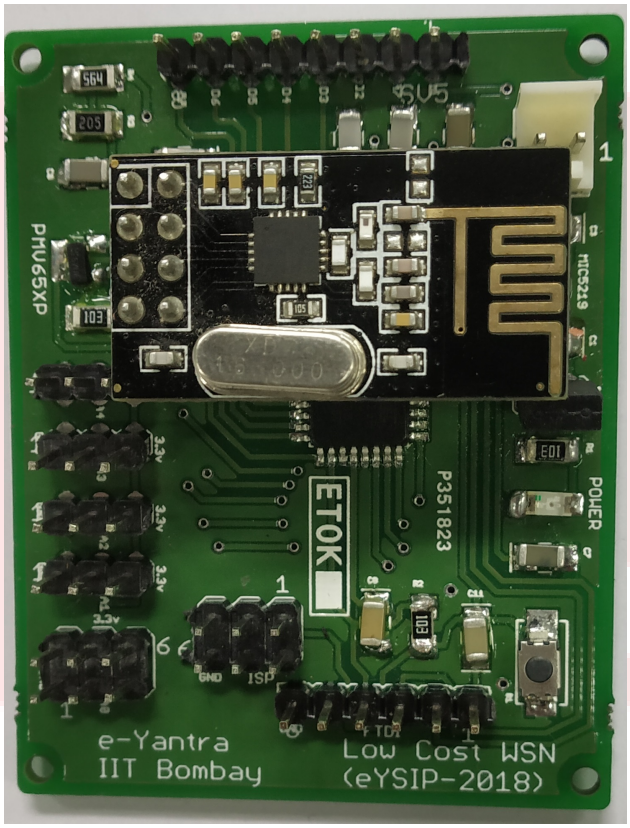
eYSIP2018

Low cost sensor node for wsn



Sachin Mahadev Jadhav
Nithin Thilakappan
Nishit Patel
Parin Chheda
Kalind Karia

Duration of Internship: 21/05/2018 – 06/07/2018

2018, e-Yantra Publication

Low cost sensor node for WSN

Abstract

The prime motive of this project is to make sensor node based on Atmega328 platform that has the
Following features:

1. custom built power supply for optimized for low power sensor node applications
2. Ability to program via Arduino IDE/ Atmel Studio
3. Use NRF2401 for RF communication
4. Completely open source design and sample codes to make it useful for WSNs
5. Can be used as general purpose microcontroller board for learning interfacing and C programming

Completion status

1.1 Hardware parts

- List of hardware
    1. Atmega328p
    2. nRF24L01
    3. PMV65XP (MosFET)
    4. AP2112 (LDO)
    5. ESP32 (Gateway)

- Detail of each hardware:

1. **Atmega328p** is used as on board main microcontroller for each and every node due to its high performance and low power consumption. nRF24L01 is connected to its SPI bus for radio communication. Different types of sensors can be interfaced to its ADC port. Also its GPIO pins are taken out on the board so that board can be used as a microcontroller development board for testing purpose. Atmega328p Datasheet, Vendor link
2. **nRF24L01** is used to send data via radio communication between each nodes. The **nRF24L01** is a single chip **2.4GHz transceiver** with an embedded baseband protocol engine (Enhanced ShockBurst), designed for ultra low power wireless applications. The textbfnRF24L01 is designed for operation in the world wide ISM frequency band at **2.400 - 2.4835GHz.** An MCU (microcontroller) and very few external passive components are needed to design a radio system with the **nRF24L01.** The **nRF24L01** is configured and operated through a Serial Peripheral Interface (SPI.) Through this interface the register map is available. The register map contains all configuration registers in the **nRF24L01** and is accessible in all operation modes of the chip. The embedded baseband protocol engine (Enhanced ShockBurst) is based on packet communication and supports various modes from manual operation to advanced autonomous protocol operation. Internal FIFOs ensure a smooth data flow between the radio front end and the systems MCU. Enhanced ShockBurst reduces system cost by handling all the high-speed link layer operations. nRF24L01 Datasheet, Vendor link
3. **PMV65XP** is P-channel enhancement mode Field-Effect Transistor (FET) in a small SOT23 (TO-236AB) Surface-Mounted Device (SMD) plastic package using Trench MOSFET technology.
   - Low threshold voltage
   - Low on-state resistance
   - Trench MOSFET technology

   PMV65Xp Datasheet, Vendor link
4. The **AP2112** is CMOS process low dropout linear regulator with enable function, the regulator delivers a guaranteed 600mA (min.) continuous load current. AP2112 Datasheet, Vendor link

5. **ESP32** is a single 2.4 GHz Wi-Fi-and-Bluetooth combo chip designed with the TSMC ultra-low-power 40 nm technology. It is designed to achieve the best power and RF performance, showing robustness, versatility and reliability in a wide variety of applications and power scenarios. ESP32 Datasheet, Vendor link

- PCB schematic design of circuit

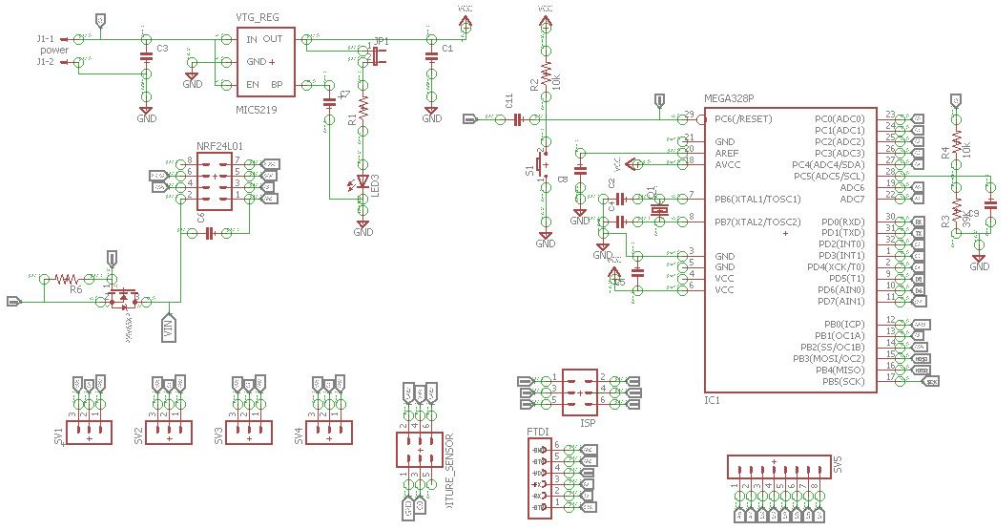


Figure 1. **Schematic Diagram**

1.2 Software Used

- List of software used

1. Arduino IDE
2. Atmel Studio IDE

- Detail of software:

1. **Arduino IDE** is used for programming the main microcontroller atmega328p as per user requirement. version 1.8.5 Arduino IDE download
2. **Atmel Studio 7.0** download

- Installation steps
  1. Instruction for installation of **Arduino IDE 1.8.5** click here
  2. Instruction for installation of **Atmel Studio IDE 7.0** click here

1.3 Set up and Programming

Setting up fusebits on atmega328p

- NEX AVR USB ISP STK500V2 is used for setting up fusebits and flashing program files into the microcontroller because of its high speed. In-System USB programmer for AVR family of microcontrollers. It can be used with AVR Studio on Win XP platforms.
- For Windows 7 and Windows 10 it can be used in HID mode with Avrdude command prompt as programming interface. Its adjustable clock speed allows programming of microcontrollers with lower clock speeds.
- Using HID mode of stk500v2 does not require any driver installation before we start to use it. Because it uses generic windows drivers.
- Download necessary files of AVRdude to run the programmer in HID mode click to download
- After downloading necessary files from above link click here for basic setup and installation refer page 19 and 20.
- TO know more about AVRDUDE commands AVRDude Commands meaning
- **Steps for setting up fusebits using stk500v2 commands**
  1. **Show default settings all Fuse bits, lock bits ,mode of programming of microcontroller**
     avrdude -c stk500v2 -p m328p -P NEX-USB-ISP -v
  2. **Modifying the default fusebits**
     avrdude -c stk500v2 -p m328p -P NEX-USB-ISP -U efuse:w:0x05:m -U hfuse:w:0xda:m -U lfuse:w:0xff:m
  3. **Uploading a program file(.hex file) into microcontroller**
     avrdude -c stk500v2 -p m328p -P NEX-USB-ISP -U flash:w:codename.hex
  4. **setting of lock bits**
     avrdude -c stk500v2 -p m328p -P NEX-USB-ISP -U lock:w:ox0f:m

Burning bootloader into atmega328p

- Click here to refer steps for burning a bootloader
- Bootloader which we used was from a arduino directory i.e., local disk C: then Program Files (x86) then Arduino then hardware then arduino then avr then bootloaders then atmega then ATmegaBOOT168atmega328pro8MHz.hex

1.4 Assembly of hardware

Block Diagram

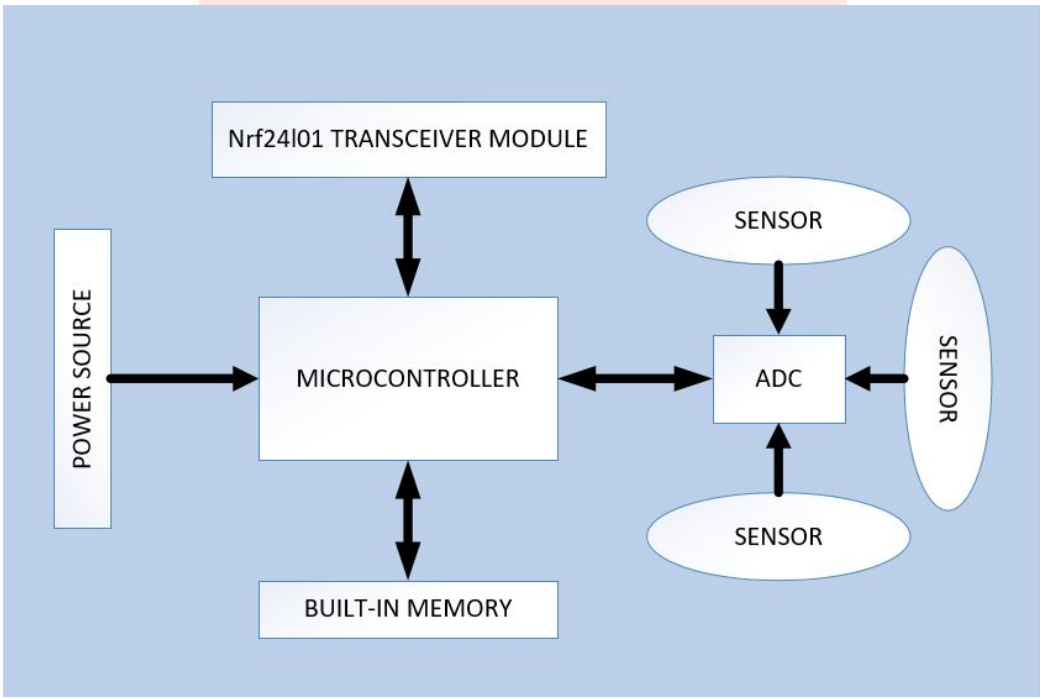
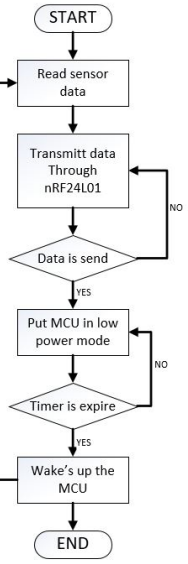
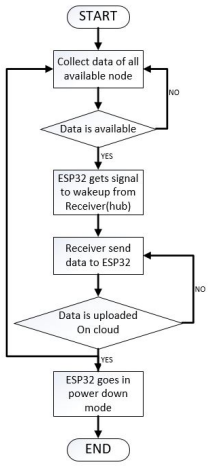


Figure 2. **Block diagram of system**



(a) Transmitter flow chart



(b) Receiver flow chart

- This project was mainly concentrated on urban farming where the sensor node will keep monitoring the farming parameters which should be taken an immense attention towards it
- So multiple sensors will be placed at different locations from where it will send parameters like soil moisture data, temperature data, humidity data etc
- By including multiple sensor nodes into the field it will create a network through which it will communicate with each other and data will the

received at a hub point and from the hub point there is an ESP32 module acting as a gateway of this network for uploading the received data to the internet i.e., thingspeak cloud. It is possible to implement different type of network as per the complexity of the requirement and how much distance is to be covered

- Types of network are
  1. Star network
  2. Mesh network
  3. Hybrid network
- Here we have implemented a star network completely on the c language which means we developed the required libraries which is used in the star network
- Parallel to that we have also implemented the whole system in mesh network too but we are using arduino libraries for the same c language libraries is remaining to be developed.

Implementation of mesh network using three sensor nodes and an esp32 module.

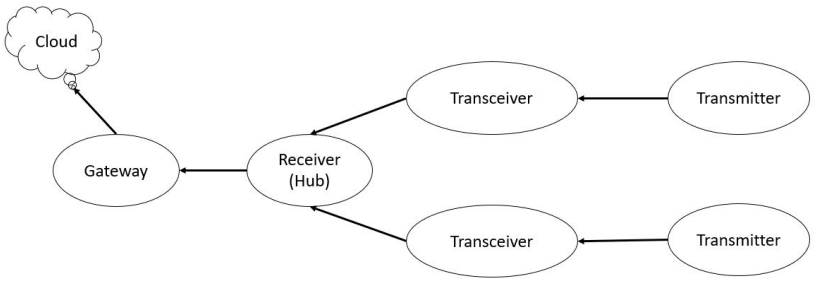


Figure 3. **Flow diagram of mesh network**

- Those three nodes were given a particular node number like node 1, node 2, node 3 here node one is a transmitter node 2 is an intermediate transceiver node and node number 3 is said to be an hub of the network which is serially connected with the ESP32. Out of three sensor nodes node number one was not in the range of node 3 so we kept an intermediate node 2 in between node 1 and node 3 so that data from node 1 can be transmitted to node 3 via node 2.

- Initially node 1 and node 2 with be in sleep mode it will wake up at every 20 seconds to transmit it's data to the hub node 3. At the time of data transmission node 1 wakes up and transmit it's data to node 2 and node 2 wake up and receive the data of node 1 and it will send it's own data along with the received data from node 1 to the hub node 3 from node 3 it will be serially transmitted to the ESP32 module which will upload the data to the thingspeak server.

How does ESP32 (Gateway) works?

- Here ESP32 is used along with an NRF based receiver to act as a gateway.
- ESP32 is connected with receiver (hub) via serial UART2 (RX, TX) for serial Communication.
- Normally ESP32 will be in a sleep mode as soon as when the hub receives all the data's from all the nodes it will send an interrupt signal to the ESP32 to wakeup from the sleep mode and setup a wlan connection through WiFi and connect to the thingspeak server.
- After setting up a successful connection with the server, then it will send an acknowledgement to the receiver hub regarding the setup of connection.
- As soon as the acknowledgement is received at the hub, hub will send all the data to the ESP32 via serial Communication, while the data is received at ESP32 it will ask for a post request on the thingspeak server through which the data will be uploaded to the thingspeak server.
- Measure current of PCB board
  - Transmit mode current = **6.7 mA**
  - Receiver mode current = **22-25mA**
  - Sleep mode current = **100 uA**
- Measure current of nRF24L01
  - Normal mode current = **1.2 mA**
  - stand by mode current = **40 uA**
  - Sleep mode current = **900 nA**

- Current measurement of Pcb board and nRF24L01 on DSO

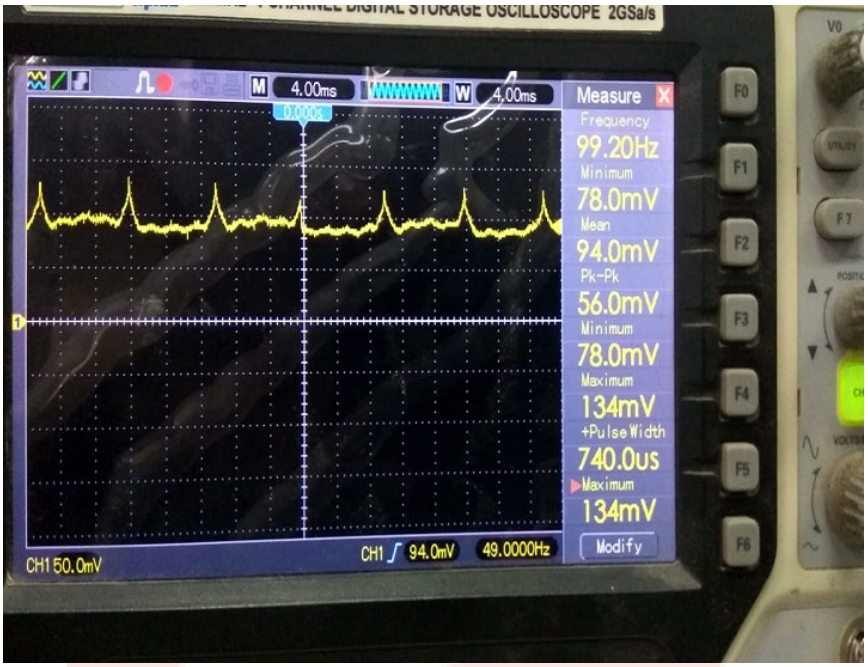


Figure 4. **Current of PCB Board (transmit mode(6.7mA))**

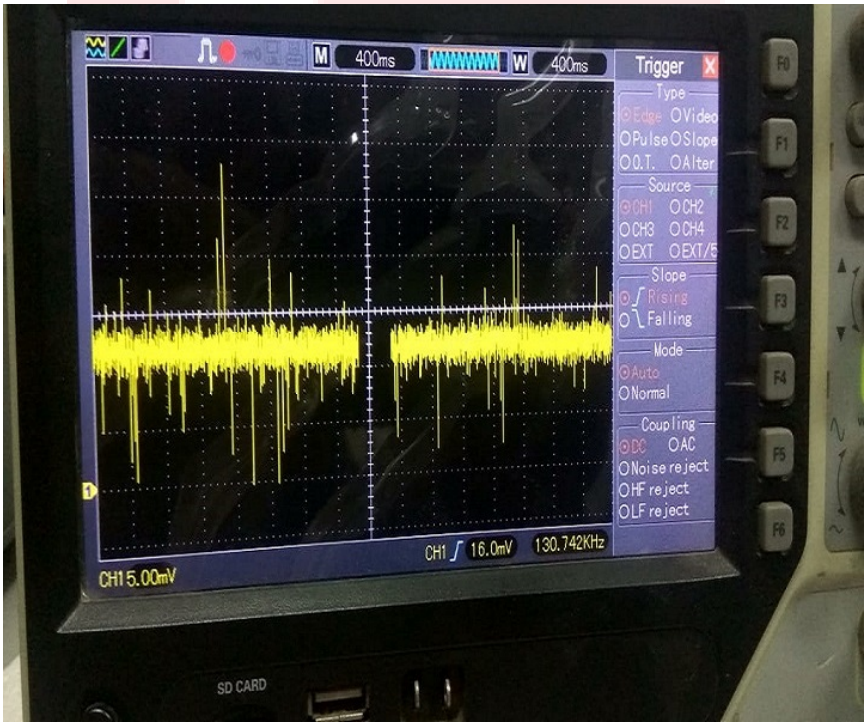


Figure 5. **Current of PCB Board (receiver mode(22-25mA))**

- Test the range of nRF24L01 in outdoor environment with different data rate
  - MIN (-18 dBm) power = **0 to 6 m**
  - LOW (-12 dBm) power = **0 to 8 m**
  - HIGH (-6 dBm) power = **0 to 12 m**
  - MAX (0 dBm) power = **0 to 16 m**

Table 1.1: Range testing of nRF24L01 (At different data rates)

Transmission Power level Distance (meter)	MIN power (-18 dBm)	LOW power (-12 dBm)	HIGH power (-6 dBm)	MAX power (0 dBm)
3.8	100%	100%	100%	100%
4.9	100%	100%	100%	100%
5.9	100%	100%	100%	100%
6.9	47%	100%	100%	100%
8	0%	100%	100%	100%
8.2	0%	100%	100%	100%
10	0%	74%	100%	100%
12.4	0%	0%	100%	100%
15.6	0%	0%	86%	100%

- 100% means after transmitting 15 packets from transmitter side all 15 packets have successfully received at receiver end.
- Range testing was carried out at the terrace of KReSiT building where there was greenery because of the farming carried out over there also there was few wifi routers functioning nearby the farming area which forced us to use a different radio communication channel number i.e., 76 which won't be effected by the wifi networks.
- As we know that greenery absorbs 2.4 GHz frequency due to which some of our data packets failed to receive at receiver end in some conditions.
- As shown in the above table 3.8- 5.9 meters distance communication is completely working very well in all the power levels of nRF24L01.

- While increasing the distance at 6.9 meters the received packets were moderate which is 47 percent of 15 packets.
- Again testing nRF24L01 to its extreme level the distance was kept 15.6 meters and the complete packets received at receiver end only in max power level rest all other modes failed.
- Hardware prototype model

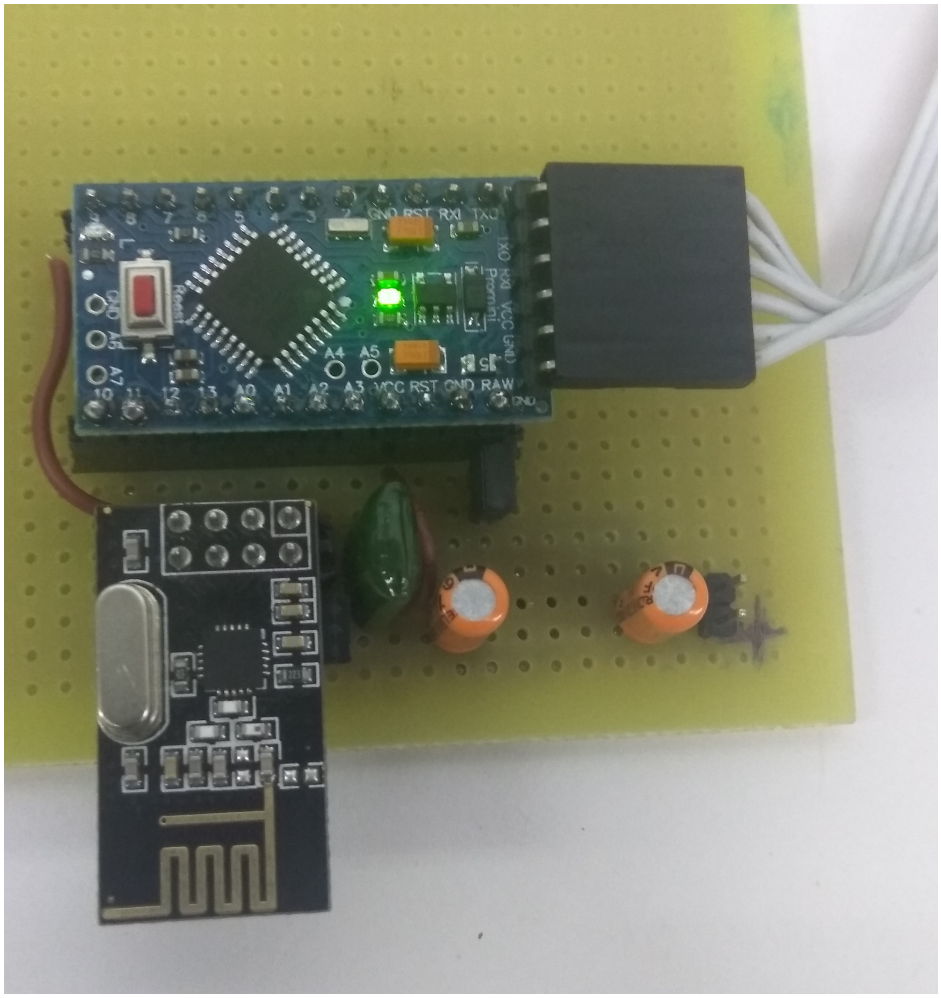


Figure 6. **Implementation of handmade prototype model**

- Printed PCB hardware of Node

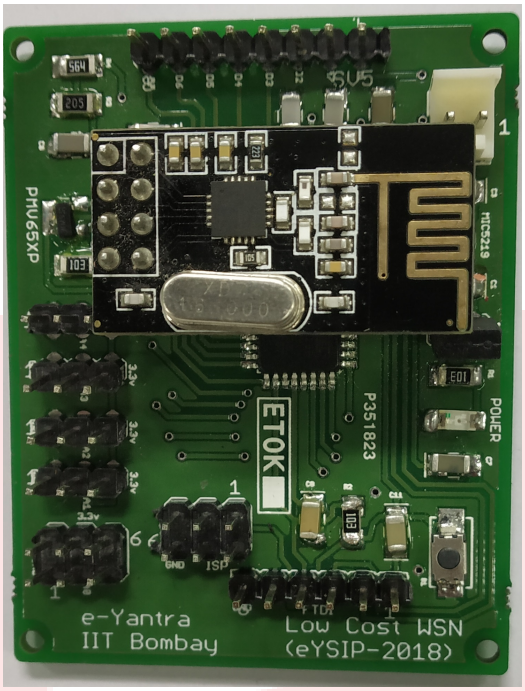


Figure 7. **Printed PCB of node**

- Pin configuration and connection

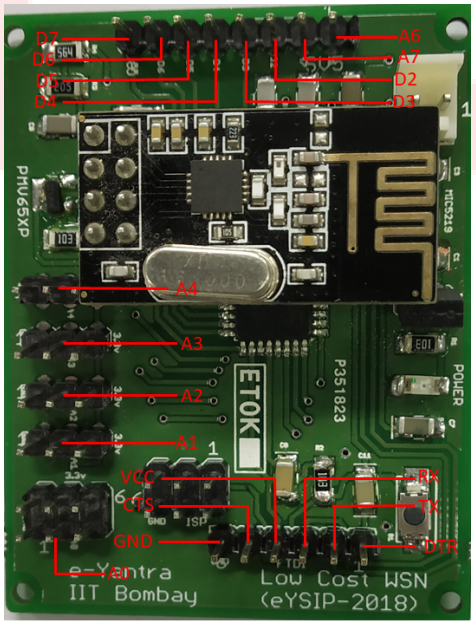


Figure 8. **Pin configuration of board**

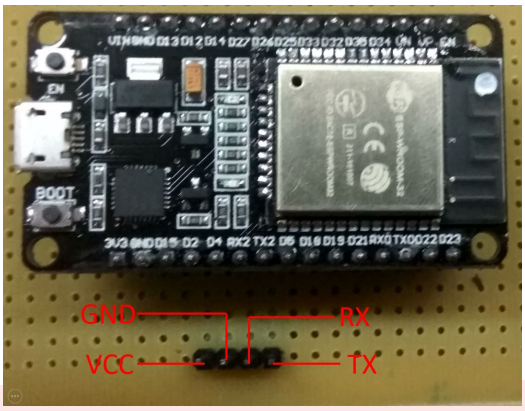


Figure 9. **Pin out connection of ESP32**

- Here we implemented a star network in which there was three transmitters and one receiver were the receiver node was said to be a hub of the whole network which is connected to ESP32 gateway for uploading the received data to the thingspeak server.

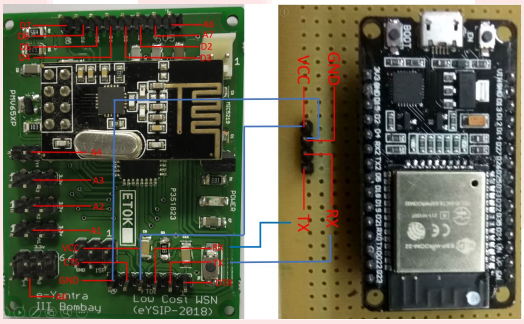


Figure 10. **Connection description of hub and ESP32**

1.5 Software and Code

For the repository of code Github link

Brief explanation of various parts of code

1.6 Use and Demo

Final Setup Image

User Instruction for demonstration

Youtube Link of demonstration video

1.7 Future Work

- We have implemented two types of network star network and mesh network which consist of 3 transmitting nodes and one receiver node usually known as a Hub. This implementation can be extended more by implementing a hybrid network which is a combination of star network and mesh network.
- JSON packet can be used for uploading bulk data to the thingspeak cloud. So that it can replace the current method of uploading data to the cloud which is sending a post request to the server.
- The onboard LED must be connected on D13 pin for debugging of NRF data receiption & transmission.

1.8 Bug report and Challenges

- Prototype testing of nRF24L01
  - In the starting days we were connecting nRF24L01 with Arduino pro mini via one to one connectors so we were facing loose connection and sometimes due to some fluctuations in the voltage the nRF24L01 was not giving appropriate reading. So we implemented an hardware on which we carried out our testings which mitigated our fluctuations and loose connection problem.
- Range testing of nRF24L01 in outdoor environment
  - After implementation we took our prototype model for testing it at the terrace of KReSiT building as there are lot of Wi-Fi connections available at the terrace so it was creating lot of disturbance in the communication as Wi-Fi so we used 76 channel number for using radio frequency as nRF24L01 works on 2.4 GHz frequency greenery present over there was absorbing 2.4 GHz frequency up to some level.
- Setting of fuse bits (Low, High, Extended) using AVRDude
  - While setting fusebits we were facing some problem for setting it initial and also reading the fusebits. Finally we found that commands we were using was having some error in it so we used a appropriate command for it and it worked well.

- At starting we are using FTDI chip for modifying fuse bits through serial communication,but Fuse bit and lock bits can not be changed through serial communication i.e. through FTDI chip. so after that we have used stk500v2 programmer which uses SPI to acess direct flash and usig proper commands for writing fuse bits in AVRDUDE problem is solved.

- Importing RF24 library in Atmel Studio

  1. RF24.h is made for any arduino device along with ESP & R-PI.For this is using arduinocore i.e.arduino.h , SPI.h ,Each library of approx thousands of lines.
  2. So first of all i am not getting where to start . Then I have started step by step.
     - Listed down all the functions used by RF24.cpp from arduino.h same for SPI.h
     - Written all this function in Embedded C and added GPIO.h file and GPIO.c file in same folder in SPI.h and RF24.h and removed Aduino.h
     - There are lot of unnecessary macro which are not needed for Atmega328p in RF24.h , SPI.h all of this modified properly.
  3. For proper SPI speed and settings I have rewritten the **SPI.h** library at register level referring from Atmega328p datasheet.
  4. After doing this, Project Compiled successfully but nrf is not receving data.Because for reading the NRF pipe addresses and it needs some proper macros to be defined to read data from Program Flash memory with *pgmspace.h* included.

- Differentiating data of multiple transmitter at one receiver

  1. Differentiating data from multiple transmitter in star network
     - We faced problem while receiving data from multiple transmitters to identify which data belongs to which transmitter so to solve this problem we used a pipenum function which will provide the pipe number from which it has received the data. By using pipe number we were able to differentiate multiple data's of different transmitters but this worked only in the star network.
  2. Differentiating data from multiple transmitter in mesh network

  - While implementation of mesh network we faced the same problem as in the star network of differentiating different data at the receiver end. So for this problem we used a string data who's first character will consist of the node ID or address of the transmitter data which is parsed at the receiver end for identification of the node ID which is send along with the transmitted string.

- Importing C file in C++ ATMEL project.
  - The gpio library written in .C and project main file is in .cpp format , so while using function form .C inrary in main the error is "Undefined refrence to function".
  - This is because Name mangling of compilers , linker can not link files with diffrent symbols . Compiler creates two diffrent symbols for same function in diffrent because of function oerloading in C++.
  - This bug is solved by including gpio.h header file inside extern "C" command which disables name mangling for functions unside that .h header file and treat it as a C file.

- Timing issue with SPI library
  - After sending data through MOSI pin and then making NRF24L01+ as RX or TX CE pin written with low and high value respectively.
  - After writing level value to pin it needs some time to stable and that much time MCU needs to be wait . so I have find that time for NRF24L01+ for its datasheet.

- Transmit and Receive float value in string
  - While transmitting a float data from the transmitter it was not able transmit float data from it. So we converted float data into character and stored it into string for sending it.

- Synchronization in Star network with duty cycling on both sides
  - We searched for diffrent protocols for synchronization like **RBS,TPSN,FTSP**.
  - Tried by flooding method in which Recevier hub sends some count value to each node and for that much count value that particular node goes in sleep and after sending all count value recevier goes into sleep for the count value which is minimum and after that it will wake up to take data form first node.

  - Tested with one transmitter and one receiver worked fine by putting both the TX & RX in sleep.
  - While implementing with more than one node it is not working with that much of accuracy because sometime it happens like node is transmitting but receiver in sleeep at that time & vice versa.One as for flooding each Node be a used as transrecevier with its own address properly but because power down timing accuracy some cases arises like all the node are in transmitting and nomone is listening and vice versa also.
  - So for this method we need a global clock that we don't have so we kept receiver continuously on and did duty cycling using power down API at each node.

- Serial communication between arduino nano(hub) and ESP32
  - While sending data from hub to ESP32 gateway we opt to transmit data serially from hub to gateway while doing that we faced lot of problems for synchronization of both. So we solved it by sending an acknowledgement from gateway to hub as soon as the data is received at the gateway side from the hub and also it will send a character while it gets connected to the available Wi-Fi network. So that hub can send data to it and esp32 is ready to receive data from it.

- It was a very tough challenge to upload bulk data to thingspeak server at once.
  - While uploading data to the thingspeak server the main challenge was to upload multiple channels in a single instant and to view it on the same screen. So we tried to upload multiple data to multiple channels of the thingspeak server but by doing so data cannot be viewed on a single screen so we uploaded different data's to different fields of the GUI. So that we can easily view all the uploaded data on a single screen.
  - Also we tried to implement JSON packet for uploading bulk data to the server but it didn't worked perfectly.

- After completing soldering of all smd components on the printed pcb it was ready to be implemented, while implementing it we found a hardware issue which was all about the LDO and its capacitor combination, we were using a non polarized capacitor, which was not that capable

to fulfill the need of the whole pcb. So the problem was solved by replacing the non polarized capacitor with a polarized capacitor.

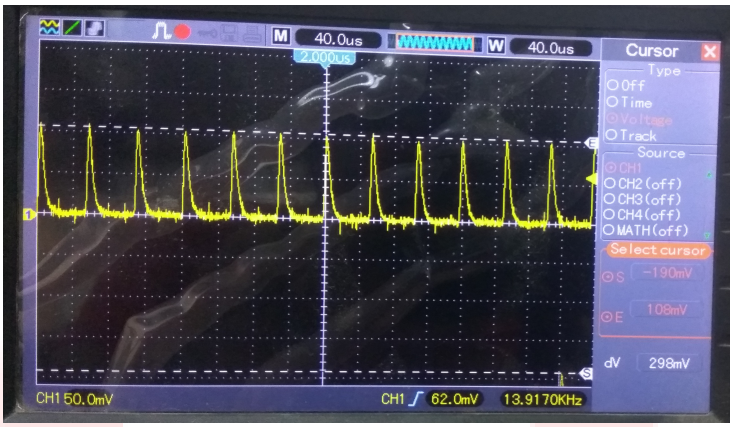


Figure 11. **CRO reading of change in voltage w.r.t. time**

- The above figure shows that the change in voltage with respect to time of CRO reading which we obtained.
- MOSFET Switching on Board
  - The bug is the code/api for initializing of NRF24L01+ is outside the while 1 loop ,and after that MOSFET switching is done. That is at the time of initialization nrf is not powered and because of nrf is not receving/transmitting data properly.
  - Switch on MOSFET outside while 1 loop and then call the API's of initialising of NRF24L01+.

Bibliography

[1] Sachin Gajjar, Nilav Choksi, Mohanchur Sarkar and Kankar Dasgupta, *Comparative analysis of Wireless Sensor Network Motes*, 2014.

[2] Ad Kamerman and Leo Monteban, *Low cost wireless sensor networks for environment monitoring*, 2015.

[3] ZHU Jian-hong, WANG Jun and LIU Di, *Design of A Wireless Sensor Network Node Based on nRF2401.*

[4] Sergey Y. Yurish, Javier Caete and Francisco Puerta, *Cost-effective Sensor Nodes for Wireless Sensor Networks*, 2012.

[5] S. S. Sonavane, V. Kumar and B. P. Patil, *Designing Wireless Sensor Network with Low Cost and Low Power.*

[6] Vidyasagar Potdar, Atif Sharif and Elizabeth Chang, *Wireless Sensor Networks: A Survey*, 2009.

[7] Rajinder Kumar Math and Nagaraj V Dharwadkar, *A Wireless Sensor Network Based Low Cost and Energy Efficient Frame Work for Precision Agriculture*, 2009.