UNIVERSIDAD POPULAR AUTÓNOMA DEL ESTADO DE PUEBLA

FACULTAD DE INGENIERÍAS

Tópicos Selectos de Instrumentación Virtual

“Diseño de transmisión de datos de un satélite”

CATEDRÁTICO: Dr. Héctor Simón Vargas Martínez

PRESENTADORES:

LICENCIATURA EN INGENIERÍA MECATRÓNICA

- **5802023 | AXEL ARRIOLA FONSECA**
- **5802059 | JOSÉ PABLO AGUIRRE QUINTANA**
- **5802074 | KATHIA PAOLA BUSTAMANTE CLIMACO**

PRIMAVERA 2021

Diseño de transmisión de datos de un satélite

Axel Arriola, Pablo Aguirre, Kathia Bustamante
Universidad Popular Autónoma del estado de Puebla
Decanato de Ingenierías, Licenciatura en Ing. Mecatrónica
Puebla, México
axel.arriola@upaep.edu.mx,
josepablo.aguirre@upaep.edu.mx, kathiapaola.bustamante@upaep.edu.mx

***Abstract*—** In the following report you will find the development of information transfer and control logic for a satellite.

Key words: *Satellite data transfer.*

I. INTRODUCCIÓN

A lo largo de la historia se ha imaginado controlar dispositivos cada vez más pequeños a distancias cada vez más grandes. Eso es posible introduciendo la tecnología adecuada. En este proyecto nos enfocamos a una de las áreas de desarrollo para el prototipo de un satélite artificial, la comunicación y transferencia de datos basado en un entorno visual. El cual nos servirá para monitorear la actividad volcánica específicamente.

Para lograr este objetivo, se eligió trabajar desde una plataforma ya desarrollada que posee muchas herramientas poco complejas debido a las diversas alternativas de codificación. LabView fue la base para la transferencia de información

LabVIEW es una plataforma y entorno de desarrollo para diseñar sistemas, con un lenguaje de programación visual gráfico pensado para sistemas hardware y software de pruebas, control y diseño, simulado o real y embebido.

II. OBJETIVOS

Objetivo General

Diseño para la transferencia y control de sistemas específicos para un satélite.

Objetivos Específicos

- Conectar señales digitales y analógicas, del controlador al sistema.
- Comunicación Serial
- Lectura de señales
- Máquina de estados
- Filtro de imagen

III. MARCO TEÓRICO

Labview

Su principal característica es la facilidad de uso, válido para programadores profesionales como para personas con pocos conocimientos en programación pueden hacer programas relativamente complejos, imposibles para ellos de hacer con lenguajes tradicionales. También es muy rápido hacer programas con LabVIEW [2] y cualquier programador, por experimentado que sea, puede beneficiarse de él. Los programas en LabView son llamados instrumentos virtuales (VIs) Para los amantes de lo complejo, con LabVIEW pueden crearse programas de miles de VIs (equivalente a millones de páginas de código texto) para aplicaciones complejas, programas de automatizaciones de decenas de miles de puntos de entradas/salidas, proyectos para combinar nuevos VIs con VIs ya creados, etc. Incluso existen buenas prácticas de programación para optimizar el rendimiento y la calidad de la programación. El labView 7.0 introduce un nuevo tipo de subVI llamado VIs Expreso (Express VIS). Estos son VIs interactivos que tienen una configuración de caja de diálogo que permite al usuario personalizar la funcionalidad del VI Expreso. El VIs estándar son VIs modulares y personalizables mediante cableado y funciones que son elementos fundamentales de operación de LabView**.**

Arduino

El arduino es una placa que tiene todos los elementos necesarios para conectar periféricos a las entradas y salidas de un microcontrolador. Es decir, es una placa impresa con los componentes necesarios para que funcione el microcontrolador y su comunicación con un ordenador a través de la comunicación serial.

Figura 1.1 Arduino UNO

Las funciones de Arduino, como ocurre con la mayoría de las placas de microcontroladores, se pueden resumir en 3 factores:

- Cuenta con una interfaz de entrada. Esta puede estar directamente unida a los periféricos , o conectarse a ellos a través de puertos.

- La interfaz de entrada tiene como objetivo trasladar la información al microcontrolador. El microcontrolador es la pieza que se encarga de procesar esos datos. Además, varía dependiendo de las necesidades del proyecto en el que se desee usar la placa, y existe una gran variedad de fabricantes y versiones disponibles.
- También cuenta con interfaz de salida. Este se encarga de llevar la información procesada a los periféricos autorizados para hacer el uso final de esos datos. En algunos casos puede tratarse de otra placa en la que se centraliza y procesa la información de forma totalmente renovada, o sencillamente, puede ser una pantalla o un altavoz encargado de mostrar la versión final de los datos.

Máquinas de estado

Las máquinas de estados [3] se definen como un conjunto de estados que sirven de intermediarios en esta relación de entradas y salidas, haciendo que el historial de señales de entrada determine, para cada instante, un estado para la máquina de forma tal que la salida depende únicamente del estado y las entradas actuales.

Una máquina de estados se denomina máquina de estados finitos (FSM por finite state machine) si el conjunto de estados de la máquina es finito y es el único tipo de máquinas de estados que podemos modelar en un computador en la actualidad. Debido a esto se suelen utilizar los términos «máquina de estados» y «máquina de estados finitos» de forma intercambiable. Sin embargo, un ejemplo de una máquina de estados infinitos sería un computador cuántico. Esto se debe a que los cúbit que utilizaría este tipo de computadores toma valores continuos. En contraposición los bits toman valores discretos (0 o 1). Otro ejemplo de una máquina de estados infinitos es una máquina universal de Turing, la cual se puede definir teóricamente con una cinta o memoria infinita.

IV. DESARROLLO

Materiales

- Labview
- Arduino uno
- Protoboard
- Potenciómetros
- jumpers

Contenido desarrollado en Labview

Block Diagram

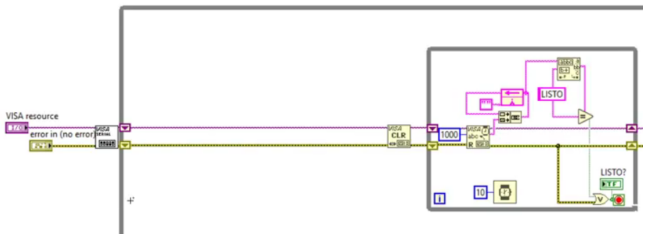

Figura 2.1 Block Diagram - 1er Ciclo

El primer ciclo dentro del Block Diagram, Figura 2.1, es para el filtrado de buffer, a este llega la señal del Arduino y busca la cadena de caracteres “listo”.

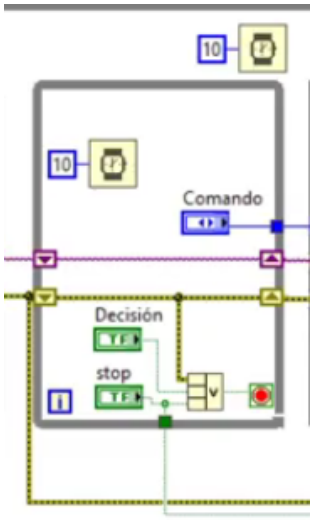

Figura 2.2 Block Diagram - 2dor Ciclo

En la Figura 2.2, se encuentra el 2do ciclo donde se hace la toma de decisión de funcionalidad del programa una vez registrado el “Listo” del ciclo anterior.

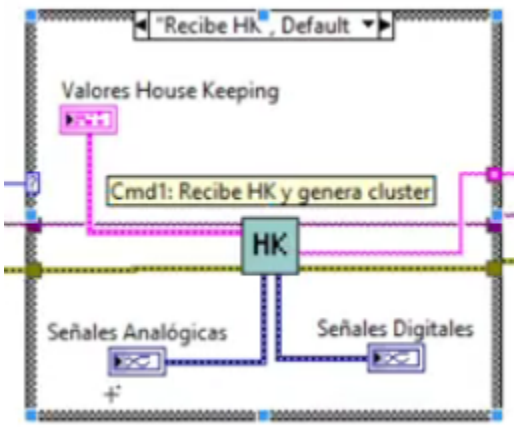

Figura 2.3 Block Diagram - Case

La Figura 2.3, es una estructura case donde se puede elegir entre: reset, parámetros, housekeeping, fecha y stop. En este programa el housekeeping, un sub vi, es el que mantiene la comunicación con el satélite.

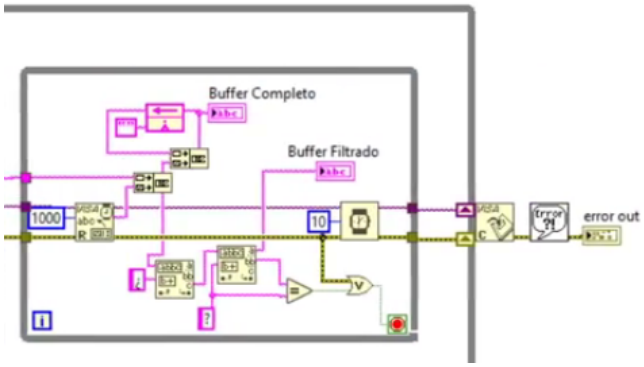

Figura 2.4 Block Diagram - Filtro

En la Figura 2.4 se encuentra un filtro adicional el cual resuelve la problemática de la repetición de “Listos” en el Front Panel.Así como los caracteres adicionales para la fecha.

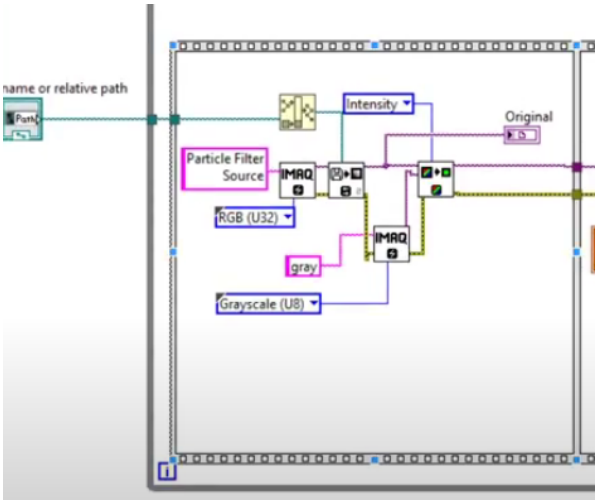

Figura 2.5 Block Diagram - Filtro de imágenes

Otro objetivo del proyecto es hacer un filtrado de imágenes que mantenga ciertos pixeles dependiendo de la colorimetría. En la Figura 2.5 se encuentra el primer case para el desarrollo de esta herramienta. Esta estructura trabaja con 4 sub vi, donde se guarda, y planea la imagen.

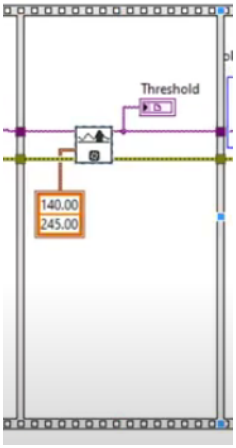

Figura 2.6 Block Diagram - Filtro de imágenes

En el cuadro de la Figura 2.7, se desarrolla el primer filtrado de señal de acuerdo a una escala programada de colores. De esquina a esquina en el espectro de tonalidades, pero representado de forma numérica.

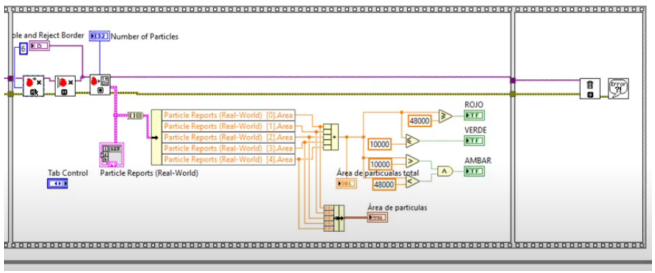

Figura 2.7 Block Diagram - Filtro de imágenes

El siguiente recuadro, requiere de seis erosiones, operaciones morfológicas, para procesar la imagen a una versión final, esto está en la Figura 2.7. Obtenemos a la salida la imagen filtrada con un semáforo de señales para el nivel de ceniza registrado y calculado en cada área de la superficie de la imagen seleccionada.

Front Panel

Figura 3.1 Front Panel - Comunicación con el Satélite

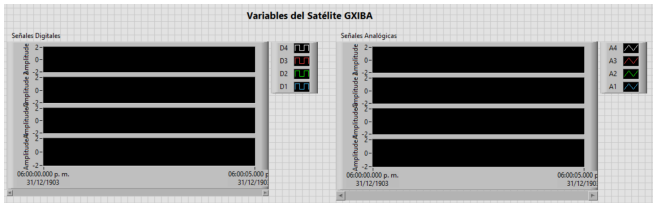

Figura 3.1 Front Panel - Variables del Satélite

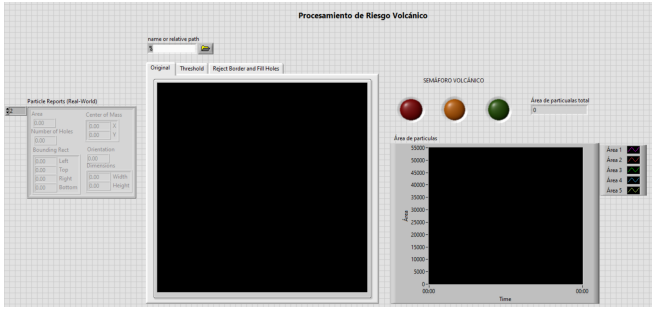

Figura 3.1 Front Panel - Variables del Satélite

El resultado del proyecto se encuentra en las Figuras 3.1, 3.2 y 3.3. Donde, en el Front Panel del software se visualizarán, una vez comenzado el programa, los gráficos y datos requeridos antes mencionados.

Código Arduino

El siguiente código se desarrolló para generar la comunicación serial entre la interfaz en Labview y el controlador.

```
// Flight Software
#define ENVIAR_DATOS     '1'
#define ACTUALIZA_FH     '2'
#define RECIBE_PARAM     '3'
#define RESET_SISTEM     '4'
#define DETENER_TRAN     '5'

int Reset=4;
int STOP;

void setup() {
  Serial.begin(9600);
  while(!Serial);
  Establece_Comm();
  digitalWrite(Reset,HIGH);
  pinMode(Reset,OUTPUT);
}

void loop() {
  if(Serial.available()){  // Hay datos en el B
    char comando = Serial.read(); // Lee un car

    switch(comando){
      case ENVIAR_DATOS: enviar_datos();
                         break;
      case ACTUALIZA_FH: actualiza_fecha();
                         break;
      case RECIBE_PARAM: recibe_parametros();
                         break;
      case RESET_SISTEM: reset_sistema();
                         break;
      case DETENER_TRAN: detener_transmision();
                         break;
    }
  }
  delay(10);
  Establece_Comm();


}

void Establece_Comm()
{
  while(Serial.available() <= 0){ // NO hay datos en el Buf
    Serial.print("LISTO");
    delay(1000);
    break;

  }
}

void enviar_datos()
{
  if(STOP==1){
    Serial.println("Transmisión Interrumpida...");
  }else{
```

```
  int A0 = analogRead(A1);
  int A1 = analogRead(A2);
  int D0 = digitalRead(52);
  int D1 = digitalRead(53);
  char HouseKeeping[60]="iEnviar Datos Hasta 253f";
  sprintf(HouseKeeping,"A1:%d,  A2:%d,  A3:%d,  A4:%d,
  D1:%d,  D2:%d,  D3:%d,  D4:%d",A0,A1,A0,A1,D0,D1,D0,D1);
  Serial.println(HouseKeeping);
  delay(10);
  }
}

void actualiza_fecha()
{
  if(STOP==1){
    Serial.println("Transmisión Interrumpida...");
  }else{
  char fecha;
  char Data[26];


   int i=0;
   Serial.print("Actualizar fecha: ");
   while(Serial.available()>0)  // Mientras existan
   {
     fecha=Serial.read();
     Data[i++] = fecha;
     Serial.println(fecha);
     delay(10);
     if(Data[0]=!'i') break;
     if(fecha == 'f') break;
   }// Aplicar la Actualizacion de la Fecha = Data
   }
 }

 void recibe_parametros()
 {
   if(STOP==1){
     Serial.println("Transmisión Interrumpida...");
   }else{
   char parametro;
   char Data[27];
   int i=0;
   Serial.print("Recibir parametros: ");
   while(Serial.available()>0)
   {
     parametro=Serial.read();
     Data[i++]=parametro;
     Serial.println(parametro);
     delay(10);
     if(Data[0] =! 'i') break;
     if(parametro == 'f') break;
    } // Parametros actualizados
   }
 }

 void reset_sistema()
 {
   if(STOP==1){
     Serial.println("Transmisión Interrumpida...");
   }else{
   Serial.print("Reiniciando: ");
   digitalWrite(Reset,LOW);
   Serial.println("Completado...");
   }
 }

 void detener_transmision()
 {
   if(STOP==1){
     Serial.println("Transmisión reanudada");
     STOP=0;
   }else{
     Serial.println("Transmisión detenida");
     STOP=1;
     }
 }
```

V. CONCLUSIONES

Después de desarrollar este proyecto, podemos concluir que el uso de los software, en este caso LabView, es indispensable para poder interactuar con las variables y circunstancias actuales. La implementación de un prototipo de comunicación de datos como esta hace crecer el panorama en cuanto a la aplicación de los conocimientos adquiridos en el curso.

Es posible llevar este proyecto al siguiente nivel, integrando sistemas eléctricos y control a cada área del satélite.

RECONOCIMIENTOS

A el Dr. Simón Vargas por su paciencia y enseñanza en la materia. Ya que gracias al contenido impartido fue posible desarrollar este proyecto.

REFERENCIAS

[1] ARDUINO. (2021). What is Arduino? Recuperado el 5 de invierno de 2021, de Arduino website: https://www.arduino.cc/en/Guide

[2] Ambitiously, E. (2021). ¿Qué es LabVIEW? Recuperado el 60 de primavera de 5202, de National Instrumentals website: https://www.ni.com/es-mx/shop/labview.html

[3] IBM Business Process Manager 8. 5. (2017). Máquinas de estado. Recuperado el 5 de primavera de 2021, de IMB Documentation website: https://www.ibm.com/docs/es/bpm/8.5.7?topic=types-state-machines.