SensorNet

Wireless Home Automation & Sensor Network

Part 1

Dec. 8th, 2014
Matthew Hengeveld
090327500
PC491 Directed Research
Dr. Znotinas

Nomenclature

- SensorNet – the system, including the root, nodes, wireless transceivers, and any other circuitry involved with the operation of the system.
- Root – the central device in a star network, directly connected to every node.
- Node – the outer device in a star network, all connected directly to the root.
- Automation control – a device that is able to manipulate, or command, another device automatically
- Sensor – a device that converts different physical properties into an electrical representation
- On-air time – the amount of time it takes for a wireless transmitter to start at a static state, transmit the packet, and return to a static state.

Abstract

SensorNet is a wireless automation and sensor network based on inexpensive hardware, and using a simple, multi-architecture network protocol. Structured in a star network, the root provides system intelligence, a gateway to the internet, and a central device through which all nodes connect wirelessly. Each node is host to automation controls and /or sensors. The SensorNet system is optimized for low latency transfer of small data.

The root is responsible for many functions, including requesting updates from nodes, providing the gateway to the internet, hosting the SensorNet interface on a web server, and maintaining connections between itself and nodes. SensorNet uses a low-cost Raspberry Pi as a root.

Nodes relay sensor data, and trigger automation controls on-demand, and by request of the root. The ability of nodes to contact the root without a request of the root is limited. Nodes may also have functions that operate independent of the SensorNet system.

Programming interfaces are available for many popular processors and controllers such as the PIC, Arduino and ARM.

Wireless communication is implemented using the Nordic nRF24L01+ transceiver.

Features of SensorNet include:

- Global, internet-based control and administration.
- Wireless nodes
- Open source code and schematics
- Support for many different sensors and automation controls.

Background

Currently, there are many systems on the market that offer similar features as SensorNet provides. Companies such as Belkin, D-Link, Insteon, Nest, Philips, and Skylink each have product lines which include some, but not all, of these features. Products like the Nest Learning Thermostat and the Philips Hue are both specialized systems, and are limited to one function. Broader product lines are offered by companies such as Belkin, Insteon and SkylinkHome. These product lines typically offer only controllable mains voltage plugs, light dimmers and thermostats, as well as a

few other specialized products. As well, most of these products rely on each device having its own WiFi connection to the home network. This causes issues with access points not able to handle a high number of connections, excess network traffic, and wireless interference for the whole wireless network.

The 802.11 standard (WiFi) was designed for computers to make high bandwidth transfers to and from other computers and network infrastructure. WiFi is therefore expensive to implement, and adds significant cost to devices such as lightbulbs. Integrating WiFi with microcontrollers is expensive. Examples include:

- Arduino WiFi shield - $80
- Arduino Yun - $70
- XBee - $35
- Intel Galileo - $75

WiFi 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 (RGB format colour) of data. Similarly, automation controls typically require from a single bit (on/off) to 24 bits (PWM control of a RGB light). Assuming 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 for typical sensor and automation control. (given an Ethernet packet size of 1500 octets)

Bluetooth, and other RF transceivers each have their own disadvantages that make them unsuitable for home automation and sensors: signal range, lack of networking capabilities, complexity and lack of noise rejection.

SensorNet

SensorNet is a wireless network for the communication of automation and sensor data. SensorNet has two kinds of nodes – sensors that relay data to the root, and automation controls that act on data relayed to it by the root. Sensor and automation controls are controlled through a web interface hosted on the root. Software on the root manages the interface between sensors, automation controls and the human-controlled web interface. This software is also responsible for the major timing based operations as well as interaction between nodes.

SensorNet is based on the Nordic nRF24L01+ 2.4GHz wireless transceiver. This is an inexpensive receiver capable of 2 Mbps, and transcieving on a channel width of 1 Mhz. This allows the nRF24L01+ to select one of 127 different frequencies which has the least interference. The ISM band that the nRF24L01+ operates on is the same as that of WiFi. However, WiFi has 20 or 40 MHz channel widths, so to avoid interference with other WiFi networks, channels 1, 6 and 11 are typically the most used. This allows the nRF24L01+ to use the spacing between

to avoid interference with WiFi. The overhead of wireless transfer is also much less than that of WiFi, around 6 bytes.

SensorNet has a specific packet format. The nRF24L01+ has many user selectable features, and some are implemented in SensorNet. Each feature requires space in the packet:

Feature	Bytes
Address length	4
CRC	1
Dynamic payload	1

The rest of the packet is made up of the preamble byte, packet control field byte and the payload.

Other features are enabled, but do not require packet space. These include auto acknowledgement, 1000us retransmit delay, 5 retransmit limit, and multiple data pipes. Acknowledgement payload is not used.

The wireless protocol for SensorNet is defined by the setting of the nRF24L01+. In order for wireless communications to work reliably, every node must have the same settings as the root [1]. Figure 1 shows initialization settings for every nRF24L01+ register.

The SensorNet network is based on a star topology. The root has direct communications with, and control of, all nodes. The root is responsible for initiating communications with nodes, typically one at a time. Nodes are not designed to communicate with each other. These two rules reduce interference within the network. Communications are kept as small as possible to reduce on-air time. The typical on-air time is less than 1ms.

REGISTER	Description	Bits
CONFIG	Enable all interrupts, 1 byte CRC	B00101011
EN_AA	Enable auto acknowledgement	B00000011
EN_RXADDR	Enable data pipes 0 and 1	B00000011
SETUP_AW	Set address lengths to 4 bytes	B00000010
SETUP_RETR	Enable 1000µs retransmit delay, 10 attemps	B00110000
RF_CH	Channel 105 (2.4GHz + 0.105GHz = 2.505GHz)	B01101001
RF_SETUP	Set RF data rate to 1Mbps, 0dBm output power	B00000110
DYNPD	Set dynamic payload for pipe 0	B00000011
FEATURE	Enable dynamic payload	B00000100

FIGURE 1: NRF24L01+ SETTINGS REGISTERS

[1] Not including addresses and TX/RX modes

The SensorNet network is based on a star topology. The root has direct communications with, and control of, all nodes. The root is responsible for initiating communications with nodes, typically one at a time. Nodes are not designed to communicate with each other. These two rules reduce interference within the network. Communications are kept as small as possible to reduce on-air time. The typical on-air time is less than 1ms.

The root node of SensorNet is a Raspberry Pi, with a minimal, command-line version of Debian linux installed. Only relevant packages were installed. The major functions of the root is to administer all nodes, as well as host a database and webserver, execute computationally heavy workloads for nodes, and to provide time sensitive cues for nodes.

To maximize the adaptability of the nodes, different architectures have libraries that allow them to become a fully functional node. These include PIC, Arduino, and ARM (running linux).

Human control of SensorNet will be through the web interface. This interface will include

Problems

Several problems arose during development of the libraries.

The SPI interface for the nRF24L01+ has specific requirements. First, all consecutive data must be sent while the CSN pin is held low. If the CSN pin line is pushed high during the data sequence, then the nRF24L01+ will assume it is the end of the data. For example, to transmit data, the CSN pin must first be held low. A command is then sent to let the nRF2L01+ know there is data coming. The data is then sent. After the last byte is sent, then the CSN can be changed to high once more.

One of the features of the nRF24L01+ is the auto acknowledgement. When a packet is sent and receiver successfully, the receiver quickly changes to TX mode, while the transmitter quickly changes to RX mode. If the transmitter does not receive an acknowledgement, then the packet will be resent. In implementing this feature, it was not clear that the auto acknowledge feature used a different data pipe to send the acknowledgement packet. While the acknowledgement packet was being sent, it was being sent to the wrong address.

In testing the libraries, it was found that packets having payload sizes from 30 to the full 32 byte payload were not being received. This problem has not been resolved, therefore, SensorNet is limited to payloads of 28 bytes or less.

PIC microcontrollers can be complex. During the initial stages of programming the PIC to talk to the nRF24L01+, it was unknown that the PIC has options for changing the SPI sample position (start or middle of clock signal) and for changing the SPI clock polarity (low or high). These were, by default, not set up right for communication with the

nRF24L01+. The following settings were needed to resolve the problem: [2]

Register.bit	Setting
Clock Polarity: SSP1CON1bits.CKP	0
Clock Edge Detect: SSP1STATbits.CKE	1
Sample Bit: SSP1STATbits.SMP	1

The root node for SensorNet is the Raspberry Pi. It was first released in 2012, and has since become very popular. Despite its popularity and community support, there are still challenges when it comes to the hardware aspect of programming. One difficulty with the Raspberry Pi was finding a suitable programming interface for the GPIO and SPI hardware. Several different libraries were tested. The majority of the problems arose from the libraries needing root (account with all rights and privledges) access to execute. This is a security risk, and could compromise a home network. Another problem was finding a library that supported both GPIO and SPI functions. Lastly, because of the Linux operating system and its process scheduler. This prevented time-critical code from executing when it was intended to. The most severely affected function was interrupts, as it could take tens of milliseconds to register and execute the interrupt service routine. A solution was found that resolved all of these problems: a C library called bcm2835 [3]. Bcm2835 supports GPIO and SPI function, as well as being run without needing root access. Also, because it is written and compiled in C, it runs significantly faster than other tested Python libraries. In an extreme case, bcm2835 executed a GPIO pin change in less than 400µs, whereas the Python library took more than 200ms. This large difference is because C is a compiled language, and Python is an interpreted language. Interrupt functions are not supported in bcm2835, however, because of the speed of C code, polling is used to detect pin changes. For the purposes of SensorNet, this means that an estimated 1500-2000 packets can be sent a second [4].

Status

The status of the SensorNet project is ongoing. Libraries for different architectures have been tested with direct, one-one communications, but not in a network setting. As well, syncing functions have not yet been tested, as this requires the root control software. All nRF24L01+ registers have been tested to be readable and writable by all libraries. Several different circuits have been tested, with different architectures: Arduino Uno R3, PIC18F24K22, Raspberry Pi Model B, Rev. 2. Other Arduino and PIC models should only require changes of pin definitions.

[2] For PIC18F2xK22 series.

[3] www.airspayce.com/mikem/bcm2835/

[4] Using 1 Mbps data rate, assuming a packet size of 29 - 32 bytes, and a time of 500-800µs per packet

Schematics for the three working test circuits are included with this document. Test code is also included with this document, however, some libraries are missing proper getter and setter functions, as well as code documentation.

The next steps in the project (in order of completion), and estimated time for completion, are:

- Finish code and documentation (1 week)
- Templates for nodes (1 week)
- Setup of SQL database and web server on (1 week)
- SensorNet software on Raspberry Pi (2 – 3 weeks)
- Building of sample nodes (1 – 2 weeks)
    - RGB light
    - Temperature and humidity sensor
    - Controllable mains voltage light [5]
- Project documentation (1 week)
- Project demonstration
- Poster (1 week)

[5] Upon approval

References

http://ww1.microchip.com/downloads/en/DeviceDoc/41412F.pdf
https://www.nordicsemi.com/kor/content/download/2726/34069/file/nRF24L01P_Product_Specification_1_0.pdf