Final Report
Design Credit Evaluation

Design and Developed a Trans-Receiver System using LoRa Technology

1. Aims and Objectives:

a) To design and implement mid-range (2 km) transceiver capable of LoS communication with a data rate of 9.6 kbps using low power LoRa technology.
b) Design an interfacing circuit to transmit data from various analog sensors like ECG Sensor, Heart Beat Sensor, Temperature Sensor, etc.
c) Try to achieve data transmission without using the Internet.

2. Introduction:

LoRa technology is a wireless communication technology that uses a radio modulation technique that allows long-range transmissions at low data-rates. The technology is designed to be used in Internet of Things (IoT) applications, and is particularly well suited to applications where power consumption is a key concern.

A LoRa module is a transceiver that is used for communication over a long range. It uses the LoRaWAN protocol to send and receive data. The module can be used to communicate with other LoRa modules or with a LoRa gateway.

3. Materials Used:

a) LoRa Module x 2
b) ECG Sensor x 1
c) Heart Rate Sensor x 1
d) Temperature Sensor x 1
e) Ultrasonic Sensor x 1
f) Breadboard x 1
g) Jumper Wires

4. Experimental Setup & Results:

Two sets of experiments were performed. In the first set of experiments performed in which the transmitter were kept stationary on the top of Department of Electrical
Engineering building and the data rate is set to 2.5kbps. While the receiver was mounted on a moving vehicle. In this setup, a communication range of 2 km was obtained with the packet loss of 46.42%.
And in the second set we have changed the data rate to 9.6kbps and obtained a communication range of 2 km with the packet loss of 56.3%.

Data Rate = **2.4kbps**

Destination	**Distance**	**% Packet Loss**
EE Department	0m (Source)	0%
Shamiyana Cafe	275m	0%
CSE Department	500m	1.9%
Lecture Hall	741m	31.06%
Faculty Quarters	800m	34.56%
Clock Tower	916m	37.53%
Knowledge Tree	1100m	41.33%
AI Department	2000m	46.42%

Data Rate = **9.6kbps**

Destination	**Distance**	**% Packet Loss**
EE Department	0m (Source)	0%
Shamiyana Cafe	275m	0.3%
CSE Department	500m	2.75%
Lecture Hall	741m	34.46%
Faculty Quarters	800m	37.57%
Clock Tower	916m	41.53%
Knowledge Tree	1100m	47.43%
AI Department	2000m	56.3%

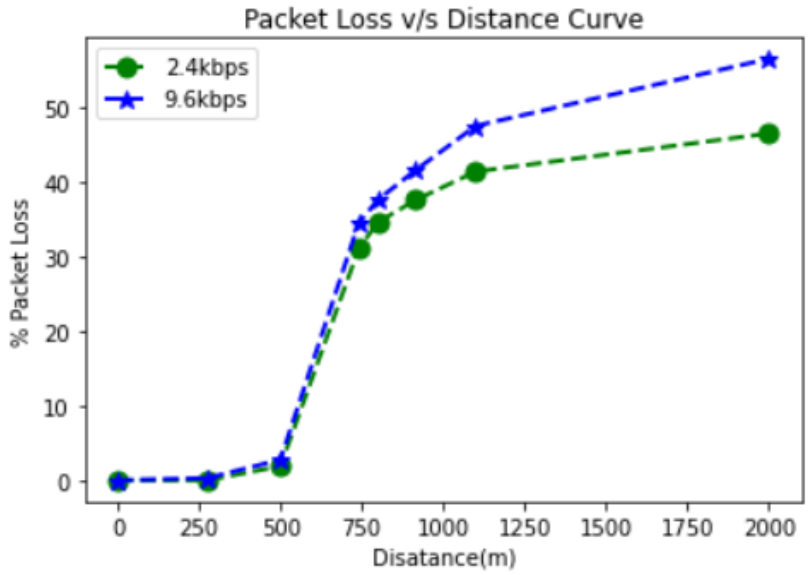

Fig.1 Packet Loss v/s Distance Curve

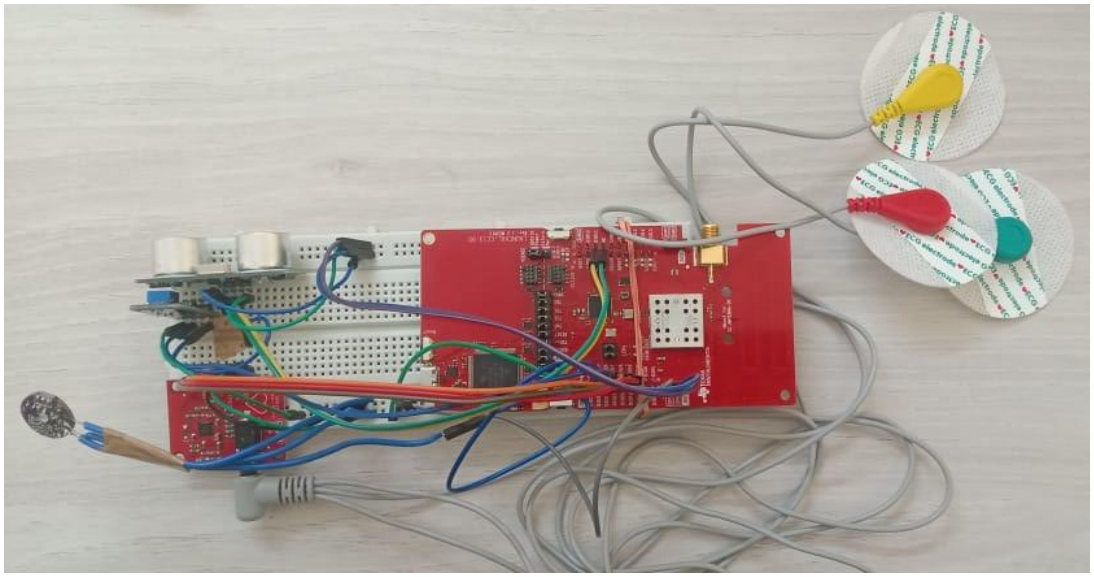

Fig. 2 Actual Circuit of Transmitter Side

Fig. 3 Actual Site Image (Department of Electrical Engineering)

We have started from the Department of Electrical Engineering and reached the AI and Data Science Department which is at a distance of 2km via Faculty Quarters which is at a distance of 800m.

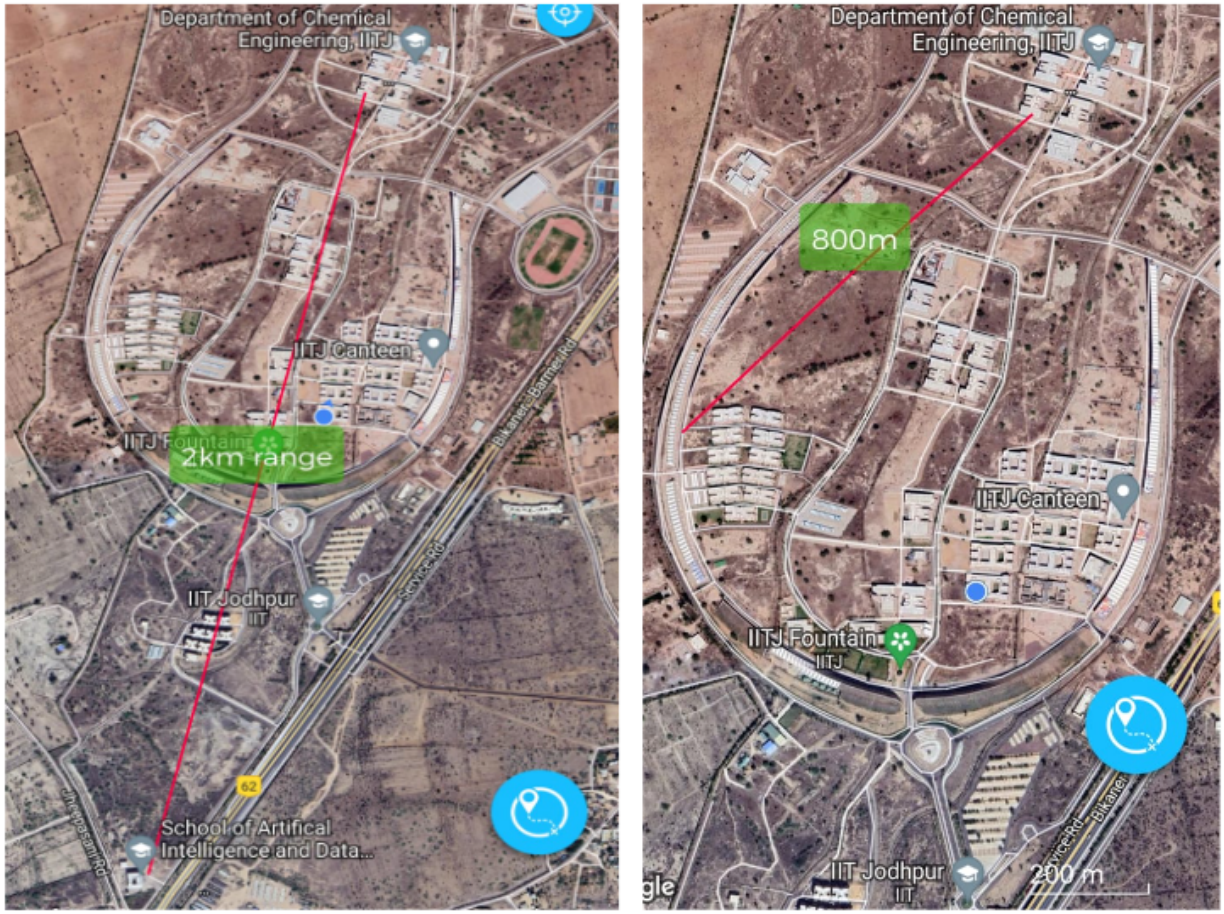

Fig.4 AI Department (2km) & Faculty Quarters (800m)

5. Source Code for Transmitter:

```
// Code for Tx
#include "EasyLink.h"
EasyLink_TxPacket txPacket;
EasyLink myLink;

#define HGM A5
#define LNA_EN A6
#define PA_EN A7

uint16_t value;
int analog_val;

// c_packet have 30 elements first 10 elements contains digits of number of packet
// transmitted successfully from transmitter
// in unsigned long int format
char c_packet[30] =
{'0','0','0','0','0','0','0','0','0','0',';','S',';','0','0','0','0','0',';','0','0
','0','0','0','0',' ','R','/','H','r'};
String s = ""; // string to contain sensor data
unsigned long int counter = 1; // number of packet sent successfully, range is [0,
4294967295] 10 digits beyond it max range counter will reset to 0
float t; // stores current time
float f = 1; // frequency of running the program
float dt = 1/f; // time period in seconds of running program i.e. void loop
float loop_timer = dt*1000000; // dt in microseconds
bool string_validity = false; // a valid string is received from the sensor


// Variables for sensors
int Signal;                // Store incoming ADC data. Value can range from 0-1024
int Threshold = 550;       // Determine which Signal to "count as a beat" and which
to ignore.

const int trig = 38; // D20
const int echo = 37; // D19
int distance;  // This variable will store the object's distance from the sensor.

void setup() {
  Serial.begin(9600);

  // Ultrasonic
  pinMode(trig, OUTPUT);
  pinMode(echo, INPUT);
```

```
  // ECG
  pinMode(27, INPUT); // Setup for leads off detection LO + D19
  pinMode(28, INPUT); // Setup for leads off detection LO - D20

  // Code for data transmission
  pinMode(RED_LED, OUTPUT);
  pinMode(GREEN_LED, OUTPUT);
  //Enable CC1190 Module ON
  pinMode(HGM, OUTPUT);
  pinMode(LNA_EN, OUTPUT);
  pinMode(PA_EN, OUTPUT);
  digitalWrite(HGM, HIGH);
  digitalWrite(LNA_EN, LOW);
  digitalWrite(PA_EN, HIGH);
  digitalWrite(RED_LED, LOW);

  digitalWrite(GREEN_LED, LOW);
  // begin defaults to EasyLink_Phy_50kbps2gfsk
  myLink.begin();
  // Set the destination address to 0xaa
  txPacket.dstAddr[0] = 0xaa;
  t = micros();
}

void loop() {
  //  int ecgdata = -1;

  // Heart Rate Sensor
  Signal = analogRead(A2); // Read the sensor value

  // Temperature Sensor
  float temp = analogRead(A3);

  // Ultrasonic Sensor
  digitalWrite(trig, LOW);
  delayMicroseconds(2);
  digitalWrite(trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig, LOW);

  // ECG Sensor
  //  if ((digitalRead(27) == 1) || (digitalRead(28) == 1)) {
  //    Serial.println('!');
  //  }
  //  else {
  // send the value of analog input 0:
```

```
  int ecgdata = analogRead(A0);
  //    Serial.println(analogRead(A0));
  //  }

  // Calculating the distance by multiplying the speed of sound by the time of echo
  distance = pulseIn(echo, HIGH) * 0.034 / 2;
  String ultrasonic = String(distance) + " cm";
  Serial.println(ultrasonic);


  // read analog volt from sensor and save to variable temp
  temp = temp * 0.48828125;
  // convert the analog volt to its temperature equivalent
  Serial.print("TEMPERATURE = ");
  Serial.print(temp); // display temperature value
  Serial.print("*C");
  Serial.println();

  String heartRateData = String(Signal / 10) + " BPM";
  Serial.println(heartRateData);                // Send the signal value to serial
plotter

  // Collecting All Data
  String data = String(Signal / 10) + "-" + temp + "-" + String(distance) + "-" +
String(ecgdata) + "!";
  Serial.println(data);

  // Code for transmission of data

  // Copy Counter Value to c_packet;
  String counter_str = String(counter);
  int n = counter_str.length();
  if(n>=10) {
    counter = 0;
    counter_str = "0";
  }
  int j=n-1;
//  Serial.println(counter_str);
  for(int i=9; i>=0 && j>=0; i--) {
    c_packet[i] = counter_str[j];
    j--;
  }
  String test = "";
  for(int i=9; i>=0; i--) {
    test+=c_packet[i];
  }
  Serial.println(test);
```

```
  for (int i = 0; i <= data.length(); i++) // element no 10 is ; so copy in element
no 11 to last i.e 29 so maximum 19 characters can be sent
  {
    c_packet[i + 11] = data[i];
  }
  // copy data to be sent
  memcpy(&txPacket.payload, &c_packet, sizeof(c_packet));
  // Set the length of the packet
  txPacket.len = sizeof(c_packet);
  // Transmit immediately
  txPacket.absTime = EasyLink_ms_To_RadioTime(0);
  // read transmission status
  EasyLink_Status status = myLink.transmit(&txPacket);
  if (status == EasyLink_Status_Success) // if transmission is successful
  {
    counter++; // increase data sent counter by one
    // string correct and data transmission successful
    // green led blink & red led off
    digitalWrite(GREEN_LED, HIGH);
    delay(50);
    digitalWrite(GREEN_LED, LOW);
    delay(50);
    digitalWrite(RED_LED, LOW);
  }
  else
  {
    // string correct but transmission failed
    // green led blink & red led on
    digitalWrite(GREEN_LED, HIGH);
    delay(50);
    digitalWrite(GREEN_LED, LOW);
    delay(50);
    digitalWrite(RED_LED, HIGH);
  }

  delay(50);
}
```

6. Source Code for Receiver:

```
// program for Rx
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////
#include "EasyLink.h"
EasyLink_RxPacket rxPacket;
EasyLink myLink;
```

```
// char c_packet[19]; // number of packet sent by transmitter(5 digits); analog
value(4 digits)
char c_packet[30];
#define HGM A5    //
#define LNA_EN A6 // low noise amplifier pin
#define PA_EN A7  // power amplifier enable pin
float t;          // stores current time in microseconds
unsigned long int num_packet_received_from_tx = 0;
unsigned long int num_packet_sent_from_tx = 0;
unsigned long int num_of_packet_loss = 0;
float percentage_of_packet_loss = 0;
unsigned long int num_packet_received_from_tx1 = 0;
unsigned long int num_packet_sent_from_tx1 = 0;
unsigned long int num_of_packet_loss1 = 0;
float percentage_of_packet_loss1 = 0;
unsigned long int num_packet_received_from_tx2 = 0;
unsigned long int num_packet_sent_from_tx2 = 0;
unsigned long int num_of_packet_loss2 = 0;
float percentage_of_packet_loss2 = 0;

void setup()
{
    Serial.begin(9600);
    // begin defaults to EasyLink_Phy_50kbps2gfsk
    // Enable c_packet1190 Module ON
    pinMode(HGM, OUTPUT);
    pinMode(LNA_EN, OUTPUT);
    pinMode(PA_EN, OUTPUT);
    digitalWrite(HGM, HIGH);
    digitalWrite(LNA_EN, HIGH);
    digitalWrite(PA_EN, LOW);
    myLink.begin();
    Serial.println(myLink.version());
    t = micros();
}

unsigned long long int getNumOfPacketSent(char data[]) {
  unsigned long long int num=0;
  int itr=0;
  for(int i=9; i>=0; i--) {
    num+=(data[i]-'0')*pow(10, itr);
    itr++;
  }
  return num;
}

void loop()
```

```
{
    for (int i = 0; i < 30; i++)
    {
        c_packet[i] = '-';
    }
    // rxTimeout is in Radio time and needs to be converted from miliseconds to
RadioTime
    rxPacket.rxTimeout = EasyLink_ms_To_RadioTime(3000);
    // Turn the receiver on immediately
    rxPacket.absTime = EasyLink_ms_To_RadioTime(0);
    // check status of receiving and decoding the received packet
    EasyLink_Status status = myLink.receive(&rxPacket);
    if (status == EasyLink_Status_Success)
    {
        memcpy(&c_packet, &rxPacket.payload, sizeof(c_packet)); // copy receive
payload to c_packet
        // if payload received is from device 170 (it is the id of device Tx
(0Xaa))
        int tx_addr = rxPacket.dstAddr[0];
        Serial.print("Transmitter Address: ");
        Serial.print(tx_addr);
        Serial.println();

//        num_packet_sent_from_tx = 1 * (int(c_packet[9]) - 48) + 10 *
(int(c_packet[8]) - 48) + 100 * (int(c_packet[7]) - 48) + 1000 * (int(c_packet[6])
- 48) + 10000 * (int(c_packet[5]) - 48) + 100000 * (int(c_packet[4]) - 48) +
1000000 * (int(c_packet[3]) - 48) + 10000000 * (int(c_packet[2]) - 48) + 100000000
* (int(c_packet[1]) - 48) + 1000000000 * (int(c_packet[0]) - 48);
        num_packet_sent_from_tx = getNumOfPacketSent(c_packet);
//Serial.print(c_packet[9]);
        Serial.print("Packets Sent: ");
        Serial.print(num_packet_sent_from_tx);
        Serial.println();

        num_packet_received_from_tx++;
        if (num_packet_received_from_tx > num_packet_sent_from_tx) // in case the
tx0 is reset in between the communication to avoid negative packet loss
        {
            num_packet_received_from_tx = num_packet_sent_from_tx;
        }

        Serial.print("Packets Received: ");
        Serial.print(num_packet_received_from_tx);
        Serial.println();

        num_of_packet_loss = num_packet_sent_from_tx -
                             num_packet_received_from_tx;
```

```
        float percentage_of_packet_loss = ((float)num_of_packet_loss /
(float)num_packet_sent_from_tx) * 100;

        Serial.print("% Packet Lost: ");
        Serial.print(percentage_of_packet_loss);
        Serial.println();

        // print sensor data
        Serial.println("Sensor Data:");
        for (int i = 11; i <= 30; i++)
        {
            Serial.print(c_packet[i]);
        }
        Serial.println();
    }
    else {
      Serial.println("Error");
    }
}
```

END OF REPORT

THANKS

Mentor: Dr. Arpit Arvind Khandelwal

Students: Aryan Garg (B19EE015)

Akshat Agarwal (B19EE007)