J. Basic. Appl. Sci. Res., 3(2s)187-195, 2013
© 2013, TextRoad Publication

ISSN 2090-4304
Journal of Basic and Applied Scientific Research
www.textroad.com

Designing and Construction of the Transceiver System by Using a New Generation of Telecommunication and Communication Devices in order to Forecast Radio Weather

Mohamadreza Mohamadzadeh Shadmehri[1], Eqbal Fallah Heravi [2], Fatemeh Peyravi [3], Ali Mohammadian[4*]

[1] Department of Electronic Engineering, Khorasan Razavi, Neyshabur, Science and Research branch, Islamic Azad University, Neyshabur, Iran
[2,3,4] Department of Electronic Engineering, Khorasan Shomali, Bojnourd, Science and Research branch, Islamic Azad University, Bojnourd, Iran

Received: June 10 2013
Accepted: July 10 2013

ABSTRACT

In the transmitter, temperature and humidity are measured and sent to the micro, then into the micro are turned to digital and while the displays on the LCD module through the SPI protocol are sent to RFM12, FSK modulation by module, then this information is sent to the receiver module.
At the receiver, the data received by means of the module via the same protocol SPI will be transferred to the micro and micro also through the USART Protocol and MAX232 IC pass data to a computer and the Terminal software Code Vision displayed.
KEYWORDS: microcontroller, MAX232 IC, USART protocol , SPI protocol

INTRODUCTION

1) Introduction of remote and transmitter system:

In the remote part of the system that is placed in the target environment sensor, processor and transmitter circuits there will be where environmental information (temperature, pressure and humidity) at any moment by the corresponding analog sensors are received, and analog to digital converters through the internal microcontroller are converted into digital signals**.** Processed information first to the Crystal Viewer available on the remote circuit has been sent and will be displayed. Then through the protocols of the SPI (Serial Peripheral Interface) transceiver module (rfm 12) will follow the same protocol is sent. This module transfer information byte to byte to the receiver located at the monitoring center.

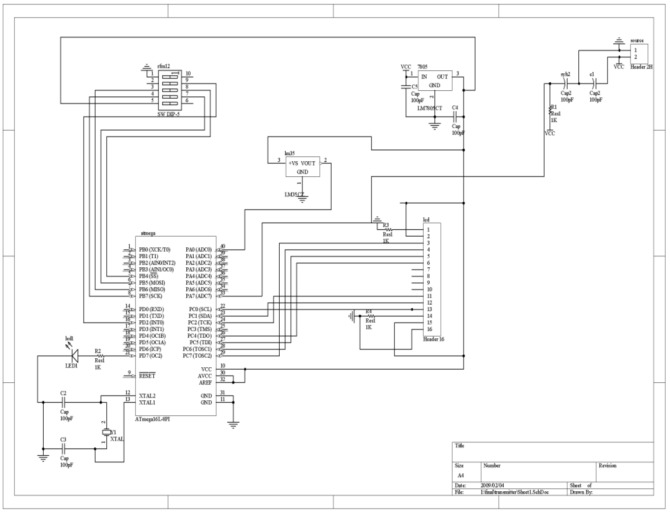

Figure1: Transmitter Circuit

2) Introduction the performance of receiver:

This circuit is that there is a maintenance center in the information sent by the sender are received through the SPI interface to a microcontroller circuit in the receiver.in the receiver moves as well as data recovery

***Corresponding Author:** Mohamadreza Mohamadzadeh Shadmehri, Department of Electronic Engineering, Khorasan Razavi, Neyshabur, Science and Research branch, Islamic Azad University, Neyshabur, Iran. m_mohamadzadeh.talent@yahoo.com or mohamadreza_mohamadzadeh@hotmail.com

through the Max232 IC connected to RS232 port to the PC then the information received through the special online software is displayed.

One advantage of this circuit is that it can be created with a small change on the circuit at any given moment than to have the remote management and monitoring circuit.

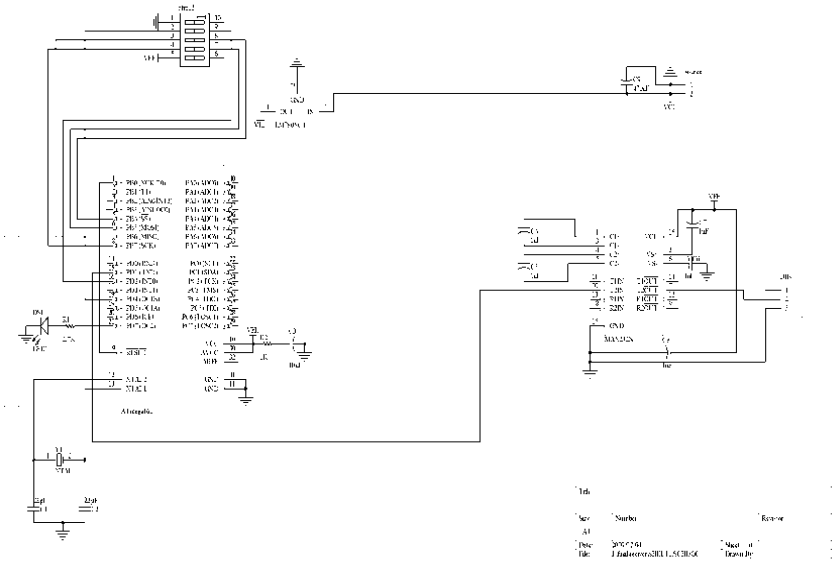

Figure2: Receiver Circuit

3) Transmitter program :

As was already stated in the sender's information circuit temperature, pressure and humidity sensors, analog to digital converter with corresponding micro reads and then to send the sender module as well as character LCD display to be sent to. For micro planning beginning with software Code wizard the following required codes active for:

4) Define chip and Crystal :

select the 4MHZ clock frequency and ATmega16L from chip type window**.**

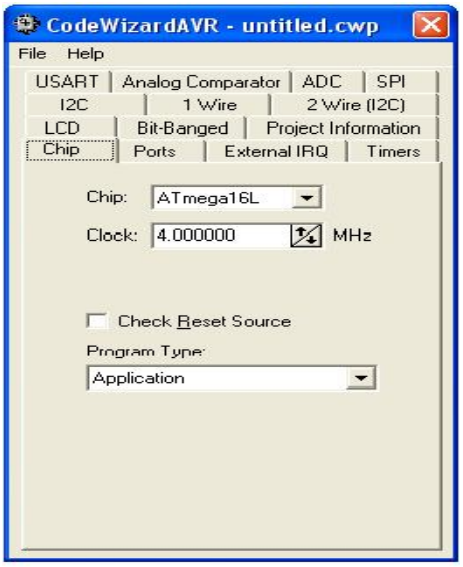

Figure3: Set the chip in code wizard

5) Input and Output ports:

Of the Windows Port, the status of the input and output ports. In this case, according to the module base is connected the transmitter to port B, base required to define output port B.

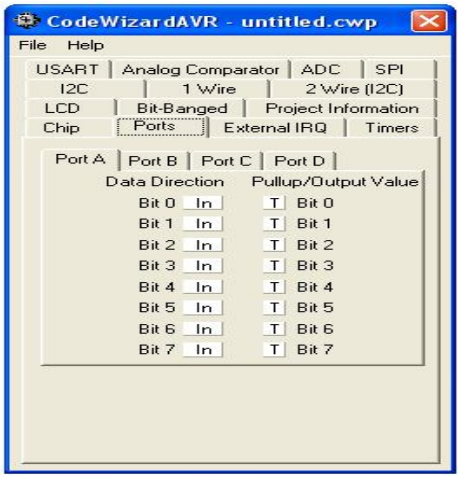

Figure4: Set the input/output port

6) External interrupts:

In the transmitter circuit for synchronize modules sender and micro need a source of external interruptions that required settings of External IRQ window.

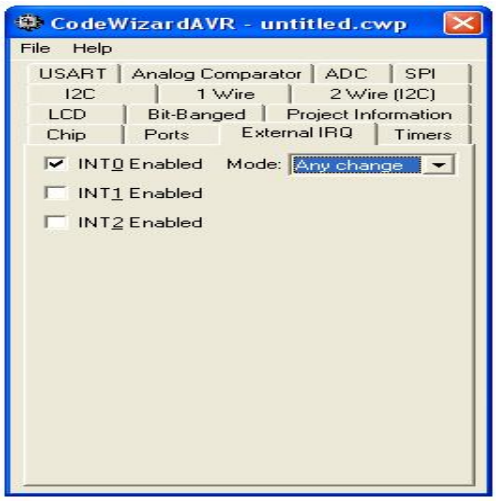

Figure5: Set the external interrupt

After doing the above steps of Windows File option to select Generate Save and Exit. After you create these codes by Code Vision Avr turn to write the main program. To do this first to ease the initial definitions as well as a series of calls to the functions required to be carried out as follows:

```
// Alphanumeric LCD Module functions
#asm
   .equ __lcd_port=0x15 ;PORTC
#endasm
#include <lcd.h>
#include <mega16.h>
#include <stdlib.h>
//#include <stdio.h>
#include <delay.h>
//#include <string.h>
#include <math.h>
#define DDR_IN          0
#define DDR_OUT         1
 #define PORT_SEL       PORTB
#define PIN_SEL         PINB
#define DDR_SEL         DDRB
 #define PORT_SDI       PORTB
#define PIN_SDI         PINB
#define DDR_SDI         DDRB
 #define PORT_SCK       PORTB
#define PIN_SCK         PINB
#define DDR_SCK         DDRB
        #define PORT_SDO        PORTB
        #define PIN_SDO         PINB
        #define DDR_SDO         DDRB
        #define PORT_IRQ        PORTD
        #define PIN_IRQ         PIND
        #define DDR_IRQ         DDRD
        #define PORT_DATA       PORTD
        #define PIN_DATA        PIND
        #define DDR_DATA        DDRD
         #define PB7                    7//--\
        #define PB6                     6//  |
        #define RFXX_SCK        7//  |
        #define RFXX_SDO        6//  |RF_PORT
        #define RFXX_SDI        5//  |
        #define RFXX_SEL        4//  |
        #define NC                      1//  |
        #define PB0                     0//--/
         #define SEL_OUTPUT()    DDR_SEL |=
(1<<RFXX_SEL)
        #define HI_SEL()        PORT_SEL|=
(1<<RFXX_SEL)
        (1<<RFXX_SEL)
```

Now we use set up and define the transmitter module of the boot program written by hope electronic company manufacturer RFM modules.

```
void RFsend_PORT_INIT(void)
{
  HI_SEL();
  HI_SDI();
  LOW_SCK();
  SEL_OUTPUT();
  SDI_OUTPUT();
  SDO_INPUT();
  SCK_OUTPUT();
}

nsigned int RFsend_WRT_CMD(unsigned int aCmd)
{
  unsigned char i;
  unsigned int temp=0;
  LOW_SCK();
  LOW_SEL();
  for(i=0;i<16;i++){
    temp<<=1;
    if(SDO_HI()){
      temp|=0x0001;
    }
    LOW_SCK();
    if(aCmd&0x8000)
      {
          HI_SDI();
      }
      else
      {
          LOW_SDI();
      }
    HI_SCK();
    aCmd<<=1;
  };
  LOW_SCK();
  HI_SEL();
  return(temp);
```

After you define the Write command to launch the module needs to write information on the desired addresses in the stability we have for this RFM function RF12send_INIT to create the basic definitions and set up the module according to the following definition:

```
void RF12send_INIT(void){
    RFsend_WRT_CMD(0x80D7);//EL,EF,433band,12.0pF
    RFsend_WRT_CMD(0x8239);//!er,!ebb,ET,ES,EX,!eb,!ew,DC
    RFsend_WRT_CMD(0xA640);//A140=430.8MHz
    RFsend_WRT_CMD(0xC647);//19.2kbps
    RFsend_WRT_CMD(0x94A0);//VDI,FAST,134kHz,0dBm,-103dBm
    RFsend_WRT_CMD(0xC2AC);//AL,!ml,DIG,DQD4
    RFsend_WRT_CMD(0xCA81);//FIFO8,SYNC,!ff,DR
    RFsend_WRT_CMD(0xC483);//@PWR,NO RSTRIC,!st,!fi,OE,EN
    RFsend_WRT_CMD(0x9850);//!mp,9810=30kHz,MAX OUT
    RFsend_WRT_CMD(0xE000);//NOT USE
    RFsend_WRT_CMD(0xC800);//NOT USE
    RFsend_WRT_CMD(0xC040);//1.66MHz,2.2V
}
```

According to the RFM register each of the lines above the module a AO function. After the initial definitions of the data sent to a function definition. According to the internal register module each phrase within the registry is written to send B8 address ready. That's why getting this entry with the upload function in mind, as we defined the following :

```
void RF12_SEND(unsigned char aByte)
{
  while(PIND&(1<<2));//wait for previously TX over
        RFsend_WRT_CMD(0xB800+aByte);  }
```

In the second line of this function, the phrase While to stop the previous module to send complete byte is used. After the call and write the sender module startup programs are for samples and get the required data from the sensors, two functions called readRH and Temperature For moisture and temperature reading environments, we define:

```
int readRH(void)
{
float t,k;
unsigned int t2;
   t=read_adc(1);
   t2 = 9 - t;
   t*= 820000;
   t = t/t2;
   k = log10(t);
   k = 4.45 - k;
   k = (float)(k/0.0485);
   return k;                }
int Temprature(void)
{
 f=read_adc(0);
  c1 =f>>1;
return c1;
```

Main function:

According to the setting in Code wizard as described earlier, many of the required settings in the form of function main () was produced by the following software:

```
        void main(void)
        {
        // Declare your local variables here
         unsigned int i;
         unsigned char str[50],c1,c2;
         int f;
        // Input/Output Ports initialization
        // Port A initialization
        // Func0=In Func1=In Func2=In Func3=In Func4=In
Func5=In Func6=In Func7=In
        // State0=T State1=T State2=T State3=T State4=T
State5=T State6=T State7=T
        PORTA=0x00;
        DDRA=0x00;
        // Port B initialization
        // Func0=In Func1=In Func2=In Func3=In Func4=In
Func5=In Func6=In Func7=In
        // State0=T State1=T State2=T State3=T State4=T
State5=T State6=T State7=T
        PORTB=0x00;
        DDRB=0x00;
        // Port C initialization
        // Func0=In Func1=In Func2=In Func3=In Func4=In
Func5=In Func6=In Func7=In
        // State0=T State1=T State2=T State3=T State4=T
State5=T State6=T State7=T
        PORTC=0x00;
        DDRC=0x00;
        // Port D initialization
        // Func0=In Func1=In Func2=In Func3=In Func4=In
Func5=In Func6=In Func7=In
        // State0=T State1=T State2=T State3=T State4=T
State5=T State6=T State7=T
        PORTD=0x00;
        DDRD=0x00;
        // Timer/Counter 0 initialization
        // Clock source: System Clock
        // Clock value: Timer 0 Stopped
        // Mode: Normal top=FFh
        // OC0 output: Disconnected
        TCCR0=0x00;
        TCNT0=0x00;
        OCR0=0x00;
        // Timer/Counter 1 initialization
        // Clock source: System Clock
        // Clock value: Timer 1 Stopped
        // Mode: Normal top=FFFFh
        // OC1A output: Discon.
        // OC1B output: Discon.
        // Noise Canceler: Off
        // Input Capture on Falling Edge
        TCCR1A=0x00;
        TCCR1B=0x00;
        TCNT1H=0x00;
        TCNT1L=0x00;
        OCR1AH=0x00;
        OCR1AL=0x00;
        OCR1BH=0x00;
        OCR1BL=0x00;
        // Timer/Counter 2 initialization
        // Clock source: System Clock
        // Clock value: Timer 2 Stopped
        // Mode: Normal top=FFh
        // OC2 output: Disconnected
        ASSR=0x00;
        TCCR2=0x00;
        TCNT2=0x00;
        OCR2=0x00;
        // External Interrupt(s) initialization
        // INT0: On
        // INT0 Mode: Low level
        // INT1: Off
        // INT2: Off
        // Timer(s)/Counter(s) Interrupt(s) initialization
        TIMSK=0x00;
        // Analog Comparator initialization
        // Analog Comparator: Off
        // Analog Comparator Input Capture by Timer/Counter
 1: Off
        // Analog Comparator Output: Off
        ACSR=0x80;
        SFIOR=0x00;
        // ADC initialization
        // ADC Clock frequency: 125.000 kHz
        // ADC Voltage Reference: AREF pin
        // ADC High Speed Mode: Off
        // ADC Auto Trigger Source: None
        ADMUX=ADC_VREF_TYPE;
        ADCSRA=0x85;
        SFIOR&=0xEF;
        // Global enable interrupts
        #asm("sei")
        #asm("cli");
        DDRD =0x80;
        for(i=0;i<10;i++)
        {
              PORTD.7 =! PORTD.7;
              delay_ms (50);
        }
        delay_ms(500);
        LEDG_ON();
        LEDR_ON();
        delay_ms(500);
        LEDG_OFF();
        LEDR_OFF();
        LEDG_OFF();
        LEDR_OFF();
        RFsend_PORT_INIT();
        RF12send_INIT();
        DDRD|=(1<<RF12_DATA);
        PORTD|=(1<<RF12_DATA);// SET nFFS pin HI when
 using TX register
        DDRD&=~(1<<2);       //PD2(INT0)
lcd_init(24);
```

Number of functions here after launching the module and LED off the first three times the amount that the AA has a package that the diagnosis module is the recipient of the information is ready to be sent to the sender and the recipient after sencron with the number of functions readRH () and temperature( )the moisture content Temperature (s) in the c1 and c2 of the characters.

In this case for the possibility of separation of coding in this way supports the recipient before we send the characters c1 is related to the temperature and the amount of that 20 before c2 (moisture content) submit the number 30. Recent functions for continuous repeat in a loop, and finally we put:

```
while(1) {
 RFsend_WRT_CMD(0x0000);//read status register
          RFsend_WRT_CMD(0x8239);//!er,!ebb,ET,ES,EX,!eb,!ew,DC
                    RF12_SEND(0xAA);//PREAMBLE
               RF12_SEND(0xAA);//PREAMBLE
               RF12_SEND(0xAA);//PREAMBLE
               RF12_SEND(0x2D);//SYNC HI BYTE
               RF12_SEND(0xD4);//SYNC LOW BYTE
               Temprature();
               c2 =readRH();
               RF12_SEND(c1);
               RF12_SEND(c2);
               delay_ms (500);
               PORTD.7 = ! PORTD.7;
               RF12_SEND(0xAA);//DUMMY BYTE
      RF12_SEND(0xAA);//DUMMY BYTE
          RF12_SEND(0xAA);//DUMMY
BYTE

RFsend_WRT_CMD(0x8201);
               //LCD MONITORING
                 lcd_gotoxy(0,0);
                 lcd_putsf("temprature:");
                 lcd_gotoxy(0,10);
                 lcd_puts(c1);
                 lcd_gotoxy(1,0);
                 lcd_putsf("Humidity: %");
                 lcd_gotoxy(1,10);
                 lcd_puts(c2);
            }
```

7) Receiver program:

In the primary circuit of the recipient after the definitions and call once again upon the functions similar to the functions of the transmitter module Builder to define the setup of:

```
// Alphanumeric LCD Module functions
#asm
  .equ __lcd_port=0x15 ;PORTC
  #endasm
#include <lcd.h>
#include <mega16.h>
#include <stdlib.h>
#include <stdio.h>
#include <delay.h>
 #define DDR_IN        0
#define DDR_OUT        1
 #define PORT_SEL      PORTB
#define PIN_SEL       PINB
#define DDR_SEL       DDRB
 #define PORT_SDI      PORTB
#define PIN_SDI       PINB
#define DDR_SDI       DDRB
 #define PORT_SCK      PORTB
#define PIN_SCK       PINB
#define DDR_SCK       DDRB
 #define PORT_SDO      PORTB
#define PIN_SDO       PINB
#define DDR_SDO       DDRB
#define PORT_IRQ      PORTD
#define PIN_IRQ       PIND
#define DDR_IRQ       DDRD
#define PORT_DATA     PORTD
#define PIN_DATA      PIND
#define DDR_DATA      DDRD
#define PB7           7//--\
#define PB6           6//  |
#define RFXX_SCK      7//  |
#define RFXX_SDO      6//  |RF_PORT
#define RFXX_SDI      5//  |
#define RFXX_SEL      4//  |
#define NC            1//  |
#define PB0           0//--/
 #define SEL_OUTPUT()   DDR_SEL |=
 (1<<RFXX_SEL)
#define HI_SEL()      PORT_SEL|=
 (1<<RFXX_SEL)
#define LOW_SEL()
 PORT_SEL&=~(1<<RFXX_SEL)
 #define SDI_OUTPUT()   DDR_SDI |=
 (1<<RFXX_SDI)
#define HI_SDI()      PORT_SDI|=
 (1<<RFXX_SDI)
#define LOW_SDI()
 PORT_SDI&=~(1<<RFXX_SDI)
 #define SDO_INPUT()    DDR_SDO&=
 ~(1<<RFXX_SDO)
#define LOW_SDO()     PORT_SDO&=
 (1<<RFXX_SDO)
#define SDO_HI()
 PIN_SDO&(1<<RFXX_SDO)
 #define SCK_OUTPUT()   DDR_SCK |=
#define LEDG_OFF()     PORTD|= (1<<6)
#define LEDR_ON()     PORTD&=~(1<<7)
#define LEDR_OFF()    PORTD|= (1<<7)
#define RF12_RES      PORTB.0
#define ON            0
#define OFF           1
bit edge;
float T1,T2;
// External Interrupt 0 service routine
interrupt [EXT_INT0] void ext_int0_isr(void)
{
// Place your code here
if (edge)
 {
   T2=TCNT1;
   TCNT1=0;
   // INT0 Mode: Falling Edge
   MCUCR=0x02;
   edge=0;
 }else
 {
   T1=TCNT1;
   TCNT1=0;
   // INT0 Mode: Rising Edge
   MCUCR=0x03;
   edge=1;
 }
}
// Declare your global variables here
void RFrecieve_PORT_INIT(void){
 HI_SEL();
 HI_SDI();
 LOW_SCK();
 //SET nFFS pin HI when using FIFO
 HI_DATA();
 SEL_OUTPUT();
 SDI_OUTPUT();
 SDO_INPUT();
 SCK_OUTPUT();
 IRQ_IN();
 DATA_OUT();
}
unsigned int RFrecieve_WRT_CMD(unsigned int aCmd){
 unsigned char i;
 unsigned int temp;
 temp=0;
 LOW_SCK();
 LOW_SEL();
 for(i=0;i<16;i++){
  if(aCmd&0x8000){
    HI_SDI();
  }else{
    LOW_SDI();
  }
  HI_SCK();
  temp<<=1;
```

```
  (1<<RFXX_SCK)
#define HI_SCK()        PORT_SCK|=
  (1<<RFXX_SCK)
#define LOW_SCK()
  PORT_SCK&=~(1<<RFXX_SCK)
////////////////////////////////
#define RF12_IRQ        2
#define IRQ_IN()        DDR_IRQ
  &=~(1<<RF12_IRQ)
#define WAIT_IRQ_LOW()
  while(PIND&(1<<RF12_IRQ))
////////////////////////////////
#define RF12_DATA       4//PD4
#define DATA_OUT()
  DDR_DATA|=1<<RF12_DATA
#define HI_DATA()
  PORT_DATA|=1<<RF12_DATA
 #define LEDG_OUTPUT()   DDRD|=~(1<<6)
#define LEDR_OUTPUT()   DDRD|=~(1<<7)
#define LEDG_ON()       PORTD&=~(1<<6)
```

```
    if(SDO_HI()){
      temp|=0x0001;
    }
    LOW_SCK();
    aCmd<<=1;
  };
  HI_SEL();
  return(temp);
}
void RF12recieve_INIT(void){
  RFrecieve_WRT_CMD(0x80D7);//EL,EF,433band,11.5pF
  RFrecieve_WRT_CMD(0x82D9);//!er,!ebb,ET,ES,EX,!eb,!ew,DC
  RFrecieve_WRT_CMD(0xA640);//434MHz
  RFrecieve_WRT_CMD(0xC647);//4.8kbps
  RFrecieve_WRT_CMD(0x94A0);//VDI,FAST,134kHz,0dBm,-103dBm
  RFrecieve_WRT_CMD(0xC2AC);//AL,!ml,DIG,DQD4
  RFrecieve_WRT_CMD(0xCA81);//FIFO8,SYNC,!ff,DR
  RFrecieve_WRT_CMD(0xC483);//@PWR,NO RSTRIC,!st,!fi,OE,EN
  RFrecieve_WRT_CMD(0x9850);//!mp,9810=30kHz,MAX OUT
  RFrecieve_WRT_CMD(0xE000);//NOT USE
  RFrecieve_WRT_CMD(0xC800);//NOT USE
  RFrecieve_WRT_CMD(0xC400);//1.66MHz,2.2V
}
      char RF12_RECV(void){
  unsigned int FIFO_data;
  WAIT_IRQ_LOW();
  RFrecieve_WRT_CMD(0x0000);
  FIFO_data=RFrecieve_WRT_CMD(0xB000);
  return(FIFO_data&0x00FF);
}
```

Main function:

According to the desired conditions and using the Code wizard function main () is defined as we provide the following:

```
void main(void)
{
// Declare your local variables here
      char i,i1,i2,i3,c1,c2,c3,c11,c21,c31;
// Input/Output Ports initialization
// Port A initialization
// Func0=In Func1=In Func2=In Func3=In Func4=In Func5=In
  Func6=In Func7=In
// State0=T State1=T State2=T State3=T State4=T State5=T
  State6=T State7=T
PORTA=0x00;
DDRA=0x00;
// Port B initialization
// Func0=In Func1=In Func2=In Func3=In Func4=In Func5=In
  Func6=In Func7=In
// State0=T State1=T State2=T State3=T State4=T State5=T
  State6=T State7=T
PORTB=0x00;
DDRB=0x00;
// Port C initialization
// Func0=Out Func1=Out Func2=Out Func3=Out Func4=Out
  Func5=Out Func6=Out Func7=Out
// State0=0 State1=0 State2=0 State3=0 State4=0 State5=0
  State6=0 State7=0
PORTC=0x00;
DDRC=0xFF;
// Port D initialization
// Func0=In Func1=In Func2=In Func3=In Func4=In Func5=In
  Func6=In Func7=In
// State0=T State1=T State2=T State3=T State4=T State5=T
  State6=T State7=T
PORTD=0x00;
DDRD=0x00;
// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;
// Timer/Counter 1 initialization
```

```
// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;
// External Interrupt(s) initialization
// INT0: On
// INT0 Mode: Falling Edge
// INT1: Off
// INT2: Off
GICR|=0x40;
MCUCR=0x02;
MCUCSR=0x00;
GIFR=0x40;
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;
// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
// Analog Comparator Output: Off
ACSR=0x80;
SFIOR=0x00;
// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud Rate: 9600
UCSRA=0x00;
UCSRB=0x18;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x19;
// Global enable interrupts
#asm("sei")
LEDG_OFF();
LEDR_OFF();
LEDG_OUTPUT();
```

```
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
LEDR_OUTPUT();
for(i=0;i<3;i++)
{
      delay_ms(200);
      LEDG_ON();
      LEDR_ON();
      delay_ms(200);
      LEDG_OFF();
      LEDR_OFF();
}
LEDG_OFF();
LEDR_OFF()
```

;After the recipient's definitions for the required functions of the module setup it will call the lift:

```
//Initialize command port
RFrecieve_PORT_INIT();
//Initialize RF12 chip
RF12recieve_INIT();
//Init FIFO
```

RFrecieve_WRT_CMD(0xCA81);Had to download and view the information in a ring of active buffer FIFO has to end, then we check the bytes received, according to the sender if the first byte is received, then the next byte number 20 is related to the temperature and call it after that to decimal, and single in c1 and c2 and c3 of the character set, Otherwise call the function RF12_RES with (a) the recipient micro once startup again (RESET) until get receive again.

```
while (1)
    {
     //Enable FIFO
     RFrecieve_WRT_CMD(0xCA83);
          //Receive Check sum
     i=RF12_RECV();
     i1=RF12_RECV();
     i2=RF12_RECV();
    i3=RF12_RECV();
          //Disable FIFO
     RFrecieve_WRT_CMD(0xCA81);
          //Temprature Recognition
     if (i=20)
     {
      c1 =i1/100;
      i1  =i1-100*c1;
      c2 =i1/10;
      i1  =i1-10*c2;
      c3 =i1;
      c1 +=0x30;
      c2 +=0x30;
            c3 +=0x30;
            }

       else
       {
       RF12_RES = ON;
       Delay_ms(500);
RF12_RES = OFF
```

;Finally, using the commands send a serial that will be more of the expression of the desired character by contact micro USART and transferred to the computer through a COM port may.

```
     //Humidity Recognition
          if (i2=30)
          {
          c11 =i3/100;
          i3  =i3-100*c11;
          c21 =i3/10;
          i3  =i3-10*c21;
          c31 =i3;
          c11 +=0x30;
          c21 +=0x30;
          c31 +=0x30;
          }
     else
     {
     RF12_RES = ON;
     Delay_ms(500);
     RF12_RES = OFF;
     }
          //Temprature sending to USART
          if(c1 != 0)
          {
          putchar (c1);
          }
          if(c2 !=0)
          {
          putchar (c2);
          }
          if(c3 !=0)
          {
          putchar (c3);
          }
          putchar(0x0D);

       //Humidity Sending To USART
          if(c11 != 0)
          {
          putchar (c11);
          }
          if(c21 !=0)
          {
          putchar (c21);
          }
          if(c31 !=0)
          {
          putchar (c31);
          }
          putchar(0x0D);
         };
```

Conclusions:

In this paper it was trying to use the new generation telecommunication and communication devices, a new generation of radio air condition ahead of the nose has designed and build. Also in this article to detail the program written and all the settings on the micro and the transmitter and receiver have been carefully described.

REFERENCES

[1] Joe Pardue, C Programming For Microcontrollers, Knoxville TN.

[2] Mazidi, Jonice Gillispie, Micro controller, Dr Sepidnam:1381.

[3] Engineer jafarnejad qmi, C language programming, publications of the University Jihad, autumn 1999.

[4] Amir rahafrooz, micro controller AVR and their applications, Tehran, nas, 2007.

[5] Reza. thanks for rapid training of companion, micro controller AVR.

[6] Ali salmanian, structure of micro controller AVR.

[7] Mohamad reza Mohamad zadeh, bachelor thesis, Iran, 2010