User Manual

Central Forecasting Server

The Central Forecasting Server (CFS) is comprised of two separate computers: The CET-SIF server, and the CET-Processing computer.

CET-SIF Server

The CET-SIF server is equipped with a static IP address, which allows the Remote Smartphone Sensor (RSS) and Power Data Module (PDM) to communicate with the server without being connected to the same network. This computer handles the downloading of incoming images and power measurements from the other two subsystems and then sends these incoming messages to the CET-Processing computer so that they can processed into a forecast appropriately.

Folder Structure

The server code base root is located within the **sif** user account under **~/SIF_Server/**, where '~' is the unix/linux shorthand for home**.** Within it, there are two directories, and a number of files. All of the files and the outputFiles directory are considered to be common to both this server and the CET-research computer.

The 'sockets' directory contains socketServer.py, which is the python script which runs the non blocking socket server. This file can be considered the main file on the server computer, since all other programs on this computer are ultimately called as a result of calling socketServer.py.

The outputFiles directory is a particularly important common directory to both the CET-SIF server and the CET-Research computer. This folder is the 'Processing Queue' for our system. The server places any incoming images at the back of the queue by placing it in the Unprocessed folder. Note that while the directories within outputFiles are the same on both CET-SIF and CET-Research, the contents may not always be the same. This is because we only sync new images to CET-Research, and we do not sync any anything that may happen to it back to CET-SIF.

Run Instructions

To run the server code, you need to VPN into the CET network. If you do not know how to do this, you will need to talk to a CET employee for help. With vpn access, you can access the CET-SIF server computer via ssh by calling: *ssh sif@cet-sif.colorado.edu.* You will need the password to access this computer (ask Eric McIntyre or Al Gasiewski to get this password).

cd into ~/SIF_Server then call: *python sockets/socketServer.py* to call activate the server. The prompt will let you know what IP address the server is listening for connections on. Note that this IP address, assuming you are running it on CET-SIF and not your local computer, is the domain name resolution of cet-sif.colorado.edu. To activate this program in debug mode, which prints

more information which may be useful for debugging, just include ‘debug’ in the program call (i.e. *python sockets/socketServer.py debug*).

Once the program is running, the functionality of everything else should be automated. Images will be downloaded whenever a smartphone makes a connection (given it has our message signature, which is required for security purposes). Power data will be saved to the database if a PDM sends a measurement (which must also have the signature). Note that the server has logic in place to distinguish an incoming image from an incoming power data measurement, so no other identification needs to be done. Following any successful transmission, the program automatically sends this information over to the CET-Research computer via the rsync module.

As of the writing of this manual, the Office of Information Technology (OIT) has not permitted any firewall exceptions for port 5000 on the CU Boulder Firewall. As a result, sensors can only communicate when on the CU Boulder network. Note that sensors do NOT need to be connected to the CET Network via VPN or otherwise to communicate, since the CET firewall does have firewall exceptions in place for port 5000 of cet-sif.colorado.edu.

CET-Research Processing Computer

The CET-Research Processing computer performs all of the image processing, forecast creation, and forecast distribution of the CFS. This computer is necessary because it is equipped with MATLAB, and a NX Server which one can connect to via an NX Client like NoMachine Player (http://www.nomachine.com/) for a remote desktop. For access to this computer, please talk to Al Gasiewski or Eric McIntyre. Note that the password for this computer is different than that for CET-SIF.

Images arrive within the file structure shortly after the CET-SIF server receives one from a RSS. Power data measurements arrive in a similar manner, except that they are simply added to the CapstoneDatabase.db file rather than to the file structure.

Folder Structure

The code base is within the **sif** user account under **~/SIF_Processing**. Within that is a number of common files and one common directory which is explained furthur in the CET-SIF section, and the ‘Matlab’ and ‘Forecast’ subdirectories.

The Matlab directory contains all image processing MATLAB code and a python file called ’MatlabHandler.py’ which facilitates interaction between the python portions of the code and the MATLAB portions of the code.

The Forecast directory contains anything relating to the creation of Google Earth forecasts. This includes the makeKML.py file which interprets the results of the MATLAB processing step, and generates a KML forecast file. KML file generation is done through the KMLtemplate.tmpl

template file. This template is written using the Cheetah Templating Engine's language (http://www.cheetahtemplate.org/).
Once a KML file is generated, it is placed within **~/public_html/** folder. This folder was made to be a publically accessible directory, so that users do not need to be on the CET network to view a Google Earth Forecast.

NetworkLink.kml is the only file one needs to view the generated forecasts from any computer. This file is not included in either the CET-SIF or the CET-Research code base. a KML file that is not present on either the CET-SIF codebase. NetworkLink.kml simply creates a connection from whichever computer the file is located on to the generated KML files for viewing in Google Earth. Once Google Earth is installed, one only needs to double click it to use it.

Run Instructions

Running the processing code is done in a similar manner as running the CET-SIF server. Just cd into ~/SIF_Processing, and run: *python Matlab/MatlabHandler.py* to run the whole codebase. Include the optional debug argument for additional output which may be helpful for debugging.

Assuming all other subsystems and the CET-SIF server are running, then the program should recognize all incoming information and produce forecasts automatically. A remote user with NetworkLink.kml would be able to click refresh within google earth to view any generated forecast files.

On-Grid PV Array Power Data Module (PDM)

The PDM is the residential component that is placed on a structure that has an existing solar panel. This module will read data from the solar panel on the roof and in real-time send this information to the server. This data is more of an affirmation of what is happening and if the forecasts are accurate.

Setup

The PCB in the appendix details out the schematic for how the PDM. The first step is to plug in the PCB to the 120V of the residence into a wall outlet. Then take the MSP430-UFET debugger and connect it to a computer running Code Composer Studio (CCS). Using the JTAG connection open up the software labeled basic_wifi_application.c and enter in the IP address and port number within that project that is labeled in the code. The wifi network of the house must be changed within the code which is precisely labeled where the address and securtiy code can be input so it can successfully connect to the house.

Then, run the debug to upload the code to the PDM. In CCS stop the code and unplug the debugger. The code is now uploaded onto the PDM and no further action is necessary programing wise. The last step is to plug the PDM power measurement into the wire after the inverter of the solar panel on top of the house. This data must be measured after the inverter on

the solar panel in order for the PDM to safely read it. This is all the setup necessary to run it. The CFS will further detail out how this data is transmitted to the server.

Notes

There is no weather protection for the PDM so the unit must be placed inside the house in a safe place with wires leading to the roof. The module consumes very little power so it will not make a significant impact on home power consumption. These units are easy to setup and have a great wifi range and capability for transmitting data.

Remote Smartphone Sensor

Android Application

Initial Installations
- Preferred Java IDE (Eclipse Juno recommended)
- Android SDK
- Samsung Galaxy S3 Driver

Setup
In SkyImager.java:
Line 340 is where the device ID is adjusted the first value is the ID, in this case 1:

String photoFileUnformatted = "1_" + date + geotext + "_" + ioioRead;

Alarm can be set in AlarmBroadcastReceiver.java:
Lines 45-46:

cal.set(Calendar.HOUR_OF_DAY,16);
cal.set(Calendar.MINUTE, 27);
cal.set(Calendar.SECOND, 0);

Time is in military time. If alarm is set, uncomment lines 96 and 97 in SkyImager.java, leave commented for immediate start of application.

alarm = new AlarmManagerBroadcastReceiver();
alarm.SetAlarm(topContext);

For testing without using the IOIO comment out lines 120 and 137 in in SkyImager.java

ioio = IOIOFactory.create();
runIOIOLoop();

Running Application

Once application is setup application can be uploaded to the Samsung Galaxy S3. The screen will load with a button on the top reading begin as well as a textbox. Clicking the textbox will allow for the input of the IP address the server that the images as wished to be sent to. Format is displayed in the initial textbox. If not IP address is input the application will still run, no attempt will be made to send the photo. After inputting the IP address, pressing the begin button will begin the application. The app then waits until the clock is either at a :00 or :30 second interval. This is to help the server with processing groups. The application will then wait an additional 30 seconds in order to establish a GPS signal and obtain a location as well as initialize a IOIO connection. Photos are saved with the format ID_HHMMSS_lat_long_ioioVoltReading, in the Pictures folder created on the phone. In order to view the photo the first line needs to be removed. It is used a security check by the server. Photos from the past application run will be deleted every time the application is ran.

IOIO/Heating and Cooling

Connect a USB cable between the IOIO and the Android Smartphone. Make sure the potentiometer is adjusted to allow maximum current. Plug the 12V supply into the 15-5V regulator. Plug the power and ground of the heating and cooling system into the 5V power and ground