Smart Moisture Sensing Watering System (SWS)

SWE 690

Group Green
Anish Raj Poudel

Introduction:

Problem

Everyone loves a beautiful bloom of home grown flowers, but busy lifestyle and a lack of knowledge often leads to frustration and failure in maintaining healthy plants. Overwatering is one of the few reasons that can lead to the death of the plant. Even currently available automatic time based watering systems are susceptible. Some may explore the route of hiring a professional gardener, but this is not a solutions for everyone as it may not fit the budget. A low-cost, intuitive, automated and educational watering system that provides insights into, or in many case resolve, the aforementioned headaches is expected to be popular amongst busy and novice plant lovers.

Why this project is related to the class

This project is related to the class because it covers the total software development stack including the knowledge of hardware like sensors and software. These days every company would like to invest in IOT (Internet of things). Smart plant watering will explore some of the technology that is being used to develop the infrastructure for to build the connected devices. Also, it covers the cloud computing aspect of the software development and will demonstrate how a device could be controlled and connected through the internet.

Areas or scope of investigation:

There are many areas of scope of investigation for the smart watering system. Some of them are listed below:

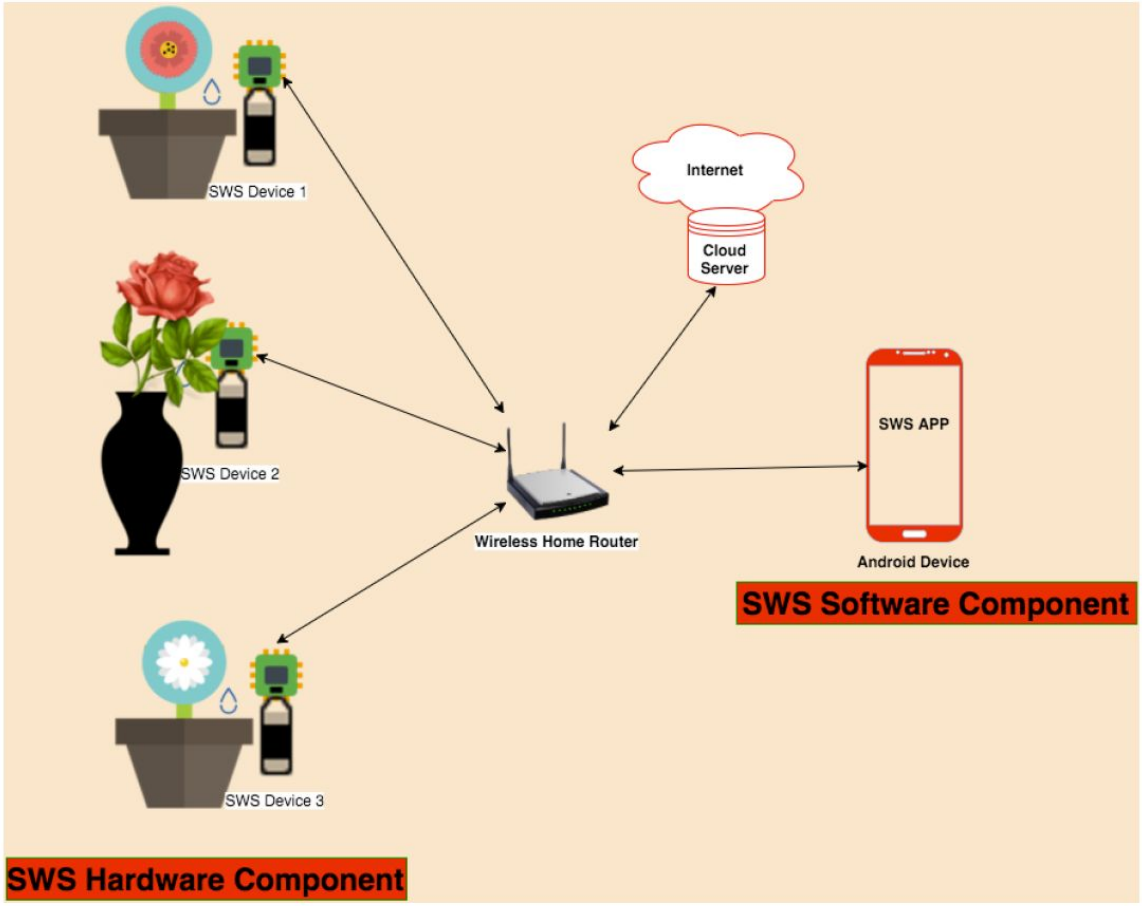
- Detailed analysis of soil type and the level of moisture required to grow plant in that type of soil.
- Image processors that will analyze the growth of the plant and compares the growth rate on one seasons with another and present the analytics data to the owner.

- Image processors to detect weed or any other kind of unwanted plants that might grow up with the intended plant, and present the weed information to the owners

Theoretic bases and literature review

Proposed Solution:

Our proposed solution is to build the smart measuring system aims to automate the watering of household plants and provide them with the optimal watering schedule based on either preset times, moisture level or plant type. The Smart Moisture sensing watering system achieves customizability, affordability, and portability as it attaches to any existing planting pot. The proposed system will present the novice planter with the database of the plant watering profiles to deploy the watering schedule to his or her plant using an Android application.



Solutions uniqueness

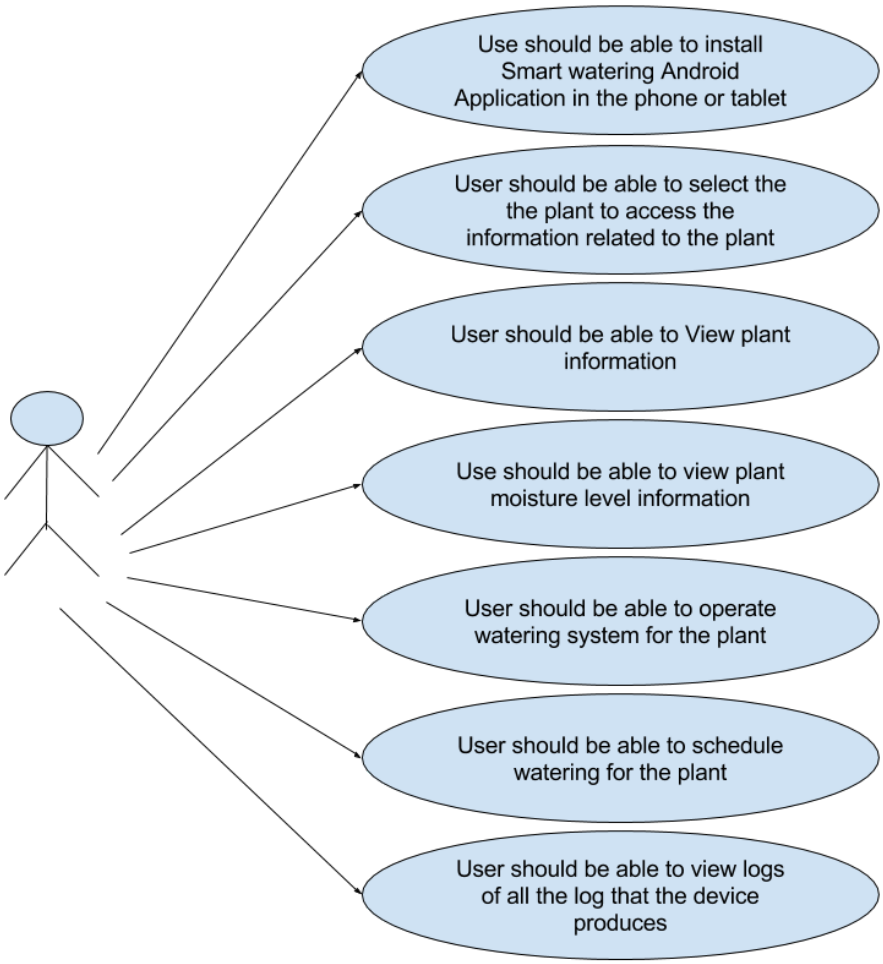
There are so many commercials products out in the market for the watering plant, but most of them are not smart. Most of the available products are time based watering system. Our proposed system is smart with the moisture sensing capability, it provides data access about most of the home grown plants and their watering information. Besides that the system is connected to the internet. Therefore, the plant owner, from any where in the world can look at and control the watering status and schedule.
Solution uniqueness are:

- Smart with the moisture sensing capability
- Information about the plants requirements and their watering information
- Connected to the internet, therefore controlled and supervised from anywhere
- Just with an install of an app you should be able to connect to your beautiful home garden.

Hypothesis

1. System should be able to detect the soil moisture and decide whether to water a plant or not
2. System should be able to allow users to manually override the process
3. User should be able to connect to the system using a graphical user interface
4. User should be able to configure the watering system with his preferences
5. All the data should be stored in the cloud database so that the user should be able to view it anytime from anywhere
6. When user clicks on the water plant button, it should start watering the plant and when the user clicks on the stop button, it should stop watering the plant
7. System should be preset for sensing and watering based on the plant type.

Solutions Use Cases



Methodology

Devices used:

- Soil Moisture Sensor
- TMP36 Temperature Sensor
- TSSS® Submersible Water Pump
- L293D Stepper Motor Driver
- Raspberry Pi 3 Model B
- 8mm plastic pipe
- Android phone or tablet
- Network infrastructure

Software Used:

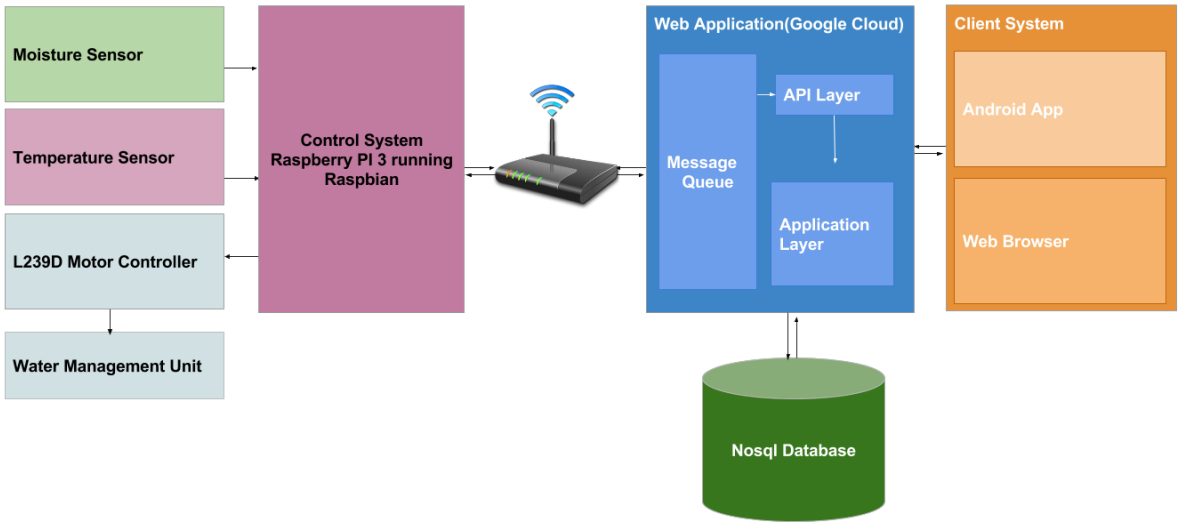
- Android Studio

- Raspberry GPIO library for Python
- Raspbian (OS for Raspberry PI)
- Google App Engine (Server Side Programming)
- Google Datastore (Cloud Data Storage)

Programming Language Used:

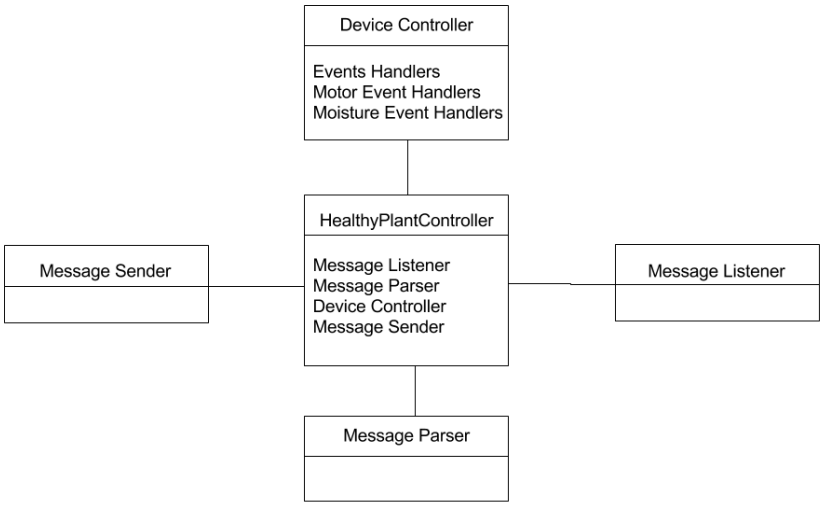
- Java 7+
- Python 2.7
- Shell Scripting
- C

High Level Architecture Diagram:

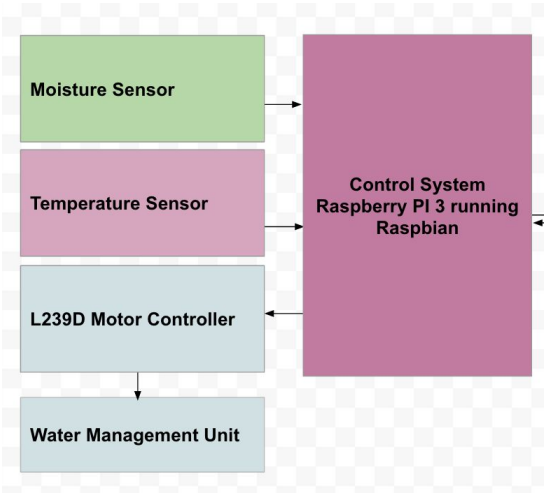


Detail Design:

Device Components Class Diagram



Device Components Diagram



Device Component Functionality description:

When the devices boot up. It will run device controller software. The software has three main components

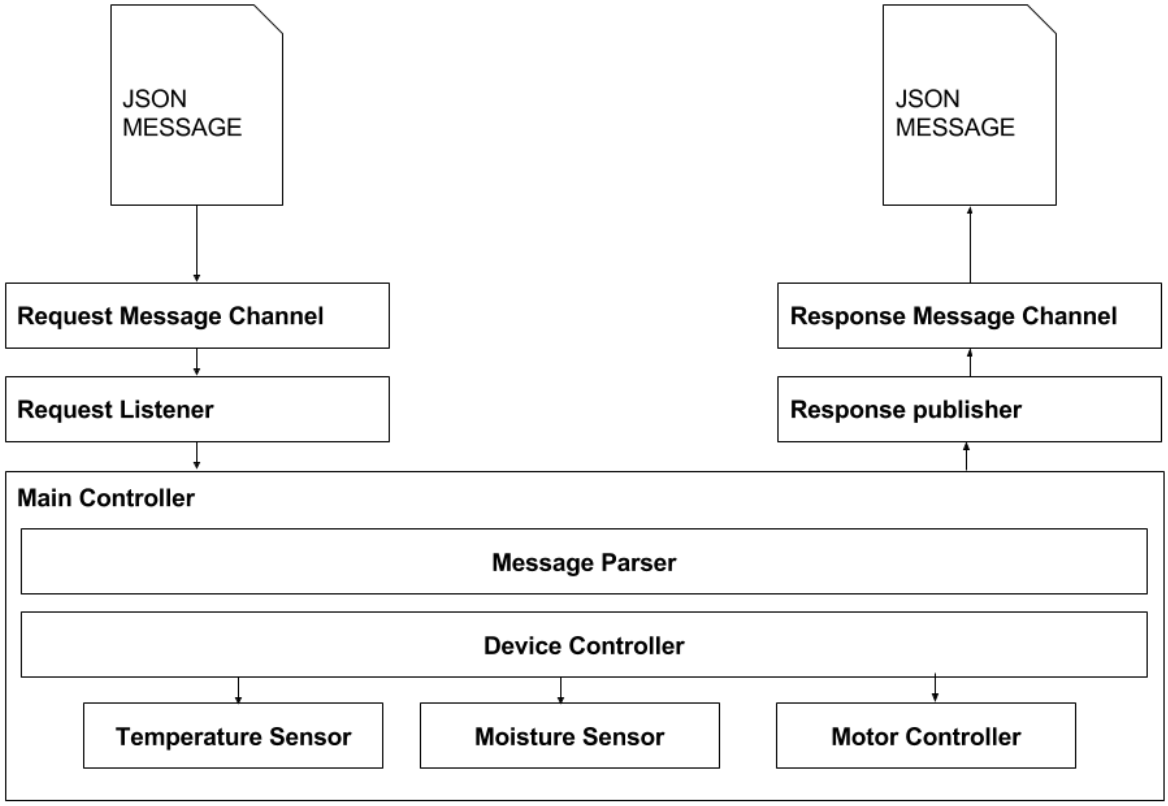
1. Message Listener
2. Message Parser
3. Device Controller
4. Message Sender

Message listener is initiated on the start of the application and is live till the program runs. The main responsibility of the the message listener is to listen to any message that is sent to the devices from the web server or the mobile devices. Each devices will have its own publisher keys and subscriber keys that it will be using to communicate with user devices.
When the listener receives the message will calls the message parser to parse the message. For example the message will be in the form of JSON structure.

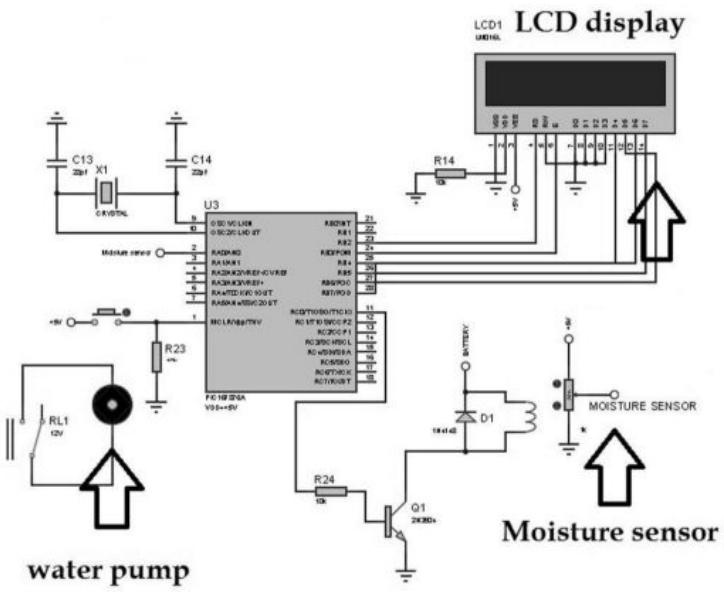
{ “command” : “READ_MOISTURE” }

The parser will parse the message and grabs the command to run within the devices. The Main Controller will use devices controller to execute the receive message. It also sends back the success or the failure response to the message queue using the response channel.

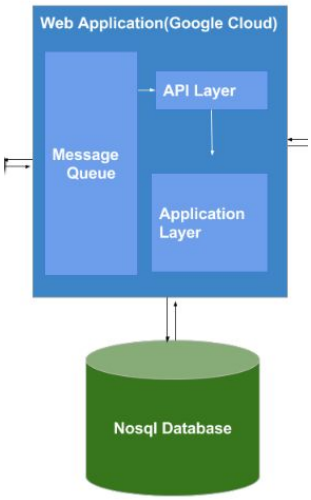
The overall flow diagram of the system would look like given below.



Sensor and Circuit Diagram



Web Server Detail Design

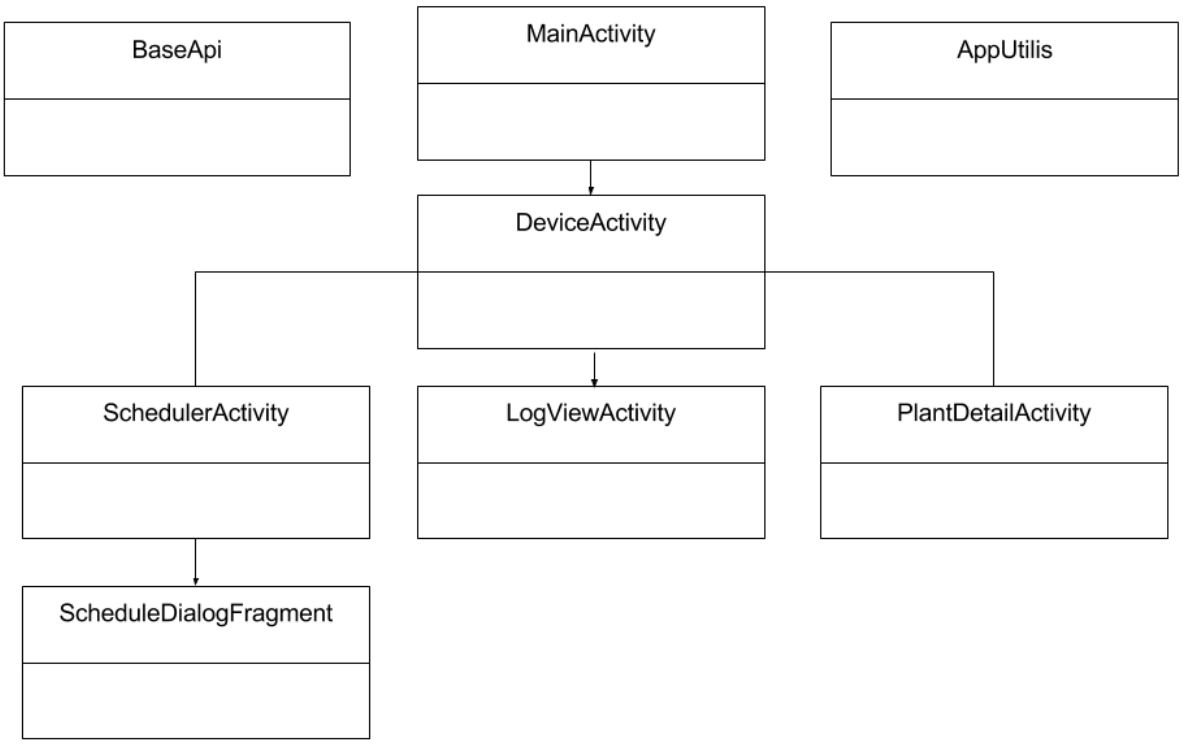


Web Application consists of the Message Queue. For the Smart Watering Application, we have used Pubnub message queue for the application messaging.
Devices talks to the Web server using the server endpoints. And server responses back to the devices with the respect to the request received from the devices. All the user and devices interaction are logged in the cloud datastore. Besides that, application also, logs the amount of water dispensed and time of watering for each plant connected to the devices.

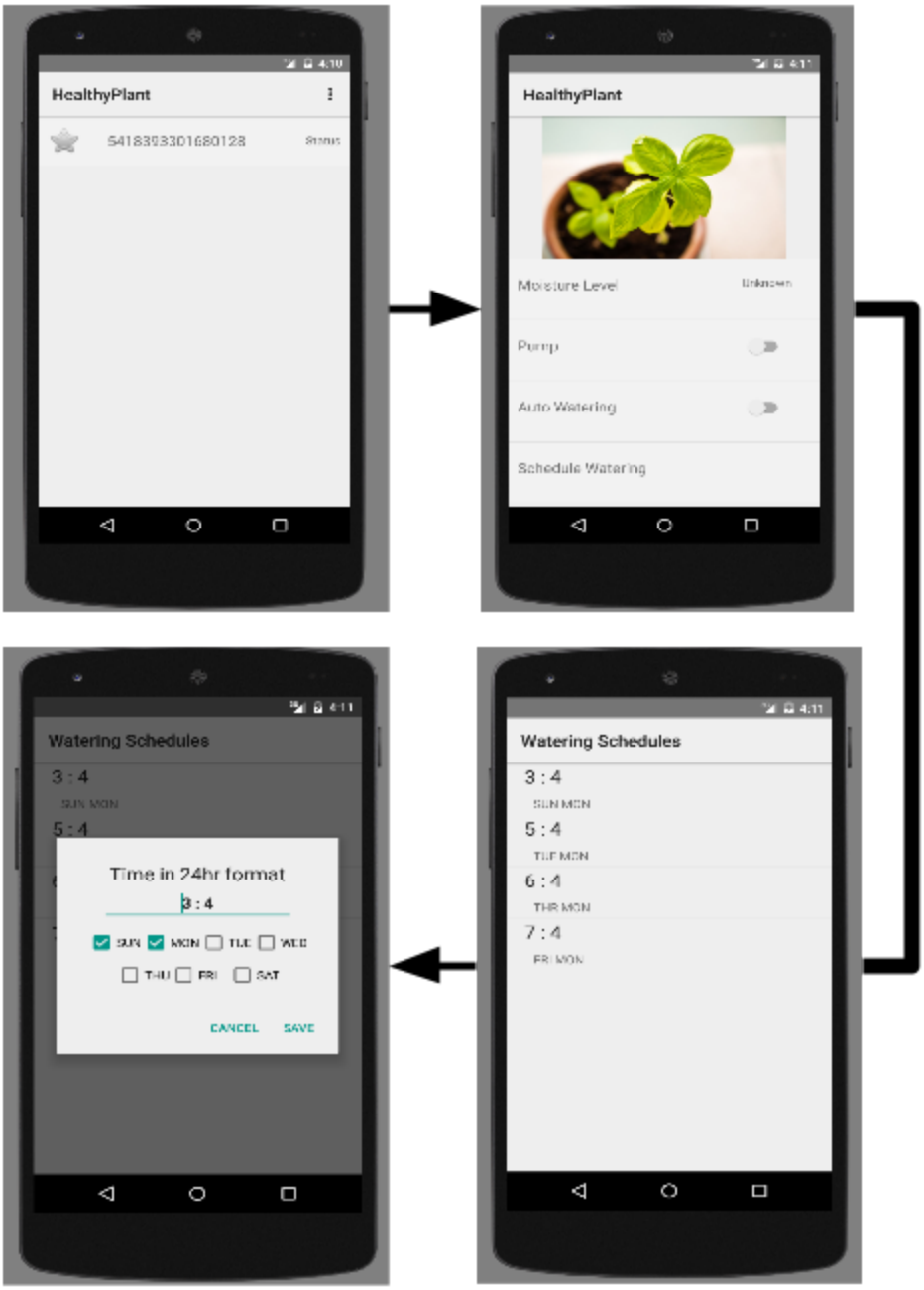
Web Application server pushes messages to the user devices if it receives messages from the devices. The messages could be devices control instruction response or device status response.

Client System Detail Description:

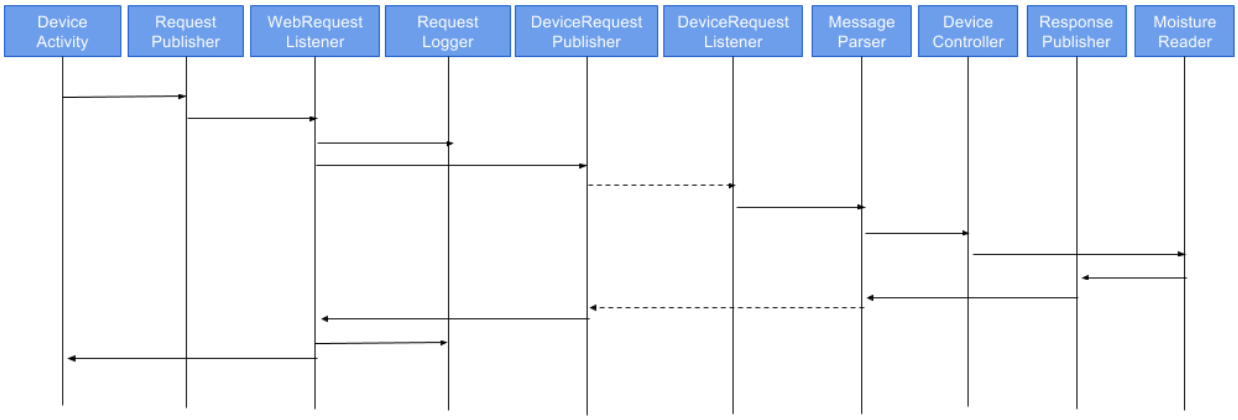
Android Application Classes Diagram:



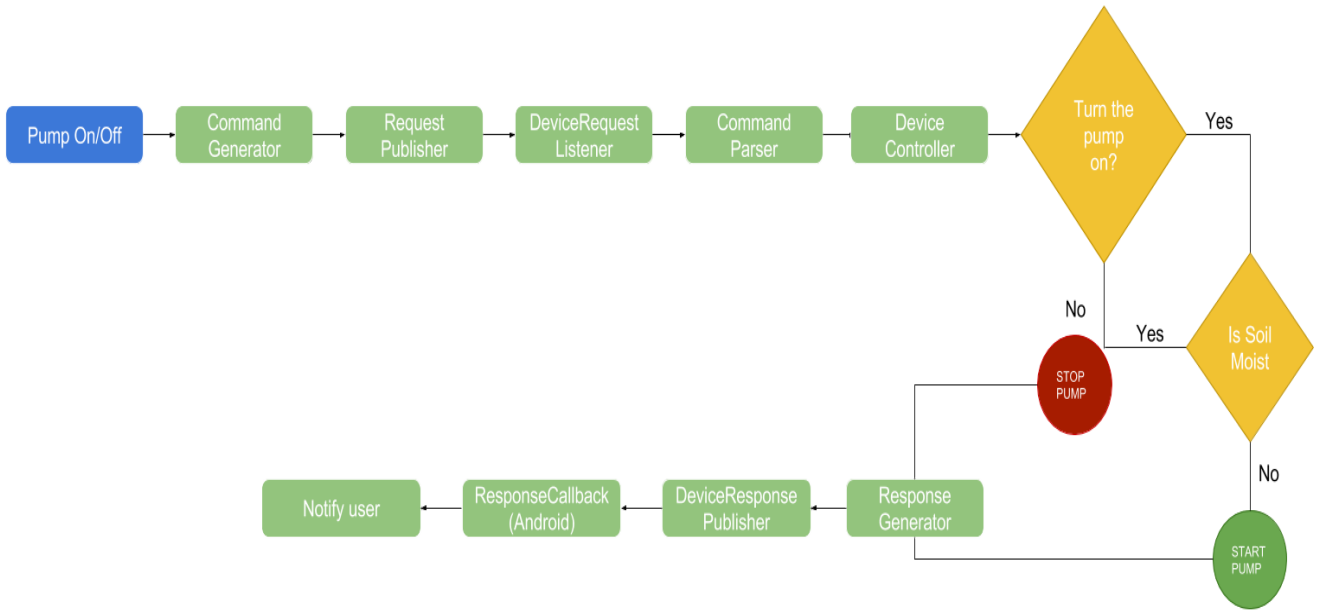
Android Application UIs



Process Flow Diagram of Reading Soil Moisture Data



Flow Chart of Turning Pump on and off



Data Collection:

The SWS utilizes a moisture sensor that is planted in the soil of the plant to track and retrieve moisture levels. The gathered information is represented in form of a moisture level percentage(%) where a moisture level of 80% represents the volumetric ratio between water and soil content. The sensor calculates the moisture level based on proven methods that measure the electrical resistance or the dielectric constant. The percentage value is utilized for a number of key function in the project which will include watering schedules based on a pre-set moisture level threshold(eg maintain 20% moisture) alerting based on moisture level, and a log of history of the moisture level for determining absorption/evaporation rates. Besides the moisture sensor, the system will also have a temperature sensor to calculation the temperature around the system.

Test Plan:

Test Plan Identifier

The structure of this test plan is strictly based on IEEE 829 test plan outline.
This test plan, here is the master test plan for “Smart Watering System” first release. This test plan will have subsequent test plan relating to different level of tests.

References:

All the information, plan and processes included in this master test plan is based on the following documentation listed below:

Hypothesis

This master test plan consists of following phase:

- Unit tests and integration level-adherence to coding standard and successful communication between units
- System level-performance, usability, functionality tests

Software risk issues:

For the “Smart Watering System” to be operational it will need plenty of communication channels between external as well as internal parties. Bearing that in mind, there are some risks when it comes to integrating the software with third party API’s such as Google Appengine, and Google Cloud datastore and pubnub message queue. There may be multiple service implemented that is intertwined between one another. The software could also be prone to security risks, so user authentication and authorization features should also be implemented and tested properly to avoid security exploitations. Also, the documents for future enhancement and change request should be properly managed to avoid longer implementations periods and unnecessary side effects due to minimal changes.

- Test all the third party and public API properly. See if there are any certificates necessary to acquire services and that they are placed in a secure server.
- Unit test all the utilities functional and see if the exposed functions are desirable.
- Api's that are exposed should be carefully analyzed so that it is not prone to cyber threats.
- Check the performance of functions so that memory is not leaked. Check for closed connections after they are used.
- Keep track of build and release management and keep the change logs descriptive and transparent.
- Use tools to manage defects and change requests.
- Follow government as well as company's standards and abide by the compliance set forth by these parties.

Features to be tested

Risk*: 0 - Low, 1 – Medium, 2 – High, 3 – Critical*

Features	**Risk**
1. User should be able to install android app for the Smart Watering system.	3
2. User should be able to connecting to the devices wirelessly through the internet	3
2. User should be able to see the list of connected devices in his devices list	3
3. User should be able select the particular devices that he want to monitor and operate on	3
4. User should be able to get the moisture information for each device reading the plant soil data	3
5. User should be able to control watering for the plant from their mobile device	3
6. User should be able to On the pump to water the plant	3
7. User should be able to turn the water pump off using the mobile device	3
8. If the soil is already moist, the system should be able to detect it and should not turn on the pump	3
9. User should be able to turn the auto watering functionality on for automatically water if the plant soil is dry	3
10. User should be able to configure the scheduler to water the plant.	3

11. System should log all the user and plant interaction in the cloud data storage.	3

Features not to be tested:

Future enhancement such as analysing the moisture and water dispenser log and predicting the estimated watering time is not testable. Also, test on the multiple devices is also not possible because only single device is used in the project. Similarly, detecting the weeds on the plants and informing the user about is also a future feature therefore it can not be tested as well.

Tests and Test Results:

System Validation Tests:

Test case	Expected Results	Actual Results
Moisture sensor should be able to read moisture data	Result should be moist or dry	Result should be moist or dry
Motor Controller should be able to start and stop the water dispenser	YES	YES
Temperature sensor should be able to read the temperature data	YES	YES
User should be able to install the android app to the their mobile device	YES	YES
User should be able to get the list of the devices connected to the plant	YES	YES
User should be able to see the moisture data in their wireless devices	YES	YES
User should be able to control the watering pump on and off button	YES	YES
User should be able to create new schedule for watering	YES	YES
User should be able to edit scheduling for the watering	YES	YES
User should be able to delete schedule for the watering	YES	YES

Application should be able to log moisture data into the database	YES	YES
Application should be able to log the watering log into the database	YES	YES
System should be able to detect the moistures and turn the watering system off if the soil is already moist	YES	YES
System should notify user that the soil is already moist so turing the watering system off.	YES	YES

Database Layer Test

Test Cases	Expected Result	Actual Result
System should be able to save and retrieve devices	YES	YES
System should be able to save devices logs	YES	YES
System should be able to save connection logs	YES	YES

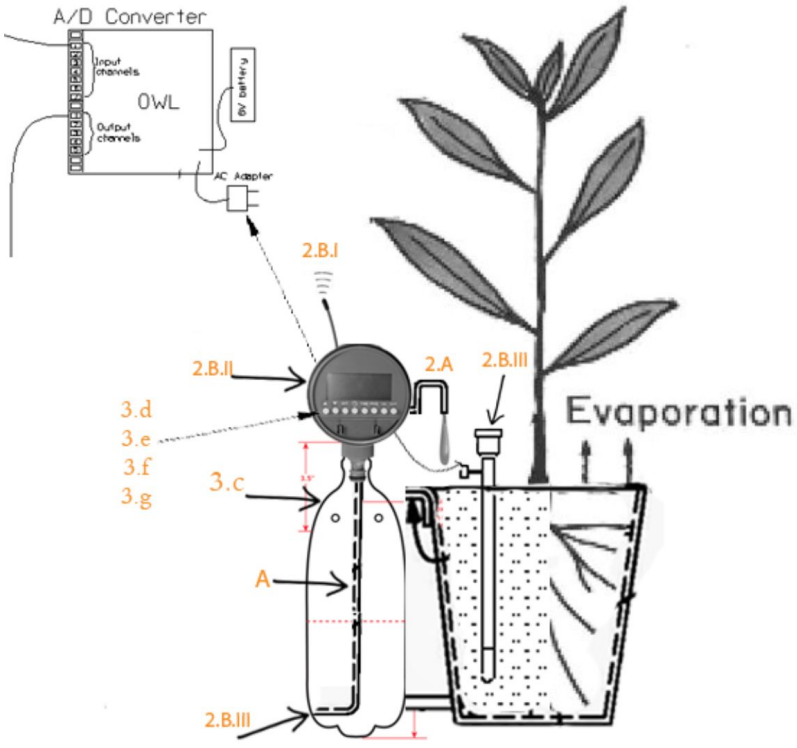
Device Layer Test

Test Cases	Expected Result	Actual Result
Device should be able to communicated with the servers and the message queue	YES	YES
Device should be able to get data from the connected sensors	YES	YES
Devices should be able to receive commands as message from the web server or mobile device and control the connected sensors	YES	YES

Recommendation for Smart Moisture sensing watering system:

The smart Moisture sensing watering system is able to sense the presence of the water level in the soil and inform the user about the same base on which user can operate the watering system. The further recommendation for the product would be to recommend optimal watering schedule for watering based on the data collected from the watering log.
Similarly, another recommendation system would be to automatically recommend the watering information and plant information base on the picture of the plant that user will provide to the system.
The architecture of the system in build in the way the a user can add multiple sensing devices or the control devices like home window controller, oven controller, tv controller to the system and control the complete home automation through a mobile app.

General view of the product



Bibliography

https://www.fujitsu.com/global/documents/about/resources/publications/fstj/archives/vol46-4/paper15.pdf
http://www.cs.odu.edu/~nadeem/classes/cs795-CPS-S13/papers/app-008.pdf
https://www.raspberrypi.org/products/raspberry-pi-3-model-b/
http://www.instructables.com/answers/L239D-Motor-Driver-IC/
https://developer.android.com/guide/index.html
http://www.instructables.com/id/Soil-Moisture-Sensor/

Files the goes on each devices (Embedded System)

```
.
├── api.py
├── api.pyc
├── config.py
├── config.pyc
├── healthyplant.py
├── pitask.py
└── pitask.pyc
```

device_file/api.py

```python
import requests
import config

class Utils:
    """
    Utils Class for the devices
    """

    @classmethod
    def get_local_ip(cls):
        '''
        This function gets the local ip address of the devices
        '''
        import socket
        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        s.connect(('8.8.8.8', 80))  # connecting to a UDP address doesn't send packets
        return s.getsockname()[0]

class HealthyPlantApi:

    URL_BASE = "http://healthy-plant.appspot.com"
    #URL_BASE = "http://localhost:27080"
    API_BASE = URL_BASE + '/api'
    API_CONNECTION_STATUS = API_BASE + '/connection'
    API_MOISTURE_STATUS = API_BASE + '/moisturelog'

    @classmethod
    def log_connection(cls):
        '''
        Log_connnection function logs the connection to the cloud datastore
        This logs the timestamp and which ip address the device is connected with.
        '''
        r = requests.post(cls.API_CONNECTION_STATUS,
            data = {
                'serial_number' : config.DEVICE_SERIAL_NUMBER,
```

```
                'internal_ip' : Utils.get_local_ip()
            })

    @classmethod
    def log_moisture(cls, status):
        '''
        log_moisture function logs the mositure data to the cloud database
        with the timestamp
        '''
        r = requests.post(cls.API_MOISTURE_STATUS,
            data = {
                'serial_number' : config.DEVICE_SERIAL_NUMBER,
                'status' : status
            })
```

device_file/config.py

```
DEVICE_SERIAL_NUMBER = '5418393301680128'
PUBLISH_KEY = 'pub-c-6e26ca7f-f358-474a-8b58-74cf5ff76be0'
SUBSCRIBE_KEY = 'sub-c-fd449834-4bb8-11e6-82fe-0619f8945a4f'
PUBSUBCHANNEL_REQUEST = 'healthy-plan-request-channel'
PUBSUBCHANNEL_RESPONSE = 'healthy-plan-response-channel'
MOISTURE_CHANNEL = 12

MOISTURE_ANALOG = 40
PUMP_CHANNEL = 18

Motor1A = 16
Motor1B = 18
Motor1E = 22
```

device_files/healthyplant.py

```
from pubnub import Pubnub
import config
#from pitask import is_soil_moist, signal_pump, moisture_event
from api import HealthyPlantApi, Utils
import json

class MessageParser:

       @classmethod
       def requestParser(cls, msg):
              print msg
              msg = json.loads(msg)
              print msg
              command= msg.get('command')
```

```
print command
params = msg.get('request')
print params
if command != "":
	if command == 'CHECK_MOISTURE':
		value = is_soil_moist()
		return (
			200,
			{"request" : msg,
			"response" :
				{
					"data" : value

				}
			})
		#return is_soil_moist()

	elif command == 'TURN_PUMP_ON':
		print "trun pump on"
		value = signal_pump(True)
		return (
			200,
			{"request" : msg,
			"response" :
				{
					"data" : value

				}
			})
	elif command == 'TURN_PUMP_OFF':
		print "trun pump off"
		value = signal_pump(False)
		return (
			200,
			{"request" : msg,
			"response" :
				{
					"data" : value

				}
			})

	elif command == 'CHECK_CONNECTION':
		HealthyPlantApi.log_connection()
		return (
			200, {
			"request" : msg,
			"response" : {
				"data" : True
				}
```

```
                                        })


def init():
        pubnub = Pubnub(
                publish_key=config.PUBLISH_KEY,
                subscribe_key=config.SUBSCRIBE_KEY
                )

        def callback(message, connect):
                code, response = MessageParser.requestParser(message)
                pubnub.publish(config.PUBSUBCHANNEL_RESPONSE, response )

        def connect(message):
                print ("Connected")


        def disconnect(message):
                print ("Disconnect")

        def reconnect(message):
                print "Reconnect"


        pubnub.subscribe(channels=config.PUBSUBCHANNEL_REQUEST, callback = callback,
error=callback, connect=connect, reconnect=reconnect, disconnect=disconnect)
        HealthyPlantApi.log_connection()
        moisture_event()


def main():
        try:
                local_ip = Utils.get_local_ip()
                if local_ip:
                        init()
        except Exception as e:
                print "Not able to initiate the applicaton "
                print e

main()
```

device_files/pitask.py

```
import RPi.GPIO as GPIO
import time
import config
```

```
from api import HealthyPlantApi


MOISTURE_CHANNEL = config.MOISTURE_CHANNEL
PUMP_CHANNEL = config.PUMP_CHANNEL
MOISTURE_ANALOG = config.MOISTURE_ANALOG
Motor1A = config.Motor1A
Motor1B = config.Motor1B
Motor1E = config.Motor1E

def setup():
        GPIO.setmode(GPIO.BOARD)
        GPIO.setwarnings(True)
        GPIO.setup(MOISTURE_CHANNEL, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
        GPIO.setup(Motor1A,GPIO.OUT)
        GPIO.setup(Motor1B,GPIO.OUT)
        GPIO.setup(Motor1E,GPIO.OUT)
        GPIO.output(Motor1E,GPIO.LOW)


def _moisture_evt_callback(c):
        sig = True
        if GPIO.input(MOISTURE_CHANNEL):
                sig = False
        HealthyPlantApi.log_moisture(sig)
        if sig:
                GPIO.output(Motor1E,GPIO.LOW)
        return sig

def _moisture_analog(c):
        print c
        print GPIO.input(MOISTURE_ANALOG)

def is_soil_moist():
        #GPIO.setmode(GPIO.BOARD)
        #GPIO.setup(MOISTURE_CHANNEL, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
        return _moisture_evt_callback(None)


def signal_pump(signal):
        if signal and not _moisture_evt_callback(None):
                GPIO.output(Motor1A,GPIO.HIGH)
                GPIO.output(Motor1B,GPIO.LOW)
                GPIO.output(Motor1E,GPIO.HIGH)
                return True
        else:
                GPIO.output(Motor1E,GPIO.LOW)
                if not signal:
                        return True
                return False
```

```
def moisture_event():
        #GPIO.setmode(GPIO.BOARD)
        #GPIO.setup(MOISTURE_ANALOG, GPIO.IN)
        GPIO.add_event_detect(MOISTURE_CHANNEL, GPIO.BOTH, bouncetime=300)
        GPIO.add_event_callback(MOISTURE_CHANNEL, _moisture_evt_callback)

setup()
```

Web application code (Hosted on appengine)

```
src/
├── __init__.py
├── api.py
├── api.pyc
├── app.yaml
├── index.yaml
├── main.py
├── main.pyc
├── model_manager.py
├── model_manager.pyc
├── models.py
├── models.pyc
├── script.py
└── script.pyc
```

src/api.py

```
import webapp2
import json
from model_manager import save_connection_log, save_mositure_log, get_devices_list
from script import create_device

class JsonResponse(webapp2.RequestHandler):

        def get_data(self):
                data = self.request.POST.items()
                d = dict((x, y) for x, y in data)
                print d
                return d

        def renderJson(self,response):
                self.response.headers['Content-Type'] = "application/json"
```

```
                self.response.out.write(json.dumps(response))

class MainPage(JsonResponse):
    def get(self):
        self.renderJson({"key": "anish"})

class LogConnection(JsonResponse):

        def post(self):
                data = self.get_data()
                save_connection_log(data)
                self.renderJson({"status" : 200})

class GetDevices(JsonResponse):
        def get(self):
                data = get_devices_list()
                print data
                self.renderJson(data)

class CreateDevice(JsonResponse):
        def get(self):
                create_device()

class LogMoisture(JsonResponse):
        def post(self):
                data = self.get_data()
                save_mositure_log(data)
                self.renderJson({"status" : 200 })
```

src/app.yaml

```
runtime: python27
api_version: 1
threadsafe: true

handlers:
- url: /.*
  script: main.app
```

src/index.yaml

```
indexes:

# AUTOGENERATED

# This index.yaml is automatically updated whenever the dev_appserver
# detects that a new type of query is run.  If you want to manage the
# index.yaml file manually, remove the above marker line (the line
```

```
# saying "# AUTOGENERATED").  If you want to manage some indexes
# manually, move them above the marker line.  The index.yaml file is
# automatically uploaded to the admin console when you next deploy
# your application using appcfg.py.

- kind: Device
  properties:
  - name: publish_key
  - name: serial_number
  - name: subscribe_key
```

src/main.py

```
# Copyright 2016 Google Inc.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

import webapp2

from api import (
        MainPage,
        LogConnection,
        CreateDevice,
        LogMoisture,
        GetDevices
        )

app = webapp2.WSGIApplication([
    ('/', MainPage),
    ('/api/connection', LogConnection ),
    #('/create/device', CreateDevice),
    ('/api/moisturelog', LogMoisture),
    ('/api/devicelist', GetDevices)
], debug=True)
```

src/model_manager.py

```
from models import Device, Connection, MoistureLog

def save_connection_log(data):
	if data:
		dev = Device.get_by_serial_number(data.get('serial_number'))
		conn = Connection(
			device = dev.get().key,
			internal_ip = data.get('internal_ip'))
		conn.put()

def save_mositure_log(data):
	if data:
		dev = Device.get_by_serial_number(data.get('serial_number'))
		log = MoistureLog(
			device = dev.get().key,
			status = True if data.get('status') == 'True' else False
			)
		log.put()

def get_devices_list():
	query = Device.get_all_active_devices()
	data = query.fetch(projection=[Device.serial_number, Device.publish_key,
Device.subscribe_key])
	result = []
	for d in data:
		result.append({
			"serial_number" : d.serial_number,
			"publish_key" : d.publish_key,
			"subscribe_key" : d.subscribe_key
			})
	return result
```

src/models.py

```
from google.appengine.ext import ndb

class Plant(ndb.Model):
    name = ndb.StringProperty()
    kind = ndb.StringProperty()
    description = ndb.StringProperty()

class Device(ndb.Model):
    serial_number = ndb.StringProperty()
    publish_key = ndb.StringProperty()
    subscribe_key = ndb.StringProperty()
```

```
    @classmethod
    def get_by_serial_number(self, serial_number):
        return Device.query(Device.serial_number == serial_number)

    @classmethod
    def get_all_active_devices(cls):
        return Device.query()


class Connection(ndb.Model):
    device = ndb.KeyProperty(kind="Device")
    internal_ip = ndb.StringProperty()
    timestamp = ndb.DateTimeProperty(auto_now=True)

class MoistureLog(ndb.Model):
    device = ndb.KeyProperty(kind="Device")
    timestamp = ndb.DateTimeProperty(auto_now=True)
    status = ndb.BooleanProperty()

class PumpSession(ndb.Model):
    device = ndb.KeyProperty(kind="Device")
    open_stamp = ndb.DateTimeProperty()
    close_stamp = ndb.DateTimeProperty()
    complete = ndb.BooleanProperty()


class WaterSechdule(ndb.Model):
    device = ndb.KeyProperty(kind="Device")
    datetime = ndb.DateTimeProperty()
    days_in_week = ndb.IntegerProperty(repeated=True)
    time_in_second = ndb.IntegerProperty()
```

src/script.py

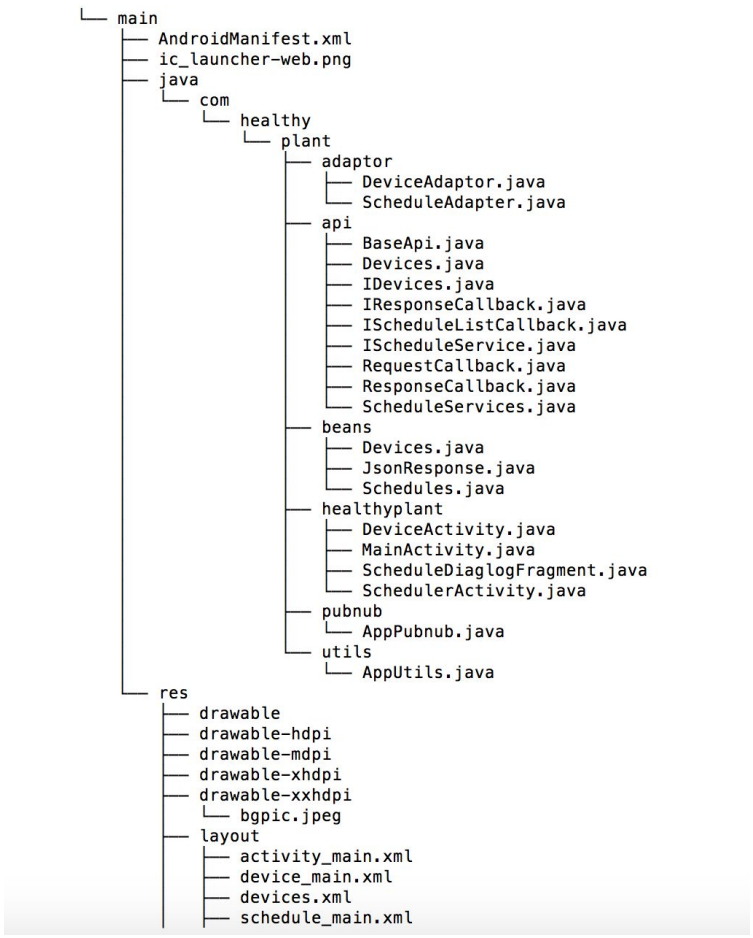
```
from models import (
       Device,
       )


def create_device():

       d = Device(
              serial_number="5418393301680128",
              publish_key="pub-c-6e26ca7f-f358-474a-8b58-74cf5ff76be0",
```

```
        subscribe_key="sub-c-fd449834-4bb8-11e6-82fe-0619f8945a4f")
d.put()
```

Android Application Code (Healthy Plant Android App)



```
│   ├── schedule_main.xml
│   ├── scheduleeditview.xml
│   └── schedules.xml
├── menu
│   └── menu_main.xml
├── mipmap-hdpi
│   └── ic_launcher.png
├── mipmap-mdpi
│   └── ic_launcher.png
├── mipmap-xhdpi
│   └── ic_launcher.png
├── mipmap-xxhdpi
│   └── ic_launcher.png
├── mipmap-xxxhdpi
│   └── ic_launcher.png
├── values
│   ├── dimens.xml
│   ├── strings.xml
│   └── styles.xml
├── values-v21
│   └── styles.xml
└── values-w820dp
    └── dimens.xml
├── build
│   └── intermediates
│       ├── dex-cache
│       │   └── cache.xml
│       └── gradle_project_sync_data.bin
├── build.gradle
├── google-services.json
├── gradle
│   └── wrapper
│       ├── gradle-wrapper.jar
│       └── gradle-wrapper.properties
├── gradle.properties
├── gradlew
├── gradlew.bat
├── local.properties
└── settings.gradle
```

AndroidManifest.XML

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.healthy.plant.healthyplant" >

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".DeviceActivity"
            android:label="@string/app_name" >

        </activity>
        <activity
            android:name=".SchedulerActivity"
            android:label="@string/schedulerActivity" >

        </activity>
    </application>
```

```
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />


</manifest>
```

pubnub/AppPubnub.java

```
package com.healthy.plant.pubnub;

import com.pubnub.api.Pubnub;

/**
* Created by anishpoudel on 8/9/16.
*/
public class AppPubnub {

    private  static Pubnub pubnub = null;
    private static final String publishKey = "pub-c-6e26ca7f-f358-474a-8b58-74cf5ff76be0";
    private static final String subscriberKey = "sub-c-fd449834-4bb8-11e6-82fe-0619f8945a4f";

    public static Pubnub getInstance(){
        if(pubnub == null){
            pubnub = new Pubnub(publishKey, subscriberKey);
        }
        return pubnub;
    }
}
```

healthyPlant/MainActivity.java

```
package com.healthy.plant.healthyplant;

import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;

import com.healthy.plant.adaptor.DeviceAdaptor;
import com.healthy.plant.api.Devices;
import com.healthy.plant.api.IDevices;
```

```
import java.net.URL;
import java.util.ArrayList;
import java.util.List;


public class MainActivity extends Activity implements IDevices {

   ProgressDialog pd;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_main);
       Devices d = new Devices(this);
       d.execute();
       pd = new ProgressDialog(this);
       pd.setMessage("Loading Devices");
       pd.setCancelable(false);
       pd.setInverseBackgroundForced(false);
       pd.show();
       //getGCMToken();
   }

   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
       // Inflate the menu; this adds items to the action bar if it is present.
       getMenuInflater().inflate(R.menu.menu_main, menu);
       return true;
   }

   @Override
   public boolean onOptionsItemSelected(MenuItem item) {
       // Handle action bar item clicks here. The action bar will
       // automatically handle clicks on the Home/Up button, so long
       // as you specify a parent activity in AndroidManifest.xml.
       int id = item.getItemId();

       //noinspection SimplifiableIfStatement
       if (id == R.id.action_settings) {
           return true;
       }

       return super.onOptionsItemSelected(item);
   }

   @Override
   public void onDeviceReadyCallback(List<com.healthy.plant.beans.Devices> d) {
       System.out.println("Callback is called");
       for (com.healthy.plant.beans.Devices dd : d){
           System.out.println(dd.toString());
```

```
        }
        DeviceAdaptor da = new DeviceAdaptor(this,
(ArrayList<com.healthy.plant.beans.Devices>)d);
        ListView listView = (ListView) findViewById(R.id.devicelistView);
        listView.setAdapter(da);
        listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
                System.out.println("Clicked on the view");
                com.healthy.plant.beans.Devices d = (com.healthy.plant.beans.Devices)
parent.getItemAtPosition(position);
                System.out.println(d.toString());
                if (d.getIsActive()) {
                    Intent intent = new Intent(getApplicationContext(), DeviceActivity.class);
                    startActivity(intent);
                }


            }
        });
        pd.hide();
    }
}
```

healthyPlant/DeviceActivity.java

```
package com.healthy.plant.healthyplant;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.util.AttributeSet;
import android.util.JsonReader;
import android.view.View;
import android.widget.CompoundButton;
import android.widget.RelativeLayout;
import android.widget.Switch;
import android.widget.TextView;
import android.widget.Toast;

import com.healthy.plant.api.IResponseCallback;
import com.healthy.plant.api.RequestCallback;
import com.healthy.plant.api.ResponseCallback;
import com.healthy.plant.pubnub.AppPubnub;
import com.healthy.plant.utils.AppUtils;
import com.pubnub.api.Callback;
import com.pubnub.api.Pubnub;
```

```
import com.pubnub.api.PubnubError;

import org.json.JSONException;
import org.json.JSONObject;

/**
* Created by anishpoudel on 8/6/16.
*/
public class DeviceActivity extends Activity implements IResponseCallback {

   private Pubnub pubnub;
   private static TextView isMoistTV;
   private Boolean isMoist;
   private RelativeLayout waterSchedule;
   private static Switch pumpSwitch;
   private AppUtils utils;
   private Boolean pumpSwitchState = false;
   private String pumpStatusMsg = null;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);

       setContentView(R.layout.device_main);
       pumpSwitch = (Switch) findViewById(R.id.pump_switch);

       isMoistTV = (TextView) findViewById(R.id.is_moist);
       waterSchedule = (RelativeLayout) findViewById(R.id.viewSchedule);
       waterSchedule.setOnClickListener(new View.OnClickListener() {
           @Override
           public void onClick(View v) {
               Intent intent = new Intent(getApplicationContext(), SchedulerActivity.class);
               startActivity(intent);
           }
       });
       System.out.println("The current text is " + isMoistTV.getText());
       pubnub = AppPubnub.getInstance();

       pumpSwitch.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
           @Override
           public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
               String command = "TURN_PUMP_OFF";
               if (isChecked){
                   command = "TURN_PUMP_ON";
               }
               pubnub.publish("healthy-plan-request-channel", utils.createJson("command",
command).toString(), new RequestCallback() );
           }
       });
```

```
        pubnub.publish("healthy-plan-request-channel", utils.createJson("command",
"CHECK_MOISTURE").toString(), new RequestCallback());
        try{
            pubnub.subscribe("healthy-plan-response-channel", new ResponseCallback(this));
        }catch (Exception e){
            System.out.print("Error in sub..");
            System.out.println(e.getMessage());
        }
    }

    private void updateStatusText(String status){
        this.isMoistTV.setText(status);
    }


    @Override
    public void onDetachedFromWindow() {
        super.onDetachedFromWindow();
        System.out.println("***Detached from window***");

    }

    @Override
    public void moistureRespCallback(final Boolean isMoists) {
        System.out.println("(((((((((" + isMoists.toString());
        this.isMoist = isMoists;
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                System.out.println("Moisture value inside");
                System.out.println(isMoist);
                if (isMoist) {
                    System.out.println("in true statement");
                    updateStatusText("Moist");
                } else {
                    System.out.println("in false statement");

                    updateStatusText("Dry");
                }

            }
        });

    }

    @Override
    public void pumpRespCallback(String message, boolean value) {
        System.out.println("PPP");
        pumpStatusMsg = message;
```

```
        pumpSwitchState = value;
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                if (pumpStatusMsg != null){
                    Toast toast = Toast.makeText(getApplicationContext(), pumpStatusMsg,
Toast.LENGTH_SHORT);
                    toast.show();
                }
                pumpSwitch.setChecked(pumpSwitchState);

                }
            });


    }


}
```

healthyPlant/ScheduleActivity.java

```
package com.healthy.plant.healthyplant;

import android.app.Activity;
import android.app.DialogFragment;
import android.app.ProgressDialog;
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;

import com.healthy.plant.adaptor.ScheduleAdapter;
import com.healthy.plant.api.IScheduleListCallback;
import com.healthy.plant.beans.Schedules;

import java.util.ArrayList;

/**
* Created by anishpoudel on 8/7/16.
*/
public class SchedulerActivity extends FragmentActivity implements IScheduleListCallback,
ScheduleDiaglogFragment.ScheduleDiaglogListner {

   ProgressDialog pd;
```

```
ArrayList<Schedules> schedules;
ScheduleAdapter sa;
ListView lv;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.schedule_main);
    scheduleListCallback(null);

}



@Override
public void scheduleListCallback(ArrayList<Schedules> schedulesd) {
    schedules = new ArrayList<Schedules>();
    schedules.add(new Schedules(3,4,"pm",new String[]{"SUN", "MON"}));
    schedules.add(new Schedules(5,4,"pm",new String[]{"TUE", "MON"}));
    schedules.add(new Schedules(6,4,"pm",new String[]{"THR", "MON"}));
    schedules.add(new Schedules(7,4,"pm",new String[]{"FRI", "MON"}));

    sa = new ScheduleAdapter(this, schedules);
    lv = (ListView) findViewById(R.id.scheduleListview);
    lv.setAdapter(sa);
    sa.notifyDataSetChanged();
    lv.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
            ScheduleDiaglogFragment dialog = new ScheduleDiaglogFragment();
            dialog.setSchedules((Schedules) parent.getItemAtPosition(position));
            dialog.setPosition(position);
            dialog.show(getFragmentManager(), "Edit Schedule");
        }
    });


}

@Override
public void onSave(DialogFragment dialog) {
    ScheduleDiaglogFragment d = (ScheduleDiaglogFragment)dialog;
    System.out.println("**************");
    System.out.println(d.getSchedules().getDaysToString());
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            sa.notifyDataSetChanged();
            lv.deferNotifyDataSetChanged();
        }
```

```
        });
    }

    @Override
    public void onCacel(DialogFragment dialog) {

    }
}
```

healthyPlant/ScheduleDialogFragment.java

```
package com.healthy.plant.healthyplant;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.app.DialogFragment;
import android.content.DialogInterface;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.CheckBox;
import android.widget.EditText;

import com.healthy.plant.beans.Schedules;

import java.util.ArrayList;

/**
* Created by anishpoudel on 8/14/16.
*/
public class ScheduleDiaglogFragment extends DialogFragment {

    private Schedules schedules;
    private Integer position;
    EditText time;
    CheckBox cbsun;
    CheckBox cbmon;
    CheckBox cbtue;
    CheckBox cbwed;
    CheckBox cbthu;
    CheckBox cbfri;
    CheckBox cbsat;


    public void setSchedules(Schedules schedule){
        this.schedules = schedule;
    }
```

```
public Schedules getSchedules(){

    String t = time.getText().toString();
    ArrayList<String> days = new ArrayList<String>();
    if(t.trim().length()>0){
        String[] ts = t.split(":");
        schedules.setHour(Integer.parseInt(ts[0].trim()));
        schedules.setMin(Integer.parseInt(ts[1].trim()));
    }
    if(cbsun.isChecked())
        days.add("SUN");
    if(cbmon.isChecked())
        days.add("MON");
    if(cbtue.isChecked())
        days.add("TUE");
    if(cbwed.isChecked())
        days.add("WED");
    if(cbthu.isChecked())
        days.add("THU");
    if(cbfri.isChecked())
        days.add("FRI");
    if(cbsat.isChecked())
        days.add("SAT");

    schedules.setDays(days.toArray(new String[days.size()]));

    return schedules;
}

public interface ScheduleDiaglogListner{
    public void onSave(DialogFragment dialog);
    public void onCacel(DialogFragment dialog);
}

public void setPosition(Integer position){
    this.position = position;
}

public Integer getPosition(){
    return this.position;
}

ScheduleDiaglogListner sListner;



@Override
public void onAttach(Activity activity) {
    super.onAttach(activity);
```

```
    try{
        sListner = (ScheduleDiaglogListner) activity;
    }catch (ClassCastException e){
        throw new ClassCastException(activity.toString()
                + "must implement ScheduleDialoglistner");
    }
}

private void initFields(View v){

    time = (EditText) v.findViewById(R.id.hrmintime);
    cbsun = (CheckBox) v.findViewById(R.id.cbsun);
    cbmon = (CheckBox) v.findViewById(R.id.cbmon);
    cbtue = (CheckBox) v.findViewById(R.id.cbtue);
    cbwed = (CheckBox) v.findViewById(R.id.cbwed);
    cbthu = (CheckBox) v.findViewById(R.id.cbthu);
    cbfri = (CheckBox) v.findViewById(R.id.cbfri);
    cbsat = (CheckBox) v.findViewById(R.id.cbsat);
    if (schedules != null){
        time.setText(schedules.getTimeToString());
        for (String s : schedules.getDays()){
            switch (s){
                case "SUN":
                    cbsun.setChecked(true);
                    break;
                case "MON":
                    cbmon.setChecked(true);
                    break;
                case "TUE":
                    cbtue.setChecked(true);
                    break;
                case "WED":
                    cbwed.setChecked(true);
                    break;
                case "THR":
                    cbthu.setChecked(true);
                    break;
                case "FRI":
                    cbfri.setChecked(true);
                    break;
                case "SAT":
                    cbsat.setChecked(true);
                    break;
            }
        }

    }


}
```

```
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
        LayoutInflater inflater = getActivity().getLayoutInflater();
        final View aa = inflater.inflate(R.layout.scheduleeditview, null);
        builder.setView(aa)
                .setPositiveButton("Save", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        sListner.onSave(ScheduleDiaglogFragment.this);
                    }
                })
                .setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        ScheduleDiaglogFragment.this.getDialog().cancel();
                    }
                });

        initFields(aa);
        return builder.create();
    }
}
```

utils/AppUtils.java

```
package com.healthy.plant.utils;

import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.HashMap;

/**
* Created by anishpoudel on 8/13/16.
*/
public class AppUtils {

    public static JSONObject createJson(String channel, String command){
        JSONObject j = new JSONObject();
        try{
           j.put(channel, command);
        }catch (JSONException e){
            System.out.println(e);
        }
        return j;
```

```
    }
}
```

beans/Devices.java

```
package com.healthy.plant.beans;

import com.fasterxml.jackson.annotation.JsonProperty;

/**
* Created by anishpoudel on 8/6/16.
*/
public class Devices {
    private String name;
    private String serial_number;
    private String publish_key;
    private String subscribe_key;
    private Boolean isActive = true;


    public Devices(@JsonProperty("serial_number") String serial_number,
@JsonProperty("publish_key") String publish_key, @JsonProperty("subscribe_key") String
subscribe_key) {
        this.subscribe_key = subscribe_key;
        this.serial_number = serial_number;
        this.publish_key = publish_key;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getSerial_number() {
        return serial_number;
    }

    public void setSerial_number(String serial_number) {
        this.serial_number = serial_number;
    }

    public String getPublish_key() {
        return publish_key;
    }
```

```
    public void setPublish_key(String publish_key) {
        this.publish_key = publish_key;
    }

    public String getSubscribe_key() {
        return subscribe_key;
    }

    public void setSubscribe_key(String subscribe_key) {
        this.subscribe_key = subscribe_key;
    }

    public Boolean getIsActive() {
        return isActive;
    }

    public void setIsActive(Boolean isActive) {
        this.isActive = isActive;
    }

    @Override
    public String toString() {
        StringBuilder builder = new StringBuilder();
        builder.append(this.getSerial_number().toString());
        builder.append(this.subscribe_key.toString());
        builder.append(this.publish_key.toString());
        return builder.toString();
    }
}
```

beans/JsonResponse.java

```
package com.healthy.plant.beans;

/**
* Created by anishpoudel on 8/6/16.
*/
public class JsonResponse {
    private String request;

}
```

beans/Scheduler.java

```
package com.healthy.plant.beans;
```

```java
import java.util.ArrayList;
import java.util.List;

/**
* Created by anishpoudel on 8/14/16.
*/
public class Schedules {
   private Integer hour;
   private Integer min;

   private String[] days = new String[6];

   public Schedules(int hour, int min, String dayInd, String[] days){
       setHour(hour);
       setMin(min);
       setDays(days);
   }
   public Schedules(){};

   public Integer getHour() {
       return hour;
   }

   public void setHour(Integer hour) {
       this.hour = hour;
   }

   public Integer getMin() {
       return min;
   }

   public void setMin(Integer min) {
       this.min = min;
   }



   public String[] getDays() {
       return days;
   }

   public void setDays(String[] days) {
       this.days = days;
   }

   public String getDaysToString(){
       StringBuilder sb = new StringBuilder();
       for (String d : getDays()){
           sb.append(d + " ");
       }
```

```
        return sb.toString();
    }

    public String getTimeToString(){
        return getHour() + " : " + getMin();
    }
}
```

api/BaseApi.java

```
package com.healthy.plant.api;

import android.util.Log;

import org.json.JSONObject;

import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.Objects;

/**
* Created by anishpoudel on 8/6/16.
*/
public class BaseApi {

    private final String API_BASE = "http://healthy-plant.appspot.com/";

    private HttpURLConnection conn = null;


    private URL makeURL(String url) throws MalformedURLException{

        URL u = new URL(API_BASE+url);
        return u;
    }

    public Object get(String url){
        try {
            URL _url = makeURL(url);
            conn = (HttpURLConnection) _url.openConnection();
            conn.setRequestMethod("GET");
            InputStream is = conn.getInputStream();
            BufferedReader rd = new BufferedReader(new InputStreamReader(is));
            String line;
            StringBuffer response = new StringBuffer();
```

```
            while ((line = rd.readLine()) != null) {
                response.append(line);
                response.append('\r');
            }
            rd.close();
            String responseStr = response.toString();
            Log.d("Server response", responseStr);
            return responseStr;


        }catch (Exception e){
            System.out.println(e.getMessage());
        }


        return null;
    }


}
```

api/Devices.java

```
package com.healthy.plant.api;

import android.app.Activity;
import android.os.AsyncTask;

import com.fasterxml.jackson.core.JsonParseException;
import com.fasterxml.jackson.databind.ObjectMapper;

import java.net.HttpURLConnection;
import java.net.URL;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

/**
* Created by anishpoudel on 8/6/16.
*/
public class Devices extends AsyncTask<String, Integer, String> {

    private IDevices idevices;


    public Devices(IDevices devices) {
        this.idevices = devices;
    }

    @Override
```

```
    protected String doInBackground(String... params) {
        BaseApi ba = new BaseApi();
        Object response = ba.get("api/devicelist");
        System.out.println(response.toString());
        return response.toString();
    }

    @Override
    protected void onPostExecute(String response){
        System.out.println("************ d ************");
        try{
            ObjectMapper mapper = new ObjectMapper();
            List<com.healthy.plant.beans.Devices> d = new
ArrayList<com.healthy.plant.beans.Devices>(Arrays.asList(mapper.readValue(response,
com.healthy.plant.beans.Devices[].class)));
            System.out.println("************ d ************");
            this.idevices.onDeviceReadyCallback(d);
        }catch(Exception e){
            System.out.println("Exception has occured");
            System.out.println(e.getMessage());
        }

    }

    @Override
    protected void onProgressUpdate(Integer... values) {
        super.onProgressUpdate(values);
    }
}
```

api/IDevice.java

```
package com.healthy.plant.api;

import com.healthy.plant.beans.*;
import com.healthy.plant.beans.Devices;

import java.util.List;

/**
* Created by anishpoudel on 8/6/16.
*/
public interface IDevices {

    public void onDeviceReadyCallback(List<Devices> d);
}
```

api/IResponseCallback.java

```
package com.healthy.plant.api;

/**
* Created by anishpoudel on 8/7/16.
*/
public interface IResponseCallback {

   public void moistureRespCallback(Boolean isMoist);
   public void pumpRespCallback(String message, boolean result);

}
```

api/IScheduleListCallback.java

```
package com.healthy.plant.api;

import com.healthy.plant.beans.Schedules;

import java.util.ArrayList;
import java.util.List;

/**
* Created by anishpoudel on 8/14/16.
*/
public interface IScheduleListCallback {

   public void scheduleListCallback(ArrayList<Schedules> sechdules);
}
```

api/RequestCallback.java

```
package com.healthy.plant.api;

import com.pubnub.api.Callback;
import com.pubnub.api.PubnubError;

import org.json.JSONException;
import org.json.JSONObject;

/**
* Created by anishpoudel on 8/6/16.
*/
public class RequestCallback extends Callback {

   public void successCallback(String channel, Object response) {
```

```
        System.out.println(response.toString());



    }
    public void errorCallback(String channel, PubnubError error) {
        System.out.println(error.toString());
    }
}
```

api/ResponseCallback.java

```
package com.healthy.plant.api;

import android.content.Context;
import android.widget.Toast;

import com.pubnub.api.Callback;
import com.pubnub.api.PubnubError;

import org.json.JSONException;
import org.json.JSONObject;

/**
* Created by anishpoudel on 8/6/16.
*/
public class ResponseCallback extends Callback {

    private IResponseCallback responseCallback;

    public ResponseCallback( IResponseCallback responseCallback){
        this.responseCallback = responseCallback;
    }

    private String parseResponse(JSONObject jobj) throws Exception{
        JSONObject response = jobj.getJSONObject("response");
        String data = response.getString("data");
        System.out.println("The data is is " + data.toString());
        return data.toString();
    }



    public void successCallback(String channel, Object response) {
        System.out.println(response.toString());
        try{
            JSONObject jobj = (JSONObject)response;//new JSONObject((JSONObject)response);
            JSONObject req = jobj.getJSONObject("request");
            String command = req.getString("command");
            String resp = parseResponse(jobj);
```

```
            switch (command){
                case "CHECK_MOISTURE":
                    if (resp.toString() == "true"){
                        responseCallback.moistureRespCallback(true);
                    }else{
                        responseCallback.moistureRespCallback(false);
                    }
                    break;
                case "TURN_PUMP_ON":
                    if (resp.toString() == "true"){
                        responseCallback.pumpRespCallback("Pump has turned on", true);
                    }else{
                        responseCallback.pumpRespCallback("Seems like soil is already moist",
false);
                    }
                    break;
                case "TURN_PUMP_OFF":
                    responseCallback.pumpRespCallback("Turning the pump off!", false);
                    break;

            }


        }catch (Exception e){
            System.out.println("Json parsor error");
            System.out.println(e.getMessage());

        }


    }
    public void errorCallback(String channel, PubnubError error) {

        System.out.println(error.toString());
    }
}
```

api/ScheduleService.java

```
package com.healthy.plant.api;

import android.os.AsyncTask;

/**
* Created by anishpoudel on 8/14/16.
*/
public class ScheduleServices extends AsyncTask<String, Integer, String> {
    @Override
```

```
    protected String doInBackground(String... params) {
        BaseApi ba = new BaseApi();
        Object response = ba.get("api/devicelist");
        System.out.println(response.toString());
        return response.toString();
    }

    @Override
    protected void onPostExecute(String s) {
        super.onPostExecute(s);
    }
}
```

adaptor/DeviceAdaptor.java

```
package com.healthy.plant.adaptor;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;

import com.healthy.plant.beans.Devices;
import com.healthy.plant.healthyplant.R;

import java.util.ArrayList;

/**
* Created by anishpoudel on 8/6/16.
*/
public class DeviceAdaptor extends ArrayAdapter<Devices> {

    public DeviceAdaptor(Context context, ArrayList<Devices> devices){
        super(context, 0, devices);
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        Devices devices = getItem(position);
        if (convertView == null){
            convertView = LayoutInflater.from(getContext()).inflate(R.layout.devices, parent,
false);
        }
        TextView dname = (TextView) convertView.findViewById(R.id.dname);
        dname.setText(devices.getSerial_number());

        return convertView;
```

```
    }


}
```

adaptor/ScheduleAdaptor.java

```
package com.healthy.plant.adaptor;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;

import com.healthy.plant.beans.Devices;
import com.healthy.plant.beans.Schedules;
import com.healthy.plant.healthyplant.R;

import java.util.ArrayList;

/**
* Created by anishpoudel on 8/14/16.
*/
public class ScheduleAdapter extends ArrayAdapter<Schedules> {

    public ScheduleAdapter(Context context, ArrayList<Schedules> devices){
        super(context, 0, devices);
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        Schedules schedule = getItem(position);
        if (convertView == null){
            convertView = LayoutInflater.from(getContext()).inflate(R.layout.schedules, parent,
false);
        }

        TextView timeView = (TextView) convertView.findViewById(R.id.timeView);
        timeView.setText(schedule.getTimeToString());

        TextView daysView = (TextView) convertView.findViewById(R.id.dayView);
        daysView.setText(schedule.getDaysToString());

        return convertView;

    }

}
```

layout/activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
   android:layout_height="match_parent"  tools:context=".MainActivity">

   <LinearLayout
       android:orientation="vertical"
       android:layout_width="match_parent"
       android:layout_height="match_parent"
       android:layout_alignParentStart="true"
       android:layout_alignParentTop="true">
       <ListView
           android:layout_width="match_parent"
           android:layout_height="wrap_content"
           android:id="@+id/devicelistView"
           android:layout_gravity="center_horizontal" />
   </LinearLayout>
</RelativeLayout>
```

layout/device_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical" android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:weightSum="1">


   <ImageView
       android:layout_width="2956dp"
       android:layout_height="wrap_content"
       android:id="@+id/displayimg"
       android:layout_gravity="center_horizontal"
       android:src="@drawable/bgpic"
       android:layout_weight="0.71" />

   <RelativeLayout
       android:layout_width="match_parent"
       android:layout_height="77dp"
       android:textAlignment="inherit"
       android:background="@color/f2f2f2"
       android:layout_marginBottom="2dp"
       android:backgroundTint="@color/f2f2f2">

       <TextView
           android:layout_width="match_parent"
```

```
        android:layout_height="match_parent"
        android:text="Moisture Level"
        android:id="@+id/sdd"
        android:layout_gravity="center"
        android:textAlignment="center"
        android:textSize="18dp"
        android:textStyle="normal"

        android:singleLine="true"
        android:paddingLeft="10dp"
        android:paddingBottom="10dp"
        android:layout_marginTop="20dp"
        android:layout_alignParentBottom="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Unknown"
        android:id="@+id/is_moist"
        android:layout_marginEnd="33dp"
        android:layout_alignTop="@+id/sdd"
        android:layout_alignParentEnd="true"
        android:clickable="true"/>


</RelativeLayout>
<View
    android:layout_width="fill_parent"
    android:layout_height="1dip"
    android:background="#d9d9d9" />

<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="77dp"
    android:textAlignment="inherit"
    android:background="@color/f2f2f2"
    android:layout_marginBottom="2dp"
    android:backgroundTint="@color/f2f2f2">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Pump"
        android:id="@+id/s"
        android:layout_gravity="center"
        android:textAlignment="center"
        android:textSize="18dp"
        android:textStyle="normal"

        android:singleLine="true"
```

```
        android:paddingLeft="10dp"
        android:paddingBottom="10dp"
        android:layout_marginTop="20dp" />

    <Switch
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/pump_switch"
        android:layout_alignTop="@+id/s"
        android:layout_alignParentEnd="true"
        android:layout_marginEnd="40dp"
        android:checked="false" />


</RelativeLayout>
<View
    android:layout_width="fill_parent"
    android:layout_height="1dip"
    android:background="#d9d9d9" />

<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="77dp"
    android:textAlignment="inherit"
    android:background="@color/f2f2f2"
    android:layout_marginBottom="2dp"
    android:backgroundTint="@color/f2f2f2">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Auto Watering"
        android:id="@+id/ss"
        android:layout_gravity="center"
        android:textAlignment="center"
        android:textSize="18dp"
        android:textStyle="normal"

        android:singleLine="true"
        android:paddingLeft="10dp"
        android:paddingBottom="10dp"
        android:layout_marginTop="20dp" />

    <Switch
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/switch12"
        android:layout_alignTop="@+id/ss"
        android:layout_alignParentEnd="true"
        android:layout_marginEnd="40dp"
```

```
            android:checked="false" />


    </RelativeLayout>
    <View
        android:layout_width="fill_parent"
        android:layout_height="1dip"
        android:background="#d9d9d9" />

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="77dp"
        android:textAlignment="inherit"
        android:background="@color/f2f2f2"
        android:layout_marginBottom="2dp"
        android:backgroundTint="@color/f2f2f2"
        android:id="@+id/viewSchedule">

        <TextView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:text="Schedule Watering"
            android:id="@+id/waterScheduler"
            android:layout_gravity="center"
            android:textAlignment="center"
            android:textSize="18dp"
            android:textStyle="normal"

            android:singleLine="true"
            android:paddingLeft="10dp"
            android:paddingBottom="10dp"
            android:layout_marginTop="20dp"
            />




    </RelativeLayout>


</LinearLayout>
```

layout/device.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
android:layout_height="?android:attr/listPreferredItemHeight"
    android:id="@+id/deviceLayout"
    android:background="@color/f2f2f2"
```

```
    android:gravity="bottom">

    <ImageView
        android:layout_width="60dp"
        android:layout_height="fill_parent"
        android:id="@+id/imageView"
        android:layout_marginRight="20dp"
        android:src="@android:drawable/btn_star"
        android:layout_alignBottom="@+id/textView2"
        android:layout_alignTop="@+id/textView2"
        android:layout_alignParentStart="true" />


    <TextView
        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="Device 1"
        android:id="@+id/dname"
        android:layout_alignParentTop="true"
        android:paddingTop="20dp"
        android:paddingBottom="20dp"
        android:layout_toEndOf="@+id/imageView" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceSmall"
        android:text="Status"
        android:id="@+id/textView2"
        android:padding="10dp"
        android:layout_centerVertical="true"
        android:layout_alignParentEnd="true" />
</RelativeLayout>
```

layout/schedule_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent"
     tools:context=".SchedulerActivity">

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
```

```
        android:layout_alignParentStart="true"
        android:layout_alignParentTop="true">
        <ListView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/schedulelistview"
            android:layout_gravity="center_horizontal" />
    </LinearLayout>
</RelativeLayout>
```

layout/scheduleeditview.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent">


    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Time in 24hr format"
        android:id="@+id/textView"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:paddingTop="30dp" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="time"
        android:ems="10"
        android:id="@+id/hrmintime"
        android:layout_below="@+id/textView"
        android:layout_centerHorizontal="true"
        android:textAlignment="center"
        android:textStyle="bold"
        android:gravity="center_vertical|center_horizontal" />

    <TableLayout
        android:layout_width="match_parent"
        android:layout_height="100dp"
        android:layout_below="@+id/hrmintime"
        android:layout_centerHorizontal="true"


        android:id="@+id/sss"
        android:gravity="center_vertical">
```

```
<TableRow
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center_horizontal"
    android:paddingTop="10dp"
    android:paddingBottom="10dp">

    <CheckBox
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="SUN"
        android:id="@+id/cbsun"
        android:layout_column="7"
        android:checked="false" />

    <CheckBox
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="MON"
        android:id="@+id/cbmon"
        android:layout_column="8"
        android:checked="false" />

    <CheckBox
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TUE"
        android:id="@+id/cbtue"
        android:layout_column="9"
        android:checked="false" />

    <CheckBox
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="WED"
        android:id="@+id/cbwed"
        android:layout_column="10"
        android:checked="false" />
</TableRow>
<TableRow
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center_horizontal"
    android:paddingBottom="20dp">

    <CheckBox
        android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:text="THU"
            android:id="@+id/cbthu"
            android:layout_column="7"
            android:checked="false" />

        <CheckBox
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="FRI"
            android:id="@+id/cbfri"
            android:layout_column="8"
            android:checked="false" />

        <CheckBox
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="SAT"
            android:id="@+id/cbsat"
            android:layout_column="9"
            android:checked="false" />

    </TableRow>


</TableLayout>

</RelativeLayout>
```

layout/schedule.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
android:layout_height="?android:attr/listPreferredItemHeight"
    android:id="@+id/schedulerLayout"
    android:gravity="center_vertical">


    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="3:45 pm"
        android:id="@+id/timeView"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true"
        android:paddingLeft="20dp"
        android:paddingBottom="10dp" />
```

```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceSmall"
        android:text="SU, MO, TU, WD, TH, FR, SA"
        android:id="@+id/dayView"
        android:paddingLeft="30dp"
        android:layout_below="@+id/timeView"
        android:layout_centerHorizontal="true"
        android:layout_alignLeft="@+id/timeView" />

</RelativeLayout>
```