### **Rfu-328 Progress report:**

**Week two**

# Currently implemented:

Continuing the use of radio frequency to transfer messages from the devices, the code has been updated to create better and more efficient use of sending data and receiving data. This is because both devices would have to needlessly waste processing time on messages that would fill the buffer of the recipient. To avoid this we have to restrict the sending of data but at the same time check that the message being sent are actually being received. Eventually the hub would have to make a decision based on the values sent by the controller and so a response was built into the acknowledgment message so that a led could be turned off/on with a good response time.

Some messages will by default not get received by the Hub or end up being corrupted for various reasons. One way we considered to deal with this was to create a backlog of messages then send them all at once until an acknowledgement was received. This idea was scrapped since due the maximum message size is too small and so would require the resending of separate messages. Whilst this idea could be implemented using ID’s for each message and acknowledgement it was deemed to complex and the end user would not lose any sufficient data over the course of a day/running of the device.

Instead to notify the user of consecutive non acknowledged messages the controller was equipped with a buzzer.

# Sleep Mode:

One method to try to improve the system was sending the controllers into sleep mode when they were not receiving messages. The issue with this however is that the device also sends the radio SF into sleep preventing messages being seen. Having the device sleep for a small amount of time was also tested but the device did not gain allot of sleep time from doing this , weather this is a notable improvement over a long period of time has yet to be tested.

# Code Explanation:

***Controller***

#include <LLAPSerial.h>

#define DEVICEID "A2" // this is the LLAP device ID

char incomingByte; // the command buffer

String commandBuilder = "";

int sensorValue = 0;

boolean lightOn;

int volatile counter= 0;

int lastReading;

int missedAcknowledgementNumber = 0;

boolean signalAcknowledgement = false;;

//String extraCommands[10];

***Code set up – sets up the pins that will be used for input and contains basic sleep test that can be removed***

void setup(){

//sleep mode stuff

pinMode(4,OUTPUT); // hardwired XinoRF / RFu328 SRF sleep pin

digitalWrite(4,LOW); // pull sleep pin high - sleep 2 disabled

Serial.print("+++"); // enter AT command mode

delay(1500); // delay 1.5s

Serial.println("ATSM2"); // enable sleep mode 2 <0.5uA

delay(2000);

Serial.println("ATDN"); // exit AT command mode\*/

delay(2000);

//Done

//Pins and extras

pinMode(12, OUTPUT); // initialize pin 12 as digital output (LED)

pinMode(8, OUTPUT); // initialize pin 8 to control the radio

pinMode(6, OUTPUT); // led pin;

pinMode(1, INPUT); // pin for variable Resistor input

digitalWrite(6,HIGH ); //start of the LED to off

digitalWrite(8, HIGH); // select the radio

Serial.begin(115200); // start the serial port at 115200 baud

delay(1000); // allow the radio to startup

LLAP.init(DEVICEID);

//Start up

Serial.println("STARTED");

Serial.print("ABCDEFGHIJKLMNOPQRSTUVWX");

Serial.flush();

}

***Main loop – Will listen for a command input for 10 then send for 1. Resulting in a message being sent every 1 second and the device will listen in the meantime for an acknowledgement or another possible command***

void loop(){

//Check for messages

if(counter>10){

//sending message to the hub

sendDataToHub();

}else{

//listern to responses from the hub

listernToHub();

}

delay(100);

}

***Sending data to the hub – counts if the last message has not been acknowledged, outputting a noise to the buzzer if more than 5 consecutive missed acknowledgements. Will also sends a reading of the sensor pin to the hub.***

void sendDataToHub(){

sendSensorData();

counter=0;

//check if our signal has been accepted

if(signalAcknowledgement == false){

//signal not received

missedAcknowledgementNumber++;

}else{

signalAcknowledgement = false;

}

if(missedAcknowledgementNumber == 5){

tone(5,1000,250);

missedAcknowledgementNumber=0;

}

}

***Checks if the hub has sent any messages to the controller and increments the counter used to balance the sending and listening times. If an acknowledgement was sent will also take note of it.***

void listernToHub(){

counter++;

if(LLAPmessageReceiver()){

//not received

}else{

//signal received

signalAcknowledgement = true;

//resset the counter since we are still in comunication with the hub

missedAcknowledgementNumber=0;

}

}

***Receives a string and compares it to a list of commands; commands will each perform some task. Function will return false if the command sent was an acknowledgement.***

boolean commandHandler(String incomingCommand){

//takes in the command

if(incomingCommand =="DONEON---"){

//message Acknowledgement

digitalWrite(6, HIGH);

//Serial.println("MESSAGE ACKNOWLEDGED");

return false;

}

else if(incomingCommand =="DONEOFF--"){

//message Acknowledgement

digitalWrite(6, LOW);

//Serial.println("MESSAGE ACKNOWLEDGED");

return false;

}else{

//command not recognised

//Serial.println("COMMAND NOT RECOGNISED");

}

return true;

}

***Will check if a message has been sent. If so then the command will be stripped from the message and passed for processing. Returns false if a command was an acknowledgement.***

boolean LLAPmessageReceiver(){

//used only to acknolege a previous message;

boolean messageAcknowledgement;

if (LLAP.bMsgReceived) {

//collects message

//Serial.print("message is:");

//Serial.println(LLAP.sMessage);

String command = LLAP.sMessage;

messageAcknowledgement = commandHandler(command);

LLAP.bMsgReceived = false; // if we do not clear the message flag then message processing will be blocked

}

return messageAcknowledgement;

}

***Since the library used for the messaging system does not include a method to package a command to a given ID , one had to be created. Taking in a command and an ID of the recipient the message will be converted to upper case and check if the message is a suitable size. If the message is too small padding is added so that messages are kept to the guidelines.***

boolean LLAPmessageSender(String ID,String message){

//takes in recipiant ID and the Message

if(message.length()<=9){

// message is needed to be less than 9 characters

String command = "a";

int messageLength = 1;

command.concat(ID);

//required to be upper case so can be seen

message.toUpperCase();

command.concat(message);

messageLength = command.length();

//add padding

while(messageLength <12){

command.concat("-");

messageLength = command.length();

}

Serial.println(command);

//return true meaning the message was created okay

return true;

}

//return true meaning the message was too long

return false;

}

***Tested sleep mode function – sleeps device for a second***

void sleepMode(){

delay(10); // allow radio to finish sending

digitalWrite(4, HIGH); // pull sleep pin high to enter SRF sleep 2

LLAP.sleepForaWhile(1000); // sleep ATmega328 for 5s (ms)

digitalWrite(4, LOW); // when ATmega328 wakes up, wake up SRF Radio

delay(10);

}

***Gets data from the input pin and passes this to the message sender with the Hubs ID to be sent***

void sendSensorData(){

//read data

sensorValue = analogRead(1);

//if(sensorValue!= lastReading){

lastReading = sensorValue;

//send input

String valCommand = "VAL:";

// sending value from the resistor

valCommand.concat(String(sensorValue));

LLAPmessageSender("A1",valCommand);

//}

# Code Explanation:

***Hub***

#include <LLAPSerial.h>

#define DEVICEID "A1" // this is the LLAP device ID

#include <LCD16x2.h>

#include <Wire.h>

//LCD SCREEN

LCD16x2 lcd;

int buttons;

String commandBuilder = "";

String lastCommand;

***Set up – makes use of various pins and activating the radio***

//Led on;

boolean isLedOn = false;

void setup(){

pinMode(12, OUTPUT); // initialize pin 12 as digital output (LED)

pinMode(8, OUTPUT); // initialize pin 8 to control the radio

digitalWrite(8, HIGH); // select the radio

Serial.begin(115200); // start the serial port at 115200 baud

delay(1000); // allow the radio to startup

LLAP.init(DEVICEID);

Serial.println("STARTED");

//Start up

Serial.print("ABCDEFGHIJKLMNOPQRSTUVWX");

Serial.flush();

Wire.begin();

//LCD setup

setUpLCD();

}

***Displaying data to the lcd on setup***

void setUpLCD(){

//setup LCD

lcd.lcdClear();

lcd.lcdGoToXY(1,1);

lcd.lcdWrite("LED:");

lcd.lcdGoToXY(1,2);

lcd.lcdWrite("VALUE:");

}

***Main loop, will check for messages being sent to the hub and check if a user has pressed any of the buttons on the LCD screen using the LCD library.***

void loop(){

//main loop

//Check for messages

LLAPmessageReceiver();

//check if the button has been pressed

checkButtonPress();

delay(100);

}

***Hubs version of controllers command handler, will use the extracted command from an sent message and compare it to a list of possible commands. IF one is matched will perform some task. In this case we check for a values being sent then output the values to the screen, making sure to send a acknowledgement to the controller.***

void commandHandler(String incomingCommand){

//takes in the command

//splits the comand from the data

String command = incomingCommand.substring(0,4);

String dataVal = incomingCommand.substring(4);

//checks commands against list

if(command =="VAL:"){

// converts data to a formate needed for the LCD

char dataPointer[20];

dataVal.toCharArray(dataPointer, 6);

lcd.lcdGoToXY(8,2);

lcd.lcdWrite(dataPointer);

//displays the value of the resistance

if(isLedOn){

//send the on state

LLAPmessageSender("A2","DONEON");

}else{

LLAPmessageSender("A2","DONEOFF");

}

}else{

//command does not meet up to one expected for the hub

LLAPmessageSender("A2",lastCommand);

Serial.println("COMMAND NOT RECOGNISED");

//Serial.println(command);

}

}

***Checks for an incoming message, if so will extract the message and send the string to the command handler. This is done to reduce the amount of code in one function.***

void LLAPmessageReceiver(){

if (LLAP.bMsgReceived) {

//collects message

//Serial.println(LLAP.sMessage);

String command = LLAP.sMessage;

commandHandler(command);

LLAP.bMsgReceived = false; // if we do not clear the message flag then message processing will be blocked

}

}

***Hubs version of a command builder to a custom ID, as used in the controller also.***

boolean LLAPmessageSender(String ID,String message){

//takes in recipiant ID and the Message

lastCommand = message;

if(message.length()<=9){

// message is needed to be less than 9 characters

String command = "a";

int messageLength = 1;

command.concat(ID);

//required to be upper case so can be seen

message.toUpperCase();

command.concat(message);

messageLength = command.length();

//add padding

while(messageLength <12){

command.concat("-");

messageLength = command.length();

}

Serial.println(command);

//return true meaning the message was created okay

return true;

}

//return true meaning the message was too long

return false;

}

***Checks if a button on the LCD screen has been pressed, if will set a Boolean to on or off that corresponds to the LED on the controller being on. This Boolean is used in the acknowledgement to define the state of the LED on the controller. NOTE : This used to send a message of its own , however was more efficient to combine this with the acknowledgement .***

void checkButtonPress(){

//gets data for the buttons

buttons = lcd.readButtons();

//moves to x = 6 y = 1 possition on the LCD

//Displays if the buton is pressed

lcd.lcdGoToXY(6,1);

if(buttons & 0x01){

}else{

if(!isLedOn){

lcd.lcdWrite("ON ");

//LLAPmessageSender("A2","ON");

isLedOn = true;

}

}

if(buttons & 0x02){

}else{

if(isLedOn){

lcd.lcdWrite("OFF");

//LLAPmessageSender("A2","OFF");

isLedOn = false;

}

}

}