Greenhouse project

Embedded Operating Systems VIA UC

November 2017

Silvija Krupaviciute

Dmitry Rachkovsky

Table of Contents

[Introduction 2](#_Toc498982996)

[Greenhouse Model 3](#_Toc498982997)

[Light Intensity Sensor 5](#_Toc498982998)

[Circuit 5](#_Toc498982999)

[Pins 5](#_Toc498983000)

[Code 5](#_Toc498983001)

[Usage 5](#_Toc498983002)

[Light Intensity Control 6](#_Toc498983003)

[Circuit 6](#_Toc498983004)

[Pins 6](#_Toc498983005)

[Code 6](#_Toc498983006)

[Usage 6](#_Toc498983007)

[Temperature and Humidity Sensor 7](#_Toc498983008)

[Circuit 7](#_Toc498983009)

[Pins 7](#_Toc498983010)

[Code 7](#_Toc498983011)

[Usage 7](#_Toc498983012)

[Servo Motor Control 8](#_Toc498983013)

[Circuit 8](#_Toc498983014)

[Pins 8](#_Toc498983015)

[Code 8](#_Toc498983016)

[Usage 8](#_Toc498983017)

[Heater Control 9](#_Toc498983018)

[Circuit 9](#_Toc498983019)

[Pins 9](#_Toc498983020)

[Code 9](#_Toc498983021)

[Usage 9](#_Toc498983022)

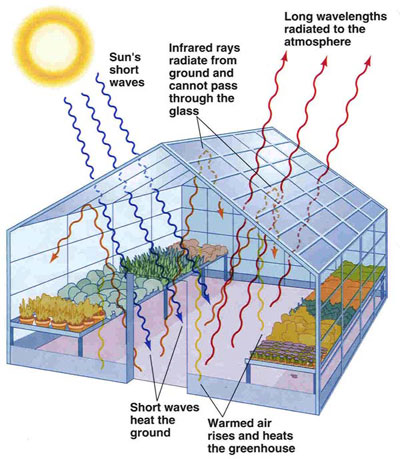
[Server and Client 10](#_Toc498983023)

[Nodejs 10](#_Toc498983024)

[REST web services 10](#_Toc498983025)

# Introduction

There are several aspects that need to be taken in consideration while creating a greenhouse control system. The greenhouse flora need optimal levels and duration of **daylight** for photosynthesis and plant respiration. **Ventilation** is also one of the most important components in a successful greenhouse in order to prevent bacteria and infections due to humid environment, regulate the temperature when it gets too hot and ensure air movement to prevent buildup of necrotic fungus. Finally, the **heating** is important for the greenhouse to operate properly in colder climates. Therefore, the main aspects that need to be considered are:

Monitoring:

* Humidity
* Temperature
* Daylight

Environment Control:

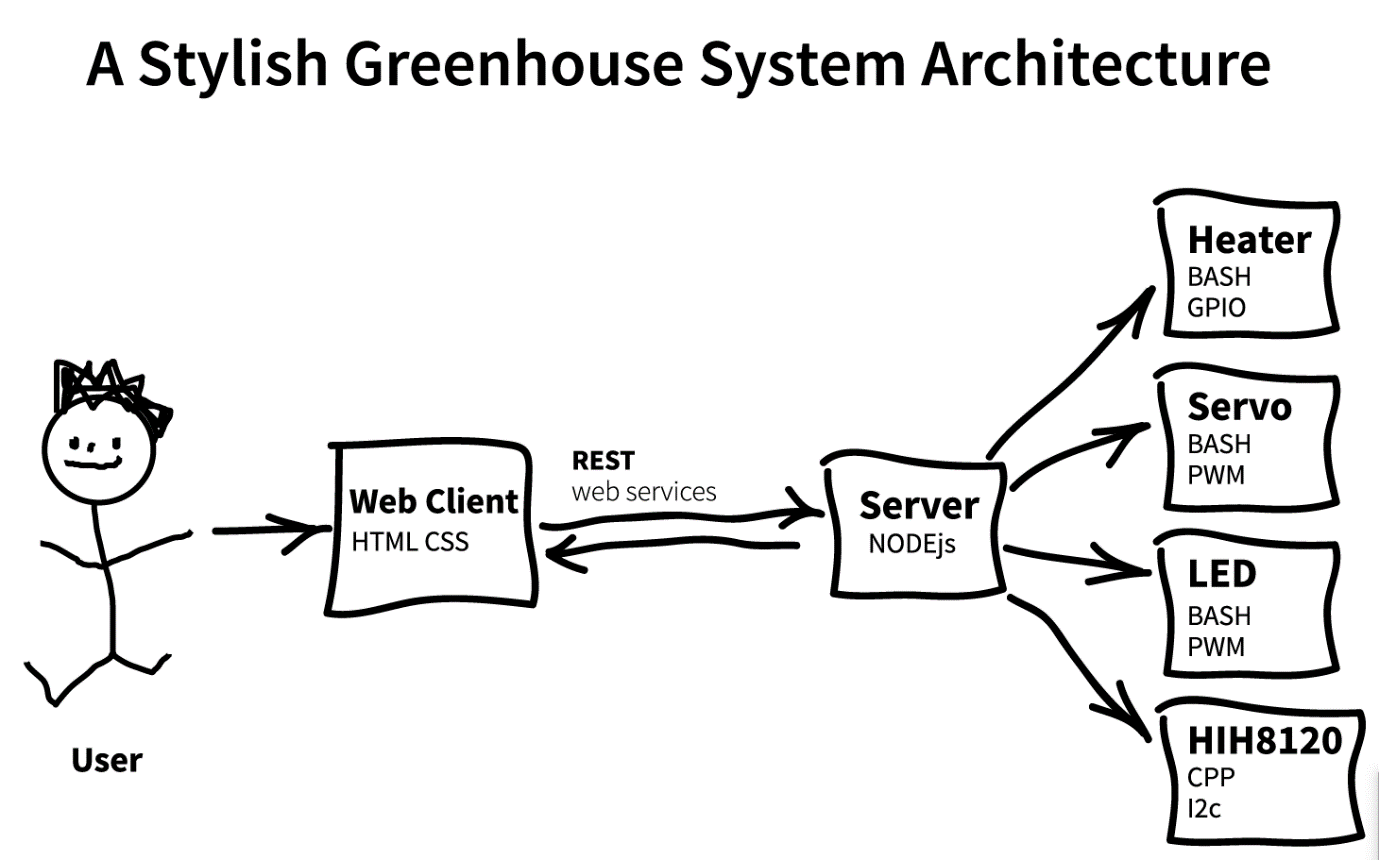
* Window Opening
* Heater
* Artificial Daylight

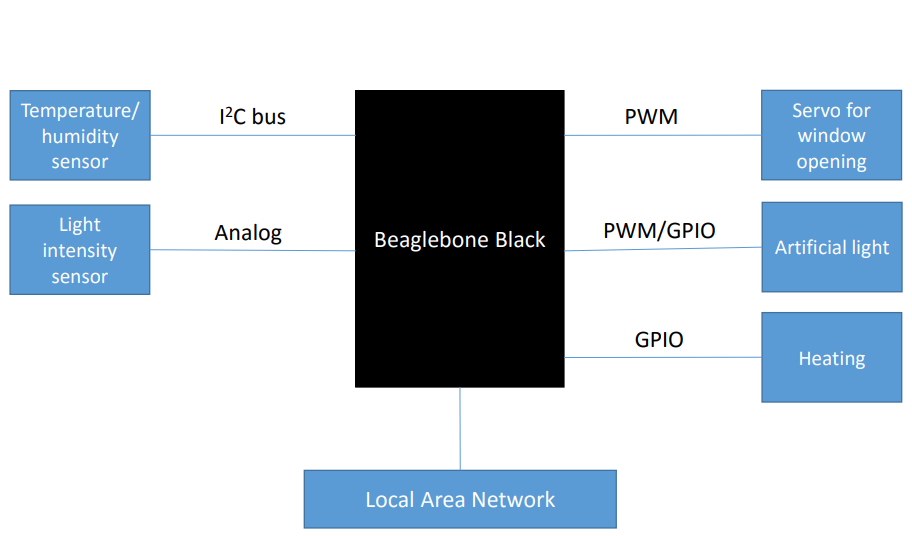
User Interface:

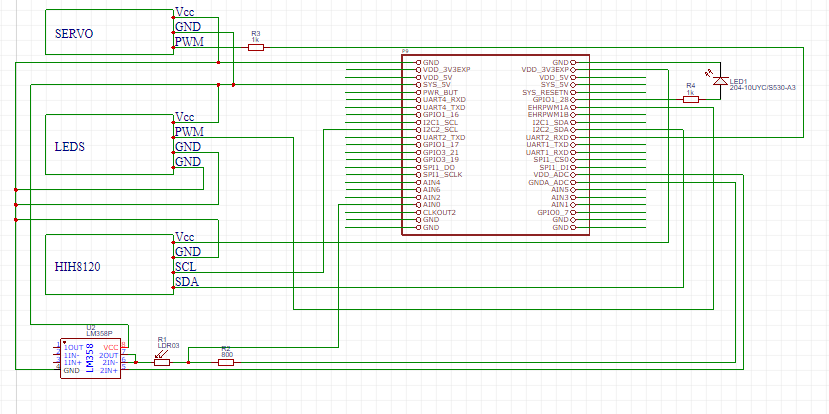
* Web Interface for Monitoring and Controlling remotely

# Greenhouse Model

|  |  |
| --- | --- |
| Feature | Technology |
| Measure temperature | I2C bus |
| Measure Humidity | I2C bus |
| Measure Light Intensity | Analog |
| Control servo motor for window | PWM |
| Control heater | Digital on/off |
| Control light intensity | PWM |
| Monitor and control remotely | Node.js server, RESTful API |

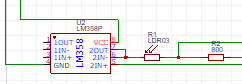






# Light Intensity Sensor

## Circuit



## Pins

P9\_07 – voltage

P9\_01 – ground

P9\_32 – ADC voltage

P9\_34 – ADC ground

P9\_39 – raw value

## Code

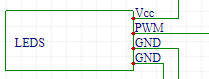
Controlled via a bash script ‘get\_light\_intensity.sh’

## Usage

Script used to get light intensity in percentage.

# Light Intensity Control

## Circuit



## Pins

P9\_07 – voltage

P9\_01 – ground

P9\_14 – PWM

## Code

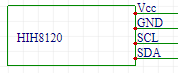
PWM controlled via bash scripts ‘set\_led.sh’ and ‘get\_led.sh’.

## Usage

Setting and getting current intensity of the LED stripe in percentage.

# Temperature and Humidity Sensor

## Circuit



## Pins

P9\_04 – voltage

P9\_01 – ground

P9\_19 – SCL

P9\_20 – SDA

## Code

I2C controlled via C++ compiled binary ‘hih8120.bin’

## Usage

Binary allows to get current value of temperature and/or humidity read by sensor.

# Servo Motor Control

## Circuit



## Pins

P9\_07 – voltage

P9\_01 – ground

P9\_22 – PWM

## Code

PWM controlled via bash scripts ‘set\_lid.sh’ and ‘get\_lid.sh’. PWM is not enabled all the time – it only enables to move the motor and then disables again.

## Usage

Setting and getting current position of the box lid in percentage – 0 for completely closed and 100 for completely opened.

# Heater Control

No heater was given to us so we just used an LED.

## Circuit



## Pins

P9\_02 – ground

P9\_12 – output

## Code

Controlled via bash scripts ‘get\_heater.sh’ and ‘set\_heater.sh’

## Usage

Setting ang getting current “heater” status – 0 for off, 1 for on.

# Server and Client

The greenhouse is controlled by the client developed in HTML/CSS and vanilla JavaScript to send some Http requests for getting and posting the data to the server which was developed using NodeJS. The system is made using the REST web services principle for sending the requests between the client and the server.

## Nodejs

In the server side of the system we use routes to call functions that execute the shell and binary scrips that are either fetching the sensor data or controlling the greenhouse modules. The data then is passed using the Json format.

## REST web services

The client side is using REST web service principle to send requests via Http protocol. First, the client application use the GET requests to acquire sensor data (Humidity, Temperature and Brightness respectively). Moreover, it sends GET requests to find out the state of the greenhouse modules in order to notify the user of their status, for example whether the heater is on/off or the enclosure is open/closed or to find out the current brightness value for the LED. Finally, it uses POST requests to control the modules by triggering the onClick event listeners, for example Heater: on/off, Lid: open/close.

