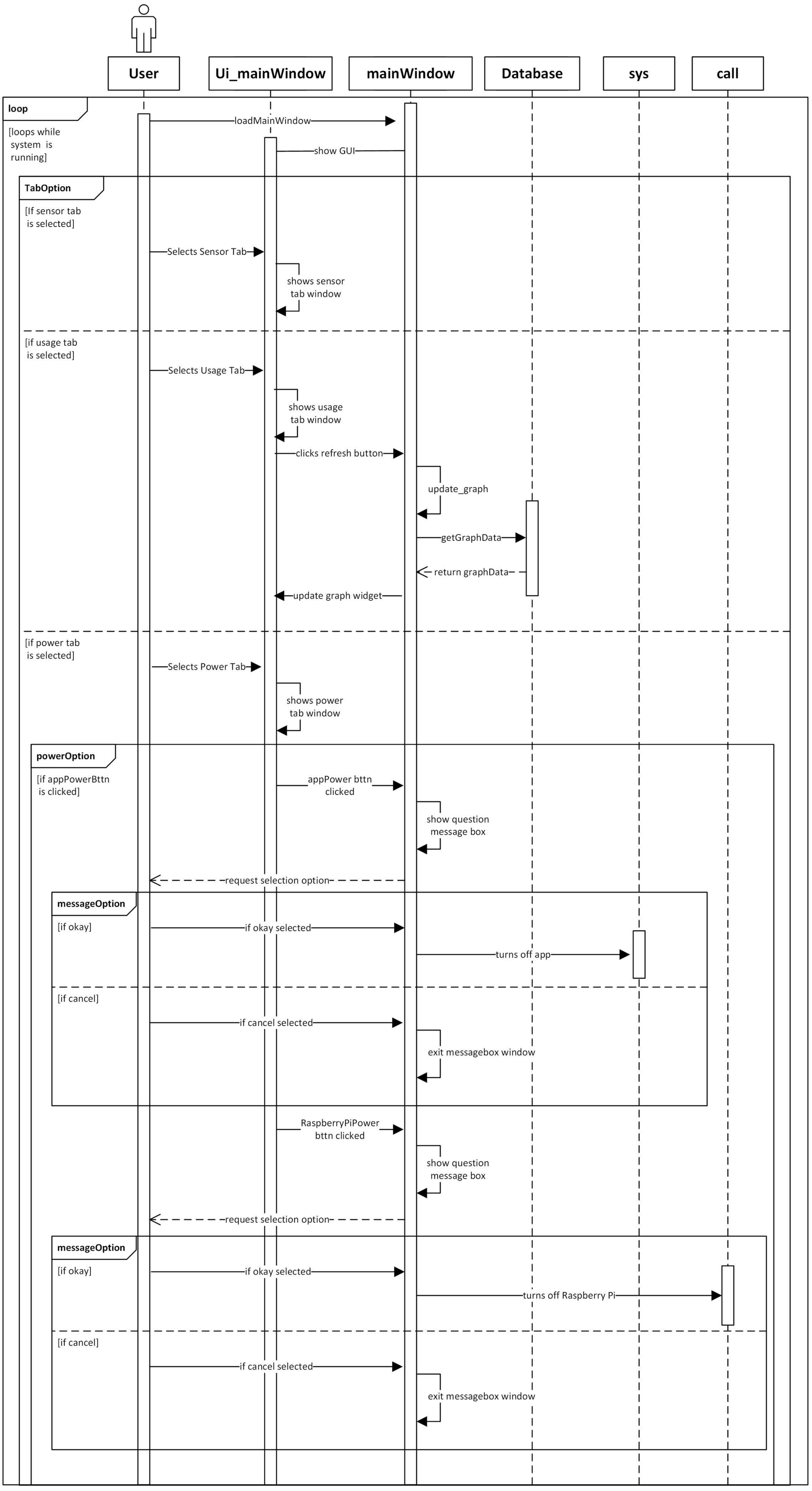
Figure 6.4.2: A window message that asks the user f they are sure they want close the application

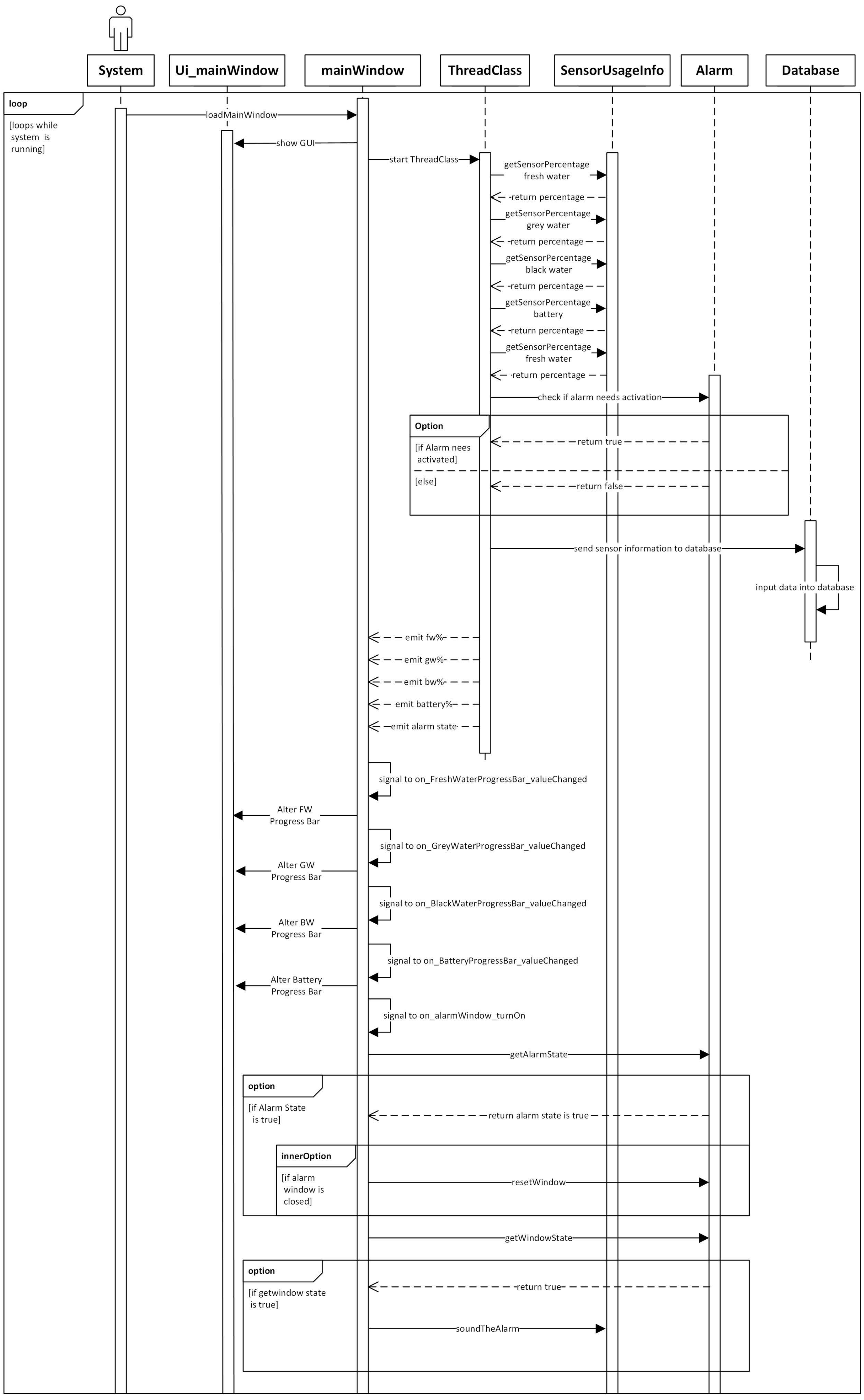


Shut Down

Do you want to turn off the Raspberry Pi?

Figure 6.4.3: A window message that asks the user f they are sure they want to power down the Raspberry Pi.


User
Ui_mainWindow
mainWindow
Database
sys
call
loop
[loops while system is running]
loadMainWindow
show GUI
TabOption
[If sensor tab is selected]
Selects Sensor Tab
shows sensor tab window
[if usage tab is selected]
Selects Usage Tab
shows usage tab window
clicks refresh button
update_graph
getGraphData
return graphData
update graph widget
[if power tab is selected]
Selects Power Tab
shows power tab window
powerOption
[if appPowerBttn is clicked]
appPower bttn clicked
show question message box
request selection option
messageOption
[if okay]
if okay selected
turns off app
[if cancel]
if cancel selected
exit messagebox window
RaspberryPiPower bttn clicked
show question message box
request selection option
messageOption
[if okay]
if okay selected
turns off Raspberry Pi
[if cancel]
if cancel selected
exit messagebox window


System
Ui_mainWindow
mainWindow
ThreadClass
SensorUsageInfo
Alarm
Database
loop
[loops while system is running]
loadMainWindow
show GUI
start ThreadClass
getSensorPercentage fresh water
return percentage
getSensorPercentage grey water
return percentage
getSensorPercentage black water
return percentage
getSensorPercentage battery
return percentage
getSensorPercentage fresh water
return percentage
check if alarm needs activation
Option
[if Alarm nees activated]
return true
[else]
return false
send sensor information to database
input data into database
emit fw%
emit gw%
emit bw%
emit battery%
emit alarm state
signal to on_FreshWaterProgressBar_valueChanged
Alter FW Progress Bar
signal to on_GreyWaterProgressBar_valueChanged
Alter GW Progress Bar
signal to on_BlackWaterProgressBar_valueChanged
Alter BW Progress Bar
signal to on_BatteryProgressBar_valueChanged
Alter Battery Progress Bar
signal to on_alarmWindow_turnOn
getAlarmState
option
[if Alarm State is true]
return alarm state is true
innerOption
[if alarm window is closed]
resetWindow
getWindowState
option
[if getwindow state is true]
return true
soundTheAlarm