CODIGO FUENTE NODO FINAL

/\*

 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Escuela Politécnica Nacional  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 \* DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DE MONITOREO DE GEOPOSICIÓN Y TEMPERATURA PARA GANADO \*

 \* Realizado por: Eduardo Chango

 \* Fecha actualización: 24/11/2021

 \*

 \* Indice de contenidos

 \*   1. Inclución de Librerias

 \*   2. Asignación de pines

 \*   3. Configuración de parámetros

 \*   4. Función de los sensores

 \*     4.1 Funcion de lectura del sensor de temperatura

 \*     4.2 Funcion de lectura del sensor de bateria

 \*     4.3 Funcion de envio LoRa

 \*     4.4 Funcion para enviar a dormir

 \*   5. Configuraciones del programa

 \*     5.1 Pin activar sensores

 \*     5.2 Serial para el GPS

 \*     5.3 Inicializar el serial para prueba

 \*     5.4 Inicializar sensor de temperatura

 \*     5.5 Configurar el modo de sueño

 \*     5.6 Configurar LoRa

 \*

 \*

 \*

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

    1. Inclusión de Librerias

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

   //Libreria para LoRa LoRa

   #include <**SPI**.h>

   #include <**LoRa**.h>

   //Libreria para sensor Temperatura

   #include <Wire.h>

   #include <Adafruit\_MLX90614.h>

   Adafruit\_MLX90614 mlx = Adafruit\_MLX90614();

   //Libreria para el GPS

   #include <**SoftwareSerial**.h>

   #include <**TinyGPS**.h>  // Por: Mikael Hart

**TinyGPS** gps;

   //Libreria para el BT

   #include <BluetoothSerial.h>

   BluetoothSerial ESP\_BT;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

    2. Asignación de Pines

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

   //Pines chip LoRa sx1276

   #define SCK 18

   #define MISO 19

   #define MOSI 23

   #define SS 13

   #define RST 14

   #define DIO0 4

   //Pin activación de sensores GPS/ MXL90614/ Sensor de batería

   #define ACTSENSOR 15

   //Pin sensor de batería

   #define SENSORBATERIA 34

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

    3. Configuración de parámetros

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

   //Parámetros LoRa

   //433E6 for Asia

   //866E6 for Europe

   //915E6 for North America

   #define BAND 915E6

   #define uS\_TO\_S\_FACTOR 1000000  //Conversion factor for micro seconds to seconds

   #define TIME\_TO\_SLEEP  120        //Time ESP32 will go to sleep (in seconds)

   //Variables de tiempo

   const int vectMin[6] = {0,10,20,30,40,50};

   const int vectSeg[6] = {60,60,60,60,60,60};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

    4. Funciones de los sensores

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

     //4.1 Funcion de lectura del sensor de temperatura

     float leerTemperatura() {

       float temperatura = mlx.readObjectTempC() + 3.0; //Compensación de 3 Grados respecto al Fluke 62 MAX

       return temperatura;

     }

     //4.2 Funcion de lectura del sensor de bateria

     float leerBateria() {

       float bateria = 0;

       for(int i=0; i<20; i++){

         int bateriaAux = (0.1768\*analogRead(SENSORBATERIA))-314.31;

         bateria=bateria+bateriaAux;

         }

       bateria=bateria\*0.05;

       return bateria;

     }

     //4.3 Funcion de envio LoRa

     void enviarInformacion(float temperatura,float bateria,float flon1,float flat1,String id){

         flat1 = flat1; //Encriptado simple para la latitud

         flon1 = flon1; //Encriptado simple para la longitud

**LoRa**.beginPacket();

**LoRa**.print("123"); //Mensaje de arranque, sin relevancia

**LoRa**.print(",");

**LoRa**.print(String(temperatura, 2)); //Envio de temperatura con dos digitos decimales

**LoRa**.print(",");

**LoRa**.print(String(bateria, 1)); //Envio de nivel de bateria con un digito decimal

**LoRa**.print(",");

**LoRa**.print(String(flon1,7)); //Envio de latitud con siete digitos decimales

**LoRa**.print(",");

**LoRa**.print(String(flat1,7));//Envio de longitud de bateria con siete digitos decimales

**LoRa**.print(",");

**LoRa**.print(id); //Envio del id del dispositivo

**LoRa**.endPacket();

     }

     //4.4 Funcion para enviar a dormir

     void veteaDormir(int segundosRestantes){

**LoRa**.sleep();  //Enviar a dormir el chip LoRa

       digitalWrite(ACTSENSOR,LOW); // Apagar el GPS/ Chip LoRa/ Sensor de temperatura

       esp\_sleep\_enable\_timer\_wakeup((segundosRestantes) \* uS\_TO\_S\_FACTOR); // Fijar el tiempo de sueño

       esp\_deep\_sleep\_start();//ESP32 entra en deep sleep

     }

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*FUNCION DE ENVIO DE DATOS SERIAL GPS

void sendUBX(byte \*MSG, uint8\_t len) {

 for(int i=0; i<len; i++) {

**Serial2**.write(MSG[i]);

 }

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

    5. Configuraciones del programa

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int LEDstate = 0;

uint8\_t contador = 0;

void setup() {

   pinMode(2,OUTPUT);

 //5.1 Pin activar sensores

   pinMode(ACTSENSOR, OUTPUT);

   digitalWrite(ACTSENSOR, HIGH);

 //5.2 Serial para el GPS

**Serial2**.begin(9600,SERIAL\_8N1,16,17);

   //byte CFG\_RXM\_fop[10] = {0xB5,0x62,0x06,0X11,0X02,0X00,0X08,0X00,0X21,0X91};

   //sendUBX(CFG\_RXM\_fop, sizeof(CFG\_RXM\_fop)/sizeof(byte));

   byte GPSon[12] = {0xB5, 0x62, 0x06, 0x04, 0x04, 0x00,

   0x00, 0x00,0x09, 0x00, 0x17, 0x76};

   sendUBX(GPSon, sizeof(GPSon)/sizeof(byte));

 //5.3 Inicializar el serial para pruebas

**Serial**.begin(115200);

 //5.4 Inicializar sensor de temperatura

   mlx.begin();

   mlx.writeEmissivity(0.96);

   //Serial.print(mlx.readEmissivity());

 //5.5 Configurar el modo de sueño

   //esp\_sleep\_enable\_timer\_wakeup(TIME\_TO\_SLEEP \* uS\_TO\_S\_FACTOR);

 //5.6 Configurar LoRa

   //SPI.begin(SCK, MISO, MOSI, SS);

   //setup LoRa transceiver module

**LoRa**.setPins(SS, RST, DIO0);

   if (!**LoRa**.begin(BAND)) {

**Serial**.println("Starting LoRa failed!");

     while (1);

   }

**LoRa**.setSyncWord(0x12);

   //LoRa.setTxPower(20);

**LoRa**.setSpreadingFactor(7);

**Serial**.print(**LoRa**.getSpreadingFactor());

**Serial**.println("LoRa Inicializacion OK!");

   delay(2000);

   //Variable de inicio

   contador = 0;

}

void loop() {

 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

                         VARIABLES LOCALES

  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

   bool newData = false;

   unsigned long chars;

   unsigned short sentences, failed;

   float flon1 = 1.0;//-78.5126792;

   float flat1 = 1.0;//-0.2743293;

   /\*

     Tambillo

     longitud= -78.537574;

     latitud= -0.407238;

     Casa

     longitud = -78.512724

     latitud = -0.274338

   \*/

   int minuto = 0, segundo = 0;

   int minutoTarget = 0;

   uint8\_t numeroSatelites = 0;

 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

                         LEER SENSORES

  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

   float temperatura = leerTemperatura();

   float bateria = leerBateria();

   /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

                         ACTUALIZAR DATOS DEL GPS

  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

 //Esperar que haya pasado un segundo antes de intentar obtener el siguiente dato GPS

   for (unsigned long start = millis(); millis() - start < 1000;)

   {

     while (**Serial2**.available())

     {

       char c = **Serial2**.read();

       //Serial.print(c); // Datos gps en crudo

       if (gps.encode(c)) //Si existe un dato válido la bandera cambia a True

         newData = true;

     }

   }

 //Si los datos del GPS son válidos obtener datos de latitud, longitud, minuto y segundo

   if (newData)

   {

     /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

             Variables locales para datos gps

     \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

     unsigned long age, time;

     int year;

     byte month, day, hour, minute, second, hundredths;

     /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

             Obtener datos de latitud y longitud

     \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

     gps.f\_get\_position(&flat1, &flon1, &age);

     flat1 == **TinyGPS**::GPS\_INVALID\_F\_ANGLE ? 0.0 : flat1, 6;

     flon1 == **TinyGPS**::GPS\_INVALID\_F\_ANGLE ? 0.0 : flon1, 6;

     /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

             Obtener el número de satelites al que esta conectado el GPS

     \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

     gps.satellites() == **TinyGPS**::GPS\_INVALID\_SATELLITES ? 0 : gps.satellites();

     numeroSatelites = gps.satellites();

     /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

             Obtener el minuto y segundo actual, para calcular el tiempo de deep sleep

     \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

     gps.crack\_datetime(&year, &month, &day, &hour, &minute, &second, &hundredths, &age);

     minuto = minute;

     segundo = second;

     int index = 0;

     while(minuto >= vectMin[index]){

       index = index + 1;

       if(index == 6){

         index = 0;

         break;

       }

     }

     //Aqui sale el index con el minuto restante calculado

     minutoTarget = vectMin[index];

     // si el vector de minutos fue seleccionado el minuto 0, entonces toma el valor de 60

     if(minutoTarget == 0){

       minutoTarget = 60;

     }

   }

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

             Envio de datos mediante LoRa

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

 if(flat1!=1.0 && flon1!=1.0){

   //si latitud y longitud son datos válidos, entonces se enviara la información mediante LoRa 3 veces

   enviarInformacion(temperatura, bateria, flon1, flat1, "2");

   if (contador<=3){

     enviarInformacion(temperatura, bateria, flon1, flat1, "2");

     delay(1000);

     contador++;

   }else{

     contador=0;

     //Cacular los segundos que el dispositivo estará en sueño profundo

     int segundosADormir = ((minutoTarget - minuto - 1)\*60) + (60 - segundo);

     //La ESP entra en sueño profundo.

     veteaDormir(segundosADormir);

   }

 }else{ //Para pruebas mientras no se tenga un dato valido el LED de la tarjeta parapadeara

   LEDstate =  1 - LEDstate;

   digitalWrite(2,LEDstate);

   enviarInformacion(temperatura, bateria,-78.489057,-0.212065, "2");

   //Si pasaron 90 segundos calcular el tiempo restante para que el dispositivo se quede apagado para ahorrar bateria

   if(millis()>=90000){

     int segundoCalculado = 60 - (millis()\*0.001);

     //Serial.println(TIME\_TO\_SLEEP + segundoCalculado);

     veteaDormir(TIME\_TO\_SLEEP + segundoCalculado);

   }

 }

}