“WIRELESS SENSOR ADHOCK NETWORK”

(Phase 1 Report)



**Submitted By**

**WAJID ALI KHAN 24600**

**SYED NISAR SHERAZI 24592**

**MUHAMMAD TARIQ KHAN 24569**

**SYED NADEEM SHERAZI 24591**

**Supervisor**

**SIR FAHEEM AWAN**

BS Telecommunication Department of Information Technology

Hazara University Mansehra

February 19, 201**.**

**Introduction:**

A wireless sensor network (WSN) is a wireless network consisting of spatially distributed autonomous tiny computing devices, each equipped with a wireless radio, a processor, and a power source. In this project we are adding an intelligence factor. Intelligence factor means that the sensors will be connected to the main circuit where all the calculations will be carried out and then the sensors data will be forwarded wirelessly to the intermediate node and then the intermediate node will decide where to send the each sensor data. Sensor networks are envisioned to be deployed in the physical environment in order to monitor a wide range of environmental phenomena.

Previously, sensor networks consisted of small number of sensor nodes that were wired

to a central processing station. However, nowadays, the focus is more on wireless, distributed, sensing nodes. In most cases, the environment to be monitored does not have an

existing infrastructure for either energy or communication. It becomes imperative for sensor nodes to survive on small, finite sources of energy and communicate through a wireless

communication channel.

Sensor network application like weather monitoring needs data like temperature, light intensity, and motion detector etc. called as sensing modalities, there exist different sensors to sense each sensing modality.

Initial work on project:

Before starting the work on project we have divided our project in four different phases, firstly we will complete all phases on software i.e. Proteus and CADSOFT EAGEL (Proteus for simulation and CADSOFT EAGEL SOFTWARE for PCB designing). In last phase we will design the project on hardware.

The phases of this project are as below.

Phase (1)

Phase (2)

Phase (3)

Phase (4).

**“PHASE 1”**

**Block diagram:**

12 v

zigbee

controller

Light Intensity

Temperture

In this phase a 12v DC power supply is connected to the At mega controller with this power controller will start working. Now at the lower side of this figure we can see that two sensors are connected to the At mega controller, first one is temperature sensor which will sense the temperature from the room and send it to the At mega controller. Secondly light intensity sensor will sense light of the room and send it to At mega controller. As At mega controller collects the data from temperature and light intensity sensor it will transfer its data to intermediate node through zigbee.

“PHASE 2”

Block Diagram:

zigbee

Intermediate Node

controller

12 v

Load 2

Motion sensor

Load 1

In phase 2 zigbee will receive the data from phase one and give it to the intermediate node (controller). A 12v DC power supply is connected to controller to give the power to the controller. We have connected the load 1 and load 2 to intermediate node now when the intermediate node receives the data from zigbee it will transfer it to load 1 and load 2. We have also connected a motion detector sensor to intermediate node so that it can sense the motion in the room and sent the data to controller and then controller will send this data to another controller through zigbee .

“PHASE 3”

Block Digram:

Reciveing data from phase 2

Computer server

Monitoring and control

12 v

zigbee

Controller

Temperature

Light intensity module

In this phase zigbee will receives data from phase 2 and sent to controller. A 12v DC power supply is connected to controller to give the power to the controller. Here controller connected to computer server which will monitor and control. on computer display we can see the motion ,temperature and light intensity.

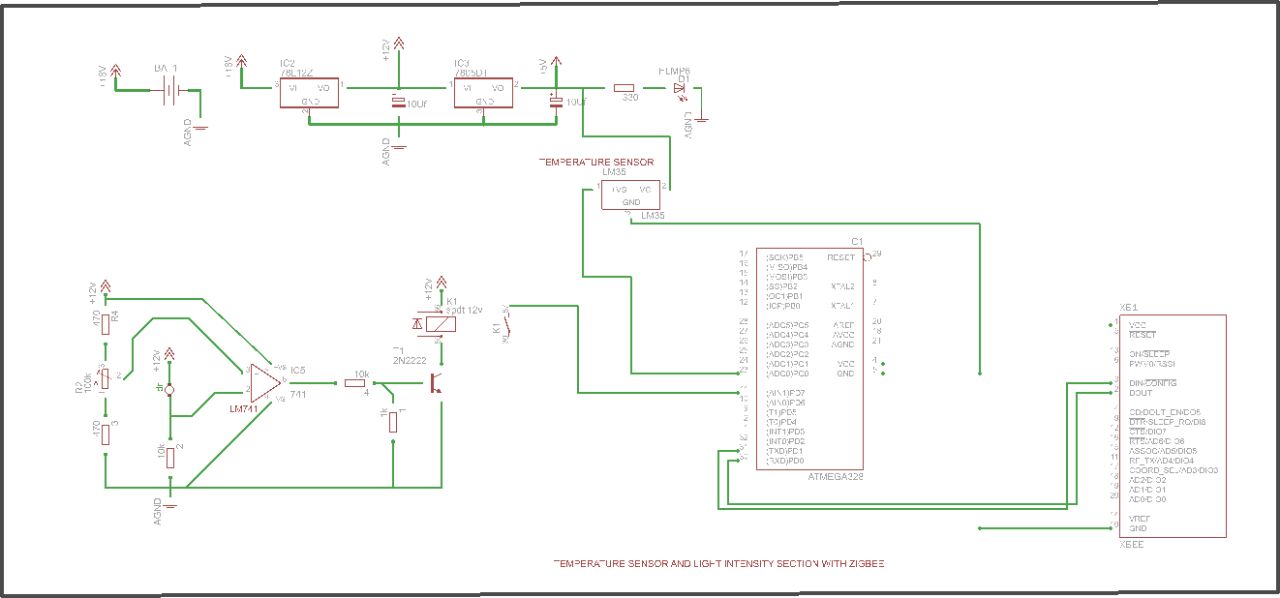
**Phase 4:**

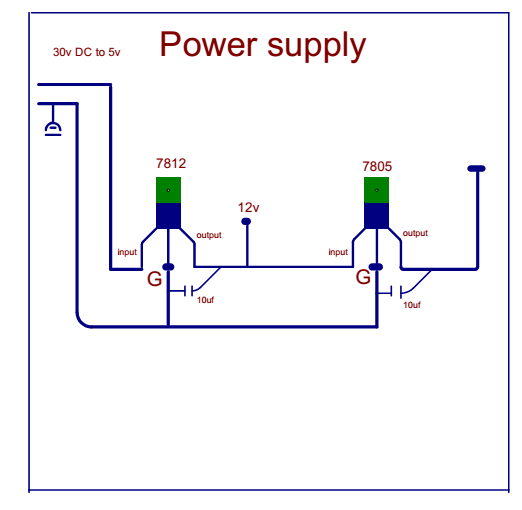
In this phase we will design the project on hardware.

**Work on phase 1:**

In phase 1 we started work form software side. Firstly we design a PCB design for 12V DC power supply, temperature sensor, light intensity sensor, controller by the help of CADE SOFT EAGLE SOFTWARE.

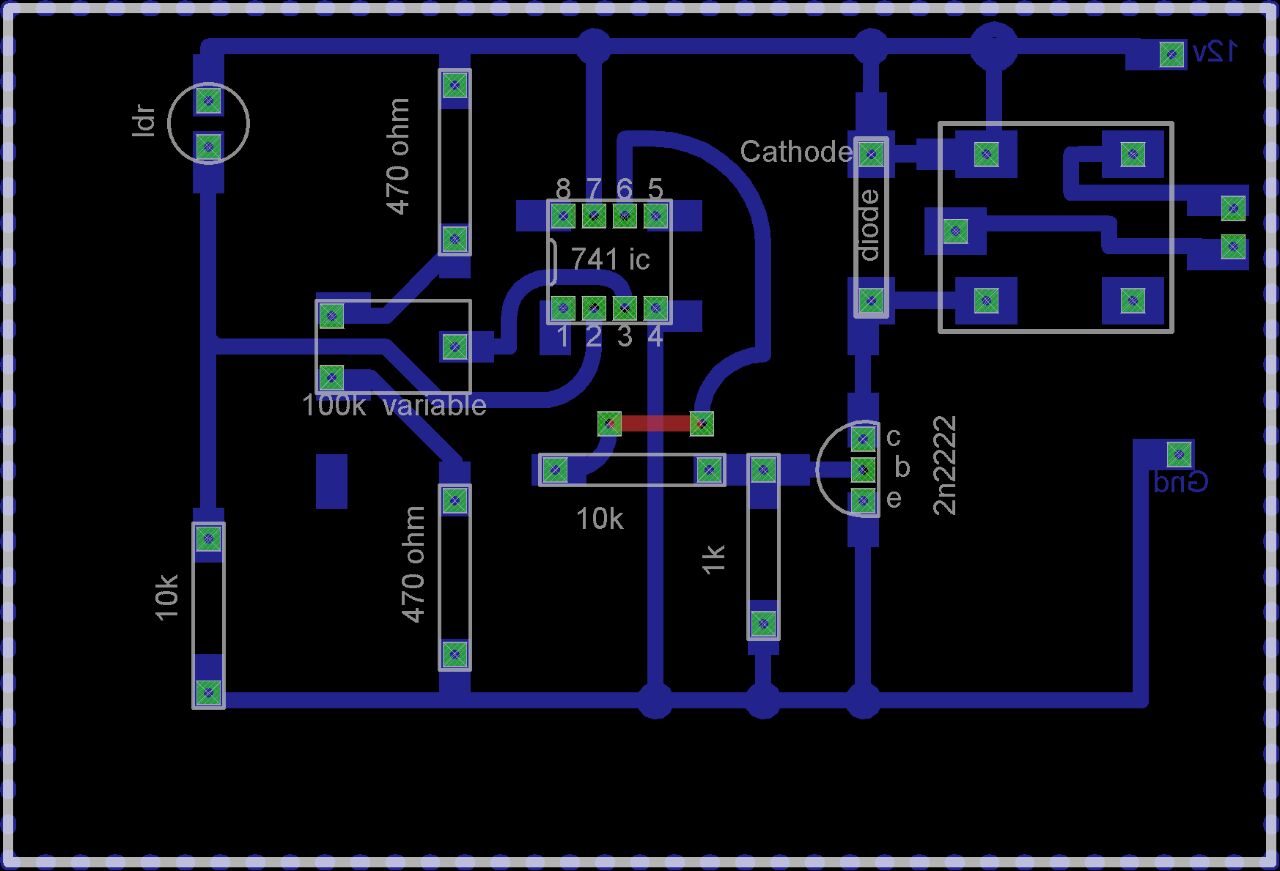
This is the circuit diagram for the whole phase 1.





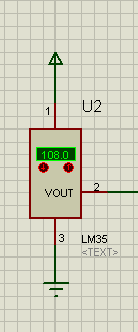
In this power supply we use 2 voltage regulator 7812 and 7805. 7812 which converts 30 DC volt to 12 volt and 7805 which will convert 12 volt into 5 volt. Here 5 volt is taken for such ic’s that cannot operate more then 5 volt.

**Light Intensity:**



The LDR and 10k resistor are used in series which makes a voltage divider. the divided voltage it fed to the 741 OPM which is used as the voltage comparator. the 100k ohm variable resistor is used to set the reference voltage. through this variable resistor we set the light intensity. the two 470 ohm resistors are used for the protection of the variable resistor to avoid the short circuit, if incase the resistance of the variable resistor drops to zero. the output from the 741 ic is given to the relay through the transistor which switch on/off the relay to signal the controller.

**Temrature senseor:**

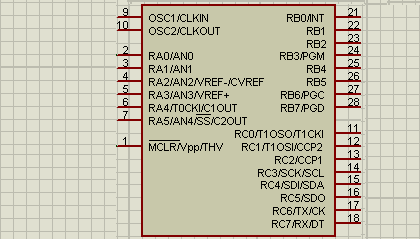


It has an output voltage that is proportional to the Celsius temperature.

The scale factor is .01V/**o**C

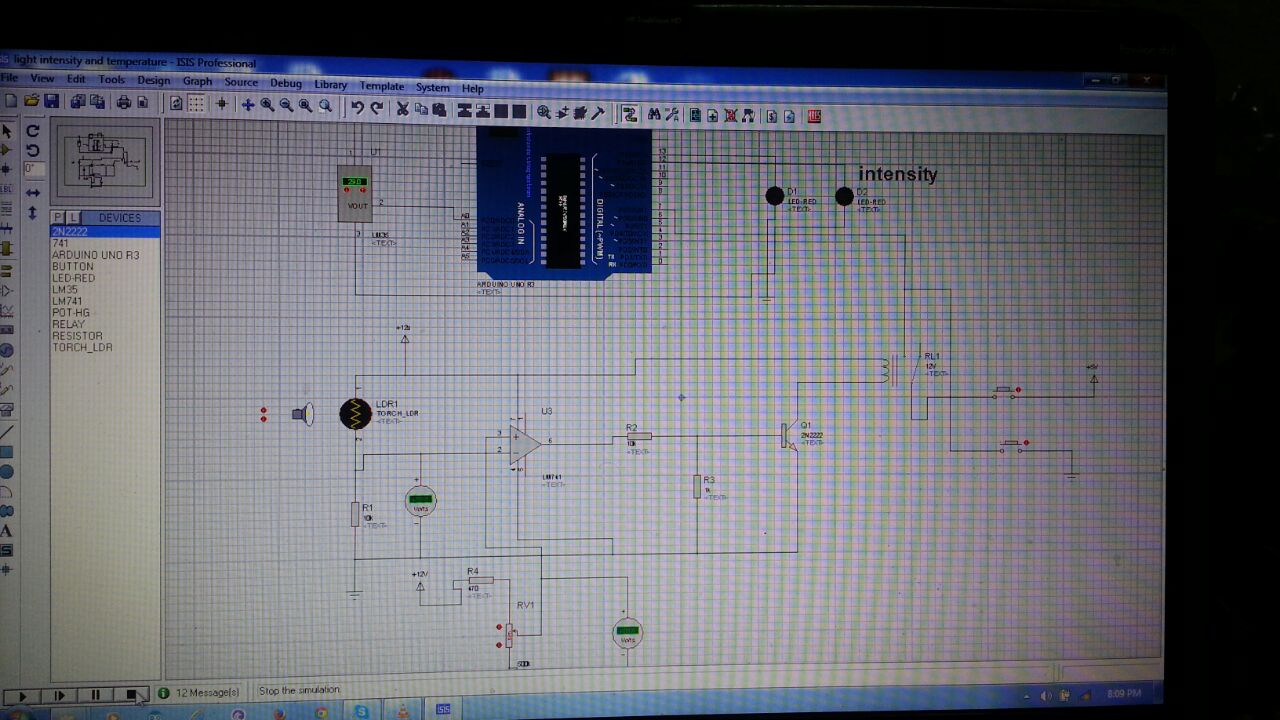
The LM35 does not require any external calibration or trimming and maintains an accuracy of  +/-0.4 **o**C at room temperature and +/- 0.8 **o**C over a range of 0 **o**C to +100 **o**C.

Another important characteristic of the LM35DZ is that it draws only 60 micro amps from its supply and possesses a low self-heating capability. The sensor self-heating causes less than 0.1 oC temperature rise in still air.

**Atmega controller:**

The Atmega [8 bit](http://en.wikipedia.org/wiki/8-bit) [AVR](http://en.wikipedia.org/wiki/Atmel_AVR) [RISC](http://en.wikipedia.org/wiki/RISC)-based microcontroller combines 32 KB [ISP](http://en.wikipedia.org/wiki/In-system_programming) [flash](http://en.wikipedia.org/wiki/Flash_memory) memory with read-while-write capabilities, 1 KB [EEPROM](http://en.wikipedia.org/wiki/EEPROM), 2 KB [SRAM](http://en.wikipedia.org/wiki/Static_random-access_memory), 23 general purpose I/O lines, 32 general purpose working [registers](http://en.wikipedia.org/wiki/Processor_register), three flexible timer/[counters](http://en.wikipedia.org/wiki/Counters) with compare modes, internal and external [interrupts](http://en.wikipedia.org/wiki/Interrupts), serial programmable [USART](http://en.wikipedia.org/wiki/USART), a byte-oriented 2-wire serial interface, [SPI](http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus) serial port, 6-channel 10-bit [A/D converter](http://en.wikipedia.org/wiki/A/D_converter) (8-channels in [TQFP](http://en.wikipedia.org/wiki/TQFP) and [QFN](http://en.wikipedia.org/wiki/QFN)/[MLF](http://en.wikipedia.org/wiki/Quad-flat_no-leads_package#Variants) packages), programmable [watchdog timer](http://en.wikipedia.org/wiki/Watchdog_timer) with internal [oscillator](http://en.wikipedia.org/wiki/Electronic_oscillator), and five software selectable power saving modes. The device operates between 1.8-5.5 volts. The device achieves throughputs approaching 1 [MIPS](http://en.wikipedia.org/wiki/Million_instructions_per_second#Million_instructions_per_second) per MHz.

**samulation :**



Programming:

**VirtualWire.h " header file" :**

VirtualWire is a library for Arduino, that provides features to send short messages, without addressing, retransmit or acknowledgment, a bit like UDP over wireless, using ASK (amplitude shift keying). Supports a number of inexpensive radio transmitters and receivers.

**SoftwareSerial.h " header file":**

**/We could use both Serial and SoftwareSerial at the same time, since we used different digital pins to send/receive data at the same time.**

**by default the arduino pins 0 and 1 are used for the serial communication. but if in case we want to use other pins for serial communication then we use SoftwareSerial head file.**

**#include <VirtualWire.h>**

**#include <SoftwareSerial.h>**

**SoftwareSerial Xbee(2, 3);**

**// over hear the softwareSerial is used to define pin2 and pin3 of the arduino to send and //receive data serially through Xbee.**

**int outputpin= 0; // for reading temperature**

**int relay = 13; // an ac load will be connected here**

**int relay2 = 12; // an ac load will be connected here**

**int intensity = 7; // light intensity circuit is connected here.**

**// the void setup() is a function having now arguments. this function excutes only once when //the controller is turned on. this function is used for the basic setting like specifying the baud rates or pins etc.**

**void setup()**

**{**

**Serial.begin(9600); // baud rate**

**pinMode(intensity , INPUT); // declared as input pin**

**pinMode(relay, OUTPUT); // as output pin**

**pinMode(relay2, OUTPUT);**

**digitalWrite(relay, LOW);**

**}**

**//main loop it executes again and again forever.**

**void loop()**

**{**

**int rawvoltage= analogRead(outputpin); // reading the temperature sensor.**

**float millivolts= (rawvoltage/1024.0) \* 5000;**

**float celsius= millivolts/10;**

**Serial.print(celsius);**

**//Serial.print() function is used to send data to the computer screen for the debugging //purposes.**

**Serial.print(" degrees Celsius, ");**

**Serial.print((celsius \* 9)/5 + 32); // formula for the temperature to display in Fahrenheit**

**Serial.println(" degrees Fahrenheit");**

**delay(1000); // 1 sec delay. as 1000 msec is qual to 1 sec**

**if( celsius > 30) // condition for temperature. if the temperature is greater then 30 it will //send " 1 " as a signal to the remote Xbee module and at meanwhile will also //turn on the load for the indication.**

**{**

**Xbee.write('1');**

**digitalWrite(relay, HIGH);**

**}**

**if(celsius < 30)**

**{**

**digitalWrite(relay, LOW);**

**Xbee.write('2');**

**}**

**if(digitalRead(intensity) == HIGH)**

**// this line of code reads the ttl value from the light intensity circuit, and if it satisfies the //condition then the following code will be executed.**

**{**

**digitalWrite(relay2, HIGH);**

**Xbee.write('a');**

**}**

**if(digitalRead(intensity) == LOW)**

**{**

**digitalWrite(relay2, LOW);**

**Xbee.write('b');**

**}**

**}**

Thanks….