Capstone project logbook

Open source home energy reduction system.

Sponsor: Rescommunis

Owner: Hai Nguyen

Email: [hoanghai19487@gmail.com](mailto:hoanghai19487@gmail.com)

Phone: 5034717087

Date: 11/14/2010

### Date: 11/15/2010

Email to team member:

* Hieu Nguyen:
  + Phone: 503 471 7340
  + Email: [hieubk2006@yahoo.com](mailto:hieubk2006@yahoo.com)
* Nam Nguyen
  + Phone:
  + Email: [duynam1214@yahoo.com](mailto:duynam1214@yahoo.com)
* Micheal Stoltz:
  + Phone: 503 – 928 -9238
  + Email: [mstolt25@comcast.net](mailto:mstolt25@comcast.net)
* MIcheal White
  + Email: [White\_M@comcast.net](mailto:White_M@comcast.net); [michael.whiteme@gmail.com](mailto:michael.whiteme@gmail.com)

Decide team will have the first meeting at Fri, 11/19/12:30 with Dr Hall

### Date: 11/19/2010

* Meeting with Dr Hall. First look through the project
* Scope: An low cost, re-productable, mesh of node
  + Sensor
  + Server
  + Physical output
  + 🡪 Reduce energy use in home
* Node system builds on Jeenode
* Sensor: ambient light, motion, temperature, humidity, noise…
* Information: power usage monitoring, water-level monitoring, Water usage monitoring
* Physical output:
  + Shade actuation
  + Composting toilet maintenance
  + Dim the light
  + Shut off the outlet
* Document for reproducibility or Hobbyist
* Server software
* **White is the team facilitator**

### Date 11/28/2010

Team meeting: Decide which wiki will be used

* TRAC
* Wikispaces.com

Ask more question to clear Requirement

### Date 12/03/2010

Meeting with sponsor at site

Address: 2222 SE Woodward ST, Portland, OR, 97202

Sponsor name: Garret Moon

Phone: 503 797 7555

The house is building. Everything is just finished the basic.

Talk about project: The big project is building a house with the method to save energy as much as possible. The house is built by himself, his family and his friend. They designed, developed, built,…

They also have the webpage: rescomunis.org

They want an open project and public everything.

### Date 12/14/2010

Team meeting:

* Try to understand the requirement from sponsor
  + Water cistern
  + Sensor
    - Light
    - Temperature
    - Humidity
    - Water flow
    - Water level
  + Network:
    - Mesh? Tree?
    - RF 🡪 Jeenode???
* Hai ‘assignment for research: water level sensor

### Date 12/15/2010

Research on water level sensor

Method to measure the water level:

* Measure the pressuare 🡪 Water level
* Make the check point
* Measure the distance from the water surface,

Measure the pressure:

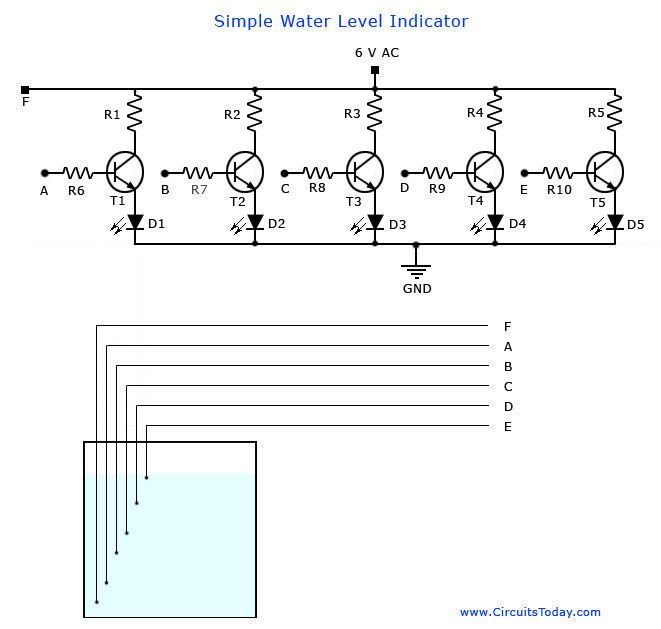
<http://www.globalw.com/products/levelsensor.html>

WL400 WATER LEVEL SENSOR Submersible Pressure Transducer for Level & Pressure

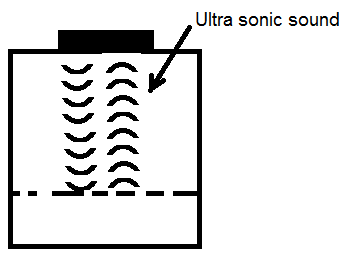
* Highly cost

Make the check point:

<http://www.circuitstoday.com/simple-water-level-idicator>



* Simple thought, cannot use to measure the exact volume of water

Measure the distance from water surface: using ultrasonic sensor

The speed of sound is 340 m/s or 29 microseconds per centimeter. The ping travels out and back, so to find the distance of the object we take half of the distance travelled.

* Easy to implement, can calculate the exact volume of water.

Conclusion: Choosing the method: Measure the distance from the water surface.

Have some research about communication

Xbee

Monnit

Jeenodes

### Date 12/21/2010

Team meeting:

* Report in research
* Project time line
* Buget
* Control in house
* PDS draft
* Proposal

Research more in water level sensor, decide to use the ultrasonic to measure the distance to the water surface:

* Easy connect to AVR + arduino
* Cheap
* Accurately

Communication:

Stick on Jeenode or Monnit:

* Base on in the arduino 🡪 a lot of library
* Easy to connect with many sensor

### Date 12/29/2010

Team meeting:

Floor plan

Communication

* Monnit: <http://www.monnit.com/>
  + $239: 1 reciever + 6 modul no sensor
* Jeenode: <http://shop.moderndevice.com/products/jeenode-kit>
  + Avr + ARDUINO
  + Support a lot sensor

PDS

Sponsor:

* Water level sensor??? Cistern
* Where to put sensor
* Dimension
* How deep of tanks.

### Date 01/11/2011

Teameeting

Working on PDS

Hai assignment: Work on Jeenode and communication protoype1

## Working on Protoype 1 – due 01/15/2011

### USB - Serial communication between [JeeLink](https://projects.cecs.pdx.edu:8443/%7Ewhitem/OSHERS/index.cgi/wiki/JeeLink) and computer:

* The computer send the message and the [JeeLink](https://projects.cecs.pdx.edu:8443/%7Ewhitem/OSHERS/index.cgi/wiki/JeeLink) responses
* The [JeeLink](https://projects.cecs.pdx.edu:8443/%7Ewhitem/OSHERS/index.cgi/wiki/JeeLink) send the message and the computer responses

### Communication between [JeeLink](https://projects.cecs.pdx.edu:8443/%7Ewhitem/OSHERS/index.cgi/wiki/JeeLink) and [JeeNode](https://projects.cecs.pdx.edu:8443/%7Ewhitem/OSHERS/index.cgi/wiki/JeeNode):

* [JeeNode](https://projects.cecs.pdx.edu:8443/%7Ewhitem/OSHERS/index.cgi/wiki/JeeNode) send message and [JeeLink](https://projects.cecs.pdx.edu:8443/%7Ewhitem/OSHERS/index.cgi/wiki/JeeLink) show that message in the computer
* [JeeLink](https://projects.cecs.pdx.edu:8443/%7Ewhitem/OSHERS/index.cgi/wiki/JeeLink) send message to the [JeeNode](https://projects.cecs.pdx.edu:8443/%7Ewhitem/OSHERS/index.cgi/wiki/JeeNode) and the [JeeNode](https://projects.cecs.pdx.edu:8443/%7Ewhitem/OSHERS/index.cgi/wiki/JeeNode) responses, show [JeeNode](https://projects.cecs.pdx.edu:8443/%7Ewhitem/OSHERS/index.cgi/wiki/JeeNode) response in the computer

### Date: 01/12/2011

Jeelink is base on Arduino.

Library in serial: <http://arduino.cc/en/Reference/Serial>

Jeelink: <http://shop.moderndevice.com/products/jeelink-module-fully-assembled>

* USB interface
* Has an AVR + arduino bootloader
* Has RF module

#### ****Test Serial in Jeelink****

* 1. Plug Jeelink on the computer, see what port of it assigned
  2. Use Arduino IDE version 18 or later (here ):
     + http://arduino.cc/en/Main/Software
  3. Open the Arduino
  4. Choose the Tools --> Serial Port --> Port that jeelink assigned
  5. Open the Arduino. Put code this to the edit windows :

/\*

TEST SERIAL in JEELINK

This program is test the communication between computer and Jeelink

Created 11 Jan. 2011

by Hai Nguyen

Rev 1.0

Test successful

\*/

int incomingByte = 0; // for incoming serial data

**void** setup()

{

// start serial port at 9600 bps:

*Serial.*begin(9600);

establishContact(); // send a byte to establish contact until receiver responds

}

**void** loop()

{

// send data only when you receive data:

if (*Serial.*available() > 0) {

// read the incoming byte:

incomingByte = *Serial.*read();

// say what you got:

*Serial.*print("Recieved: ");

*Serial.*println(byte(incomingByte));

}

}

**void** establishContact() {

while (*Serial.*available() <= 0) {

*Serial.*println("OSHERS Project"); // send an initial string

*Serial.*println("Please send information...");

delay(1000);

}

}

<https://projects.cecs.pdx.edu:8443/~whitem/OSHERS/index.cgi/browser/Coding/Prototype%201/JeeLink/Test_serial_rev1_12JAN11_HaiNguyen/Test_serial_rev1_12JAN11_HaiNguyen.pde>

* 1. Compiling Ctrl + U & wait for uploading to jeelink
  2. Open the Serial Monitor (Ctrl + Shift + M)
  3. See the introduction in the Serial Monitor:

1. OSHER Project
2. please send information
   1. Send information using Serial Monitor
   2. See the response

* successful

**Server: Jeelink + Computer**

### Date 01/14/2011

Work

RF12: Demo <http://jeelabs.org/view/libraries/RF12/>

Jeenode: <http://shop.moderndevice.com/products/jeenode-kit>

Connect with computer by the FTDI cable.

#### Communication between [JeeLink](https://projects.cecs.pdx.edu:8443/%7Ewhitem/OSHERS/index.cgi/wiki/JeeLink) and [JeeNode](https://projects.cecs.pdx.edu:8443/%7Ewhitem/OSHERS/index.cgi/wiki/JeeNode)

The jeelink and jeenode communicate to other by RF signal.

Connect Jeenode to computer through the FTDI cable (and USB port).

Connect Jeelink to computer via USB.

### I. Install RF in Jeelink

1. Plug Jeelink on the computer, see what port of it assigned
2. **Use Arduino IDE version 18 or later (http://arduino.cc/en/Main/Software), and put the "Ports" (**[**http://jeelabs.net/projects/cafe/wiki/Ports**](http://jeelabs.net/projects/cafe/wiki/Ports)**)() and "RF12" (**[**http://jeelabs.net/projects/cafe/wiki/RF12**](http://jeelabs.net/projects/cafe/wiki/RF12)**) folder inside a "libraries" folder (Inside the Arduino folder) next to your sketches.**
3. Open the Arduino
4. Choose the Tools --> Serial Port --> Port that jeelink assign
5. Open the Arduino -> Example -> RF --> RFdemo
6. Compiling Ctrl + U & wait for uploading to jeelink
7. Open the Serial Monitor (Ctrl + Shift + M)
8. Change the serial baud rate (Right + Down Corner to 57600 baud
9. See the introduction in the Serial Monitor:

[RF12demo.6] R i18\* g212 @ 915 MHz

Available commands:

<nn> i - set node ID (standard node ids are 1..26)

(or enter an uppercase 'A'..'Z' to set id)

<n> b - set MHz band (4 = 433, 8 = 868, 9 = 915)

<nnn> g - set network group (RFM12 only allows 212, 0 = any)

<n> c - set collect mode (advanced, normally 0)

t - broadcast max-size test packet, with ack

...,<nn> a - send data packet to node <nn>, with ack

...,<nn> s - send data packet to node <nn>, no ack

<n> l - turn activity LED on PB1 on or off

<n> q - set quiet mode (1 = don't report bad packets)

Remote control commands:

<hchi>,<hclo>,<addr>,<cmd> f - FS20 command (868 MHz)

<addr>,<dev>,<on> k - KAKU command (433 MHz)

Current configuration:

R i18\* g212 @ 915 MHz

1. Open the another Arduino
2. Choose the Tools --> Serial Port --> Port that jeenode assign
3. Open the Arduino -> Example -> RF --> RFdemo
4. Compiling Ctrl + U & wait for uploading to jeelink
5. Open the Serial Monitor (Ctrl + Shift + M)
6. Change the serial baud rate (Right + Down Corner to 57600 baud
7. See the introduction in the Serial Monitor:

[RF12demo.6] R i18\* g212 @ 915 MHz

Available commands:

<nn> i - set node ID (standard node ids are 1..26)

(or enter an uppercase 'A'..'Z' to set id)

<n> b - set MHz band (4 = 433, 8 = 868, 9 = 915)

<nnn> g - set network group (RFM12 only allows 212, 0 = any)

<n> c - set collect mode (advanced, normally 0)

t - broadcast max-size test packet, with ack

...,<nn> a - send data packet to node <nn>, with ack

...,<nn> s - send data packet to node <nn>, no ack

<n> l - turn activity LED on PB1 on or off

<n> q - set quiet mode (1 = don't report bad packets)

Remote control commands:

<hchi>,<hclo>,<addr>,<cmd> f - FS20 command (868 MHz)

<addr>,<dev>,<on> k - KAKU command (433 MHz)

Current configuration:

R i18\* g212 @ 915 MHz

### Follow: HOW TO SET UP A SMALL WIRELESS TEST

*RF12DEMO ========*

*The RF12demo sketch can be useful to test proper operation of RFM12(B) modules. It lets you set up two nodes and send test packets between them via commands entered on the serial USB connection. The basic settings are stored in EEPROM so that you can configure a board and then use it at the remote end without connecting it to a second computer.*

*HOW TO SET UP A SMALL WIRELESS TEST ===================================*

*1. First of all, get two units hooked up as described here:*

[*http://news.jeelabs.org/2009/02/10/rfm12b-library-for-arduino/*](http://news.jeelabs.org/2009/02/10/rfm12b-library-for-arduino/)

*I use "*[*JeeNode*](https://projects.cecs.pdx.edu:8443/%7Ewhitem/OSHERS/index.cgi/wiki/JeeNode)*" for this, but that's just me:*

[*http://news.jeelabs.org/2009/03/05/jeenode-v2-pcb/*](http://news.jeelabs.org/2009/03/05/jeenode-v2-pcb/)

*2. Compile and upload this RF12demo sketch to both boards. The serial baud*

*rate is set at 57600, you can change it in the source code if needed. You will need to install the "RF12" Arduino library module - eg. on Mac OS X I copied the "RF12/" directory to /Applications/arduino/hardware/libraries/.*

*3. When run, the demo gets config settings from EEPROM to set up the RFM12(B).*

**Begin from here:**

If this is the first time, it'll probably report "A i1 g212 @ 433 MHz".

4. Change the node ID as follows: enter "<N>i" with <N> a number from 1 to 26.

For a first test, you could just pick node ID's 1 and 2 for the two units.

5. Change the frequency band as follows: enter "<N>b" with <N> one of: 4, 8, 9.

These are the 433 MHz, 868 MHz, and 915 MHz frequency bands, respectively. You should pick the same band as your RFM12B modules, preferably.

6. Set the network group (aka "house code") with the command "212g". Groups

1..250 are available for the RFM12B, the RFM12 module \*only\* works with 212. Make sure you set all units to the same group or they won't see each other.

7. You're all set for the first unit, disconnect and repeat for the other one.

8. Ok, time to try sending a packet. First make sure both units are powered up.

It helps to have both units connected to the same machine so you can set up terminal windows for both of them, but if you only have one unit connected to the terminal window and the other just powered up that's ok too.

9. Enter "0s" on one unit. The other unit should report "OK" plus a number.

If you can't see the output of the other unit, use "a" instead of "s". This will send a packet and also request and acknowledgement. If all is well you'll see a short "OK" response on the same unit as where you typed "a".

10. In the above test, you're sending packets with no actual data other than

what the protocol itself requires to function properly. To include some test data use "1,2,3,0s" or "1,2,3,0a" with "1,2,3" being the actual data. This sends a test packet with 3 data bytes (you can send up to 66 data bytes). The "t" commands is available as shorthand for "0,1,2,...,63,64,65,0a".

11. To do some range tests, carry the second unit to a remote location and use

"0a" or "t" to try and send some data across and get an acknowledgement. At some point you'll see packets drop out (which could either be the data or the ack return). Longer packets tend to drop out more quickly, because there is a larger chance of them getting disturbed by noise along the way.

12. You may see "?" replies instead of "OK". These are packets with an invalid

checksum. These are either random noise or incorrectly received packets. If you leave the demo running long enough, you'll probably also get such "?" messages occasionally when the radio picks up some random noise.

13. Note that you \*can\* configure an RFM12(B) module to send/receive on another

frequency band than the one it was designed for. You'll just get a lower range because the RF components will not be optimally tuned for these cases.

14. The code is fully open source - feel free to browse and make changes as you

see fit. The latest version of this software is always available here:

<http://jeelabs.org/view/libraries/RF12/>

--> Successful

## Working on Protoype 2 – due 01/21/2011

### Date 01/26/2011

* send the information from Jeenode to computer through Jeelink.
  + Specific massage
  + Specific node
  + Specific code using RF12 Library

Work with Hieu.

<http://jeelabs.net/projects/cafe/wiki/RF12>

<http://jeelabs.net/projects/cafe/wiki/Rf12_sendStart%28%29>

When you put : rf12\_sendStart(0, sendData , sizeof sendData), the node will broadcast the data to all other nodes without ACK.

To send data to a specific node, we must modify the header of rf12\_sendStart a little bit:

RF12\_HDR\_ACK | RF12\_HDR\_DST | dest\_nodeid: sends only to exact node,with Ack

RF12\_HDR\_ACK :broadcasts to all nodes and requests Ack

RF12\_HDR\_ACK | dest\_nodeid: broadcasts to all nodes and requests Ack

RF12\_HDR\_DST | dest\_nodeid :sends only to exact node, no Ack

For example: rf12\_sendStart(RF12\_HDR\_DST | 1, sendData , sizeof sendData) will send the data in sendData to node 1 without ACK.

### Date 02/01/2011

Team meeting:

* Time line
* PDS
* Split work.

Jan-March: Basic completed sensor node

March-June: Me design for note

Spilt work:

Sensor: (White, Stoltz, Nam, Nguyen)

Communication: Hai. Hieu.

## Work on communication with Hieu.

### Date 02/08/2011

1. What the Jeenode platform have?

- They can send directly from jeenode 🡨🡪 Jeenode, jeeLink 🡨🡪 Jeenode.

2. Requirement:

- There are a lot of nodes in the house; each node is connected with the sensor.

- The node will send information of sensor to server; server will process, then response to the control node.

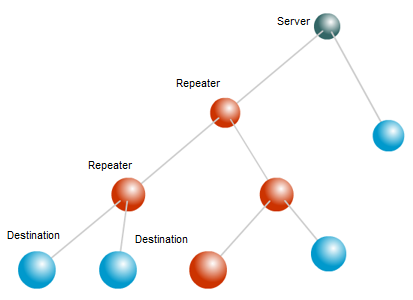
- Node and node don’t send information to each other without server.

3. Choosing the topology:

- The house is big.

- The distance which is available to send between nodes is limited.

🡪 **Decide using Tree topology.**



Each node can be Repeater or Destination

Server is the main root.

Repeater sends the information to another Repeater or Destination depends on map.

Destination will process the information.

4. **Information** in network.

* Server Request Data to the specific node.
* Node sends Data to Server.
* Server send map (route) to node.
* ACK of server & node.
* Server:
  + Send REQUEST Data to node
  + Send ACKNOWLEDGEMENT of receiving data
  + Send DATA to node (for control)
  + Send MAP to node
  + Receive DATA from node
  + Receive ACKNOWLEDGEMENT from node
* Node:
  + Receive REQUEST
  + Receive DATA
  + Send DATA
  + Receive MAP
  + Send DATA
* Jeenode can send packet with 66 bytes. So that using 66 bytes of packet of jeenode using library RF12. It will be the container for our frame.

The data is sent from server (ID:1) to Destination node(ID:3). node ID 2 is repeater.

Repeater:

* When data received data, check:
  + If Destination: process depend on data
  + If not Destination: node is Repeater.

Map: 2 nodes: 2 & 3

|  |  |  |
| --- | --- | --- |
| Route | Type of Message(Action) | Data |

Route: The route information so that data can be delievered.

|  |  |  |  |
| --- | --- | --- | --- |
| Byte0 | Byte1 | … | byteN |

Byte0: number of node in the route.

Byte1..byteN: Node ID in the route.

When the node received data, check:

* Byte0 =1 , node is Destination
* Byte0 <>1: node is repeater. If node is repeater, decrease byte0, delete current node ID

|  |  |
| --- | --- |
| 1 | 3 |

|  |  |  |
| --- | --- | --- |
| 2 | 2 | 3 |

* Type of packet: 1 byte
  + ‘R’: request
  + ‘A’: acknowledgment
  + ‘D’: Data
  + ‘M’: maps
* Data of ‘D’:

|  |  |
| --- | --- |
| Type of sensor | Data of sensor |

* + Type of sensor
    - ‘H’ : humidity
    - ‘L’: Light
    - ‘O’: occupation
    - ‘W’: water ;eve;
  + Data of sensor: 🡪 ??? wait information from sensor team
* Data of ‘M’:

|  |  |  |  |
| --- | --- | --- | --- |
| Byte0 | Byte1 | … | byteN |

* + Map information to the server from this node. Map information is sent by server

Example:

1. Server send maps:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Route | | | Type of message(Action) | Data | | |
| 2 | 2 | 3 | ‘M’ | 2 | 2 | 1 |

1. Server request humidity

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Route | | | Type of message(Action) | Data |
| 2 | 2 | 3 | ‘R’ | ‘H’ |

1. Node 3 send data

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Route | | | Type of message(Action) | Data | | |
| 2 | 2 | 1 | ‘D’ | 1 | 3 | 1 |

**How about ACK?**

### Date 02/10/2011

Team meeting

* Format of proposal
  + Overview: Hai + Hieu
  + Communication protocol: Hai + Hieu

### Date 02/17/2011

Coding for communication:

Purpose: writing the code for Server: node 1, Repeater: node 2, and Destination: node 3.

1. Server - Jeelink code:
   1. This code use the RF12 and Ports library
   2. The code will send the the Map to node 3.
2. Code:
   1. Setup()
      1. Set up the serial, to send information on screen of Serial Monitor ( of Arduino)
      2. Initalize the RF12: NodeID =1, 915Mhz, group ID 212
   2. Loop()
      1. SendData ={0x02,0x02,0x03,’M’,0x02,0x02,0x01,0xFF}
      2. rf12\_recvDone() 🡪 The recvDone() function should be called often, to keep receptions and transmissions going. The driver is interrupt driven for all low-level byte-by-byte operation, but needs to be polled to go through the different stages needed to fully process a received packet and to resume reception after sending out a packet.( <http://jeelabs.net/projects/11/wiki/Rf12_recvDone%28%29>)
      3. if rf12\_canSend() then rf12\_sendStart(RF12\_HDR\_DST | SendData[1], SendData , sizeof SendData) 🡪 send the SendData{} to node 2.

#include <RF12.h>

#include <Ports.h>

byte SendData[60];

//int length=1; //default length =1, just 1 node from central node

void setup ()

{

Serial.begin(9600);

rf12\_initialize(1, RF12\_915MHZ, 212);

}

void loop ()

{

SendData[0]=0x02;

SendData[1]=0x02;

SendData[2]=0x03;

SendData[3]=byte('M');

SendData[4]=0x02;

SendData[5]=0x02;

SendData[6]=0x01;

SendData[7]=0xFF; //end of message

//Serial.print(char(SendData[3]));

rf12\_recvDone();

if (rf12\_canSend())

{

Serial.print(" da gui \n");

rf12\_sendStart(RF12\_HDR\_DST | SendData[1], SendData , sizeof SendData);

}

delay(1000);

}

Repeater

1. Repeater - Jeenode code:
   1. This code use the RF12 and Ports library
   2. The code will be the repeater.
2. Code
   1. Setup()
      1. Set up the serial, to send information on screen of Serial Monitor ( of Arduino)
      2. Initalize the RF12: NodeID =2, 915Mhz, group ID 212
   2. Check\_status()
      1. Check byte0 of *ReceiveData{}*
         1. Byte0 =1 🡪 node is repeater
            1. Delete the current node ID:

for (i = 1; i < rf12\_len-1; ++i)

ReceiveData[i]=ReceiveData[i+1];

* + - 1. Byte0 <>1 -> node is not repeater
  1. Loop()
     1. Receive Rf data
     2. Check if repeater than send the information to next node (node 3).

#include <RF12.h>

#include <Ports.h>

byte ReceiveData[60];

byte Map[10];

int i;

void setup ()

{

Serial.begin(9600);

i=2;

rf12\_initialize(2, RF12\_915MHZ, 212);

Serial.print(" Chuong trinh node 2 \n");

Serial.print(char(i+48));

}

int check\_status()

{Serial.print(" da vao status \n");

if (ReceiveData[0]!=0x01)

{

ReceiveData[0]=ReceiveData[0]-1;

for (i = 1; i < rf12\_len-1; ++i)

ReceiveData[i]=ReceiveData[i+1];

Serial.print(" da la repeater \n");

return 1; //repeater

}

else return 0; //destination

}

void loop ()

{

if (rf12\_recvDone() && rf12\_crc == 0)

{

for (i = 0; i < rf12\_len; ++i)

ReceiveData[i]=rf12\_data[i];

if (check\_status()) //this is a repeater

{

rf12\_recvDone();

if (rf12\_canSend())

{

rf12\_sendStart(RF12\_HDR\_DST | ReceiveData[1], ReceiveData , sizeof ReceiveData);

Serial.print(" da gui den ID3 \n");

Serial.print(char(ReceiveData[0]+48));

Serial.print(char(ReceiveData[1]+48));

Serial.print(char(ReceiveData[2]));

}

}

} //end if

}

Destination

1. Destination - Jeenode code:
   1. This code use the RF12 and Ports library
   2. The code will be the Destination.
2. Code
   1. Variable:
      1. Data{}: store the information will send
         1. Route
         2. Action
         3. Data
         4. ReceiveData[60]: received massage from RF
   2. Setup()
      1. Set up the serial, to send information on screen of Serial Monitor ( of Arduino)
      2. Initalize the RF12: NodeID =2, 915Mhz, group ID 212
   3. Check\_status()
      1. Check byte0 of *ReceiveData{}*
         1. Byte0 =1 🡪 node is repeater
            1. Delete the current node ID:

for (i = 1; i < rf12\_len-1; ++i)

ReceiveData[i]=ReceiveData[i+1];

* + - 1. Byte0 <>1 -> node is not repeater
  1. Send\_Data(): Send data to next node using the RF library.
  2. Loop()
     1. Receive Rf data
        1. Check if node repeater than send the information to next node.
        2. If node is destination :
           1. Choose action in received data :

‘R’: request 🡪 respond by sending data

Choosing the request

‘H’ 🡪 send the humidity data

‘T’ 🡪 Temperatiure

‘L’ 🡪 Light sensor

‘O’ 🡪 Occupacy

‘W’

‘D’: data

‘M’: maps

Store the map to the data{}

#include <RF12.h>

#include <Ports.h>

byte ReceiveData[60];

byte Data[60];

int i;

void setup ()

{

Serial.begin(9600);

rf12\_initialize(3, RF12\_915MHZ, 212);

}

int check\_status()

{

int temp=0;

Serial.print("Checking node status... \n");

if (ReceiveData[0]!=0x01)

{

ReceiveData[0]=ReceiveData[0]-1;

for (i = 1; i < rf12\_len-1; ++i)

ReceiveData[i]=ReceiveData[i+1];

temp= 1; //repeater

}

return temp;

}

void send\_Data()

{

Serial.print("Sending... \n");

rf12\_recvDone();

if (rf12\_canSend())

{

rf12\_sendStart(RF12\_HDR\_DST |Data[1],Data , sizeof Data);

}

}

void loop ()

{

Data[59]= 0xFF;

if (rf12\_recvDone() && rf12\_crc == 0)

{

for (i = 0; i < rf12\_len; ++i)

ReceiveData[i]=rf12\_data[i];

if (check\_status()==1) //this is a repeater

{

Serial.print("Node is Repeater \n");

rf12\_recvDone();

if (rf12\_canSend())

{

rf12\_sendStart(RF12\_HDR\_DST | ReceiveData[1], ReceiveData , sizeof ReceiveData);

}

}

else //this is the final destination

{

Serial.print("Node is Destination \n");

switch (ReceiveData[2])

{

case byte('R'):

{

Serial.print(" Recieved Request: \n ");

i = Data[0];

Data[i+1] = byte('D');

Data[i+2] = 1;

Data[i+5] = 0xFF;

switch (ReceiveData[3]) // Type of Request

{

case byte('H'):

{

Serial.print(" Recieved Request Information of Humidity: \n ");

Data[i+3] = byte('H');

Data[i+4] = 98; // information of Humidity put on here

// send data

send\_Data();

Serial.print(" Sent Request Information of Humidity: \n ");

break;

} // end of case byte ("H")

case byte('T'):

{

Serial.print(" Recieved Request Information of Temperature: \n ");

Data[i+3] = byte('T');

Data[i+4] = 42; // information of Temperature put on here

// send data

send\_Data();

Serial.print(" Sent Request Information of Temperature: \n ");

break;

} // end of case byte ("T")

case byte('L'):

{

Data[i+3] = byte('L');

Data[i+4] = 50; // information of Light put on here

// send data

send\_Data();

Serial.print(" Sent Request Information of Light \n ");

break;

} // end of case byte ("L")

case byte('O'):

{

Data[i+3] = byte('0');

Data[i+4] = 1; // information of Occupancy put on here

// send data

send\_Data();

Serial.print(" Sent Request Information of Occupancy \n ");

break;

} // end of case byte ("O")

case byte('W'):

{

Serial.print(" Recieved Request Information of Waterlevel \n ");

Data[i+3] = byte('W');

Data[i+4] = 105; // information of Occupancy put on here

break;

} // end of case byte ("W")

default:

{

Data[i+2] = 4; // Number of sensor

// then

Serial.print(" Default: Send all information ");

break;

}// end of default

;

}// end switch (ReceiveData[3])

break;

} // end of case byte (R)

case byte('D'):

{

Serial.print(" Recieved Data: \n");

break;

} // end of case byte("D")

case byte('M'): //if Data, store the return direction into Data[]

{

Serial.print("Recieved Data \n");

for (i = 0; i < 3; ++i)

{

//if (ReceiveData[3+i]==0xFF); //if the end character, exit for

Data[i]=ReceiveData[3+i];// bytes after 'M'

}

Serial.print("Map Information: \n");

Serial.print("Number of node: "); Serial.print(Data[0],10);

Serial.print(" ID[1] = "); Serial.print(Data[1],10);

Serial.print(" ID[2] = "); Serial.print(Data[2],10);

if (Data[0]==0x02 && Data[1]==0x02 && Data[2]==0x01)

Serial.print("\nMap is correct\n");

break;

} // end of case byte("M")

default:

Serial.print("Recieved BAD Data \n");

;

// if nothing else matches, do the default

// default is optional

} // end switch (ReceiveData[2])

} //end else

} //end if

} //end void

### Date 02/18/2010

1. Join code of Repeater and Destination

2. Add function ***show\_data(byte \*m)*** to show the value of array m to serial monitor.

**3. Algorithm**

1. Variable:
   * 1. Data{}: store the information will send
        1. Route
        2. Action
        3. Data
        4. ReceiveData[60]: received massage from RF
2. Setup()
   * 1. Set up the serial, to send information on screen of Serial Monitor ( of Arduino)
     2. Initalize the RF12: NodeID =2 *(it can be change)*, 915Mhz, group ID 212
3. Check\_status(): check the status of node is repeater or destination
   * 1. Check byte0 of *ReceiveData{}*
        1. Byte0 =1 🡪 node is repeater
           1. Delete the current node ID:

for (i = 1; i < rf12\_len-1; ++i)

ReceiveData[i]=ReceiveData[i+1];

* + - 1. Byte0 <>1 -> node is not repeater

1. Send\_Data(): Send data to next node.
2. **Loop()**
   * 1. Check: Receive Rf data
        1. Check if node repeater than send the information to next node.
        2. If node is destination :
           1. Choose action in received data :

‘R’: request 🡪 respond by sending data

Choosing the request

‘H’ 🡪 send the humidity data

‘T’

‘L’

‘O’

‘W’

‘D’: data

‘M’: maps

Store the map to the data{}

#include <RF12.h>

#include <Ports.h>

byte ReceiveData[60];

byte Data[60];

int i;

int LengthOfMap;

//-----------

void setup ()

{

Serial.begin(9600);

rf12\_initialize(3, RF12\_915MHZ, 212);

Serial.print("This is node 3 \n");

} // end of setup()

int check\_status()

{

int temp=0;

Serial.print("Checking node status... \n");

if (ReceiveData[0]!=0x01)

{

ReceiveData[0]=ReceiveData[0]-1;

for (i = 1; i < rf12\_len-1; ++i)

ReceiveData[i]=ReceiveData[i+1];

temp= 1; //repeater

}

return temp;

}

//--------

void send\_Data()

{

Serial.print("Sending... \n");

rf12\_recvDone();

if (rf12\_canSend())

{

rf12\_sendStart(RF12\_HDR\_DST |Data[1],Data , sizeof Data);

}

}

//--------

**void show\_data(byte \*m)**

**{**

**byte i;**

**for (i = 0; i<15 ; ++i)**

**{**

**Serial.print(m[i],10);**

**Serial.print(" ");**

**}**

**Serial.print(" | \n");**

**}**

//---------

void loop ()

{

Data[59]= 0xFF;

if (rf12\_recvDone() && rf12\_crc == 0)

{

**Serial.print(" ReceiveData: ");**

**show\_data(ReceiveData);**

for (i = 0; i < rf12\_len; ++i)

ReceiveData[i]=rf12\_data[i];

if (check\_status()==1) //this is a repeater

{

Serial.print("Node is Repeater \n");

rf12\_recvDone();

if (rf12\_canSend())

{

rf12\_sendStart(RF12\_HDR\_DST | ReceiveData[1], ReceiveData , sizeof ReceiveData);

}

}

else //this is the final destination

{

Serial.print("Node is Destination \n");

switch (ReceiveData[2])

{

case byte('R'):

{

Serial.print(" Recieved Request: \n ");

i = Data[0];

Data[i+1] = byte('D');

Data[i+2] = 1;

Data[i+5] = 0xFF;

switch (ReceiveData[3]) // Type of Request

{

case byte('H'):

{

Serial.print(" Recieved Request Information of Humidity: \n ");

Data[i+3] = byte('H');

Data[i+4] = 98; // information of Humidity put on here

// send data

send\_Data();

Serial.print(" Sent Request Information of Humidity: \n ");

break;

} // end of case byte ("H")

case byte('T'):

{

Serial.print(" Recieved Request Information of Temperature: \n ");

Data[i+3] = byte('T');

Data[i+4] = 42; // information of Temperature put on here

// send data

send\_Data();

Serial.print(" Sent Request Information of Temperature: \n ");

break;

} // end of case byte ("T")

case byte('L'):

{

Data[i+3] = byte('L');

Data[i+4] = 50; // information of Light put on here

// send data

send\_Data();

Serial.print(" Sent Request Information of Light \n ");

break;

} // end of case byte ("L")

case byte('O'):

{

Data[i+3] = byte('0');

Data[i+4] = 1; // information of Occupancy put on here

// send data

send\_Data();

Serial.print(" Sent Request Information of Occupancy \n ");

break;

} // end of case byte ("O")

case byte('W'):

{

Serial.print(" Recieved Request Information of Waterlevel \n ");

Data[i+3] = byte('W');

Data[i+4] = 105; // information of Occupancy put on here

break;

} // end of case byte ("W")

default:

{

Data[i+2] = 4; // Number of sensor

// then

Serial.print(" Default: Send all information ");

break;

}// end of default

;

}// end switch (ReceiveData[3])

break;

} // end of case byte (R)

case byte('D'):

{

Serial.print(" Recieved Data: \n");

break;

} // end of case byte("D")

case byte('M'): //if Data, store the return direction into Data[]

{

Serial.print("Recieved Data \n");

LengthOfMap = ReceiveData[0]+1;

for (i = 0; i < LengthOfMap; ++i) // assume that just have 3 node

{

//if (ReceiveData[3+i]==0xFF); //if the end character, exit for

Data[i]=ReceiveData[3+i];// bytes after 'M'

}

Serial.print("Map Information: \n");

Serial.print("Number of node: "); Serial.print(Data[0],10);

Serial.print(" ID[1] = "); Serial.print(Data[1],10);

Serial.print(" ID[2] = "); Serial.print(Data[2],10);

if (Data[0]==0x02 && Data[1]==0x02 && Data[2]==0x01)

Serial.print("\nMap is correct\n");

break;

} // end of case byte("M")

default:

Serial.print("Recieved BAD Data \n");

;

// if nothing else matches, do the default

// default is optional

} // end switch (ReceiveData[2])

Serial.print(" Data: ");

show\_data(Data);

} //end else

} //end if

} //end void

Adding the information to the serial monitor:

#include <RF12.h>

#include <Ports.h>

byte ReceiveData[60];

byte Data[60];

int i;

int LengthOfMap;

//-----------

void setup ()

{

Serial.begin(9600);

rf12\_initialize(2, RF12\_915MHZ, 212);

Serial.print("This is node 2 \n");

} // end of setup()

int check\_status()

{

int temp=0;

Serial.print("Checking node status... \n");

if (ReceiveData[0]!=0x01)

{

ReceiveData[0]=ReceiveData[0]-1;

for (i = 1; i < rf12\_len-1; ++i)

ReceiveData[i]=ReceiveData[i+1];

temp= 1; //repeater

}

return temp;

}

//--------

void send\_Data()

{

Serial.print("Sending... \n");

rf12\_recvDone();

if (rf12\_canSend())

{

rf12\_sendStart(RF12\_HDR\_DST |Data[1],Data , sizeof Data);

}

}

//--------

void show\_data(byte \*m)

{

byte i;

for (i = 0; i<15 ; ++i)

{

Serial.print(m[i],10);

Serial.print(" ");

}

Serial.print(" | \n");

}

//---------

void loop ()

{

Data[59]= 0xFF;

if (rf12\_recvDone() && rf12\_crc == 0)

{

Serial.print(" ---=== Begin ===--- \n");

for (i = 0; i < rf12\_len; ++i)

ReceiveData[i]=rf12\_data[i];

Serial.print(" ReceiveData: ");

show\_data(ReceiveData);

if (check\_status()==1) //this is a repeater

{

Serial.print("Node is Repeater \n");

rf12\_recvDone();

if (rf12\_canSend())

{

rf12\_sendStart(RF12\_HDR\_DST | ReceiveData[1], ReceiveData , sizeof ReceiveData);

}

}

else //this is the final destination

{

Serial.print("Node is Destination \n");

switch (ReceiveData[2])

{

case byte('R'):

{

Serial.print(" Recieved Request: \n ");

i = Data[0];

Data[i+1] = byte('D');

Data[i+2] = 1;

Data[i+5] = 0xFF;

switch (ReceiveData[3]) // Type of Request

{

case byte('H'):

{

Serial.print(" Recieved Request Information of Humidity: \n ");

Data[i+3] = byte('H');

Data[i+4] = 98; // information of Humidity put on here

// send data

send\_Data();

Serial.print(" Sent Request Information of Humidity: \n ");

break;

} // end of case byte ("H")

case byte('T'):

{

Serial.print(" Recieved Request Information of Temperature: \n ");

Data[i+3] = byte('T');

Data[i+4] = 42; // information of Temperature put on here

// send data

send\_Data();

Serial.print(" Sent Request Information of Temperature: \n ");

break;

} // end of case byte ("T")

case byte('L'):

{

Data[i+3] = byte('L');

Data[i+4] = 50; // information of Light put on here

// send data

send\_Data();

Serial.print(" Sent Request Information of Light \n ");

break;

} // end of case byte ("L")

case byte('O'):

{

Data[i+3] = byte('0');

Data[i+4] = 1; // information of Occupancy put on here

// send data

send\_Data();

Serial.print(" Sent Request Information of Occupancy \n ");

break;

} // end of case byte ("O")

case byte('W'):

{

Serial.print(" Recieved Request Information of Waterlevel \n ");

Data[i+3] = byte('W');

Data[i+4] = 105; // information of Occupancy put on here

break;

} // end of case byte ("W")

default:

{

Data[i+2] = 4; // Number of sensor

// then

Serial.print(" Default: Send all information ");

break;

}// end of default

;

}// end switch (ReceiveData[3])

break;

} // end of case byte (R)

case byte('D'):

{

Serial.print(" Recieved Data: \n");

break;

} // end of case byte("D")

case byte('M'): //if Data, store the return direction into Data[]

{

LengthOfMap = ReceiveData[0]+1;

for (i = 0; i < LengthOfMap; ++i) // assume that just have 3 node

{

//if (ReceiveData[3+i]==0xFF); //if the end character, exit for

Data[i]=ReceiveData[3+i];// bytes after 'M'

}

Serial.print("Map Information: \n");

Serial.print("Number of node: "); Serial.print(Data[0],10);

Serial.print(" ID[1] = "); Serial.print(Data[1],10);

Serial.print(" ID[2] = "); Serial.print(Data[2],10);

if (Data[0]==0x02 && Data[1]==0x02 && Data[2]==0x01) Serial.print("\nMap is correct\n");

break;

} // end of case byte("M")

default:

Serial.print("Recieved BAD Data \n");

;

// if nothing else matches, do the default

// default is optional

} // end switch (ReceiveData[2])

Serial.print(" Data sent: ");

show\_data(Data);

Serial.print(" ---=== End ===--- \n");

} //end else

} //end if

} //end void

### Date 02/19/2011

This follow code can be uploaded to any node:

* It can be repeater
* Or it can be destination node
  + Send the ‘fake’ sensor data
  + Received maps.

Adding:

1. Send\_data()
   1. Check if node has map information or not
2. The variable *LengthOfMap 🡪* the number of node in the path; 255 means no map

#include <RF12.h>

#include <Ports.h>

byte ReceiveData[60];

byte Data[60];

byte i;

byte LengthOfMap=255;

byte NodeID = 2; // Node ID of this node

//-----------

void setup ()

{

Serial.begin(9600);

rf12\_initialize(3, RF12\_915MHZ, 212);

Serial.print("This is node 3 \n");

} // end of setup()

int check\_status()

{

int temp=0;

Serial.print("Checking node status... \n");

if (ReceiveData[0]!=0x01)

{

ReceiveData[0]=ReceiveData[0]-1;

for (i = 1; i < rf12\_len-1; ++i)

ReceiveData[i]=ReceiveData[i+1];

temp= 1; //repeater

}

return temp;

}

//--------

void send\_Data()

{

Serial.print("Sending... \n");

if (LengthOfMap==255)

{

Serial.print("No information of Map to Server \n");

return;

}

rf12\_recvDone();

if (rf12\_canSend())

{

rf12\_sendStart(RF12\_HDR\_DST |Data[1],Data , sizeof Data);

}

}

//--------

void show\_data(byte \*m)

{

byte i;

for (i = 0; i<20 ; ++i)

{

Serial.print(m[i],10);

Serial.print(" ");

}

Serial.print("\n");

}

//---------

void loop ()

{

Data[59]= 0xFF;

if (rf12\_recvDone() && rf12\_crc == 0)

{

Serial.print(" ---=== Begin ===--- \n");

for (i = 0; i < rf12\_len; ++i)

ReceiveData[i]=rf12\_data[i];

Serial.print(" ReceiveData: ");

show\_data(ReceiveData);

if (check\_status()==1) //this is a repeater

{

Serial.print("Node is Repeater \n");

rf12\_recvDone();

if (rf12\_canSend())

{

rf12\_sendStart(RF12\_HDR\_DST | ReceiveData[1], ReceiveData , sizeof ReceiveData);

}

Serial.print(" Data sent: ");

show\_data(ReceiveData);

Serial.print(" ---=== End ===--- \n");

}

else //this is the final destination

{

Serial.print("Node is Destination \n");

switch (ReceiveData[2])

{

case byte('R'):

{

Serial.print(" Recieved Request: \n ");

i = Data[0];

Data[i+1] = byte('D');

Data[i+2] = 1;

Data[i+5] = 0xFF;

switch (ReceiveData[3]) // Type of Request

{

case byte('H'):

{

Serial.print(" Recieved Request Information of Humidity: \n ");

Data[i+3] = byte('H');

Data[i+4] = 98; // information of Humidity put on here

// send data

send\_Data();

Serial.print(" Sent Request Information of Humidity: \n ");

break;

} // end of case byte ("H")

case byte('T'):

{

Serial.print(" Recieved Request Information of Temperature: \n ");

Data[i+3] = byte('T');

Data[i+4] = 42; // information of Temperature put on here

// send data

send\_Data();

Serial.print(" Sent Request Information of Temperature: \n ");

break;

} // end of case byte ("T")

case byte('L'):

{

Data[i+3] = byte('L');

Data[i+4] = 50; // information of Light put on here

// send data

send\_Data();

Serial.print(" Sent Request Information of Light \n ");

break;

} // end of case byte ("L")

case byte('O'):

{

Data[i+3] = byte('O');

Data[i+4] = 1; // information of Occupancy put on here

// send data

send\_Data();

Serial.print(" Sent Request Information of Occupancy \n ");

break;

} // end of case byte ("O")

case byte('W'):

{

Serial.print(" Recieved Request Information of Waterlevel \n ");

Data[i+3] = byte('W');

Data[i+4] = 105; // information of Waterlevel put on here

send\_Data();

break;

} // end of case byte ("W")

default:

{

Data[i+2] = 4; // Number of sensor

Data[i+3] = byte('H'); Data[i+4] = 98; // sensor huminity

Data[i+5] = byte('T'); Data[i+6] = 42; // sensor temp

Data[i+7] = byte('O'); Data[i+8] = 1; // sensor occupation

Data[i+9] = byte('L'); Data[i+10] = 30; // sensor Light

Data[i+11] = 255;

// then

Serial.print(" Default: Send all information \n");

send\_Data();

break;

}// end of default

;

}// end switch (ReceiveData[3])

break;

} // end of case byte (R)

case byte('D'):

{

Serial.print(" Recieved Data: \n");

break;

} // end of case byte("D")

case byte('M'): //if Data, store the return direction into Data[]

{

LengthOfMap = ReceiveData[3];

for (i = 0; i < LengthOfMap + 1; ++i) // assume that just have 3 node

{

//if (ReceiveData[3+i]==0xFF); //if the end character, exit for

Data[i]=ReceiveData[3+i];// bytes after 'M'

}

Serial.print("Map Information: \n"); Serial.print(LengthOfMap,10);

Serial.print("Number of node: "); Serial.print(Data[0],10);

Serial.print(" ID[1] = "); Serial.print(Data[1],10);

Serial.print(" ID[2] = "); Serial.print(Data[2],10);

if (Data[0]==0x02 && Data[1]==0x02 && Data[2]==0x01) Serial.print("\nMap is correct\n");

break;

} // end of case byte("M")

default:

Serial.print("Recieved BAD Data \n");

;

// if nothing else matches, do the default

// default is optional

} // end switch (ReceiveData[2])

Serial.print(" Data sent: ");

show\_data(Data);

Serial.print(" ---=== End ===--- \n");

} //end else

} //end if

} //end void

<https://projects.cecs.pdx.edu:8443/~whitem/OSHERS/index.cgi/browser/Coding/Communication/node3/node3.pde>

## Done with communication

### Date 02/22/2011

Team meeting: discuss about proposal.

Representation next week

### Date 02/24/2011

Team meeting

Preparation for presentation 02/28/2011

## Adding the information of data to communication code - Work with Nam

### Date 03/03/2011

add: read\_sensor(): reading the sensor information.

adding light data, after the request:

**Data[i+4] = lightData[0]; // information of Light put on here**

**Data[i+5] = lightData[1]; // information of Light put on here**

**Data[i+6] = lightData[2]; // information of Light put on here**

**Data[i+7] = lightData[3]; // information of Light put on here**

<https://projects.cecs.pdx.edu:8443/~whitem/OSHERS/index.cgi/browser/Coding/Communication/Communication_sensor/commu_sensor.pde?rev=34>

compeleted code:

#include <DHT22.h>

#include <Ports.h>

#include <RF12.h>

#include <stdlib.h>

#include <NewSoftSerial.h>

//////////////////////////////////////////////////////////////////////

// PIR connect to port 1 (pins 4 and 14(A0) of arduino)

NewSoftSerial PIR(14, 4); //4 is TX (connect to RX), 14 is RX (connect to TX)

// Ambient light sensor connect to port 2 (pins analog 1 of arduino)

Port LightSensor(2); // This pin is used as analog input from sensor (A0 in Arduino = pin A of port 1 in JeeNode)

// Humidity/ Temp sensor connect to port 3 JeeNode(pin 6 arduino)

#define DHT22\_PIN 6

//Port DHT22\_PIN (3);

int myLight; // value read from the light sensor

float myTemp; // value read from the Humid/temp sensor

float myRH;

char lightData[4];

char humidData[4];

char tempData[4];

char pirData[1];

char data[13]; //data that will be sent to the JeeLink

// Setup a DHT22 instance

DHT22 myDHT22(DHT22\_PIN);

static char getch()

{

while (!PIR.available())

;

return PIR.read();

}

void read\_sensor()

{

int i;

DHT22\_ERROR\_t errorCode;

delay(2000); //delay for the humid/temp sensor stabelized

//read the humidity/temp sensor

errorCode = myDHT22.readData();

if(errorCode == DHT\_ERROR\_NONE) //if non error occur

{

myRH = myDHT22.getHumidity();

myTemp = myDHT22.getTemperatureC();

}

else //if error occur, data is all zero

{

myRH = 0;

myTemp = 0;

}

dtostrf(myRH, 4, 2, humidData); //convert float to char array

dtostrf(myTemp, 4, 2, tempData); //convert float to char array

// read the ambient light sensor value

myLight = LightSensor.anaRead();

itoa(myLight, lightData, 10); //convert int to char arry

// read the PIR

PIR.print('a');

pirData[0] = getch();

}

////////////////////////////////////////////////////////////////////////////////////

byte ReceiveData[60];

byte Data[60];

byte i;

byte LengthOfMap=255;

//byte NodeID = 3; // Node ID of this node

//-----------

void setup ()

{

Serial.begin(9600);

rf12\_initialize(3, RF12\_915MHZ, 212);

Serial.print("This is node 3 \n");

} // end of setup()

int check\_status()

{

int temp=0;

Serial.print("Checking node status... \n");

if (ReceiveData[0]!=0x01)

{

ReceiveData[0]=ReceiveData[0]-1;

for (i = 1; i < rf12\_len-1; ++i)

ReceiveData[i]=ReceiveData[i+1];

temp= 1; //repeater

}

return temp;

}

//--------

void send\_Data()

{

Serial.print("Sending... \n");

if (LengthOfMap==255)

{

Serial.print("No information of Map to Server \n");

return;

}

rf12\_recvDone();

if (rf12\_canSend())

{

rf12\_sendStart(RF12\_HDR\_DST |Data[1],Data , sizeof Data);

}

}

//--------

void show\_data(byte \*m)

{

byte i;

for (i = 0; i<20 ; ++i)

{

Serial.print(m[i],10);

Serial.print(" ");

}

Serial.print("\n");

}

//---------

void loop ()

{

// Reading sensor value function.

// read\_sensor();

Data[59]= 0xFF;

if (rf12\_recvDone() && rf12\_crc == 0)

{

Serial.print(" ---=== Begin ===--- \n");

for (i = 0; i < rf12\_len; ++i)

ReceiveData[i]=rf12\_data[i];

Serial.print(" ReceiveData: ");

show\_data(ReceiveData);

if (check\_status()==1) //this is a repeater

{

Serial.print("Node is Repeater \n");

rf12\_recvDone();

if (rf12\_canSend())

{

rf12\_sendStart(RF12\_HDR\_DST | ReceiveData[1], ReceiveData , sizeof ReceiveData);

}

Serial.print(" Data sent: ");

show\_data(ReceiveData);

Serial.print(" ---=== End ===--- \n");

}

else //this is the final destination

{

Serial.print("Node is Destination \n");

switch (ReceiveData[2])

{

case byte('R'):

{

Serial.print(" Recieved Request: \n ");

// read\_sensor();

i = Data[0];

Data[i+1] = byte('D');

Data[i+2] = 1;

Data[i+10] = 0xFF;

switch (ReceiveData[3]) // Type of Request

{

case byte('H'):

{

Serial.print(" Recieved Request Information of Humidity: \n ");

Data[i+3] = byte('H');

Data[i+4] = 98; // information of Humidity put on here

// send data

send\_Data();

Serial.print(" Sent Request Information of Humidity: \n ");

break;

} // end of case byte ("H")

case byte('T'):

{

Serial.print(" Recieved Request Information of Temperature: \n ");

Data[i+3] = byte('T');

Data[i+4] = 42; // information of Temperature put on here

// send data

send\_Data();

Serial.print(" Sent Request Information of Temperature: \n ");

break;

} // end of case byte ("T")

case byte('L'):

{

Data[i+3] = byte('L');

// lightData[4]

**Data[i+4] = lightData[0]; // information of Light put on here**

**Data[i+5] = lightData[1]; // information of Light put on here**

**Data[i+6] = lightData[2]; // information of Light put on here**

**Data[i+7] = lightData[3]; // information of Light put on here**

// send data

send\_Data();

Serial.print(" Sent Request Information of Light \n ");

break;

} // end of case byte ("L")

case byte('O'):

{

Data[i+3] = byte('O');

Data[i+4] = 1; // information of Occupancy put on here

// send data

send\_Data();

Serial.print(" Sent Request Information of Occupancy \n ");

break;

} // end of case byte ("O")

case byte('W'):

{

Serial.print(" Recieved Request Information of Waterlevel \n ");

Data[i+3] = byte('W');

Data[i+4] = 105; // information of Waterlevel put on here

send\_Data();

break;

} // end of case byte ("W")

default:

{

Data[i+2] = 4; // Number of sensor

Data[i+3] = byte('H'); Data[i+4] = 98; // sensor huminity

Data[i+5] = byte('T'); Data[i+6] = 42; // sensor temp

Data[i+7] = byte('O'); Data[i+8] = 1; // sensor occupation

Data[i+9] = byte('L'); Data[i+10] = 30; // sensor Light

Data[i+11] = 255;

// then

Serial.print(" Default: Send all information \n");

send\_Data();

break;

}// end of default

;

}// end switch (ReceiveData[3])

break;

} // end of case byte (R)

case byte('D'):

{

Serial.print(" Recieved Data: \n");

break;

} // end of case byte("D")

case byte('M'): //if Data, store the return direction into Data[]

{

LengthOfMap = ReceiveData[3];

for (i = 0; i < LengthOfMap + 1; ++i) // assume that just have 3 node

{

//if (ReceiveData[3+i]==0xFF); //if the end character, exit for

Data[i]=ReceiveData[3+i];// bytes after 'M'

}

Serial.print("Map Information: \n"); Serial.print(LengthOfMap,10);

Serial.print("Number of node: "); Serial.print(Data[0],10);

Serial.print(" ID[1] = "); Serial.print(Data[1],10);

Serial.print(" ID[2] = "); Serial.print(Data[2],10);

if (Data[0]==0x02 && Data[1]==0x02 && Data[2]==0x01)

Serial.print("\nMap is correct\n");

break;

} // end of case byte("M")

default:

Serial.print("Recieved BAD Data \n");

;

// if nothing else matches, do the default

// default is optional

} // end switch (ReceiveData[2])

Serial.print(" Data sent: ");

show\_data(Data);

Serial.print(" ---=== End ===--- \n");

} //end else

} //end if

} //end void

### Date 03/04/2011

Add humidity and temperature data:

#include <DHT22.h>

#include <Ports.h>

#include <RF12.h>

#include <stdlib.h>

#include <NewSoftSerial.h>

//////////////////////////////////////////////////////////////////////

// PIR connect to port 1 (pins 4 and 14(A0) of arduino)

NewSoftSerial PIR(14, 4); //4 is TX (connect to RX), 14 is RX (connect to TX)

// Ambient light sensor connect to port 2 (pins analog 1 of arduino)

Port LightSensor(2); // This pin is used as analog input from sensor (A0 in Arduino = pin A of port 1 in JeeNode)

// Humidity/ Temp sensor connect to port 3 JeeNode(pin 6 arduino)

#define DHT22\_PIN 6

//Port DHT22\_PIN (3);

int myLight; // value read from the light sensor

float myTemp; // value read from the Humid/temp sensor

float myRH;

char lightData[4];

char humidData[4];

char tempData[4];

char pirData[1];

char data[13]; //data that will be sent to the JeeLink

// Setup a DHT22 instance

DHT22 myDHT22(DHT22\_PIN);

static char getch()

{

while (!PIR.available())

;

return PIR.read();

}

void read\_sensor()

{

int i;

DHT22\_ERROR\_t errorCode;

Serial.print(" Reading sensor... \n ");

delay(2000); //delay for the humid/temp sensor stabelized

//read the humidity/temp sensor

errorCode = myDHT22.readData();

if(errorCode == DHT\_ERROR\_NONE) //if non error occur

{

myRH = myDHT22.getHumidity();

myTemp = myDHT22.getTemperatureC();

}

else //if error occur, data is all zero

{

myRH = 0;

myTemp = 0;

}

dtostrf(myRH, 4, 2, humidData); //convert float to char array

dtostrf(myTemp, 4, 2, tempData); //convert float to char array

// read the ambient light sensor value

myLight = LightSensor.anaRead();

itoa(myLight, lightData, 10); //convert int to char arry

// read the PIR

PIR.print('a');

pirData[0] = getch();

Serial.print(" Done reading sensor \n ");

}

////////////////////////////////////////////////////////////////////////////////////

byte ReceiveData[60];

byte Data[60];

byte i;

byte LengthOfMap=255;

//byte NodeID = 3; // Node ID of this node

//-----------

void setup ()

{

Serial.begin(9600);

rf12\_initialize(3, RF12\_915MHZ, 212);

Serial.print("This is node 3 \n");

PIR.begin(9600);

} // end of setup()

int check\_status()

{

int temp=0;

Serial.print("Checking node status... \n");

if (ReceiveData[0]!=0x01)

{

ReceiveData[0]=ReceiveData[0]-1;

for (i = 1; i < rf12\_len-1; ++i)

ReceiveData[i]=ReceiveData[i+1];

temp= 1; //repeater

}

return temp;

}

//--------

void send\_Data()

{

Serial.print("Sending... \n");

if (LengthOfMap==255)

{

Serial.print("No information of Map to Server \n");

return;

}

rf12\_recvDone();

if (rf12\_canSend())

{

rf12\_sendStart(RF12\_HDR\_DST |Data[1],Data , sizeof Data);

}

}

//--------

void show\_data(byte \*m)

{

byte i;

for (i = 0; i<20 ; ++i)

{

Serial.print(m[i],10);

Serial.print(" ");

}

Serial.print("\n");

}

//---------

void loop ()

{

// Reading sensor value function.

// read\_sensor();

Data[59]= 0xFF;

if (rf12\_recvDone() && rf12\_crc == 0)

{

Serial.print(" ---=== Begin ===--- \n");

for (i = 0; i < rf12\_len; ++i)

ReceiveData[i]=rf12\_data[i];

Serial.print(" ReceiveData: ");

show\_data(ReceiveData);

if (check\_status()==1) //this is a repeater

{

Serial.print("Node is Repeater \n");

rf12\_recvDone();

if (rf12\_canSend())

{

rf12\_sendStart(RF12\_HDR\_DST | ReceiveData[1], ReceiveData , sizeof ReceiveData);

}

Serial.print(" Data sent: ");

show\_data(ReceiveData);

Serial.print(" ---=== End ===--- \n");

}

else //this is the final destination

{

Serial.print("Node is Destination \n");

switch (ReceiveData[2])

{

case byte('R'):

{

Serial.print(" Recieved Request: \n ");

read\_sensor();

i = Data[0];

Data[i+1] = byte('D');

Data[i+2] = 1;

Data[i+10] = 0xFF;

switch (ReceiveData[3]) // Type of Request

{

case byte('H'):

{

Serial.print(" Recieved Request Information of Humidity: \n ");

Data[i+3] = byte('H');

Data[i+4] = humidData[0]; // information of Humidity put on here

Data[i+5] = humidData[1]; // information of Humidity put on here

Data[i+6] = humidData[2]; // information of Humidity put on here

Data[i+7] = humidData[3]; // information of Humidity put on here

// send data

send\_Data();

Serial.print(" Sent Request Information of Humidity: \n ");

break;

} // end of case byte ("H")

case byte('T'):

{

Serial.print(" Recieved Request Information of Temperature: \n ");

Data[i+3] = byte('T');

Data[i+4] = tempData[0]; // information of Temperature put on here

Data[i+5] = tempData[1]; // information of Temperature put on here

Data[i+6] = tempData[2]; // information of Temperature put on here

Data[i+7] = tempData[3]; // information of Temperature put on here

// send data

send\_Data();

Serial.print(" Sent Request Information of Temperature: \n ");

break;

} // end of case byte ("T")

case byte('L'):

{

Data[i+3] = byte('L');

// lightData[4]

Data[i+4] = lightData[0]; // information of Light put on here

Data[i+5] = lightData[1]; // information of Light put on here

Data[i+6] = lightData[2]; // information of Light put on here

Data[i+7] = lightData[3]; // information of Light put on here

// send data

send\_Data();

Serial.print(" Sent Request Information of Light \n ");

break;

} // end of case byte ("L")

case byte('O'):

{

Data[i+3] = byte('O');

Data[i+4] = 1; // information of Occupancy put on here

// send data

send\_Data();

Serial.print(" Sent Request Information of Occupancy \n ");

break;

} // end of case byte ("O")

case byte('W'):

{

Serial.print(" Recieved Request Information of Waterlevel \n ");

Data[i+3] = byte('W');

Data[i+4] = 105; // information of Waterlevel put on here

send\_Data();

break;

} // end of case byte ("W")

default:

{

Data[i+2] = 4; // Number of sensor

Data[i+3] = byte('H'); Data[i+4] = 98; // sensor huminity

Data[i+5] = byte('T'); Data[i+6] = 42; // sensor temp

Data[i+7] = byte('O'); Data[i+8] = 1; // sensor occupation

Data[i+9] = byte('L'); Data[i+10] = 30; // sensor Light

Data[i+11] = 255;

// then

Serial.print(" Default: Send all information \n");

send\_Data();

break;

}// end of default

;

}// end switch (ReceiveData[3])

break;

} // end of case byte (R)

case byte('D'):

{

Serial.print(" Recieved Data: \n");

break;

} // end of case byte("D")

case byte('M'): //if Data, store the return direction into Data[]

{

LengthOfMap = ReceiveData[3];

for (i = 0; i < LengthOfMap + 1; ++i) // assume that just have 3 node

{

//if (ReceiveData[3+i]==0xFF); //if the end character, exit for

Data[i]=ReceiveData[3+i];// bytes after 'M'

}

Serial.print("Map Information: \n"); Serial.print(LengthOfMap,10);

Serial.print("Number of node: "); Serial.print(Data[0],10);

Serial.print(" ID[1] = "); Serial.print(Data[1],10);

Serial.print(" ID[2] = "); Serial.print(Data[2],10);

if (Data[0]==0x02 && Data[1]==0x02 && Data[2]==0x01) Serial.print("\nMap is correct\n");

break;

} // end of case byte("M")

default:

Serial.print("Recieved BAD Data \n");

;

// if nothing else matches, do the default

// default is optional

} // end switch (ReceiveData[2])

Serial.print(" Data sent: ");

show\_data(Data);

Serial.print(" ---=== End ===--- \n");

} //end else

} //end if

} //end void

**Work with Nam in the sensor code. Work with Hieu in the server code.**

**Don’t have any change in the communication code.**

### Date 04/07/2011

version Apr/7/2011 - Hai Nguyen:

\* add NodeID variable

\* Combine information of three sensor H, T, L, in one packet. Reponse to the Reques ‘A’

**Algorithm**

1. Variable:
   * 1. Data{}: store the information will send
        1. Route
        2. Action
        3. Data
     2. ReceiveData[60]: received massage from RF
     3. myLight: light information
     4. myTemp: temperature information
     5. myRH: relative humidity information
     6. NodeID
     7. LengthOfMap
2. read\_PIR\_sensor()
3. read\_other\_sensors()
4. Setup()
   * 1. Set up the serial, to send information on screen of Serial Monitor ( of Arduino)
     2. Initalize the RF12: NodeID =2 *(it can be change)*, 915Mhz, group ID 212
5. Check\_status(): check the status of node is repeater or destination
   * 1. Check byte0 of *ReceiveData{}*
        1. Byte0 =1 🡪 node is repeater
           1. Delete the current node ID:

for (i = 1; i < rf12\_len-1; ++i)

ReceiveData[i]=ReceiveData[i+1];

* + - 1. Byte0 <>1 -> node is not repeater

1. Send\_Data(): Send data to next node.
2. **Loop()**
   * 1. Check: Receive Rf data
        1. Check if node repeater than send the information to next node.
        2. If node is destination :
           1. Choose action in received data :

‘R’: request 🡪 respond by sending data

Choosing the request

**“A” : send information of H,T,L**

**“O” : send occupancy information**

**“W”**

‘D’: data

‘M’: maps

Store the map to the data{}

version Apr/6/2011:

split float to 4 bytes and send 4 bytes to server. Using the union:

typedef

union {float asFloat;byte asBytes[4];} f2b;

typedef

union {int asInt;byte asBytes[4];} i2b;

i2b myLight; // value read from the light sensor

f2b myTemp; // value read from the Humid/temp sensor

f2b myRH;

Completed code:

/\*

SENSOR NOTE

The Program is to get data from sensor and send to server.

Communication: Hai Nguyen

Sensor: Nam Nguyen.

version Apr/7/2011 - Hai Nguyen:

\* add nodeID

\* Combine information of three sensor H, T, L, in one packet.

version Apr/6/2011:

split float to 4 bytes and send 4 bytes to server.

\*/

#include <DHT22.h>

#include <Ports.h>

#include <RF12.h>

#include <stdlib.h>

//////////////////////////////////////////////////////////////////////

// PIR connect to analog port 1

Port PIRsensor(1);

// Ambient light sensor connect to analog port 4

Port LightSensor(4); // This pin is used as analog input from sensor (A0 in Arduino = pin A of port 1 in JeeNode)

**typedef**

**union {float asFloat;byte asBytes[4];} f2b;**

**typedef**

**union {int asInt;byte asBytes[4];} i2b;**

**i2b myLight; // value read from the light sensor**

**f2b myTemp; // value read from the Humid/temp sensor**

**f2b myRH;**

int myPIR;

char pirData[1];

int counter;

int occupacy;

int stop\_counting;

**byte NodeID =3;**

///////////////////////////////////////////////////////////////////////////////

void read\_PIR\_sensor()

{

myPIR = PIRsensor.anaRead();

if(myPIR > 0)

{

pirData[0] = 'Y';

occupacy = 1; //indicate that there is an occupacy in the room

counter = 0; // reset counter

stop\_counting = 0; // allow counting

}

else

{

pirData[0] = 'N'; // no motion

if(stop\_counting == 0) // if allowed to count

counter = counter + 1; // increase the counter by 1

}

if(counter == 50) // if counter = 1000 or after 1000 times 'N' in a row

{

occupacy = 0; // indicate that there is no occupacy in the room

counter = 0; // reset counter

stop\_counting = 1; //stop counting until there is a 'Y' again

}

Serial.print(" \n exit PIR function");

Serial.print(pirData[0]);

Serial.print("-counter = ");

Serial.print(counter);

delay(100); //delay 0.1s each time -> 1000 times = 100s

}

///////////////////////////////////////////////////////////////////////////////

void read\_other\_sensors()

{

// Humidity/ Temp sensor connect to port 3 JeeNode(pin 6 arduino)

#define DHT22\_PIN 6

//Port DHT22\_PIN (3);

// Setup a DHT22 instance

DHT22 myDHT22(DHT22\_PIN);

int i;

DHT22\_ERROR\_t errorCode;

Serial.print(" Reading sensor... \n ");

delay(2000); //delay for the humid/temp sensor stabelized

//read the humidity/temp sensor

errorCode = myDHT22.readData();

if(errorCode == DHT\_ERROR\_NONE) //if non error occur

{

myRH.asFloat = myDHT22.getHumidity();

myTemp.asFloat = myDHT22.getTemperatureC();

}

else //if error occur, data is all zero

{

myRH.asFloat = 0;

myTemp.asFloat = 0;

}

// read the ambient light sensor value

myLight.asInt = LightSensor.anaRead();

// itoa(myLight, lightData, 10); //convert int to char arry

}

////////////////////////////////////////////////////////////////////////////////////

byte ReceiveData[60];

byte Data[60];

byte i;

byte LengthOfMap=255;

//byte NodeID = 3; // Node ID of this node

//-----------

void setup ()

{

Serial.begin(9600);

rf12\_initialize(NodeID, RF12\_915MHZ, 212);

Serial.print("This is node:"); Serial.print(NodeID,10); Serial.print(". \n");

counter = 0;

occupacy = 1;

stop\_counting = 0;

} // end of setup()

int check\_status()

{

int temp=0;

Serial.print("Checking node status... \n");

if (ReceiveData[0]!=0x01)

{

ReceiveData[0]=ReceiveData[0]-1;

for (i = 1; i < rf12\_len-1; ++i)

ReceiveData[i]=ReceiveData[i+1];

temp= 1; //repeater

}

return temp;

}

//--------

void send\_Data()

{

Serial.print("Sending... \n");

if (LengthOfMap==255)

{

Serial.print("No information of Map to Server \n");

return;

}

rf12\_recvDone();

if (rf12\_canSend())

{

rf12\_sendStart(RF12\_HDR\_DST |Data[1],Data , sizeof Data);

}

}

//--------

void show\_data(byte \*m)

{

byte i;

for (i = 0; i<40 ; ++i)

{

Serial.print(m[i],10);

Serial.print(" ");

}

Serial.print("\n");

}

//---------

void loop ()

{

Data[59]= 0xFF;

// Reading PIR sensor

/\* read\_PIR\_sensor(); //read PIR sensor to determine if there is an occupacy or not

// if there is no occupacy -> send a 'O' to server

if(occupacy == 0)

{

i = LengthOfMap;

Data[i+3] = byte('O');

Data[i+4] = pirData[0]; // information of Occupancy put on here

Data[i+5] = 255;

// send data

send\_Data(); // send data

Serial.print(" Sent Information of Occupancy \n ");

occupacy = 1; // only send one time

}\*/

if (rf12\_recvDone() && rf12\_crc == 0) // waiting for recieving data

{

Serial.print(" ---=== Begin ===--- \n");

for (i = 0; i < rf12\_len; ++i)

ReceiveData[i]=rf12\_data[i];

Serial.print(" ReceiveData: ");

show\_data(ReceiveData);

if (check\_status()==1) //this is a repeater

{

Serial.print("Node is Repeater \n");

rf12\_recvDone();

if (rf12\_canSend())

{

rf12\_sendStart(RF12\_HDR\_DST | ReceiveData[1], ReceiveData , sizeof ReceiveData);

}

Serial.print(" Data sent: ");

show\_data(ReceiveData);

Serial.print(" ---=== End ===--- \n");

}

else //this is the final destination

{

Serial.print("Node is Destination \n");

switch (ReceiveData[2])

{

case byte('R'):

{

Serial.print(" Recieved Request: \n ");

// Reading sensor value function.

read\_other\_sensors();

i = Data[0];

Data[i+1] = byte('D');

Data[i+2] = 1;

Data[i+10] = 0xFF;

switch (ReceiveData[3]) // Type of Request

{

**case byte('A'):**

**{**

**Serial.print(" Recieved Request Information of Humidity & Temperature & Light: \n ");**

**Data[i+3] = byte('H');**

**Data[i+4] = myRH.asBytes[0]; // information of Humidity put on here**

**Data[i+5] = myRH.asBytes[1]; // information of Humidity put on here**

**Data[i+6] = myRH.asBytes[2]; // information of Humidity put on here**

**Data[i+7] = myRH.asBytes[3]; // information of Humidity put on here**

**Data[i+8] = byte('T');**

**Data[i+9] = myTemp.asBytes[0]; // information of Temperature put on here**

**Data[i+10] = myTemp.asBytes[1]; // information of Temperature put on here**

**Data[i+11] = myTemp.asBytes[2]; // information of Temperature put on here**

**Data[i+12] = myTemp.asBytes[3]; // information of Temperature put on here**

**Data[i+13] = byte('L'); // lightData[4]**

**Data[i+14] = myLight.asBytes[0]; // information of Light put on here**

**Data[i+15] = myLight.asBytes[1]; // information of Light put on here**

**Data[i+16] = myLight.asBytes[2]; // information of Light put on here**

**Data[i+17] = myLight.asBytes[3]; // information of Light put on here**

**// send data**

**send\_Data();**

**Serial.print(" Sent Request Information of Light \n ");**

**break;**

**} // end of case byte ("L")**

case byte('O'):

{

Data[i+3] = byte('O');

Data[i+4] = pirData[0]; // information of Occupancy put on here

Data[i+5] = 255;

// send data

send\_Data();

Serial.print(" Sent Request Information of Occupancy \n ");

break;

} // end of case byte ("O")

case byte('W'):

{

Serial.print(" Recieved Request Information of Waterlevel \n ");

Data[i+3] = byte('W');

Data[i+4] = 105; // information of Waterlevel put on here

send\_Data();

break;

} // end of case byte ("W")

default:

{

Serial.print("Recieved BAD REQUEST. Don't send anything \n");

}// end of default

;

}// end switch (ReceiveData[3])

break;

} // end of case byte (R)

case byte('D'):

{

Serial.print(" Recieved Data: \n");

break;

} // end of case byte("D")

case byte('M'): //if Data, store the return direction into Data[]

{

LengthOfMap = ReceiveData[3];

Serial.print("Map information: ");

for (i = 0; i < LengthOfMap + 1; ++i) // assume that just have 3 node

{

//if (ReceiveData[3+i]==0xFF); //if the end character, exit for

Data[i]=ReceiveData[3+i];// bytes after 'M'

Serial.print(Data[i],10);Serial.print(" ");

}

Serial.print(" \n ");

break;

} // end of case byte("M")

default:

Serial.print("Recieved BAD Data \n");

;

// if nothing else matches, do the default

// default is optional

} // end switch (ReceiveData[2])

Serial.print(" Data sent: ");

show\_data(Data);

Serial.print(" ---=== End ===--- \n");

} //end else

} //end if

} //end void

### Date 04/15/2011

Work with Nam in Occupancy code.

### Date 04/17/2011

Work with Nam in Occupancy code.

1. Change in sending Occupancy code. When Occupancy information is sent, the node will check the acknowledgment from server after each 10s. If the acknowledgment is not received, the node send the Occupancy information again.

2. Add the end of package: add NodeID after information packet:

Data[i+18] = byte ('I');

Data[i+19] = NodeID;

Data[i+20] = 255;

3. Add the report bad data

4. Add acknowledgment of Map

5. Add receive Acknowledgment Action.

**Algorithm**

1. Variable:
   * 1. Data{}: store the information will send
        1. Route
        2. Action
        3. Data
     2. ReceiveData[60]: received massage from RF
     3. myLight: light information
     4. myTemp: temperature information
     5. myRH: relative humidity information
     6. NodeID
     7. LengthOfMap
2. read\_PIR\_sensor()
3. read\_other\_sensors()
4. Setup()
   * 1. Set up the serial, to send information on screen of Serial Monitor ( of Arduino)
     2. Initalize the RF12: NodeID =2 *(it can be change)*, 915Mhz, group ID 212
5. Check\_status(): check the status of node is repeater or destination
   * 1. Check byte0 of *ReceiveData{}*
        1. Byte0 =1 🡪 node is repeater
           1. Delete the current node ID:

for (i = 1; i < rf12\_len-1; ++i)

ReceiveData[i]=ReceiveData[i+1];

* + - 1. Byte0 <>1 -> node is not repeater

1. Send\_Data(): Send data to next node.
2. **Loop()**
3. Check: Occupacy
4. Check: Receive Rf data
5. Check if node repeater than send the information to next node.
6. If node is destination :
   * + - 1. Choose action in received data :

‘R’: request 🡪 respond by sending data

Choosing the request

**“A” : send information of H,T,L**

**Else: report BAD data.**

‘D’: data

‘M’: maps

Store the map to the data{}

Send acknowledgment of map

‘A’: acknowledgment

“O”: acknowledgment of server about occupancy

Else: report bad data

/\*

SENSOR NOTE

The Program is to get data from sensor and send to server.

Communication: Hai Nguyen

Sensor: Nam Nguyen.

Version Apr/15/2011

add ACK:

After 10s, check ACK for PIR after sending

-> if server received -> stop checking ack

send ACK to server after received Map from server

Version Apr/9/2011 - Hai Nguyen

add nodeID in the data packet

version Apr/7/2011 - Hai Nguyen:

\* add nodeID

\* Combine information of there sensor H, T, L, in one packet.

\* Timer in reading PIR, not use delay.

version Apr/6/2011:

split float to 4 bytes and send 4 bytes to server.

\*/

#include <DHT22.h>

#include <Ports.h>

#include <RF12.h>

#include <stdlib.h>

#include <TimerOne.h>

//////////////////////////////////////////////////////////////////////

byte NodeID = 6;

/////////////////////////////////////////////////////////////////////

// PIR connect to analog port 1

Port PIRsensor(1);

// Ambient light sensor connect to analog port 4

Port LightSensor(4); // This pin is used as analog input from sensor (A0 in Arduino = pin A of port 1 in JeeNode)

typedef

union {float asFloat;byte asBytes[4];} f2b;

typedef

union {int asInt;byte asBytes[4];} i2b;

i2b myLight; // value read from the light sensor

f2b myTemp; // value read from the Humid/temp sensor

f2b myRH;

int myPIR;

char pirData[1];

char last\_pirData[1];

byte sent\_PIR =0;

byte received\_PIR =0;

int counter;

int occupacy;

int NotOccupacy;

int stop\_counting;

int first\_time;

int minute = 1;

//int start\_up = 1; //not sending PIR data the first time

int timer\_counter;

///////////////////////////////////////////////////////////////////////////////

void read\_PIR\_sensor()

{

myPIR = PIRsensor.anaRead();

if(myPIR > 0)

{

pirData[0] = 'Y';

if (first\_time) //if this is the first Y after N

occupacy = 1; //indicate that there is a NEW motion in the room

counter = 0; // reset counter

stop\_counting = 0; // allow counting

}

else

{

pirData[0] = 'N'; // no motion

if(stop\_counting == 0) // if allowed to count

counter = counter + 1; // increase the counter by 1

}

if(counter == (minute\*120)) // if counter = minute\*120 or 'N' after 'minute'

{

NotOccupacy = 1; // indicate that there is no occupacy in the room

counter = 0; // reset counter

stop\_counting = 1; //stop counting until there is a 'Y' again

}

timer\_counter = timer\_counter + 1; //counter for calling the check\_PIR\_sent()

/\*

Serial.print("PIR:");

Serial.print(pirData[0]);

Serial.print("counter = ");

Serial.print(counter);

Serial.print(" \n ");

//delay(100); //delay 0.1s each time -> 1000 times = 100s\*/

}

///////////////////////////////////////////////////////////////////////////////

void read\_other\_sensors()

{

// Humidity/ Temp sensor connect to port 3 JeeNode(pin 6 arduino)

#define DHT22\_PIN 6

//Port DHT22\_PIN (3);

// Setup a DHT22 instance

DHT22 myDHT22(DHT22\_PIN);

int i;

DHT22\_ERROR\_t errorCode;

Serial.print(" Reading sensor... \n ");

delay(2000); //delay for the humid/temp sensor stabelized

//read the humidity/temp sensor

errorCode = myDHT22.readData();

if(errorCode == DHT\_ERROR\_NONE) //if non error occur

{

myRH.asFloat = myDHT22.getHumidity();

myTemp.asFloat = myDHT22.getTemperatureC();

}

else //if error occur, data is all zero

{

myRH.asFloat = 0.00;

myTemp.asFloat = 0.00;

}

// read the ambient light sensor value

myLight.asInt = LightSensor.anaRead();

}

////////////////////////////////////////////////////////////////////////////////////

byte ReceiveData[60];

byte Data[60];

byte i;

byte LengthOfMap=255;

//byte NodeID = 3; // Node ID of this node

//-----------

void setup ()

{

Serial.begin(9600);

rf12\_initialize(NodeID, RF12\_915MHZ, 212);

Serial.print("Version 11 \n");

Serial.print("This is node:"); Serial.print(NodeID,10); Serial.print(". \n");

// Initialize timer

Timer1.initialize(500000); // read the PIR after 0.5s

Timer1.attachInterrupt(read\_PIR\_sensor);

counter = 0;

occupacy = 1;

stop\_counting = 0;

// start\_up = 1;

} // end of setup()

int check\_status()

{

int temp=0;

Serial.print("Checking node status... \n");

if (ReceiveData[0]!=0x01)

{

ReceiveData[0]=ReceiveData[0]-1;

for (i = 1; i < rf12\_len-1; ++i)

ReceiveData[i]=ReceiveData[i+1];

temp= 1; //repeater

}

return temp;

}

//--------

void send\_Data()

{

Serial.print("Sending... \n");

if (LengthOfMap==255)

{

Serial.print("No information of Map to Server \n");

return;

}

rf12\_recvDone();

if (rf12\_canSend())

{

rf12\_sendStart(RF12\_HDR\_DST |Data[1],Data , sizeof Data);

}

Serial.print(" Data sent: ");

show\_data(Data);

}

//--------

void show\_data(byte \*m)

{

byte i;

for (i = 0; i<40 ; ++i)

{

Serial.print(m[i],10);

Serial.print(" ");

}

Serial.print("\n");

}

//----------------------------------------------------------

void send\_PIR\_data(char PIR\_data)

{

int i = LengthOfMap;

Data[i+1] = byte('D');

Data[i+2] = 1;

Data[i+3] = byte('O');

Data[i+4] = PIR\_data; // information of Occupancy put on here

Data[i+5] = byte('I');

Data[i+6] = NodeID;

Data[i+7] = 255;

// send data

Serial.print(" Sent Information of Occupancy \n ");

send\_Data(); // send data

}

//---------------------------------------------------------

void check\_PIR\_sent()

{

if(received\_PIR == 0) //if server haven't received the PIR data -> resend

{

Serial.print(" Don't receive ACK. send PIR again. \n ");

send\_PIR\_data(last\_pirData[0]); //re-send PIR data

/////////////////////////////////////////////

//sent\_PIR = 0; //reset for TESTING ONLY

//////////////////////////////////////////////

}

else //server successfully received data -> reset flags

{

sent\_PIR = 0;

received\_PIR = 0;

}

}

//----------------------------------------------------------

void loop ()

{

Data[59]= 0xFF;

if(timer\_counter == 20) //checking after 10s

{

Serial.print(sent\_PIR,10);

if(sent\_PIR)

check\_PIR\_sent(); //checking PIR data ack from server

timer\_counter = 0; //reset counter

}

// check occupacy

if(occupacy || NotOccupacy)

{

send\_PIR\_data(pirData[0]);

last\_pirData[0] = pirData[0];

if(NotOccupacy)

first\_time = 1; // the next Y will be the first Y

else

first\_time = 0; //only send Y one time until NotOccupacy=1

NotOccupacy = 0; //only send N one time

occupacy = 0; //only send Y one time

sent\_PIR = 1; //sending PIR data

received\_PIR = 0; //haven't received ack from server

timer\_counter = 0; //reset counter

}

if (rf12\_recvDone() && rf12\_crc == 0) // waiting for recieving data

{

Serial.print(" ---=== Begin ===--- \n");

for (i = 0; i < rf12\_len; ++i)

ReceiveData[i]=rf12\_data[i];

Serial.print(" ReceiveData: ");

show\_data(ReceiveData);

if (check\_status()==1) //this is a repeater

{

Serial.print("Node is Repeater \n");

rf12\_recvDone();

if (rf12\_canSend())

{

rf12\_sendStart(RF12\_HDR\_DST | ReceiveData[1], ReceiveData , sizeof ReceiveData);

}

Serial.print(" \n Data sent: ");

show\_data(ReceiveData);

Serial.print(" ---=== End ===--- \n");

}

else //this is the final destination

{

Serial.print("Node is Destination \n");

switch (ReceiveData[2])

{

case byte('R'):

{

Serial.print(" Recieved Request: \n ");

i = Data[0];

Data[i+1] = byte('D');

if (ReceiveData[3] == byte('A')) // Type of Request

{

Serial.print(" Recieved Request Information of Humidity & Temperature & Light: \n ");

// Read other sensors.

read\_other\_sensors();

Data[i+2] = 3;

Data[i+3] = byte('H');

Data[i+4] = myRH.asBytes[0]; // information of Humidity put on here

Data[i+5] = myRH.asBytes[1]; // information of Humidity put on here

Data[i+6] = myRH.asBytes[2]; // information of Humidity put on here

Data[i+7] = myRH.asBytes[3]; // information of Humidity put on here

Data[i+8] = byte('T');

Data[i+9] = myTemp.asBytes[0]; // information of Temperature put on here

Data[i+10] = myTemp.asBytes[1]; // information of Temperature put on here

Data[i+11] = myTemp.asBytes[2]; // information of Temperature put on here

Data[i+12] = myTemp.asBytes[3]; // information of Temperature put on here

Data[i+13] = byte('L'); // lightData[4]

Data[i+14] = myLight.asBytes[0]; // information of Light put on here

Data[i+15] = myLight.asBytes[1]; // information of Light put on here

Data[i+16] = myLight.asBytes[2]; // information of Light put on here

Data[i+17] = myLight.asBytes[3]; // information of Light put on here

Data[i+18] = byte ('I');

Data[i+19] = NodeID;

Data[i+20] = 255;

// send data

send\_Data();

} // end of case byte ("L")

else

{

Data[i+2] = 0;

Data[i+3] = byte('I');

Data[i+4] = NodeID;

Data[i+5] = 255;

Serial.print("Recieved BAD REQUEST. \n");

send\_Data();

}// end of default

// end switch (ReceiveData[3])

} // end of case byte (R)

case byte('D'):

{

Serial.print(" Recieved Data: \n");

break;

} // end of case byte("D")

case byte('M'): //if Data, store the return direction into Data[]

{

LengthOfMap = ReceiveData[3];

Serial.print("Map information: ");

for (i = 0; i < LengthOfMap + 1; ++i) // assume that just have 3 node

{

//if (ReceiveData[3+i]==0xFF); //if the end character, exit for

Data[i]=ReceiveData[3+i];// bytes after 'M'

Serial.print(Data[i],10);Serial.print(" ");

}

Serial.print(" \n ");

Serial.print("Send ACK: ");

i = Data[0];

Data[i+1] = byte('A');

Data[i+2] = byte('M');

Data[i+3] = byte('I');

Data[i+4] = NodeID;

Data[i+5] = 255;

send\_Data();

break;

} // end of case byte("M")

case byte('S'): //if Data, store the return direction into Data[]

{

Serial.print("Set up information for occupacy.");

minute = ReceiveData[3];

Serial.print("Time before deciding that there is no occupacy. Minute:");Serial.print(minute,DEC);Serial.print(" \n");

break;

} // end of case byte("M")

case byte('A'): //if Data, store the return direction into Data[]

{

Serial.print("Recieved ACK: ");

if (ReceiveData[3] == byte('O'))

{

Serial.print("Occupacy \n ");

received\_PIR = 1;

}

break;

} // end of case byte("A")

default:

{

Data[i+1] = byte('B');

Data[i+2] = byte('I');

Data[i+3] = NodeID;

Data[i+4] = 255;

send\_Data();

Serial.print("Recieved BAD Data \n");

}

;

// if nothing else matches, do the default

// default is optional

} // end switch (ReceiveData[2])

Serial.print(" ---=== End ===--- \n");

} //end else

} //end if

} //end void

### Date 04/26/2011

Add store Map to EEPROM and read Map from EEPROM

/\*

SENSOR NOTE

The Program is to get data from sensor and send to server.

Communication: Hai Nguyen

Sensor: Nam Nguyen.

Version apr/26:

add: + write data of map to EEPROM.

+ update the map to EEPROM.

Version Apr/15/2011

add ACK:

After 10s, check ACK for PIR after sending

-> if server received -> stop checking ack

send ACK to server after received Map from server

Version Apr/9/2011 - Hai Nguyen

add nodeID in the data packet

version Apr/7/2011 - Hai Nguyen:

\* add nodeID

\* Combine information of there sensor H, T, L, in one packet.

\* Timer in reading PIR, not use delay.

version Apr/6/2011:

split float to 4 bytes and send 4 bytes to server.

\*/

#include <DHT22.h>

#include <Ports.h>

#include <RF12.h>

#include <stdlib.h>

#include <TimerOne.h>

#include <EEPROM.h>

//////////////////////////////////////////////////////////////////////

byte NodeID = 15;

/////////////////////////////////////////////////////////////////////

// PIR connect to analog port 1

Port PIRsensor(1);

// Ambient light sensor connect to analog port 4

Port LightSensor(4); // This pin is used as analog input from sensor (A0 in Arduino = pin A of port 1 in JeeNode)

typedef

union {float asFloat;byte asBytes[4];} f2b;

typedef

union {int asInt;byte asBytes[4];} i2b;

i2b myLight; // value read from the light sensor

f2b myTemp; // value read from the Humid/temp sensor

f2b myRH;

int myPIR;

char pirData[1];

char last\_pirData[1];

byte sent\_PIR =0;

byte received\_PIR =0;

int counter;

int occupacy;

int NotOccupacy;

int stop\_counting;

int first\_time;

int minute = 1;

//int start\_up = 1; //not sending PIR data the first time

int timer\_counter;

byte loaded\_map=0;

///////////////////////////////////////////////////////////////////////////////

void read\_PIR\_sensor()

{

myPIR = PIRsensor.anaRead();

if(myPIR > 0)

{

pirData[0] = 'Y';

if (first\_time) //if this is the first Y after N

occupacy = 1; //indicate that there is a NEW motion in the room

counter = 0; // reset counter

stop\_counting = 0; // allow counting

}

else

{

pirData[0] = 'N'; // no motion

if(stop\_counting == 0) // if allowed to count

counter = counter + 1; // increase the counter by 1

}

if(counter == (minute\*120)) // if counter = minute\*120 or 'N' after 'minute'

{

NotOccupacy = 1; // indicate that there is no occupacy in the room

counter = 0; // reset counter

stop\_counting = 1; //stop counting until there is a 'Y' again

}

timer\_counter = timer\_counter + 1; //counter for calling the check\_PIR\_sent()

Serial.print("PIR:");

Serial.print(pirData[0]);

Serial.print("counter = ");

Serial.print(counter);

Serial.print(" \n ");

//delay(100); //delay 0.1s each time -> 1000 times = 100s\*/

}

///////////////////////////////////////////////////////////////////////////////

void read\_other\_sensors()

{

// Humidity/ Temp sensor connect to port 3 JeeNode(pin 6 arduino)

#define DHT22\_PIN 6

//Port DHT22\_PIN (3);

// Setup a DHT22 instance

DHT22 myDHT22(DHT22\_PIN);

int i;

DHT22\_ERROR\_t errorCode;

Serial.print(" Reading sensor... \n ");

delay(2000); //delay for the humid/temp sensor stabelized

//read the humidity/temp sensor

errorCode = myDHT22.readData();

if(errorCode == DHT\_ERROR\_NONE) //if non error occur

{

myRH.asFloat = myDHT22.getHumidity();

myTemp.asFloat = myDHT22.getTemperatureC();

}

else //if error occur, data is all zero

{

myRH.asFloat = 0.00;

myTemp.asFloat = 0.00;

}

// read the ambient light sensor value

myLight.asInt = LightSensor.anaRead();

}

////////////////////////////////////////////////////////////////////////////////////

byte ReceiveData[60];

byte Data[30];

byte i;

byte LengthOfMap=255;

//byte NodeID = 3; // Node ID of this node

//-----------

void setup ()

{

Serial.begin(9600);

rf12\_initialize(NodeID, RF12\_915MHZ, 212);

Serial.print("Version 13.");

Serial.print("This is node:"); Serial.print(NodeID,10); Serial.print(". \n");

// Initialize timer

Timer1.initialize(500000); // read the PIR after 0.5s

Timer1.attachInterrupt(read\_PIR\_sensor);

counter = 0;

occupacy = 1;

stop\_counting = 0;

// start\_up = 1;

} // end of setup()

int check\_status()

{

int temp=0;

Serial.print("Checking node status... \n");

if (ReceiveData[0]!=0x01)

{

ReceiveData[0]=ReceiveData[0]-1;

for (i = 1; i < rf12\_len-1; ++i)

ReceiveData[i]=ReceiveData[i+1];

temp= 1; //repeater

}

return temp;

}

//--------SEND DATA----------

void send\_Data()

{

Serial.print("Sending... \n");

if (LengthOfMap==255)

{

Serial.print("No information of Map to Server \n");

return;

}

rf12\_recvDone();

if (rf12\_canSend())

{

rf12\_sendStart(RF12\_HDR\_DST |Data[1],Data , sizeof Data);

}

Serial.print(" Data sent: ");

show\_data(Data);

}

//--------SHOW DATA to serial port-------------

void show\_data(byte \*m)

{

byte i;

for (i = 0; i<29 ; ++i)

{

Serial.print(m[i],10); Serial.print(" ");

}

Serial.print("\n");

}

//--------------------SEND PIR DATA-------------------------------

void send\_PIR\_data(char PIR\_data)

{

int i = LengthOfMap;

Data[i+1] = byte('D');

Data[i+2] = 1;

Data[i+3] = byte('O');

Data[i+4] = PIR\_data; // information of Occupancy put on here

Data[i+5] = byte('I');

Data[i+6] = NodeID;

Data[i+7] = 255;

// send data

Serial.print(" Sent Information of Occupancy \n ");

send\_Data(); // send data

}

//---------------------------------------------------------

void check\_PIR\_sent()

{

if(received\_PIR == 0) //if server haven't received the PIR data -> resend

{

Serial.print(" Don't receive ACK. send PIR again. \n ");

send\_PIR\_data(last\_pirData[0]); //re-send PIR data

/////////////////////////////////////////////

//sent\_PIR = 0; //reset for TESTING ONLY

//////////////////////////////////////////////

}

else //server successfully received data -> reset flags

{

sent\_PIR = 0;

received\_PIR = 0;

}

}

**//-----------------------Update map to eeprom---------------------- //**

**void update\_map()**

**{**

**byte i;**

**EEPROM.write(0,LengthOfMap);**

**Serial.println("Update map information to EEPROM... ");**

**for (i = 0; i < LengthOfMap + 1; ++i) // assume that just have 3 node**

**{**

**EEPROM.write(i,Data[i]);**

**}**

**}**

**//----------------------------------------------------------**

**//----------------------Load map from eeprom-------------------**

**void load\_map()**

**{**

**byte i;**

**loaded\_map=1;**

**Data[0]=EEPROM.read(0);**

**LengthOfMap = Data[0];**

**Serial.println("Reading map information to EEPROM... ");**

**Serial.print(Data[i],10);Serial.print(" ");**

**if (LengthOfMap==255)**

**{**

**Serial.println("No map information is stored ");**

**return;**

**}**

**Serial.print("Map information:");**

**for (i = 0; i < LengthOfMap + 1; ++i) // assume that just have 3 node**

**{**

**Data[i] = EEPROM.read(i);**

**Serial.print(Data[i],10);Serial.print(" ");**

**}**

**Serial.print("\n ");**

**}**

void loop ()

{

**if (loaded\_map ==0) load\_map(); // load map from eeprom**

**Data[29]= 0xFF;**

if(timer\_counter == 20) //checking after 10s

{

Serial.print(sent\_PIR,10);

if(sent\_PIR) check\_PIR\_sent(); //checking PIR data ack from server

timer\_counter = 0; //reset counter

}

// check occupacy

if(occupacy || NotOccupacy)

{

send\_PIR\_data(pirData[0]);

last\_pirData[0] = pirData[0];

if(NotOccupacy)

first\_time = 1; // the next Y will be the first Y

else

first\_time = 0; //only send Y one time until NotOccupacy=1

NotOccupacy = 0; //only send N one time

occupacy = 0; //only send Y one time

sent\_PIR = 1; //sending PIR data

received\_PIR = 0; //haven't received ack from server

timer\_counter = 0; //reset counter

}

if (rf12\_recvDone() && rf12\_crc == 0) // waiting for recieving data

{

Serial.print(" ---=== Begin ===--- \n");

for (i = 0; i < rf12\_len; ++i)

ReceiveData[i]=rf12\_data[i];

Serial.print(" ReceiveData: ");

show\_data(ReceiveData);

if (check\_status()==1) //this is a repeater

{

Serial.print("Node is Repeater \n");

rf12\_recvDone();

if (rf12\_canSend())

{

rf12\_sendStart(RF12\_HDR\_DST | ReceiveData[1], ReceiveData , sizeof ReceiveData);

}

Serial.print(" \n Data sent: ");

show\_data(ReceiveData);

Serial.print(" ---=== End ===--- \n");

}

else //this is the final destination

{

Serial.print("Node is Destination \n");

switch (ReceiveData[2])

{

case byte('R'):

{

Serial.print(" Recieved Request: \n ");

i = Data[0];

Data[i+1] = byte('D');

if (ReceiveData[3] == byte('A')) // Type of Request

{

Serial.print(" Recieved Request Information of Humidity & Temperature & Light: \n ");

// Read other sensors.

read\_other\_sensors();

Data[i+2] = 3;

Data[i+3] = byte('H');

Data[i+4] = myRH.asBytes[0]; // information of Humidity put on here

Data[i+5] = myRH.asBytes[1]; // information of Humidity put on here

Data[i+6] = myRH.asBytes[2]; // information of Humidity put on here

Data[i+7] = myRH.asBytes[3]; // information of Humidity put on here

Data[i+8] = byte('T');

Data[i+9] = myTemp.asBytes[0]; // information of Temperature put on here

Data[i+10] = myTemp.asBytes[1]; // information of Temperature put on here

Data[i+11] = myTemp.asBytes[2]; // information of Temperature put on here

Data[i+12] = myTemp.asBytes[3]; // information of Temperature put on here

Data[i+13] = byte('L'); // lightData[4]

Data[i+14] = myLight.asBytes[0]; // information of Light put on here

Data[i+15] = myLight.asBytes[1]; // information of Light put on here

Data[i+16] = myLight.asBytes[2]; // information of Light put on here

Data[i+17] = myLight.asBytes[3]; // information of Light put on here

Data[i+18] = byte ('I');

Data[i+19] = NodeID;

Data[i+20] = 255;

// send data

send\_Data();

} // end of case byte ("L")

else

{

Data[i+2] = 0;

Data[i+3] = byte('I');

Data[i+4] = NodeID;

Data[i+5] = 255;

Serial.print("Recieved BAD REQUEST. \n");

send\_Data();

}// end of default

// end switch (ReceiveData[3])

} // end of case byte (R)

case byte('D'):

{

Serial.print(" Recieved Data: \n");

break;

} // end of case byte("D")

case byte('M'): //if Data, store the return direction into Data[]

{

LengthOfMap = ReceiveData[3];

Serial.print("Map information: ");

for (i = 0; i < LengthOfMap + 1; ++i) // assume that just have 3 node

{

//if (ReceiveData[3+i]==0xFF); //if the end character, exit for

Data[i]=ReceiveData[3+i];// bytes after 'M'

Serial.print(Data[i],10);Serial.print(" ");

}

Serial.print(" \n ");

update\_map(); // update map to eeprom;

Serial.print("Send ACK: ");

i = Data[0];

Data[i+1] = byte('A');

Data[i+2] = byte('M');

Data[i+3] = byte('I');

Data[i+4] = NodeID;

Data[i+5] = 255;

send\_Data();

break;

} // end of case byte("M")

case byte('S'): //if Data, store the return direction into Data[]

{

Serial.print("Set up information for occupacy.");

minute = ReceiveData[3];

Serial.print("Time before deciding that there is no occupacy. Minute:");Serial.print(minute,DEC);Serial.print(" \n");

break;

} // end of case byte("M")

case byte('A'): //if Data, store the return direction into Data[]

{

Serial.print("Recieved ACK: ");

if (ReceiveData[3] == byte('O'))

{

Serial.print("Occupacy \n ");

received\_PIR = 1;

}

break;

} // end of case byte("A")

default:

{

Data[i+1] = byte('B');

Data[i+2] = byte('I');

Data[i+3] = NodeID;

Data[i+4] = 255;

send\_Data();

Serial.print("Recieved BAD Data \n");

}

;

// if nothing else matches, do the default

// default is optional

} // end switch (ReceiveData[2])

Serial.print(" ---=== End ===--- \n");

} //end else

} //end if

} //end void

## Working with Ping sensor

### Date 04/26/2011

/\*

SENSOR NOTE

The Program is to get data from sensor and send to server.

Communication: Hai Nguyen

Sensor: Nam Nguyen.

ver 4/21 Add water level sensor - Hai Nguyen

\*/

#include <DHT22.h>

#include <Ports.h>

#include <RF12.h>

#include <stdlib.h>

#include <TimerOne.h>

#include <EEPROM.h>

//////////////////////////////////////////////////////////////////////

byte NodeID = 12;

/////////////////////////////////////////////////////////////////////

// PIR connect to analog port 1

Port PIRsensor(1);

// Ambient light sensor connect to analog port 4

Port LightSensor(4); // This pin is used as analog input from sensor (A0 in Arduino = pin A of port 1 in JeeNode)

typedef

union {float asFloat;byte asBytes[4];} f2b;

typedef

union {int asInt;byte asBytes[4];} i2b;

int duration;

i2b inches, cm; // use for ping

int pingPin=4;

//-----------------Communication variable-------------------------

byte loaded\_map=0; // check load map from EEprom.

byte ReceiveData[60];

byte Data[30];

byte i;

byte LengthOfMap=255;

//byte NodeID = 3; // Node ID of this node

//-----------

void setup ()

{

Serial.begin(9600);

rf12\_initialize(NodeID, RF12\_915MHZ, 212);

Serial.print("Version 3 for Ping- Water level sensor.");

Serial.print("This is node:"); Serial.print(NodeID,10); Serial.print(". \n");

} // end of setup()

void loop ()

{

if (loaded\_map ==0) load\_map(); // load map from eeprom

if (rf12\_recvDone() && rf12\_crc == 0) // waiting for recieving data

{

Serial.print(" ---=== Begin ===--- \n");

for (i = 0; i < rf12\_len; ++i)

ReceiveData[i]=rf12\_data[i];

Serial.print(" ReceiveData: ");

show\_data(ReceiveData);

if (check\_status()==1) //this is a repeater

{

Repeater();

}

else //this is the final destination

{

Serial.print("Node is Destination \n");

switch (ReceiveData[2])

{

case byte('R'):

{

Serial.print(" Recieved Request: \n ");

i = Data[0];

Data[i+1] = byte('D');

if (ReceiveData[3] == byte('W')) // Type of Request

{

Serial.print("Recieved Request Information of Water level sensor: \n");

ping();

Data[i+2] = 1;

Data[i+3] = byte('W');

Data[i+4] = cm.asBytes[0]; // information of Humidity put on here

Data[i+5] = cm.asBytes[1]; // information of Humidity put on here

Data[i+6] = cm.asBytes[2]; // information of Humidity put on here

Data[i+7] = cm.asBytes[3]; // information of Humidity put on here

Data[i+8] = byte ('I');

Data[i+9] = NodeID;

Data[i+10] = 255;

send\_Data();

}

else

{

Data[i+2] = 0;

Data[i+3] = byte('I');

Data[i+4] = NodeID;

Data[i+5] = 255;

Serial.print("Recieved BAD REQUEST. \n");

send\_Data();

}

// end switch (ReceiveData[3])

break;

} // end of case byte (R)

case byte('M'): //if Data, store the return direction into Data[]

{

LengthOfMap = ReceiveData[3];

Serial.print("Map information: ");

for (i = 0; i < LengthOfMap + 1; ++i) // assume that just have 3 node

{

//if (ReceiveData[3+i]==0xFF); //if the end character, exit for

Data[i]=ReceiveData[3+i];// bytes after 'M'

Serial.print(Data[i],10);Serial.print(" ");

}

Serial.print(" \n ");

update\_map(); // update map to eeprom;

Serial.print("Send ACK: ");

i = Data[0];

Data[i+1] = byte('A');

Data[i+2] = byte('M');

Data[i+3] = byte('I');

Data[i+4] = NodeID;

Data[i+5] = 255;

send\_Data();

break;

} // end of case byte("M")

case byte('A'):

{

} // end of case byte("A")

default:

{

Data[i+1] = byte('B');

Data[i+2] = byte('I');

Data[i+3] = NodeID;

Data[i+4] = 255;

send\_Data();

Serial.print("Recieved BAD Data \n");

};

Serial.print(" ---=== End ===--- \n");

} //end else

} //end if

// ---------- Check status the node is repeater or destination

int check\_status()

{

int temp=0;

Serial.print("Checking node status... \n");

if (ReceiveData[0]!=0x01)

{

ReceiveData[0]=ReceiveData[0]-1;

for (i = 1; i < rf12\_len-1; ++i)

ReceiveData[i]=ReceiveData[i+1];

temp= 1; //repeater

}

return temp;

}

//--------SEND DATA----------

void send\_Data()

{

Serial.print("Sending... \n");

if (LengthOfMap==255)

{

Serial.print("No information of Map to Server \n");

return;

}

rf12\_recvDone();

if (rf12\_canSend())

{

rf12\_sendStart(RF12\_HDR\_DST |Data[1],Data , sizeof Data);

}

Serial.print(" Data sent: ");

show\_data(Data);

}

//--------SHOW DATA to serial port-------------

void show\_data(uint8\_t \*m)

{

uint8\_t i;

for (i = 0; i<29 ; ++i)

{

Serial.print(m[i],10); Serial.print(" ");

}

Serial.print("\n");

}

//-----------------------Update map to eeprom---------------------- //

void update\_map()

{

byte i;

EEPROM.write(0,LengthOfMap);

Serial.println("Update map information to EEPROM... ");

for (i = 0; i < LengthOfMap + 1; ++i) // assume that just have 3 node

{

EEPROM.write(i,Data[i]);

}

}

//----------------------------------------------------------

//----------------------Load map from eeprom-------------------

void load\_map()

{

byte i;

loaded\_map=1;

Data[0]=EEPROM.read(0);

LengthOfMap = Data[0];

Serial.println("Reading map information to EEPROM... ");

Serial.print(Data[i],10);Serial.print(" ");

if (LengthOfMap==255)

{

Serial.println("No map information is stored ");

return;

}

Serial.print("Map information:");

for (i = 0; i < LengthOfMap + 1; ++i) // assume that just have 3 node

{

Data[i] = EEPROM.read(i);

Serial.print(Data[i],10);Serial.print(" ");

}

Serial.print("\n ");

}

void Repeater()

{

Serial.print("Node is Repeater \n");

rf12\_recvDone();

if (rf12\_canSend())

{

rf12\_sendStart(RF12\_HDR\_DST | ReceiveData[1], ReceiveData , sizeof ReceiveData);

}

Serial.print(" \n Data sent: ");

show\_data(ReceiveData);

Serial.print(" ---=== End ===--- \n");

}

///////////////////////////////////////////////////////////////////////////////

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Water level \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ping()

{

// establish variables for duration of the ping,

// and the distance result in inches and centimeters:

// The PING))) is triggered by a HIGH pulse of 2 or more microseconds.

// Give a short LOW pulse beforehand to ensure a clean HIGH pulse:

Serial.print(" Reading PING sensor... ");

pinMode(pingPin, OUTPUT);

digitalWrite(pingPin, LOW);

delayMicroseconds(2);

digitalWrite(pingPin, HIGH);

delayMicroseconds(5);

digitalWrite(pingPin, LOW);

// The same pin is used to read the signal from the PING))): a HIGH

// pulse whose duration is the time (in microseconds) from the sending

// of the ping to the reception of its echo off of an object.

pinMode(pingPin, INPUT);

duration = pulseIn(pingPin, HIGH);

// convert the time into a distance

inches.asInt = microsecondsToInches(duration);

cm.asInt = microsecondsToCentimeters(duration);

Serial.print(cm.asInt,10);

Serial.print(" cm. \n ");

}

long microsecondsToInches(int microseconds)

{

// According to Parallax's datasheet for the PING))), there are

// 73.746 microseconds per inch (i.e. sound travels at 1130 feet per

// second). This gives the distance travelled by the ping, outbound

// and return, so we divide by 2 to get the distance of the obstacle.

// See: http://www.parallax.com/dl/docs/prod/acc/28015-PING-v1.3.pdf

return microseconds / 74 / 2;

}

long microsecondsToCentimeters(int microseconds)

{

// The speed of sound is 340 m/s or 29 microseconds per centimeter.

// The ping travels out and back, so to find the distance of the

// object we take half of the distance travelled.

return microseconds / 29 / 2;

}

} //end void

}

### Date 04/27/2011

Split the file communication of sensor to 3 file:

### Date 04/29/2011

Nam changes something in the sensor part.

/\*

SENSOR NOTE

The Program is to get data from sensor and send to server.

Communication: Hai Nguyen

Sensor: Nam Nguyen.

Version Apr/27/2011

Split into file:

+ ..ver14.pde

+ ..communication.pde

+PIR.pde

Version apr/26:

add: + write data of map to EEPROM.

+ update the map to EEPROM.

Version Apr/15/2011

add ACK:

After 10s, check ACK for PIR after sending

-> if server received -> stop checking ack

send ACK to server after received Map from server

Version Apr/9/2011 - Hai Nguyen

add nodeID in the data packet

version Apr/7/2011 - Hai Nguyen:

\* add nodeID

\* Combine information of there sensor H, T, L, in one packet.

\* Timer in reading PIR, not use delay.

version Apr/6/2011:

split float to 4 bytes and send 4 bytes to server.

\*/

#include <DHT22.h>

#include <Ports.h>

#include <RF12.h>

#include <stdlib.h>

#include <TimerOne.h>

#include <EEPROM.h>

//////////////////////////////////////////////////////////////////////

byte NodeID = 16;

/////////////////////////////////////////////////////////////////////

// PIR connect to analog port 1

Port PIRsensor(1);

// Ambient light sensor connect to analog port 4

Port LightSensor(4); // This pin is used as analog input from sensor (A0 in Arduino = pin A of port 1 in JeeNode)

typedef

union {float asFloat;byte asBytes[4];} f2b;

typedef

union {int asInt;byte asBytes[4];} i2b;

//---------------------------Sensor variable -------------//

i2b myLight; // value read from the light sensor

f2b myTemp; // value read from the Humid/temp sensor

f2b myRH;

int myPIR;

char pirData[1];

char last\_pirData[1];

byte sent\_PIR =0;

byte received\_PIR =0;

int counter;

int occupacy;

int NotOccupacy;

int stop\_counting;

int first\_time;

int minute = 1;

int timer\_counter;

int data\_counter = 10;

// ----------------- Communication variable---------//

byte loaded\_map=0;

byte ReceiveData[60];

byte Data[30];

byte i;

byte LengthOfMap=255;

//------------------

void setup ()

{

Serial.begin(9600);

rf12\_initialize(NodeID, RF12\_915MHZ, 212);

Serial.print("Version 17.");

Serial.print("This is node:"); Serial.print(NodeID,10); Serial.print("\nInitialize.......... \n");

data\_counter = 120;

// Initialize timer

Timer1.initialize(500000); // read the PIR after 0.5s

Timer1.attachInterrupt(read\_PIR\_sensor);

counter = 0;

occupacy = 1;

stop\_counting = 0;

PIRsensor.mode(INPUT);

PIRsensor.digiWrite(1);

delay(10000);

} // end of setup()

void loop ()

{

if (loaded\_map ==0) load\_map(); // load map from eeprom

Data[29]= 0xFF;

if(timer\_counter > 20) //checking after 10s

{

//Serial.print(sent\_PIR,10);

if(sent\_PIR)

check\_PIR\_sent(); //checking PIR data ack from server

timer\_counter = 0; //reset counter

}

// check occupacy

if(occupacy || NotOccupacy)

{

send\_PIR\_data(pirData[0]);

last\_pirData[0] = pirData[0];

if(NotOccupacy)

first\_time = 1; // the next Y will be the first Y

else

first\_time = 0; //only send Y one time until NotOccupacy=1

NotOccupacy = 0; //only send N one time

occupacy = 0; //only send Y one time

sent\_PIR = 1; //sending PIR data

received\_PIR = 0; //haven't received ack from server

timer\_counter = 0; //reset counter

}

//---------- Check RF------------------------------

if (rf12\_recvDone() && rf12\_crc == 0) // waiting for recieving data

{

data\_counter = 20;

Serial.print(" ---=== Begin ===--- \n");

for (i = 0; i < rf12\_len; ++i)

ReceiveData[i]=rf12\_data[i];

Serial.print(" ReceiveData: ");

show\_data(ReceiveData);

if (check\_status()==1) //this is a repeater

{

Repeater();

}

else //this is the final destination

{

Serial.print("Node is Destination \n");

switch (ReceiveData[2])

{

case byte('R'):

{

Serial.print(" Recieved Request: \n ");

i = Data[0];

Data[i+1] = byte('D');

if (ReceiveData[3] == byte('A')) // Type of Request

{

Serial.print(" Recieved Request Information of Humidity & Temperature & Light: \n ");

// Read other sensors.

read\_other\_sensors();

//data\_counter = 10;

Data[i+2] = 3;

Data[i+3] = byte('H');

Data[i+4] = myRH.asBytes[0]; // information of Humidity put on here

Data[i+5] = myRH.asBytes[1]; // information of Humidity put on here

Data[i+6] = myRH.asBytes[2]; // information of Humidity put on here

Data[i+7] = myRH.asBytes[3]; // information of Humidity put on here

Data[i+8] = byte('T');

Data[i+9] = myTemp.asBytes[0]; // information of Temperature put on here

Data[i+10] = myTemp.asBytes[1]; // information of Temperature put on here

Data[i+11] = myTemp.asBytes[2]; // information of Temperature put on here

Data[i+12] = myTemp.asBytes[3]; // information of Temperature put on here

Data[i+13] = byte('L'); // lightData[4]

Data[i+14] = myLight.asBytes[0]; // information of Light put on here

Data[i+15] = myLight.asBytes[1]; // information of Light put on here

Data[i+16] = myLight.asBytes[2]; // information of Light put on here

Data[i+17] = myLight.asBytes[3]; // information of Light put on here

Data[i+18] = byte ('I');

Data[i+19] = NodeID;

Data[i+20] = 255;

// send data

send\_Data();

} // end of case byte ("L")

else

{

Data[i+2] = 0;

Data[i+3] = byte('I');

Data[i+4] = NodeID;

Data[i+5] = 255;

Serial.print("Recieved BAD REQUEST. \n");

send\_Data();

}// end of default

// end switch (ReceiveData[3])

} // end of case byte (R)

case byte('D'):

{

Serial.print(" Recieved Data: \n");

break;

} // end of case byte("D")

case byte('M'): //if Data, store the return direction into Data[]

{

LengthOfMap = ReceiveData[3];

Serial.print("Map information: ");

for (i = 0; i < LengthOfMap + 1; ++i) // assume that just have 3 node

{

//if (ReceiveData[3+i]==0xFF); //if the end character, exit for

Data[i]=ReceiveData[3+i];// bytes after 'M'

Serial.print(Data[i],10);Serial.print(" ");

}

Serial.print(" \n ");

update\_map(); // update map to eeprom;

Serial.print("Send ACK Map: ");

i = Data[0];

Data[i+1] = byte('A');

Data[i+2] = byte('M');

Data[i+3] = byte('I');

Data[i+4] = NodeID;

Data[i+5] = 255;

send\_Data();

break;

} // end of case byte("M")

case byte('S'): //if Data, store the return direction into Data[]

{

Serial.print("Set up information for occupacy.");

minute = ReceiveData[3];

Serial.print("Time before deciding that there is no occupacy. Minute:");Serial.print(minute,DEC);Serial.print(" \n");

Serial.print("Send ACK Setup: ");

i = Data[0];

Data[i+1] = byte('A');

Data[i+2] = byte('S');

Data[i+3] = byte('I');

Data[i+4] = NodeID;

Data[i+5] = 255;

send\_Data();

break;

} // end of case byte("M")

case byte('A'): //if Data, store the return direction into Data[]

{

Serial.print("Recieved ACK: ");

if (ReceiveData[3] == byte('O'))

{

Serial.print("Occupacy \n ");

received\_PIR = 1;

}

break;

} // end of case byte("A")

default:

{

Data[i+1] = byte('B');

Data[i+2] = byte('I');

Data[i+3] = NodeID;

Data[i+4] = 255;

send\_Data();

Serial.print("Recieved BAD Data \n");

}

;

// if nothing else matches, do the default

// default is optional

} // end switch (ReceiveData[2])

Serial.print(" ---=== End ===--- \n");

} //end else

} //end if

} //end void

// ---------- Check status the node is repeater or destination

int check\_status()

{

int temp=0;

Serial.print("Checking node status... \n");

if (ReceiveData[0]!=0x01)

{

ReceiveData[0]=ReceiveData[0]-1;

for (i = 1; i < rf12\_len-1; ++i)

ReceiveData[i]=ReceiveData[i+1];

temp= 1; //repeater

}

return temp;

}

//--------SEND DATA----------

void send\_Data()

{

Serial.print("Sending... \n");

if (LengthOfMap==255)

{

Serial.print("No information of Map to Server \n");

return;

}

rf12\_recvDone();

if (rf12\_canSend())

{

rf12\_sendStart(RF12\_HDR\_DST |Data[1],Data , sizeof Data);

}

Serial.print(" Data sent: ");

show\_data(Data);

}

//--------SHOW DATA to serial port-------------

void show\_data(uint8\_t \*m)

{

uint8\_t i;

for (i = 0; i<29 ; ++i)

{

Serial.print(m[i],10); Serial.print(" ");

}

Serial.print("\n");

}

//-----------------------Update map to eeprom---------------------- //

void update\_map()

{

byte i;

EEPROM.write(0,LengthOfMap);

Serial.println("Update map information to EEPROM... ");

for (i = 0; i < LengthOfMap + 1; ++i) // assume that just have 3 node

{

EEPROM.write(i,Data[i]);

}

}

//----------------------------------------------------------

//----------------------Load map from eeprom-------------------

void load\_map()

{

byte i;

loaded\_map=1;

Data[0]=EEPROM.read(0);

LengthOfMap = Data[0];

Serial.println("Reading map information to EEPROM... ");

Serial.print(Data[i],10);Serial.print(" ");

if (LengthOfMap==255)

{

Serial.println("No map information is stored ");

return;

}

Serial.print("Map information:");

for (i = 0; i < LengthOfMap + 1; ++i) // assume that just have 3 node

{

Data[i] = EEPROM.read(i);

Serial.print(Data[i],10);Serial.print(" ");

}

Serial.print("\n ");

}

void Repeater()

{

Serial.print("Node is Repeater \n");

rf12\_recvDone();

if (rf12\_canSend())

{

rf12\_sendStart(RF12\_HDR\_DST | ReceiveData[1], ReceiveData , sizeof ReceiveData);

}

Serial.print(" \n Data sent: ");

show\_data(ReceiveData);

Serial.print(" ---=== End ===--- \n");

}

///////////////////////////////////////////////////////////////////////////////

void read\_PIR\_sensor()

{

if(data\_counter > 0)

{

data\_counter = data\_counter - 1;

return;

}

myPIR = PIRsensor.digiRead();

if(myPIR == 1)

{

pirData[0] = 'Y';

if (first\_time) //if this is the first Y after N

occupacy = 1; //indicate that there is a NEW motion in the room

counter = 0; // reset counter

stop\_counting = 0; // allow counting

}

else

{

pirData[0] = 'N'; // no motion

if(stop\_counting == 0) // if allowed to count

counter = counter + 1; // increase the counter by 1

}

if(counter > (minute\*120)) // if counter = minute\*120 or 'N' after 'minute'

{

NotOccupacy = 1; // indicate that there is no occupacy in the room

counter = 0; // reset counter

stop\_counting = 1; //stop counting until there is a 'Y' again

}

timer\_counter = timer\_counter + 1; //counter for calling the check\_PIR\_sent()

Serial.print("PIR:");

Serial.print(pirData[0]);

Serial.print("counter = ");

Serial.print(counter);

Serial.print(" \n ");

}

///////////////////////////////////////////////////////////////////////////////

void read\_other\_sensors()

{

// Humidity/ Temp sensor connect to port 3 JeeNode(pin 6 arduino)

#define DHT22\_PIN 6

//Port DHT22\_PIN (3);

// Setup a DHT22 instance

DHT22 myDHT22(DHT22\_PIN);

int i;

DHT22\_ERROR\_t errorCode;

Serial.print(" Reading sensor... \n ");

delay(2000); //delay for the humid/temp sensor stabelized

//read the humidity/temp sensor

errorCode = myDHT22.readData();

if(errorCode == DHT\_ERROR\_NONE) //if non error occur

{

myRH.asFloat = myDHT22.getHumidity();

myTemp.asFloat = myDHT22.getTemperatureC();

}

else //if error occur, data is all zero

{

myRH.asFloat = 0.00;

myTemp.asFloat = 0.00;

}

// read the ambient light sensor value

myLight.asInt = LightSensor.anaRead();

}

//--------------------SEND PIR DATA-------------------------------//

void send\_PIR\_data(char PIR\_data)

{

int i = LengthOfMap;

Data[i+1] = byte('D');

Data[i+2] = 1;

Data[i+3] = byte('O');

Data[i+4] = PIR\_data; // information of Occupancy put on here

Data[i+5] = byte('I');

Data[i+6] = NodeID;

Data[i+7] = 255;

// send data

Serial.print(" Sent Information of Occupancy \n ");

send\_Data(); // send data

}

//---------------------------------------------------------

void check\_PIR\_sent()

{

if(received\_PIR == 0) //if server haven't received the PIR data -> resend

{

data\_counter = 10;

Serial.print(" Don't receive ACK. send PIR again. \n ");

send\_PIR\_data(last\_pirData[0]); //re-send PIR data

}

else //server successfully received data -> reset flags

{

sent\_PIR = 0;

received\_PIR = 0;

}

}

### Date 05/14/2011

***Working on communication part for water flow sensor.***

This program sent the waterflow sensor data each 1 minute (defaut value, it can be changed by server).

After sending the water flow information, each 10s, the program check the acknowledgment from sever, if it is not received ack, it will resend the water flow information.

/\*

SENSOR NOTE

WATER FLOW SENSOR

\*/

#include <DHT22.h>

#include <Ports.h>

#include <RF12.h>

#include <stdlib.h>

#include <TimerOne.h>

#include <EEPROM.h>

//////////////////////////////////////////////////////////////////////

byte NodeID = 16;

/////////////////////////////////////////////////////////////////////

typedef

union {float asFloat;byte asBytes[4];} f2b;

typedef

union {int asInt;byte asBytes[4];} i2b;

//---------------------------Sensor variable -------------//

int myPIR;

char pirData[1];

char last\_pirData[1];

byte sent\_WATER =0;

byte received\_WATER =0;

int last\_WATER;

int Water\_data =0;

int first\_time;

int minute = 1;

int timer\_counter = 0;

int timer\_counter2 = 0;;

int data\_counter = 10;

// ----------------- Communication variable---------//

byte loaded\_map=0;

byte ReceiveData[60];

byte Data[30];

byte i;

byte LengthOfMap=255;

//------------------

void setup ()

{

Serial.begin(9600);

rf12\_initialize(NodeID, RF12\_915MHZ, 212);

Serial.print("Version for Water flow sensor. Version 1");

Serial.print("This is node:"); Serial.print(NodeID,10); Serial.print("\nInitialize.......... \n");

data\_counter = 120;

// Initialize timer

Timer1.initialize(500000); // read the PIR after 0.5s

Timer1.attachInterrupt(Counter\_0\_5s);

// delay(50000);

// read\_Water;

} // end of setup()

void loop ()

{

if (loaded\_map ==0) load\_map(); // load map from eeprom

Data[29]= 0xFF;

if(timer\_counter > 20) //checking after 10s

{

//Serial.print(sent\_WATER,10);

if(sent\_WATER)

check\_WATER\_sent(); //checking PIR data ack from server

timer\_counter = 0; //reset counter

}

if(timer\_counter2 > minute\*120) //checking after 10s

{

read\_Water();

timer\_counter2 = 0; //reset counter

}

// check occupacy

//---------- Check RF------------------------------

if (rf12\_recvDone() && rf12\_crc == 0) // waiting for recieving data

{

data\_counter = 20;

Serial.print(" ---=== Begin ===--- \n");

for (i = 0; i < rf12\_len; ++i)

ReceiveData[i]=rf12\_data[i];

Serial.print(" ReceiveData: ");

show\_data(ReceiveData);

if (check\_status()==1) //this is a repeater

{

Repeater();

}

else //this is the final destination

{

Serial.print("Node is Destination \n");

switch (ReceiveData[2])

{

case byte('R'):

{

Serial.print(" Recieved Request: \n ");

i = Data[0];

Data[i+1] = byte('D');

if (ReceiveData[3] == byte('A')) // Type of Request

{

Serial.print(" Recieved Request Information of Humidity & Temperature & Light: \n ");

// Read other sensors.

} // end of case byte ("L")

else

{

Data[i+2] = 0;

Data[i+3] = byte('I');

Data[i+4] = NodeID;

Data[i+5] = 255;

Serial.print("Recieved BAD REQUEST. \n");

send\_Data();

}// end of default

// end switch (ReceiveData[3])

} // end of case byte (R)

case byte('D'):

{

Serial.print(" Recieved Data: \n");

break;

} // end of case byte("D")

case byte('M'): //if Data, store the return direction into Data[]

{

LengthOfMap = ReceiveData[3];

Serial.print("Map information: ");

for (i = 0; i < LengthOfMap + 1; ++i) // assume that just have 3 node

{

//if (ReceiveData[3+i]==0xFF); //if the end character, exit for

Data[i]=ReceiveData[3+i];// bytes after 'M'

Serial.print(Data[i],10);Serial.print(" ");

}

Serial.print(" \n ");

update\_map(); // update map to eeprom;

Serial.print("Send ACK Map: ");

i = Data[0];

Data[i+1] = byte('A');

Data[i+2] = byte('M');

Data[i+3] = byte('I');

Data[i+4] = NodeID;

Data[i+5] = 255;

send\_Data();

break;

} // end of case byte("M")

case byte('S'): //if Data, store the return direction into Data[]

{

Serial.print("Set up information for Water level flow.");

minute = ReceiveData[3];

Serial.print("Time interval to sending Water flow information. Minute:");Serial.print(minute,DEC);Serial.print(" \n");

Serial.print("Send ACK Setup: ");

i = Data[0];

Data[i+1] = byte('A');

Data[i+2] = byte('S');

Data[i+3] = byte('I');

Data[i+4] = NodeID;

Data[i+5] = 255;

send\_Data();

break;

} // end of case byte("M")

case byte('A'): //if Data, store the return direction into Data[]

{

Serial.print("Recieved ACK: ");

if (ReceiveData[3] == byte('G'))

{

Serial.print("Water flow \n ");

received\_WATER = 1;

}

break;

} // end of case byte("A")

default:

{

Data[i+1] = byte('B');

Data[i+2] = byte('I');

Data[i+3] = NodeID;

Data[i+4] = 255;

send\_Data();

Serial.print("Recieved BAD Data \n");

}

;

// if nothing else matches, do the default

// default is optional

} // end switch (ReceiveData[2])

Serial.print(" ---=== End ===--- \n");

} //end else

} //end if

} //end void

// ---------- Check status the node is repeater or destination

int check\_status()

{

int temp=0;

Serial.print("Checking node status... \n");

if (ReceiveData[0]!=0x01)

{

ReceiveData[0]=ReceiveData[0]-1;

for (i = 1; i < rf12\_len-1; ++i)

ReceiveData[i]=ReceiveData[i+1];

temp= 1; //repeater

}

return temp;

}

//--------SEND DATA----------

void send\_Data()

{

Serial.print("Sending... \n");

if (LengthOfMap==255)

{

Serial.print("No information of Map to Server \n");

return;

}

rf12\_recvDone();

if (rf12\_canSend())

{

rf12\_sendStart(RF12\_HDR\_DST |Data[1],Data , sizeof Data);

}

Serial.print(" Data sent: ");

show\_data(Data);

}

//--------SHOW DATA to serial port-------------

void show\_data(uint8\_t \*m)

{

uint8\_t i;

for (i = 0; i<29 ; ++i)

{

Serial.print(m[i],10); Serial.print(" ");

}

Serial.print("\n");

}

//-----------------------Update map to eeprom---------------------- //

void update\_map()

{

byte i;

EEPROM.write(0,LengthOfMap);

Serial.println("Update map information to EEPROM... ");

for (i = 0; i < LengthOfMap + 1; ++i) // assume that just have 3 node

{

EEPROM.write(i,Data[i]);

}

}

//----------------------------------------------------------

//----------------------Load map from eeprom-------------------

void load\_map()

{

byte i;

loaded\_map=1;

Data[0]=EEPROM.read(0);

LengthOfMap = Data[0];

Serial.println("Reading map information to EEPROM... ");

Serial.print(Data[i],10);Serial.print(" ");

if (LengthOfMap==255)

{

Serial.println("No map information is stored ");

return;

}

Serial.print("Map information:");

for (i = 0; i < LengthOfMap + 1; ++i) // assume that just have 3 node

{

Data[i] = EEPROM.read(i);

Serial.print(Data[i],10);Serial.print(" ");

}

Serial.print("\n ");

}

void Repeater()

{

Serial.print("Node is Repeater \n");

rf12\_recvDone();

if (rf12\_canSend())

{

rf12\_sendStart(RF12\_HDR\_DST | ReceiveData[1], ReceiveData , sizeof ReceiveData);

}

Serial.print(" \n Data sent: ");

show\_data(ReceiveData);

Serial.print(" ---=== End ===--- \n");

}

///////////////////////////////////////////////////////////////////////////////

void Counter\_0\_5s()

{

timer\_counter = timer\_counter + 1; //counter for calling the read\_Water

timer\_counter2 = timer\_counter2 + 1;

}

///////////////////////////////////////////////////////////////////////////////

//

void read\_Water()

{

Serial.print(" Reading Information of Water... \n ");

Water\_data = Water\_data + 1; // put information here, and just here.

Serial.print(" Water = "); Serial.println(Water\_data,10);

send\_WATER\_data(Water\_data);

last\_WATER = Water\_data;

sent\_WATER = 1; //sending PIR data

received\_WATER = 0; //haven't received ack from server

timer\_counter2 = 0; //reset counter

}

//--------------------SEND PIR DATA-------------------------------//

void send\_WATER\_data(int WATER\_data)

{ i2b send\_inf;

int i = LengthOfMap;

send\_inf.asInt = WATER\_data;

Data[i+1] = byte('D');

Data[i+2] = 1;

Data[i+3] = byte('G');

Data[i+4] = send\_inf.asBytes[0]; // information of Water put on here

Data[i+5] = send\_inf.asBytes[1]; // information of Water put on here

Data[i+6] = send\_inf.asBytes[2]; // information of Water put on here

Data[i+7] = send\_inf.asBytes[3]; // information of Water put on here

Data[i+8] = byte('I');

Data[i+9] = NodeID;

Data[i+10] = 255;

// send data

Serial.print(" Sent Information of Water \n ");

send\_Data(); // send data

}

//---------------------------------------------------------

void check\_WATER\_sent()

{

if(received\_WATER == 0) //if server haven't received the PIR data -> resend

{

data\_counter = 10;

Serial.print(" Don't receive ACK. send Water flow data again. \n ");

send\_WATER\_data(last\_WATER); //re-send WATER data

}

else //server successfully received data -> reset flags

{

sent\_WATER = 0;

received\_WATER = 0;

}

}

***Work on power monitor communication part.***

*Same as waterflow.*