**Latest Codes Documentation**

Final Mesh Network Configuration( Including Handshaking):

1.1 Sensor node:

**#include <WaspSensorEvent\_v30.h>**

**#include <WaspXBeeDM.h>**

**#include <WaspFrame.h>**

**char RX\_ADDRESS[] = "0013A200416B4D87";**

**char WASPMOTE\_ID[] = "A";**

**uint8\_t error;**

**float value1 = 0;**

**float temp;**

**float humd;**

**float pres;**

**float Motion;**

**float value;**

**pirSensorClass pir(SOCKET\_1);**

**void setup()**

**{USB.ON();**

**USB.println(F("Start program"));**

**frame.setID( WASPMOTE\_ID );**

**Events.ON();**

**Events.attachInt();}**

**void loop()**

**{Motion = pir.readPirSensor();**

**temp = Events.getTemperature();**

**humd = Events.getHumidity();**

**pres = Events.getPressure();**

**if (Motion == 1)**

**{USB.println(F(" Presence detected")); }**

**else { USB.println(F(" Presence not detected"));}**

**USB.print("Temperature: ");**

**USB.println(temp);**

**USB.print("Humidity: ");**

**USB.println(humd);**

**USB.print("Pressure: ");**

**USB.println(pres);**

**USB.println("-----------------------------");**

**USB.println(F("enter deep sleep"));**

**PWR.deepSleep("00:00:00:03", RTC\_OFFSET, RTC\_ALM1\_MODE1, SENSOR\_ON);**

**USB.ON();**

**USB.println(F("wake up\n"));**

**if (intFlag & RTC\_INT)**

**{USB.println(F("-------------------------------------"));**

**USB.println(F("RTC INT captured"));**

**USB.println(F("-------------------------------------"));**

**intFlag &= ~(RTC\_INT);}**

**sendInfoData();}**

**void sendInfoData()**

**{xbeeDM.ON();**

**temp = Events.getTemperature();**

**humd = Events.getHumidity();**

**pres = Events.getPressure();**

**Motion = pir.readPirSensor();**

**frame.createFrame(ASCII);**

**frame.addSensor(SENSOR\_EVENTS\_TC, temp);**

**frame.addSensor(SENSOR\_EVENTS\_HUM, humd);**

**frame.addSensor(SENSOR\_EVENTS\_PRES, pres);**

**frame.addSensor(SENSOR\_EVENTS\_PIR, Motion);**

**frame.showFrame();**

**error = xbeeDM.send( RX\_ADDRESS, frame.buffer, frame.length );**

**if ( error == 0 )**

**{USB.println(F("send ok"));**

**Utils.blinkGreenLED(); }**

**else{USB.println(F("send error"));**

**Utils.blinkRedLED();}**

**USB.print(F("Battery Level: "));**

**USB.print(PWR.getBatteryLevel(),DEC);**

**USB.print(F(" %"));**

**USB.print(F(" | Battery (Volts): "));**

**USB.print(PWR.getBatteryVolts());**

**USB.println(F(" V"));**

**delay(1000);}**

* 1. Sensor and Relay node Configuration(prefer to use for all the nodes):

**#include <WaspXBeeDM.h>**

**#include <WaspFrame.h>**

**#include <WaspSensorEvent\_v30.h>**

**char RX\_ADDRESS[] = "0013A200416B4D87";**

**char WASPMOTE\_ID[] = "M";**

**uint8\_t error;**

**float value1 = 0;**

**float temp;**

**float humd;**

**float pres;**

**float Motion;**

**float value;**

**pirSensorClass pir(SOCKET\_1);**

**void setup()**

**{ USB.ON();**

**USB.println(F("Sending packets and scanning example"));**

**frame.setID( WASPMOTE\_ID );**

**xbeeDM.ON();**

**Events.ON();**

**Events.attachInt();}**

**void loop()**

**{ Motion = pir.readPirSensor();**

**temp = Events.getTemperature();**

**humd = Events.getHumidity();**

**pres = Events.getPressure();**

**if (Motion == 1)**

**{ USB.println(F(" Presence detected")); }**

**else { USB.println(F(" Presence not detected")); }**

**USB.print("Temperature: ");**

**USB.println(temp);**

**USB.print("Humidity: ");**

**USB.println(humd);**

**USB.print("Pressure: ");**

**USB.println(pres);**

**USB.println("-----------------------------");**

**temp = Events.getTemperature();**

**humd = Events.getHumidity();**

**pres = Events.getPressure();**

**Motion = pir.readPirSensor();**

**frame.createFrame(ASCII);**

**frame.addSensor(SENSOR\_EVENTS\_TC, temp);**

**frame.addSensor(SENSOR\_EVENTS\_HUM, humd);**

**frame.addSensor(SENSOR\_EVENTS\_PRES, pres);**

**frame.addSensor(SENSOR\_EVENTS\_PIR, Motion);**

**frame.showFrame();**

**error = xbeeDM.send( RX\_ADDRESS, frame.buffer, frame.length );**

**if( error == 0 )**

**{USB.println(F("send ok"));**

**Utils.blinkGreenLED(); }**

**else {USB.println(F("send error"));**

**Utils.blinkRedLED();}**

**delay(5000);**

**xbeeDM.scanNetwork();**

**USB.print(F("\n\ntotalScannedBrothers:"));**

**USB.println(xbeeDM.totalScannedBrothers,DEC);**

**printScanInfo();**

**delay(5000);}**

**void printScanInfo()**

**{ USB.println(F("----------------------------"));**

**for(int i=0; i<xbeeDM.totalScannedBrothers; i++)**

**{ USB.print(F("MAC:"));**

**USB.printHex(xbeeDM.scannedBrothers[i].SH[0]);**

**USB.printHex(xbeeDM.scannedBrothers[i].SH[1]);**

**USB.printHex(xbeeDM.scannedBrothers[i].SH[2]);**

**USB.printHex(xbeeDM.scannedBrothers[i].SH[3]);**

**USB.printHex(xbeeDM.scannedBrothers[i].SL[0]);**

**USB.printHex(xbeeDM.scannedBrothers[i].SL[1]);**

**USB.printHex(xbeeDM.scannedBrothers[i].SL[2]);**

**USB.printHex(xbeeDM.scannedBrothers[i].SL[3]);**

**USB.print(F("\nNI:"));**

**USB.print(xbeeDM.scannedBrothers[i].NI);**

**USB.print(F("\nDevice Type:"));**

**switch(xbeeDM.scannedBrothers[i].DT)**

**{ case 0: USB.print(F("End Device"));**

**break;**

**case 1: USB.print(F("Router"));**

**break;**

**case 2: USB.print(F("Coordinator"));**

**break; }**

**USB.print(F("\nPMY:"));**

**USB.printHex(xbeeDM.scannedBrothers[i].PMY[0]);**

**USB.printHex(xbeeDM.scannedBrothers[i].PMY[1]);**

**USB.print(F("\nPID:"));**

**USB.printHex(xbeeDM.scannedBrothers[i].PID[0]);**

**USB.printHex(xbeeDM.scannedBrothers[i].PID[1]);**

**USB.print(F("\nMID:"));**

**USB.printHex(xbeeDM.scannedBrothers[i].MID[0]);**

**USB.printHex(xbeeDM.scannedBrothers[i].MID[1]);**

**USB.println(F("\n----------------------------")); }**

**USB.print(F("Battery Level: "));**

**USB.print(PWR.getBatteryLevel(),DEC);**

**USB.print(F(" %"));**

**USB.print(F(" | Battery (Volts): "));**

**USB.print(PWR.getBatteryVolts());**

**USB.println(F(" V"));}**

* 1. Sync Node:

**#include <WaspXBeeDM.h>**

**#include <WaspFrame.h>**

**#define DEF\_BAUD\_RATE 9600 //for sync node only**

**char filename[]="FILE2.TXT";**

**uint8\_t sd\_answer;**

**int32\_t numLines=0;**

**uint8\_t error;**

**uint8\_t answer[256];**

**char response[256];**

**char x[256];**

**bool conf = false;**

**bool flag1 = false;**

**int i,f,n,c = 0;**

**int m =-1;**

**void setup()**

**{**

**settingup();**

**}**

**void loop()**

**{**

**if(!conf){**

**error = xbeeDM.receivePacketTimeout( 3000 ); //receive packets from nearby nodes**

**// check answer**

**if( error == 0 )**

**{**

**// Show data stored in '\_payload' buffer indicated by '\_length'**

**USB.print(F("Data: "));**

**USB.println( xbeeDM.\_payload, xbeeDM.\_length);**

**// Show data stored in '\_payload' buffer indicated by '\_length'**

**USB.print(F("Length: "));**

**USB.println( xbeeDM.\_length,DEC);**

**for(uint8\_t i = 0; i < xbeeDM.\_length; i++){ //save data received into char array x**

**x[i] = xbeeDM.\_payload[i];**

**}**

**sd\_answer = SD.appendln(filename,x); //add char array x to our sd card file**

**}**

**else //xbee receive timeout was reached or another error occured**

**{**

**// Print error message:**

**/\***

**\* '7' : Buffer full. Not enough memory space**

**\* '6' : Error escaping character within payload bytes**

**\* '5' : Error escaping character in checksum byte**

**\* '4' : Checksum is not correct**

**\* '3' : Checksum byte is not available**

**\* '2' : Frame Type is not valid**

**\* '1' : Timeout when receiving answer**

**\*/**

**USB.print(F("Error receiving a packet:"));**

**USB.println(error,DEC);**

**}**

**SD.showFile(filename); //show our sd card file**

**delay(200);**

**numLines = SD.numln(filename); //this is to see how many frames are currently saved in sd card**

**serialFlush(UART1); //resetting buffer in order to listen for drone node**

**memset(answer, 0x00, sizeof(answer) ); //resetting buffer in order to listen for drone node**

**USB.println("number of stored frames is : ");**

**USB.println(numLines);**

**if(numLines > 60)receivereq(); //listen for drone only after 60 or more packets are in sd card**

**}**

**if(conf){**

**m++;**

**sendframe();**

**receiveack();**

**delay(1500);**

**if(m==numLines){**

**sendLastframe();**

**m=-1;**

**conf=false;**

**sleepnode();**

**}**

**}**

**}**

**void receivereq(){**

**Utils.setMuxAux2(); //aux1 for hc12, aux2 for hf**

**delay(500);**

**USB.println("listening for 12s...");**

**memset(answer, 0x00, sizeof(answer) );**

**i=0;**

**c=0;**

**while(!serialAvailable(1) && c < 120){**

**delay(100);**

**c++;**

**}**

**while(serialAvailable(1)) //read the buffer from UART1, which is connected to aux2(hf)**

**{**

**answer[i]=serialRead(1);**

**i++;**

**delay(10);**

**}**

**for (uint8\_t a = 0; a < i; a++)**

**{**

**USB.print(answer[a]); //type out the response**

**}**

**USB.println();**

**for(uint8\_t o = 0; o < i; o++) //convert answer from bytes to chars**

**{**

**response[o] = (char)answer[o];**

**}**

**if(response[0] == '1' && response[1] == '2' && response[2] == '3' && response[3] == '4'){ //check for correct response**

**USB.println("pass matched, drone detected!");**

**conf = true; //change states**

**Utils.setMuxAux1(); //turn uart to hc12**

**response[0] = '\0'; //reset response to empty string in order to prepare to receive ack from drone node**

**delay(500);**

**for(int f = 0; f < 3; f++){ //send response back to the drone 3 times, 2sec between each time**

**printString("4321\n", 1);**

**USB.println("4321");**

**delay(2000);**

**}**

**}**

**delay(1500);**

**}**

**void sendframe(){**

**Utils.setMuxAux1(); //aux1 for hc12, aux2 for hf**

**delay(100);**

**// read new line (it is stored in SD.buffer)**

**SD.catln(filename,m,1);**

**printString(SD.buffer, 1); //send frame from sd card to drone node**

**delay(1200);**

**}**

**void sendLastframe(){**

**Utils.setMuxAux1(); //aux1 for hc12, aux2 for hf**

**delay(500);**

**printString("last frame was sent from network 1, going to sleep ...", 1); //send this string to**

**USB.println("last frame was sent from network 1, going to sleep ..."); //indicate that we finished transmission**

**}**

**void receiveack(){**

**Utils.setMuxAux2(); //aux1 for hc12, aux2 for hf**

**delay(500);**

**USB.println("listening...");**

**memset(answer, 0x00, sizeof(answer) );**

**i=0;**

**n=0;**

**while(!serialAvailable(1) && n < 70){**

**delay(100);**

**n++;**

**}**

**USB.println(n);**

**while(serialAvailable(1))**

**{**

**answer[i]=serialRead(1);**

**i++;**

**delay(10);**

**}**

**USB.println("response as bytes: ");**

**for (uint8\_t a = 0; a < i; a++)**

**{**

**USB.print(answer[a]); //type out the response**

**}**

**USB.println();**

**//response[0] = '\0';**

**USB.println("response as chars: ");**

**for(uint8\_t o = 0; o < i; o++) //convert req from bytes to chars**

**{**

**response[o] = (char)answer[o];**

**}**

**USB.println(response);**

**if(n==70 || (response[0] == '1' && response[1] == '2' && response[2] == '3' && response[3] == '4')){**

**conf=false;**

**USB.println("timeout 7s reached or drone in wrong loop switching back to listening ");**

**}**

**response[0] = '\0';**

**}**

**void settingup(){**

**RTC.ON();**

**ACC.ON();**

**USB.ON();**

**PWR.powerSocket(UART1, HIGH); //powering up**

**PWR.setSensorPower(SENS\_3V3, SENS\_ON); //set hc12 and hf on**

**WaspUART uart = WaspUART();**

**uart.setUART(UART1);**

**delay(500);**

**uart.setBaudrate(9600);**

**delay(500);**

**Utils.setMuxAux2(); //aux1 for hc12, aux2 for hf**

**delay(500);**

**serialFlush(UART1);**

**uart.beginUART();**

**delay(1500);**

**SD.ON();**

**// Delete file**

**sd\_answer = SD.del(filename);**

**if( sd\_answer == 1 )**

**{**

**USB.println(F("file deleted"));**

**}**

**else**

**{**

**USB.println(F("file NOT deleted"));**

**}**

**// Create file**

**sd\_answer = SD.create(filename);**

**if( sd\_answer == 1 )**

**{**

**USB.println(F("file created"));**

**}**

**else**

**{**

**USB.println(F("file NOT created"));**

**}**

**xbeeDM.ON();**

**xbeeDM.writeValues();**

**}**

**void sleepnode(){**

**USB.println("finished sending sd card file, entering deep sleep mode");**

**PWR.deepSleep("00:00:00:17", RTC\_OFFSET, RTC\_ALM1\_MODE1,**

**SENSOR\_ON);**

**USB.ON();**

**USB.println(F("wake up\n"));**

**if (intFlag & RTC\_INT)**

**{**

**USB.println(F("-------------------------------------"));**

**USB.println(F("RTC INT captured"));**

**USB.println(F("-------------------------------------"));**

**// clear flag**

**intFlag &= ~(RTC\_INT);**

**settingup();**

**}**

**}**

1.3 Drone node:

**// created by Ahmad Abdullah (German Jordanian University WSN project, supervised**

**// by Dr. Ala' Khalifeh)**

**#include <SD.h>**

**#include <SPI.h>**

**String Data;**

**File file;**

**char myFileName[] = "test.txt";**

**String pass = "1234\n";**

**String Data2;**

**String Data3;**

**String Data4;**

**String ack = "ACK\n";**

**int conf = 0;**

**int m = 0;**

**void setup() {**

**pinMode(10, OUTPUT);**

**digitalWrite(10, HIGH);**

**Serial3.begin(9600); //hc12 link**

**Serial2.begin(9600); //hf link**

**Serial.begin(9600); //for usb**

**pinMode(53, OUTPUT); //slave select for sd card**

**digitalWrite(53, LOW); //making sure sd card is selected**

**pinMode(SS, OUTPUT); //for sd card**

**if(!SD.begin(53))Serial.println("SD card not initialized"); //incase sd card is not working**

**else Serial.println("SD card initialized");**

**}**

**void loop(){**

**if(conf==0)sendandlisten(); //drone searching for sync nodes**

**if(conf==1){**

**ReadFrameandSendAck(); //receiving from sync node**

**delay(3500);**

**if(m==3){**

**conf=0;**

**m=0;**

**file.close(); //close file after reception is complete**

**}**

**}**

**}**

**void sendandlisten(){**

**Serial2.print(pass);**

**Serial.println(pass);**

**delay(2000);**

**Data2 = Serial3.readString();**

**Serial.println("received...");**

**Serial.print(Data2);**

**if(Data2 == "4321\n") //this if statement can be modified to filter out network clusters,**

**{ //since variable Data2 could differ from one sync node to the other**

**Serial.println("Correct response, network detected");**

**file = SD.open(myFileName, FILE\_WRITE); //open a file**

**file.println("----new initialization---"); //add a new line indicating a new session**

**file.flush(); //save the file**

**conf++; //change state to network not found to network found**

**}**

**}**

**void ReadFrameandSendAck(){**

**if(Serial3.available()){ //check to see if we have new data**

**delay(200);**

**Data4 = Serial3.readString(); //read the data**

**Serial.println("from HF: ");**

**Serial.println(Data4);**

**file.println(Data4); //write data into sd card**

**file.flush(); //save data into sd card**

**Serial.println("sending ack ");**

**delay(500);**

**Serial2.print(ack); //send ack back to sync node**

**m = 0; //zeroing counter m since we're in range**

**}**

**else {**

**m++; //we stopped receiving frames, at m=3 we'll go back to sending requests**

**Serial.println("no new frame received ");**

**}**

**}**

**Matlab Parsing Code:**

function [t] = parse(a)

sz = size(a);

edit exp.txt;

fileID = fopen('exp.txt','w');

for i=1:1:sz(2)

a1 = string(a(1,i));

str1 = extractBetween(a1,'TC:','#HUM');

str2 = extractBetween(a1,'HUM:','#PRES');

str3 = extractBetween(a1,'PRES:','#PIR');

str4 = extractBetween(a1,'PIR:','#');

fprintf(fileID,'Temperature : ');

fprintf(fileID,'%s \n',str1);

fprintf(fileID,'Humidity : ');

fprintf(fileID,'%s \n',str2);

fprintf(fileID,'Pressure : ');

fprintf(fileID,'%s \n',str3);

fprintf(fileID,'Motion detection : ');

fprintf(fileID,'%s \n',str4);

fprintf(fileID,'--------------------------- \n');

end

fclose(fileID);

return

**How to open the text file in matlab**

Make sure your text file and matlab code file in the same directory and opened. When you call the function using the following command on the window command:

temp = regexp(fileread('TEST.txt'), '\r?\n', 'split');

Then call the function:

wip(temp)

The output of this function:

