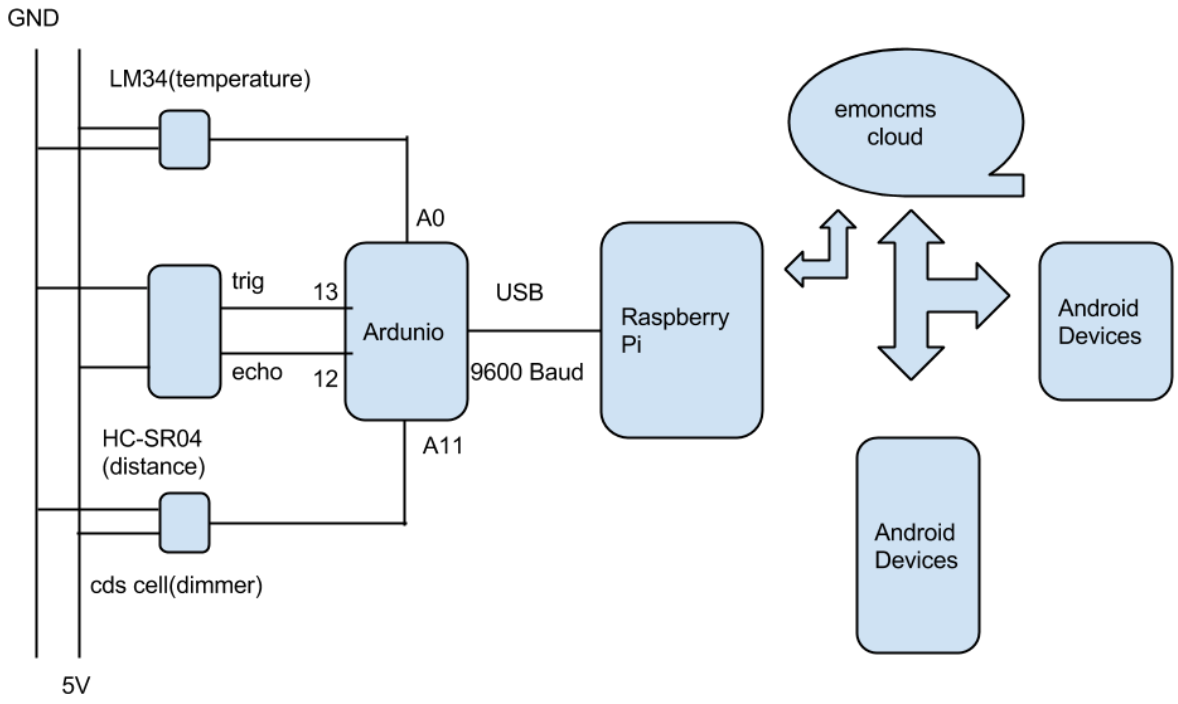
CIS 542 final project
HomeKeeper
Huayi Guo, Zhi Li
PennKey: huayi, zhili

System Overview



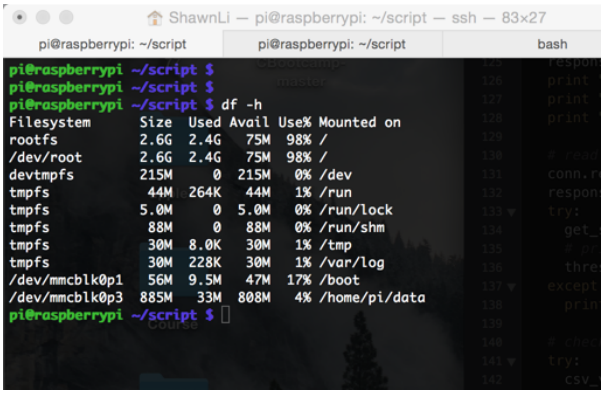
The aim is to build a remote home monitor system that could be accessed by user anywhere they want via our Android app. Hardwares to be setted up at home include a raspberry pi with wifi dongle, an Arduino DUE board, a thermometer, a dimmer and a distance sensor. A python DAQ is running on rpi to read data from Arduino via serial port, then the data is sent to emoncms cloud server and read by android devices running our app for data visualization. If the data read from Arduino exceeds the threshold set, alarm messages will be sent to users via notifications(RemoteAlert) and emails containing detail information. Also, user could change the current threshold setup on his device and send to emoncms cloud, then the settings will be read and applied by local server on rpi, thus bi-directional communication is established.

Setup Raspbian

We followed the standard routine of setting up raspberry pi system using a 16GB SD card. Considering the functionality to implement, we choose the Raspbian image instead of NOOBS and others. First we reformat SD card, then download from http://www.raspberrypi.org/downloads and flash the image to SD card.

"The Power Troll"

The biggest troll during this project was when we started to use raspberry pi. It took us so long time to figure out why the connected monitor is not working, the keyboard is not responding, and ethernet cable is not working. We changed several times of HDMI-VGA cable, switched to different screen and keyboard, but didn't work out until we finally found out that the key problem here is the **power adapter**. Then we switched to a new adapter with same voltage output of 5V but higher current output ~2A, everything started to function just right.



Setup wifi

Then, in order to ssh over to pi, we setted up the wifi dongle following the instructions in the following two links.

http://www.howtogeek.com/167425/how-to-setup-wi-fi-on-your-raspberry-pi-via-the-command-line/, http://www.raspberrypi.org/forums/viewtopic.php?f=28&t=72282

Note that the university wifi needs 802.11 authentication, which needs a little bit more tricky settings, thus we choose to use our own router at home. Now we could ssh over to raspberry pi via pi@ip.

Setup partition the disk

Specifically, we created a directory /home/pi/data that will be a mount point for the read and write data partition. Also, we set the pi to reboot in read-only mode, in order to increase the lifespan of current SD card.

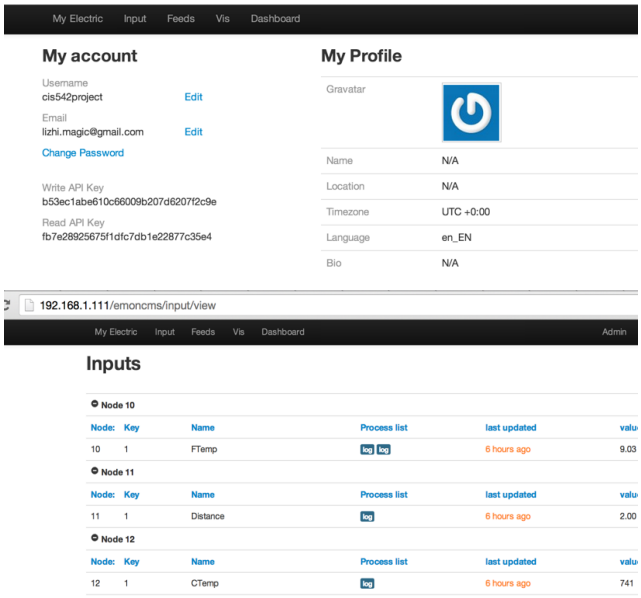
Security Concern

In order to raise the security standard of current application, we added the following packages.

1. **ufw**
   We setuped ufw to control your server access rules.
   sudo ufw allow 80/tcp
   sudo ufw allow 443/tcp
   sudo ufw allow 22/tcp
   sudo ufw enable
2. **secure mySQL**
   we choose to use mysql_secure_installation
3. **secure ssh**
   disabled root login
   sudo nano /etc/ssh/sshd_config

Setup emoncms

For this application, in case of server, we choose to use **emoncms server**, which is an open source server source provided by http://openenergymonitor.org/emon/. OpenEnergyMonitor is a project to develop open-source energy monitoring tools to help us relate to our use of energy, our energy systems and the challenge of sustainable energy. Developer could program to store sensor data in local database and post to cloud server held by emoncms.org, and the android app could fetch sensor data from the cloud server as well as set alarm threshold to the cloud server then read by the local server.



In order to build and run the server, we follow the instructions in the following link, basically we setuped the emonhub as mediator between sensor hardware and emoncms server, setted up the website using php, and created scripts to enable and disable local emoncms server. The example of local server webpage is shown on the right.
https://github.com/emoncms/emoncms/blob/bufferedwrite/docs/install.md

Set arduino on rpi

In order to listen Arduino output via serial port on raspberry pi, we compiled and loaded the following linux kernel modules (cdc-acm, usbserial, ftdi_sio), which could be found in the pi's root directory. The source codes were adapted locally from online source http://openenergymonitor.org/emon/node/488 on emoncs forum.

Setup python DAQ

First enable the local server via localemoncms-enable.

Then run the python DAQ program “read_data_internet.py” under script directory, which is designed for data logging into cloud server held by emoncms.org, so that the data could be fetched as long as the pi is connected to internet. The “read_data_local.py” is written for data logging into the local server and database.

Our python DAQ has the following functionalities.

1. **data logging**

   The most important feature above all is data logging. The python script could read data from Arduino serial port via USB by lines, then parse down data from string and send to cloud server via HTTP post method.

2. **Alert sending**

   Aim and threshold

   The sensor group of distance sensor, a photo sensor, and a thermometer is aiming to simulate the home monitor environment, where distance sensor is setted up on doors and the other two layed on desks.

   Trigger mechanism

   To avoid false alarm, we set the trigger to be three continuous readings for each sensor. We check the threshold vs current value every iteration.

   Alarm

   Once the alarm is triggered, it is sent out in three forms. The first form is locally printout on server monitor screen. The second form is to send notification to registered android device via RemoteAlert service provided by a 3rd party program, even when user is not using the app, he/she will be notified. The third form is to send email notification about which sensor has gone above threshold to the user. The latter two forms can only be triggered three times in order not to border the user too much.

Arduino Interface

The system uses Arduino DUE as interface between server and sensors. The existence of this interface makes it easier for the server to process synchronized sensor data as well as manipulating the sensors.

The sensors of choice includes:

LM34 temperature sensor, HC-SR04 Ultrasonic distance sensor and cds photoresistor for lightness sensing. They are connected to Arduino analog pins (distance sensor uses 2 digital pins), powered up under 5V and all common grounded.

Arduino communicate with the Server using USB connection. The basic operations of Arduino program is to acquire the three sensors’ data and sends the data through the serial USB port to the server in the following format:

temperature: XX.XX celcuis
light: XX.XX lux
distance: X.XX cm

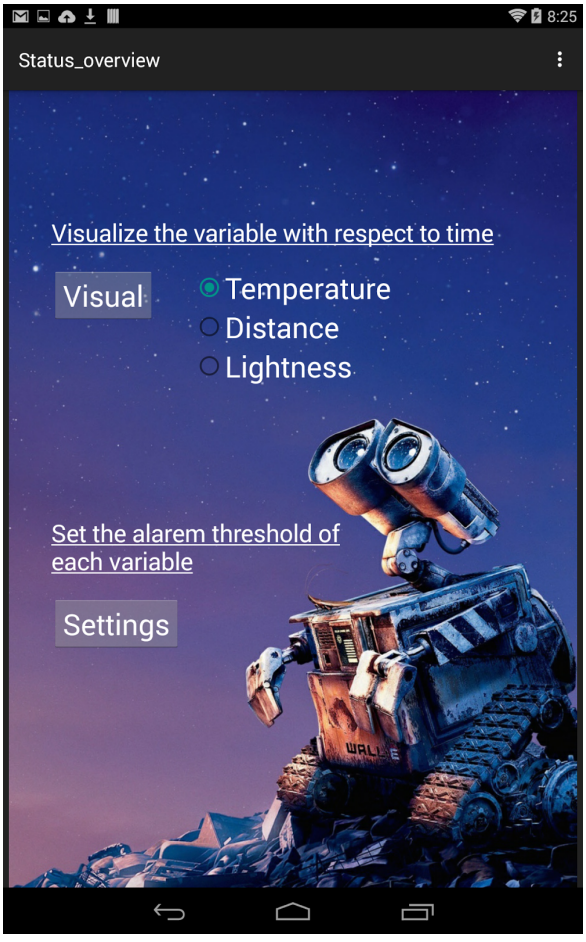
It repeats the operation every 5 seconds and it's scalable to more sensors as well.

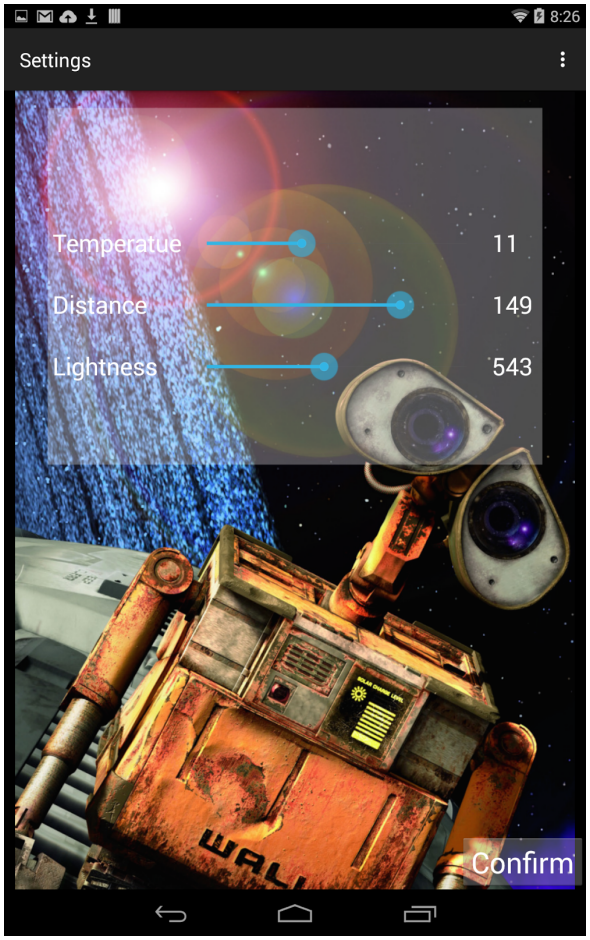
Android:
The Android application in the system basically serves as the user interface. The functionality of the Android app includes: monitoring the temperature/lightness/distance variables at home via the Emoncms Cloud, Visualization of the factors with respect to a period of time and capable of communicating with the server in order to modify the alarm conditions.

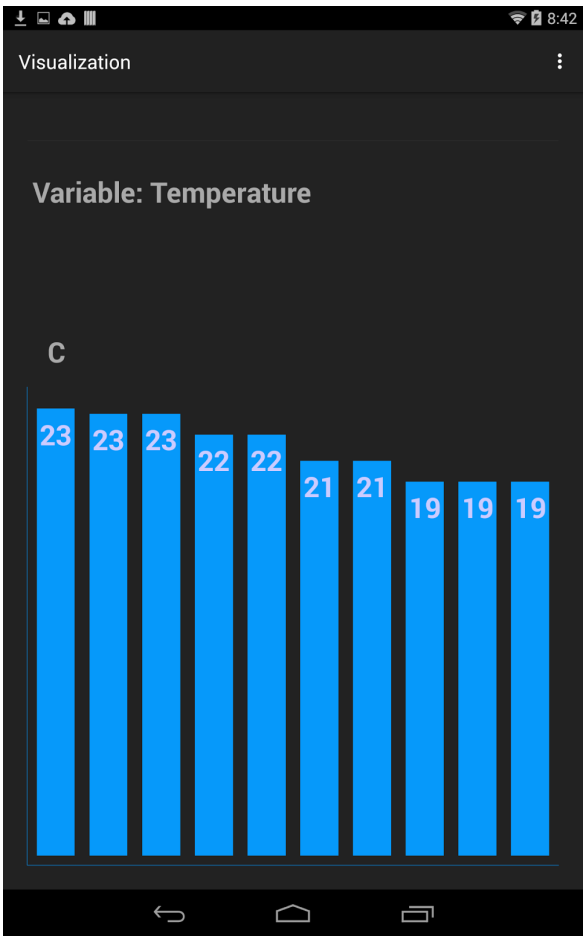
Read and write value from data nodes on server
The Android program utilized the HttpUrlconnection class to conduct http get and http post methods. Also it has a pair of apikeys associated with the same account on the server side **(emoncms cloud server)** to send requests and receive response.

Visualization
The visualization functionality is achieved following the instructions in the link below.
https://github.com/openenergymonitor/documentation/blob/master/BuildingBlocks/AndroidApp/AndroidAppPart2.md
The method provided on the web is basically refresh the contentview of the current activity and replace it with a manually painted canvas. The Android application of the system maintains a special thread that retrieves a variable status every second and refresh the UI with the new data accordingly.

Some screenshots for the android application:







Picture on the first row and first column shows the welcoming screen. It is followed by the overview and navigation activity which allows user to either take a look at the statistics of any of the given variables, or jump to settings to adjust the alarm threshold, as shown in the picture on first column of second row. The last screenshot is a statistics of variable temperature within 10 seconds.