[CS 308 Project Report]

Smart Greenhouse

(Tutorial @ http://youtu.be/UuTVauUsmHI)

Team **TU-02** **f005ba11**

Saswat Padhi 100050061
Harshita Meena 100050020
Sanket Totala 100050011
Prateek Agrawal 100050025

Table of Contents

1 Introduction

The broad aim of the project is to lay down a basic model for automating a greenhouse using technologies which would be affordable to most Indian farmers.

With the recent increase in the number and popularity of greenhouses, their effective maintenance becomes a critical issue. This product aims at automating the maintenance of a greenhouse and allowing a user to monitor and control it remotely. The product further not only maintains the crop but also improves the overall productivity. It monitors various environmental factors inside the greenhouse and accordingly triggers actuators, so as to provide optimal growth environment to plants.

2 Problem Statement

2.1 Product Perspective

The project aims to automate the functioning of greenhouse such that the plants are well taken care of even when the greenhouse is not visited and maintained for days. Thus, the greenhouse must be able to self-balance the humidity, temperature, light intensity and other such factors without manual human intervention. It not only provides an automated control mechanism, but also provides a mechanism for the user to to override the automatic control. It provides the user an almost completely tool to manage his greenhouse remotely from any location.

2.2 Product Functions

The product monitors and controls the following aspects of the environment within a greenhouse:

- Atmospheric temperature
- Ambient light intensity
- Atmospheric relative humidity

It has a web dashboard which provides a live camera feed, current sensor readings for light, temperature & humidity and methods to control the same. Also, a complete activity log is displayed on the dashboard, listing the events that took place in the greenhouse. This gives the user complete power over the greenhouse, even when he is not present in the greenhouse itself.

2.3 User Characteristics

The intended user of this product is an owner/maintenance incharge of a greenhouse. This product allows the user to monitor the greenhouse remotely from anywhere with internet connectivity. Along with monitoring, it also allows the user to fine-tune several parameters of the greenhouse by allowing him to override the in-built environmental automation. The product is designed so as to be affordable by most Indian farmers. It is quite inexpensive, power efficient and can be easily manufactured in large quantities.

2.4 Constraints

- The model is susceptible to generalization of local effects. Currently we only place 2 RPi modules at the extreme ends of the longer side of the greenhouse. So, local adversities at the extreme points would be generalized for the entire greenhouse. For example,
    - If water vapour build up at the extreme ends of the greenhouse as it is at the border of the internal and external environment; the RPi modules would conclude that the internal humidity is too high.
    - Similarly for light intensity; if the greenhouse is partially lighted only at the extreme ends, the RPi modules would be fooled into believing that there is sufficient ambient light.

  This constraint can be solved by placing multiple sensors at strategic locations.

- The electrical components within the greenhouse would be exposed to the environment and may be damaged soon if they are not properly insulated from the external factors.

2.5 Assumptions and Dependencies

➔ In this prototype, we assume that the total power drawn by all sensors can be provided by the Raspberry Pi board itself. This is not scalable and once enough number of sensors are required, external power has to be provided to drive the sensor and actuator modules.

➔ Also, the Raspberry Pi is not a very powerful board. We went for this board keeping the total budget in mind. RPi is inexpensive but a very basic CPU. One can go for a more powerful CPU which could make features like real-time feed from camera, or having more than a dozen sensors on same board or being able to have hundreds of clients monitoring the greenhouse; plausible.

➔ We also assume that the environmental conditions within which the greenhouse operates are typical to India. The sensors have been chosen so that their limits handle the range of variations (for example in temperature), we see within the Indian climatic conditions. For usage of this model outside

India, some of the sensors might have to be replaced by others so as to properly detect and handle the variations there.

3 Requirements

3.1 Functional Requirements

The functional requirements for this project include both monitoring and control of the following parameters inside the greenhouse:

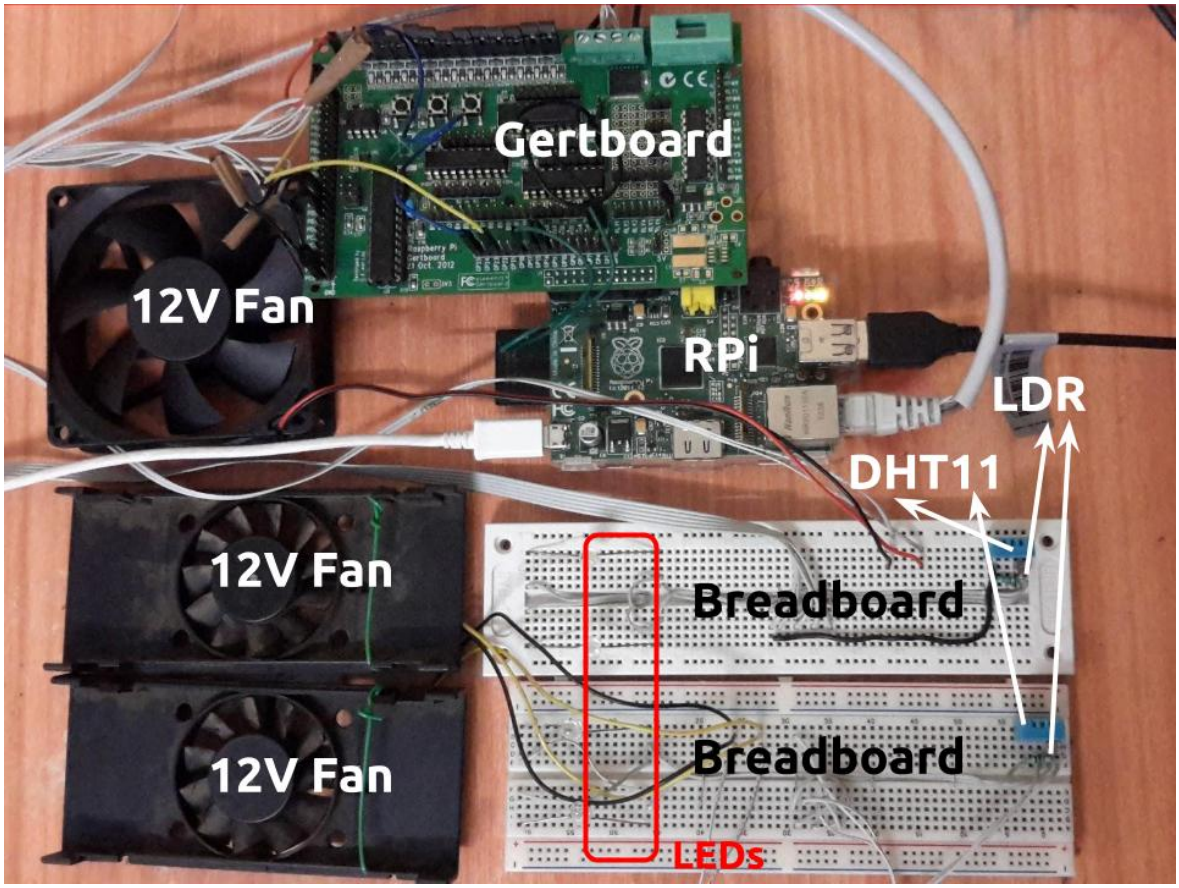
- *Light Intensity:* Low/no light hampers plant growth. The product will trigger growth lights during night/cloudy days to enhance productivity. This would also make the greenhouse visible on the live feed allowing the user to check the greenhouse even while it's dark.
- *Temperature:* Optimal temperature is crucial for photosynthesis. The product will trigger fans or cooling systems to ventilate the greenhouse.
- *Humidity:* Affects photosynthesis and water loss. The product triggers fans or ventilating systems which let the humidity balance with the outside humidity. Using a humidifier or mistifier is another alternative.

3.2 Non-Functional Requirements

- *Performance:* Sensors should have *minimum possible response time* to ensure close to real-time automation of the greenhouse control systems.
- *Efficiency:* The power consumed by the entire circuit should be minimum as the cost for the same would be an indirect recurring investment for the farmer.
- *Maintainability:* As most of the sensors and control systems would be directly in contact with the environment, they should be well protected from wear and tear and should be easily replaceable in case of failures.
- *Accessibility:* Considering the level of technological awareness among the target consumers, the dashboard and the greenhouse control systems should be as user-friendly and intuitive as possible.
- *Price:* Keeping the Indian agricultural scenario in mind, the cost for the entire setup should be reasonable enough to be affordable by most Indian farmers.
- *Resilience:* Even in case of adverse conditions or lack of attention for some period of time; the greenhouse should be able to maintain itself in an acceptable state.

- *Open Source:* Knowledge belongs to everyone.

3.3 Hardware Requirements



- *LDR (Photo Sensor):* This is an analog sensor used to detect the amount of light falling on the greenhouse and accordingly control the environment for optimum growth. We use 2 small 5mm LDRs for this project.

- *Temperature and Humidity Sensors:* This is a digital sensor which reads the temperature and humidity values of the greenhouse. We use 2 DHT11 sensors.

- *LEDs:* The LEDs are used to light up the greenhouse when there is low light intensity, as indicated by the LDR sensor. We use multiple (max 4) LEDs.

- *Gertboard:* This is an extension board designed for RPi. This powerful board has an ATmega 328p processor, a level convertor and a motor driver among other circuits. The motor driver and level convertor are used to drive fans.

- *RPi:* Raspberry Pi is the CPU for this project. It fetches the temperature, light and humidity values from the Gertboard and takes actions (like turning on the fans and LEDs) according to the values. It also runs a web server which is used to monitor the greenhouse remotely. Live feed of camera and sensor values are given by the dashboard. It also includes the functionality to adjust the threshold values of the sensors, and also the activity log of RPi.

- *Fans:* The fans are used to control the temperature and humidity of the greenhouse. They are turned on when the warm, dry air of greenhouse needs to be thrown out and outside fresh air needs to be thrown in.

- *Camera:* In order to monitor the condition of the greenhouse, a camera is connected to give a live feed.

- *SD Card:* The RPi operating system resides on an SD card.

- *Adapters (3.5 and 12 V):* These chargers are required to supply power to RPi (3.3V) and fans (12V).

- *Breadboards:* The basic circuit containing sensors, fans and LED's is build on it which is further connected to gertboard.

- *Resistors:* Several resistors are required to be connected along with the LEDs and LDRs.

3.4 Software Requirements

- *Raspbian:* A debian derivative operating system, which drives the RPi board.

- *wiringPi:* wiringPi allows one to directly read or write digital or analog values to the RPi / Gertboard GPIO pins.

- *nginx server:* A light-weight http server which is the backbone of the user interface.

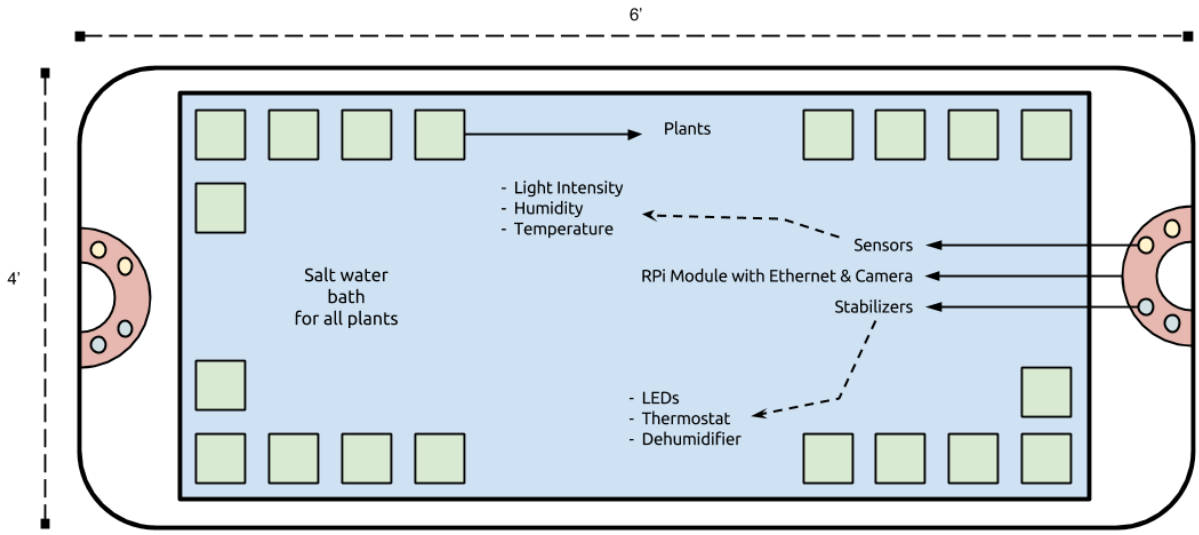
- *PHP FPM:* FPM (FastCGI Process Manager) is a variant of the original PHP server; with much more additional features and lesser memory footprint & CPU usage.

- *Twitter Bootstrap and jQuery:* These are third party frameworks, which provide easier alternatives to implement fully responsive UIs which can be accessed on devices of all resolutions: PCs, tablets, phones etc.

- *Arduino:* The Arduino software is to program the ATmega processor of gertboard and for communication of the gertboard and RPi.
- *fswebcam:* fswebcam is required to capture live feed from the camera.

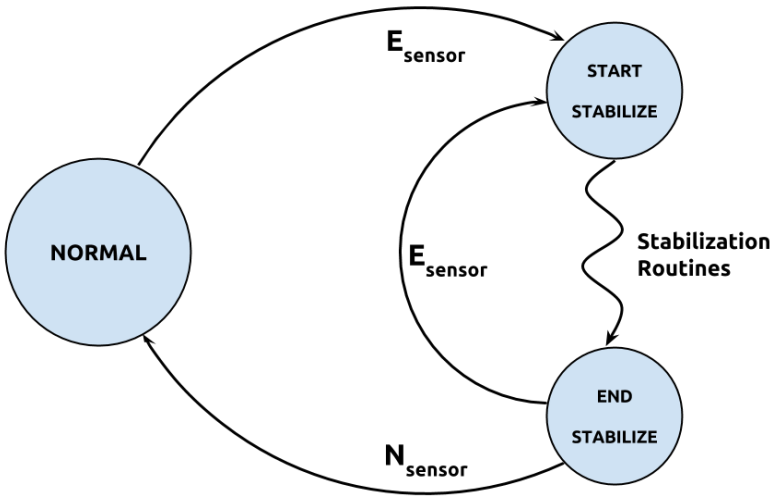
4 System Design

Design Blueprint



FSM for Automation

There are 4 FSMs that are active concurrently on both the RPi modules. Each of these FSMs are responsible for monitoring and controlling one specific aspect of the greenhouse environment. So, there are 4 FSMs each for pH, Light, Humidity and Temperature which behave as shown in the template below:

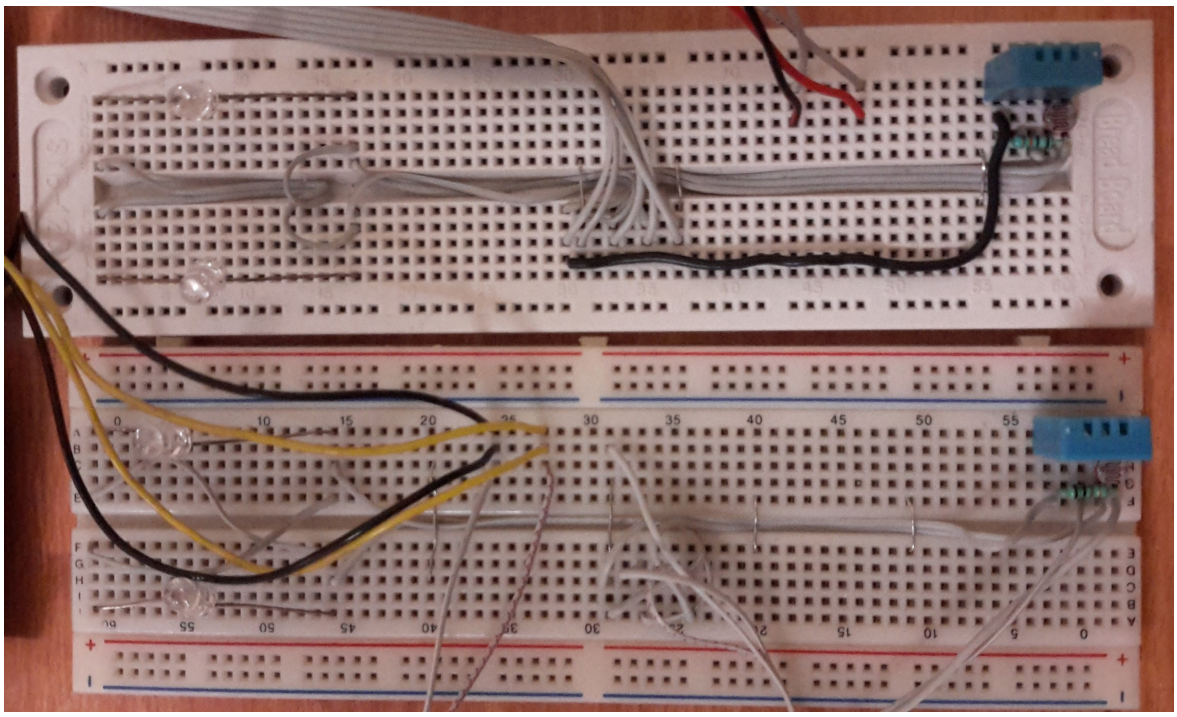


In the FSM, **NORMAL** is typically the expected state of the sensor whose value is within the acceptable range. $\mathbf{E}_{\mathbf{sensor}}$ and $\mathbf{N}_{\mathbf{sensor}}$ are triggers that are fired by the monitoring routines on RPi when they detect the value of sensor going beyond the acceptable range or within the acceptable range, respectively.

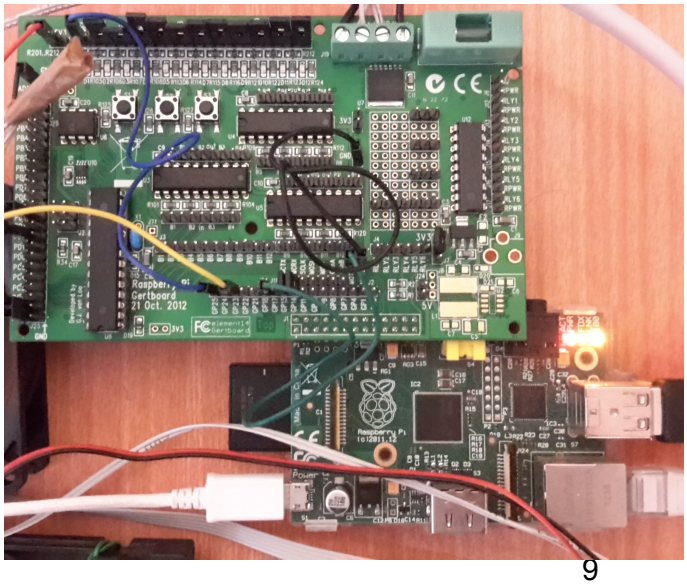
START_STABILIZE is the state which indicates that a sensor value is critical and a stabilization routine is then fired. After one single round of stabilization, the state is **END_STABILIZE**. At this state, depending on whether the $\mathbf{E_{sensor}}$ or $\mathbf{N_{sensor}}$ trigger is fired, either stabilization is done again or the state becomes normal.

Circuits

Breadboard connections (Breadboard.jpeg)



Jumpers on Gertboard (Gertboard.jpeg)



List of jumpers:

- ➔ *Power :* On top 2 pins in block J7
- ➔ *Serial In :* MCRX in JP11 to GP14
- ➔ *Serial Out :* MCTX in JP11 to GP15
- ➔ *Fan GND :* MOTB in J5 to GND
- ➔ *Fan Control :* MOTA in J5 to GP18
- ➔ *Usage LED :* BUF1 to GP25

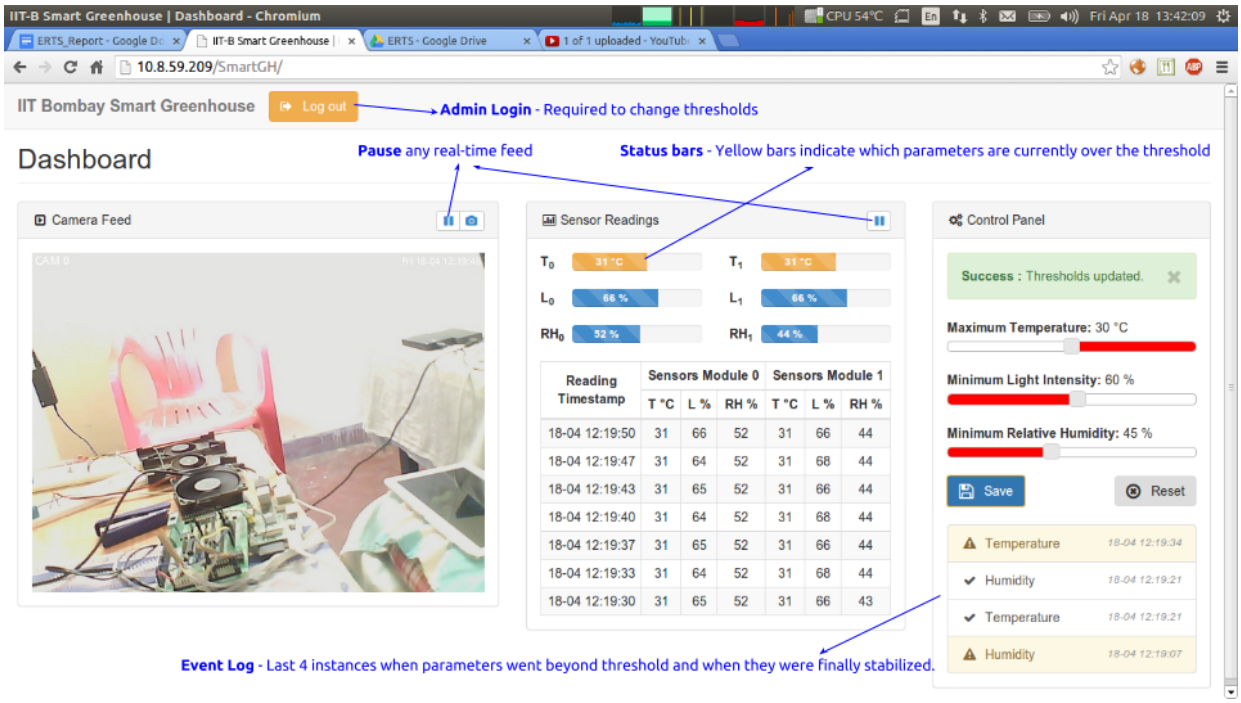
Jumpers for writing to ATmega:

- ➔ GP8 to ISP 5 (RESET) in J23
- ➔ GP9 to ISP 1 (MISO) in J23
- ➔ GP10 to ISP 4 (MOSI) in J23
- ➔ GP11 to ISP 3 (SCLK) in J23

You should additionally place jumpers on all BUF pins with GND to turn off the unused Gertboard LEDs.

5 Working of the System and Test results

The Web UI - Greenhouse Dashboard



Reading Timestamp	Sensors Module 0			Sensors Module 1		
	T °C	L %	RH %	T °C	L %	RH %
18-04 12:19:50	31	66	52	31	66	44
18-04 12:19:47	31	64	52	31	68	44
18-04 12:19:43	31	65	52	31	66	44
18-04 12:19:40	31	64	52	31	68	44
18-04 12:19:37	31	65	52	31	66	44
18-04 12:19:33	31	64	52	31	68	44
18-04 12:19:30	31	65	52	31	66	43

- ➔ The web interface displays all sensor values from both breadboards placed in the greenhouse.
- ➔ The progress bars indicate the current values of the parameters which are immediately visible. For the last few readings, a table is also shown below.
- ➔ The progressbar colour indicates the status of that parameter -- YELLOW indicates that the value has crossed thresholds and is being stabilized, BLUE indicates that the parameter value is within the thresholds.
- ➔ It also shows a scale to set the thresholds for temperature, light and humidity which are trigger the appropriate controlling systems when the parameter values cross these thresholds.
- ➔ Changing thresholds in the control panel is restricted to the admin only, to avoid unauthorized damage to greenhouse. A login button on top allows the admin to login and only then he can change thresholds.
- ➔ In the control panel, a list box also displays an event log showing instances when any of the parameter values crossed their threshold.
- ➔ A live feed of the greenhouse is displayed on the left side of the screen for keeping better watch on greenhouse.

➔ All live feeds from camera or sensor values can be paused at any instance using the pause buttons. This allows the user to closely examine the greenhouse automation at any suspicious instance.

The Backend - Monitoring & Controlling routines

Monitoring Systems

To fetch the sensor readings from the breadboards, we use the Gertboard extension board on our RPi. The ATmega 328 chip on the Gertboard is flashed with a program (*SensorMonitor.ino*) which polls the sensors and reads their values ~2sec. The DHT11 being a slow sensor (because of the 1bit digital signal protocol) is the bottleneck. The ATmega chip after processing the signal values from the sensors, writes them to the serial port of the Gertboard.

We have written a linux service (*serialmon*) on the RPi which continuously monitors the serial port (*/dev/ttyAMA0*) for any input and if the input matches the form expected from the ATmega chip (because the serial port may also be used by other services), then only we fetch the record and log it within the system.

Finally, when a client opens up the dashboard in his browser, recurring AJAX calls are made to a PHP script (*get_last_readings.php*) on RPi which delivers the sensor values by reading the system logs.

We have also written another linux service (*cameramon*) on the RPi which continuously monitors the camera (*/dev/video0*) every 2 - 3 sec using *fswebcam* software. This service dumps the latest image from the camera again to the system logs which is later picked up from a PHP script (*get_current_view.php*) via AJAX calls when the user opens the dashboard.

All the above 3 monitoring routines run concurrently on the systems. So there is minimum blocking and dependency. Even in one of the stages takes a long time, that would not block the system for the next stage. The next stage would keep polling till the previous stage returns any new data.

Controlling Systems

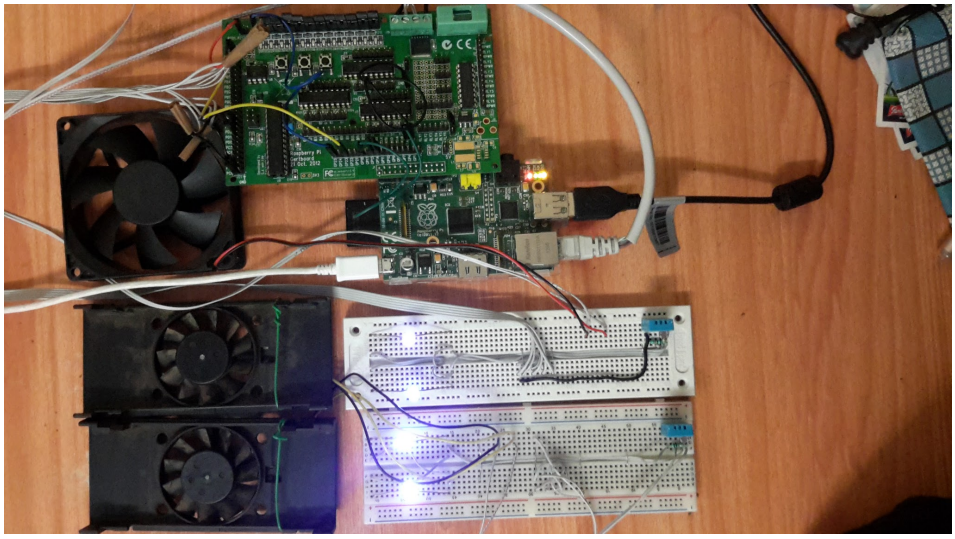
The controlling systems are spread over 2 stages:

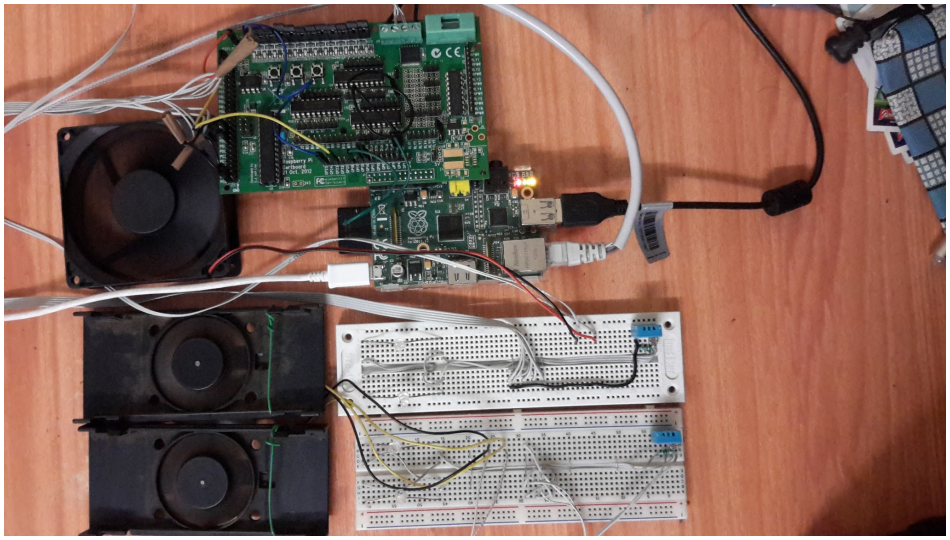
The user can set thresholds from the web user interface. When the user submits the new thresholds using a PHP script (*submit_thresholds.php*), the thresholds are recorded in a system logs folder which has proper permissions to be accessible by the active monitoring scripts also. The three thresholds are stored separately so as not to waste time in text processing and splitting the threshold values.

Now, the actual controlling happens within the serialmon script. serialmon continuously monitors the threshold files in the system logs folder as and when it reads a new line from /dev/ttyAMA0 serial port. If the new values from the sensors fall beyond the threshold values from the system logs, then serialmon itself fires the stabilizing routines. A new linux service was not created at this stage so as to provide immediate response to the adversities.

Testing Strategies & Results

Light Intensity Result: OK	We cover the greenhouse with a dark material and check if the sensors detect the decrease in light and turn on the artificial lights.
Temperature Result: OK	We can use a hair dryer to heat up the greenhouse and observe if the temperature rise is detected and fans stabilize it.
Humidity Result: OK	We can induce dry conditions by placing a piece of deliquescent substance inside the greenhouse and observe if the fans are turned on to facilitate ventilation





6 Discussion of the System

1. **What components of the project work as per plan?**
    a. Remote dashboard & live feed.
    b. Ability to override automatic control remotely.
    c. Monitoring temperature, humidity and light levels.
    d. Automatic controlling of fan and lights based on triggers from sensors.
    e. Combining data from multiple sensor modules for greater accuracy & reliability.

2. **What have we improved upon over what was mentioned in the SRS?**

a. *Better portability & power efficiency:* We have optimized the entire circuit to reside on a single RPi instead of two of them. Initially we had planned to let each RPi sense a sensor module & control a set of actuators. Later we found that the CPU usage of RPi stayed below 20% most of the time and there were several ADC pins which were still free. Thus we thought of utilizing the free pins & the free CPU by adding longer cables to different sensors modules.
b. *Usage Indicator:* We often came across the issue that the RPi was in use on a remote session or a system process and a user unaware of this fact turned it off and thus risked corruption of data or the system. Thus, we wrote a small script to turn on one of the low power LEDs on Gertboard if a user was logged in to RPi or in case of a system failure. The script can easily be extended for monitoring if and when the nginx server is in use (which would imply a client is monitoring the greenhouse remotely).
c. *Events log:* On the web user interface, we display an event log, which was initially unplanned. The rationale behind this log is that the user should have some way to verify the actions of the smart greenhouse. By viewing this log, the user can later check if the greenhouse took the right actions at the right time and would accordingly set the thresholds henceforth.

3. Changes made in plan from SRS

a. *pH sensors not used:* The sensors for detecting the pH value were not available in the nearby store and had to be imported.
b. *Used single RPi for everything instead of using two RPi:* As explained above, using one RPi was enough considering power and CPU usage.
c. *Misting system not used for controlling humidity:* This was not used due to time constraint and also to avoid the risk of damaging hardware.

7 Future Work

- The project currently places sensors at only two locations only i.e. along the length of the greenhouse. In order to get more reliable readings we can increase the number of sensors and place them at strategic locations.

- The types of sensors currently used are humidity, temperature and light intensity. The software is completely modular and thus, pH, electrical conductivity or any other sensor can also be easily plugged in and monitored.

- Currently, fans are switched on when either temperature is high or humidity is low. A much better solution would be to use a misting system which increases atmospheric humidity in case of low humidity.

- Better quality sensors can be used then the ones used presently. DHT11 is a cheap but an especially slow and inaccurate sensor (5% error rate!).

- The RPi has very limited power output (~50mA at 3.3V). We should use external power supply for sensors and actuators at any practical scale implementation.

- To compensate high light intensity, movable shades can be used for greenhouse. It can be opened during low light, and closed at peak sun time.

8 Conclusions

After a month's hard work, we were able to come up with a good design of the automation project, and also successfully implement it on an small scale. It was also tested in adverse conditions, and successfully responded to the conditions.

9. References

Setting up Gertboard https://projects.drogon.net/raspberry-pi/gertboard
RPi wiringPi setup https://projects.drogon.net/raspberry-pi/wiringpi
nginx on RPi setup http://elinux.org/RPi_Nginx_Webserver
DHT11 Library for Arduino https://github.com/adafruit/DHT-sensor-library
Setting up nginx + PHP FPM on RPi
http://www.ducky-pond.com/posts/2013/Sep/setup-a-web-server-on-rpi/