

CanSat Laika: Informe Final

Bucaramanga, Colombia
Universidad Industrial de Santander

Abstract—En el siguiente trabajo se presentan los resultados de la implementación de un CanSat destinado a funcionar a una altura de 100 [m]. El proyecto fue realizado por estudiantes de octavo semestre de ingeniería electrónica de la Universidad Industrial de Santander como proyecto final para la asignatura de Instrumentación Electrónica.

Index Terms—CanSat, Arduino Nano, PCB, Telemetría, Sensórica, Transmisión, Recepción, ESA

I. INTRODUCCIÓN

Como proyecto final de la asignatura de instrumentación electrónica del programa de ingeniería electrónica de la Universidad Industrial de Santander, se realizó un CanSat, dispositivo con funcionalidad similar a la de un satélite que está empacado en una estructura pequeña, como la de una lata de gaseosa. Para diseñar e implementar este proyecto se contó con un equipo de 27 estudiantes, estos recursos humanos fueron distribuidos en 7 subsistemas diferentes, con el propósito de distribuir el trabajo de los diferentes requerimientos técnicos del satélite.

Los subsistemas del CanSat son: Diseño estructural, sistema de potencia, microcontrolador/PCB, instrumentación, sistema de despliegue/paracaídas, sistema de comunicaciones y estación de telemetría.

El diseño responde a las necesidades planteadas a partir de la dinámica de funcionamiento establecida: El CanSat debe volar a una altura máxima de 100 metros colgando de una estructura (diseñada también en el proyecto) que va amarrada a una góndola que a su vez va colgando de un dron, una vez alcance esta altura, el CanSat debe desacoplarse y caer de manera segura mientras transmite datos y graba el descenso desde su parte superior para obtener visión del dron mientras cae.

II. DISEÑO E IMPLEMENTACIÓN

El diseño responde a las necesidades planteadas a partir de la dinámica de funcionamiento establecida: Volar a una altura máxima de 100 metros colgando de una estructura (diseñada también en el proyecto) amarrada a la góndola, colgando de un dron, una vez alcanzada la altura, debe desacoplarse y caer de manera segura mientras recoge datos y se graba el descenso desde su parte superior para obtener visión del dron mientras cae.

Los subsistemas enfocaron su trabajo alrededor de esta premisa para darle una solución enfocada.

III. DISEÑO ESTRUCTURAL

1) Objetivos:

- Diseñar una estructura que cumpla con normas de la ESA en sus dimensiones.
- Buscar un material de construcción capaz soportar los componentes dentro de este y que los resguarde de colisiones o fuertes movimientos.
- Diseñar e implementar una propuesta de estructura interna que almacene los componentes para su fácil extracción y manipulación de su circuitería interna.

A. Implementación

Se diseña una estructura imprimible en 3D junto con una tapa desenróscable, vigas, columnas y conjunto de baquetas con el fin de hacer lo más liviana y comprimida pueda ser.

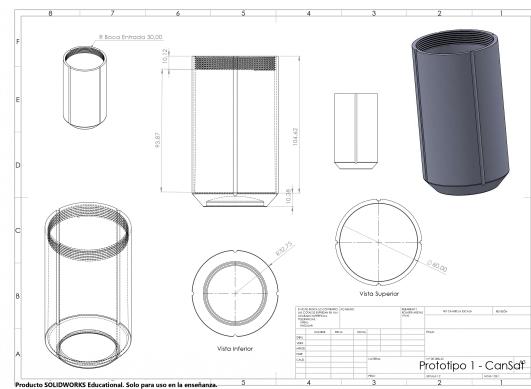


Fig. 1. Planos hechos en SolidWorks

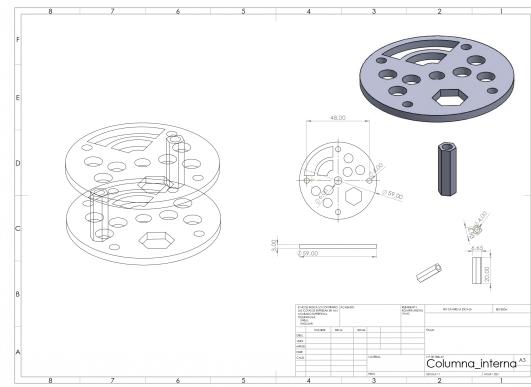


Fig. 2. Planos hechos en SolidWorks

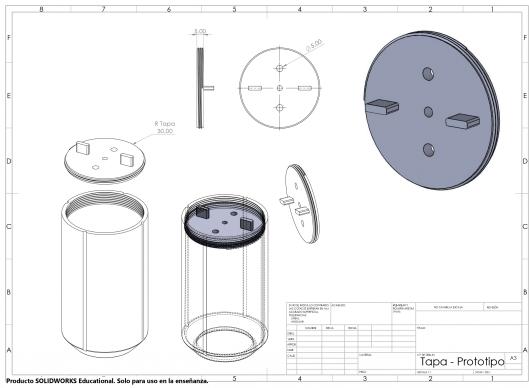


Fig. 3. Planos hechos en SolidWorks



Fig. 4. Impresión ya implementada

IV. SISTEMA DE DESPLIEGUE

1) Objetivo General: Diseñar e implementar un sistema que desacople el CanSat y posteriormente el despliegue del paracaídas para disminuir la velocidad de descenso.

2) Objetivo Específicos:

- Identificar las variables que intervienen en el momento de desacople del CanSat y apertura del paracaídas.
- Determinar el área del paracaídas por medio de la velocidad de aterrizaje deseada, el peso, el coeficiente de arrastre y la densidad del aire.
- Implementar el sistema diseñado.

3) Requerimientos:

- Para la implementación del sistema de despliegue se utilizó un paracaídas StratoChute el cual fue suministrado por el grupo de investigación SCUA a través del profesor Julian Rodriguez Ferreira, este paracaídas cuenta con un área de 24" y esta elaborado en Rip-Stop Nylon que ofrece una relación ligereza/resistencia, durabilidad y protección antidesgarro.

Los cálculos realizados se llevaron a cabo teniendo en cuenta el área del paracaídas, contemplando las diversas



Fig. 5. Gondola utilizada

variables que afectan el movimiento de caída libre.

- Al momento de desacople, se utiliza un servomotor con un giro de 180° donde el Eje 0° estará indicado hacia la parte inferior de la estructura del CanSat.



Fig. 6. Servomotor

4) Metodología:

- En el desarrollo de este sistema se realizó una investigación profunda con el fin de entender y comprender el movimiento de caída libre, el uso y las variables que

afectan el paracaídas, para poder obtener una disminución de velocidad en el descenso.

- Se realizaron diversas pruebas con diversos elementos que simularan el peso real del cansat para tener un promedio de tiempo en el cual el paracaídas se desplegaría totalmente.
- Una vez llegue a la altura deseada (100 m) el altímetro dará un señal que permita al microcontrolador activar el servomotor, el cual a través del giro por medio de un elástico liberara el CanSat del brazo que lo soporta.

V. SISTEMA DE POTENCIA

1) *Objetivo General:* Diseñar e implementar un sistema de distribución de potencia que garantice el funcionamiento de todos los subsistemas.

2) *Objetivos específicos:*

- Distribuir de manera adecuada el consumo de energía de las baterías en los diferentes sistemas.
- Garantizar la energía suficiente para el funcionamiento del cansat por lo menos de 1 hora en uso constante.

3) *Requerimientos impuestos:*

- Espacio de las baterías: Largo*Alto*Ancho: 4[cm]*2[cm]* 0.8[cm].
- Espacio de los reguladores de voltaje: Largo*Alto*Ancho: 3.6[cm]*1.7[cm]* 0.7[cm].
- Peso de la batería: 40 g.
- Peso de los Reguladores Voltaje MT3608: 4*5[g]= 20[g].
- Ubicación de la batería: Parte inferior del CanSat, altura de 25 [mm].
- Ubicación del regulador: PCB's sobre la batería las cuales llevan cada una 2 reguladores y tienen un alto de 15[mm].
- Ubicación del sistema de inicialización a un costado o en la parte superior del CanSat.

4) *Metodología:* Tomando los sensores y dispositivos a utilizar se realizo el cálculo de la energía necesaria a distribuir para todos los subsistemas, el cálculo inicial correspondio a 1943[mWh] como se muestra en la siguiente figura:

Sensores	Voltaje[V]	Corriente[mA]	Potencia[mW]	Horas de uso	Potencia [mWh]
Sensor de temperatura LM35	5	0,06	0,3	1	0,3
GPS NEO-7M	5	35,00	175	1	175
Sensor inercial pololu altimu 10v5	5,00	5,00	25	1	25
Microcontrolador(Arduino NANO)	5	19	95	1	95
Servo SG90 espera	5	3	15	1	15
Servo SG90	5,00	200,00	1000	0,017	17
Camara(España)	12,00	100,00	1200	1	1200
Camara(Active)	12	300	3600	0,017	61,2
Transmisor: NRF24L01	3,50	110,00	385	1	385
Potencia Total					1973,5

Fig. 7. Baterías tipo Lipo

Sin embargo, las diferentes tensiones a utilizar sumado a la búsqueda de un peso y dimensiones apropiadas para las baterías llevo a utilizar 2 baterías de polímero de litio (LiPo) las cuales tienen la mejor relación potencia-peso, la capacidad de descarga de la batería es de 20C lo que indica que en caso de requerir una 20 veces más corriente que su valor nominal

la batería puede suministrar la corriente, las especificaciones de la batería son de 3.7[V] y 700[mAh] para una energía disponible total de 2590 [mWh] por batería.



Fig. 8. Baterías tipo Lipo

Estas baterías se ubicaron en la parte inferior del CanSat y suministran energía a dos PCB que se realizaron con el fin de distribuir de mejor manera la potencia requerida en todos los demás dispositivos (servo, cámara, transmisión, etc.). Las dos PCB mencionadas se ilustran a continuación:

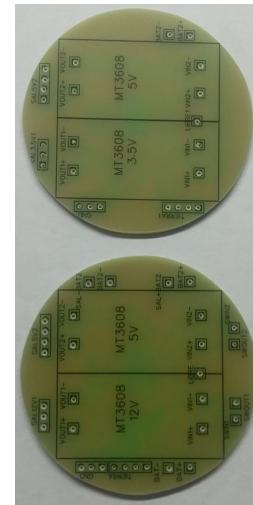


Fig. 9. PCB para la distribución de potencia (Placa Superior)

5) *Distribución de la alimentación a las PCB:* Puesto que se necesitan 3 tensiones diferentes, 3.5[V], 5[V] y 12[V] y se tienen 2 baterías de 3.7[V] se requiere el uso de elevadores de tensión, sumado a esto dispositivos como el servomotor tienen un pico de corriente alto el cual puede dar lugar a fallas en el funcionamiento de los demás dispositivos llegando incluso a dañarlos, con esto en mente se decidió utilizar 4 reguladores MT3608 los cuales elevaran las tensiones a las deseadas.

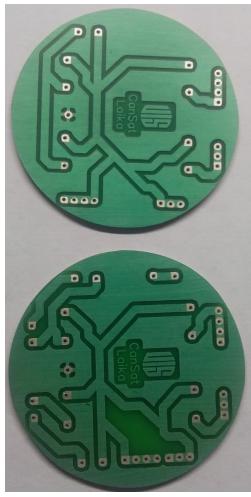


Fig. 10. PCB para la distribución de potencia (Capa Inferior)

La distribución es la siguiente:

- 2 tensiones de 5[V], una destinada para el microcontrolador arduino y los diferentes sensores y otra especifica para el servomotor.
- Una tensión de 3,7[V] para el transmisor RF.
- Una tensión de 12[V] para la cámara y su transmisor.

La distribución se realiza conectando una batería a 2 PCB, con esto en mente la idea de distribución optima es una donde los consumos de las PCB sean similares con el fin de que las baterías tengan un desgaste parecido.

Distribución elegida:

El dispositivo que más energía requiere es la cámara, necesitando para su funcionamiento casi 1300[mWh], contrario a ella el que menos consume es el servomotor, el cual, en pruebas físicas se obtuvo el valor de la corriente que consume su activación, el valor es de 200 [mA] pero la activación dura unos pocos segundos, por tanto la energía que requiere es de tan solo 16[mWh], Con base en estos cálculos se decidió dejar estos 2 componentes conectados a 1 PCB y el transmisor, el Arduino y los sensores a la otra PCB.

La autonomía de la batería se puede calcular tomando la energía consumida y dividiéndola entre la energía total, para que este valor de en minutos se multiplica esta razón por 60, los cálculos de autonomía son los siguientes:

Potencia consumida por los dispositivos conectados a los 2 reguladores de cada PCB	Consumo total por PCB [mWh]	Autonomía[Minutos]
Potencia consumida por la PCB 1(Cámara y transmisor de la cámara+ Servomotor)	1261,88	123,1495863
Potencia consumida por la PCB 2(Sensores y arduino + Transmisor RF)	658,8	235,8834244

Fig. 11. Cálculo de la autonomía teórica del Cansat)

6) *Diseño de las PCBs de potencia:* En el diseño de las PCB's de potencia, se tomaron en cuenta los diferentes parámetros y requerimientos del sistema, los cuales fueron definidos por la cantidad de elementos usados, corriente y potencia a suministrar, etc. Con la finalidad de suministrar los

voltajes requeridos del sistema teniendo en cuenta la corriente de consumo de estos, se usaron reguladores-elevadores de voltaje MT3608 dos por cada placa, los cuales soportan 1.5 A de suministro gracias a su bobina interna.

El diseño de la placa fue realizado con el software gratuito de EasyEDA, la cual posee la entrada de 2 baterías de 3,7 V señaladas por los pines BAT+, BAT-, BAT2+, BAT2-. A su vez posee 2 salidas de energía a la segunda pcb identificado por los pines SAL-BAT2 y SAL+BAT2. El sistema posee un switch push como sistema de inicialización el cual va conectado en los pines SWN1, SWOUT1 para la primer batería la cual es encargada de suministrar energía a la pcb inferior, los pines SWN2, SWOUT2 para la pcb superior. Con el uso de los reguladores-elevadores MT3608, se logra otorgar 4 salidas de 5V y 3 salidas de 12 V con sus respectivas entradas de tierra. El diseño se realizó acorde a los requerimientos impuestos por los demás subsistemas del Cansat Laika, teniendo una ranura para el paso de cableado referente a sus baterías. Por otro lado, el tamaño de sus pistas se realizó de 2mm capaces de otorgar la corriente exigida por los demás sistemas sin sufrir fracturas o recalentamiento en estas, la pcb tiene una capa de ruteado al torno del pin libre facilitando así el trabajo con CNC. Además de eso, la placa tiene impreso el logotipo de la universidad y el nombre de Cansat.

El siguiente grafico muestra a detalle las medidas impuestas en la placa.

- Placa inferior

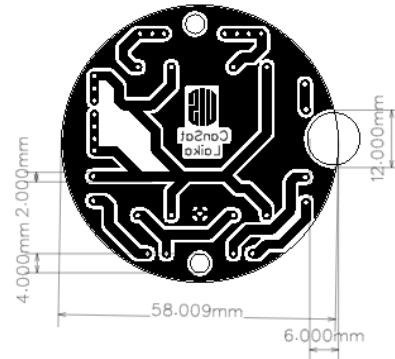


Fig. 12. Dimensiones de PCB diseño en Easy EDA (Placa Inferior)

• Placa superior El diseño de la placa fue realizado con el software gratuito de EasyEDA, la cual posee una entrada de batería de 3,7 V de la placa inferior, señalada por los pines BAT2+, BAT2-. El sistema posee un switch push como sistema de inicialización el cual va conectado en los pines SWN2, SWOUT2 para la pcb superior.

Con el uso de los reguladores-elevadores MT3608, se logra otorgar 4 salidas de 5V y 3 salidas de 3.5 V con sus respectivas entradas de tierra. El diseño se realizó acorde a los requerimientos impuestos por los demás subsistemas del Cansat Laika, teniendo dos ranuras para

el paso de cableado referente a las salidas de la pcb inferior. Por otro lado, el tamaño de sus pistas se realizó de 2mm capaces de otorgar la corriente exigida por los demás sistemas sin sufrir fracturas o recalentamiento en estas, la pcb tiene una capa de ruteado al torno del pin *libre₁* facilitando así el trabajo con CNC. Además de eso, la placa tiene impreso el logotipo de la universidad y el nombre de Cansat.

El siguiente gráfico muestra a detalle las medidas impuestas en la placa.

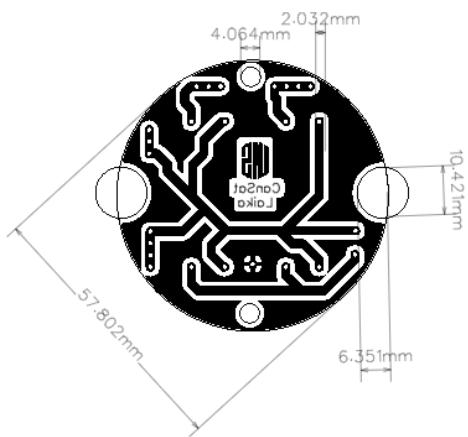


Fig. 13. Dimensiones de PCB diseño en Easy EDA (Placa superior)

El resultado final de las PCB es el siguiente:

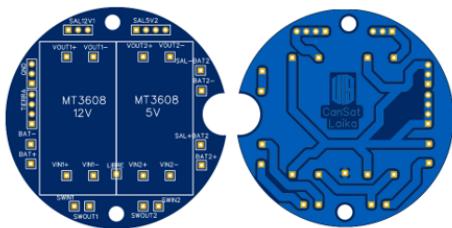


Fig. 14. PCB diseño en Easy EDA (Placa Inferior)

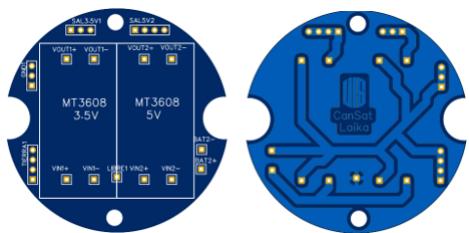


Fig. 15. PCB diseño en Easy EDA (Placa Superior)

La inicialización en el diseño de las pcbs de potencia, se

tomaron en cuenta los diferentes parámetros y requerimientos del sistema, los cuales fueron definidos por la cantidad de elementos usados, corriente y potencia a suministrar, etc. Con la finalidad de suministrar los voltajes requeridos del sistema teniendo en cuenta la corriente de consumo de estos, se usaron reguladores-elevadores de voltaje MT3608 dos por cada placa, los cuales soportan 1.5 A de suministro gracias a su bobina interna. de toda la potencia y por consiguiente del funcionamiento del Cansat en total se realizó mediante un switch estratégicamente ubicado en la parte superior interna del Cansat, este switch se compone de dos canales no conectados entre sí como muestra la siguiente figura los cuales conmutan entre un pin central y sus extremos, dando así lugar a los estados de encendido y apagado.

Los módulos reguladores elevadores de tensión utilizados para obtener las tensiones deseadas son los modelos MT3608, estos módulos toman voltajes entre 2[V] y 24 [V] y las eleva hasta 28 [V] con una salida de corriente de hasta 1[A], su tamaño lo hace perfecto para colocar 2 módulos por PCB y así conseguir las tensiones deseadas en el menor espacio posible

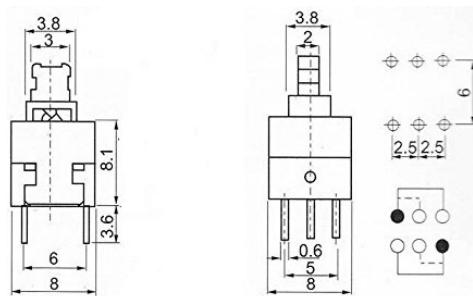


Fig. 16. Switch de inicialización



Fig. 17. Modulo elevador MT3608

VI. INSTRUMENTACIÓN

A. Objetivo General

Medir a lo largo del trayecto las variables de temperatura, presión, altitud, posición, aceleración y orientación.

B. Objetivos específicos

- Realizar la búsqueda de los sensores que cumplen con los requerimientos generales para las condiciones del lugar donde se realizará la prueba del prototipo de CanSat.
- Realizar una comparación objetiva, en lo que a calidad precio se refiere, escogiendo así el mejor candidato de las opciones planteadas y definiendo entonces un presupuesto.

C. Requerimientos Generales a tener en cuenta para la selección de sensores

- Las dimensiones internas del cansat son un alto de 115 mm (sin contar al paracaídas) y un diámetro de 66 mm, los sensores deben operar dentro de los siguientes rangos: temperaturas entre -7.6°C y 42°C además de presiones entre 90 kPa y 103.4 kPa.

D. Selección de sensores

- En el proceso de selección de los sensores que se adaptaran mejor a los requisitos impuestos para el Cansat se realizaron comparaciones entre algunos candidatos que ya cumplian con los parametros impuestos.
- En el caso de la temperatura se plantearon los siguientes sensores:
 - sensor DS18B20+
 - Sensor Lm35
 - Sensor tmp36
- Despues de un analisis de cada una de las propuestas, por temas de facilidad comercial(se conseguia más facil), presupuesto y facilidad de lectura de datos se tomo la desicion de seleccionar el lm35 con su lectura analogica de la temperatura.
- En el caso de la presión y altitud se plantearon los siguientes sensores:
 - sensor LPS25H (altimu10-v5)
 - Sensor LPS331AP
 - Sensor MPX5100
- Despues de un analisis de cada una de las propuestas, se escogió el sensor LPS25H el cual al estar incluido dentro del altimu se facilitaba en terminos de implementacion, puesto que no consume espacio adicional en el ensamblaje y el altimu fue facilitado por el director del proyecto, por otra parte el sensor MPX5100 esta pensado para implementaciones más robustas por lo que este sensor podría funcionar para la implementacion del cansat de 40km.
- En el caso de la aceleracion y orientación se plantearon los siguientes sensores:
 - sensor IMU GY-89
 - Sensor Altimu-10 v5

- Ya que se decidido utilizar el altimu10-v5 para la medición de altitud y presión se planteo usar este para medir la aceleracion y la orientacion con el mismo instrumento y gastar menor espacio dentro del cansat.
- En el caso de la ubicación se plantearon los siguientes sensores:

Gps Neo-7m

Gps Neo-6m

- Para el caso del GPS se habia escogido en un principio el Neo-7m pero ya que su predecesor el Neo-6m tambien cumplia con la mayoria de caracteristicas que este ostentaba, ademas de que el 6m ya se tenia en posesion, se cambio de seleccion para evitar incurrir en gastos innecesarios.

- En el caso de la cámara se plantearon los siguientes sensores:

Esp32 Cam

Usaq 700tvl Mini Camara Fpv 13 Ccd 28mm Ntsc
Avion No Tripul

- Para el caso de la camará se investigo acerca la camara de la esp pero se encontraron problemas para la transmision de la imagen ya que esta camara funciona generalmente para aplicaciones con wifi y su resolucion es mucho mas limitada que la de la cámara fpv.

E. Descripción de Sensores

- Sensor LM35:
 - Sensor de temperatura analógico.
 - Precisión calibrada de 1°C.
 - La salida es lineal y cada grado centígrado equivale a 10mV.
 - El empaquetado de este sensor es through-hole TO-92.
 - Alimentación: 5 VDC a 60 uA.
- GPS NEO-6M:
 - Ultra sensibilidad: -165dBm
 - Máxima altura medible: 18000
 - Alimentación: (3.5 – 5)VDC a 35 mA
 - Frecuencia de actualización 5Hz
 - velocidad de desplazamiento máxima: 500m/seg
 - Protocolo NMEA (a 9600bps)
 - 01 puerto serial
 - Antena incorporada de 18.2 x 18.2 x 4.0 mm
 - Rango de temperatura: -40 to 85 C
 - Cumple estándar RoHS
 - Tamaño reducido 30mm x20mm x 11.4mm
- Transmisor TS832
 - Alimentación: 12 VDC a 220 mA
 - De largo alcance más de 3 km gama en el área abierta
 - Plug and Play
 - Dos botones de commutación: fácil cambiar canales de frecuencia y bandas
 - Apagado de memoria: Repetir la última banda de frecuencia y canal
 - 40 canales: 5 bandas y todas las frecuencias

compatibles, para a, b, e, f, bandas de r

- Transmisor universal de vídeo y el receptor para todos, compatible con varios tipos de juguetes de control remoto (RC helicóptero, Quadcopter, avión y barco, etc.).

- Sensor inercial pololu altimu 10-v5:

- Alimentación: 5 VDC a 5 mA
 - Dimensiones: 25x13x3 mm
 - Peso: 1 gramo (sin pines)
 - Alimentación: 2.5 a 5.5V
 - Consumo: 6 mA
 - Formato de salida: I2C
 - Sensibilidad: Gyro: ± 250 , ± 500 , or $\pm 2000^\circ/\text{s}$,
Acelerómetro: ± 2 , ± 4 , ± 8 , or ± 16 g, magnetómetro:
 ± 1.3 , ± 1.9 , ± 2.5 , ± 4.0 , ± 4.7 , ± 5.6 , o ± 8.1 gauss,
Barómetro: 260 mbar a 1260 mbar (26 kPa a 126 kPa)

- FPV Camera:

- Distancia focal de la lente: 2.1mm 2.5mm 2.8mm
3.6mm 6mm 8mm 12mm opcional.
 - Iluminación mínima: 0.1Lux/F1.2(color) 0Lux(black and white)(IR ON)
 - Balance de blanco automático ON/OFF
 - Velocidad de fotograma de video:
PAL: 1020Hx596V NTSC:1020Hx508V
PAL: 976Hx582V NTSC:976Hx494V
 - Salida de video 1.0VP-P/75
 - Potencia de entrada FPV DC12V
 - Consumo de energía 70mA
 - Rango de temperatura: -20°C to 60°C

F. Conexiones

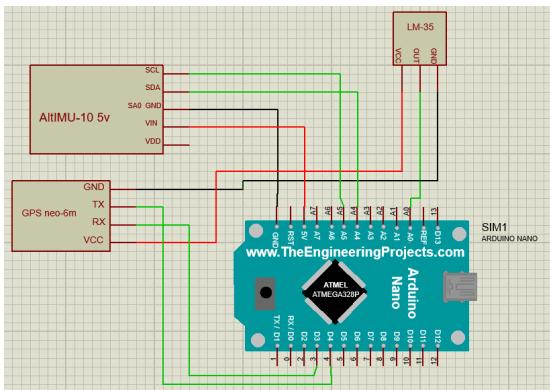


Fig. 18. Esquemático de conexiones entre sensores

G. Entregables

- 1 Transmisor TS832, 1 FPV Camera, 1 Sensor LM35, 1 GPS NEO-6M, 1 Sensor inercial pololu altimu 10 v5
 - Repositorio en Github con los códigos para la validación de la lectura de datos de los sensores y cámara [2]

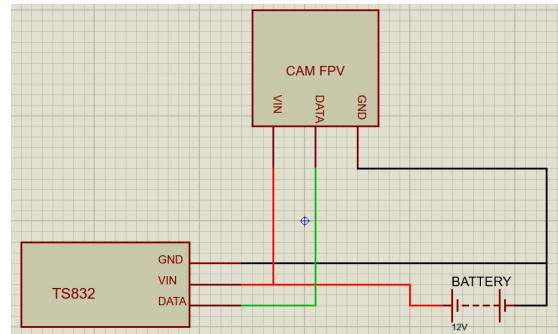


Fig. 19. Esquemático de conexión cámara y transmisor

H. Recursos financieros

Presupuesto máximo para el subgrupo de Instrumentación:
150.000 (COP).

I. requerimientos impuestos por el subgrupo de instrumentación

requerimientos presupuestos	Concepto	unidades	Valor total []
	masa	[g]	70
	monetario	[\\$]	5000
	volumen	[cm ³]	44.313
	potencia	[w]	3.6828

TABLE I
PRESUPUESTOS TOTALES

VII. SISTEMA DE COMUNICACIONES

1) Objetivo General: Diseñar, implementar y validar un sistema de comunicaciones de radiofrecuencia que permita la transmisión de datos e imágenes a la góndola en un enlace de mínimo 1 km.

2) Objetivos específicos:

- Seleccionar un módulo de comunicaciones que fuese adecuado para la aplicación.
 - Diseñar un protocolo de comunicaciones que garantice el envío y recepción de la información proveniente de los sensores de: Temperatura, IMU y GPS.

3) Requerimientos impuestos:

- La antena debe ir en la parte lateral superior del CanSat y requiere un orificio para poder sacar la misma.
 - El módulo de transmisión se comunica con el microcontrolador mediante el protocolo SPI. Por tanto, necesita los pines de VCC, GND, CSN, CE, SCK, MOSI, MISO.
 - El peso del módulo de comunicaciones escogido es de 25g y tiene dimensiones de: 4.6 cm*1.7 cm*1.2 cm.
 - En emisión el módulo consume 3.3V a 115mA.
 - Cada módulo tiene un costo de 15000 COP.

4) *Metodología:* El módulo de comunicaciones seleccionado fue el nRF24L01

La decisión fue tomada teniendo en cuenta diferentes parámetros con base en la funcionalidad del módulo y el precio. Este ofrece la posibilidad de seleccionar entre tres



Fig. 20. Módulo de Comunicaciones seleccionado

posibles velocidades de transferencia de datos: 250 kbps, 1 Mbps y 2 Mbps. Específicamente el módulo escogido fue el nRF24l01 + PA + LNA que trae antena como la que se puede apreciar en la imagen, estas le dan un alcance máximo de 1000 metros, predominantemente a línea de vista, pues la frecuencia a la que opera este módulo es a 2.4 GHz con la posibilidad de seleccionar 126 canales diferentes. En el lado receptor se tiene adicional al módulo de recepción un Arduino UNO que hace de puente entre lo que recibe el módulo y lo que va a la estación de Telemetría.

Al implementar el sistema de comunicaciones se cayó en cuenta de una limitación existente en cuanto a estos módulos y es que a nivel de software la transmisión de la información se hace con el comando `radio.write(data, Sizeof((data))` y no se pueden transmitir más de 32 bytes por comando. Teniendo en cuenta la cantidad de información que se requiere enviar por el radioenlace, un solo comando de escritura no es suficiente para enviar toda la información, por lo que la solución implementada fue separar los datos en diferentes paquetes y enviarlos en diferentes comandos de escritura.

```
31.25&90689.45&926&NOLa&NOLo*
-0.49&0.04&-0.82#
0.04&-0.07&-0.02~
```

Fig. 21. Datos en el módulo de transmisión

Intuitivamente, se pensaría que enviar dos comandos de escritura implica tener que usar dos comandos de lectura en el receptor, dichos comandos son el `radio.read(data)` pero esto abre dos canales de escritura haciendo que todo se guarde en duplicado en las variables de recepción. Para solucionar

este problema se realizó un protocolo de comunicaciones, dicho protocolo funciona gracias a la selección de caracteres especiales.

```
*31.25&90689.45&926&NOLa&NOLo*-0.49&0.04&-0.82#~0.04&-0.07&-0.02~
```

Fig. 22. Datos en el módulo de recepción

Los & son utilizados para separar las variables dentro del paquete y los caracteres *, y por ejemplo:

Por ejemplo, en la figura 13 el paquete 31.25&90689.45&926&NOLa&NOLo* los ampersand identifican la separación intrapaquete de variables, entonces 31.25 queda guardado en una variable, 90689.45 en otra, 926 en otra y lo mismo sucede con NOLa y NOLo (de hecho estas son las variables de temperatura, presión, altitud, latitud y longitud, tenga en cuenta que estas dos últimas marcan NOLa y NOLo pues esta prueba del radio enlace se hizo sin disponibilidad del GPS), el * le indica al sistema que esta es la finalización del primer paquete. Cabe destacar que surgieron problemas al implementar este sistema enviando variables de tipo flotante por lo que se enviaron variables de tipo String. La implicación de esto es que un flotante en Arduino pesa 4 bytes, siendo cualquier número entre 3.4028235E+38 y -3.4028235E+38, mientras que en un String cada carácter pesa 1 byte, por lo que, volviendo al ejemplo de la trama de temperatura, presión y altitud, esta tendría un peso de 20 bytes.

Para la recepción de los datos, queriendo trabajar a una potencia máxima para garantizar el alcance en la prueba de los 100 m, en esa condición el módulo de recepción consume una corriente aproximada de 45 mA, para poder garantizar que el módulo siempre tuviese esa corriente, en vez de alimentarlo con los 3.3 V de Arduino, se generó una solución basada en alimentarlo con la salida de 5 V (Esta provee más corriente) y pasar estos 5 V por un regulador LD1117V33C para bajar la tensión a 3.3 V, ya que en alimentaciones superiores a los 3.6 V el fabricante no garantiza la integridad del modulo.

VIII. ESTACIÓN DE TELEMETRÍA

1) *Objetivo General:* Crear una interfaz de software que permita procesar y visualizar los datos provenientes del CanSat.

2) *Objetivos específicos:*

- Diseño y maquetación de la interfaz.
- Visualización gráfica de datos en tiempo real de los valores entregados por los sensores, tales como temperatura, presión, altitud, ubicación GPS y las imágenes de la cámara.
- Almacenamiento de los datos y las imágenes de la lectura de la cámara.
- Repositorio en GitHub del software.

3) *Requerimientos impuestos:*

- Para la recepción de los datos del puerto serial a un PC, se debe conectar el módulo receptor a un Arduino y a partir de este se procesan los datos.

4) Metodología:

- Decodificación de los datos: Se estuvo en constante dialogo con el equipo de comunicaciones para conocer la forma en que nos iban a hacer llegar los datos al monitor serial.

Se llegó a la decisión de que los datos serían enviados en forma de filas de caracteres, separados en dos tramas. Las cuales se reconocían ya que la primera trama empezaba y finalizaba con asteriscos (*) y la segunda trama empezaba y finalizaba con numeral (#). Los datos dentro de cada trama eran separados con el símbolo Ampersand (&).

Los datos se leían desde el monitor serial de la IDE de Arduino para hacer su respectivo fraccionamiento y tratamiento en Python.

- Procesamiento de los datos: Luego de recibir los datos en bytes en Python tomados desde el monitor serial de Arduino, estos se decodifican a formato utf-8 lo cual los convierte en una cadena de caracteres que es procesada de acuerdo a sus tramas.

También se crea un buffer el cual guarda los últimos datos en caso de que exista alguna perdida de tramas.

- Almacenamiento de los datos: Paralelamente al procesamiento de los datos se trabajó en el almacenamiento de estos mismos. Los datos entregados por los sensores fueron recopilados en un archivo .csv, mientras que el vídeo de la cámara en .avi y finalmente las imágenes o pantallazos de la cámara en .jpg.
- Visualización de los datos: Se diseñó la interfaz en Python, usando el framework QtDesigner como se había definido inicialmente y se emplearon librerías adicionales como geopandas para visualizar el mapa de Bucaramanga y obtener las coordenadas de su geolocalización, opencv para visualizar las imágenes de la cámara y pandas para graficar los datos recibidos de los sensores.

Finalmente, el código de este desarrollo se encuentra en el repositorio [3] y el resultado final se puede visualizar en la Imagen 23 y 24.



Fig. 23. Interfaz Laika sin cámara.



Fig. 24. Interfaz Laika.

IX. MICRO-CONTROLADOR Y PCB'S

1) Objetivos:

- Seleccionar un microcontrolador óptimo en términos de precio, capacidad de procesamiento y espacio. (Que sea adecuado con los requerimientos de los otros sistemas).
- Recopilar e implementar el código en el microcontrolador, para la recepción de datos, el control del despliegue, toma de la fotografía, etc.
- Diseñar una PCB en la cual irán los diferentes instrumentos de medición.
- Soldar los diferentes componentes electrónicos en la PCB.

2) Requerimientos impuestos:

- Dimensiones del Arduino nano: Largo*Ancho: 45mm*18mm.
- Peso del Arduino nano: 7g.
- Área máxima de la PCB: 24 cm²
- Volumen máximo del Arduino nano: 16.2 cm³
- Peso de la PCB: 15g
- Lenguaje de programación: C

3) Metodología: Se realizó una búsqueda en la base de datos de la biblioteca de la Universidad Industrial de Santander, sobre los microcontroladores más usados en otros proyectos CanSant. Se hizo una pre-selección de los más viables y finalmente se seleccionó uno de los microcontroladores en la lista, teniendo en cuenta los requerimientos generales del CanSat que incluyen: Capacidad de procesamiento, menor costo económico, peso muy ligero y mínimo tamaño.

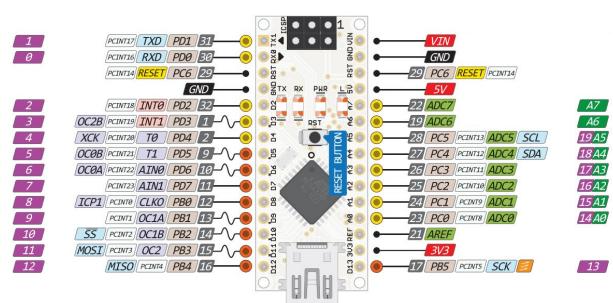


Fig. 25. Arduino Nano

El microcontrolador elegido corresponde al sistema embebido Arduino Nano, el cual cuenta con un voltaje de operación de 5 [V], memoria flash de 32 [kB] de los cuales 2 [kB] son utilizados por bootloader, 8 pines de E/S analógicas, voltaje de entrada de 7 a 12 [V], 22 pines de E/S digitales, 6 salidas PWM, y un consumo de corriente de 19 [mA]. Su tamaño es de 18 [mm] x 45 [mm] y tiene un peso de 7 [g].

Una vez definidos todos los componentes a utilizar (sensores, actuadores, etc.) que se van a usar en el CanSat se realiza el diseño de una PCB, teniendo en cuenta las características en el datasheet de cada uno de los componentes.

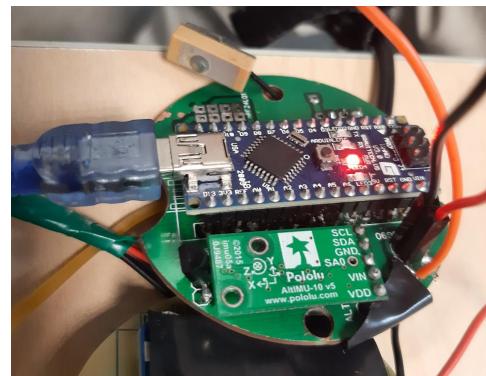


Fig. 28. PCB y componentes

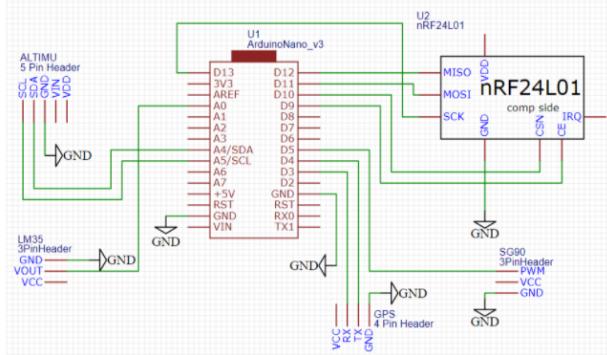


Fig. 26. Esquemático de conexiones al Arduino Nano

Para comenzar a diseñar el circuito impreso, se realiza el diagrama de conexiones entre el microcontrolador y todos los demás componentes electrónicos con ayuda de los pinouts de dichos componentes, adicionalmente de los pines de potencia y demás detalles necesarios para la exitosa conexión articulada de todos los subsistemas.

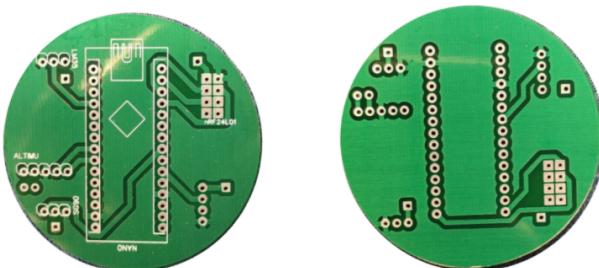


Fig. 27. PCB'S

Para finalizar se realiza el montaje y soldadura de todos los componentes y cables de conexión a la PCB.

Se recopilan los códigos implementados en cada subsistema y se crea un código principal en el software Arduino IDE, lo que permite articular en un solo código todos los subsistemas para garantizar el funcionamiento total del CanSat.

Al comienzo del archivo .ino, se agregan todas las librerías necesarias para poder manipular en código la información que entregan todos los sensores y dispositivos que conforman el CanSat. Luego, se encuentran declaradas todas las variables globales.

TransmisorFinal

```
#include <SPI.h>
#include <Wire.h>
#include <LPS.h>
#include <LSM6.h>
#include <TinyGPSPlus.h>
#include <SoftwareSerial.h>
#include <nRF24L01.h>
#include <RF24.h>
//Motor
#include <Servo.h>
```

Fig. 29. Librerías de los sensores, cámara, antenas y servo

```
//Para comunicaciones:
const byte identificacion[6] = "00001"; //Mismo código en el RX y TX
RF24 radio(9, 10); // asigna pines para CSN y CE
```

```
//Para sensorica:
LPS ps;
LSM6 imu;
TinyGPSPlus gps;
char report[80];
static const int RXPin = 4, TXPin = 3;
static const uint32_t GPSBaud = 9600;
SoftwareSerial ss(RXPin, TXPin);

//DECLARACIÓN SERVO
Servo myservo; // crea el objeto servo
float AltitudRef=930;
```

Fig. 30. Variables Globales

En el setup se definen las acciones que se van a ejecutar al encender el CanSat, como conectar, detectar e inicializar, todos los componentes a su respectivo pin del microcontrolador Arduino Nano.

```
void setup()
{
    myservo.attach(5);
    myservo.write(50);
    //myservo.write(180);
    //pinMode(5, OUTPUT);
    Serial.begin(9600);
    ss.begin(GFSBaud);
    Wire.begin();
    if (!imu.init())
    {
        Serial.println("Failed to detect and initialize IMU!");
        while (1);
    }
    imu.enableDefault();

    if (!ps.init()) //PB
    {
        Serial.println("Failed to autodetect pressure sensor!");
        while (1);
    }
    ps.enableDefault();
    radio.begin(); //Comienza la operación del chip
    radio.setAutoAck(false); //Auto-acknowledgement responds
}
```

Fig. 31. Inicio del Setup

```
/**CONFIGURACIÓN**/
radio.setaDataRate(RF24_2MBPS); // configura 250KBPS, 1BMP, 2MBPS
radio.openWritingPipe(identificación); // Open a pipe for writing with a specific address
radio.setPALevel(RF24_PA_MIN); // Set transmit power
radio.setChannel(127); // Set the channel to 2 (2527 MHz)
radio.stopListening(); // Stop listening for incoming messages, and switch to transmit mode.
// digitalWrite(S,HIGH);
//
```

Fig. 32. Final del Setup

En el loop se incluyen todas las funciones a realizar durante el vuelo del CanSat, como la recolección constante de datos de temperatura, altitud, presión, ubicación GPS, etc; a demás de ir transmitiendo dichos datos constante y la ejecución adecuada del desacople del CanSat con el mecanismo del ServoMotor.

```
void loop()
{
    vector[0] = analogRead(A0); // Variable para almacenar el valor obtenido del sensor (0 a 1023)
    vector[0] = ((vector[0] * 100)/1023)-10;//conversión a celsius
    imu.read();
    float pressure = ps.readPressureMillibars();
    vector[1] = pressure*100;
    vector[2] = ps.pressureToAltitudeMeters(pressure);
    float imax=imu.a.x;
    vector[3]=imax*0.061*0.001;
    float imay=imu.a.y;
    vector[4]=imay*0.061*0.001;
    float imaz=imu.a.z;
    vector[5]=imaz*0.061*0.001;
    float imgx=imu.g.x;
    vector[6]=imgx*0.061*0.001;
    float imgy=imu.g.y;
    vector[7]=imgy*0.061*0.001;
    float imgz=imu.g.z;
    vector[8]=imgz*0.061*0.001;
    String lati_s;
    String longi_s;
```

Fig. 33. Recolección de los datos tomados por la sensórica

```
// GPS NEO 6M
while (ss.available() > 0)
    gps.encode(ss.read());
if (millis() > 5000 && gps.charsProcessed() < 10)
{
    Serial.println(F("No GPS detected: check wiring."));
    while(true);
}
if (gps.location.isValid())
{
    vector[9] = gps.location.lat();
    lati_s = String(vector[9], 2);
    vector[10] = gps.location.lng();
    longi_s = String(vector[10], 2);
}
else
{
    lati_s = "0.0";
    longi_s = "0.0";
}
```

Fig. 34. Detección del GPS y recolección de sus datos

```
//Despliegue de paracaídas
float AltitudActual = vector[2];

if( AltitudActual - AltitudRef >= 100 )
{
    Serial.println("entro altura");
    myservo.write(50);
    delay(500);
    myservo.write(180);
}
```

Fig. 35. Condición de despliegue y movimiento del servo

```
String temp_s, presion_s, altura_s, imax_s, imaz_s, imay_s, imgx_s, imgy_s, imgz_s, paquetel, paquete2, paquete3;
char trama1[32], trama2[32], trama3[32];
temp_s = String(vector[0], 2);
presion_s = String(vector[1], 2);
altura_s = String(vector[2], 2);
imax_s = String(vector[3], 1);
imaz_s = String(vector[4], 1);
imay_s = String(vector[5], 1);
imgx_s = String(vector[6], 1);
imgy_s = String(vector[7], 1);
imgz_s = String(vector[8], 1);
paquetel = temp_s + " " + presion_s + " " + altura_s + " " + lati_s + " " + longi_s + " ";
paquetel += imax_s + " " + imaz_s + " " + imay_s + " " + imgx_s + " " + imgy_s + " " + imgz_s + " ";
paquete2 = imax_s + " " + imaz_s + " " + imay_s + " " + imgx_s + " " + imgy_s + " " + imgz_s + " ";
paquetel.toCharArray(trama1, 32);
paquete2.toCharArray(trama2, 32);
paquete3.toCharArray(trama3, 32);
//Serial.println(paquetel);
//Serial.println(paquete2);
//Serial.println(paquete3);
Serial.println(paquetel);
//Serial.println(trama1);
radio.write(trama1, sizeof(trama1));
radio.write(trama2, sizeof(trama2));
radio.write(trama3, sizeof(trama3));
delay(1000);
bool result = radio.isChipConnected();
Serial.println(result);
```

Fig. 36. Transmisión de los datos de la sensorica

X. RESULTADOS

El proceso de implementación del proyecto tuvo sus impedimentos e inconveniencias, la adquisición de algunos de los componentes fue demorada y esto tuvo una repercusión negativa en los tiempos determinados para realizar el montaje. El trabajo realizado por cada uno de los subgrupos fue bastante satisfactorio, esto debido a que en las fases iniciales del proyecto se estableció muy bien la parte de requerimientos de manera que todos los subgrupos tuviesen un trabajo en coherencia con el de los demás, nuevamente, lo único negativo de haber invertido mucho tiempo en esas fases iniciales es que no dejó tiempo suficiente para la implementación; Es por esta razón que el Cansat se alcanzó a terminar, pero de todos los objetivos previamente establecidos no se alcanzó a integrar la cámara por un problema existente

con la alimentación de esta misma.

Adicional a este Cansat, se entrega el repositorio en Github del proyecto en donde se encuentra el código utilizado en las diferentes etapas [1]

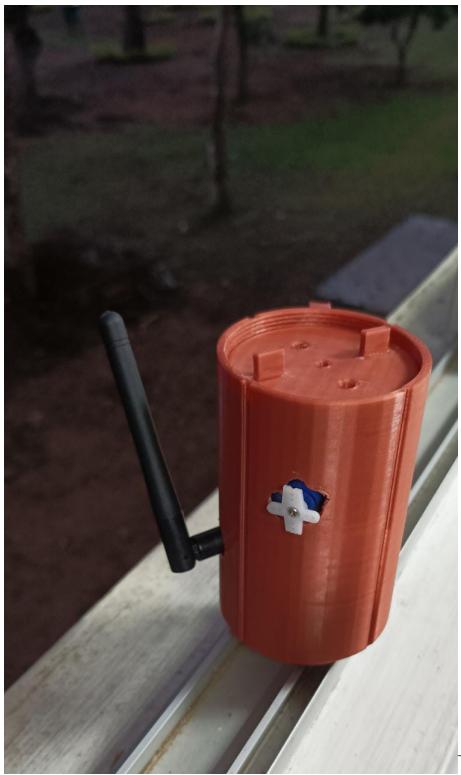


Fig. 37. Cansat ensamblado

REFERENCES

- [1] Cansat Laika. *Repositorio del Código General*. 2021.
URL: <https://github.com/cansatlaika?tab=repositories> (visited on 03/16/2022).
- [2] Cansat Laika. *Repositorio del Código Sensores*. 2021.
URL: <https://github.com/Instrumentacio/codigos-sensores.git> (visited on 03/16/2022).
- [3] Cansat Laika. *Repositorio del Código Telemetria*. 2021.
URL: <https://github.com/DaniSTexe/Telemetry> (visited on 03/16/2022).