The home automation system that we are building has a few central components with several supplementary features. The central software components include the code for the WiFi module, the user-web interface, the database, the server and the communication between all components of the system. These parts are essential because they are interdependent. The failure of any one of these features causes the whole system to fail. Supplementary features include light control, music control, monitoring of door locks, and system control using a mobile application. These features are supplementary because they can fail without affecting other parts of the home automation system. It is important to keep in mind that even though we call these features supplementary, this does not mean these features are unimportant.

Following this distinction, the way we design the acceptance tests for our home automation system follows two phases of “Bottom Up” integration testing. In the first phase, we perform unit tests on the central components. Once the system passes the unit tests, individual components are put into small groups and tested together. We continue combining groups until we test each central component as a whole and finally perform integration testing as we test the components together. After the central components have passed the integration test phase, we begin the second phase. This phase begins with unit tests on the supplementary features. Following the same procedure that was used to test the central components, these supplementary features are tested from the bottom upwards. After each supplementary feature has passed its own “Bottom Up” testing, it is tested with the system of central components in integration testing. Although it was previously stated that the second phase of testing starts after the first phase, we can perform the unit tests for both phases concurrently.

The section below shows plans for unit tests of each class shown in the class diagram. We start by isolating individual functions in a class for testing.

**Unit Tests**

**DeviceController**

addDevice(deviceID); removeDevice(deviceID)

Success:

* The user sends a request to add or remove a device to his or her account, a device ID unique within the user’s account is passed to the function and the function sends the request to the server via communications. In the database, the device is added to or removed from the account.

Failure:

* The user sends a request to add or remove a device from his account and the request is not received by the server.
* The user sends a request to add or remove a device from his account and the device ID generated is not unique within the user’s account.
* The user sends a request to add or remove a device from his account, server receives the request, but the device ID is not stored in the database.

updateDeviceStatus(deviceID, deviceStatus)

Success:

* The device controller sends a message to the server via communications. This message asks the database to update the status of the specific user’s device.

Failure:

* Some or all of the message sent is lost during communications.
* The message is sent and received, but the wrong user account is updated.
* The message is sent and received, but the wrong device is updated.
* The message is not sent.

getDeviceStatus(deviceID)

Success:

* The device controller gets the status of a device from the Arduino via data from communications.

Failure:

* The device controller gets the correct device, but the wrong status.
* The device controller gets the status of the wrong device.
* The device controller gets a null status for a device that exists in the user’s account.
* The device controller gets a status for a device that does not exist in the user’s account.

requestConnection(deviceID)

Success:

* The device controller receives a connection request from the Arduino via the device interface module. The controller then sends the request to server via communications and waits for a boolean reply. The function returns the boolean reply.

Failure:

* The controller receives a connection request, but does not send it.
* The controller sends a connection request when no request was received.
* The controller receives and sends a connection request to server, but never receives a reply and is stuck waiting.
* The reply is not boolean.

**DeviceInterfaceModule**

turnOn()

Success:

* The user presses a button on the device interface to turn on a specific device. The button changes appearance to indicate to the user its current status. The user’s input is sent to the Arduino and the device corresponding to the button pressed turns on.

Failure:

* The user presses a button to turn on a specific device and the signal is not sent to the Arduino.
* The user presses a button to turn on a specific device and the signal is sent to the Arduino, but the device does not turn on.
* The user presses a button to turn on a specific device and the signal is sent to the Arduino, but the wrong device turns on.
* The button does not change appearance after being pressed.

turnOff()

Success:

* The user presses a button on the device interface to turn off a specific device. The button changes appearance to indicate to the user its current status. The user’s input is sent to the Arduino and the device corresponding to the button pressed turns on.

Failure:

* The user presses a button to turn off a specific device and the signal is not sent to the Arduino.
* The user presses a button to turn off a specific device and the signal is sent to the Arduino, but the device does not turn off.
* The user presses a button to turn off a specific device and the signal is sent to the Arduino, but the wrong device turns off.
* The button does not change appearance after being pressed.

getData()

Success:

* The device interface module retrieves data from the Arduino.

Failure:

* The Arduino sends data, but the device interface module cannot or does not retrieve the data.
* The Arduino does not send data, but the device interface module retrieves data that is not null from the Arduino.
* The Arduino has data and the device interface module can retrieve data from the Arduino, but the data is corrupted.

sendData(data)

Success:

* The device interface module sends data to the Arduino.

Failure:

* The device interface module sends data, but the Arduino cannot or does not receive the data.
* The device interface module does not send data, but the Arduino receives data.
* The device interface module sends data and the Arduino receives data, but this data gets corrupted.

getMessage()

Success:

* The device interface module gets a message from a queue of messages sent by the Arduino. This message may be code, a signal or an error message.

Failure:

* The message retrieved is corrupted.
* The device interface module gets a message with content when the message queue is empty.
* The device interface module does not get a message when there are messages in the message queue.

sendMessage(message, deviceID)

Success:

* The device interface module sends a message to a queue of messages for the Arduino. This message should only be code.

Failure:

* The message sent is not code.
* The message is not queued.
* The message is not sent.

setDigital()

Success:

* The device interface module sets a digital value as a parameter for the Arduino. This digital value is used to change a device’s output.

Failure:

* The device interface module does not pass the digital value to the Arduino.
* The device interface module passes the digital value to the Arduino, but the device’s output does not change.
* The device interface module passes the digital value to the Arduino and the device’s output changes, but not in the correct manner.
* The device interface module passes anything that is not a digital value to the Arduino.

requestConnection(deviceID)

Success:

* The device interface module passes a connection request from the Arduino to the device controller.

Failure:

* The request from the device interface module is not sent to the device controller even though a request was sent by the Arduino.
* The request is sent, but not received.
* The request is sent with the wrong device ID.
* The request is sent to the Arduino.
* A request is sent to the device controller when no request was sent from the Arduino.