XinoRF - Learning the basics

Category: Ciseco product documentation
Last Updated on Wednesday, 22 May 2013 08:41

XINO RF - LETS GET STARTED WITH SOME FUN

The aim of this document, is to give you a few simple code examples to show just how easy "doing" radio really is with Ciseco kit.

If you at all unsure about how things fit together, please read the Ciseco Product Introduction and the Getting Started Guide.

You can refer to the XinoRF general description, technical data, and troubleshooting guide for detailed information on the XinoRF.

The XinoRF is an Arduino UNO R3 compatible electronics development board with an onboard 2-way Ciseco SRF data radio, which supports over-the-air programming. We will cover programming over the air in later documents because it's easy to do but just as easy to get wrong by missing a setting.

For the experiments in this document, you will need the following:

1. 1 x XinoRF
2. 1 x USB mini cable
3. A PC set up to work with Ciseco hardware (see this guide)
4. Download and install the Arduino IDE (integrated development editor) from http://arduino.cc/en/Main/Software# on your PC
5. 1 x Ciseco SRF stick for your PC
6. For the temprature sensor example, you'll need a XinoRF starter kit or a thermistor with a 10kOhm resistor.

LETS WRITE SOME CODE!

The first example just had to be "blink" this is the standard example from within the Arduino IDE, this works just the same on a XinoRF

```
void setup() {

  pinMode(13, OUTPUT);     // initialize pin 13 as digital output (LED)

}

void loop() {

  digitalWrite(13, HIGH);   // set the LED on

  delay(1000);              // wait for a second

  digitalWrite(13, LOW);    // set the LED off

  delay(1000);              // wait for a second

}
```

You can extend this program to send the LED status via the on-board SRF on the radio network:

```
/*
  BlinkwithRadio
  Turns on an LED on for one second, then off for one second, repeatedly.
  Turns on the SRF radio and reports LED status.
  This example code is based on Blink, which is in the public domain.
 */
void setup() {
  pinMode(13, OUTPUT);   // initialize pin 13 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 (correct for XinoRF and RFu, if using XRF + Arduino you might need 9600)
}
void loop() {
  digitalWrite(13, HIGH);   // set the LED on
  Serial.print("LED ON");
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // set the LED off
  Serial.print("LED OFF");
  delay(1000);              // wait for a second
}
```

When you run this program should see the messages "LED ON" and LED OFF" on a serial monitor that is listening on the radio network. You should use a URF or SRF stick as a receiver for your PC, Mac or Linux system. If you are using a PC, it is worth using the serial monitor of our XRF Config Manager (XCM). On the Mac or Linux, use your favourite serial monitor.

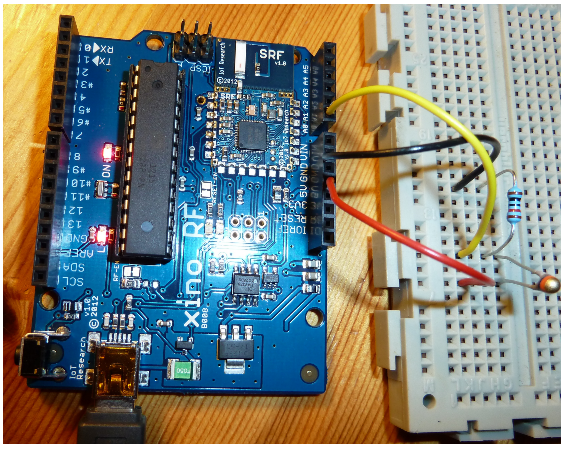
Let's now turn the LED on and off via a command received by radio. The example below switches the LED on when it receives a “1” and off when a “0” is received. It also sends a message to show the current status of the LED, each time it receives a command.

```
//
// Turn on/off the LED on pin 13 by command received from the radio
//  0 turns the LED off
//  1 turns the LED on
//  any other character sent has no effect
//
byte msg;  // the command buffer
void setup()
{
  pinMode(13, OUTPUT);   // initialize pin 13 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 (correct for XinoRF and RFu, if using XRF + Arduino you might need 9600)

  Serial.print("STARTED");
}
void loop()
{
  if (Serial.available()>=1) // character received
  {
    msg = (char)Serial.read();
```

```
    if (msg == '0')  // turn LED off
    {
      digitalWrite(13, LOW);
      Serial.print("  LED OFF  ");
    }
    else if (msg == '1')  // turn LED on
    {
      digitalWrite(13, HIGH);
      Serial.print("  LED ON   ");
    }
  }
}
```

Next let's try and read the temperature with a thermistor and send it over the radio. For this we need a 10kOhm thermistor and a 10kOhm resistor in series between the 5V output of the XinoRF and Ground. You then connect the mid point between the two components to input Analog 0 on the Arduino. Here is a photo of my set-up:



The code for reading the temperature and sending it out is as follows:

```
// ReadThermistor
// Reads the temperature via a Thermistor set up
// Switches the LED on when the temperature goes over 20C
// Sends the value periodically over the radio
// Thermistor NTCLE100E3103JB0; 10kOhm at 25C or 298.15K, Beta value 3977K

#include <math.h>

// Connections:
// Thermistor connected between 5V and Analog 0 (A0)
// 10kOhm resistor connected between Analog 0 and ground

void setup()
{
  pinMode(13, OUTPUT);    // initialize pin 13 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 (correct for XinoRF and RFu, if using XRF + Arduino you might need 9600)
  Serial.print("STARTED");// transmit started packet
}

float Thermistor(int ADCvalue)
{
  // calculate the temperature from an ADC value
  float T;                    // temperature
  int Beta = 3977;            // beta value for the thermistor
  float Rtemp = 25.0 + 273.15;  // reference temperature (25C)
  float Rresi = 9775.0;         // reference resistance at reference temperature - adjust to calibrate
  float Rtherm = (1024.0/ADCvalue - 1)*10000;  // value of the resistance of the thermistor
  T = Rtemp*Beta/(Beta+Rtemp*(log(Rtherm/Rresi)));
                       // see http://en.wikipedia.org/wiki/Thermistor for an explanation of the formula
  T = T - 273.15;               // convert from Kelvin to Celsius
  return T;
}

void loop()
{
  float temp = Thermistor(analogRead(0));  // read sensor and convert to temperature
  if (temp > 20) {
    digitalWrite(13, HIGH);  // turn LED ON
  }
  else digitalWrite(13, LOW);
  Serial.println(temp);
  delay(10000);
}
```

You can calibrate your thermometer by adjusting the reference resistance (Rresi) value in the Thermistor function in the sketch. Start by using the resistance of the thermistor at 25 degrees C or with 10 kOhm if you cannot measure it accurately. Then adjust it up and down to get the right reading, perhaps comparing with a reference thermometer if you have one. As you can see, we also turn the LED on when the temperature reaches 20C and off again when it falls below. It was warm in my lab when I took the picture, so my LED was ON.

The trouble with this program is that if you have more than one temperature sensor, you won't know which temperature is read at what sensor. Moreover, if you have sensors that read different things (temperature, light, etc.) then you don't know what numbers relate to what measurements.

To solve these types of problem, you could add all this information in the message in an ad hoc sort of way. You could send: "The temperature in the lab is 19.01 degrees C". But you would have to upload a different sketch to each device .... tedious. Instead, we devised a very simple protocol called LLAP. The human readable messages in LLAP are exactly 12 ASCII characters (bytes) long, and every message starts with a lower case 'a', followed by two characters that indicate the device identity. Following that there are 9 characters (bytes) for the actual message itself. So, if our device identity was XX and it was sending temperature, the message we would want to receive is

aXXTMPA19.01

meaning temperature sensor A on device XX shows 19.01 degrees Celsius. For more information on LLAP take a look at the LLAP introduction.

Here is some simple code for an LLAP message compliant thermometer:

```
/*
   ReadThermistorLLAP
   Reads the temperature via a Thermistor set up
   Switches the LED on when the temperature goes over 20C
   Sends the value periodically over the radio in an LLAP message
   Thermistor NTCLE100E3103JB0; 10kOhm at 25C or 298.15K, Beta value 3977K
   Connections:
      Thermistor connected between 5V and Analog 0 (A0)
      10kOhm resistor connected between Analog 0 and ground
*/

#include <math.h>
#define deviceID1 'X'     // first character of device identifier
#define deviceID2 'X'     // second character of device identifier
String hdr = "";          // message header

void setup()
{
  pinMode(13, OUTPUT);    // initialize pin 13 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 (correct for XinoRF and RFu, if using XRF + Arduino you might need 9600)
  hdr = hdr + "a" + deviceID1 + deviceID2;          // message header
  Serial.print(hdr + "STARTED");// transmit started packet
}

float Thermistor(int ADCvalue)
{
  // calculate the temperature from an ADC value
  float T;                      // temperature
  int Beta = 3977;              // beta value for the thermistor
  float Rtemp = 25.0 + 273.15;  // reference temperature (25C)
  float Rresi = 9775.0;         // reference resistance at reference temperature - adjust to calibrate
  float Rtherm = (1024.0/ADCvalue - 1)*10000;  // value of the resistance of the thermistor
  T = Rtemp*Beta/(Beta+Rtemp*(log(Rtherm/Rresi)));
              // see http://en.wikipedia.org/wiki/Thermistor for an explanation of the formula
  T = T - 273.15;               // convert from Kelvin to Celsius
  return T;
}

void loop()
{
  static char tempbuffer[4];        // to store the ASCII chars for temp
  double temp = Thermistor(analogRead(0));  // read sensor and convert to temperature
  if (temp > 20) {
    digitalWrite(13, HIGH);         // turn LED ON
  }
  else digitalWrite(13, LOW);
  dtostrf(temp,4,2,tempbuffer);     // convert double to string
  Serial.print(hdr + "TMPA" + tempbuffer); // send message
  delay(10000);
}
```

A variation on LLAP is Pinata, a protocol to read and write to specific pins of your micro-controller. It uses the same message format as LLAP. Below is the code for the XinoRF, where the device id has been set to XX (change it to what you want).

Useful first test - Wiggling the on board LED on D13 by wireless (you will need a XRF, URF or SRF Stick on your PC and terminal software eg XCM)

Send in a single burst (not type as that's too slow) the following text commands, you will get a confirmation back each time

aXXD13OUTPUT `Sets pin D18 to an output`

aXXD13HIGH-- `Turns on the LED`

aXXD13LOW--- `Turns off the LED`

```
//
// LLAP - Lightweight Local Automation Protocol
//
// Arduino pinata code for the XinoRF
//

#define deviceID1 'X'
#define deviceID2 'X'

String msg;                              // storage for incoming message
String reply;  // storage for reply

void setup()
{
  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 (correct for XinoRF and RFu, if using XRF + Arduino you might need 9600)
  String hdr = hdr + "a" + deviceID1 + deviceID2;         // message header
  Serial.print(hdr + "STARTED");// transmit started packet
}

void loop() // repeatedly called
{
  if (Serial.available() >= 12) // have we got enough characters for a message?
  {
    if (Serial.read() == 'a') // start of message?
    {
    msg = "a";
    for (byte i=0; i<11; i++)  // 11 characters in the message body
    {
      msg += (char)Serial.read();
    }
    if (msg.charAt(1) == deviceID1 && msg.charAt(2) == deviceID2)  // message is for us
    {
      reply = msg;
      msg = msg.substring(3);
      if (msg.compareTo("HELLO----") == 0)
      {
        ;  // just echo the message back
      }
      else  // it is an action message
      {
```

```
        byte typeOfIO;
        byte ioNumber;
        typeOfIO = msg.charAt(0);
        ioNumber = (msg.charAt(1) - '0') * 10 + msg.charAt(2) - '0';
        msg = msg.substring(3);
        if (msg.compareTo("INPUT-") == 0)
        {
          if (ioNumber > 1) pinMode(ioNumber,INPUT);
        }
        else if (msg.compareTo("OUTPUT") == 0)
        {
          if (ioNumber > 1) pinMode(ioNumber,OUTPUT);
        }
        else if (msg.compareTo("HIGH--") == 0)
        {
          if (ioNumber > 1) digitalWrite(ioNumber,HIGH);
        }
        else if (msg.compareTo("LOW---") == 0)
        {
          if (ioNumber > 1) digitalWrite(ioNumber,LOW);
        }
        else if (msg.startsWith("PWM"))
        {
          byte val = ((msg.charAt(3) - '0') * 10 + msg.charAt(4) - '0') * 10 + msg.charAt(5) - '0';
          if (ioNumber > 1) analogWrite(ioNumber,val);
        }
        else if (msg.compareTo("READ--") == 0)
        {
          reply = reply.substring(0,6);
          if (typeOfIO == 'A')
          {
            int val = analogRead(ioNumber);
            reply = reply + "+" + val;

          }
          else
          {
            byte val = digitalRead(ioNumber);
            if (val)
            {
              reply = reply + "HIGH";
            }
            else
            {
              reply = reply + "LOW";
            }
          }
        }
        else
          reply = reply.substring(0,3) + "ERROR----";
      }
    }
    if (reply.length() < 12)
    {
      byte i = 12-reply.length();
      while (i--) reply += '-';
    }
        Serial.print(reply);
      }
  }
}
```

Be careful when using pin 8 because it controls the radio! The next version of this code should probably test for the use of pin 8 and return an error!

By building on the examples above, you are now ready to explore the wonderful world of wireless measurement, interrogation and control with the XinoRF.

Also take a look at the LLAP library for how to use `serialEvent` to more reliably receive messages.