Universidad Autónoma del Estado de México
Centro Universitario UAEM Atlacomulco

Licenciatura en Ingeniería en Computación

Manual de Prácticas de la Unidad de Aprendizaje de Sistemas Embebidos

Octavo Semestre

Laboratorio de Electrónica

Elaboró: Dr. en C.I.E. Allan Antonio Flores Fuentes

Fecha: 20 de enero de 2025

	H. Consejo Académico	H. Consejo de Gobierno
Fecha de Aprobación:	**10-07-2025**	**10-07-2025**

ÍNDICE DE CONTENIDOS

PRESENTACIÓN

A lo largo de la línea de unidades de aprendizaje del área de arquitectura de computadoras dentro del programa educativo de Ingeniería en Computación, se ha abordado el estudio de los sistemas analógicos y digitales cada uno en su contexto de usos y aplicaciones. Sin embargo, uno de los campos en donde pueden divergir ambas disciplinas, es el de los microcontroladores. El estudio de estos permitirá al estudiante entender que no solo se pueden programar sistemas basados en microprocesadores que soportan aplicaciones de alto nivel orientadas en su mayoría al software, sino que también existen otros dispositivos electrónicos en los cuales se pueden aplicar los conceptos y paradigmas de programación en ensamblador o en alto nivel, para aplicaciones específicas de software y hardware, tales como control de procesos, adquisición de datos y señales o aplicaciones en tiempo real. La Unidad de Aprendizaje Sistemas Embebidos consta de cuatro unidades temáticas, cuyo nivel de complejidad irá incrementándose y además será acumulativo. En la unidad uno se introduce al estudiante en el contexto del estado del arte de los microcontroladores desde una panorámica general. En la unidad dos se definen y describen conceptos generales respecto a la arquitectura interna que todo microcontrolador tiene y debe conocerse, para ello se sugiere que el docente defina qué microcontrolador se usará en el curso, considerando la disponibilidad en el mercado y las aplicaciones que se realicen en las prácticas. Dado que cada fabricante de microcontroladores tiene sus propias herramientas de programación, llámese lenguaje de programación e interfaz de usuario (IDE), la unidad tres aborda los pormenores de las reglas sintácticas del lenguaje en el cual se trabajará a lo largo del curso en función del microcontrolador seleccionado, así como el aprendizaje del entorno de programación y las herramientas disponibles para ello. Finalmente, en la unidad cuatro se integran todos los conocimientos para manipular en su totalidad todos los recursos con los que cuenta el microcontrolador que se eligió en la unidad dos y poder realizar la síntesis del conocimiento para dar solución a diversos problemas y aplicaciones de control de procesos y adquisición de datos y señales. Con los conocimientos adquiridos en esta UA, el alumno será capaz, no solo de utilizar la plataforma del microcontrolador elegido para el curso (sea sistema mínimo o embebido) sino también aventurarse al aprendizaje de nuevas plataformas basadas en microcontroladores, teniendo como base la experiencia adquirida a lo largo del curso y la capacidad de elegir la plataforma y recursos necesarios en el desarrollo de sistemas embebidos acorde con las necesidades específicas de solución de proyectos.

Este manual integra prácticas de laboratorio que cubren al menos los temas principales de cada Unidad con el objetivo de poner en práctica las habilidades del alumno. Además, integra los procesos para que cada alumno pueda repetir las veces necesarias las prácticas, con los recursos necesarios.

Centro Universitario UAEM Atlacomulco
Km. 60 Carretera Toluca – Atlacomulco, Col. San Francisco Chalchihuapan C.P. 50450, Atlacomulco, Estado de México.
Tels. (712) 12204 36, 1220446, 1220535 **e-mail:** cuatla@uaemex.mx **sitio web:** https://cuatlacomulco.uaemex.mx **Facebook:** /cua.uaemex.atlacomulco

Evaluación de prácticas.

1. Entrega de práctica funcional 40 % del total de la práctica.
2. Entrega de informe de práctica 60 %, bajo los requerimientos particulares de cada una de estas.

El informe de práctica debe contener las secciones siguientes ponderadas de la siguiente manera:

Sección	Porcentaje %
Resumen con *Abstract*	10
Estudio de la literatura (Introducción)	10
Método	10
Resultados presentados	10
Conclusión	10
Referencias en formato APA	10
Total	**60**

Práctica 1

Unidad de Competencia 1. Una revisión a los microcontroladores y sistemas embebidos.	
Tema:	4.2. Descripción general de sistema embebido.
Contenidos:	4.2.1. Arquitecturas existentes.
Duración:	1 hora con 40 minutos.
Valor de la Práctica:	10 puntos
Trabajo:	Individual

Los microcontroladores en los sistemas embebidos

Introducción

Los sistemas embebidos NO son una computadora personal, servidor Workstation, supercomputadora, clúster distribuido, ni mucho menos un sistema programable de propósito general. En tanto, SI son sistemas diseñados para cumplir limitadas funciones dedicadas y está empotrado o "embebido" en un dispositivo de hardware completo, por ejemplo; en la industria automotriz en un navegador GPS (*Global Position System*, por sus siglas en inglés), en la aviación como un piloto automático, en telecomunicaciones como rúter y módems, otro otros. Estos sistemas utilizan software y hardware específico, donde este último puede ser a base de tecnología, tal como; microcontroladores, procesadores digitales de señales, arreglos de compuertas programables, circuitos integrados lineales digitales y analógicos. El objetivo de los sistemas embebidos es procesar información específica a través de sistemas de uso específico.

Objetivo(s):

Desarrollar una matriz de referencias que contenga información respecto a sistemas embebidos con algún tipo de hardware y sus aplicaciones, además de aportaciones.

Material y Equipo para utilizar.

Plataformas libres, VosViewer, Connected Papers, Schoolar Google.

Metodología:

1. Realizar una búsqueda de literatura en Schoolar Google con al menos cinco años de antigüedad.
2. Elegir una literatura específica, la cual deba ser de acceso libre por algún tipo de repositorio, se recomienda en este orden, artículos de investigación, capítulos de libro, libros, tesis, y reportes

técnicos. Sugerencia, puede utilizar la biblioteca de la UAEMex: Biblioteca Digital - Home (uaemex.mx)

3. Realizar un mapa de referencia con la herramienta de uso libre *Connected papers*: Connected Papers | Find and explore academic papers
4. Realizar un mapa de conceptos con la versión gratuita de *VosViewer*.
5. Presentar informe documentado.

Resultados de la práctica.

El informe documentado debe tener las siguientes secciones:

1. Portada
2. Resumen con Abstract, máximo 150 palabras.
3. Introducción, máximo 1000 palabras.
4. Desarrollo del tema central con enlace de documento libre (*Open Access*) descargable*
5. Mapa de *Connected Papers**
6. Mapa conceptual *VosViewer**
7. Matriz de referencias*
8. Conclusión, máximo 250 palabras.
9. Referencias, mínimo diez referencias.

Actividad de Integración

Estas actividades deben incluirse en el reporte de práctica para evaluación (*).

Material complementario para el desarrollo de la práctica.

1. Repositorio Institucional UAEMex, consultado de RI UAEMex
2. Navegador Schoolar Google.
3. Aplicación en línea de Connected papers, consultado de Connected Papers | Find and explore academic papers
4. *VOSViewer: Download*. (s. f.). VOSviewer. Sitio de consulta, VOSviewer :: Download

Práctica 2

Unidad de Competencia 2. Arquitectura de un microcontrolador	
Tema:	4.2. Arquitectura del microcontrolador ATMEGA.
Contenidos:	4.2.1. Especificaciones y configuración
Duración:	1 hora con 40 minutos.
Valor de la Práctica:	10 puntos
Trabajo:	Colaborativo

Arquitectura de un microcontrolador

Introducción

El microcontrolador ATMEGA es un circuito integrado CMOS (Semiconductor complementario de óxido metálico, por su traducción al español) de 8 bits de bajo consumo basado en la arquitectura RISC (computación de conjunto de instrucciones reducido, por su traducción al español) mejorada AVR. Al ejecutar potentes instrucciones en un solo ciclo de reloj, el ATmega48P/88P/168P/328P logra rendimientos cercanos a 1 MIPS (Mega instrucciones por segundo) por MHz, lo que permite al diseñador del sistema optimizar el consumo de energía frente a la velocidad de procesamiento. La arquitectura se puede detallar en la hoja de datos (Atmel, 2015). Este microcontrolador procesa la información en sistemas de una sola placa (SBC, por su acrónimo al inglés) como Arduino, en los modelos, UNO, Micro y Leonardo, principalmente (Arduino, 2024). Actualmente está SBC, es una de las más implementadas en sistemas embebidos, y mayormente para aplicaciones en internet de las cosas. En esta práctica el alumno conocerá aplicaciones con Arduino, en alguna de sus categorías, para la configuración de sensores, tales como, temperatura, presión, humedad, entre los más comunes.

Objetivo(s):

Configurar sensores de presión, temperatura, y humedad a una SBC Arduino para conocer aplicaciones reales de un microcontrolador para aplicaciones específicas.

Material y Equipo para utilizar.

Placa de Arduino Uno®, placa con sensor de temperatura, presión y humedad, *protoboard*. Equipo de laboratorio: Fuente de voltaje, y osciloscopio. Software el que la SBC requiera, el cual se puede descargar en la página de Arduino oficial (Arduino, 2024)

Metodología:

1. Configure Arduino en cualquiera de sus versiones a través de las librerías específicas para el sensor BMP280. Lea la hoja de datos, primeramente, la cual está en el siguiente recurso: BMP280 Datasheet by TinyCircuits | Digi-Key Electronics (digikey.com.mx)

2. A continuación, realice el circuito de la Figura 1.

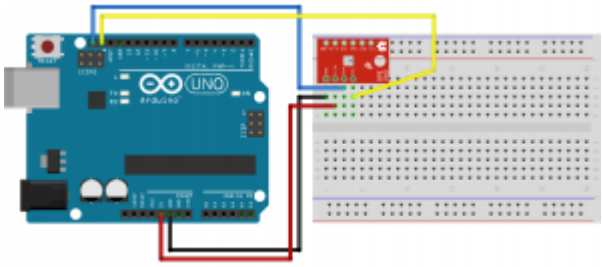

Figura 1. Circuito de conexión.
Fuente: Imagen propia elaborada con software libre

3. Descargar e instalar las librerías de uso libre Adafruit para el sensor BME280. El procedimiento consiste en ir al menú Programa\Incluir librería\administrar Bibliotecas, de la IDE de Arduino.

Una vez descarga e instalada se debe probar, a partir del siguiente código:

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
```

La primera línea es para la comunicación I2C y la segunda llama a una librería de sensor unificada precisa para casi cualquier librería Adafruit. La tercera línea invoca la librería del BME280 que acabamos de instalar.

4. Las lecturas de presión, temperatura y humedad son, absolutas con respecto a una referencia interna. La altitud se calcula por estimación a partir de la presión atmosférica ya que es una variable dependiente de factores como la humedad ambiente y otras mandangas *(Termino científico para designar imponderables varios)*, por lo tanto, es necesario calibrar la altitud con una referencia de la presión actual a nivel de mar, a partir de la siguiente línea:

```
#define SEALEVELPRESSURE_HPA (1015)
```

5. Se define la presión a cota cero (*Nivel del mar*) medida en hectopascales, como 1015.
6. La siguiente línea crea una instancia del sensor y la llamamos BM (por ejemplo)


```
Adafruit_BME280 bme;
```

En el setup()

```
void setup()
  {  Serial.begin(115200);
     if (!bme.begin(0x76))
      {  Serial.println("¡No encuentro un sensor BME280 valido!");
         while (1);
      }
  }
```

Se habilita el puerto serie e inicializa el sensor con bme.beginv( ) para enviar la dirección a través del I2C del sensor. Si no se encuentra el sensor el código permite salir del bucle. Caso contrario, se habilita el sensor, y es posible capturar valores en la consola:

```
void loop()
{   Serial.print("Temperatura = ");
    Serial.print(bme.readTemperature());
    Serial.println("*C");
    Serial.print("Presion = ");
    Serial.print(bme.readPressure() / 100.0F);
    Serial.println("hPa");

    Serial.print("Altitud estimada = ");
    Serial.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
    Serial.println("m");
```

```
    Serial.print("Humedad = ");
    Serial.print(bme.readHumidity());
    Serial.println("%");
    Serial.println();
    delay(1000);
}
```


Programa disponible de manera gratuita en: BME280_y_Arduino_UNO

7. Realice lecturas cada 0.5 segundos, y capture durante al menos 5 minutos estas.
8. Descargue los datos y grafíquelos en Excel libre.
9. Aplique un filtro medio móvil para ver el efecto de las mediciones.
10. Documente los resultados.

Resultados de la práctica.

El informe documentado debe tener las siguientes secciones:

1. Portada
2. Resumen con Abstract
3. Introducción
4. Desarrollo del tema central con enlace de documento libre descargable
5. Diagramas*
6. Configuración del dispositivo*
7. Resultados graficados en Excel*
8. Conclusión
9. Referencias

Actividad de Integración

Estas actividades deben incluirse en el reporte de práctica para evaluación (*).

Material complementario para el desarrollo de la práctica:

1. Datasheet ATMEGA328P, ATMEL 2015, recuperado de: ATmega328P (microchip.com)
2. Arduino 2024, Documentación recuperada de: Arduino Docs | Arduino Documentation
3. Arduino 2024, Documentación recuperada de: Creating CSV File - Using Arduino / Storage - Arduino Forum

Práctica 3

Unidad de Competencia 3. Protocolos de comunicación	
Tema:	4.2. Descripción general de los protocolos de comunicación.
Contenidos:	4.2.1. Protocolo de comunicación con unidad de posicionamiento GPS
Duración:	1 hora con 40 minutos.
Valor de la Práctica:	10 puntos
Trabajo:	Colaborativo

Configuración de puertos con microcontrolador ATMEGA

Introducción

Los protocolos de comunicación disponibles para las plataformas Arduino son principalmente a través de los siguientes periféricos: a) 2x temporizador/contador de 8 bits con un registro de período dedicado y canales de comparación, b) 1x temporizador/contador de 16 bits con un registro de período dedicado, captura de entrada y canales de comparación, c) 1x USART con generador de velocidad de baudios fraccional y arranque Detección de fotograma, d) 1 controlador/periférico Interfaz periférica serie (SPI), e) 1 controlador de modo dual/periférico I2C, f) 1 comparador analógico (AC) con una entrada de referencia escalable, g) Temporizador de vigilancia con oscilador en chip separado, h) Seis canales PWM (Modulación por Ancho de Pulso, por su traducción al español) (Arduino, 2015). A través del módulo GPS 15733 del fabricante Sparfun (Sparkfun, 2019). El módulo GPS GY-GPS6MV2 está basado en el SoM U-Blox NEO-6M equipado en el PCB, una EEPROM con configuración de fábrica, una pila de botón para mantener los datos de configuración en la memoria EEPROM, un indicador LED y una antena cerámica. También posee los pines o conectores VCC, RX, TX y GND por el que se puede conectar a algún microcontrolador mediante una interfaz serial. Para que nuestro módulo GPS funcione a la perfección se recomienda hacer las pruebas en un ambiente abierto o cercano a una ventana para una correcta recepción de la señal de los satélites.

Objetivo(s):

Configurar el módulo GPS-15733 para llevar a cabo la comunicación entre los datos registrado y la SBC Arduino mediante alguno de los protocolos de comunicación: NMEA, UBX y RTCM a través de interfaces UART o I2C.

Material y Equipo para utilizar.

Placa de Arduino Uno®, módulo GPS 15733. Equipo de laboratorio: Fuente de voltaje, y osciloscopio. Software el que la SBC requiera, el cual se puede descargar en la página de Arduino oficial (Arduino, 2024)

Metodología:

1. Configure Arduino en cualquiera de sus versiones a través de las librerías específicas para el módulo GPS 15733. Utilice la siguiente información:
2. La figura 2 muestra la conexión del GPS 15733 para su configuración. Se debe cargar el siguiente código para su configuración:

```
#include <SoftwareSerial.h>
SoftwareSerial gps(4,3);
char dato=' ';
void setup()
{
 Serial.begin(115200);
 gps.begin(9600);
}

void loop()
{
  if(gps.available())
  {
    dato=gps.read();
    Serial.print(dato);
  }
}
```

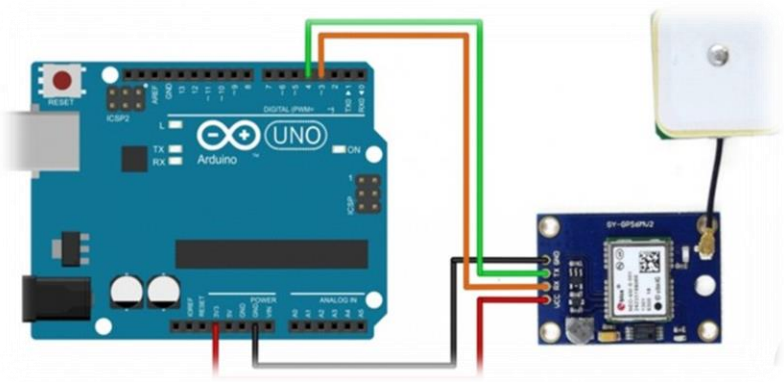

Figura 2. Diagrama de conexión del GPS 15733.
Fuente: Imagen propia elaborada con software libre

Los datos recibidos en el módulo GPS siguen el protocolo NMEA (siglas de National Marine Electronics Asociation), las cuales son sentencias estándares para la recepción de datos GPS. Una de ellas y la más usada son las sentencias $GPRMC, las cuales tienen la siguiente estructura:

$GPRMC,044235.000,A,4322.0289,N,00824.5210,W,0.39,65.46,020615,A*44

Tomando como base el protocolo NMEA, podríamos determinar las siguientes variables:

a. **044235.000**: Representa la hora GMT (04:42:35)
b. **"A"**: es la indicación de que el dato de posición está fijado y es correcto. "V" sería no válido
c. **4322.0289:** representa la longitud (43º 22.0289´)
d. **N**: representa el Norte
e. **00824.5210**: representa la latitud (8º 24.5210´)
f. **W**: representa el Oeste
g. **0.39:** representa la velocidad en nudos
h. **65.46**: representa la orientación en grados
i. **020615**: representa la fecha (2 de junio del 2015)

A partir de la trama de datos que envía el módulo GPS se obtienen diferentes variables, entre las más importantes son la latitud y la longitud. Entonces, es necesario la librería TinyGPS disponible de manera gratuita en:

GitHub - mikalhart/TinyGPS: A compact Arduino NMEA (GPS) parsing library

Toda vez descargada la librería, se requiere importarla copiándola en la carpeta "Libraries" donde se instaló el IDE de Arduino, después reiniciar el programa para que sea cargada correctamente. La librería TinyGPS facilitará la identificación tanto de la latitud y longitud, así como las otras variables. Por ejemplo, es posible dirigirse a ***Archivo/Ejemplos/TinyGPS/simple_test*** en nuestro IDE de Arduino.

Finalmente, es posible abrir el monitor serial y visualizar los datos.

1. Realice lecturas cada 0.5 segundos, y capture durante al menos 5 minutos estas.
2. Descargue los datos y grafíquelos en Excel.
3. Aplique un filtro medio móvil para ver el efecto de las mediciones.
4. Documente los resultados.

Resultados de la práctica.

El informe documentado debe tener las siguientes secciones:

1. Portada
2. Resumen con Abstract
3. Introducción
4. Desarrollo del tema central con enlace de documento libre descargable
5. Desarrollo de diagrama de flujo*
6. Obtención de valores de manera correcta y su grafica*
7. Interpretar resultados por medio de una breve discusión en 150 palabras*
8. Conclusión
9. Referencias

Actividad de Integración

Estas actividades deben incluirse en el reporte de práctica para evaluación (*).

Material complementario para el desarrollo de la práctica:

1. Sparkfun, 2017, Datasheet, recuperada de: https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/211/GPS-15733_Web.pdf

Práctica 4

Unidad de Competencia 3. Interfaces con circuitos analógicos y digitales	
Tema:	4.2. Descripción general de las interfaces con circuitos analógicos y digitales
Contenidos:	4.2.1. Aplicaciones integradoras
Duración:	1 hora con 40 minutos.
Valor de la Práctica:	10 puntos
Trabajo:	Colaborativo

Interfaces con aplicaciones integradoras

Introducción

Los sistemas embebidos tienen una función específica, por lo tanto, son diseñados bajo requerimientos específicos. Para su diseño se requieren conocimientos que integren los sistemas, no sólo en cuestión de sistemas digitales, sino también analógicos. En este sentido, el alumno debe integrar las capacidades a través del desarrollo de un sistema embebido con aplicación específica. En este contexto, los Cansat son microsatélites que tienen el objetivo de integrar conocimientos académicos para desarrollo específico de tareas o mejor conocidas como misiones. Uno de los parámetros físicos tipo para medir es la posición, aceleración y movimiento en los ejes, x, y y z. Lo último se logra a través de un sensor de giro en 3 ejes, o a veces nombrado magnetómetro. El AK8963 es un sensor de 3 ejes a base del efecto hall, además Incorpora sensores magnéticos para la detección del magnetismo terrestre en los ejes X, Y y Z. En esta práctica el alumno no solo configurará el módulo a base de este sensor, sino también, llevará a cabo una fase de calibración para su mejor desempeño.

Objetivo(s):

Configurar el módulo AK8963 para calibrar la información y lograr un máximo desempeño de las lecturas del sensor de 3 ejes.

Material y Equipo para utilizar.

Placa de Arduino Uno®, módulo AK8963. Equipo de laboratorio: Fuente de voltaje, y osciloscopio. Software el que la SBC requiera, el cual se puede descargar en la página de Arduino oficial (AKM, 2012)

Metodología:

1. Configure Arduino en cualquiera de sus versiones a través de las librerías específicas para el módulo AK8963.

2. Descargue la hoja de datos y lea con atención la configuración del protocolo: RS-MPU-6000A-00 (strawberry-linux.com)
3. El Código de configuración lo puedes encontrar en la página oficial de Arduino a través del siguiente enlace:

Not getting data from AK8963 magnetometer in mpu9250 - Community / General Discussion - Arduino Forum

4. Realice lecturas cada 0.5 segundos, y capture durante al menos 5 minutos estas.
5. Calibre el AK8963. Utilice el siguiente recurso: https://ijece.iaescore.com/index.php/IJECE/article/view/28308/16519

Resultados de la práctica.

El informe documentado debe tener las siguientes secciones:

1. Portada
2. Resumen con Abstract
3. Introducción
4. Desarrollo del tema central con enlace de documento libre descargable
5. Diagramas de funcionamiento, a bloques o de requerimientos*
6. Captura y descarga de lecturas para su grafica*
7. Discusión de los resultados*
8. Conclusión
9. Referencias

Actividad de Integración

Estas actividades deben incluirse en el reporte de práctica para evaluación (*).

Material complementario para el desarrollo de la práctica:

1. AKM, 2012, datasheet, recuperada de: https://pdf1.alldatasheet.com/datasheet-pdf/view/535561/AKM/AK8963.html

Práctica 5

Unidad de Competencia 4. Interfaces con circuitos analógicos y digitales	
Tema:	4.2. Descripción general de las interfaces con circuitos analógicos y digitales
Contenidos:	4.2.1. Aplicaciones integradoras
Duración:	1 hora con 40 minutos.
Valor de la Práctica:	10 puntos
Trabajo:	Colaborativo

Módulo LoRa RYLR998

Introducción

En el ámbito de las comunicaciones inalámbricas, la tecnología LoRa (Long Range) ha ganado una creciente popularidad debido a su capacidad para establecer enlaces de bajo consumo energético y larga distancia. Uno de los módulos más populares que implementa esta tecnología es el RYLR998, el cual se puede integrar fácilmente con plataformas populares como Arduino. El presente reporte tiene como objetivo detallar el proceso de configuración y puesta en marcha de un sistema de comunicación inalámbrica utilizando el módulo LoRa RYLR998 en conjunto con una placa Arduino Uno®. Se implementará un transmisor y un receptor con el fin de evaluar el desempeño del enlace inalámbrico, midiendo parámetros clave como el tiempo de encendido, tiempo de apagado, duración del pulso transmitido y la frecuencia de ruido presente en el canal. La configuración del transmisor y receptor LoRa utilizando Arduino Uno®permitirá comprender los conceptos fundamentales de esta tecnología y su implementación práctica. Además, el análisis de los parámetros medidos brindará una visión clara del comportamiento del enlace inalámbrico y su capacidad para transmitir información de manera eficiente y confiable. Los resultados y experiencias obtenidas durante el proceso de configuración e implementación servirán como un valioso recurso para futuras aplicaciones que requieran enlaces de bajo consumo energético y larga distancia, abriendo camino a nuevas posibilidades en áreas como el monitoreo remoto, la automatización industrial y otras áreas emergentes.

Objetivo(s):

Configurar el módulo LoRa RYLR998 para la transmisión y recepción de datos al momento de establecer una comunicación entre dos módulos.

Material y Equipo para utilizar.

Dos placas de Arduino Uno®, 2 módulos LoRa RYLR998, jumpers, push button, led.

Metodología:

1. Para el ensamble del circuito transmisor. El pin TXD del módulo LoRa se conecta al pin RX del Arduino, el pin RXD del módulo LoRa se conecta al pin TX del Arduino la alimentación del LoRa se conectó a la salida de 3.3v del Arduino y el GND del módulo LoRa se conecta al GND del Arduino, el push button se conectó al pin 9 del Arduino y al GND. A continuación, se muestra el esquema de conexión del transmisor.

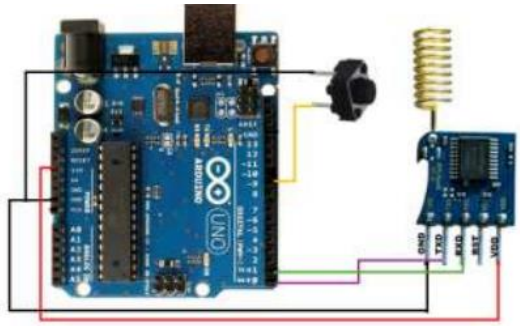

Figura 3. Esquema del circuito de transmisión mediante LoRa
Fuente: Imagen propia elaborada con software libre

2. Para el ensamble del circuito del receptor: El pin TXD del módulo LoRa se conecta al pin RX del Arduino, el pin RXD del módulo LoRa se conects al pin TX del Arduino la alimentación del LoRa se conects a la salida de 3.3v del Arduino y el GND del módulo LoRa se conecta al GND del Arduino, el led se conecta al pin 9 del Arduino y al GND. A continuación, se muestra el esquema de conexión del transmisor.
3. Código: Para establecer comunicación inalámbrica entre los módulos LoRa, se elaboraron dos códigos, uno de ellos corresponde al transmisor y el otro al receptor, en cada uno de ellos se lleva a cabo la configuración mediante comandos AT.
4. Código transmisor: El siguiente código este consiste en configurar un dispositivo LoRa para operar en la frecuencia de América del Norte y establecer ciertos parámetros de red. Luego, en el bucle principal, se monitorea un botón y, cuando se presiona, envía un mensaje a la parte receptora del sistema.

```
String lora_band = "915000000"; //enter band
as per your country
String lora_networkid = "5";    //enter Lora
Network ID
String lora_address = "1";      //enter Lora
address
String lora_RX_address = "2";   //enter Lora
RX address

#define button 9

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

  delay(1500);
  Serial.println("AT+BAND=" + lora_band);
  delay(500);
  Serial.println("AT+ADDRESS=" +
lora_address);
  delay(500);
  Serial.println("AT+NETWORKID=" +
lora_networkid);
  delay(1500);

  pinMode(button, INPUT_PULLUP);
}

void loop()
{
   if (digitalRead(button) == LOW){
  Serial.println("AT+SEND="+ lora_RX_address
+",2,FB");
  delay(50);
   }
}
```

Figura 4. Código para configurar la transmisión mediante LoRa.
Fuente: Imagen elaborada con software libre de la IDE de Arduino.

5. Código receptor: En este se configura y controla la comunicación LoRa, cambiando el estado de un led en respuesta al mensaje recibido por parte del botón que se encuentra en el transmisor.

Centro Universitario UAEM Atlacomulco
Km. 60 Carretera Toluca – Atlacomulco, Col. San Francisco Chalchihuapan C.P. 50450, Atlacomulco, Estado de México.
Tels. (712) 12204 36, 1220446, 1220535 **e-mail:** cuatla@uaemex.mx **sitio web:** https://cuatlacomulco.uaemex.mx **Facebook:** /cua.uaemex.atlacomulco

```
String lora_band = "915000000"; //enter band
as per your country
String lora_networkid = "5";    //enter Lora
Network ID
String lora_address = "2";      //enter Lora
address
String lora_RX_address = "1";   //enter Lora
RX address (for sending)

#define RelayPin 9
String incomingString;
void setup(){
  pinMode(RelayPin, OUTPUT);
  Serial.begin(9600);
  delay(1500);
  Serial.println("AT+BAND=" + lora_band);
  delay(500);
  Serial.println("AT+NETWORKID=" +
lora_networkid);
  delay(500);
  Serial.println("AT+ADDRESS=" +
lora_address);
  delay(1000);
}

void loop(){
    if(Serial.available()) {
    incomingString = Serial.readString();
    if(incomingString.indexOf("FB") >0) {
        digitalWrite(RelayPin,
!digitalRead(RelayPin));
        delay(1000);
        digitalWrite(RelayPin,LOW);
    }
    }
}
```

Figura 5. Código para configurar la recepción mediante LoRa.
Fuente: Imagen elaborada con software libre de la IDE de Arduino.

Resultados de la práctica.

El informe documentado debe tener las siguientes secciones:

1. Portada
2. Resumen con Abstract
3. Introducción
4. Desarrollo de la práctica
5. Diseño de los circuitos en software libre*
6. Configuración del código (receptor y emisor) *

7. Resultados
8. Conclusión
9. Referencias

Actividad de Integración

Estas actividades deben incluirse en el reporte de práctica para evaluación (*).

Material complementario para el desarrollo de la práctica:

1. Gonzalez, C. (2023, 26 junio). *Tutorial de cómo iniciar una comunicación de largo alcance (LoRa) con el módulo LoRa RYLR998*. Anatronic S.A. Accedido el 14 de marzo de 2024. [En línea]. Disponible: Tutorial de cómo iniciar una comunicación de largo alcance (LoRa) con el módulo LoRa RYLR998
2. P. Pickering (2017) "Desarrollar con LoRa para aplicaciones IoT de baja tasa y largo alcance". DigiKey. Accedido el 14 de marzo de 2024. [En línea]. Disponible: Descripción general de la plataforma LoRa | DigiKey
3. RYLR998 Datasheet. REYAX: IoT Solution Provider | Modules, Wireless Solution & IoT Connectivity. Accedido el 18 de marzo de 2024. [En línea]. Disponible: https://reyax.com//upload/products_download/download_file/RYLR998_EN.pdf

Práctica 6

Unidad de Competencia 2. Interfaces con circuitos analógicos y digitales	
Tema:	4.2. Descripción general de las interfaces con circuitos analógicos y digitales
Contenidos:	4.2.1. Aplicaciones integradoras
Duración:	1 hora con 40 minutos.
Valor de la Práctica:	1 puntos
Trabajo:	Colaborativo

Configuración y prueba del sensor BMP280 en plataforma IO IDE

Introducción

El campo de la electrónica y sistemas embebidos, se utilizan una amplia variedad de sensores para medir y recopilar datos del entorno. Uno de estos sensores es el BMP280, que es ampliamente utilizado para medir la presión atmosférica y la temperatura en proyectos de Arduino y otros dispositivos. El BMP280 es un sensor digital de precisión diseñado por Bosch Sensortec. Incorpora un barómetro de precisión que permite medir la altura respecto al nivel del mar, su funcionamiento está basado en la relación entre presión del aire y la altitud.

El BMP280 ofrece varias ventajas clave, como su alta precisión, bajo consumo de energía, tamaño compacto y fácil integración a través del protocolo I2C. La comunicación I2C simplifica la conexión y el intercambio de datos entre el microcontrolador y el sensor, utilizando solo dos cables. Sin embargo, también presenta algunas limitaciones, como la necesidad de calibración periódica para mantener la precisión y la posible vulnerabilidad a interferencias electromagnéticas en entornos ruidosos. Al utilizar el sensor BMP280 con comunicación I2C, es posible obtener información valiosa sobre las condiciones ambientales, lo que permite tomar decisiones informadas en diferentes aplicaciones. Por ejemplo, en una estación meteorológica, el sensor BMP280 puede proporcionar datos precisos sobre la presión atmosférica, lo que permite predecir cambios en el clima.

En sistemas de navegación, el sensor puede ayudar a determinar la altitud y la ubicación con mayor precisión. Para su configuración se hace uso de la PlatformIO IDE que es un entorno de desarrollo integrado (IDE) multiplataforma para programar sistemas embebidos. Diseñado para trabajar con microcontroladores y placas de desarrollo populares el sensor BMP280 es ampliamente utilizado debido a su capacidad para medir con precisión la presión atmosférica y la temperatura durante el vuelo. Estos datos son cruciales para determinar la altitud y el comportamiento en diferentes etapas de la trayectoria en la que se ubican los sensores.

Objetivo(s):

Configurar el sensor BMP280 para calibrar la información a través de PlatformIO IDE.

Material y Equipo para utilizar.

Placa de Arduino Uno®, Placa con sensor de temperatura, presión y humedad, protoboard. Equipo de laboratorio: Fuente de voltaje, y osciloscopio. IDE PlatformIO

Metodología:

Esquema de conexión: El BMP280 es un sensor digital de presión barométrica y temperatura fabricado por Bosch. Este sensor utiliza una interfaz I2C o SPI para comunicarse con un microcontrolador o sistema anfitrión. A continuación, explico la función de cada uno de sus pines:

1. VCC: Este pin se conecta a la fuente de alimentación del sensor, normalmente entre 1.7V y 3.6V.
2. GND: Este pin se conecta a tierra o a la referencia de voltaje cero.
3. SCL: Este pin se utiliza para la señal de reloj en el modo de comunicación I2C.
4. SDA: Transmisión de datos. Cuando el dispositivo maestro quiere enviar datos al sensor BMP280, utiliza el pin SDA para transmitir los datos bit por bit.

El pin VIN se conecta a la salida de voltaje de 5 volts del Arduino, el pin GND se conecta a la tierra del Arduino, el pin SCL se conecta al pin A4 del Arduino y el pin SDA se conecta al pin A5 del Arduino.

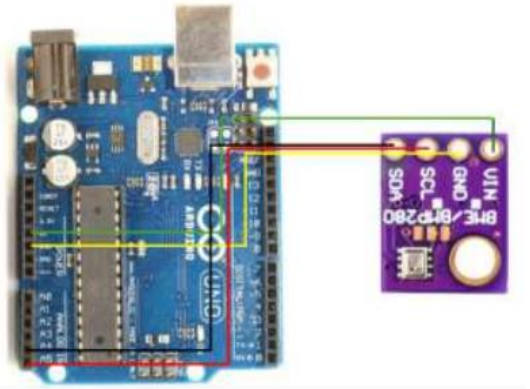

Figura 6. Conexión de Arduino Uno®con el sensor BMP280.
Fuente: Imagen elaborada con software libre.

Para llevar a cabo la prueba del sensor BMP280 dentro de la plataforma PlatformIO se elaboró el siguiente código, el cual inicializa el sensor BME280, lee sus valores de temperatura, presión, altitud y humedad, y los imprime a través de la comunicación serial de forma continua.

```
#include <Arduino.h>
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>

#define SEALEVELPRESSURE_HPA (1013.25)

Adafruit_BME280 bme; // I2C

unsigned long delayTime;
void printValues();

void setup() {
  Serial.begin(9600);
  Serial.println(F("BME280 test"));

  bool status;
  status = bme.begin(0x76);
  if (!status) {
    Serial.println("Could not find a valid BME280 sensor,
check wiring!");
    while (1);
  }
  Serial.println("-- Default Test --");
  delayTime = 1000;
  Serial.println();
}

void loop() {
  printValues();
  delay(delayTime);
}

void printValues() {
  Serial.print("Temperature = ");
  Serial.print(bme.readTemperature());
  Serial.println(" *C");

  Serial.print("Pressure = ");
  Serial.print(bme.readPressure() / 100.0F);
  Serial.println(" hPa");

  Serial.print("Approx. Altitude = ");
  Serial.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
  Serial.println(" m");

  Serial.print("Humidity = ");
  Serial.print(bme.readHumidity());
  Serial.println(" %");
  Serial.println();
}
```

Figura 7. Código para configurar el sensor mediante la plataforma PlatformIO

Fuente: Imagen elaborada con software libre PlatformIO

Resultados de la práctica.

El informe documentado debe tener las siguientes secciones:

1. Portada
2. Resumen con Abstract
3. Introducción
4. Desarrollo de practica
5. Diseño del circuito en diagrama electrónico*
6. Implementación de PlataformIO*
7. Resultados
8. Conclusión
9. Referencias

Actividad de Integración

Estas actividades deben incluirse en el reporte de práctica para evaluación (*).

Material complementario para el desarrollo de la práctica:

1. *Sensor de presión BMP280*. (s.f.). Naylamp Mechatronics - Perú. Accedido el 3 de marzo de 2024. [En línea]. Disponible: https://naylampmechatronics.com/sensoresposicioninerciales-gps/358-sensor-depresion-bmp280.html
2. Santos, S., & Santos, S. (2020, 30 julio). Guide for BME280 Sensor with Arduino (Pressure, Temperature, Humidity). Random Nerd Tutorials. Random Nerd Tutorials. Accedido el 4 de marzo de 2024. [En línea]. Disponible: Guide for BME280 Sensor with Arduino (Pressure, Temperature, Humidity) | Random Nerd Tutorials
3. *What is PlatformIO? — PlatformIO v6.1 documentation*. (s.f.). Accedido el 20 de marzo de 2024. [En línea]. Disponible: What is PlatformIO? — PlatformIO v6.1 documentation

Práctica 7

Unidad de Competencia 2. Interfaces con circuitos analógicos y digitales	
Tema:	4.2. Descripción general de las interfaces con circuitos analógicos y digitales
Contenidos:	4.2.1. Aplicaciones integradoras
Duración:	1 hora con 40 minutos.
Valor de la Práctica:	1 puntos
Trabajo:	Colaborativo

Sistema de localización embebido con módulo NEO-6M

Introducción

La capacidad de obtener información precisa sobre la ubicación geográfica en tiempo real es fundamental para una amplia gama de aplicaciones, desde sistemas de navegación hasta rastreo de activos y recopilación de datos. El módulo GPS NEO-6M, es un receptor de posicionamiento global de alta precisión y bajo consumo de energía que permite a los dispositivos embebidos determinar su ubicación gracias a su tecnología basada en el chip NEO-6M de Ublox. Esta práctica tiene como objetivo familiarizarse con la configuración y el uso del módulo GPS NEO-6M en el entorno de desarrollo integrado (IDE) PlatformIO para obtener y mostrar la posición GPS utilizando el microcontrolador Arduino y la biblioteca TinyGPS para recibir y procesar los datos del módulo. El módulo GPS NEO-6M utiliza la tecnología de Sistema de Posicionamiento Global (GPS) para calcular su ubicación geográfica. Este sistema consta de una red de satélites en órbita alrededor de la Tierra que transmiten señales de radio. Estas señales son procesadas por el módulo y, mediante el proceso de trilateración, puede determinar su latitud, longitud y altitud con una alta precisión.

La comunicación entre el módulo GPS NEO-6M y el microcontrolador se realiza mediante una interfaz serie UART (Universal Asynchronous Receiver/Transmitter), utilizando dos líneas de comunicación: Tx para transmitir los datos y Rx para recibirlos. El tipo de comunicación que realiza es de serie asíncrona, esto significa que los datos se transmiten de forma secuencial, sin una señal de reloj compartida entre el módulo y el microcontrolador. Sin embargo, se utilizan bits de inicio y de parada para sincronizar la comunicación. [4] Además, el módulo utiliza el protocolo de comunicación NMEA (National Marine Electronics Association) que permite enviar los datos en forma de sentencias donde contienen información sobre la posición, velocidad, hora y otros datos relacionados con el GPS

Objetivo(s):

Configurar el módulo NEO-6M a través de un Arduino para la lectura de datos arrojados por el GPS.

Material y Equipo para utilizar.

Placa de Arduino Uno®, módulo neo-6m. ID PlataformIO, jumpers.

Metodología:

1. En el diagrama se muestran las conexiones necesarias para el funcionamiento del módulo, este es alimentado mediante la salida de 5v del Arduino, el pin Rx del módulo se conecta al pin 3 del Arduino y el pin Tx se conecta al pin 4 del Arduino y por último el pin GND se conecta a una salida de GND del Arduino.

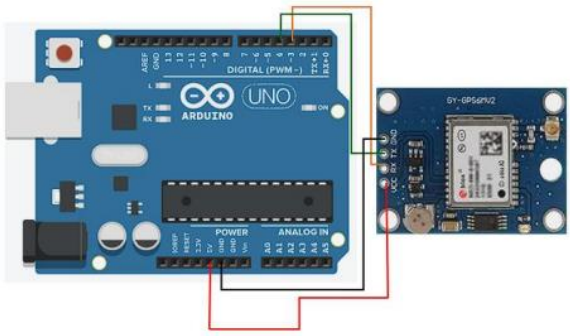

Figura 8. Conexión de Arduino Uno®y GPS NEO-6M

Fuente: Imagen elaborada con software libre

2. Para obtener datos a través del módulo GPS, se elaboró el siguiente código, el cual se encarga de leer datos GPS haciendo uso de una biblioteca llamada TinyGPS y comunicarse a través de una conexión serial con un módulo GPS.

```
#include <Arduino.h>
#include <SoftwareSerial.h>
#include <TinyGPS.h>

const int RX = 3;
const int TX = 4;

TinyGPS gps;
SoftwareSerial softSerial(RX, TX);

void setup()
{
```

```
  Serial.begin(9600);
  softSerial.begin(9600);
}

void loop()
{
  bool newData = false;
  unsigned long chars;
  unsigned short sentences, failed;

  // Intentar recibir secuencia durante un segundo
  for (unsigned long start = millis(); millis() -
start < 1000;)
  {
    while (softSerial.available())
    {
      char c = softSerial.read();
      if (gps.encode(c)) // Nueva secuencia
recibida
        newData = true;
    }
  }

  if (newData)
  {
    float flat, flon;
    unsigned long age;
    gps.f_get_position(&flat, &flon, &age);
    Serial.print("LAT=");
    Serial.print(flat ==
TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : flat, 6);
    Serial.print(" LON=");
    Serial.print(flon ==
TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : flon, 6);
    Serial.print(" SAT=");
    Serial.print(gps.satellites() ==
TinyGPS::GPS_INVALID_SATELLITES ? 0 :
gps.satellites());
    Serial.print(" PREC=");
    Serial.print(gps.hdop() ==
TinyGPS::GPS_INVALID_HDOP ? 0 : gps.hdop());
  }

  gps.stats(&chars, &sentences, &failed);
  Serial.print(" CHARS=");
  Serial.print(chars);
  Serial.print(" SENTENCES=");
  Serial.print(sentences);

 Serial.print(" CSUM ERR=");
 Serial.println(failed);
}
```

Figura 9. Imagen del código para configurar el GPS con Arduino
Fuente: Imagen elaborada con software libre de la IDE de Arduino Uno®.

Resultados de la práctica.

El informe documentado debe tener las siguientes secciones:

1. Portada
2. Resumen con Abstract
3. Introducción
4. Desarrollo de la práctica
5. Diagrama de electrónico de las conexiones*
6. Implementación de código funcuional en PlataformIO*
7. Resultados
8. Conclusión
9. Referencias

Actividad de Integración

Estas actividades deben incluirse en el reporte de práctica para evaluación (*).

Material complementario para el desarrollo de la práctica:

1. Llamas, L. (2016, 27 septiembre). *Localización GPS con Arduino y los módulos GPS NEO-6*. Luis Llamas. Accedido el 22 de abril de 2024. [En línea]. Disponible: Localización GPS con Arduino y los módulos GPS NEO-6
2. *Módulo GPS NEO-6M 7N APM2.5 – Tostatronic*. (s.f.). https://www.tostatronic.com/product/modulo-gps-neo-6m-7n-apm25/
3. *Tutorial Módulo GPS con Arduino*. (s. f.). Naylamp Mechatronics - Perú. Accedido el 22 de abril de 2024. [En línea]. Disponible: https://naylampmechatronics.com/blog/18_tutorialmodulo-gps-con-arduino.html

Práctica 8

Unidad de Competencia 1. Una revisión a los microcontroladores y sistemas embebidos.	
Tema:	4.2. Descripción general de sistema embebido.
Contenidos:	4.2.1. Arquitecturas existentes.
Duración:	1 hora con 40 minutos.
Valor de la Práctica:	1 puntos
Trabajo:	Individual

Integración de sistemas

Introducción

El concepto de CanSats fue introducido por primera vez en 1998 por el profesor Robert Twiggs de la Universidad de Stanford. La idea era crear un satélite que pudiera caber de una lata de refresco de 350 ml. El primer evento sobre CanSat tuvo lugar en 1999, con la participación de seis universidades. A lo largo de los años, se han establecido competiciones CanSat en todo el mundo, atrayendo a una amplia gama de investigadores e instituciones educativas.

La exploración del espacio y el estudio de la atmosfera terrestre han sido de los estudios más intrigantes desde años anteriores, por ello, a medida que la tecnología avanza, las nuevas herramientas y plataformas también lo hacen, esto con el fin de permitir a investigadores adentrarse a estas áreas de una manera más accesible y práctica. Un ejemplo de estas plataformas más utilizadas son los CanSats, ya que son la herramienta esencial que consta de dispositivos compactos que emulan las funciones básicas de un satélite dentro de una lata de refresco. Si bien, su principal finalidad de estos dispositivos es educativa, por lo que sus elementos suelen ser accesibles para cualquier persona que desee aprender sobre tecnología espacial; se utilizan también para probar y mejorar el hardware espacial, lo que podría beneficiar potencialmente a futuras misiones espaciales.

Objetivo(s):

Desarrollar un prototipo de CanSat que adquiera y transmita datos ambientales y de posicionamiento durante su vuelo, utilizando módulos como GPS y LoRa, con el fin de facilitar la comprensión del hardware espacial.

Material y Equipo para utilizar.

Modulo LoRa, Arduino Uno®, Sensor BMP280, Modulo GPS, Pila de 9V, Jumpers

Metodología:

El esquema de conexión se muestra en la Figura

1. **Circuito del Transmisor:** Se detallan las conexiones de los componentes, como el módulo LoRa, Arduino Uno®, sensor BMP280, módulo GPS, y la pila de 9V. Cada uno de estos elementos se conecta al Arduino de forma específica, como se menciona en el documento.

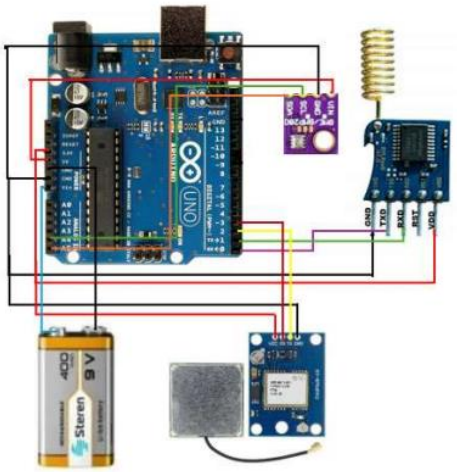

Figura 10. Conexión del GPS con Arduino y los elementos de transmisión
Fuente: Imagen propia realizado con software libre

2. **Circuito del Receptor:** Similar al transmisor, pero enfocado en recibir los datos por el transmisor mediante otro módulo LoRa y Arduino Uno®.

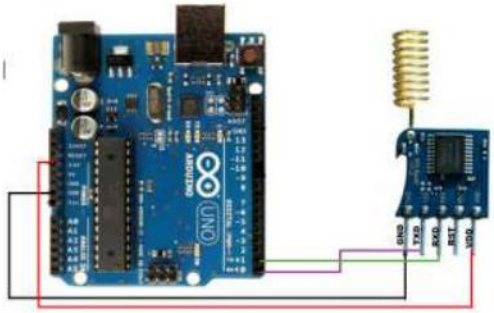

Figura 11. Conexión de la comunicación de recepción
Fuente: Imagen propia realizado con software libre

3. **Programación:** Se desarrollaron códigos para el receptor y transmisor en Arduino, además de un script en Python para leer, procesar y graficar los datos recibidos. Los códigos respecto a la transmisión y recepción se abordaron previamente en la **practica 5**, se enfoca únicamente al código que permite la lectura de los datos a través de Python.

```
import serial
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

# Configura el puerto serial
ser = serial.Serial('COM3', 9600)  # Asegúrate de
reemplazar 'COM3' por el puerto que uses
# Variables para almacenar datos
temperaturas = []
presiones = []
altitudes = []
humedades = []
latitudes = []
longitudes = []

# Archivo para guardar los datos
file_path = "datos_sensores.txt"

# Función para escribir datos en el archivo
def write_to_file(temp, pres, alt, hum, lat, lon):
    with open(file_path, 'a') as f:
        f.write(f"{temp}, {pres}, {alt}, {hum},
{lat}, {lon}\n")

def parse_data(data):
    parts = data.split()
    temp = float(parts[0].split(":")[1])
    pres = float(parts[1].split(":")[1])
    alt = float(parts[2].split(":")[1])
    hum = float(parts[3].split(":")[1])
    lat = float(parts[4].split(":")[1])
    lon = float(parts[5].split(":")[1])
    return temp, pres, alt, hum, lat, lon

def update(frame):
    line = ser.readline().decode('utf-8').strip()
    if "Temperatura" in line:
        temp, pres, alt, hum, lat, lon =
parse_data(line)
        temperaturas.append(temp)
        presiones.append(pres)
        altitudes.append(alt)
        humedades.append(hum)
        latitudes.append(lat)
        longitudes.append(lon)

        # Guardar los datos en el archivo
        write_to_file(temp, pres, alt, hum, lat,
lon)

        # Limita la cantidad de datos a mostrar
(por ejemplo, los últimos 50)
        temperaturas[:] = temperaturas[-50:]
        presiones[:] = presiones[-50:]
```

```
        altitudes[:] = altitudes[-50:]
        humedades[:] = humedades[-50:]
        latitudes[:] = latitudes[-50:]
        longitudes[:] = longitudes[-50:]

        ax1.clear()
        ax2.clear()
        ax3.clear()
        ax4.clear()
        ax5.clear()
        ax6.clear()

        ax1.plot(temperaturas, color='red',
label="Temperatura (°C)")
        ax2.plot(presiones, color='blue',
label="Presión (hPa)")
        ax3.plot(altitudes, color='green',
label="Altitud (m)")
        ax4.plot(humedades, color='orange',
label="Humedad (%)")
        ax5.plot(latitudes, color='purple',
label="Latitud")
        ax6.plot(longitudes, color='brown',
label="Longitud")

        ax1.legend(loc='upper right')
        ax2.legend(loc='upper right')
        ax3.legend(loc='upper right')
        ax4.legend(loc='upper right')
        ax5.legend(loc='upper right')
        ax6.legend(loc='upper right')

# Configura la figura y los ejes para graficar
fig, ((ax1, ax2), (ax3, ax4), (ax5, ax6)) =
plt.subplots(3, 2, figsize=(10, 12))

# Configura la animación
ani = FuncAnimation(fig, update, interval=1000)

plt.tight_layout()
plt.show()
```

Figura 12. Codificación en el ID PlataformIO.

Fuente: Imagen elaborada con software libre PlataformIO.

Resultados de la práctica.

El informe documentado debe tener las siguientes secciones:

1. Portada
2. Resumen con Abstract
3. Introducción
4. Desarrollo del CanSat con sus susbsitemas*
5. Visualización de datos a través de Python*
6. Esquemas y códigos desarrollados para la práctica*
7. Resultados
8. Conclusión
9. Referencias

Actividad de Integración

Estas actividades deben incluirse en el reporte de práctica para evaluación (*).

Material complementario

No aplica.

Referencias del manual de prácticas.

1. Camposano, R., & Wilberg, J. (1996). Embedded system design. *Design Automation for Embedded Systems*, *1*(1–2), 5–50. https://doi.org/10.1007/bf00134682
2. Marwedel, P. (2021). *Embedded system design: Embedded systems foundations of cyber-physical systems, and the internet of things* (4a ed.). Springer Nature.
3. Hayes, T. L., & Kanan, C. (2022). Online continual learning for embedded devices. En *arXiv [cs.LG]*. http://arxiv.org/abs/2203.10681
4. Llamas, L. "Localización GPS con Arduino y los módulos GPS NEO-6". Luis Llamas Ingeniería, informática y diseño. Accedido el 29 de mayo de 2024. Disponible en línea: https://www.luisllamas.es/localizacion-gps-con-arduino-y-los-modulos-gps-neo-6/
5. "Tutorial Módulo GPS con Arduino". Naylamp Mechatronics - Perú. Accedido el 29 de mayo de 2024. Disponible en línea: https://naylampmechatronics.com/blog/18_tutorial-modulo-gps-con-arduino.html
6. Gonzalez, C. "Tutorial de cómo iniciar una comunicación de largo alcance (LoRa) con el módulo LoRa RYLR998". Anatronic S.A. Accedido el 29 de mayo de 2024. Disponible en línea: https://anatronic.com/comunicacion-de-largo-alcance-lora-con-el-modulo-lora-rylr998/
7. "Guide for BME280 Sensor with Arduino (Pressure, Temperature, Humidity) | Random Nerd Tutorials". Random Nerd Tutorials. Accedido el 29 de mayo de 2024. Disponible en línea: https://randomnerdtutorials.com/bme280-sensor-arduino-pressure-temperature-humidity/
8. *Programa Espacial Universitario | Descripción CANSAT*. (s.f.). Accedido el 29 de mayo de 2024. Disponible en línea: http://peu.unam.mx/descripcionCANSAT.html
9. Chun, C., Tanveer, M. H., y Chakravarty, S. "The CanSat Compendium: A Review of Scientific CanSats." Machines, 11(7), 675. Disponible en línea: https://doi.org/10.3390/machines11070675
10. Llamas, L. "Medir temperatura y presión barométrica con Arduino y BMP280". Accedido el 29 de mayo de 2024. Disponible en línea: https://www.luisllamas.es/medir-temperatura-y-presionbarometrica-con-arduino-y-bmp280/

Centro Universitario UAEM Atlacomulco
Km. 60 Carretera Toluca – Atlacomulco, Col. San Francisco Chalchihuapan C.P. 50450, Atlacomulco, Estado de México.
Tels. (712) 12204 36, 1220446, 1220535 **e-mail:** cuatla@uaemex.mx **sitio web:** https://cuatlacomulco.uaemex.mx **Facebook:** /cua.uaemex.atlacomulco