# AUSGRE BRIEF:

ausgre stands for Automated Smart Greenhouse. This is a greenhouse system that

1. Monitors and Regulates the Greenhouse System.
2. Records and Saves the greenhouse Data
3. Presents greenhouse data in a visual format using various platforms
4. Communicates with the user and vice versa

# LAYOUT:

The basic system will have two main hardware components.

1. Raspberry Pi B+ Board
2. Arduino UNO R3

The main board will be the Raspberry which will be used to slave our micro controller which is the Arduino Board.

The raspberry will be the brain of our system. It is a mini computer on its own that runs on the Linux OS. The Arduino is the best for control actuators since it has readily available libraries for the sensors and actuators that are to be used.

The Arduino connects to the Raspberry pi through a USB cable. A python language library called Nanpy will be used to communicate with the Arduino as explained in this case <http://www.akeric.com/blog/?p=2420&cpage=1#comment-204276> . Another library called serial.py will be used for serial communication using the USB Cable.

# THE SYSTEM:

The System shall have various actuators and sensors connected to the boards. The user shall set the conditions to be maintained and the system shall maintain those conditions by taking corrective measures anytime that the readings go outside the set range.

A User shall be able to control the system from a User Interface that shall be available on different platforms.

1. Mobile(Android)
2. Web Portal
3. Local Interface ( Touch Screen )

A user shall be able to set whatever conditions they need to be maintained and also be able to manually override the actuators actions using the User interface. It shall also provide visual representations of the data

The system should be able to log and send notifications on actions and conditions to the user. This shall be supported by Push Notifications, Email or SMS.

## **Interfacing the Arduino with the Raspberry Pi:**

This can be done by slaving the Arduino to the Raspberry using the Nan.py Python library. Communication and exchange of data between the two shall be facilitated by serial.py Library.

## **Control of the System Wirelessly:**

The Raspberry will host a NodeJS application that will contain the User Interface and it will be hosted on a local setup server on the Raspberry board. This means that the Raspberry has to be connected to the internet for it to be accessible on platforms except for the local one.

Each System will have a unique IP address that can be reached from the internet, and it is this address that will be used to access the system from both mobile and web platforms through a browser. NodeJS will communicate with the python shell using the extrabacon python library (<https://github.com/extrabacon/python-shell/blob/master/README.md>) .

Therefore NodeJS can be used to control both boards actions using python.

## **Database Requirements**

The System should keep a local copy of the of the data and also sync the data with an online database at a small interval. This is so that theres an online backup just incase of dataloss. And incase of a reset. A client should be able to recover all the lost data. This is important since it will be used for analytics in the long run. This should be a module on its own

## **Power Requirements.**

The system will be fully based on a solar power system because its important to make it self sustainable. An explorable option is the use of turbines

## **Packaging**

The system should have a custom packaging that hides all the devices on the inside, this includes a screen that shall be used to control the system

## **Communication**

The System main methods of communication shall be through

1. Email
2. Push Notifications
3. SMS

The shall mean including an email to SMS service or getting a GSM module for clients without wired access to their locations.