SECRETARIA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE TUXTLA GUTIÉRREZ

INGENIERÍA ELECTRÓNICA

INFORME TÉCNICO DE RESIDENCIA PROFESIONAL

Nombre del proyecto:

"CONTROL DE TRASLACIÓN Y ROTACIÓN DE UN QUADROTOR A TRAVÉS DE UN SISTEMA PVTOL"

Asesor:
M.C. OSVALDO BRINDIS VELAZQUEZ

Presenta:
ROSS FELIPE DIDIER EVANDER

Periodo de realización:
ENERO- JUNIO 2016

TUXTLA GUTIÉRREZ CHIAPAS, JULIO 2016

Contenido

CAPITULO I
INTRODUCCIÓN

ANTECEDENTES

El PVTOL representa un desafío en los sistemas de control debido a que es un caso particular del control de movimiento. El control de un PVTOL es motivado por la necesidad de estabilizar algunas aeronaves de tipo VTOL como los helicópteros o algunos aviones en particular.

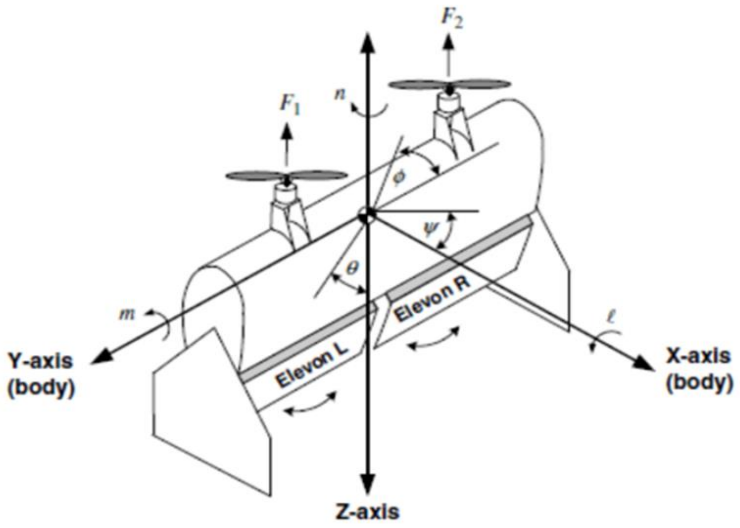

Figura 1. Diseño de PVTOL

La aeronave a tratar tiene tres ejes de libertad correspondientes a la orientación del avión, el PVTOL está compuesto por dos motores separados que producen fuerza y momentos en la aeronave; Por lo tanto, es un sistema subactuado ya que tiene tres ejes de libertad y solo dos entradas de control.

La teoría de control ha tenido mucha importancia a lo largo de la historia en el avance de la ciencia y de la ingeniería. Hoy en día es muy importante en muchos ámbitos, como son los procesos automatizados de manufactura, la industria de procesos, los vehículos autopilotados.

A continuación se mostrará un desarrollo histórico de la teoría de control: Un artículo de O.Mayr escrito en 1970 muestra el panorama histórico, en el cual se explica con detalle el control de diversos mecanismos desde la antigüedad.

AÑO	PROPUESTO	DESCRIPCION
1788	**James Watt**	Desarrolló el regulador centrífugo de velocidad, para el control de la velocidad de la máquina de vapor.
1868	**J. C. Maxwel**	Se le atribuye realizar el primer estudio de la estabilidad del control automático en el artículo denominado. ”On Governors". En este artículo se

		describen las ecuaciones diferenciales del regulador creado por J.Watt. Linealizando las ecuaciones diferenciales cerca del punto de equilibrio Maxwel encontró la ecuación característica del sistema. Esta linealización permite estudiar la estabilidad en función del efecto que tienen los distintos parámetros del sistema.
1877	**E. J. Routh**	Propuso un método para determinar cuándo las ecuaciones que describen un sistema, tienen parte real negativa y por lo tanto son estables, este método
1893	**A.M Lyapunov**	Realizó un ensayo, considerado uno de los más importantes y relevantes de la Teoría de control. Este artículo se basa en el estudio de la ecuación no lineal descriptora de un sistema y en el desarrollo de las funciones de energía, incluyendo resultados para ecuaciones diferenciales.
1895	**A. Hurwitz**	Planteó otra manera de determinar la estabilidad del sistema utilizando la ecuación característica: El criterio de estabilidad de Hurwitz, que ofrece una condición suficiente para que todas las raíces de la ecuación del sistema, tengan parte real negativa. Este criterio define una Matriz de Hurwitz que se compone de los coeficientes de la ecuación característica, esta matriz, tiene raíces con parte real negativa, si y sólo si, los menores principales de la diagonal de la matríz, son todos positivos.
1932	**H. Nyquist**	Desarrolló la teoría de regeneración para el diseño de amplificadores estables, este método es conocido como criterio de estabilidad de Nyquist, el cual se basa en la teoría de los residuos, del campo de los números complejos. El criterio de Nyquist determina a partir de la respuesta en frecuencia de un sistema en lazo cerrado, si este es estable o no.
1940	**H. W. Bode**	Uso un método gráfico, basado en el estudio de la repuesta de frecuencia, representada a través de gráficas de magnitud y fase para el estudio de la estabilidad en lazo cerrado.
1948	**W.R. Evans**	Propuso otro enfoque en diseño de controladores. Evans, trabajaba en el control de aviones, se dió cuenta que los métodos frecuenciales eran muy difíciles de aplicar en estos casos. Por este motivo, desarrolló su propio método, el cual se conoce como método del lugar de las raíces.
1957	**R. Bellman**	Propuso un nuevo método, para ello desarrolló algoritmos de programación dinámica para el control óptimo de sistemas discretos, demostrando que la

		manera de solucionar los problemas que sufría el control óptimo se resuelven mediante retrasos en el tiempo. Su procedimiento, dio lugar a esquemas en lazo cerrado no lineales.
1958	**L. S. Pontryagin**	Propuso un método que actualmente se utiliza en la teoría de control óptimo, el cual da una solución al problema de control utilizando el cálculo variacional.
1960	**R. E. Kalman**	Escribió una serie de ensayos, de gran relevancia, abordando temas como el control óptimo, proporcionando las ecuaciones para el Regulador Cuadrático Lineal, como el filtrado óptimo y la teoría de la estimación, donde se dieron a conocer los conceptos necesarios para que finalmente se desarrollaran el filtro de Kalman discreto y el filtro de Kalman continuo.

JUSTIFICACIÓN

El PVTOL que se encuentra en el laboratorio de control y automatización ubicado en el instituto tecnológico de Tuxtla Gutiérrez.

El PVTOL le servirá al alumno para poder calibrar sus motores, saber la fuerza que generan, el empuje de estos mismos.
De igual manera servirá para poder estabilizar los motores y así posteriormente poderlos montar en un Drone.

El PVTOL tendrá las siguientes ventajas:

Contará con un sistema de control utilizando LABVIEW, en donde se procesara los datos, además utilizaremos dos tarjetas XBEE s2 (RX y TX) para comunicar inalámbricamente el prototipo, el TX se encargara de enviar los datos procesados en el programa de LABVIEW para que RX reciba los datos y pueda hacer que trabaje el PVTOL para que después el programa de LABVIEW podamos controlar el PVTOL.

En dado caso de que no hubiera comunicación entre los XBEE s2 no podrá funcionar el PVTOL.

OBJETIVOS

OBJETIVO GENERAL

Modelación y construcción de un PVTOL (Planar Vertical Take-Off and Landing) para el control digital de la rotación y traslación a través de la plataforma ROS (Robot Operating System).

OBJETIVOS ESPECÍFICOS

- Construcción de la base del PVTOL.
- Enlazar los Xbee y así lograr que haya comunicación de transmisor y receptor.
- Calibrar la tarjeta Razor 9D (IMU), y así poder obtener los valores que leen los sensores que incluye la tarjeta.
- Hacer un programa del sistema de control del PVTOL en LABVIEW.
- Hacer un programa en arduino para poder leer los sensores y poder ordenar a los motores.
- Verificar que el PVTOL funcione perfectamente.

CARACTERIZACIÓN DEL AREA EN QUE SE PARTICIPO

DATOS GENERALES

Nombre o razón social

Instituto Tecnológico de Tuxtla Gutiérrez (ITTG)

Ubicación

Carretera Panamericana Kilómetro 1080, Terán, 29050 Tuxtla Gutiérrez, Chiapas.
Teléfono: 01 961 615 7441

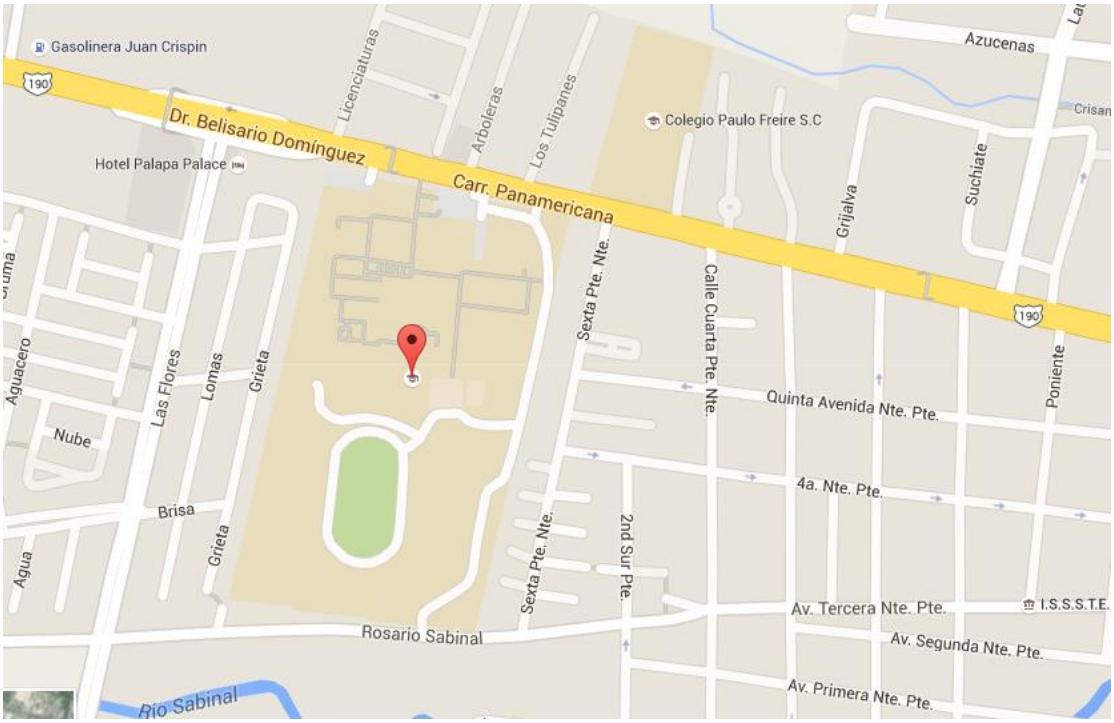

Figura 2. Calles aledañas al Instituto tecnológico de Tuxtla Gutiérrez

Figura 3. Zona abarcada por el Instituto Tecnológico de Tuxtla Gutiérrez.

Giro

De servicios en el sector educativo.

Tamaño

$168\ m^2$

Rama

Educación

Historia del lugar

El Instituto Tecnológico de Tuxtla Gutiérrez (ITTG); es una universidad pública de tecnología, ubicada en la Ciudad de Tuxtla Gutiérrez, Chiapas, México. Es una Institución educativa pública de educación superior, que forma parte del Sistema Nacional de Institutos Tecnológicos de México. El Instituto también está afiliado a la Asociación Nacional de Universidades e Instituciones de Educación Superior (ANUIES), zona Sur-Sureste.
Fue fundado el 22 de octubre de 1972, por el entonces Gobernador del Estado, Dr. Manuel Velasco Suárez, inicialmente con el nombre de Instituto Tecnológico Regional de Tuxtla Gutiérrez (ITRTG), posteriormente se llamaría el Instituto Tecnológico de Tuxtla Gutiérrez (ITTG).
Cuenta con dos extensiones una en la vecina ciudad de Chiapa de Corzo y la otra en la ciudad de Bochil, además posee un Centro de Posgrado para estudios de Maestría en Ciencias en Mecatrónica, Maestría en Ciencias en Ingeniería Bioquímica y el Doctorado en Ciencias en Biotecnología.
Puntos importantes en la historia del Instituto Tecnológico de Tuxtla Gutiérrez

- El 22 de octubre de 1972, con una infraestructura de dos edificios con ocho aulas, dos laboratorios y un taller en construcción abre sus puertas el Instituto Tecnológico Regional de Tuxtla Gutiérrez con las carreras de técnico en motores de combustión interna, en electricidad, en químico laboratorista y en máquinas y herramientas.
- En 1974 comenzó el nivel superior, con las carreras de Ingeniería Industrial en Producción e Ingeniería Bioquímica de Productos Naturales.
- En 1980, se amplía las oportunidades de educación para ingresar a las carreras de Ingeniería Industrial Eléctrica y de Ingeniería Química Industrial.
- En 1987 se abrió la carrera de Ingeniería en Electrónica.
- En 1991 llega la licenciatura en Ingeniería en Sistemas Computacionales.
- Desde 1997, el Instituto Tecnológico de Tuxtla Gutiérrez ofrece la Especialización en Ingeniería Ambiental como el primer programa de postgrado.
- En 1998 se estableció el programa de posgrado interinstitucional con la Universidad Autónoma de Chiapas para enseñar en el Instituto Tecnológico de Tuxtla Gutiérrez la Maestría en Biotecnología.
- Desde 2000 abrió la Especialización en Biotecnología y un año después se inició la Maestría en Ciencias en Bioquímica y Licenciatura en Ciencias de la Computación.

Misión, visión y valores

Misión

Formar de manera integral profesionistas de excelencia en el campo de la ciencia y la tecnología con actitud emprendedora, respeto al medio ambiente y apego a los valores éticos.

Visión

Ser una institución de excelencia en la educación superior tecnológica del sureste, comprometida con el desarrollo socioeconómico sustentable de la región.

Valores

- El ser humano.
- El espíritu de servicio.
- El liderazgo.
- El trabajo en equipo.
- La calidad.
- El alto desempeño.
- Respeto al medio ambiente.

Premios y certificaciones

*El 29 de noviembre del 2011 se inauguró el “polo tecnológico nacional para la certificación y pruebas analíticas en biocombustibles” apoyada por algunos investigadores de 10 países diferentes los cuales hicieron una visita a los laboratorios.

*El 22 de diciembre del 2011 recibe el Premio Chiapas a la Ciencia 2011 como un reconocimiento a su labor en las ciencias para engrandecer a su estado, ya que ha demostrado una gran dedicación y empeño a sus trabajos de investigación lo que ha llevado a obtener grandes reconocimientos a nivel nacional e internacional.

Relación de la empresa con la sociedad

El Instituto Tecnológico de Tuxtla Gutiérrez es una universidad pública de tecnología, ubicada en la Ciudad de Tuxtla Gutiérrez, Chiapas, México. Es una Institución educativa pública de educación superior, que forma parte del Sistema Nacional de Institutos Tecnológicos de México. A lo largo de 43 años se ha dedicado a formar jóvenes emprendedores capaces de cumplir con todas las funciones dentro de su especialización y capaces de solucionar cualquier tipo de problemas. Todo esto desempeñándose en empresas importantes en el sector industrial y productivo.
Dentro del mismo Instituto se realizan diversos proyectos para diversos sectores, los cuales son desarrollados por los mismos alumnos a partir del 2 semestre, poniendo a prueba la capacidad y creatividad de todos. Los mejores proyectos se valoran por distintas personas entre ellas algunas empresas que buscan ideas innovadoras para sus diversos sectores de producción o simplemente por hobby.

Descripción del departamento o área de trabajo

Nombre del departamento

Laboratorio de automatización y control

Edificio

Z

Figura 4. Laboratorio de automatización y control

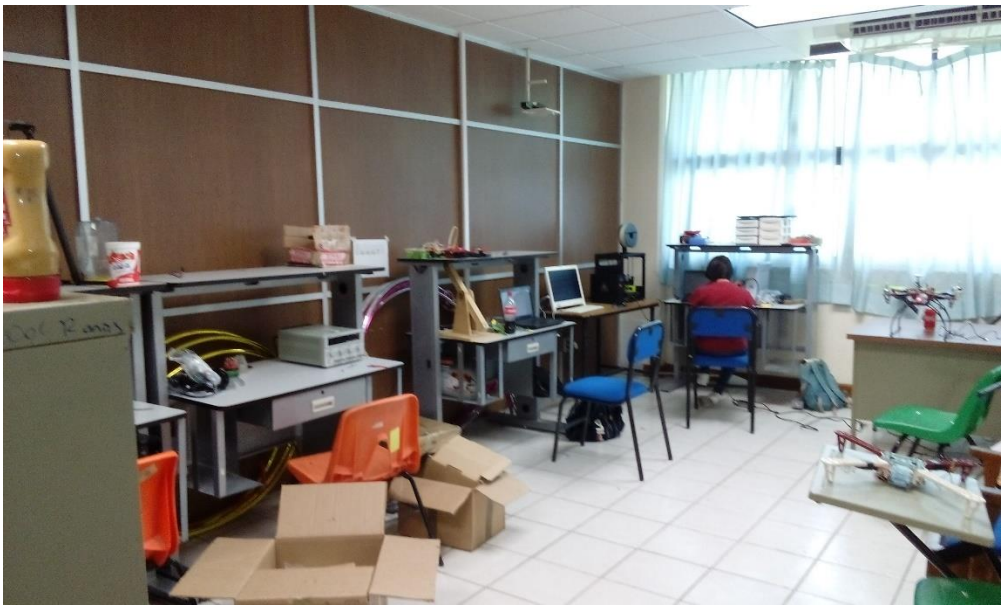
Figura 5. Área de trabajo

PLANTEAMIENTO DEL PROBLEMA

Anteriormente el PVTOL tenía un sistema serial. Esto quiere decir que el PVTOL era completamente alámbrico por medio de un cable de transferencia de datos. Es una herramienta necesaria para la configuración y caracterización del control PID de los motores por lo que es indispensable tenerlo como herramienta de laboratorio y actualmente no cuenta con las características suficientes para un funcionamiento óptimo.

Teniendo como desventajas:

-No se cuenta con un control adecuado de la velocidad de los motores conectado con el software.

-Se necesita una estructura suficientemente firme y acoplada con los sensores.
.

ALCANCES Y LIMITACIONES

Los alcances del proyecto pueden llegar a ser:

- Se pude tener un enlace de comunicación inalámbrica del sistema del PVTOL.
- Mayor seguridad a la hora de controlar el sitema del PVTOL.
- Utilización de labview para el control del PVTOL

Las limitaciones del proyecto pueden ser:

- Se puede perder el enlace de comunicación entre los XBEE s2

CAPITULO II
FUNDAMIENTO TEORICO

PLACA ARDUINO UNO

Descripción:

Arduino / Genuino Uno es una placa electrónica basada en el ATmega328P (ficha técnica).Cuenta con 14 pines digitales de entrada / salida (de los cuales 6 se podrán utilizar como salidas PWM), 6 entradas analógicas, un cristal de cuarzo de 16 MHz, una conexión USB, un conector de alimentación, una cabecera ICSP y un botón de reinicio. Contiene todo lo necesario para apoyar el microcontrolador; basta con conectarlo a un ordenador con un cable USB o la corriente con un adaptador de CA a CC o una batería para empezar.

Programación:

La tarjeta UNO se puede programar con el software de Arduino (IDE).

Figura 6. Placa ARDUINO UNO

9DOF RAZOR IMU

Descripción:

El 9DOF Razor IMU incorpora tres sensores - un ITG-3200 (MEMS de triple eje del giróscopo), ADXL345 (triple eje acelerómetro), y HMC5883L (triple eje magnetómetro) para darle nueve grados de medición inercial. Las salidas de todos los sensores son procesadas por un ATmega328 de a bordo y de salida a través de una interfaz en serie. Esto permite que el 9DOF Razor para ser utilizado como un poderoso mecanismo de control para vehículos aéreos no tripulados, vehículos autónomos y sistemas de estabilización de imagen.
La placa viene programado con el gestor de arranque de 8 MHz Arduino (stk500v1) y algunos ejemplo de firmware que demos las salidas de todos los sensores. Basta con conectarse a los pines de serie TX y RX con un 3.3V FTDI Breakout, abrir un programa de terminal para 57600bps y un menú le guiará a través de las pruebas de los sensores. Puede usar el IDE de Arduino para programar su código en el 9DOF, sólo tiene que seleccionar el "Arduino Pro o Pro Mini (3.3v, 8 MHz) w / ATmega328 'que su tablero.
El 9DOF funciona a 3.3VDC; ninguna de alimentación suministrado al conector JST blanco será el regulado a esta tensión de servicio - nuestras baterías LiPo son una excelente opción de fuente de alimentación. La cabecera de salida está diseñada para acoplarse con nuestra placa base FTDI Breakout 3,3V, para que pueda conectar fácilmente el tablero al puerto USB de un ordenador. O, para una solución inalámbrica, puede estar conectado a la del compañero Bluetooth o un explorador xbee.

Características:

*9 grados de libertad en una sola tarjeta, plana:
ITG-3200 - triple eje giroscopio de salida digital
ADXL345 - resolución de 13 bits, ± 16 g, acelerómetro de tres ejes
HMC5883L - de triple eje, magnetómetros digitales
*Las salidas de todos los sensores procesados por ATmega328 de a bordo y enviados a través de una corriente en serie
*Autorun menú de funciones y ayuda integrada en el ejemplo de firmware
*pines de salida coinciden con base FTDI Breakout, del compañero de Bluetooth, XBee Explorador
*de entrada 3.5-16VDC
*interruptor de control ON-OFF y el interruptor de restablecer
Dimensiones: 1,1 "x 1,6" (28 x 41 mm)

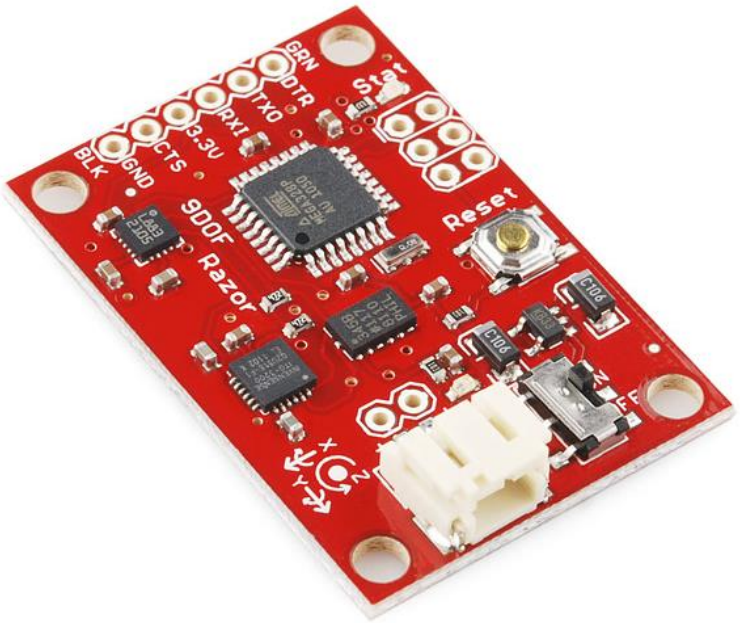

Figura 7. Tarjeta 9DOF RAZOR IMU

LABVIEW

Descripción:

LabVIEW (acrónimo de Laboratory Virtual Instrumentation Engineering Workbench) es una plataforma y entorno de desarrollo para diseñar sistemas, con un lenguaje de programación visual gráfico. Recomendado para sistemas hardware y software de pruebas, control y diseño, simulado o real y embebido, pues acelera la productividad. El lenguaje que usa se llama lenguaje G, donde la G simboliza que es lenguaje Gráfico.
Este programa fue creado por National Instruments (1976) para funcionar sobre máquinas MAC, salió al mercado por primera vez en 1986. Ahora está disponible para las plataformas Windows, UNIX, MAC y GNU/Linux. La penúltima versión es la 2013, con la increíble demostración de poderse usar simultáneamente para el diseño del firmware de un instrumento RF de última generación, a la programación de alto nivel del mismo instrumento, todo ello con código abierto. Y posteriormente la versión 2014 disponible en versión demo para estudiantes y profesional, la versión demo se puede descargar directamente de la página National Instruments.

.

Figura 8. Programa LabVIEW

XCTU

Descripción:

XCTU es una aplicación multi-plataforma libre diseñada para permitir a los desarrolladores interactuar con módulos Digi RF a través de una interfaz gráfica fácil de usar. Incluye nuevas herramientas que hacen más fácil de configurar, configurar y probar módulos RF XBee® [1].

XCTU incluye todas las herramientas que un desarrollador necesita para obtener rápidamente en marcha y funcionando con XBee. Características únicas como el punto de vista gráfico de la red, lo que representa gráficamente la red XBee junto con la intensidad de la señal de cada conexión, y el formador de tramas API XBee, que intuitivamente ayuda a construir e interpretar tramas API para XBees está utilizando en modo API, se combinan para hacer que el desarrollo en la plataforma XBee más fácil que nunca.

Características:

*Puede administrar y configurar varios dispositivos de RF, incluso de forma remota los dispositivos conectados (over-the-air).

*La actualización del firmware proceso sin problemas restaura la configuración del módulo, la manipulación automática de modo y la velocidad de transmisión cambios.

*Dos específica API y AT consolas, se han diseñado desde cero para comunicarse con sus dispositivos de radio.

*Ahora puede guardar sus sesiones de consola y cargarlos en un PC que ejecuta XCTU diferente.

*XCTU incluye un conjunto de herramientas embebidas que se pueden ejecutar sin tener ningún módulo de RF conectado:

*Generador de tramas: generar fácilmente cualquier tipo de marco de API para salvar su valor.

*Intérprete de marcos: descodificar un marco de principios activos y de ver sus valores específicos del marco.

*Recuperación: Recupera módulos de radio que han dañado firmware o están en modo de programación.

*Carga de la sesión de la consola: Cargar una sesión de consola guardada en cualquier XCTU PC en funcionamiento.

*Prueba de rango: Realizar una prueba de rango entre 2 módulos de radio de la misma red.

*Explorador de firmware: Navegar a través de la biblioteca de firmware XCTU.

*Un proceso de actualización le permite actualizar automáticamente la aplicación en sí y el firmware de radiobiblioteca sin necesidad de descargar los archivos adicionales.

*XCTU contiene documentación completa y exhaustiva que se puede acceder en cualquier momento.

Figura 9. Programa XCTU

SOFTWARE DE ARDUINO

Descripción:

El código abierto Arduino Software (IDE) hace que sea fácil de escribir código y subirlo a la junta. Se ejecuta en Windows, Mac OS X y Linux. El entorno está escrito en Java y basadas en el procesamiento y otro software de código abierto. Este software se puede utilizar con cualquier placa Arduino.

Figura 10. Programa ARDUINO UNO

TMOTOR AIR 2213 920KV CW/CCW

Descripción:

T - motor de engranajes 350 Poder Combo con 2213 920KV Motor y aire 20A ESC & T9545 Hélice

Características:

- 2213 Especificaciones : KV920
- KV : 920
- Estructura : 9N12P
- Tamaño del estator : 22 * 13mm
- Diámetro del eje : 4 mm
- Dimensiones del motor : 27,5 * 30 mm
- Peso : 54g
- Corriente de vacío: 0.5A@10V (A)
- Batería : 3-4S Lipo
- Max.continuous actual : 18A @ 180S
- Poder Max.continuous : 230W @ 180S
- Max.effective actual : (3-10a) > 83 %
- La resistencia interna: 132mohm

Figura 11. Motores TMOTOR AIR 2213 920KV CW/CCW

T9545 HÉLICE

Descripción:

Pareja de hélices Tmotor de plástico especial, perfectamente equilibradas.

Diversas posibilidades de anclaje, anclaje directo, anclaje por su propio cono o tradicional con tuerca.

Especificaciones:

-T-forma de la hélice del motor

-tamaño: 9.5*4.5 pulgadas

-3 métodos de instalación para diferentes motores

-trabajo con hilo CW motor tapón de rosca con borde negro, titular de la con borde de plata trabajar con hilo CCW motor titular (con marca blanca)
-juego para MT, MN serie motor de instalación

Figura 12. T9545 HÉLICE

HW30A ESC MOTOR

Descripción:

Función de alimentación seguro: independientemente de la válvula reguladora stick en cualquier posición el motor no arranca inmediatamente la función de calibración de la válvula reguladora: adaptarse a la diferencia de recorrido del acelerador remoto diferente, mejorar la linealidad de la respuesta del acelerador, con una sensación suave, delicado y excelente velocidad lineal modo de protección de baja tensión, umbral de protección de baja tensión

Características:

peso:25g
Dimensiones: 45 x 24 x 11 mm.
firmware: Hobbywing
alimentación entrada: 5.6v - 16.8v (2-3 celdas li-poly, o 5-12 las células ni-mh ni-mh / ni-CD)
bec:2a corriente
constante: 30a (max 40a menos de 10 segundos)

Figura 13. HW30A ESC MOTOR

BATERÍA LIPO 1500MAH - 11.1V

Descripción:

Ésta batería puede ser utilizada tanto en radiocontrol como para proyectos de robótica que necesiten gran entrega de potencia como por ejemplo cuando se utilizan motores.

Características:

- 11.1V (3 celdas)
- 1500mAh
- Descarga contínua: 20C
- Conector de carga: JST-XH
- Dimensiones: 71.6x34.5x22mm
- Peso: 114g

Figura 14. BATERÍA LIPO 1500MAH - 11.1V

XBEE 2MW CABLE ANTENA – SERIE 2

Descripción:

Este es el módulo XBee XB24-Z7WIT-004 desde Digi.Serie 2 mejora en el protocolo de salida de energía y datos. Serie 2 módulos permiten crear redes de malla complejas basadas en el firmware de malla XBee ZB ZigBee. Estos módulos permiten una comunicación muy fiable y simple entre microcontroladores, ordenadores, sistemas, realmente cualquier cosa con un puerto serie!Punto de redes de punto y multipunto son compatibles.

Características:

- 3.3V 40mA @
- velocidad de datos de 250kbps Max
- salida 2mW (+ 3dBm)
- 400 pies (120 m) Rango
- Antena incorporada
- Totalmente certificado por la FCC
- 6 pines de entrada del ADC de 10 bits
- 8 pines IO digitales
- cifrado de 128 bits

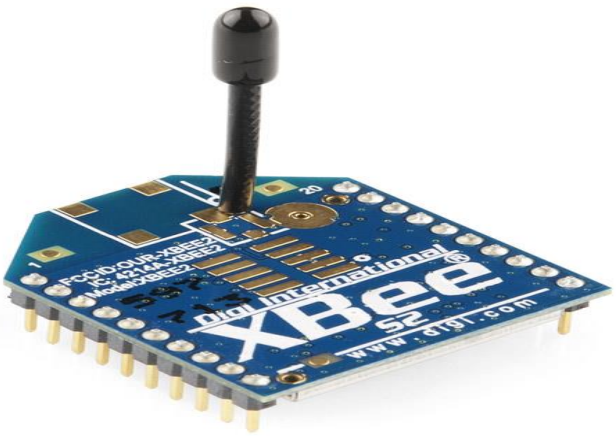

Figura 15. XBEE 2MW CABLE ANTENA - SERIE 2

PROCESSING 3.1.1

Descripción:

Es un lenguaje de programación y entorno de desarrollo integrado de código abierto basado en Java, de fácil utilización, y que sirve como medio para la enseñanza y producción de proyectos multimedia e interactivos de diseño digital. Fue iniciado por Ben Fry y Casey Reas a partir de reflexiones en el Aesthetics and Computation Group del MIT Media Lab dirigido por John Maeda.

Figura 16. Programa PROCESSING 3.1.1

CAPITULO III
PROCEDIMIENTO Y DESCRIPCION DE LAS ACTIVIDADES REALIZADAS

Comunicación inalámbrica XBEE s2 (RX y TX)

En esta imagen se puede apreciar el XBEE s2 (TX) conectado al puerto serial de la computadora, donde enviaremos los datos obtenidos de la tarjeta IMU RAZOR 9DOF.

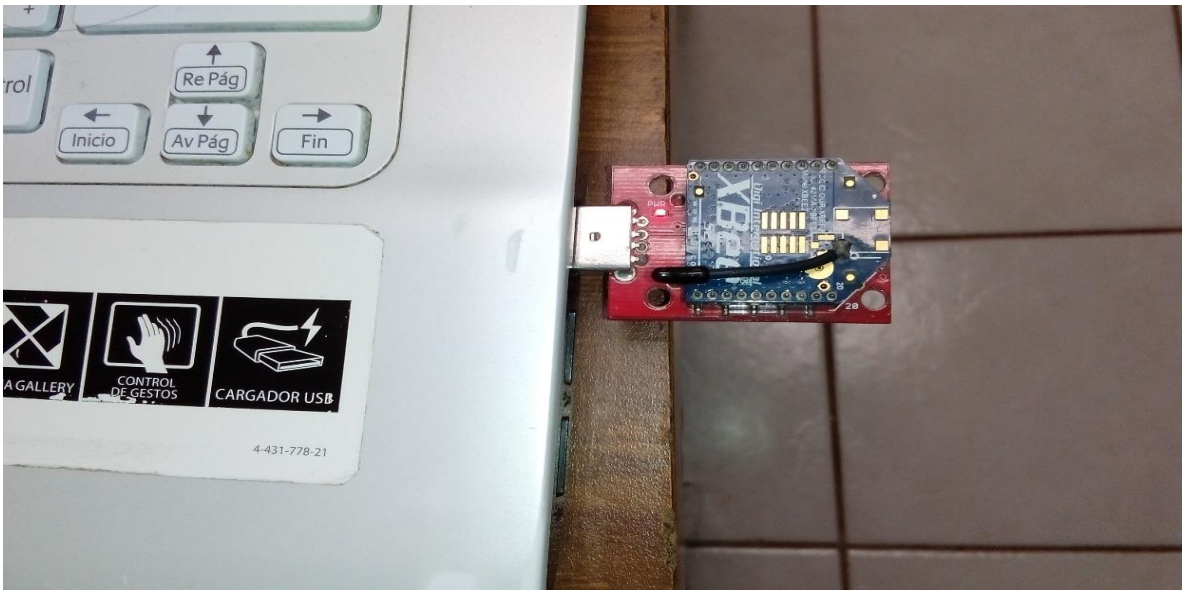

Figura 17. Conexión de XBEE s2 (TX)

En esta imagen se puede apreciar el XBEE s2 (RX) conectado a un ARDUINO, donde recibiremos los datos obtenidos del XBEE s2 (TX).

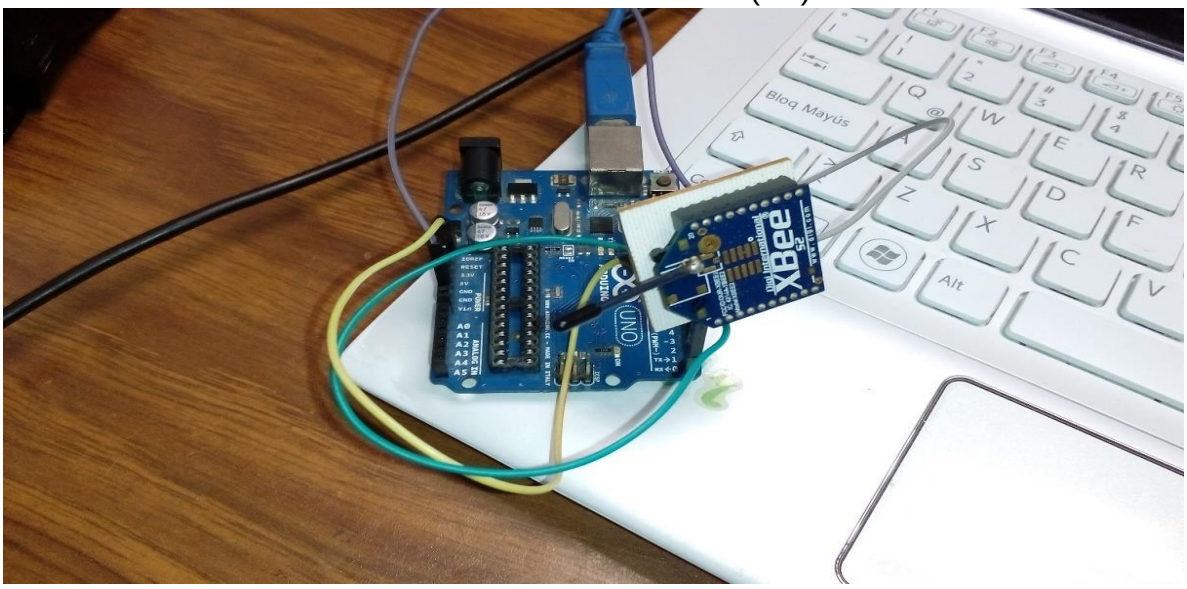

Figura 18. Conexión de XBEE s2 (RX)

En esta imagen se puede apreciar el programa de XCTU en el cual se puede observar cuando los XBEE s2 estén enlazados de la misma forma se pude visualizar el envió y el recibo de datos.

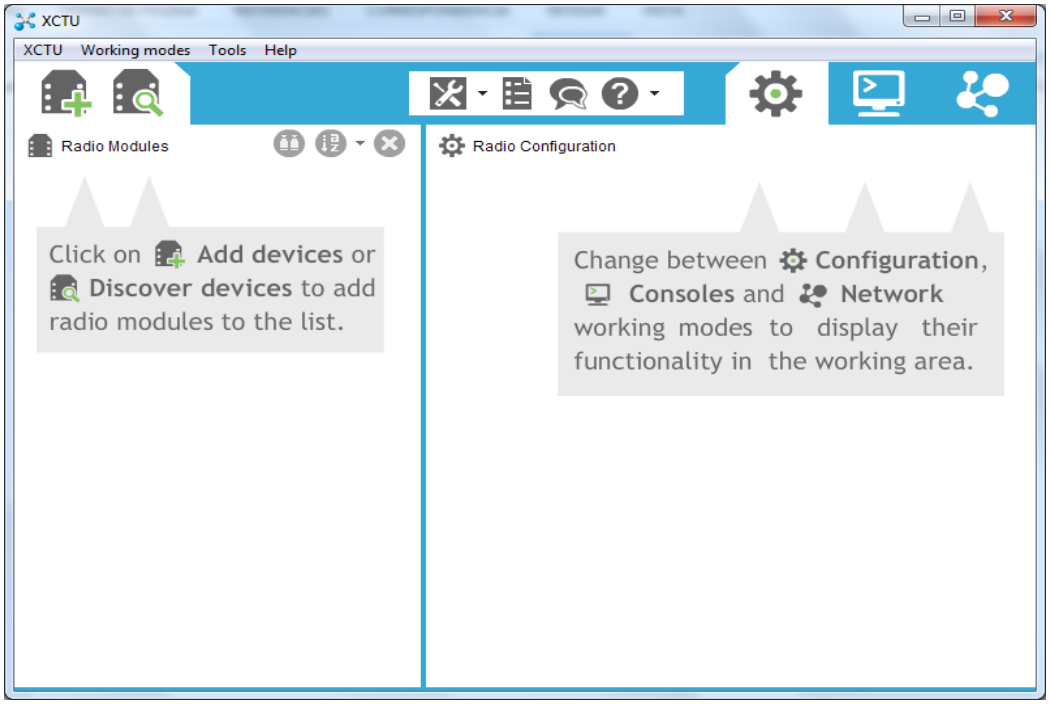

Figura 19. Programa XCTU

En la siguiente imagen se puede apreciar el programa XCTU, donde se elige el puerto de comunicación y a su vez se configura la velocidad de los XBEE s2.

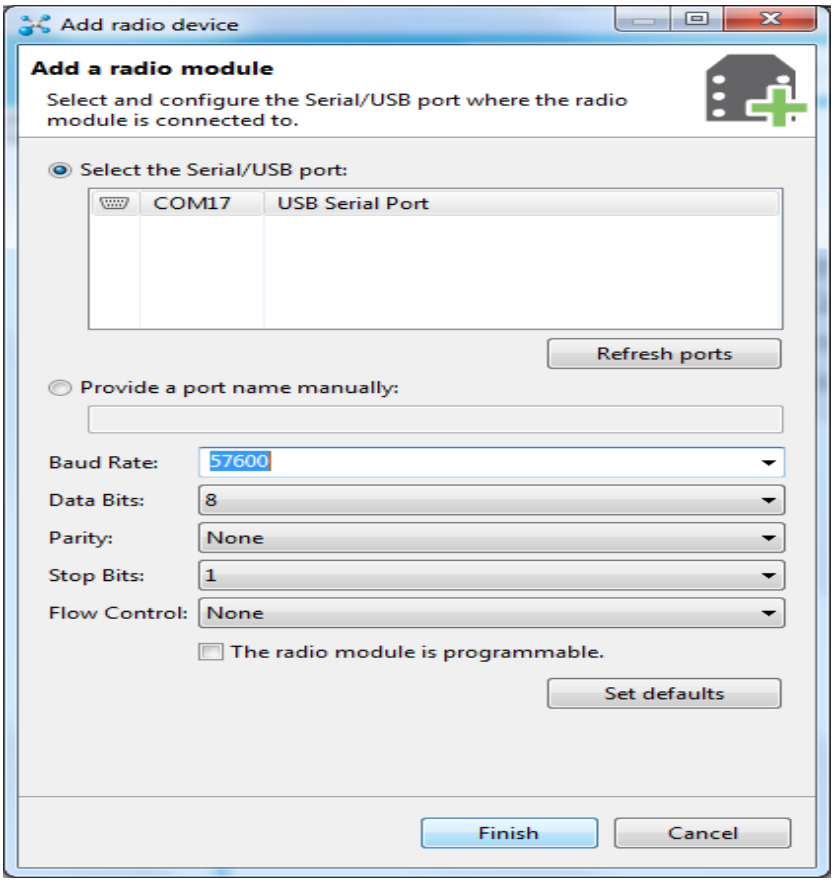

Figura 20. Agregando el dispositivo de radio (XBEE s2).COM17

En la siguiente imagen se puede apreciar el programa XCTU, donde se reconoció el XBEE s2.

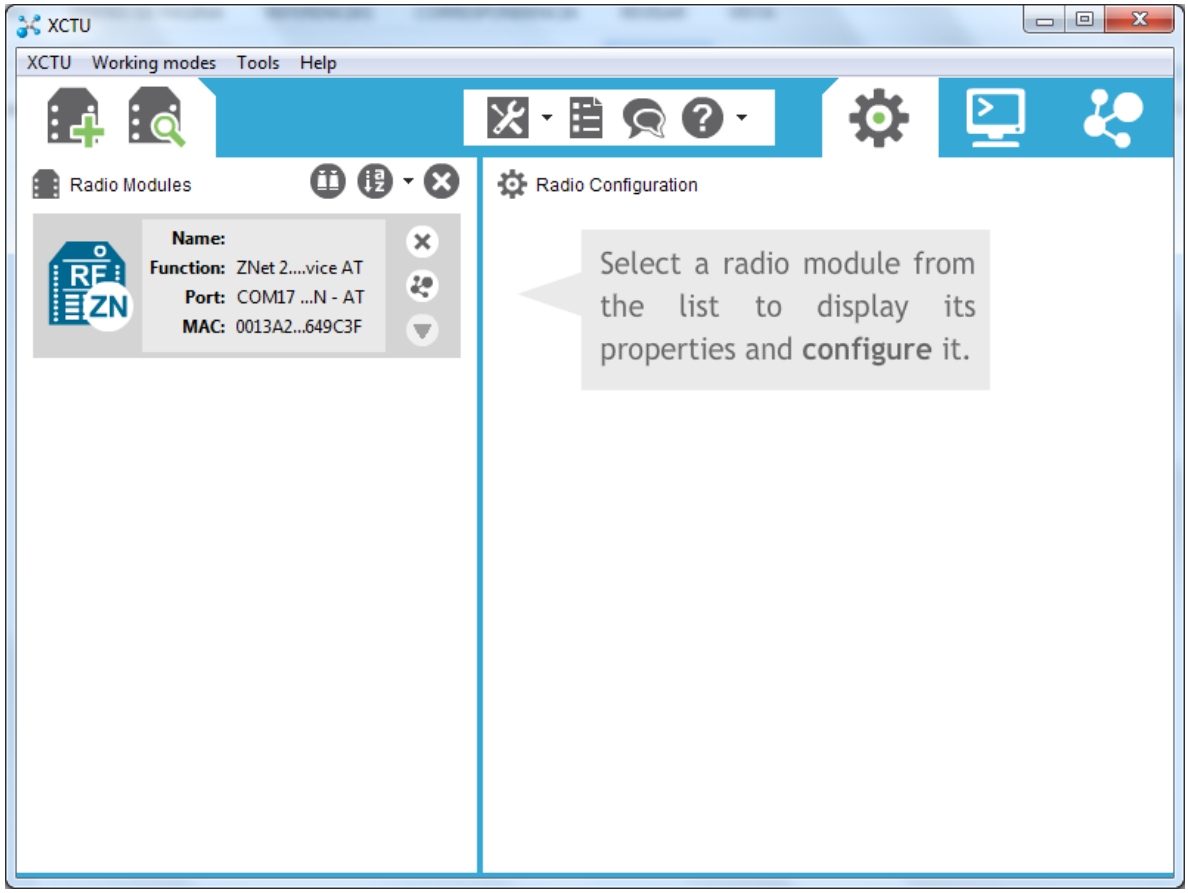

Figura 21.
Seleccionando el módulo de radio de la lista para mostrar sus propiedades y configurarlo

En la siguiente imagen se puede apreciar el programa XCTU, donde se aprecia que está configurado el XBEE s2.

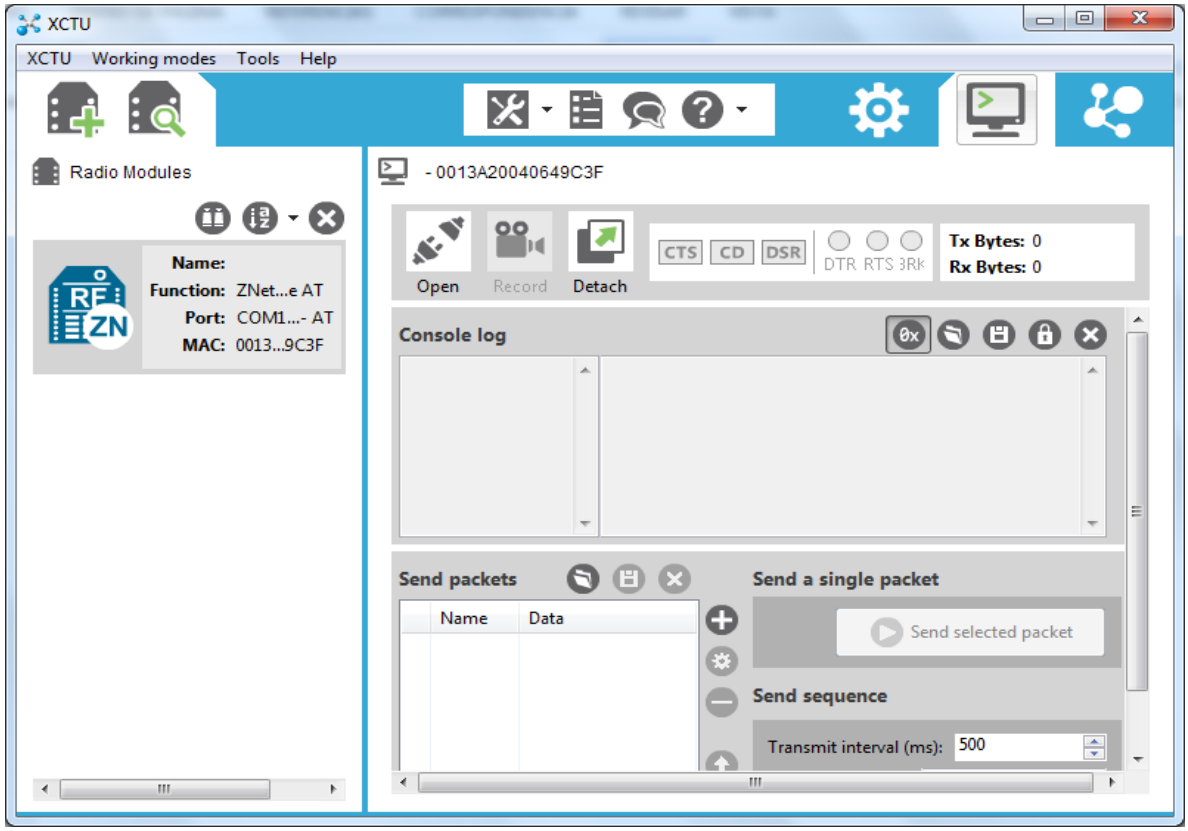

Figura 22. Enlazar las tarjetas XBEE s2

En la siguiente imagen se puede apreciar a los XBEE s2 (TX,RX), enlazados, con sus respectivas configuraciones adecuadas .

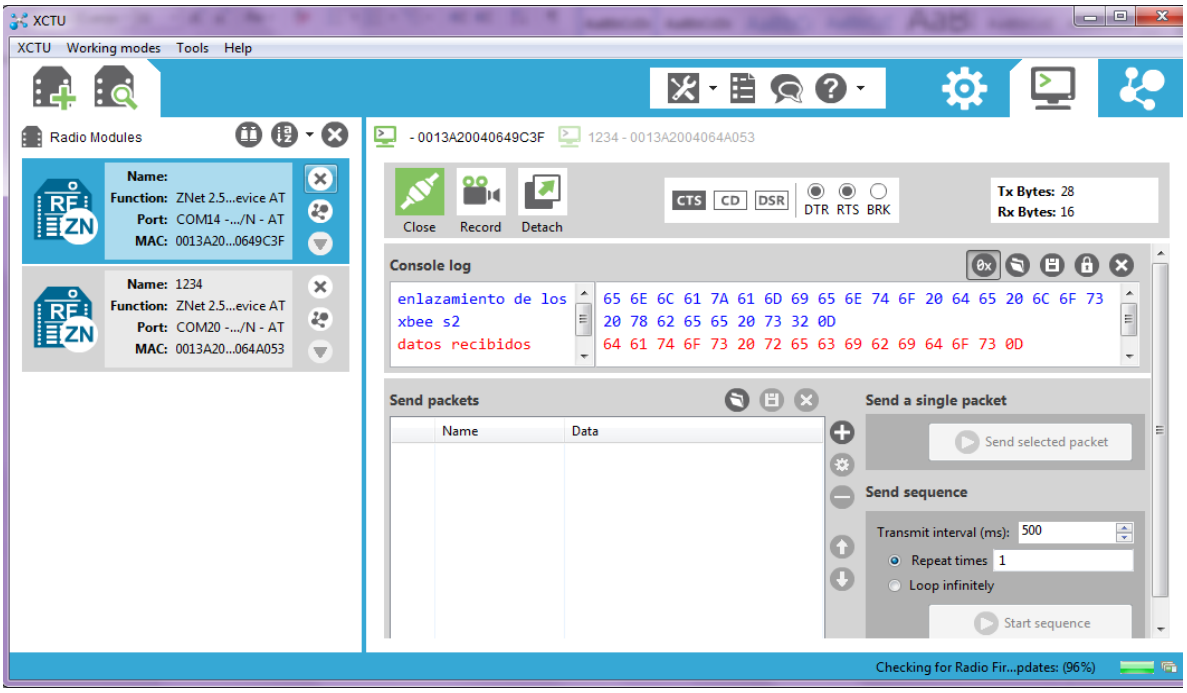

Figura 23. Enlazamiento de los XBEE s2, la tarjeta uno envía datos y la tarjeta dos recibe datos

En la siguiente imagen se aprecia, el enlazamiento de los XBEE s2, el XBEE s2 (TX) manda los datos en color rojo, y el XBEE s2 (RX) recibe los datos en color azul.

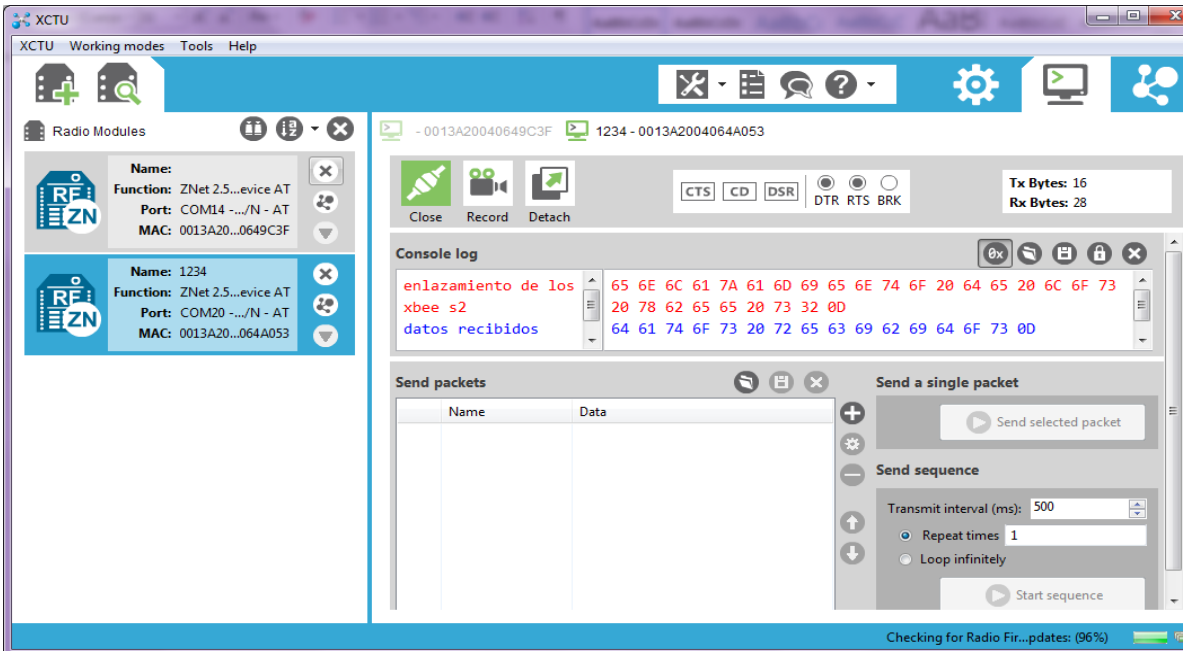

Figura 24. Enlazamiento de los XBEE s2, la tarjeta dos envía datos y la tarjeta uno recibe datos.

Cargar programa a la tarjeta IMU RAZOR 9DOF

En esta imagen se puede apreciar el programa Processing 3.1.1, en ello se carga el programa para configurar la tarjeta IMU RAZOR 9DOF.

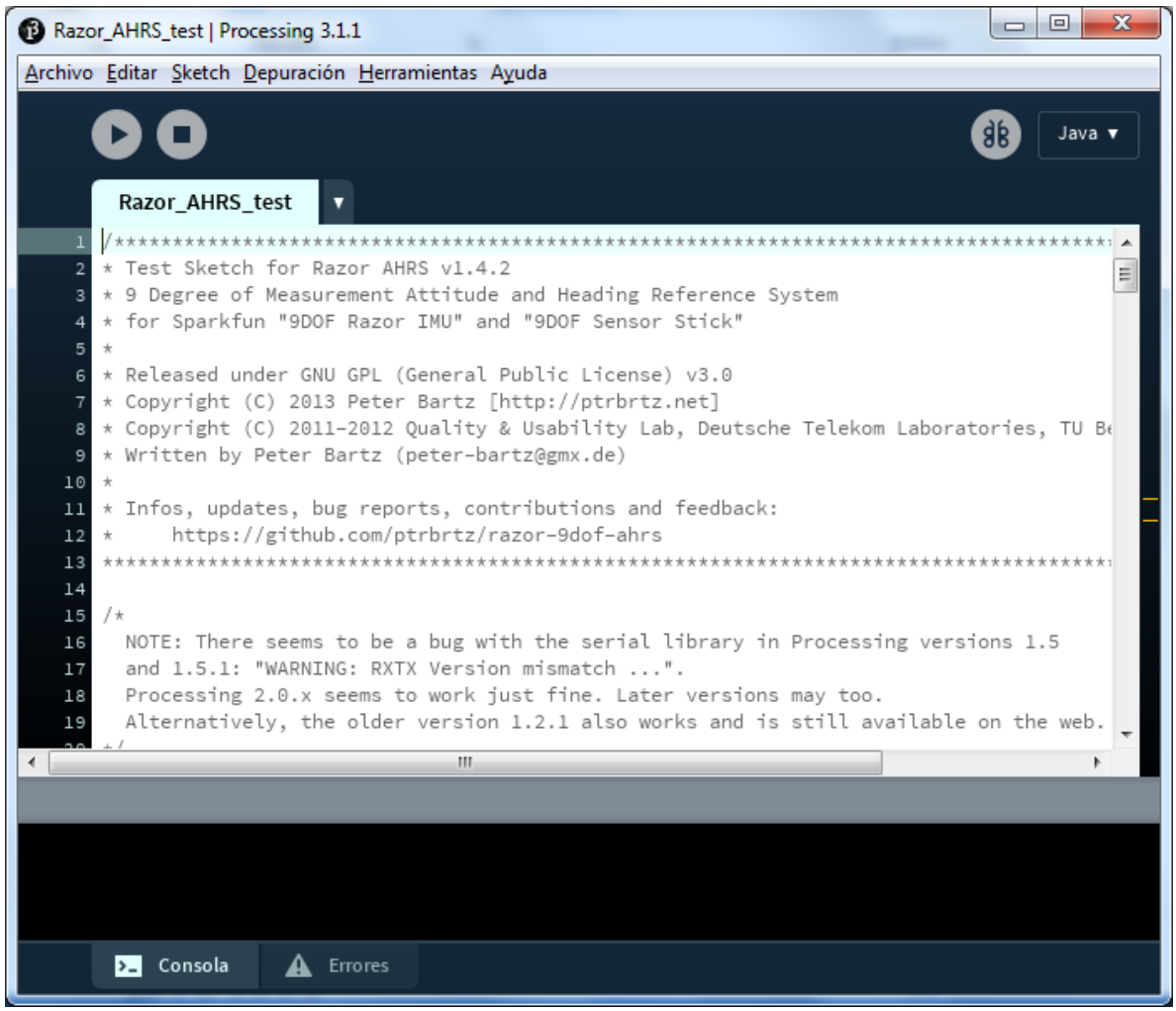

Figura 25. Programa Processing 3.1.1

A continuación se muestra el código del programa, el cual permite configurar la tarjeta IMU RAZOR 9DOF, se configura y calibra los siguientes sensores (acelerómetro, magnetómetro, giroscopio).

Código

```
/***********************************************************************
* Test Sketch for Razor AHRS v1.4.2
* 9 Degree of Measurement Attitude and Heading Reference System
* for Sparkfun "9DOF Razor IMU" and "9DOF Sensor Stick"
* Released under GNU GPL (General Public License) v3.0
* Copyright (C) 2013 Peter Bartz [http://ptrbrtz.net]
* Copyright (C) 2011-2012 Quality & Usability Lab, Deutsche Telekom Laboratories, TU Berlin
* Written by Peter Bartz (peter-bartz@gmx.de)
* Infos, updates, bug reports, contributions and feedback:
*     https://github.com/ptrbrtz/razor-9dof-ahrs
***********************************************************************/
/*
  NOTE: There seems to be a bug with the serial library in Processing versions 1.5
  and 1.5.1: "WARNING: RXTX Version mismatch ...".
  Processing 2.0.x seems to work just fine. Later versions may too.
  Alternatively, the older version 1.2.1 also works and is still available on the web.
*/
import processing.opengl.*;
import processing.serial.*;

// IF THE SKETCH CRASHES OR HANGS ON STARTUP, MAKE SURE YOU ARE USING THE RIGHT SERIAL PORT:
// 1. Have a look at the Processing console output of this sketch.
// 2. Look for the serial port list and find the port you need (it's the same as in Arduino).
// 3. Set your port number here:
final static int SERIAL_PORT_NUM = 0;
// 4. Try again.

final static int SERIAL_PORT_BAUD_RATE = 57600;

float yaw = 0.0f;
float pitch = 0.0f;
float roll = 0.0f;
float yawOffset = 0.0f;
```

```
PFont font;
Serial serial;
boolean synched = false;

void drawArrow(float headWidthFactor, float headLengthFactor) {
  float headWidth = headWidthFactor * 200.0f;
  float headLength = headLengthFactor * 200.0f;
  pushMatrix();

  // Draw base
  translate(0, 0, -100);
  box(100, 100, 200);

  // Draw pointer
  translate(-headWidth/2, -50, -100);
  beginShape(QUAD_STRIP);
    vertex(0, 0 ,0);
    vertex(0, 100, 0);
    vertex(headWidth, 0 ,0);
    vertex(headWidth, 100, 0);
    vertex(headWidth/2, 0, -headLength);
    vertex(headWidth/2, 100, -headLength);
    vertex(0, 0 ,0);
    vertex(0, 100, 0);
  endShape();
  beginShape(TRIANGLES);
    vertex(0, 0, 0);
    vertex(headWidth, 0, 0);
    vertex(headWidth/2, 0, -headLength);
    vertex(0, 100, 0);
    vertex(headWidth, 100, 0);
    vertex(headWidth/2, 100, -headLength);
  endShape();
  popMatrix();
}
void drawBoard() {
  pushMatrix();
rotateY(-radians(yaw - yawOffset));
rotateX(-radians(pitch));
rotateZ(radians(roll));

// Board body
fill(255, 0, 0);
```

```
box(250, 20, 400);

// Forward-arrow
pushMatrix();
translate(0, 0, -200);
scale(0.5f, 0.2f, 0.25f);
fill(0, 255, 0);
drawArrow(1.0f, 2.0f);
popMatrix();
popMatrix();
}
// Skip incoming serial stream data until token is found
boolean readToken(Serial serial, String token) {
// Wait until enough bytes are available
if (serial.available() < token.length())
return false;
  // Check if incoming bytes match token
  for (int i = 0; i < token.length(); i++) {
  if (serial.read() != token.charAt(i))
  return false;
  }
  return true;
  }

// Global setup
void setup() {
// Setup graphics
size(640, 480, OPENGL);
smooth();
noStroke();
frameRate(50);

// Load font
font = loadFont("Univers-66.vlw");
textFont(font);

// Setup serial port I/O
println("AVAILABLE SERIAL PORTS:");
println(Serial.list());
String portName = Serial.list()[SERIAL_PORT_NUM];
println();
println("HAVE A LOOK AT THE LIST ABOVE AND SET THE RIGHT SERIAL
PORT NUMBER IN THE CODE!");
```

```
println("  -> Using port " + SERIAL_PORT_NUM + ": " + portName);
serial = new Serial(this, portName, SERIAL_PORT_BAUD_RATE);
}
void setupRazor() {
println("Trying to setup and synch Razor...");

// On Mac OSX and Linux (Windows too?) the board will do a reset when we connect, which is really bad.
// See "Automatic (Software) Reset" on http://www.arduino.cc/en/Main/ArduinoBoardProMini
// So we have to wait until the bootloader is finished and the Razor firmware can receive commands.
// To prevent this, disconnect/cut/unplug the DTR line going to the board. This also has the advantage,
// that the angles you receive are stable right from the beginning.
delay(3000);  // 3 seconds should be enough

// Set Razor output parameters
serial.write("#ob");  // Turn on binary output
serial.write("#o1");  // Turn on continuous streaming output
serial.write("#oe0"); // Disable error message output

// Synch with Razor
serial.clear();  // Clear input buffer up to here
serial.write("#s00");  // Request synch token
}

float readFloat(Serial s) {
// Convert from little endian (Razor) to big endian (Java) and interpret as float
return Float.intBitsToFloat(s.read() + (s.read() << 8) + (s.read() << 16) + (s.read() << 24));
}

void draw() {
// Reset scene
background(0);
lights();

// Sync with Razor
if (!synched) {
textAlign(CENTER);
fill(255);
text("Connecting to Razor...", width/2, height/2, -200);
```

```
if (frameCount == 2)
setupRazor();  // Set ouput params and request synch token
else if (frameCount > 2)
synched = readToken(serial, "#SYNCH00\r\n");  // Look for synch token
return;
}

// Read angles from serial port
while (serial.available() >= 12) {
yaw = readFloat(serial);
pitch = readFloat(serial);
roll = readFloat(serial);
}


// Draw board
  pushMatrix();
  translate(width/2, height/2, -350);
  drawBoard();
  popMatrix();

  textFont(font, 20);
  fill(255);
  textAlign(LEFT);

  // Output info text
  text("Point FTDI connector towards screen and press 'a' to align", 10, 25);

  // Output angles
  pushMatrix();
  translate(10, height - 10);
  textAlign(LEFT);
  text("Yaw: " + ((int) yaw), 0, 0);
  text("Pitch: " + ((int) pitch), 150, 0);
  text("Roll: " + ((int) roll), 300, 0);
  popMatrix();
}

void keyPressed() {
  switch (key) {
    case '0':  // Turn Razor's continuous output stream off
      serial.write("#o0");
```

```
      break;
    case '1':  // Turn Razor's continuous output stream on
      serial.write("#o1");
      break;
    case 'f':   // Request one single yaw/pitch/roll frame from Razor (use when continuous streaming is off)
      serial.write("#f");
      break;
    case 'a':  // Align screen with Razor
      yawOffset = yaw;
  }
```

Calibración de los sensores de la tarjeta IMU RAZOR 9DOF

En esta imagen se puede observar los datos que nos arroja la tarjeta IMU RAZOR 9ODF al estar calibrado.

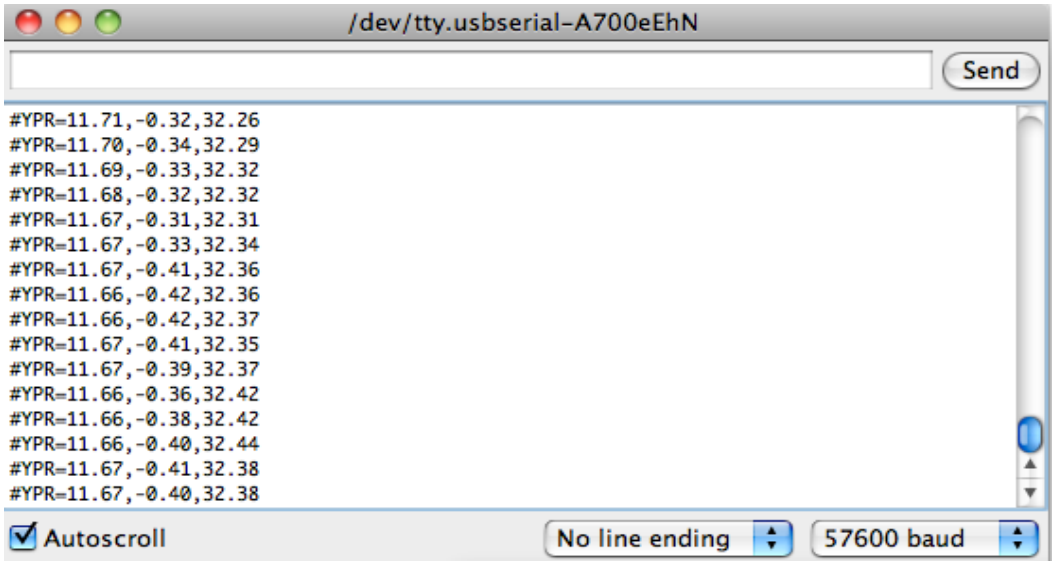

Figura 26. Probando rastreador

En esta imagen se puede apreciar el cambio de los datos obtenidos al modificar los comandos en el monitor serial del programa.

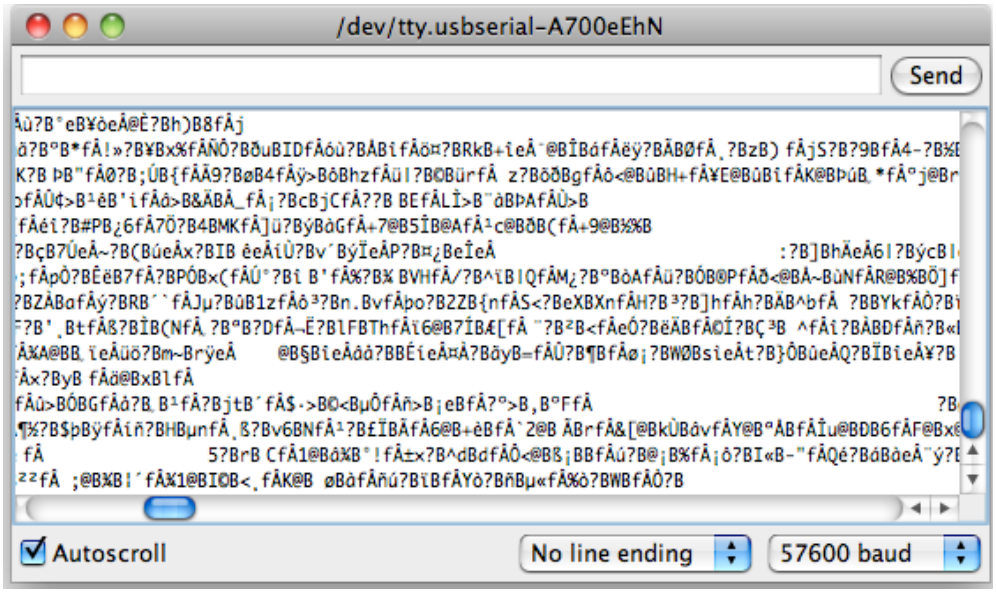

Figura 27. Cambiar a la salida binaria mediante el envío #ob y debería ver algo como esto

Visualización en 3D de la tarjeta IMU RAZOR 9DOF

En esta imagen se puede apreciar en forma 3D el comportamiento de la tarjeta IMU RAZOR 9DOF.

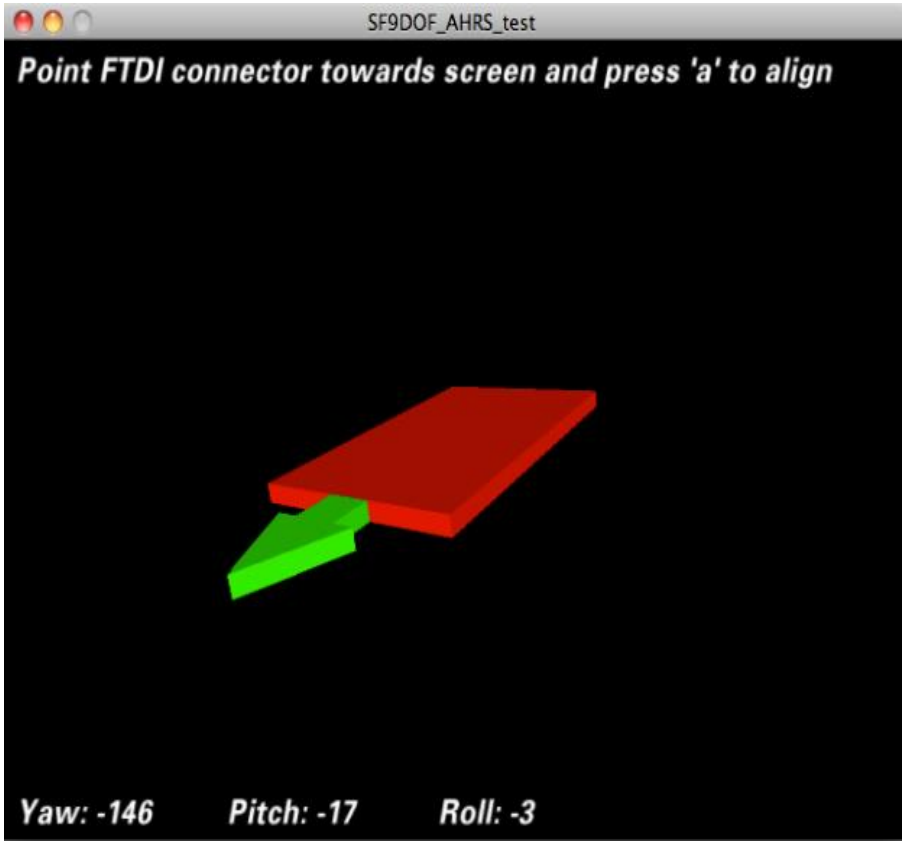

Figura 28. Procesamiento boceto prueba en PROCESSING 3.1.1

Hacer el programa del PVTOL en ARDUINO

En esta imagen se puede apreciar el programa cargado en Arduino.

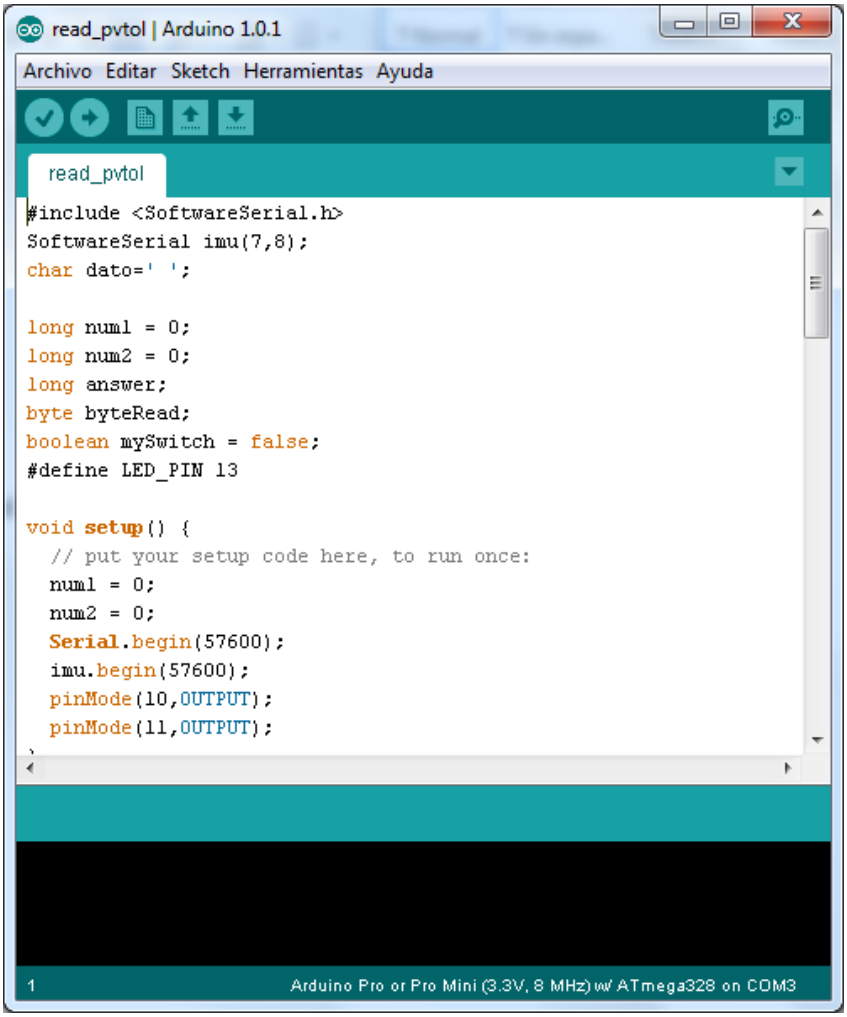

Figura 29. Programa del PVTOL en ARDUINO

El código que se muestra a continuación nos sirve para poder recibir los datos de la tarjeta IMU RAZOR 9DOF, después enviar los datos por el XBEE a la computadora y por ultimo recibir los datos del XBEE a Arduino y mandarlos por PWM a los motores.

Código

```
#include <SoftwareSerial.h>
SoftwareSerial imu(7,8);
char dato=' ';
long num1 = 0;
long num2 = 0;
long answer;
byte byteRead;
boolean mySwitch = false;
#define LED_PIN 13

void setup() {
  // put your setup code here, to run once:
  num1 = 0;
  num2 = 0;
  Serial.begin(57600);
  imu.begin(57600);
  pinMode(10,OUTPUT);
  pinMode(11,OUTPUT);
}

void loop() {
   if(imu.available())
  {
    dato=imu.read();
    Serial.print(dato);
  }
 if (Serial.available()>0) {
    /* read the most recent byte */
    byteRead = Serial.read();

    //listen for numbers between 0-9
    if(byteRead>47 && byteRead<58){
      //number found

      /* If mySwitch is true, then populate the num1 variable
         otherwise populate the num2 variable*/
```

```
      if(!mySwitch){
        num1=(num1*10)+(byteRead-48);
      }else{
        num2=(num2*10)+(byteRead-48);
      }
    }

    /*Listen for an equal sign (byte code 61)
      to calculate the answer and send it back to the
      serial monitor screen*/
    if(byteRead==61){//=
      answer=num1+num2;
//      Serial.print(num1);
//      Serial.print("+");
//      Serial.print(num2);
//      Serial.print("=");
//      Serial.println(answer);
        // if (answer>=200){
        //     digitalWrite(LED_PIN, HIGH);
        // }
        // else if (answer<200)
        // {
        //     digitalWrite(LED_PIN, LOW);
        // }
      analogWrite(10,num1);
      analogWrite(11,num2);
      //Serial.print(num1);
      //Serial.print("+");
      //Serial.print(num2);
      //Serial.print("=");
      //Serial.println(answer);
      /* Reset the variables for the next round */
      num1=0;
      num2=0;
      mySwitch=false;

    /* Listen for the addition sign (byte code 43). This is
       used as a delimiter to help define num1 from num2 */
    }else if (byteRead==44){
      mySwitch=true;
```

Construcción de la base del PVTOL

En esta imagen se puede apreciar la base del PVTOL fabricado de madera, con un balero en el centro que realiza el movimiento en eje x.

Figura 30. Construcción del PVTOL hecho de madera

Montaje de los componentes electrónicos que utilizara el PVTOL

En esta imagen se puede apreciar el armado del PVTOL con sus correspondientes componentes electrónicos ubicados adecuadamente (motores, tarjeta de conexiones, hélices de los motores, tarjeta IMU RAZOR 9DOF, Arduino, y los controladores de los motores).

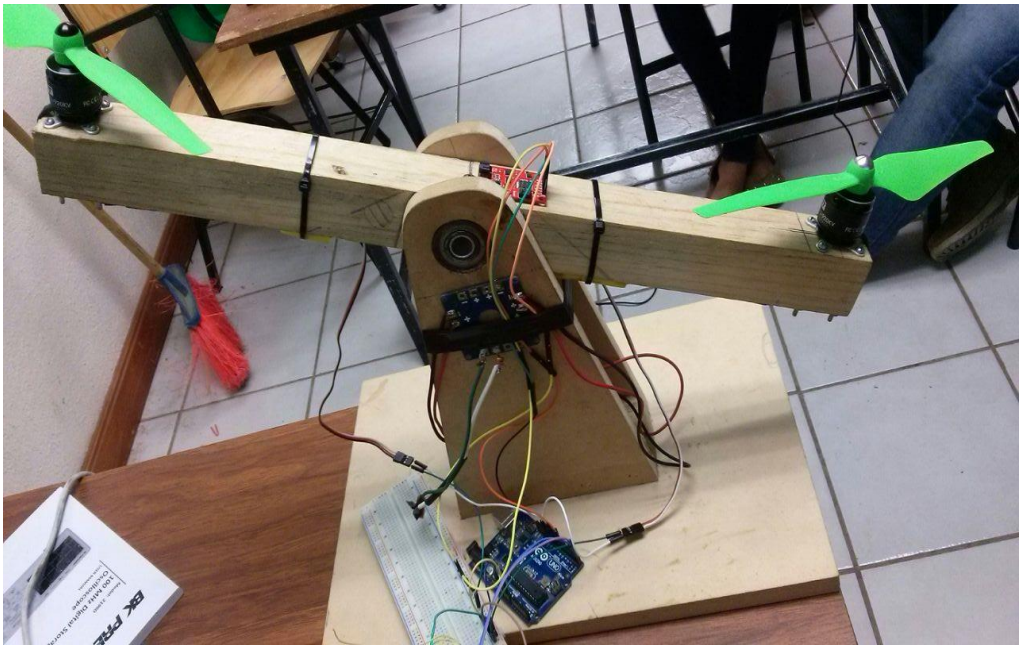

Figura 31. Componentes electronicos puestos en el PVTOL

En esta imagen se puede apreciar modificaciones del PVTOL para que no haya ningún problema.

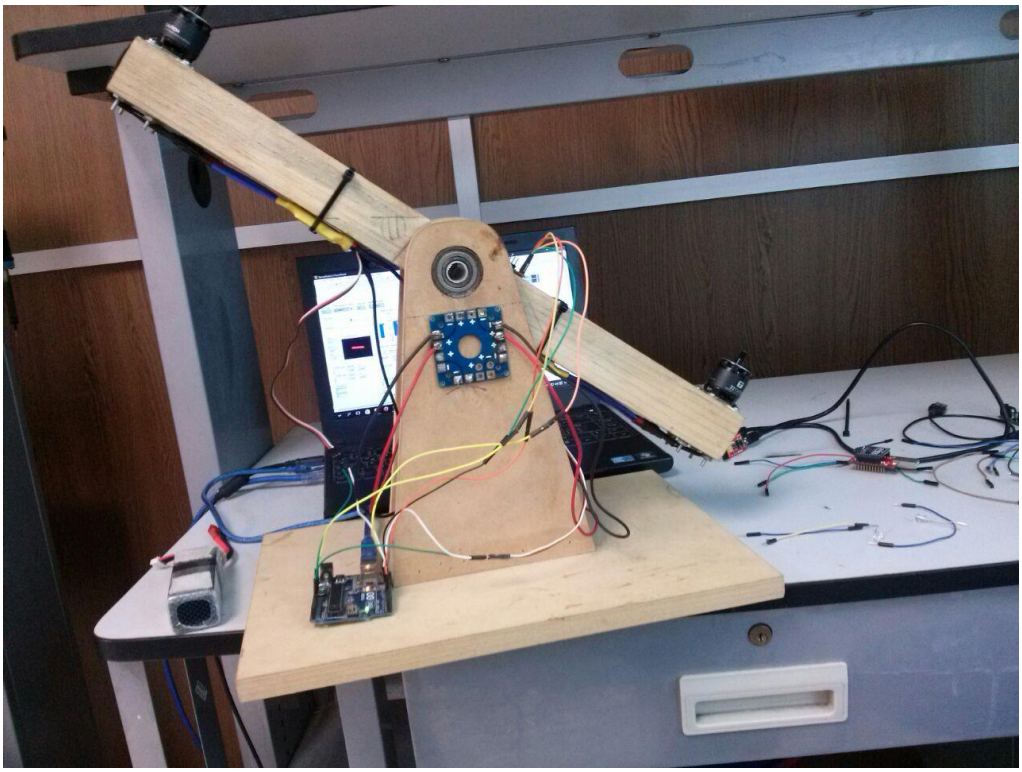

Figura 32. Componentes electrónicos puestos en el PVTOL

Sistema de control en LABVIEW

En esta imagen se puede apreciar el programa creado en LABVIEW para el control del PVTOL.

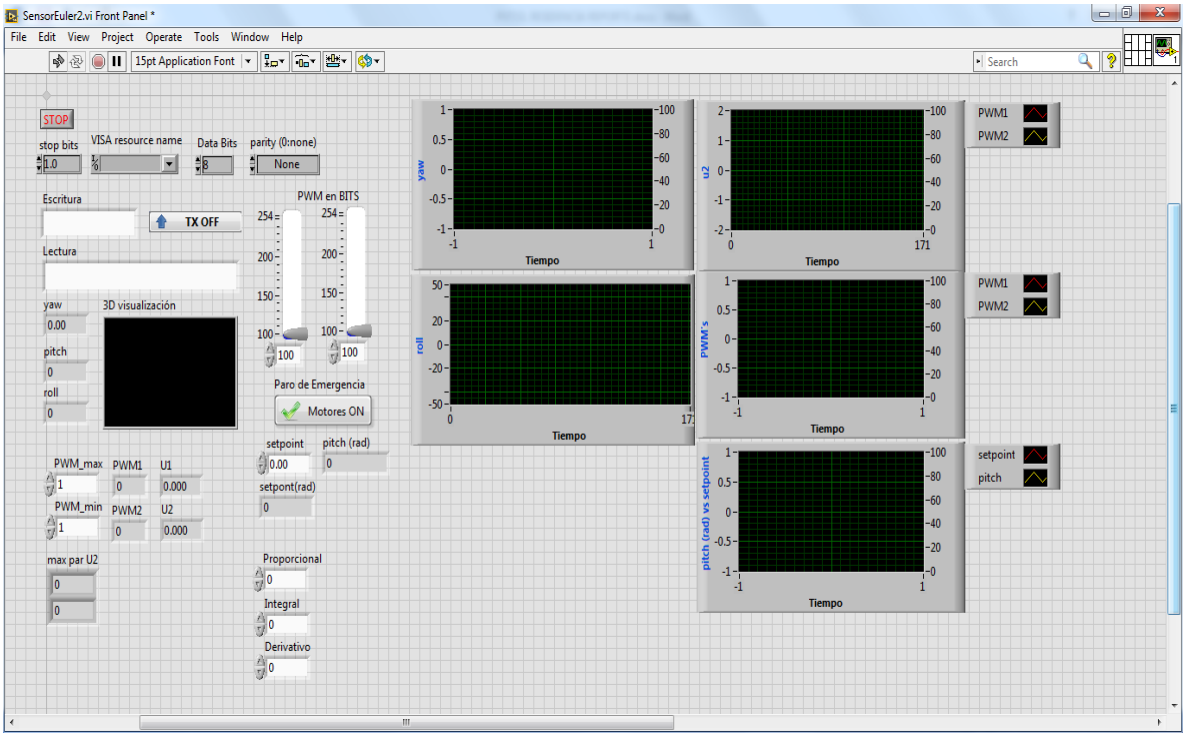

Figura 33. Programa del PVTOL en LABIEW

En esta imagen se puede apreciar las configuraciones que se debe hacer cuando se conecta el XBEE s2.

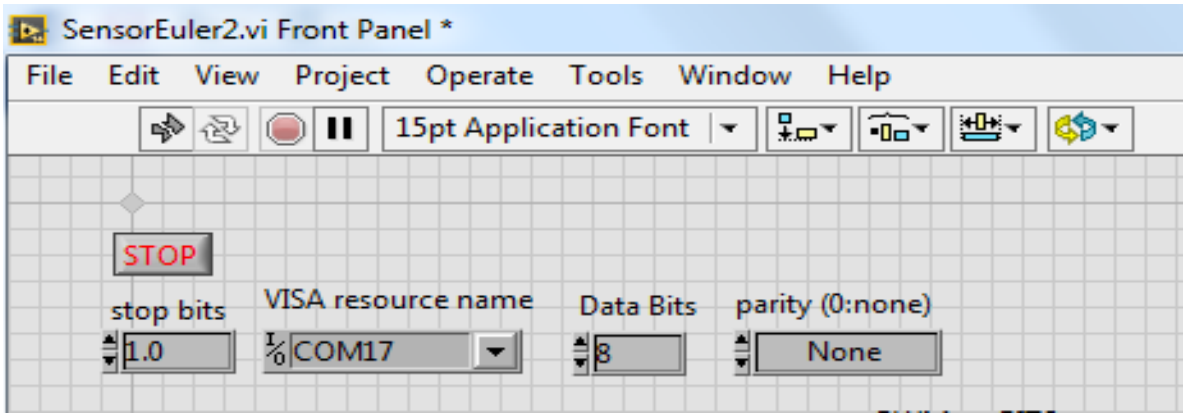

Figura 34. Configurar el COM del XBEE s2 (TX)

En esta imagen se puede apreciar que la comunicación de datos de los XBEE s2 no está funcionando, hasta que se le aplaste el botón de TX OFF.

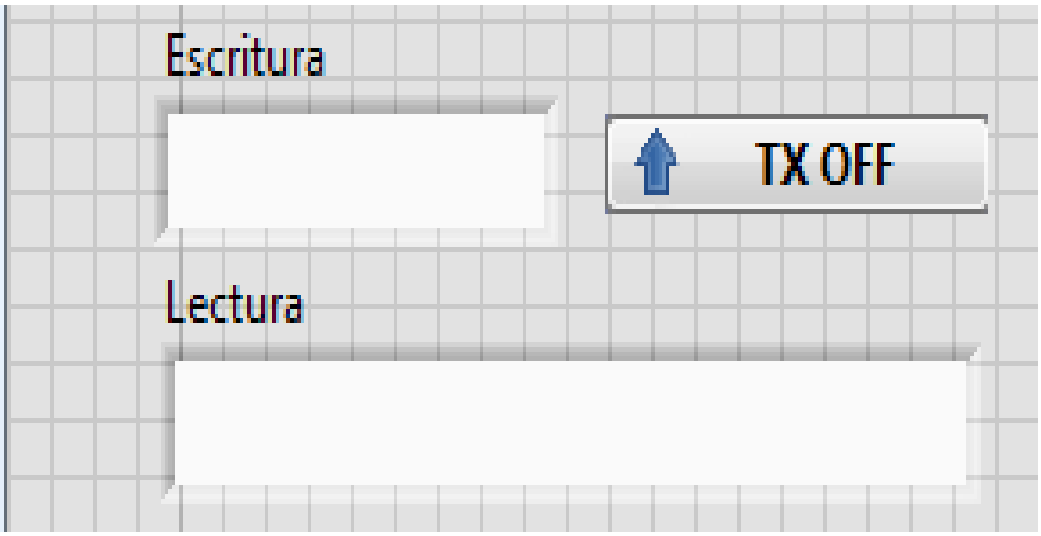

Figura 35. Comunicación inalámbrica de LABVIEW

En esta imagen se puede apreciar una tabla de variables donde se ira variando el posicionamiento del PVTOL y se verá gráficamente.

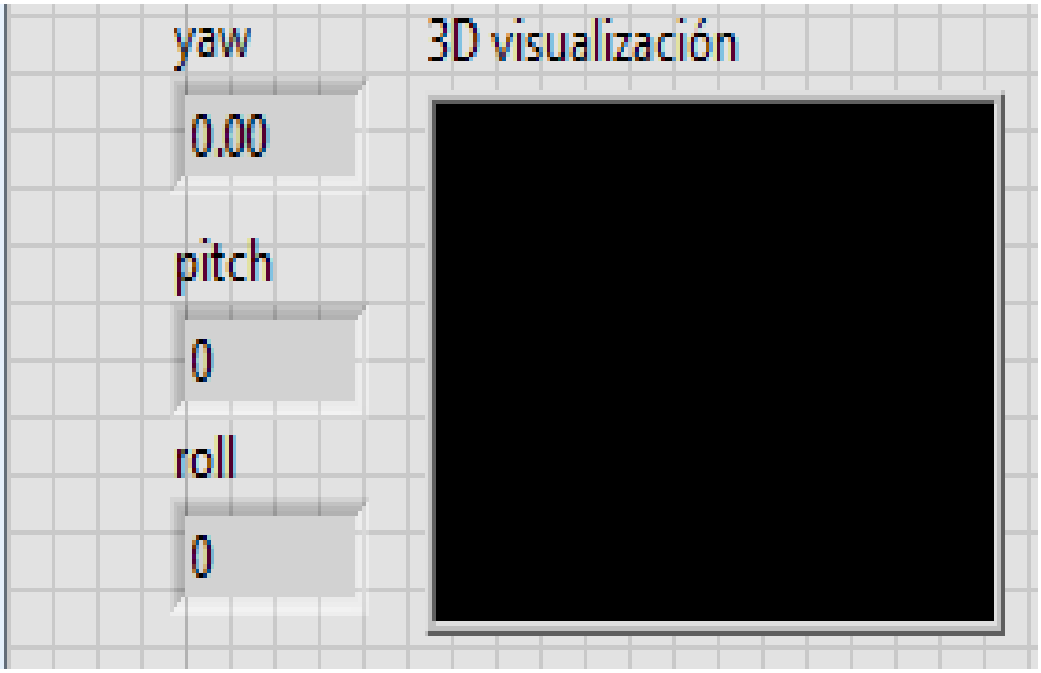

Figura 36. Visualización de la tarjeta IMU RAZOR 9DOF

En esta imagen se puede apreciar una tabla de variables el cual se configurara y controlara la velocidad de los motores del PVTOL.

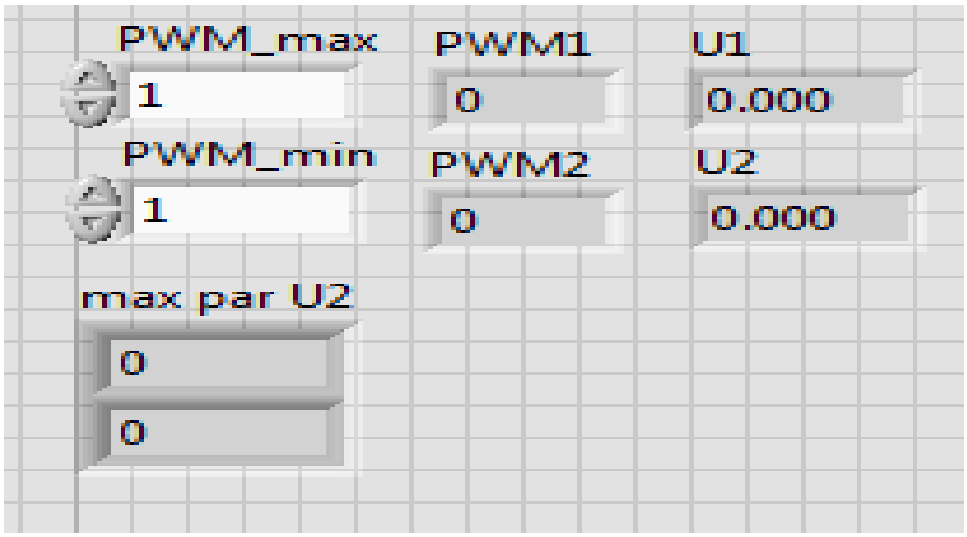

Figura 37. Control de los motores del PVTOL

En esta imagen se puede apreciar el comportamiento de los motores del PVTOL, además tiene un botón de paro de emergencia para detener los motores en caso de algún probema.

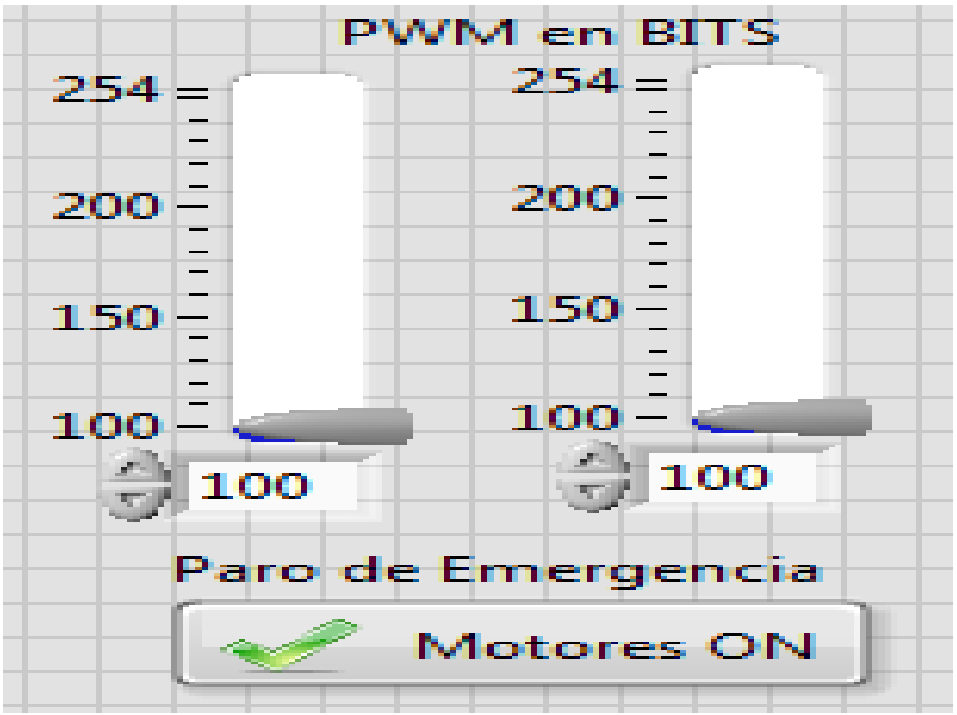

Figura 38. Visualización de los motores y paro de emergencia del PVTO

En esta imagen se puede apreciar una tabla de variables para configurar el posicionamiento del PVTOL en un determinado grado, que queramos posicionarlo y los datos será convertidos en radianes, además se podrá configurar el PID.

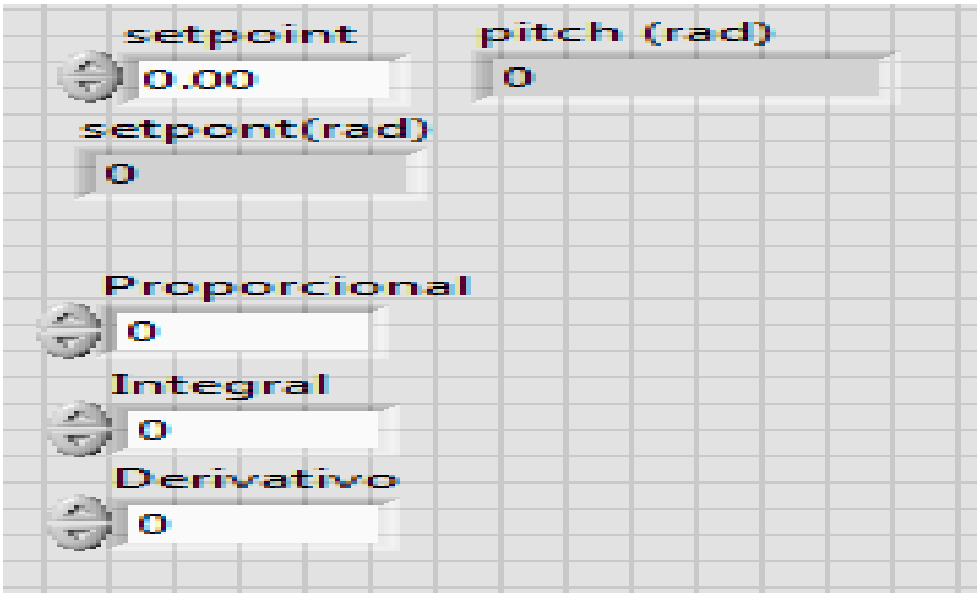

Figura 39. Configuración del posicionamiento del PVTOL y del PID

En esta imagen se puede apreciar las gráficas del sistema de control del PID, del PWM y de los grados de posicionamiento del PVTOL.

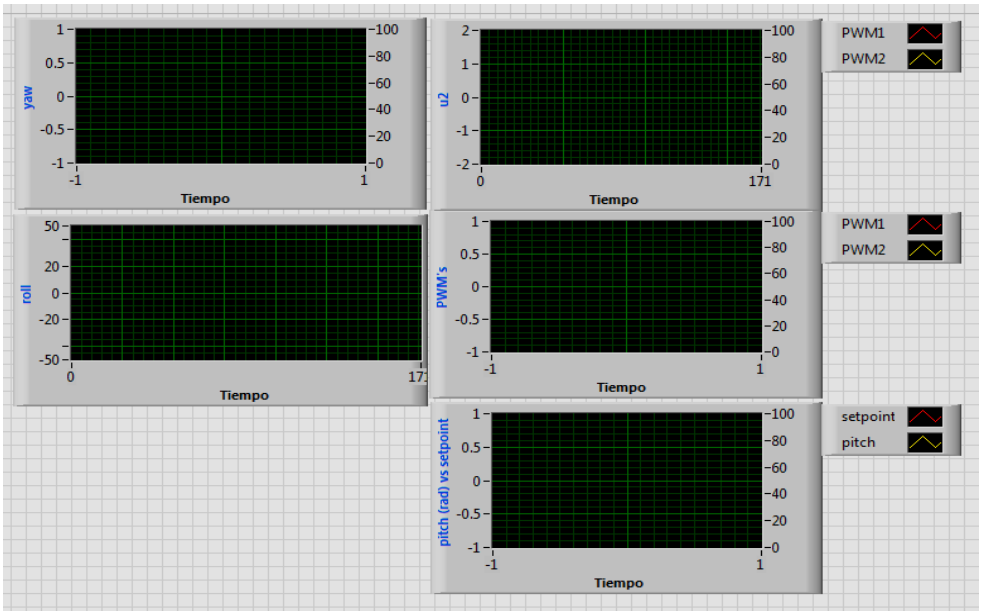

Figura 40. Graficas del sistema del PVTOL

CAPITULO IV
RESULTADOS

El sistema del PVTOL funciona perfectamente, él envió de datos inalámbricamente responde bien, la tarjeta IMU RAZOR 9DOF se posiciona perfectamente a la hora de recibir los datos del programa de LABVIEW.

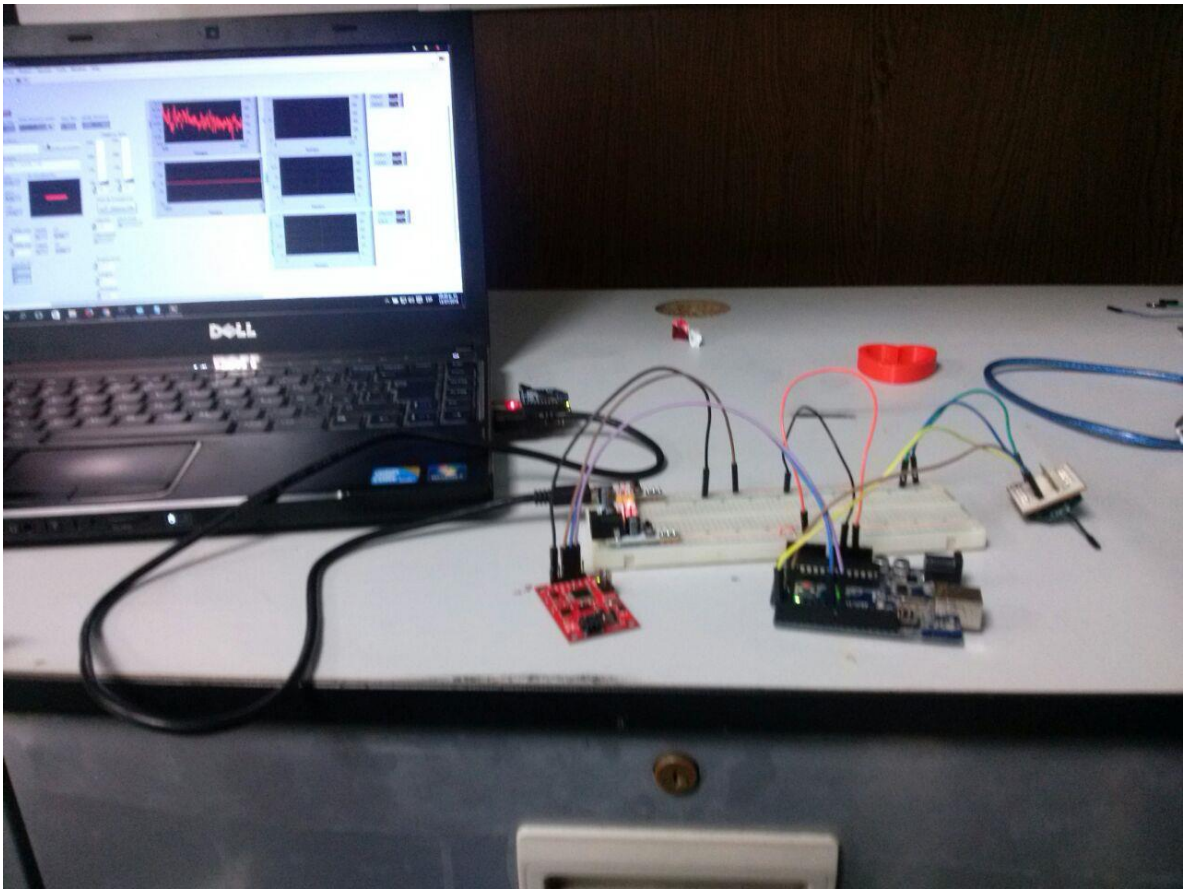

Figura 41. Verificando el programa de LABVIEW

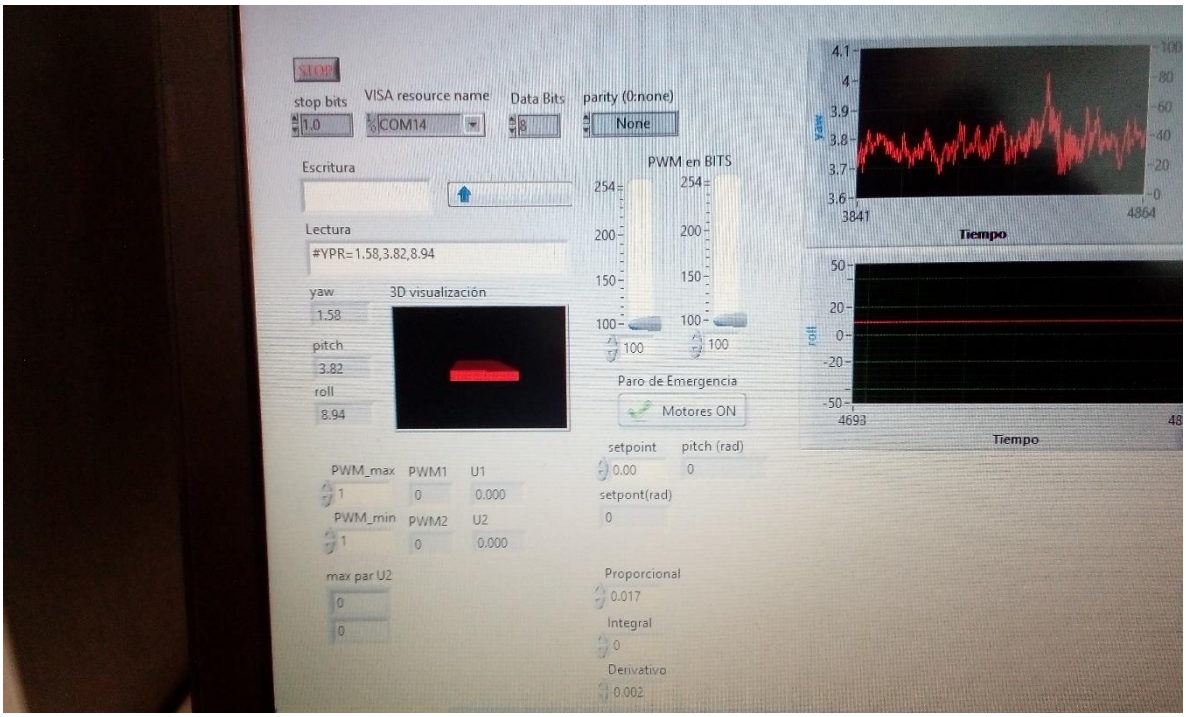

Figura 42. Observamos que el PVTOL está estable

CONCLUSIONES

Hacer un proyecto de residencia consiste en aplicar los conocimientos que a lo largo de la carrera nos enseñaron. Por ejemplo en nuestro proyecto se aplicó información acerca de un sistema de de aeronaves no tripulados. Para el control del PVTOL se necesitó conocimientos sobre sistemas de control como el uso de la tarjeta arduino entre otras más utilizadas, que fueron los encargados del control, estas se tuvieron que configurar, por medio de un programa, por lo que se aplicaron nuestros conocimientos en programación.

La elaboración de un programa en LABVIEW fue un poco complicada debido a que no conocíamos cómo hacer un PID.

Debido a que el PVTOL tenía que ser inalámbricamente se optó de hacer una comunicación entre dos tarjetas XBEE s2 para el envió de datos para que el usuario en una cierta distancia pueda controlar el PVTOL.

COMPETENCIAS DESARROLLADAS Y/O APLICADAS

Durante la elaboración de este proyecto, se entendió el funcionamiento de un sistema de control PID para el control del PVTOL. Adquirimos conocimiento sobre el programa de LABVIEW como comunicar y hacer el sistema de control, y el enlace entre los dos XBEE s2 de como enviar y recibir datos.

BIBLIOGRAFÍA

[1] Digi International Inc., «XCTU,» 2016. [En línea]. Available: http://www.digi.com/products/xbee-rf-solutions/xctu-software/xctu. [Último acceso: 25 Marzo 2016].

[2] Gerardo Ort T, «PVTOL PID controller,» [En línea]. [Último acceso: junio 2016].

[3] Gerardo Ort T, «PVTOL PID controller,» 2016. [En línea]. Available: https://www.youtube.com/watch?v=gXy5cxBjA1Q. [Último acceso: junio 2016].

[4] all rights reserved, «Mechatronics Engineering,» 2013. [En línea]. Available: http://mechatronics-rsz.blogspot.mx/p/pvtol.html. [Último acceso: mayo 2016].

[5] SparkFun Electronics, 2016. [En línea]. Available: https://www.sparkfun.com/products/10736. [Último acceso: mayo 2016].

[6] A. Z. 1. R. L. I. Fantoni, Diciembre 2002. [En línea]. Available: http://www.mit.edu/~esontag/FTP_DIR/02cdc-papers-refs-eds/810_FrP03-2.pdf. [Último acceso: mayo 2016].

[7] A. Z.-R. I. F. S. Daniela Juanita L´opez-Araujo, 17 julio 2010. [En línea]. Available: https://hal.archives-ouvertes.fr/hal-00503221/document. [Último acceso: abril 2016].

[8] M. M. C. N. M. T. Luca Consolini, julio 2010. [En línea]. Available: http://www.control.toronto.edu/people/profs/maggiore/DATA/PAPERS/JOURNALS/AUT10_2.pdf. [Último acceso: marzo 2016].

[9] Rodrigo Salazar Zugasti, «you tube,» [En línea]. Available: https://www.youtube.com/watch?v=eMmAQiAf5qo. [Último acceso: abril 2016].

[10] «imagenes,» [En línea]. Available: https://www.google.com.mx/search?q=PVTOL&espv=2&biw=1366&bih=623&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjF4qOMnKjOAhXn5IMKHTcoCpoQ_AUIBigB#imgrc=9dDOzEkJN4ZEkM%3A. [Último acceso: marzo 2016].

ANEXOS

Dentro de este CD se anexa lo siguiente

- Carpeta con el reporte del PVTOL.
- Programación con el programa Processing y Arduino.
- Programa del PVTOL en LABVIEW.