**Phase1**

In this phase we developed a home security app on Android Studio. This app interacts with the LAMPP server using PHP server scripting. Note:-LAMPP server is on a different machine. The server machine and the mobile app are connected using a router(NetGear). The mobile app comprises of the following modules.

1. Security System
2. Thermostat
3. Lights
4. Locks
5. Door/Window sensors
6. Motion Sensors
7. Garage Doors
8. Energy
9. Weather
10. Video

Note: -Each module mentioned above stands for a xml layout and a corresponding java class.

**Android Studio:**

All the above modules fetch the latest data from the server (XAMPP mysql) and display in their respective Radio Buttons, Seekbars, etc. Finally, we use the functions like “OnClickListner” to listen to the input from the mobile app user and update it to the server. Note: This fetching of latest data happens every time the app module is opened and hence we can see the latest/updated data. Only difference is in thermostat module.

In thermostat module, we fetch the current temperature from the server but we set the control temperature to the server. This is because the thermostat simulation is running on Rpi. This simulation linearly changes the current temperature according to the control temperature.

**JAVA (Eventually to deploy on Rpi):**

For all the above modules we wrote corresponding java simulation. These simulations simulate the different sensor actions. For instance if my thermostat is set at cool mode and current temp is 67 F but, the control temp is 55 F. This simulation will be running and the current temp will decrease with respect to time (Linear relationship). We also simulate the actuators (i.e. control switches & manual regulators) by giving the opportunity to set the sensor switch values at runtime. Moreover, the values entered at run time are considered as hardware actuations and are updated in the server database as the latest entries.

**Server Scripting (PHP):**

We wrote different PHP to provide required functionality (Fetch/Set from/to database on server machine). Finally in phase 2 we replace all the communication between server and client with socket communication.

**Phase2**

In this phase we extended the java simulation boundary further to Rpi board. Now, All the sensor simulations will be running on Rpi board. Moreover, we established socket communication between Rpi board Server. In addition to that now we have a local database on Rpi board which exclusively stores the data of last 24 hours and is in completely in sync with the main database on the LAMPP server. Webcam module was also Implemented in this phase. We also interfaced GPIO pins of the Rpi board to serve as event notifiers (i.e. so if the lights are ON the led turns on)

When the java simulation fetches the value from main server it will act as a client. Alternatively, When the java simulation inserts the updated value, it acts as a server and the main server acts as a client. In the first case java side is the client and PHP script is server and this is exactly opposite for the second case.

In this phase, we used pi4j library in java to interface and trigger the GPIO pins. The change in GPIO pins change the state of LED’s.

In webcam module we use vlc media player on a separate machine(This machine just acts as a separate camera device in network) to record a video with a specific video codec(H.264). Then we transfer this video using there SFTP protocol to the shared server folder (htdocs). Now, we can access this video feed from the rpi board using a bash script(separate video device in house) or in the mobile app from the video module.

**Phase3**

In this phase we implemented network mapper (Nmap). Main aim here was to make the network as dynamic as possible. Moreover, we also implemented complete 2-way handshaking between the devices and server. So, if any new device enters the network handshaking is performed in which the data (ip address & roles) of all the devices is sent to the new device and acknowledgement is received. The usage of server & client ip addresses in the code has been made dynamic (using the packet data). We use Nmap to scan the network and transfer the data in a text file and then, transmit this data as a part of handshaking.

**Code Flow:**

Java Code Flow (Rpi):

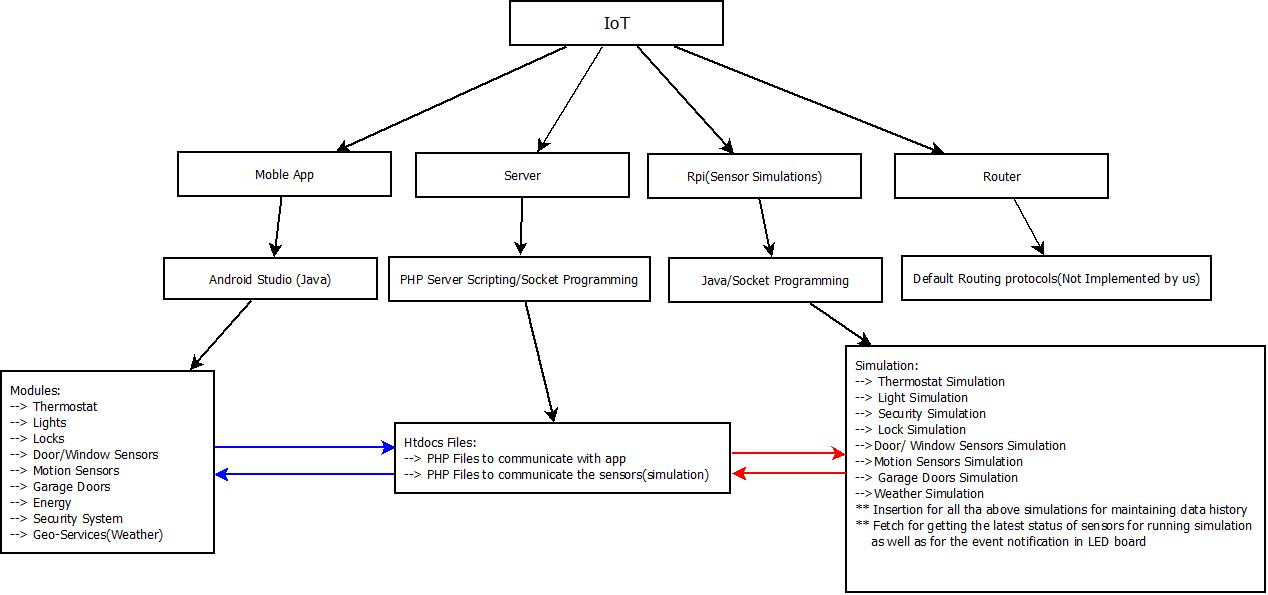
* Fetch the status of all the sensor modules from the LAMPP mysql database.
* Provide the sensor states/settings at run time.
* Start the simulation threads for all the sensor modules (runs parallelly) which inserts (into database) data continuously to provide the history feature.
* Rpi Event Notifier:
  + Fetch current sensor data continuously (infinite loop)
  + Use the pi4j functions to set the GPIO pin states and reflect the status on LED’s

**Android Studio:**

* Starts with the login module. Also, can register as a new user.
* Establish container to be used with the navigation drawer.
* Set the entry in the container according to the module clicked by the user on the navigation drawer.
* Inside the fragment class of the clicked module:
  + Fetch the latest data from the mysql server and reflect it on the display(Display may consist of radio button, seekbar, text etc.)
  + Keep listening to the user input from the screen(methods such as OnClickListener).
  + Set the listened data (such Seek bar value) to the main server.

**Note:** PHP has no code flow, these files are called according to the usage, either from the Java files (Android & Simulation) or Bash Scripts.

**Code Skeleton:**



Here, Red arrows signify Socket communication between Server and sensor simulation(Rpi) and Blue arrows signify the communication between the Mobile App and Server