Matthew Hengeveld

090327500

Directed Research Project Proposal

My project idea is to design and implement a home automation and sensor network. This project will consist of multiple parts in combination to create a low-cost, small data wireless network specifically designed to interface with sensors and home automation devices, as well as present data and controls via a web interface. The network will be structured around an embedded Linux ARM board, and remote, microcontroller-based sensors and controls.

It is my intention to make this project open source.

**Background**

Currently, there are many products on the market that offer many features, such as global, internet based control, wireless control, support for devices such as controllable RGB lights, relay controlled electrical sockets, thermostat control, etc. Such systems are available from companies including Belkin (WeMo products), Philips (Hue products), Insteon, Skylink (SkylinkHome products), and Nest. Links are provided below:

* Belkin: <http://www.belkin.com/us/Products/c/home-automation/>
* Insteon: <http://www.insteon.com/>
* Philips Hue: <http://www2.meethue.com/en-US>
* SkylinkHome: <http://www.skylinkhome.com/usa/homecontrol_overview.html>
* Nest: https://nest.com/

All the afore mentioned systems lack adaptability – providing one specialized system (Nest and Philips Hue) - or provide little diversity in the available products (Skylink provides controllable electrical sockets, dimmers, door sensor and motion sensor). As well, these systems are usually based on the 802.11 standard (WiFi), which is expensive to implement (Insteon). In the hobbyist scene, the Arduino WiFi shield is $80 without the Arduino, the Arduino Yun is $70, the XBee, alone, is $35, and other WiFi modules for microcontrollers can be $70 or more.

The 802.11 standard was created for transfer of large amounts of data, and therefore comes with complicated protocols and the significant overhead that accompanies them. Sensors and automation controls have inherently small data requirements. Sensors, such as temperature, humidity, light and hall sensors, typically output from a single bit (open/closed) to 24 bits (8-bit representation of a RGB format colour) of data. Similarly, automation controls typically require from a single bit (on/off) to 24 bits (8-bit PWM control of a RGB light). If we assume two bytes for addressing, and one byte for error detection – for a total of about 4-6 bytes – than the 802.11 standard has a typical packet size 250 - 375 times greater than required. (given an Ethernet packet size of 1500 octets)

Bluetooth, XBees 900MHz wireless modules, and standard 315MHz and 434MHz RF solutions each have their own disadvantages that make them unsuitable for home automation: signal range, cost, complexity and interference from other signals.

**Solution**

The wireless network will be based on the Nordic nRF24L01+ 2.4GHz wireless transceiver. This hardware works on the license-free ISM (Industrial Scientific Medical) band. This is the same band that the 802.11 wireless standard operates on. The advantage to the nRF24L01+ series transceivers is that they are much lower cost than 802.11 compatible devices. The typical nRF24L01+ module costs between $2 and $10. With a sensor designed for, example, temperature, the bill of materials can be as low as $9.50 without bulk discounts.

* PIC12F18xx microcontroller: $1.75
* nRF24L01+ transceiver: $2.90 http://www.nordicsemi.com/eng/Products/2.4GHz-RF/nRF24L01P
* TMP36 temperature sensor: $1.50
* Misc. resistors & capacitors, LED: $1.00
* PCB: $2.00

Along with the cost savings of nRF24L01+ modules, they are designed to transfer data without the significant overhead of the 802.11 standard, while still able to support features such as addressing, relaying, auto-acknowledgement, automatic resending of dropped packets, automatic error detection, etc. The maximum packet size supported by the nRF24L01+ is 32 bytes.

**Network Topology**

The network will have a tree topology. The root will be based on an embedded Linux ARM board. The root will provide the “intelligence” in the system, as well as the gateway to the internet. External nodes are the sensors and automation control devices. Internal nodes will be relay devices.

**Programming API**

The API will be designed to work with multiple different architectures programmed with a common API for each. This API will support a small set of primitive data types. Each node will have custom firmware to interface with sensors and control hardware. The API will be used to adhere to a transmission protocol.

Architectures that I plan to implement for this project are:

* Arduino Uno
* PIC (limited to a few 8-bit models)
* ARM (limited to either the Raspberry Pi or BeagleBone Black)

**Root Node**

The root node running Linux will administer all sensors and controls in the network. Using databases, the root will be able to record sensor data, as well as provide on-demand statistical data. Also, the time-shared scheduling of Linux will allow ‘simultaneous’ collection of data, statistical computations, and the ability to host a webserver. Possible hardware includes the Raspberry Pi and BeagleBone Black. The ability of the Linux board to host a web server allows direct control of the network and access to data. This close link allows a dynamic, on-demand control of the network through the web interface. The use of mobile-friendly webpage design will allow support of all modern mobile platforms.

**Open Source**

This project will be designed to be completely open source. This will allow third parties to implement custom sensors and control devices, as well as web interfaces for them.

|  |  |
| --- | --- |
| **Programming Languages**   * Microcontrollers   + C   + PIC Assembly   + Processing * Linux   + C   + Python   + MySQL * Web   + Python   + HTML   + CSS   + PHP   + MySQL   + JavaScript   + JQuery   + AJAX | **Hardware Required** (Root node and three external nodes)   * Root Node   + Raspberry Pi   + Suitable power supply   + SD card (>4GB)   + nRF24L01+ module * External Nodes   + (2) Arduino microcontrollers   + (1) PIC 8-bit microcontroller   + Humidity and temperature sensor   + 5V relay   + RGB LED   + Various BJT transistors   + Various MOSFETS   + Various resistors   + Various capacitors   + (3) nRF24L01+ modules * Misc.   + Wireless router   + Ethernet cable |

**Sample Flow for Humidity/Temperature Sensor Update**

* Client web browser sends AJAX request to root node
* Root node requests update from humidity/temperature sensor
  + Root looks up information about sensor based on unique address
    - Device type (sensor/control)
    - Number of sensors on device
    - Data type & size of each value returned
  + Root sends “read sensor” 8-bit command to sensor using unique 16-bit address
  + Microcontroller reads analog value from ADC (8-bit unsigned integer each)
  + Sensor sends two 8-bit values to root
* Sensor value record updates and statistical data updates
  + Root receives sensor values
  + Database is updated with new values
  + Average and other statistical calculations updated
* Web interface update
  + Root node sends new values to client web browser using AJAX