LoRa BASED PROCESS MONITORING SYSTEM

Theme: Processing Industry

A PROJECT REPORT

Submitted by

SRIVIDHYA SUTHARSAN 2019504592
MEDIKONDURU PRAVALLIKA 2019504548
SUBALAKSHMI SARVANI 2019504593

EC5511-ANALOG AND DIGITAL COMMUNICATION LABORATORY

SEPTEMBER-DECEMBER 2021

BACHELOR OF ENGINEERING
IN
DEPARTMENT OF ELECTRONICS ENGINEERING

MADRAS INSTITUTE OF TECHNOLOGY
ANNA UNIVERSITY: CHENNAI 600 044

ANNA UNIVERSITY: CHENNAI 600 044

ACKNOWLEDGEMENT

We would like to thank **Dr. T. Thyagarajan**, Dean, Madras Institute of Technology, Anna University, Chennai for providing the college facilities.

We would like to thank our Head of the Department **Dr. M. Ganesh Madhan**, Madras Institute of Technology, Anna University, Chennai for permitting and encouraging us to do projects.

We would like to thank **Mrs.G.Sumithra**, Asst.Professor, Madras Institute of Technology, Chennai, for guiding us and giving the opportunity to do projects.

We would like to thank **Mr.K.Tamilarasan**, Teaching Fellow, Madras Institute of Technology, Chennai, for guiding us and giving the opportunity to do projects.

Place: Chennai Date: 14/12/2021

TABLE OF CONTENTS

INTRODUCTION:

Accuracy plays an important role in the field of process industries. This project aims at developing a LoRa based process monitoring system for industries so that any processing data that needs to be transmitted long distance can be done easily through this technology so that further statistical processing can be done depending on the application user.

AIM:

To assist the functioning of process industry and to facilitate its development using LoRa Technology for remote communication.

WHAT IS LORA:

LoRa is a low power wide area network technology, it expands as Long Range. Cellular wide area network technologies, like 4G and 5G which are most commonly used generally consume a lot of power, to overcome this defect an IoT technology which has wireless sensor nodes over a few kilometres which transmits sensor values, and which runs on a battery, which does not die at least for a couple of years. To achieve this LoRa technology was introduced.

Thus, LoRa is used to transmit data over long distance with low power. This technology uses chirp spread spectrum modulation technology, which makes the low power long range transmission possible. The LoRaWAN is the communication protocol and system architecture for the LORA technology, it expands as Long-Range Wide Area Network Technology. It can be divided as two parts LoRa nodes and LoRa gateways. LoRa nodes are devices with LoRa radio modules, along with sensors and microcontrollers, LoRa nodes consist of several LoRa gateways, which channel the data from nodes to network server, from where they are moved to several application servers. The LoRa nodes cannot interact with each other directly. The application servers also send back the information to LoRa nodes as feedback so that the process that is being done can be made perfect and hence accuracy in the development is highly increased.

ABOUT RADIO MODULES AND GATEWAYS OF LoRa:

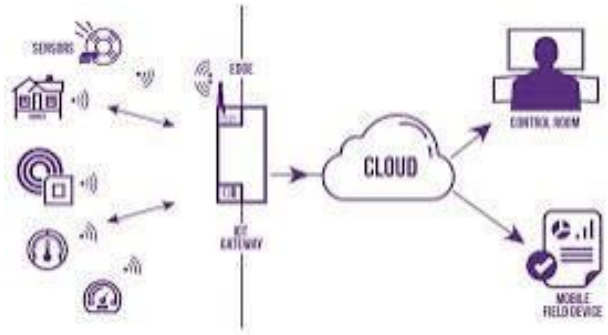
LoRa RADIO MODULE

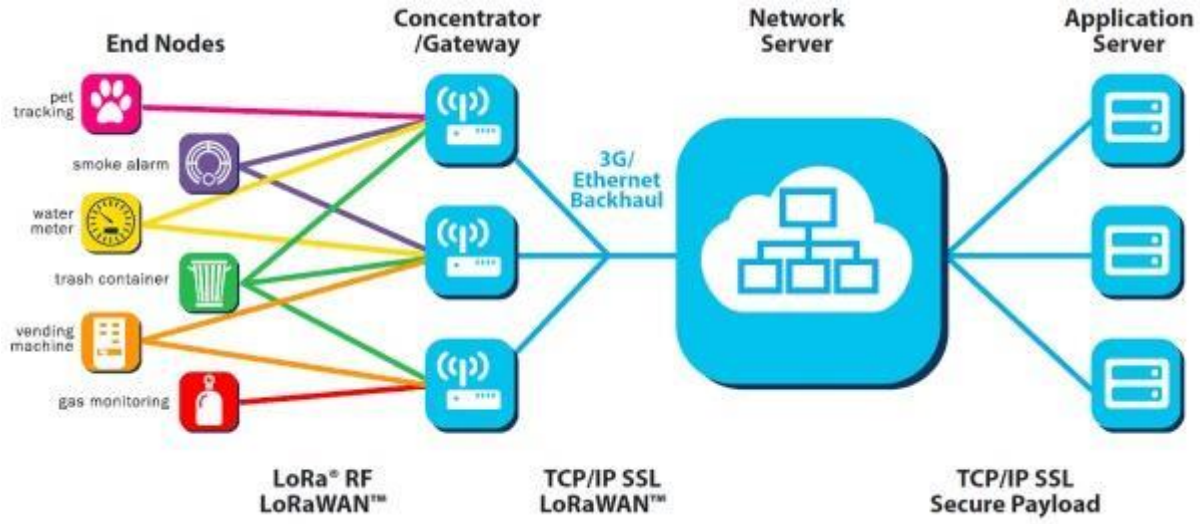
LoRa Radio Module is a type of long-range low data rate data radio modem based on Sx1276 from Semtech. It is a low-cost sub-1 GHz transceiver module designed for operations in the unlicensed ISM (Industrial Scientific Medical) and LPRD bands. Frequency spectrum modulation/demodulation, multi-channel operation, high bandwidth efficiency and anti-blocking performance make LoRa modules easy to realize the robust and reliable wireless link.

The module can work in two different modes: Standard mode and Star network mode. In the standard mode, it acts as transparent data radio modem which it communicates with the host at the pre-set data format without encoding / decoding needed. In start network mode, one module will be the configured to the central node and other modules are set to node modules. The communication between the central module and node module are bidirectional but the node modules cannot talk with each other. Please note that the module doesn't contain LoRaWAN protocol. Therefore, the star network feature of this module is used with itself protocol so it is not compatible with LoRaWAN

LORA GATEWAYS:

LoRaWAN Gateways are the link between the source of data – sensors in the environment – and people. They enable scale for the Internet of Things and are a key element in the chain that enables better visibility, reduction in costs, reduction in resource inputs, improved safety and better decision making.





ADVANTAGES OF LORA:

LoRa has the following advantages

1) Long range: Many miles on line-of-sight links.

2) Low power: Can run on battery for years.

3) Low cost: LoRa modules are pocket-friendly. It uses constant envelope modulation that brings lower cost and higher efficiency to the power amplifier.

4) Universal: Uses unlicensed bands that are globally available.

5) Bi-directional: Can send and receive data.

6) Network scalability: Easy to scale as it supports millions of messages per station. A single gateway can support thousands of end devices.

7) Easy commissioning: Easy to deploy in an existing network.

COMPONENTS REQUIRED:

1) ULTRASONIC SENSOR

An ultrasonic sensor is an electronic device that measures the distance of a target object by emitting ultrasonic sound waves, and converts the reflected sound into an electrical signal. Ultrasonic waves travel faster than the speed of audible sound (i.e., the sound that humans can hear). Ultrasonic sensors have two main components: the transmitter (which emits the sound using piezoelectric crystals) and the receiver (which encounters the sound after it has travelled to and from the target).

ULTRASONIC SENSOR

In order to calculate the distance between the sensor and the object, the sensor measures the time it takes between the emission of the sound by the transmitter to its contact with the receiver. The formula for this calculation is

$$D = ½ T \times C$$

(Where D is the distance, T is the time, and C is the speed of sound ~ 343 meters/second). We divide by 2 because the distance will be twice the original distance.

Ultrasonic sensors are used primarily as proximity sensors. Ultrasonic sensors are also used in robotic obstacle detection systems, as well as manufacturing technology.

This is the HC-SR04 ultrasonic distance sensor. This economical sensor provides 2cm to 400cm of non-contact measurement functionality with a ranging accuracy that can reach up to 3mm.

There are only four pins on the HC-SR04: VCC (Power), Trig (Trigger), Echo (Receive), and GND (Ground).

Ultrasonic Sensor Pin Configuration

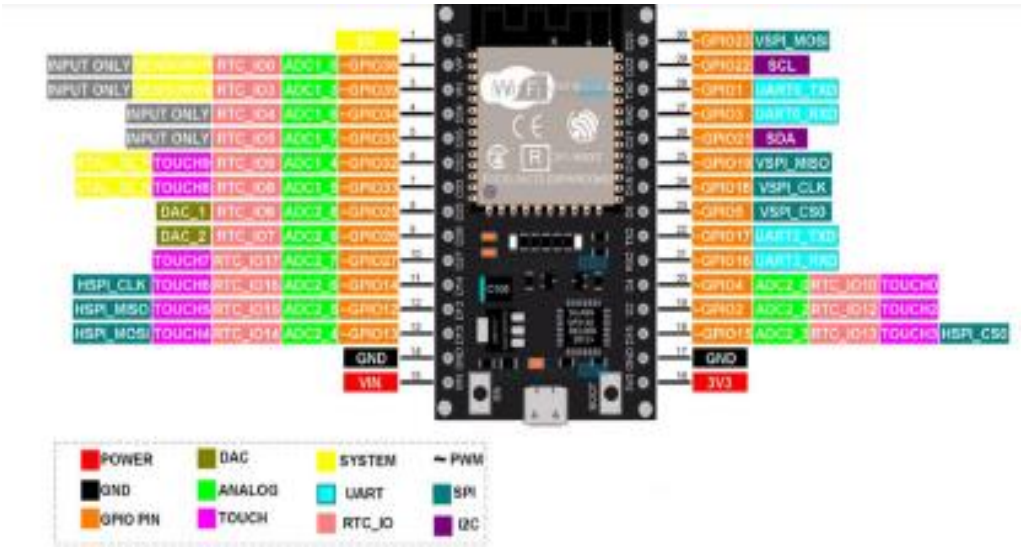
PIN NAME	PIN DESCRIPTION
Vcc	Voltage supply +5V
Trigger pin	Signal output
Echo pin	Singal input
Gnd	Ground

2) ESP32 MODULE

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.

It is designed for mobile, wearable electronics, and Internet-of-Things (IoT) applications. It features all the state-of-the-art characteristics of low-power chips, including fine-grained clock gating, multiple power modes, and dynamic power scaling. For instance, in a low-power IoT sensor hub application scenario, ESP32 is woken up periodically only when a specified condition is detected. Low-duty cycle is used to minimize the amount of energy that the chip expends. The output of the power amplifier is also adjustable, thus contributing to an optimal trade-off between communication range, data rate and power consumption.

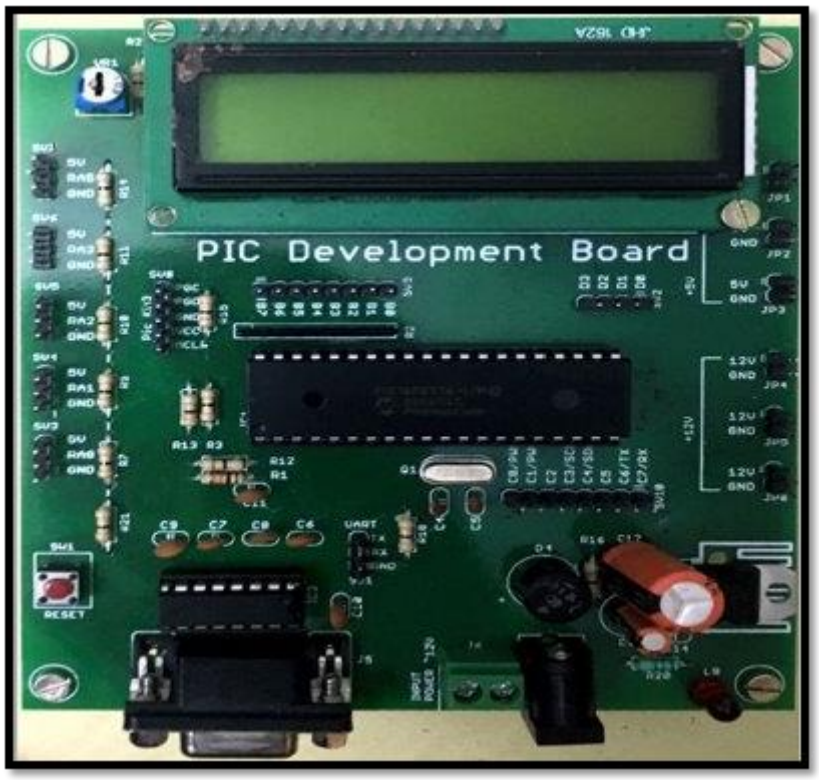
ESP32 is a highly-integrated solution for Wi-Fi-and-Bluetooth IoT applications, with around 20 external components. ESP32 integrates an antenna switch, RF balun, power amplifier, low-noise receive amplifier, filters, and power management modules. As such, the entire solution occupies minimal Printed Circuit Board (PCB) area.



ESP32 uses CMOS for single-chip fully-integrated radio and baseband, while also integrating advanced calibration circuitries that allow the solution to remove external circuit imperfections or adjust to changes in external conditions. As such, the mass production of ESP32 solutions does not require expensive and specialized Wi-Fi testing equipment.

3) PIC Microcontroller development board:

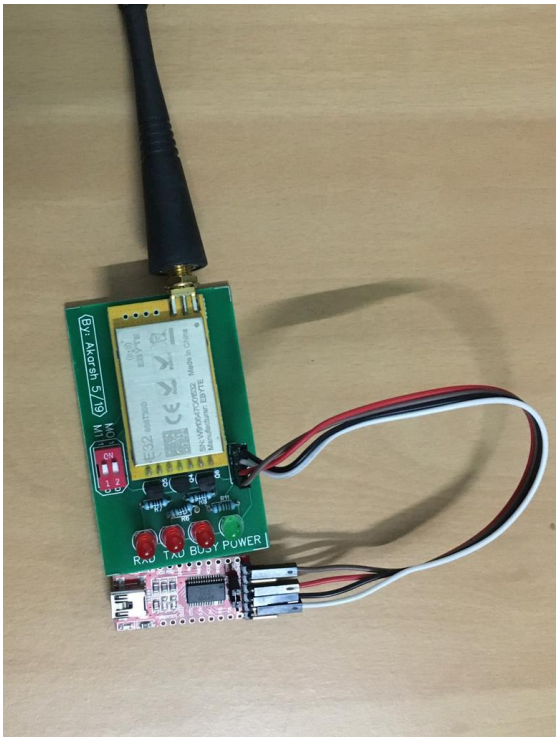
This microcontroller is used for interfacing the ultrasonic sensor which act as a node in the process monitoring system.



4) E32 LoRa module with antenna:

It is a wireless transceiver module, operates at 865-867 MHz which is an ISM band (No license required) used for LoRa communication. The module adopts LoRa spread spectrum technology.

This is developed by EByte techonolgies.



5)12 V DC power supply:

This is used to power the PIC microcontroller for displaying the process data in the LCD display present in the development board.

6) Box:

Our process reads the dimensions of a box which is transmitted to the network server using LoRa gateway.

7) FTDI board:

The FTDI USB to TTL serial converter module is a UART (universal asynchronous receiver-transmitter) board used for TTL serial communication. We use this to serially communicate the data.

8) PICKit 3 programmer:

PIC programmer (PIC kit 3.5) is used to load the .hex file to the microcontroller.

9) Ra02 – LoRa module (433MHz):

This is a LoRa module which operates at 433MHz ISM band. We used this just for trying the LoRa communication but implemented our project using E32 LoRa module which operates at 865-867MHz.

10) ESP8266:

This module is used as a microcontroller to connect to the internet so that data can be transmitted using LoRa modulation.

SOFTWARE REQUIRED:

1) MikroC:

Ultrasonic sensor node interfacing code with PIC microcontroller.

2) MongooseOS:

Mongoose OS is an Internet of Things (IoT) Firmware Development Framework. It supports low power, connected microcontrollers such as: ESP32, ESP8266,etc.One of the main features is the User friendly Over the Air (OTA) updating of embedded ICs.

3)MQTT(Message Queuing Telemetry Transport):

MQTT is an OASIS standard messaging protocol for the Internet of Things (IoT) which is based on publish subscribe model. Anyone who subscribes to that model will be able to receive the data transmitted.

4) PICKit 3 Programmer:

To load the .hex file to the microcontroller for the execution.

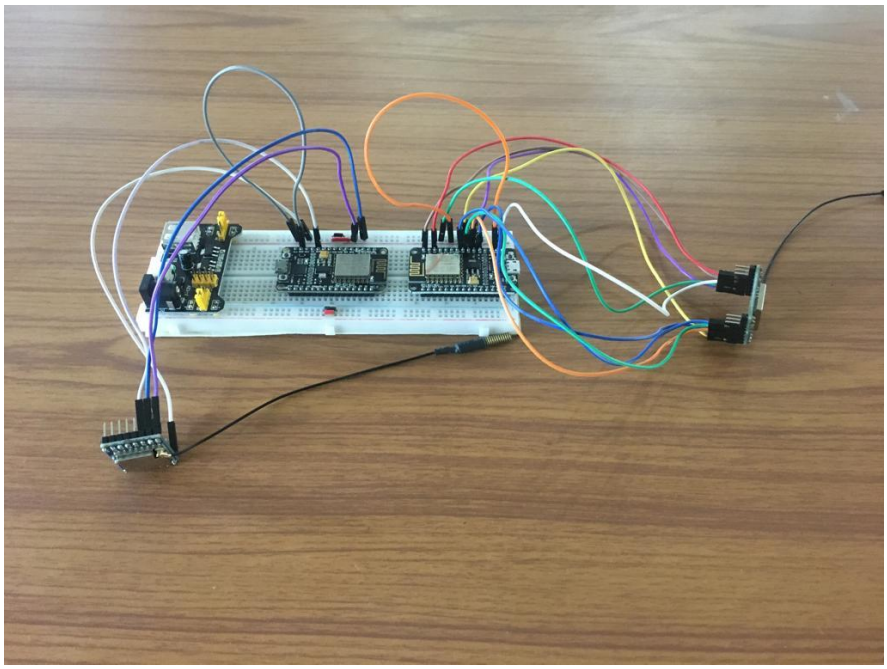
5) Arduino IDE:

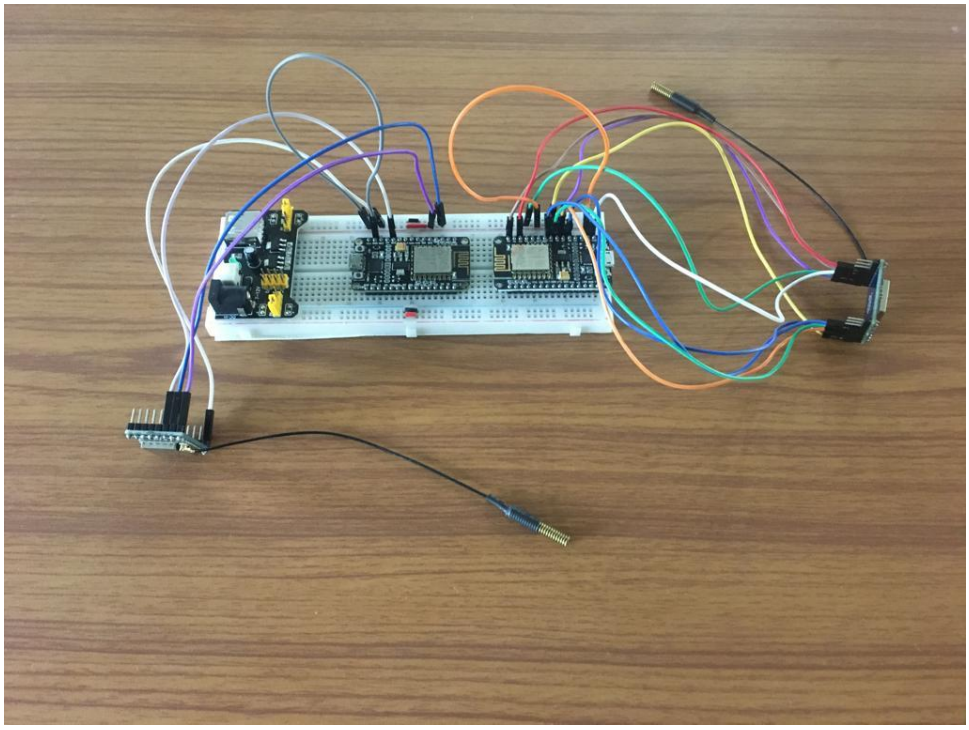
This is used for coding the ESP8266 for communicating using Ra02 LoRa module.

PROJECT WORK:

1) We first tried the basic communication of a LoRa transmitter and a receiver using Ra02 LoRa module which operates at 433MHz.

The initial prototype for the implementation with Ra02 is shown below:

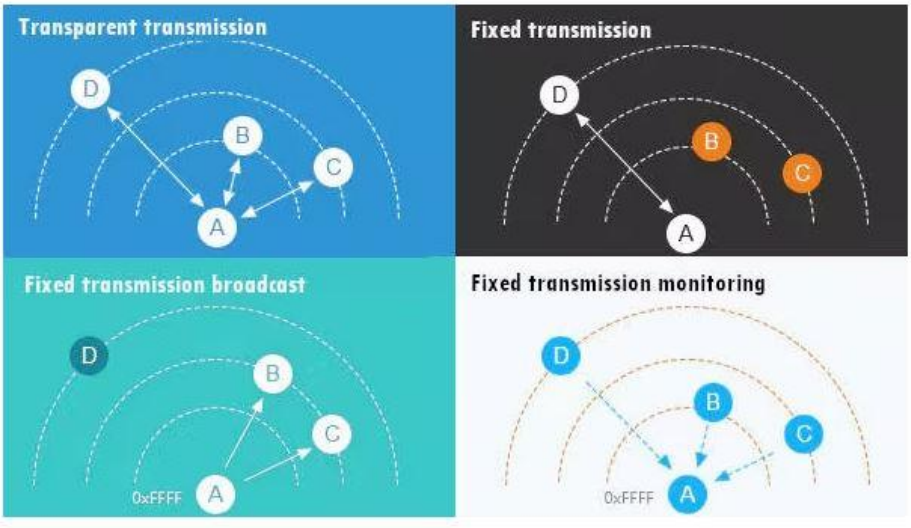




The data has been sent and received successfully.

At the initial stages we planned to use The Things Network Gateway to communicate to the network server and to the application server but since we were not able to use the device EUI for registering in the application we explored the E32 module which can be easily used.

There are different modes of transmission that we can make using LoRa E32 module.



LoRa E32 transmitting scenarios

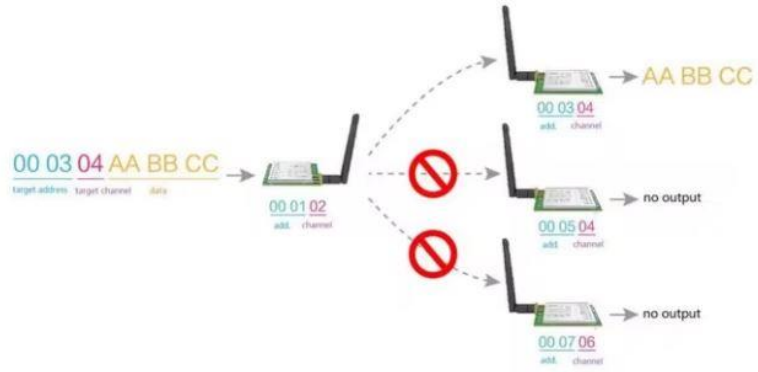
1) Transparent transmission:

In this mode by default, we can send message to all device of same configured address and channel.

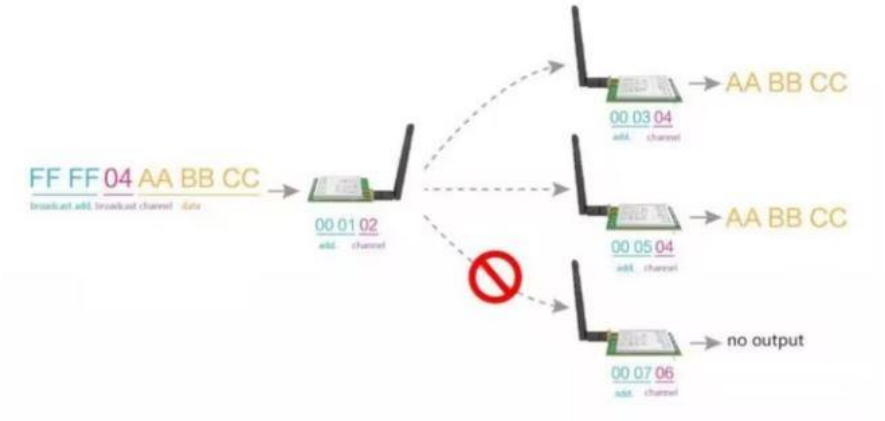
2) Fixed transmission:

In this type of transmission, we can specify an address and a channel where we want to send the message. We can send message to a:

- Specified device with a predetermined Address Low, Address High and Channel.

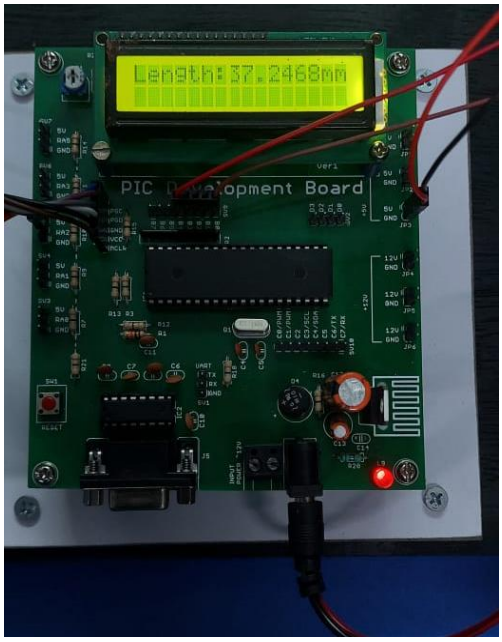


- Broadcast a message on predetermined Channel.



In this project we have set the configuration as transparent transmission.

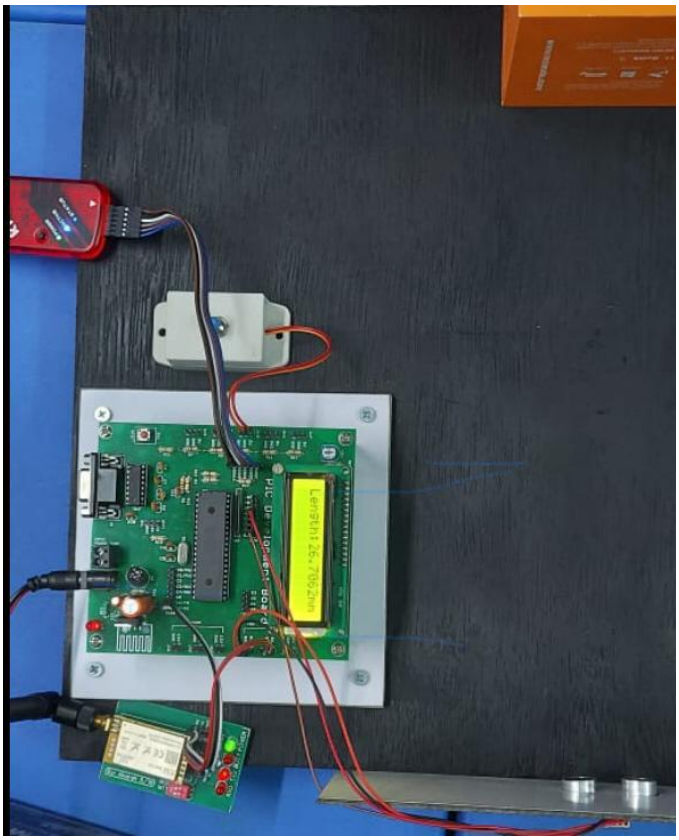
The ultrasonic sensor which acts as a node measures the dimensions of the box (processed/manufactured product) using the embedded software and it displays the dimensions in the LCD display that is inbuilt in the PIC development board.



This data is then sent from the PIC microcontroller to the E32 module via UART communication to the E32 LoRa transceiver for transmission wirelessly. LoRa gateway (E32 LoRa with ESP32) on the other side can receive and pass it to the MQTT network server. This is done by an application that resides in the ESP32 microcontroller which runs mongoose-OS operating system. The application receives the data from E32 LoRa module via UART and publishes to a MQTT server at topic "vu3uje/device1". In our case we use public server at "mqtt.eclipseprojects.io:1883".

Any user application subscribes to this topic will be able to receive the data. From this point, the process may vary depending on the requirement of the application needs which is outside the scope of implementation for this project.

Node:



PIC code for Node:

```
// Project : LoRa based Process monitoring system
// LoRa Node to measure object dimensions
// Result=> {"data":{
//              "length":142.0,
//              "width":92.2,
//              "height":51.5 }
//          } // example data
//
// Developed by : Srividhya Sutharsan, Madikonduru Pravallika, Subalakshmi Sarvani
// Department : ECE Department, MIT Madras
//
// LCD module connections
sbit LCD_RS at RE2_bit;
sbit LCD_RW at RE1_bit;
sbit LCD_EN at RE0_bit;
sbit LCD_D4 at RD4_bit;
sbit LCD_D5 at RD5_bit;
```

```
sbit LCD_D6 at RD6_bit;
sbit LCD_D7 at RD7_bit;sbit LCD_RS_Direction at TRISE2_bit;
sbit LCD_EN_Direction at TRISE0_bit;
sbit LCD_RW_Direction at TRISE1_bit;
sbit LCD_D4_Direction at TRISD4_bit;
sbit LCD_D5_Direction at TRISD5_bit;
sbit LCD_D6_Direction at TRISD6_bit;
sbit LCD_D7_Direction at TRISD7_bit;
// End LCD module connectionsfloat val;
float calib = 423.0;
char txt[7];
char* strStrt = "{\"data\":{";
char* strlng  = "\"length\":";
char* strwdt  = ",\"width\":";
char* strhgt  = ",\"height\":";
char* strEnd  = "}}\"";
char* out = "";
int faces = 0;void main()
{
  ADCON1 = 7;
  CMCON = 7;
  TRISE = 0;
  TRISE1_Bit = 0;
  TRISA1_Bit = 0;  UART1_Init(9600);
  Lcd_Init();
  Lcd_Cmd(_LCD_CLEAR);           // Clear display
  Lcd_Cmd(_LCD_CURSOR_OFF);      // Cursor off  Lcd_Out(1,1,"Welcome to");
  Lcd_Out(2,1,"ECE dept-MIT");
  Delay_ms(3000);
  Lcd_Cmd(_LCD_CLEAR);

  TRISB = 0b00010000;            //RB4 as Input PIN (ECHO)

  T1CON = 0x10;                  //Initialize Timer Module  while(1)
  {
    TMR1H = 0;                   //Sets the Initial Value of Timer
    TMR1L = 0;                   //Sets the Initial Value of Timer

    PORTB.F0 = 1;                //TRIGGER HIGH
    Delay_us(10);                //10uS Delay
    PORTB.F0 = 0;                //TRIGGER LOW

    while(!PORTB.F4);            //Waiting for Echo
    T1CON.F0 = 1;                //Timer Starts
    while(PORTB.F4);             //Waiting for Echo goes LOW
    T1CON.F0 = 0;                //Timer Stops

    val = (TMR1L | (TMR1H<<8));   //Reads Timer Value
    val = val /5.882;                 //Converts Time to Distance
    if(val <= 2000.0)          //Check whether the result is valid or not
    {
      //IntToStr(a,txt);
      val = calib - val;                    //Distance Calibration
      FloatToStr_FixLen(val, txt, 7) ;
      Ltrim(txt);
      Lcd_Cmd(_LCD_CLEAR);
      Lcd_Out(1,1,"Length:");
      Lcd_Out(1,8,txt);
      Lcd_Out(1,15,"mm");
    }
    else
    {
      Lcd_Cmd(_LCD_CLEAR);
      Lcd_Out(1,1,"Out of Range");
```

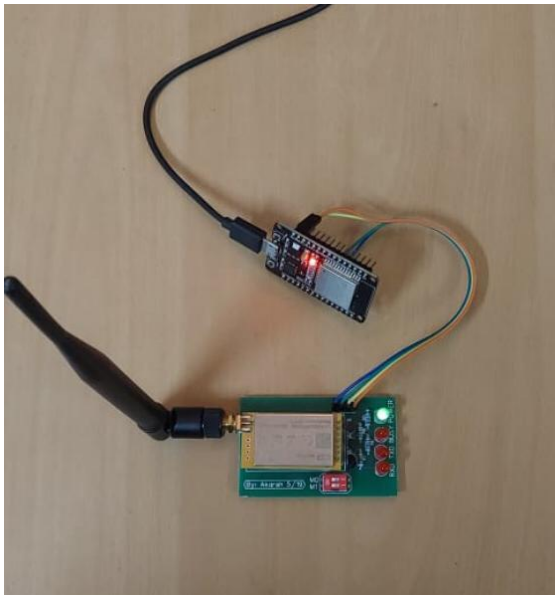
```
    }
    if (faces == 0) {
       strcat(out,strStrt);
       faces++;
    }
    if (TRISA1_Bit == 1) {
       switch (faces) {
       case 1: strcat(out,strlng);
               strcat(out,txt);
       case 2: strcat(out,strwdt);
               strcat(out,txt);
       case 3: strcat(out,strhgt);
               strcat(out,txt);
               strcat(out,strEnd);
               UART1_Write_Text(out);
               faces = 0;
       };
       faces++;
       out = "";
       Delay_ms(200);
    }
  }
}
```

Gateway :

Gateway device is made of ESP32 interfaced with E32 LoRa module via UART. Uses Mongoose-OS and custom written UART to MQTT gateway code to route the RF packet data in to the topic “vu3uje/device1” in MQTT server mqtt.eclipseprojects.io:1883.

The code for the gateway was developed using “mos ui “tool by cloning an empty project and implemented the logic for UART to MQTT transfer. Once the code is developed the same was compiled and flashed to the device using following commands.

- mos build –platform esp32
- mos flash

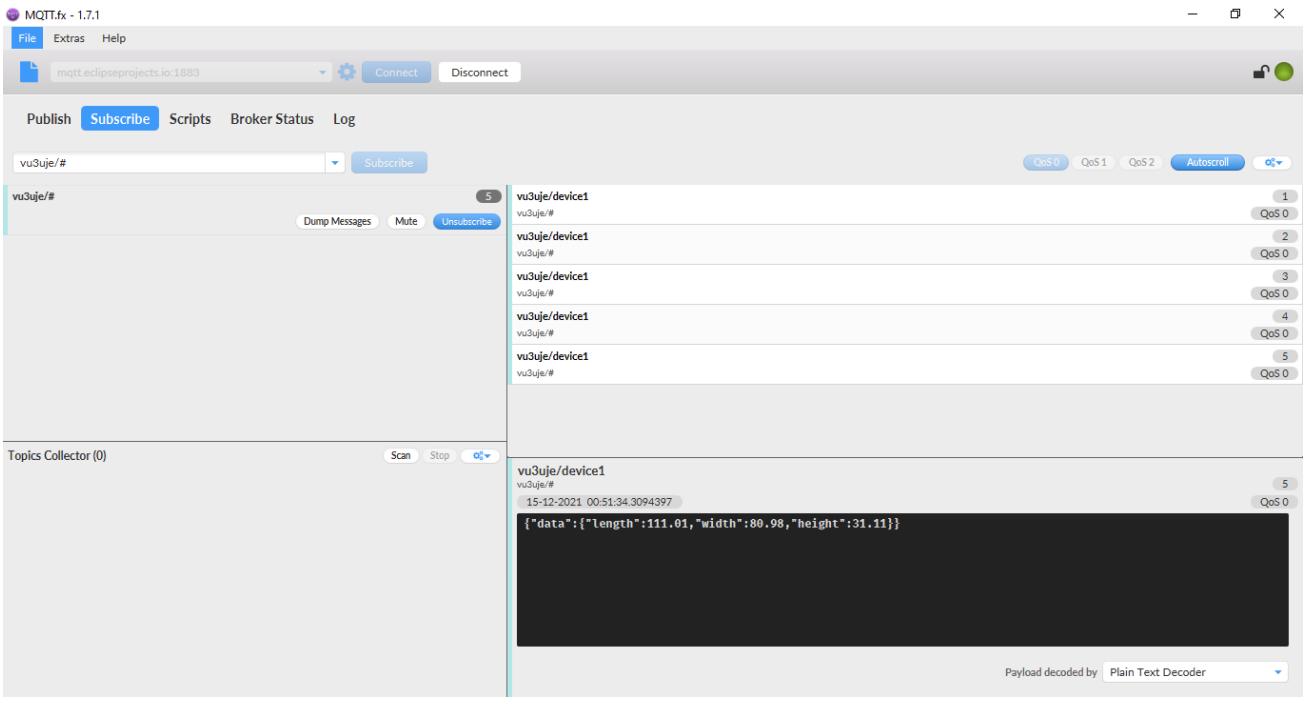


- Gateway code stored in **init.js** file in **fs** folder is given below.

Gateway Code:

```
load('api_timer.js');
load('api_uart.js');
load('api_sys.js');
load('api_mqtt.js');let uartNo = 1;    // Uart number used for this example
let rxAcc = '';    // Accumulated Rx data, will be echoed back to Tx
let value = false;// Configure UART at 9600 baud
UART.setConfig(uartNo, {
  baudRate: 9600,
  esp32: {
    gpio: {
      rx: 16,
      tx: 17,
    },
  },
});// Set dispatcher callback, it will be called whenver new Rx data or space in
// the Tx buffer becomes available
UART.setDispatcher(uartNo, function(uartNo) {
  let ra = UART.readAvail(uartNo);
  if (ra > 0) {
    // Received new data: print it immediately to the console, and also
    // accumulate in the "rxAcc" variable which will be echoed back to UART later
    let data = UART.read(uartNo);
    print('Received UART data:', data);
    let ok = MQTT.pub('vu3uje/device1', data);
    print('mqtt message sent to server?', ok);
    rxAcc += data;
  }
}, null);// Enable Rx
UART.setRxEnabled(uartNo, true);
```

Output Result in MQTT server :



Practical Applications in Process Industry

Data thus received in the MQTT server is subscribed and stored in a database for analytical purposes. We may look at the data as a time series data so the statistical approach to quality monitoring can be implemented. In our case the dimensions shall need to fall within the given specification limits. This will be usually performed with the help of Statistical Process Control approach wherein computation of Pp, Ppk, Cp, Cpk are measured. This interprets to Process Performance metrics and Process Capability metrics.

In an ideal situation, when receiving such data the data falls within normal distribution so that most of the samples lie within the normal/specified design value. In any case if it deviates outside -3sigma on the lower side or +3sigma on the upper side indicates that the samples are outside the limits.

This can be given as feedback to the process mechanism to correct the process settings.

Conclusion

This project was developed in various iterations trying out different approaches to implement the solution. During the project, we confined to implementing the RF front end using Ebyte/TI's E32 LoRa modules which support UART communication. The LoRa node was made to obtain object measurements of product in test and send the measurements in JSON format to the MQTT server. Mongoose OS helped implementing the gateway software at ease, which also provides opportunity to take the data to AWS IoT or Google Cloud Platform IoT or Samsung Artik IoT solutions. Overall the project gave us an opportunity to learn about RF transmission using LoRa, various serial data communication protocols and Statistical approach to process monitoring and control in the Industry.

References:

1) International Journal of Engineering and Technical Research (IJETR) ISSN: 2321-0869 (O) 2454-4698 (P), Volume-3, Issue-9, September 2015 - PIC Microcontroller Based Ultrasonic Distance Measurement - Y.Ege, H.Çıtak, M.Çoramık, M. Kabadayı, O.Kalender https://www.researchgate.net/publication/286440603_PIC_Microcontroller_Based_Ultrasonic_Distance_Measurement
2) E32 LoRa module user manual https://www.ebyte.com/en/pdf-down.aspx?id=672
3) Mongoose-OS developer documentation - https://mongoose-os.com/docs/mongoose-os/quickstart/setup.md
4) MQTT server – specifications - https://mqtt.org/mqtt-specification/
5) ESP32 module - https://www.espressif.com/en/products/devkits/esp32-devkitc
6) Applying the MQTT Protocol on Embedded System for Smart Sensors/Actuators and IoT Applications - https://ieeexplore.ieee.org/abstract/document/8619981
7) Engineering Statistics Handbook – Section 6 – Process or Product monitoring and control - https://www.itl.nist.gov/div898/handbook/pmc/pmc.htm
8) Communication protocols – UART, SPI, I2C communications for embedded software http://web.mit.edu/6.111/www/f2017/handouts/L13_4.pdf