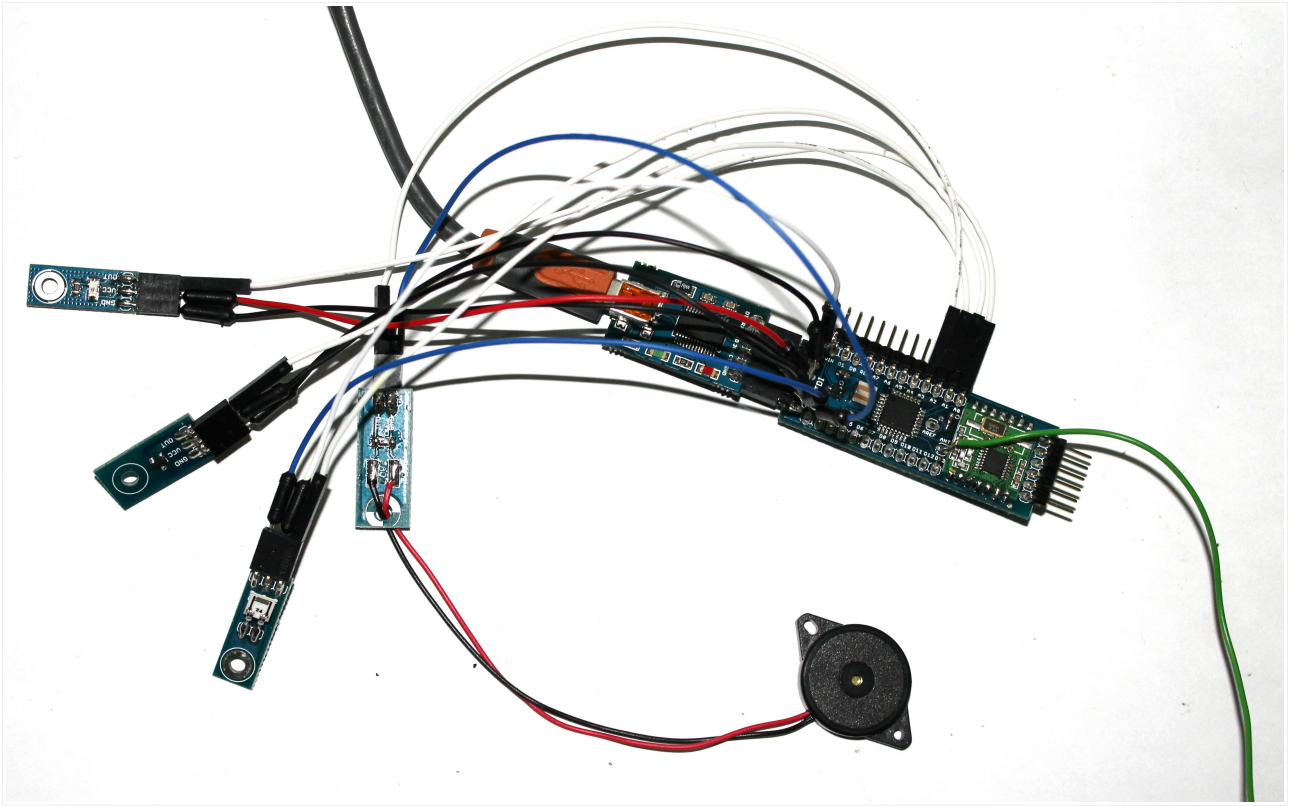
Reduino Radio + Senzori Brick

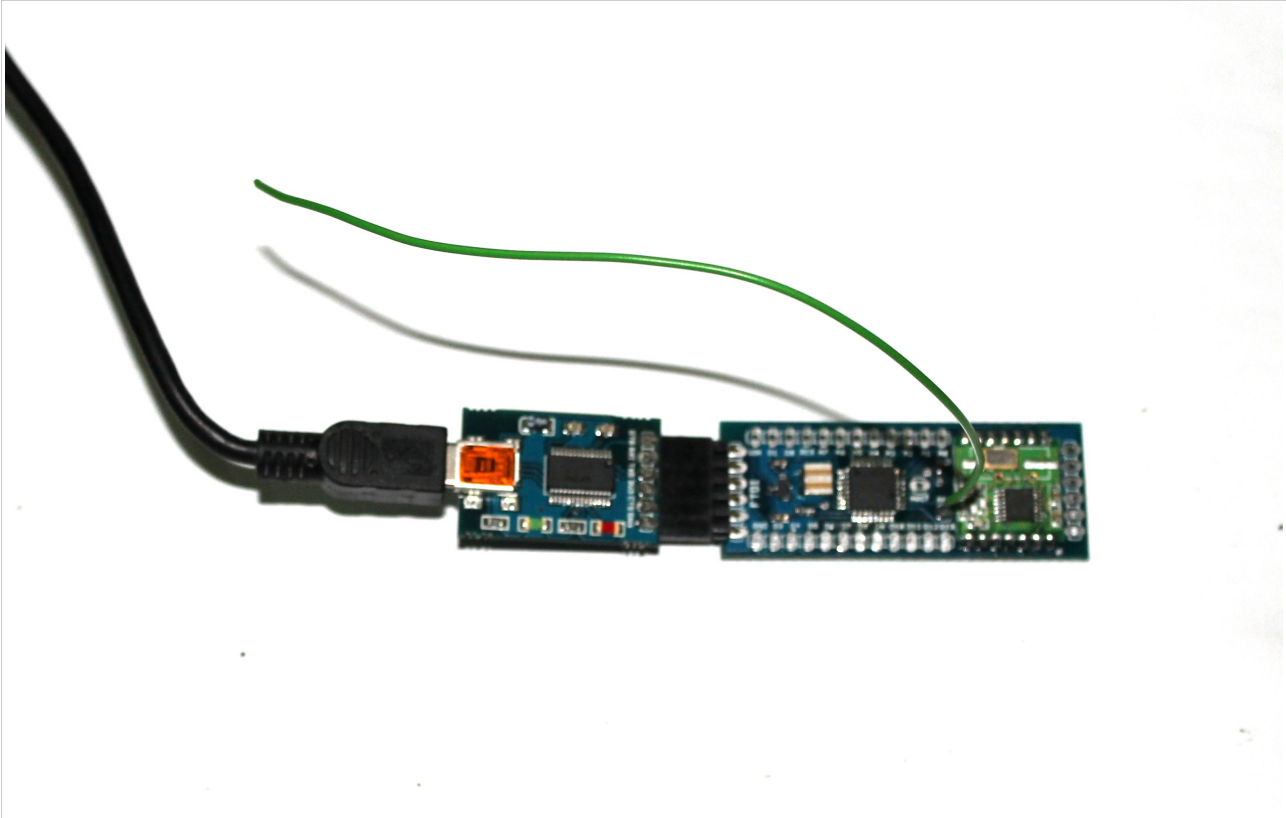
Ce este Reduino Radio ?

Reduino Radio este o placa similara cu Arduino Pro Mini 328 – 3.3/8Mhz deoarece are exact acelasi procesor ca si Arduino Pro Mini – Atmega328 in capsula SMD, are dimensiuni asemanatoare, este usor de programat si ofera aproape aceleasi facilitati pe care le ofera placa Arduino Pro Mini 328.

Pe langa similitudinea cu placa Arduino Pro Mini, placa Reduino Radio este echipata cu un transmitator radio RFM12B. Acest lucru iti permite sa dezvolti proiecte in care placa Reduino comunica wireless , adica transmite sau primeste informatii la distanta.

Spre exemplu, se poate dezvolta o retea de senzori wireless care comunica printr-unul sau mai multe noduri diverse informatii cum ar fi: temperatura, viteza, curentul, distanta, acceleratia, s.a.m.d.





Cum se construieste o retea de senzori wireless?

Pentru a construi o retea minimala de senzori wireless vei avea nevoie de cel putin 2 placi Reduino Radio si de o serie de senzori brick. Placile Reduino vor comunica intre ele valorile masurate de catre senzori. Cu alte cuvinte, prima placa la care se afla senzorii conectati va transmite catre cealalta placa o serie de valori numerice.

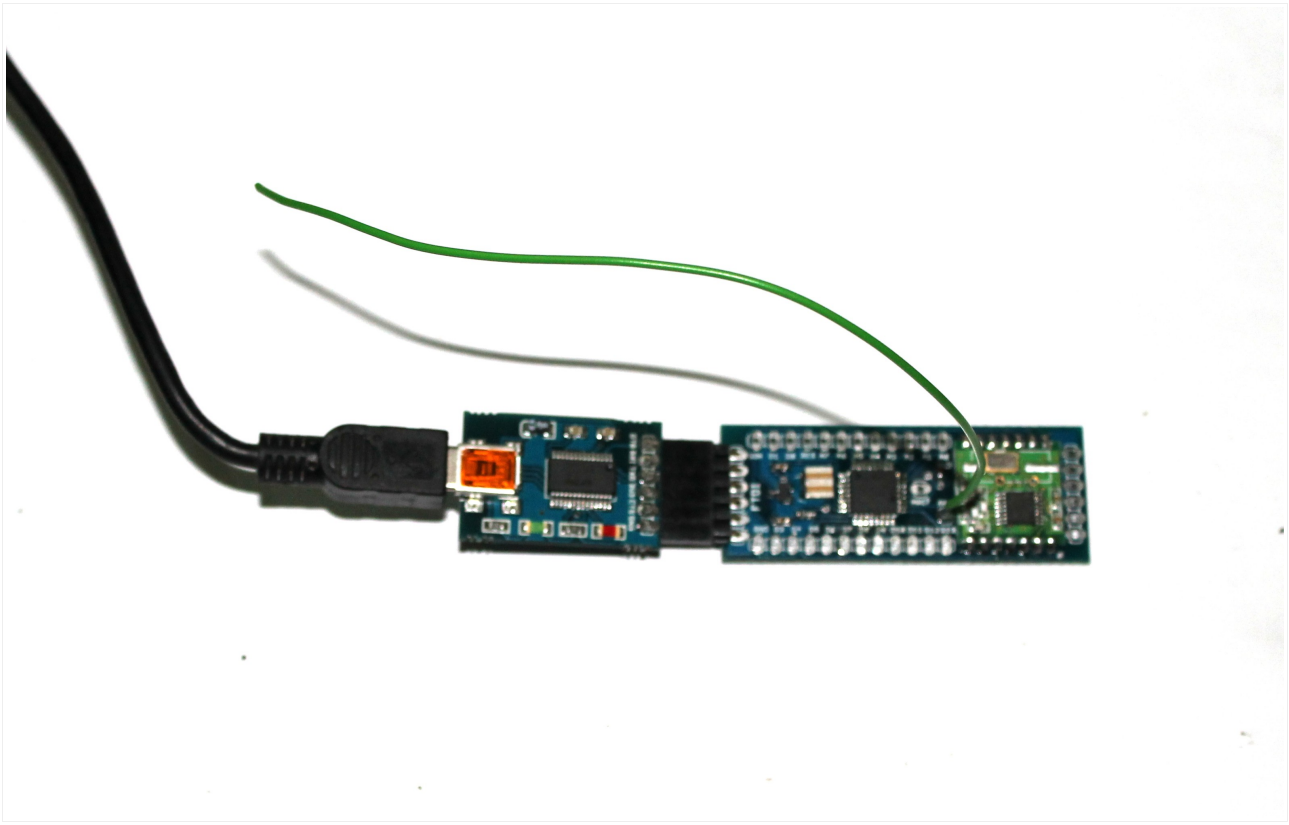
Poti conecta o gama variata de senzori: senzor de alcool brick, senzor de apasare brick, senzor de indoire brick, senzor de lumina brick, senzor magnetic brick, senzor de temperatura brick, senzor de umiditate brick, senzor de vibratii brick, accelerometre, senzori de infrarosu, senzori de vreme, senzori de distanta, senzori ID, giroscop, IMU, inclinare sau senzori de lichide.

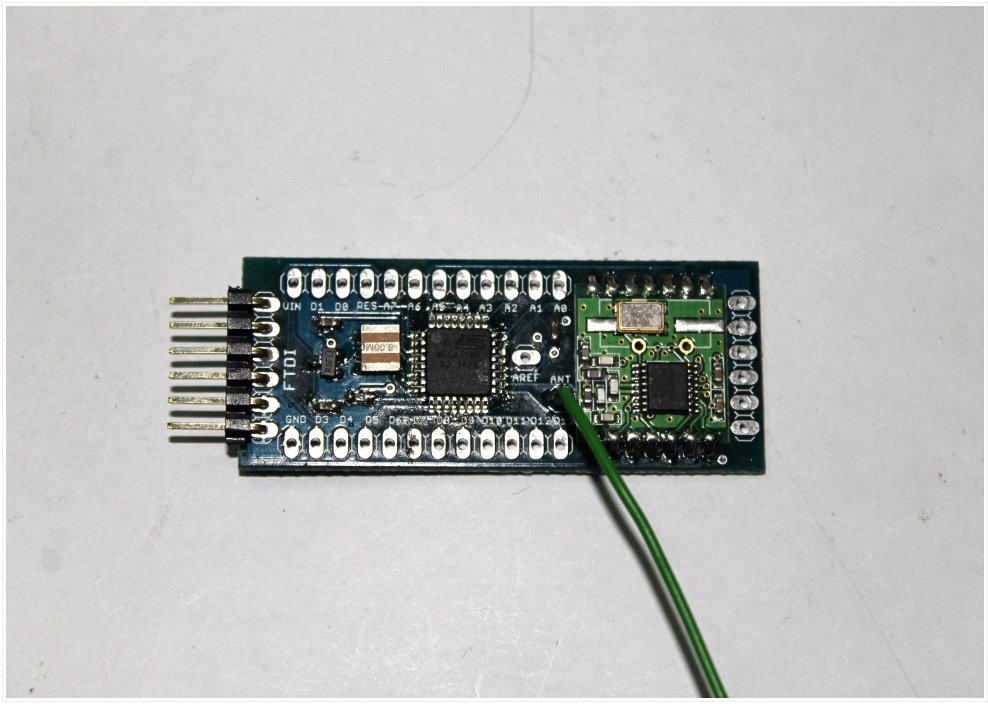
Senzorii brick se conecteaza la placa Reduino Radio conform urmatorului tabel iar :

Senzor brick pin VCC	Alimentator pin 5V
Senzor brick pin GND	Alimentator pin GND
Senzor brick pin OUT	Reduino Radio pin A<0..7>

Cealalta placa Reduino Radio va arata ca in imaginea de mai jos iar

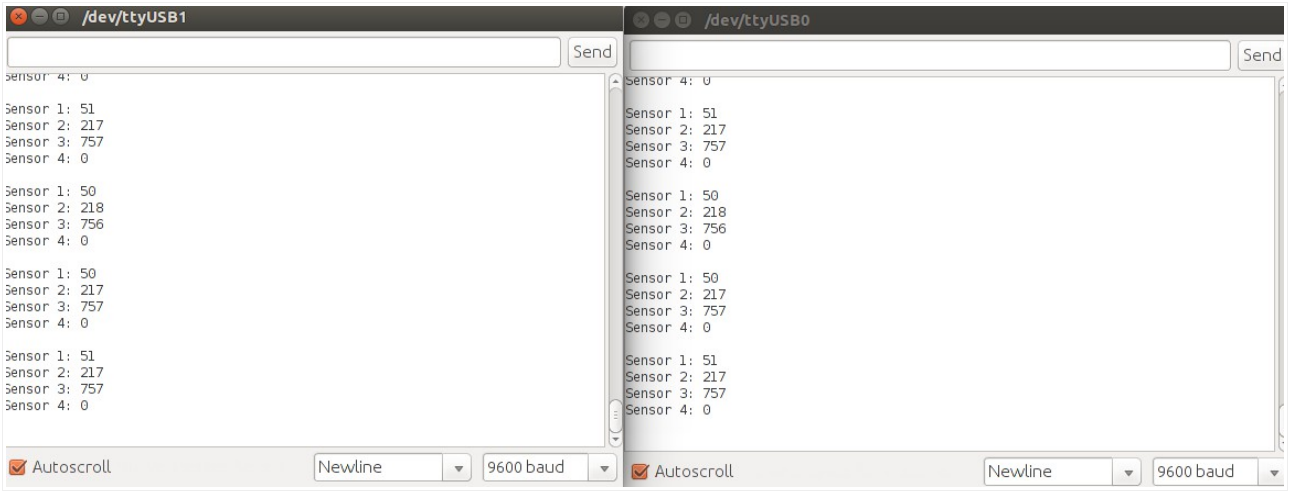
ambele placute necesita un fir de antena lipit la pinul marcat cu ANT.:





Pentru fiecare placa se incarca sketch-ul aferent de mai jos, iar imediat dupa incarcarea codurilor sursa vei deschide Monitoarele Seriale (vezi imaginea de mai jos). Ceea ce transmite Monitorul Serial din stanga va aparea in

Monitorul Serial din dreapta.



Sketch-ul Arduino pentru emitator

```
//Simple RFM12B wireless demo - transimtter - no ack
//Glyn Hudson openenergymonitor.org GNU GPL V3 7/7/11
//Credit to JCW from Jeelabs.org for RFM12

#include <JeeLib.h>  //from jeelabs.org

#define myNodeID 10          //node ID of tx (range 0-30)
#define network     210      //network group (can be in the range
1-250).
#define freq RF12_433MHZ     //Freq of RF12B can be RF12_433MHZ,
RF12_868MHZ or RF12_915MHZ. Match freq to module

#define sensorOne A0
#define sensorTwo A1
#define sensorThree A2
#define sensorFour A3


typedef struct { int sensor1, sensor2, sensor3, sensor4; }
PayloadTX;      // create structure - a neat way of packaging data
for RF comms
PayloadTX emontx;

void setup() {
  rf12_initialize(myNodeID,freq,network);   //Initialize RFM12 with
settings defined above
Serial.begin(9600);
Serial.println("RFM12B Transmitter - Simple demo");
```

```
 Serial.print("Node: ");
 Serial.print(myNodeID);
 Serial.print(" Freq: ");
 if (freq == RF12_433MHZ) Serial.print("433Mhz");
 if (freq == RF12_868MHZ) Serial.print("868Mhz");
 if (freq == RF12_915MHZ) Serial.print("915Mhz");
 Serial.print(" Network: ");
 Serial.println(network);

}

void loop() {
  emontx.sensor1=analogRead(sensorOne);
  emontx.sensor2=analogRead(sensorTwo);
  emontx.sensor3=analogRead(sensorThree);
  emontx.sensor4=analogRead(sensorFour);

  int i = 0; while (!rf12_canSend() && i<10) {rf12_recvDone(); i+
+;}
    rf12_sendStart(0, &emontx, sizeof emontx);

  Serial.print("Sensor 1: "); Serial.println(emontx.sensor1);
  Serial.print("Sensor 2: "); Serial.println(emontx.sensor2);
  Serial.print("Sensor 3: "); Serial.println(emontx.sensor3);
  Serial.print("Sensor 4: "); Serial.println(emontx.sensor4);
  Serial.println("  ");

  delay(2000);
}
```

Sketch-ul Arduino pentru receptor

```
//Simple RFM12B wireless demo - Receiver - no ack
//Glyn Hudson openenergymonitor.org GNU GPL V3 12/4/12
//Credit to JCW from Jeelabs.org for RFM12

#include <JeeLib.h>

#define myNodeID 30          //node ID of Rx (range 0-30)
#define network     210      //network group (can be in the range
1-250).
#define freq RF12_433MHZ     //Freq of RF12B can be RF12_433MHZ,
RF12_868MHZ or RF12_915MHZ. Match freq to module


typedef struct { int sensor1, sensor2, sensor3, sensor4; }
PayloadTX;      // create structure - a neat way of packaging data
for RF comms
```

```
PayloadTX emontx;

const int emonTx_NodeID=10;            //emonTx node ID

void setup() {

  rf12_initialize(myNodeID,freq,network);   //Initialize RFM12 with
settings defined above
  Serial.begin(9600);
  Serial.println("RF12B demo Receiver - Simple demo");

 Serial.print("Node: ");
 Serial.print(myNodeID);
 Serial.print(" Freq: ");
 if (freq == RF12_433MHZ) Serial.print("433Mhz");
 if (freq == RF12_868MHZ) Serial.print("868Mhz");
 if (freq == RF12_915MHZ) Serial.print("915Mhz");
 Serial.print(" Network: ");
 Serial.println(network);
}

void loop() {


 if (rf12_recvDone()){
  if (rf12_crc == 0 && (rf12_hdr & RF12_HDR_CTL) == 0) {

    int node_id = (rf12_hdr & 0x1F);         //extract nodeID from
payload

    if (node_id == emonTx_NodeID)  {             //check data is
coming from node with the corrct ID
        emontx=*(PayloadTX*) rf12_data;            // Extract the
data from the payload
       Serial.print("Sensor 1: "); Serial.println(emontx.sensor1);
       Serial.print("Sensor 2: "); Serial.println(emontx.sensor2);
       Serial.print("Sensor 3: "); Serial.println(emontx.sensor3);
       Serial.print("Sensor 4: "); Serial.println(emontx.sensor4);
       Serial.println("  ");
  }
 }
}
}
```