UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Ardupilot: piloto automático para aeromodelo con Arduino

Proyecto Final de Carrera

Enginyeria d'Informàtica en sistemes especialisació d'industrial

Autor: Josep Estarlich Pau

Director: Jose Vicente Busquets

30/09/2013

Resumen

Este proyecto esta compuesto por dos partes, pero que una sin la otra no tendríamos nada, una parte es el hardware, en el cual tendremos que discutir el tamaño del Arduino, el tamaño del Gps y todos los restantes componentes, y también su colocación; y por otra parte el software que necesitamos guiar a nuestro avión de forma autónoma mediante unos waypoints preconfigurados, corrigiendo su destino de vuelo mediante un control PID. Realizando en cada paso las pruebas oportunas, tanto de software como de hardware.

Por otro lado podremos comprobar que la teoría no siempre es lo que podemos llevar a la práctica, ya que el software puede funcionar en simulación pero después con el hardware no todo sale a la primera.

Palabras clave: *Arduino, microcontrolador, IMU, drone, A.P.M 2.0, UAV, autonomous unmanned aircraft, Atmel, Ardupilot, PID, Avión, Procesing, FTID, Piloto Automático.*

Tabla de contenidos

1. Introducción

Un vehículo no tripulado es todo aquel vehículo capaz de realizar diferentes tipos de tareas sin la necesidad de contar con un tripulante humano. Los vehículos aéreos no tripulados UAV (Unmanned Aerial Vehicle) han cobrado una gran importancia en la industria aeronáutica debido a que son capaces de cubrir un gran número de necesidades y posibilidades no exploradas hasta hace bien poco.

Aunque aquí nos centremos en vehículos aéreos también existen vehículos no tripulados en tierra y mar. Dentro de los vehículos no tripulados se encuentran vehículos controlados remotamente (Remotely Piloted Vehicle o Drones) y vehículos autónomos AV (Autonomous Vehicle).

En el caso de este proyecto, nos centraremos en UAV autónomos, capaces de realizar gran parte de las tareas sin la intervención de un humano.

Este tipo de vehículos son muy útiles en circunstancias en las que contar con un tripulante humano puede ser peligroso para su integridad, o en aquellos casos en los que por movilidad, maniobrabilidad o flexibilidad es imposible contar con un tripulante.

1.1. Evolución

Si echamos un vistazo a la evolución de los UAV, los primeros UAV eran aeronaves simples pilotadas remotamente a través de sistemas de radio control. Su uso comenzó a introducirse dentro del campo militar tras la Primera Guerra Mundial, usándose en la Segunda Guerra Mundial como blancos móviles para el entrenamiento de sistemas antiaéreos y de diferentes armas. A finales de los años 60 y debido al riesgo potencial de bajas de pilotos en misiones de reconocimiento en territorio enemigo, y debido al alto número de vuelos de este tipo que se realizaron durante la Guerra , los oficiales norteamericanos fijaron su interés en las posibilidades de los vuelos no tripulados. Esta necesidad se amplió con la entrada de Estados Unidos en la Guerra de Vietnam donde se produjo el primer uso de UAV en misiones de combate.

En los años 80 y 90 se produjo un gran avance en este tipo de vehículos, dotándolos de mayor capacidad autónoma. También los diferentes avances en las tecnologías hicieron posible el abaratamiento de costes, así como la posibilidad de crear vehículos más pequeños y versátiles.

Las capacidades autónomas han ido aumentando cada vez más, desde asistir en vuelo a un piloto situado en una estación remota, hasta el vuelo completamente autónomo siguiendo un determinado plan de vuelo precargado, y el seguimiento de un plan de misión avanzado con gestión de la carga de pago (conjunto de elementos y dispositivos destinados a adquirir datos de inteligencia).

Actualmente los UAV siguen siendo utilizados mayoritariamente en entornos militares, donde han conseguido hacerse un hueco como sistemas de vigilancia, inteligencia, reconocimiento y adquisición de blancos (ISTAR, Intelligence Surveillance Target Adquisition and Reconnaissance). Poco a poco también han ido introduciendose en aplicaciones civiles, sobretodo en los campos de vigilancia, seguridad, y reconocimiento.

1.2. Aplicaciones

1.2.1 Militares

Tal y como se ha adelantado en la introducción, la mayor parte de las aplicaciones de los UAV se centran en el campo militar. Dentro de las aplicaciones militares destacan la vigilancia y reconocimiento del entorno, incluyendo la detección y clasificación de posibles amenazas, reconocimiento y adquisición de blancos, obtención de un mapa completo geográfico del campo de batalla, e incluso el combate directo, aunque en este último caso siempre guiado y monitorizado por un piloto humano. De entre todas las aplicaciones, dos de ellas han sufrido un profundo avance en los últimos años:

Por un lado, las aplicaciones de vigilancia y reconocimiento de corto alcance, llevadas a cabo por pequeños UAV desplegados en campo por los propios soldados. Estos cuentan con una pequeña Estación de Tierra para recopilar datos y poder ser controlados en caso necesario. Estos UAV pueden comunicar la información casi en tiempo real, de manera que pueden generar un mapa tanto geográfico como de amenazas y blancos. Estos UAV son pequeñas aeronaves que funcionan generalmente con electricidad, que pueden ser desplegadas a mano por los soldados y que en algunos casos no retornarán nunca. Cuentan con baja autonomía, muy bajo coste y proporcionan una información muy valiosa a corto plazo.

Por otro lado se encuentran UAV de mayor tamaño usados tanto en misiones ISTAR como de combate. Estos vehículos en algunos casos tienen casi el tamaño de un avión de combate convencional, con autonomías que cubren grandes distancias y alturas. La información que proporcionan puede ser, en algunos casos, directamente usada por sistemas de artillería o incluso pueden contar con armamento embarcado.

Los UAVS ISTAR generalmente proporcionan información táctica muy valiosa que puede ser usada también a largo plazo. También pueden tener la capacidad de sincronizarse entre diferentes UAV para aumentar el valor de dicha información.

1.2.2 Civiles

El ámbito de aplicación de los UAV no solo se limita al área militar, poco a poco, comienzan a demandarse aplicaciones y servicios en el campo civil.

Los UAV civiles de largo alcance son utilizados en tareas de identificación de riesgos y vigilancia de zonas amplias en espacios de difícil acceso, como vigilancia de grandes instalaciones de canalización de gas, gasóleo, o similares. También en tareas de identificación de conatos de incendios en grandes zonas boscosas, incluso en los servicios de protección de fronteras, ayudando a las Fuerzas y Cuerpos de Seguridad del Estado en la identificación y seguimiento de embarcaciones que pretenden entrar ilegalmente en el país a través del mar, ya se trate de inmigración ilegal, o tráfico ilegal de diferentes sustancias o materiales.

Estos vehículos abaratan considerablemente el coste de las misiones de control y vigilancia, ya que antes de estos vehículos estas tareas eran llevadas a cabo por aviones pilotados o helicópteros, cuyo coste de adquisición, mantenimiento y entrenamiento, es muy elevado.

Los cuerpos de seguridad también comienzan a demandar pequeños UAV de corto alcance para tareas de vigilancia de perímetros, o incluso reconocimiento del campo de acción, por ejemplo, por parte de la Policía o Guardia Civil. Estos vehículos son similares a los usados en campo por el ejército, suelen tener un alcance limitado y un coste muy bajo, proporcionando una gran cantidad de inteligencia a los cuerpos desplegados en campo.

También existen otras aplicaciones civiles como servicios de fotografía o captura de vídeo aéreo, útil para conformar mapas actualizados de parcelas para la construcción o rehabilitación de inmuebles, así como otros usos dirigidos hacia el entretenimiento. Por ultimo, también se usan vehículos no tripulados en aplicaciones científicas de investigación, sustituyendo, por ejemplo, a globos sonda, incluso para el transporte de pequeñas cargas en trayectos cortos que serían muy costosos si tuvieran que ser realizados de otra manera.

Podemos decir que el campo de los vehículos no tripulados en general, y el de los aéreos en particular es un mercado que no termina de expandirse, en el que van apareciendo nuevos ámbitos de aplicaciones y usos constantemente.

En el ámbito de la observación de la tierra los UAV tienen múltiples aplicaciones y posibilidades en el mercado civil:

- Cartografía: realización de ortomapas y de modelos de elevaciones del terreno de alta resolución.
- Agricultura: gestión de cultivos.
- Servicios forestales: seguimiento de las áreas boscosas, control de incendios.
- Geología.
- Hidrología.
- Medio ambiente: estado de la atmósfera.
- Control de obras y evaluación de su impacto.
- Seguimiento de la planificación urbanística.
- Gestión del patrimonio.

1.2.3 Carga de pago

Se denomina carga de pago a aquella carga que es capaz de cargar la aeronave y que no es estrictamente parte de los sistemas de navegación y control de la misma. En el caso de los aviones de transporte de pasajeros o mercancías, la carga de pago serían los pasajeros y las mercancías a transportar, en el caso de un UAV, se denomina carga de pago a los sistemas y equipos que son usados para generar información de utilidad táctica o de inteligencia. Una característica muy importante de los UAV es el peso en carga de pago que es capaz de soportar la aeronave, ya que cuanto más carga de pago mayor número de sensores, cámaras, equipos de comunicaciones o armamento será capaz de soportar el UAV. Evidentemente, a mayor carga de pago, mayor consumo, por lo que los UAV con menor autonomía soportan menores cargas de pago.

En nuestro caso la carga pago será el Ardupilot, los sensores, la batería y los motores que le ponemos a nuestro UAV.

Entre los equipos más utilizados como carga de pago en los diferentes UAV contamos con todo tipo de sensores y equipos de medición, incluyendo cámaras fotográficas y de captura de video, cámaras de visión nocturna y con lentes capaces de captar diferentes espectros, sónar, radar, sistemas láser, etc. También equipos y computadores encargados del procesamiento de dichas señales de manera embarcada son parte de la carga de pago, así como el armamento, en caso de que lo hubiera, como por ejemplo, ametralladoras, cohetes y hasta misiles de diferentes alcances.

Los sensores y equipos con los que el UAV cuenta y que son usados por los propios sistemas de control y navegación del vehículo no son tomados en cuenta a la hora de definir la carga de pago del UAV. Entre estos sensores se pueden contar acelerómetros, giróscopos, completas unidades de medición inerciales, magnetómetros, barómetros, sistemas de posicionamiento tales como el GPS, y sistemas de comunicaciones de radio frecuencia. Estos equipos no forman parte formalmente de la carga de pago, aunque también proporcionen información que unida a la de los equipos de la carga de pago puedan generar información útil o inteligencia.

Sensor ultrasonido

Sensor de temperatura

Cámara

1.2.4 Estación de Tierra

En cualquier caso, además del vehículo no tripulado, es necesario contar con una estación en tierra o GCS (Ground Control Station) capaz de recibir la información que percibe el vehículo, y a través de la cual se pueda monitorizar y controlar toda la actividad del UAV.

En algunos casos, esta estación está formada por una estación fija, en otros por una unidad móvil. En el caso de los vehículos militares, lo más común es contar con una unidad móvil desplegable, desde la cual monitorizar y comandar al vehículo.

1.3. Estudio de Alternativas

Antes de entrar de pleno en el proceso de desarrollo es necesario realizar un breve repaso de los proyectos más significativos en el campo de los UAV, ya que gracias a ellos se pueden eliminar alternativas no válidas en el diseño, o solucionar problemas ya resueltos sobre los que no se desea profundizar demasiado.

Se pueden encontrar proyectos de vehículos de radiocontrol que cuentan con cierto grado de autonomía, entre estos, se encuentran proyectos de aeroplanos radiocontrolados con microprocesador embarcado para facilitar y mejorar el control de vuelo.

También vehículos de despegue vertical ya que el grado de autonomía del vehículo es variable en cada proyecto, y va desde pequeñas funcionalidades de auto-estabilización para facilitar el control, hasta la plena autonomía del vehículo desde el despegue al aterrizaje, pasando por vehículos radiocontrolados con capacidades de vuelo en primera persona (haciendo uso de cámaras) e implementando módulos de telemetría muy completos.

1.4. Elección de hardware

1.4.1 Mikrokopter

Mikrokopter es un proyecto no libre de origen alemán desarrollado por la empresa HiSystems con un estado muy avanzado de desarrollo.

Para controlar el Mikrokopter han utilizado un microprocesador Atmel ATMEGA644 a 200MHz como unidad central, la unidad inercial es un desarrollo cerrado propio. Las principales ventajas de este proyecto es que comenzó su desarrollo en 2006, por lo que tras 7 años de desarrollo tanto la plataforma hardware como el software son muy robustos y funcionales.

Además de la unidad inercial, cuenta con un módulo de navegación opcional con GPS, 3 magnetómetros y soporte de tarjetas de memoria Flash, gestionado por un microcontrolador ARM9.

La principal ventaja frente a proyectos similares es la capacidad de usar interfaces I2C para comandar potencia a los controladores de velocidad, ya que esto permite tasas de actualización del orden de 450Hz en la salida a los motores, lo que les permite tener un control muy _no sobre la potencia de los mismos, desembocando en un régimen de estabilización muy alto.

Además cuentan con la experiencia de haber realizado gran cantidad de pruebas con diferentes configuraciones, por lo que las características de potencia de los motores, distancia entre los mismos, tamaño de las hélices, capacidad de las baterías, así como otras características físicas del vehículo son aspectos a tener en consideración. Por lo menos para tener la seguridad de que es posible llegar al objetivo con hardware y configuraciones similares.

El estado de desarrollo es muy alto, cuentan con una estación de tierra muy completa, y el sistema es muy configurable, de modo que se puede desde radiocontrolar el vehículo apoyándose en su control de estabilización, hasta realizar un plan de vuelo completo con waypoints editables desde la GCS, pasando por la implementación de rutinas acrobáticas automáticas.

1.4.2. Aeroquad

Aeroquad es un proyecto open source de un UAV con las mismas características que Mikrokopter. Cuenta con la ventaja de que es open source y que además la plataforma hardware sobre la que se desarrolla también es abierta.

Aeroquad se basa en la plataforma Arduino para construir la unidad central (ATmega 328 a 16 MHz), sobre esta plataforma se profundizará en el capítulo de plataforma de desarrollo.

Aunque no cuenta con el nivel de finalización que tiene mikrokopter es un proyecto interesante en cuanto a que todo el código es accesible y los resultados obtenidos son muy buenos, si bien, es cierto que estos resultados dependen de la calidad de los sensores y circuitos utilizados, siendo necesario un alto desembolso económico para poder replicar el proyecto original.

Además, una de las desventajas de este proyecto es que, pese a ser un proyecto libre, intentan comercializar tanto con el hardware como con el soporte, por lo que es complicado obtener ayuda de la comunidad alrededor del mismo.

1.4.3. Ardupilot

Ardupilot es un proyecto open source dedicado a obtener una unidad de navegación inercial libre, basada en la plataforma Arduino. Esta unidad de navegación puede ser utilizada en diferentes proyectos, por un lado en aeronaves, barcos, coches radiocontrolados, y por otro en proyectos de vuelo en primera persona (radiocontrolado

pero sin visión directa del vehículo y controlándolo a través de cámaras instaladas en el vehículo), y por último en todo tipo de UAV, tanto de despegue vertical como tipo aeroplano.

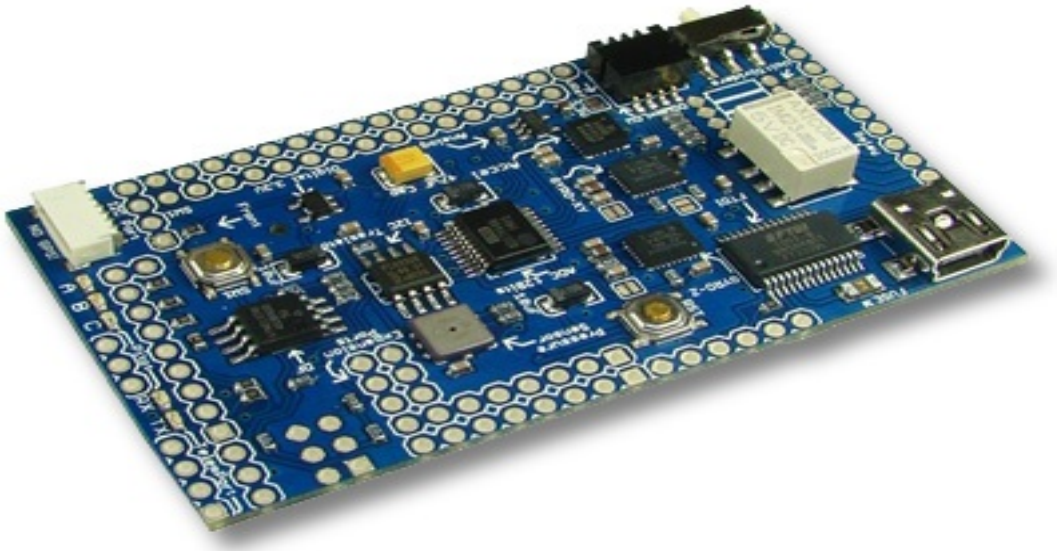
Es un proyecto maduro, que cuenta con una buena comunidad alrededor que comercializa las placas de control en diferentes niveles de integración. La ventaja principal del proyecto es que han resuelto muy bien el problema de la fusión de datos de diferentes sensores usando tanto filtros kalman avanzados, como algoritmos basados en matrices de cosenos directrices.

Cuentan en su comunidad con una verdadera eminencia en el desarrollo del algoritmo de Matrices de Cosenos Directrices como W. Premerlani , y cuentan con un código muy optimizado. Por lo que este proyecto es una gran fuente de conocimiento en el caso de tener problemas con este tipo de algoritmia.

Existen diferentes proyectos basados en el código de fusión de datos de Ardupilot. El proyecto pertenece a una comunidad aún mayor llamada DIYDrones (Do It Yourself Drones) orientada al desarrollo de UAV a pequeña escala basados, sobretodo, en aeroplanos.

En vista de las posibilidades que se ven con el hardware Ardupilot, decidimos escogerlo ya que por su pequeña estructura, como su poco peso nos dejaban margen para añadir GPS y un sensor ultrasonido, ya que si poníamos el Arduino Mega nos quedábamos sin espacio en el avión y realmente teníamos muchos conectores sin usar, y otra ventaja que disponemos es para colocar los servos y comunicarlos con el receptor de radiofrecuencia, porque en Ardupilot el hardware es más complejo y no hace falta protoboard.

1.4.4 ArduPilot Mega IMU:



El ArduPilot Mega Inertial Meassurement Unit o APM IMU, es un dispositivo electrónico que contiene los sensores necesarios para la estabilidad del UAV, entre sus características más resaltantes se encuentran:

- Relé de cámaras, luces o cargas
- ADC de 12-bits para una mejor resolución del giroscopio/ Acelerómetro / y
- Sensor de Aire.
- Posee interruptor DIP para revertir los servos o programarlos para otra función, delimitada por el usuario.
- Puerto de entrada I2C que le permite construir matrices de sensores.
- Puertos analógicos de expansión de 10 bits.
- Botón de reinicio.
- Relación de Sensores dentro del IMU:
- Giroscopio XY: InvenSense IDG 500
- Giroscopio Z: InvenSense ISZ 500
- Acelerómetro de tres ejes: ADXL335
- Sensor de Presión Absoluta Bosch: BMP085

1.5. Hardware empleado

Una vez decidido que vamos a usar ardupilot, hay que completer el hardware que vamos a usar en este PFC.

1.5.1 Comunicación

Estábamos dudando en usar bluetooth o radiofrecuencia, pero se ha podido comprobar que con el bluetooth no tenemos suficiente alcance, por lo tanto usaremos el modulo bluetooth para la comunicación serie, para programar el ardupilot, y la radiofrecuencia para tener contacto con la maqueta del avión.

Modulo bluetooth (JY-MCU)

Emisora de radio frecuencia

La emisora es de 7 canales, en la banda de 2,4 GHz, tiene indicador de batería, también es programables desde el pc, para configurar y ajustar el avión, e incluso conectar dos emisoras entre si.

1.5.2 FTDI

Los módulos FTDI que estudiaremos son los mismos puentes USB-UART que hemos usado en otros proyectos con Arduino. Son los comercialmente conocidos como conversores USB a Puerto Serie, solo que ahora los veremos en su desempeño como programadores de AVR.

Bueno, en realidad no hay ningún programador llamado como tal. Denominamos así a los módulos de comunicación basados en el transceiver FT232R o similar. Este chip es un conversor USB-UART fabricado por FTDI chip. De ahí su apelativo, aunque ello no quita la posibilidad de usar este método con un transceiver de otra marca.

El conversor USB Serial Light es una tarjeta basada en un microcontrolador ATmega8U2 programado para emular un puerto serie. El AVR funciona también como el FT232RL aunque a nivel software los drivers con que trabaja no tienen la misma consideración que los de FTDI chips.

Los tres adaptadores son muy parecidos. Lo que nos interesa a fin de cuentas son las 6 señales de salida que proveen. Las 5 primeras son las mismas: GND, CTS, VDD, TXD y RXD (los nombres cambian un poco pero son las mismas). La sexta señal es DTR en el FDTI Basic y USB Serial Light pero RTS en el FTDI Friend. Esa es la diferencia crucial. Como puente de comunicación para transferencias de mensajes con la computadora funcionan igual pues allí solo intervienen TXD y RXD además de VDD y GND claro está.

1.5.3 Entorno de programación

El entorno que vamos a elegir para programar nuestro Ardupilot será Arduino, por la gran información que tenemos sobre este, y la anterior utilización en otros proyectos, porque este entorno hace fácil la programación y directa.

Existen versiones para Mac que es lo que nos interesa, hay que elegir la más actualizada ya que en todas no nos dejaba programar el Ardupilot.

En este entorno es muy sencillo configurar los puertos con la placa, para poder programarla, y poder establecer una relación de pregunta y respuesta mediante el puerto serie y el monitor serie que dispone este entorno de programación.

1.5.4 Servos y Motor

Un servomotor de modelismo es un dispositivo actuador que tiene la capacidad de ubicarse en cualquier posición dentro de su rango de operación, y de mantenerse estable en dicha posición. Está formado por un motor de corriente continua, una caja reductora y un circuito de control, y su margen de funcionamiento generalmente es de menos de una vuelta completa, normalmente si no esta trucado de unos 180º.

Un motor eléctrico sin escobillas o motor brushless es un motor eléctrico que no emplea escobillas para realizar el cambio de polaridad en el rotor, que hace que dispongan de mayor fiabilidad, mayor rendimiento y menor mantenimiento.

1.5.5 Módulo GPS

El módulo GPS que tratamos en este caso es el modelo SKM53 del fabricante SKYLAB. Se trata de un dispositivo de alta sensibilidad y antena integrada, que podemos encontrar fácilmente en el mercado a un coste por debajo de los 20 €.

Mide tan solo 30 x 20 x 8,5 (Ancho x Alto x Grosor) y mediante una sencilla comunicación serie a **9600 bps** al microcontrolador, o lo que significa lo mismo; únicamente dos hilos, Rx y Tx, nos ofrece toda la potencia y las características de módulos destinados al mismo uso, de costes más elevados.

Como podemos ver a continuación dispone de una tira de 6 pines macho:

Solo nos interesan en principio 4, dos destinados a la comunicación serie más los destinados a GND y VCC, resumidamente una descripción de las conexiones sería:

RXD : Línea de entrada de datos de la conexión serie. Conectaremos este pin al controlador o sistema host en su línea de transmisión o TXD.

TXD : Línea de transmisión de datos. Este pin debe ir conectado al de recepción de datos en el microcontrolador. La conexión seria emplea niveles TTL, por lo que no es necesaria su conversión al emplearlo en muchos sistemas microcontrolados.

GND : Masa.

VCC : En este pin conectamos la alimentación, a una tensión de 5 Voltios.

1.5.2 Batería

La batería que usaremos será de tantos voltios y tantos amperios y de un tamaño razonable para que no pese y no ocupe demasiado para que nuestra maqueta de avión de tenga ningún problema en alzar el vuelo.

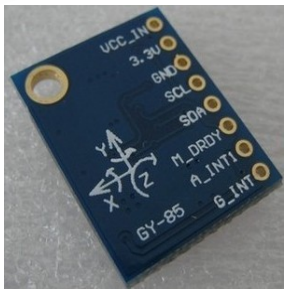
11.1 Vol 1100 mAh 15C

Un consejo que hemos leído en unos foros es que no se debe dejar ni completamente cargada ni descargada cuando se va a guardar, y tampoco es bueno desgastarla del todo, ya que pierde eficiencia.

1.6 Modelo de avión

Nuestra maqueta elegida es skyartec AVIÓN RC CESSNA 182 dispone de una bandeja donde van los servos suficiente grande para meter el Ardupilot, y otro compartimento para la batería con motor brushless y batería incluida.

1.7. Acelerómetro y compás



Nueve-ejes tiene el módulo (giroscopio de tres ejes + acelerómetro triaxial + campo magnético de tres ejes).

Chip: ITG3205 + ADXL345 + HMC5883L

Fuente de alimentación: 3-5V

Comunicación: Protocolo de comunicación IIC

1.8. Motivación

La motivación de este proyecto final de carrera viene dado a las ganas de realizar una programación de un hardware que puedas comprobar aquello que has programado.

También al venir de realizar la especialidad de industrial, y realizar un pequeño proyecto con arduino, me quede con ganas de aprender mucho más y entre un compañero y yo, decidimos escoger este tipo de proyecto ya que aplicaríamos todo aquello que ya sabemos e incluso aprender cosas que realmente no pensábamos que se usaban para este tipo de aplicaciones.

2. Objetivos

2.1. Descripción del Problema

El objetivo de este Proyecto Fin de Carrera es la definición, diseño e implementación de un UAV de corto alcance, con características de realizar un vuelo no tripulado con unas coordenadas ya preconfiguradas.

En este apartado se realizará primero una descripción general del objetivo a resolver. A continuación se resumirá la colección de requisitos impuestos al proyecto, después hacer un diagrama de componentes para, por último, definir la metodología utilizada durante todo el proceso de desarrollo.

El proyecto en cuestión se divide en tres partes o subobjetivos bien diferenciados:

1. Localización : Mediante un módulo GPS habrá que localizar el punto inicial del avión, y llevarlo a otros puntos que han sido previamente configurados, (para configurarlos utilizaremos el bluetooth por puerto serie, para no tener que desmontar el avión para cada prueba).
   Esto incluye seleccionar los dispositivos que queremos integrar, y el estudio de la comunicación bluetooth.

2. Estabilización: Implementaremos e integraremos el software que se ejecutará en el microcontrolador (Ardupilot --> ATMEGA 2560) del vehículo. Aquí incluiremos el acelerómetro, el compás, y lo que refiere a la parte mecánica del avión como son lose servos y el motor.

3. Utilización de processing para la visualización gráfica del hardware adicional como son el compás y el GPS.

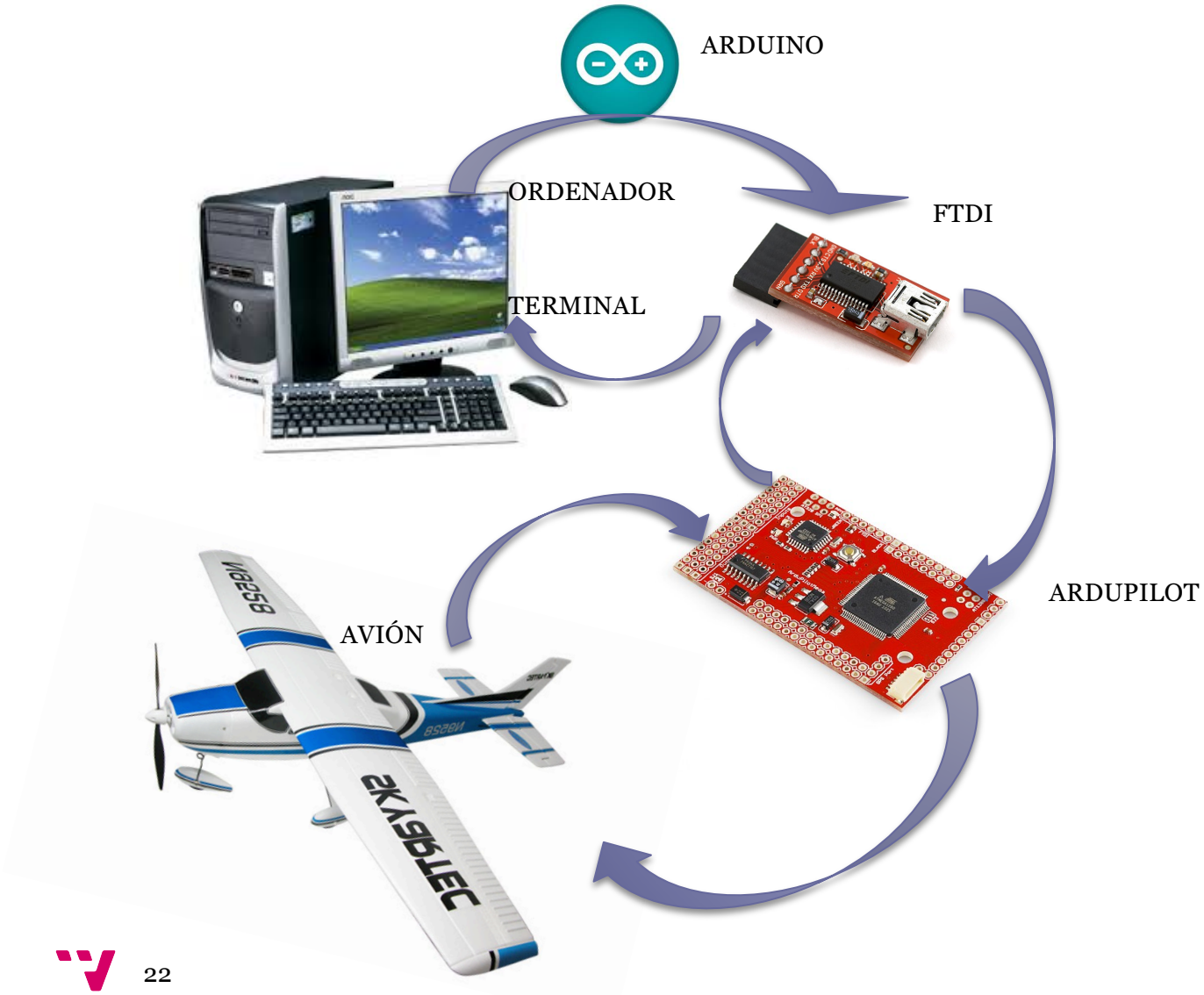
Estas tres partes diferenciadas suman características y dificultades muy diferentes, que abarcan desde la integración de componentes electrónicos, motores, baterías, sensores, etc., a la implementación de protocolos de comunicaciones por bluetooth, algoritmos que sean capaces de procesar la información de los sensores de modo que pueda ser usada por algoritmos de estabilización, la implementación de esos mismos algoritmos de estabilización, pasando por el diseño e implementación de interfaces gráficas amigables y útiles.

Sin olvidar la costosa tarea de integración, depuración y pruebas de toda la plataforma que conforma el UAV, que por supuesto ha de ser realizada con rigor.

Aunque el objetivo principal es lograr que el vehículo resultante de este PFC sea funcional, el segundo objetivo, y no menos importante, es estudiar y poner en práctica todas las fases de desarrollo necesarias para la implementación de un proyecto de Ingeniería tan completo. Es decir, analizar las diferentes fases que conllevan este tipo de proyectos, los problemas encontrados, las diferentes maneras de resolverlos, los costes de desarrollo, la capacidad necesaria para poder desarrollarlos, y al final, estudiar y concluir, sobre el resultado obtenido, las características y peculiaridades de este tipo de proyectos.

Al tratarse de un proyecto con una complejidad muy elevada, ha sido necesario poner un límite al proyecto estableciendo unas características y funcionalidades bien delimitadas de modo que sea posible implementarlas en el marco de tiempo de un proyecto fin de carrera.

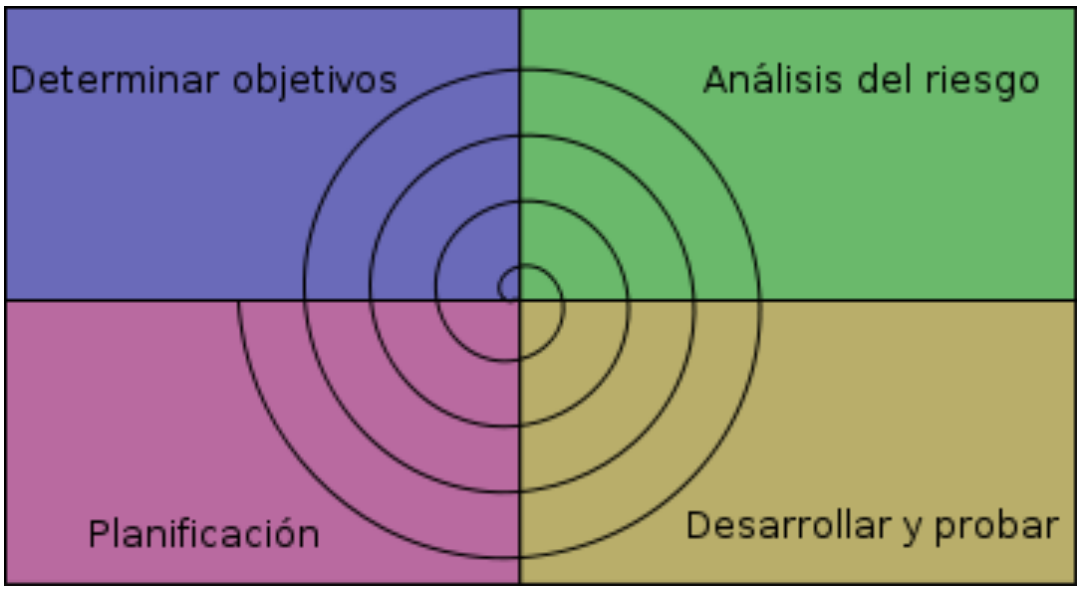
2.2. Diagrama de componentes



En este diagrama podemos comprobar lo esencial que necesitamos, ya que si adquirimos directamente un programador evitaremos muchos de los problemas que obtuvimos al principio en este proyecto, también la plataforma arduino la esencial para programar nuestro ardupilot ya que al intentar programarlo en otros nos daba errores en los códigos que realmente estaban bien, también podemos apreciar la comunicación que existe entre el ardupilot y el ordenador, ya que el ordenador programa mediante el FTDI (el programador) metemos el scketch en el ardupilot, y el se encarga de monitorizar el avión, los servos los sensores, ... y recíprocamente se comunica por el puerto serie con el terminal para recibir los printf creados en el programa para comprobar que todo esta funcionando correctamente, esto en vuelo vimos la opción de realizarlo con bluetooth pero no tiene suficiente alcance.

2.3. Metodología

La metodología a utilizar en el desarrollo de este proyecto, será iterativa y creciente, ya que la idea principal detrás de mejoramiento iterativo es desarrollar un sistema de programas de manera incremental, permitiéndonos sacar ventaja de aquello aprendido a lo largo del desarrollo anterior, incrementando, versiones del software como del harware. El aprendizaje viene de dos vertientes: el desarrollo del programa que controlara nuestro Ardupilot, y su prueba tanto simulada como en el harware. Los pasos claves en este proceso son comenzar con una implementación simple de nuestro UAV, es decir que sea capaz de realizar funciones esenciales, e iterativamente mejorar la secuencia evolutiva de versiones hasta que nuestro UAV consiga realizar todo aquello que esperamos de este PFC. En cada iteración, se realizan cambios en el diseño y se agregaran nuevas funcionalidades y capacidades a nuestro UAV.



El proceso en sí mismo consiste de dos etapas y una lista de control:

2.3.1. Etapa de inicialización

Crearemos una versión de nuestro UAV. La meta de esta etapa es crear un software con el que podamos interactuar, y retroalimentar el proceso. Debe ofrecer una muestra de aquello que queremos que realice nuestro UAV y un hardware lo suficientemente simple para ser comprendido e implementado fácilmente. Para guiar el proceso de iteración se puede crear una lista de control de proyecto, que contiene un historial de todas las tareas que necesitan ser realizadas. Incluye cosas como nuevas funcionalidades para ser implementadas, y áreas de rediseño de la solución ya existente. Esta lista de control se revisa periódica y constantemente como resultado de la fase de análisis.

2.3.2. Etapa de iteración

Esta etapa involucra el rediseño e implementación de una tarea de la lista de control de proyecto, y el análisis de la versión más reciente de nuestro UAV. La meta del diseño e implementación de cualquier iteración es ser simple, directa y modular, para poder soportar el rediseño de la etapa o como una tarea añadida a la lista de control de proyecto. El código puede, en ciertos casos, representar la mayor fuente de documentación de nuestro UAV. El análisis de una iteración se basa en la retroalimentación del hardware y en el análisis de las funcionalidades disponibles del programa. Involucra el análisis de la estructura, modularidad, usabilidad, confiabilidad, eficiencia y eficacia (alcanzar las metas). La lista de control del proyecto se modifica bajo la luz de los resultados del análisis.

3. Planificación

3.1. Empezar a programar con Arduino

En este apartado empezaremos a programar en el entorno arduino para familiarizarnos en el tipo de código, empezaremos programando algoritmos sencillos, como encender un led, estas pruebas las realizaremos con el Arduino Mega, ya que aun no disponemos del ardupilot.

3.2. Aplicar la programación a Ardupilot

En este proceso lo que haremos es trasladar aquellos pequeños squetch creados en arduino y cargarlos en la placa Ardupilot, aquí nos veremos con algunos problemas como son el programador RFID, que intentamos programar con un usbasp pero sin conseguir nada, también hay que tener cuidado que la versión de arduino este actualizada ya que en algunas no funciona la programación de Ardupilot.

3.3. Estudiar el modulo GPS

En este paso nos informaremos de las tramas que nos envía el módulo Gps, como la comunicación que tenemos que hacer con él, haremos un algoritmo sencillo que andando podamos comprobar la efectividad de este, con este estamos teniendo bastantes problemas de precisión.

3.4. Estudiar el método de control PID

En este procedo nos informaremos de este método ya que nos ayudará a la corrección del rumbo entre waypoints, este método tendremos alguna facilidad por haberlo usado ya en un pequeño proyecto realizado con un robot programado con sensores.

3.5. Estudiar la estabilización

En este otro apartado nos informaremos de la estabilización de la avioneta, para que sirven los flaps, como influye el aire en el vuelo, y estudiaremos el movimiento de los servos, y como decirle al avión que tiene que hacer, es decir corregir el rumbo.

3.6. Acelerómetro y compás

En este proceso haremos un pequeño algoritmo para ver el funcionamiento del acelerómetro y el compas, que son los que nos ayudaran en la estabilización del avión.

3.7. Crear los diagramas del programa

En este proceso es donde crearemos la estructura de nuestro programa, tanto lo que se refiere al setup como el loop.

3.8. Programación

En este proceso juntamos todos los programas creados individualmente con algunas modificaciones, para construir el código final.

3.9. Montaje de todos los componentes

En este proceso estructuraremos el avión de tal forma que todo entre en el avión, sin olvidar la batería, y hacerle alguna modificación a la maqueta.

3.10. Simulación

En este proceso intentaremos hacer dos partes, una simulación por software, y otra simulación sin soltar el avión de las manos, para evitar más percances.

3.11. Pruebas finales

En este proceso haremos un vuelo de prueba para comprobar el fruto de nuestro PFC.

Haremos pruebas de hardware, software y simulación; probando todos los componentes para que en el vuelo no tengamos ningún error ni ningún percance, ya que en el vuelo es muy fácil tener accidentes, para evitarlos hay que tener en cuenta el tiempo, como el aire y el espacio en el que se realiza el vuelo.

3.12. Presupuesto

Elemento	Descripción	Precio
Maqueta Avión	Soporte carga de pago	160€
Radio de RF	2Ghz	50€
Ardupilot	Atmel 2560	69€
FTDI	Programador	3€
GPS	9600	21€
1 Servo	Para los flaps que se rompió	5€
Hélices	Se rompen muy fácilmente	2 x 3€ = 6€
Acelerómetro/compas	Gy-85	**20€**
Total		334€

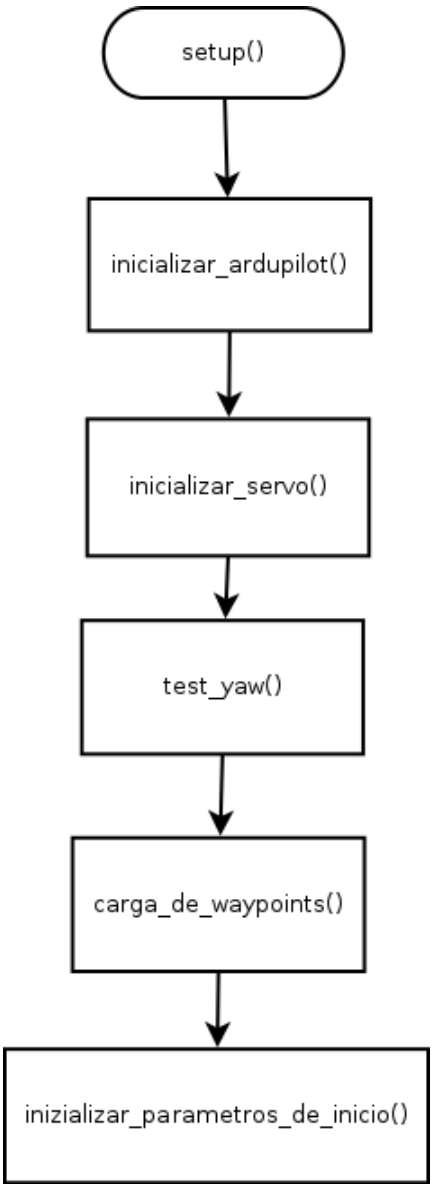
4. Análisis

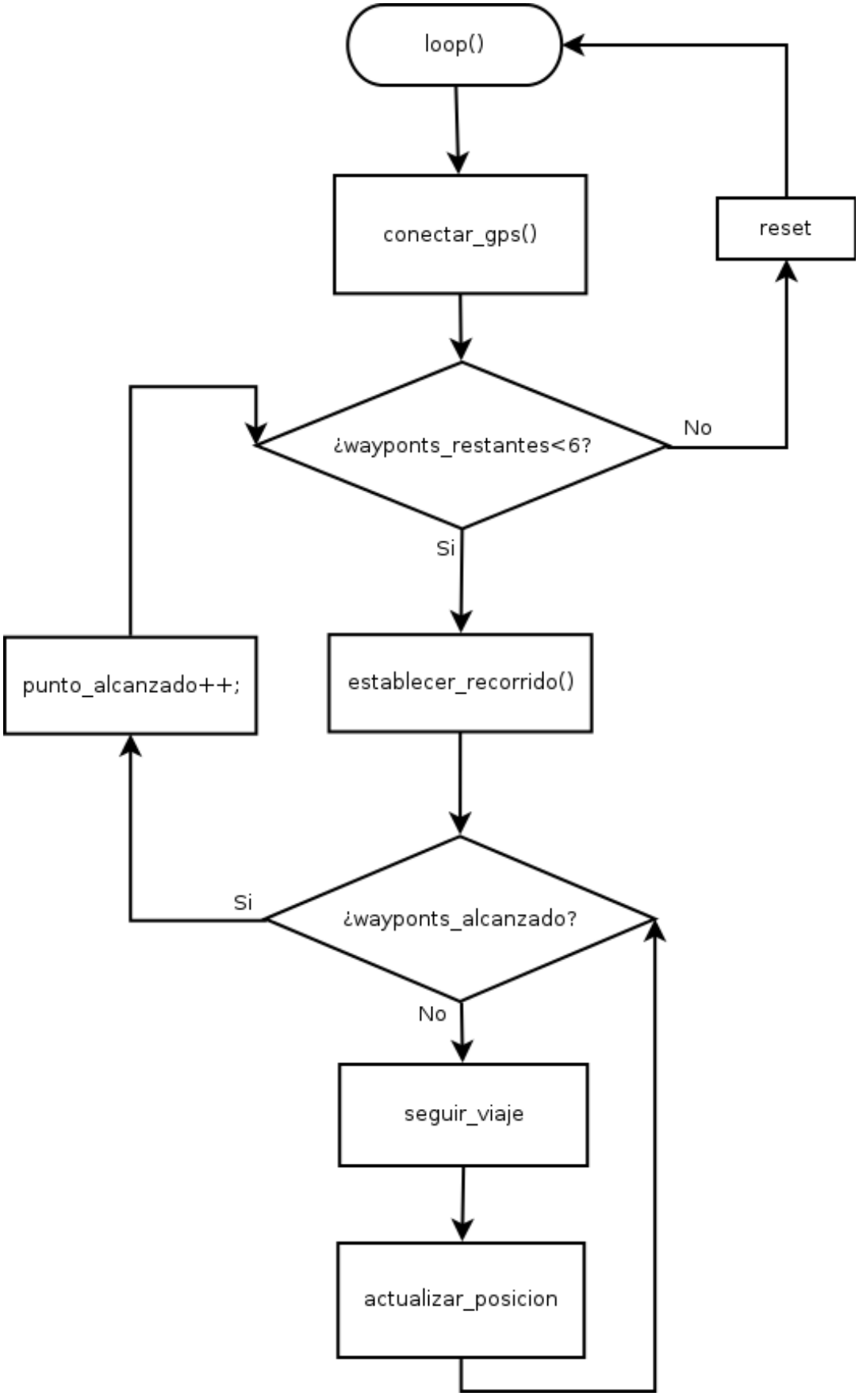
4.1. Casos de uso

Como ya hemos dicho en la introducción existen dos tipos de caso de uso que se podría decir que son en lo profesional “Militar”, para calcular territorios, buscar puntos débiles de los enemigos, etc.; o como Hobbie “civil”, que es en lo que va enfocado este proyecto, para estudiar los pasos que hay que seguir para construir un UAV, y descubrir nuevas técnicas de programación como comprobar el avance del hardware y del software que se utilizan para estos casos, y también los métodos tan utilizados como los miles de sensores utilizados en este campo tan inmenso.

En nuestro caso usaremos este proyecto para aprender a programar distintos tipos de hardware, y poder ver el fruto de nuestra programación funcionando mecánicamente.

4.2. Diagrama de flujo

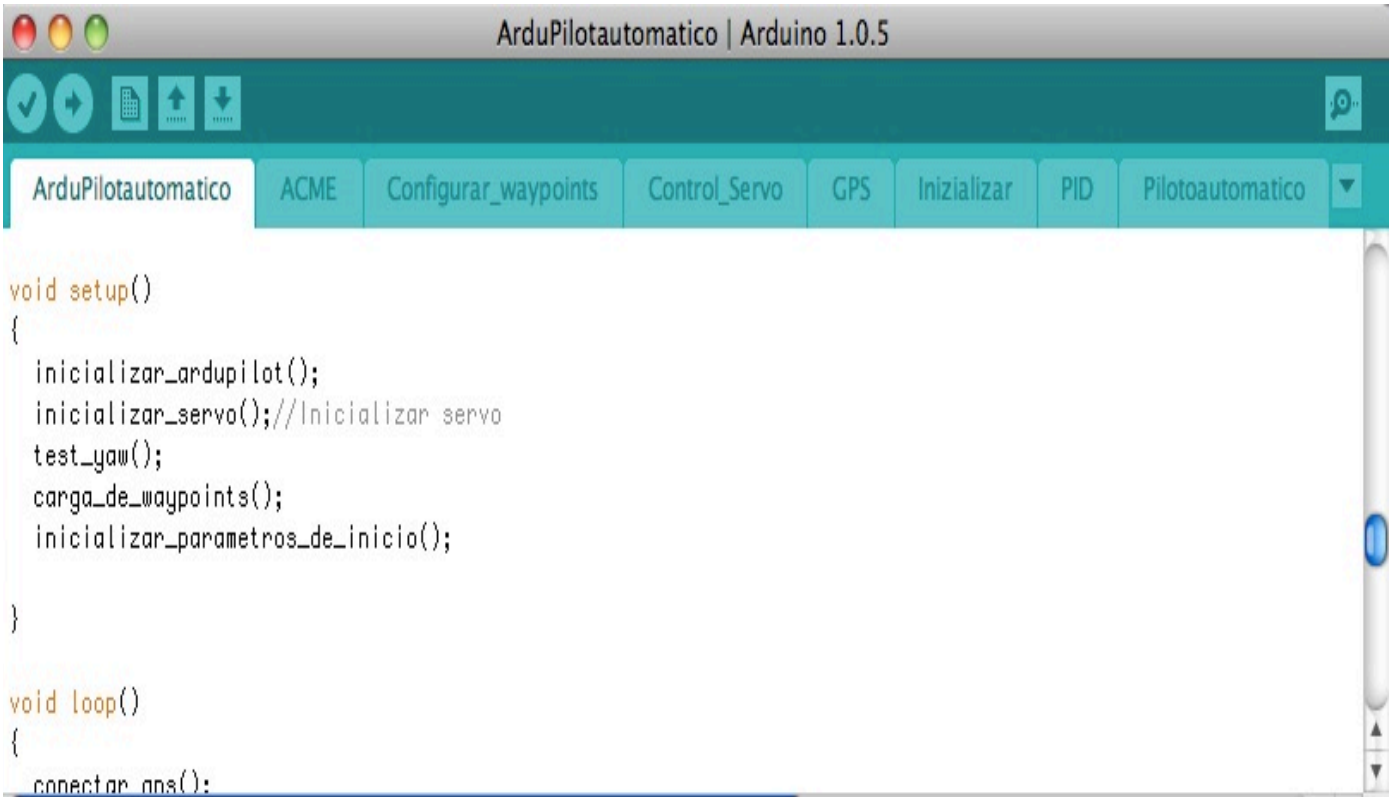



loop()
conectar_gps()
reset
¿wayponts_restantes<6?
No
Si
punto_alcanzado++;
establecer_recorrido()
Si
¿wayponts_alcanzado?
No
seguir_viaje
actualizar_posicion

5. Diseño

5.1. Estructura

Vamos a dividir nuestro código en distintos ficheros para la fácil revisión de errores y lectura del código:



5.1.1. Ardupilotautomático

En este fichero declararemos las librerías necesarias, como todas las variables globales necesarias para nuestro código.

En este fichero también encontraremos el setup() que se encargará de inicializar todas las variables como algunas funciones, y el loop() que es el bucle de nuestro programa, todo lo que esté aquí dentro estará repitiéndose indefinidamente, o hasta acabar su trabajo, aquí se irán ejecutando las funciones de los otros archivos que estén dentro de este.

5.1.2. ACME

En este fichero lo usaremos por una parte para imprimir todas aquellas variables que queremos ver por el monitor serie, para comprobar si funciona todo, tanto los waypoints como las variables del GPS, y también como se trata del fichero del acelerómetro, lo usaremos para poner la función que calcula el error del rumbo y ayuda a elegir el camino más corto para coger el rumbo deseado.

También un método para intentar mantener la altura.

5.1.3. Configurar_waypoints

Este fichero es muy sencillo, es en el que declararemos cada waypoint de destino poniendo la latitud, longitud y altitud.

```
wp_lat[1]= 40.488408;
wp_lon[1]= -3.919575;
wp_alt[1]=120;
```

5.1.4. Control_Servo

En este fichero declararemos en que pines vamos a conectar nuestros servos, necesitaremos dos servos para la cola y la dirección de la aleta, y otros dos para las alas y los flaps.

Se declaran de esta forma, en el pin 10.

```
digitalWrite(10,LOW);
pinMode(10,OUTPUT);
```

Aquí tendremos unas funciones que nos ayudaran a girar los grados correctos para ayudar a corregir el rumbo del avión.

5.1.5. GPS

En este fichero tendremos un método que será para entablar conexión con el GPS, y descifrar aquello que nos envía, gracias a unas librerías sacaremos el código necesario para separar aquella información que nos interesa de la que no.

Aquí tendremos dos funciones muy necesarias en este PFC, que son la que nos calcula el curso que tiene que seguir el avión, y la que nos calcula la distancia que falta para llegar al siguiente waypoint.

5.1.6. Inicializar

En este otro fichero declararemos los pines necesarios, como también configuraremos la velocidad a la que se comunicara nuestro ardupilot, con el monitor serie, y la de nuestro modulo GPS, que en los dos casos los configuraremos a 9600, porque el GPS trabaja a esta frecuencia, y porque el ardupilot si trabajase a una frecuencia menor el sistema seria muy lento.

5.1.7 PID

En este fichero vamos a poner distintos métodos que nos calcularán el error que existe, respecto a la formula que existe en este método, y también el error del PID respecto a la altura, y otro método más sencillo pero no menos importante el de reset que pondrá todas las variables utilizadas a 0.

Este método es uno de los más importantes ya que nos ahorrara trabajo para la realización de nuestro PFC, ya ha sido usado por nosotros en un proyecto de robótica, pero nunca lo hemos usado en vuelo por ello la importancia de este.

5.1.8 Pilotoautomatico

En este fichero tendremos un método que nos almacenará la posición inicial, tanto la altitud, la latitud y la longitud, otro método que nos controlara el motor que hace mover el avión regulando con otros métodos, y el método más importante el que guiará el avión durante el viaje y llamará a las respectivas funciones para que toda la navegación sea segura, y en el tiempo mínimo.

5.2. Processing

Tenemos pensado construir pequeñas aplicaciones para poder comunicarnos con el ardupilot, en cierto modo se podría decir que es como una pequeña simulación de aquello que veremos en la realidad.

Queremos realizar:

1. Un programa que mediante el compás simule una brújula.
2. Un programa que mediante el Gps localice las coordenadas
3. Un programa que controle la Radio

6. Implementación

6.1 Librerías

En este apartado, podemos decir que vamos a poner todas las librerías que nos han ayudado en el código, es decir que probablemente no hemos añadido la librería en si porque realmente solo necesitábamos unas funciones que tenía la librería y decidimos sacar las funciones necesarias y ponerlas con las modificaciones pertinentes, y no usarla sin mas, para así poder aprender de la función.

6.1.1. avr/eeprom

El microcontrolador de la placa Ardupilot tiene 512 bytes de memoria EEPROM, cuyos valores se mantienen cuando la placa está apagada (como un pequeño disco duro). Esta librería te permite leer y escribir esos bytes.

Pero tal cual lo usamos en nuestro proyecto este archivo de encabezado declara la interfaz para algunas rutinas sencillas de biblioteca adecuados para el manejo de la memoria EEPROM de datos contenidos en los microcontroladores AVR. La aplicación utiliza una interfaz de modo de sondeo simple. Las aplicaciones que requieren acceso EEPROM controlado por interrupción para garantizar que no se desperdicia tiempo en spinloops tendrán que implementar su propia implementación.

Todas las funciones de lectura / escritura primero asegúrese de que la EEPROM está listo para ser visitada. Ya que esto puede causar grandes retrasos si una operación de escritura se encuentra pendiente, aplicaciones de tiempo crítico debe primero sondear el ejemplo EEPROM usando eeprom_is_ready () antes de realizar cualquier E / S real Sin embargo, estas funciones no se esperan hasta SELFPRGEN en SPMCSR se convierte en cero. Hacer esto de forma manual, si su software contiene la quema de Flash.

Como estas funciones modifican registros IO. Si alguna de estas funciones se utilizan por el contexto estándar y de interrupción, las aplicaciones deben garantizar la protección adecuada (por ejemplo, mediante la desactivación de alarmas antes de acceder a ellos).

Todas las funciones de escritura fuerzan el modo de programación read_and_write.

Para XMEGA la dirección de inicio EEPROM es 0, al igual que otras arquitecturas.

6.1.2. TinyGPS

TinyGPS está diseñada para proporcionar la mayor parte de la funcionalidad GPS NMEA -posición, fecha, tiempo, altitud, velocidad y curso -.Para mantener bajo el consumo de recursos, la biblioteca evita cualquier dependencia de punto flotante obligatoria e ignora casi todos los campos clave de GPS.

De esta librería hemos sacado el código que nos ayudaba a descifrar toda la información que nos proporciona el GPS, como para entablar conversación con él.

6.1.3. Ultrasonic

Esta librería es compatible con todas las placas Arduino, nos ayuda a medir distancias a través de ultrasonidos HC-SR04.

Esta librería es bastante sencilla, con esta tuvimos un problema al principio ya que usábamos una librería desactualizada, y si no había objeto cerca que detector se quedaba pillado el programa, hasta que nos dimos cuenta que teníamos que limitar a menos segundos el pulso de entrada, y así para comprobar si tenemos objeto solo perdemos 30ms.

6.1.4. PID

Esta librería no la usaremos como tal, pero extraeremos código de ella, como será la forma de calcular el valor de "error" como la diferencia entre una medida de proceso variable y un punto de ajuste deseado.

Esta librería nos ayuda a crear en nuestro avión un controlador proporcional-integral-derivativo.

Las funciones de las que dispone son:

PID(),Compute(), SetMode(), SetOutputLimits(), SetTunings(), SetSampleTime(), SetControllerDirection()

6.1.5. Wire

Esta biblioteca le permite comunicarse con dispositivos I2C/ TWI. En las placas Ardupilot, la SDA (línea de datos) y SCL (línea de reloj) se encuentran en los cabezales de pin cercanos al GPS.

Las funciones destacadas que usaremos son *write(),* y *read()*.

6.1.6. HMC5883L

El uso de un magnetómetro puede ser un poco difícil, especialmente si no se saben las fórmulas a utilizar para obtener la orientación correcta y cuando otros objetos magnéticos están interfiriendo con su señal.

Por ello utilizaremos esta biblioteca ya que es compatible con muchas placas de interfaz HMC5883L de muchos fabricantes.

Es muy sencillo ya que sus funciones son para crear una instancia de compás, ReadRawAxis () con esta leemos los grados, y después se eligen por x, y e z. Y otra función para escalarlos que es ReadScaledAxis ().

7. Matización del código

7.1. Programación respecto a los servos

Los servos para programarlos en ardupilot son parcialmente distinto a arduino, por lo tanto vamos a hacer programas sencillos para poder mover los servos con una batería y comprobar si el funcionamiento es el deseado, tenemos que tener cuidado. Primero leer el data sheet para ver la información sobre los pines.

Vamos a estudiar un ejemplo de un programa no muy sencillo para mover los servos:

```
#include <avr/interrupt.h>
int All_PWM=1500;
int PWM_SEQ[4][8] = {
  { 2000,2000,2000,2000,2000,2000,2000,2000},
  {1000,1000,1000,1000,1000,1000,1000,1000},
  {1000,1000,1000,1000,2000,2000,2000,2000},
  {2000,2000,2000,2000,1000,1000,1000,1000}
};
//Aquí estamos declarando una matriz de 4x8 para el
movimiento de los servos.
void setup(){
  Init_PWM5(); //OUT 0&1
  Serial.begin(9600);
}

void loop(){
  for(int w=0; w<=7; w++){
    for(int y=0; y<2; y++) {
      for(int x=0; x<=1; x++) {
        OutputCh(x, PWM_SEQ[y][w]);
        delay(100);}
}}}
```

```
void OutputCh(byte ch, int pwm){
  pwm=constrain(pwm,900,2100);//si pwm se encuentra en el
rango 900-2100
  pwm*=2;

 switch(ch){
    case 0:  OCR5B=pwm; break;  //ch0
    case 1:  OCR5C=pwm; break;  //ch1
 }}

void Init_PWM5(void){
  pinMode(44,OUTPUT);
  pinMode(45,OUTPUT);
  pinMode(46,OUTPUT);
  TCCR5A =((1<<WGM51)|(1<<COM5A1)|(1<<COM5B1)|(1<<COM5C1));
  TCCR5B = (1<<WGM53)|(1<<WGM52)|(1<<CS51);
  OCR5A = 3000; //PL3,
  OCR5B = 3000; //PL4, OUT0
  OCR5C = 3000; //PL5  OUT1
  ICR5 = 40000;
}
```

En este programa podemos observar el movimiento hacia los dos lados de los servos conectados a la salida **outo** y a la salida **out1**, podemos comprobar también la intensidad de velocidad, y la forma de modificar la posición de los servos que es completamente distinta a la usada en arduino.

7.2. Programación del acelerómetro y compás

Tuvimos el problema de que se nos mojo el IMU que va conectado el Ardupilot, y no funcionaba bien el compás ni el acelerómetro.

Por lo tanto tuvimos que recurrir a un compañero para que nos prestase el GY-85.

En cuanto a la programación de este acelerómetro y compás, estuvimos perdiendo mucho tiempo en encontrar los pines que había que conectar, pero resulta que hay dos pines destinados para tal fin que son SDA y SCL.

Por lo demás es prácticamente muy parecido a la programación de Arduino Mega, por lo tanto no tendremos más complicaciones.

7.3. ACME

En este archive cabe destacar la parte del código que nos ayuda a elegir el camino más corto para llegar al destino deseado, calculando el error del rumbo.

```
int compass_error(int PID_set_Point, int PID_current_Point){
    float PID_error=0;    //variable temporal
     if(fabs(PID_set_Point-PID_current_Point) > 180) {
         if(PID_set_Point-PID_current_Point < -180){
            PID_error=(PID_set_Point+360)-PID_current_Point;
            }
         else{
           PID_error=(PID_set_Point-360)-PID_current_Point;
            }}
      else{
          PID_error=PID_set_Point-PID_current_Point;
        }
      return PID_error;
  }
```

7.4. GPS

Función que nos calcula el curso entre dos waypoints, estas formulas fueron sacadas de un foro de aeromodelismo.

```
int get_gps_course(float flat1, float flon1, float flat2, float flon2){
      float calc;
      float bear_calc;
        float x = 69.1 * (flat2 - flat1);
        float y = 69.1 * (flon2 - flon1) * cos(flat1/57.3);
        calc=atan2(y,x);
        bear_calc= degrees(calc);
        if(bear_calc<=1){
            bear_calc=360+bear_calc;
        }
    return bear_calc;
  }
```

8. Montaje

8.1. Soldadura de las patillas del ardupilot

Este apartado es muy sencillo, ya que con un soldador de estaño y estaño podemos soldar las patillas en el Ardupilot, hay que tener mucho cuidado de no soldar todas de golpe en el mismo lado, ya que podría llegar a calentarse el atmel demasiado, y no llegar a funcionar.

8.2. Conexión de los componentes

En esta fase del proyecto, vamos a poner como tienen que ir conectados cada uno de los pines que necesitamos para realizar este proyecto, cada componente que vayamos añadiendo seguirá al añadir el siguiente.

8.2.1. FTDI

Primero tenemos que preparar el FTDI para que se pueda pinjar en el Ardupilot, por lo tanto le soldamos pinchos hembra, para su fácil acople con el Ardupilot.

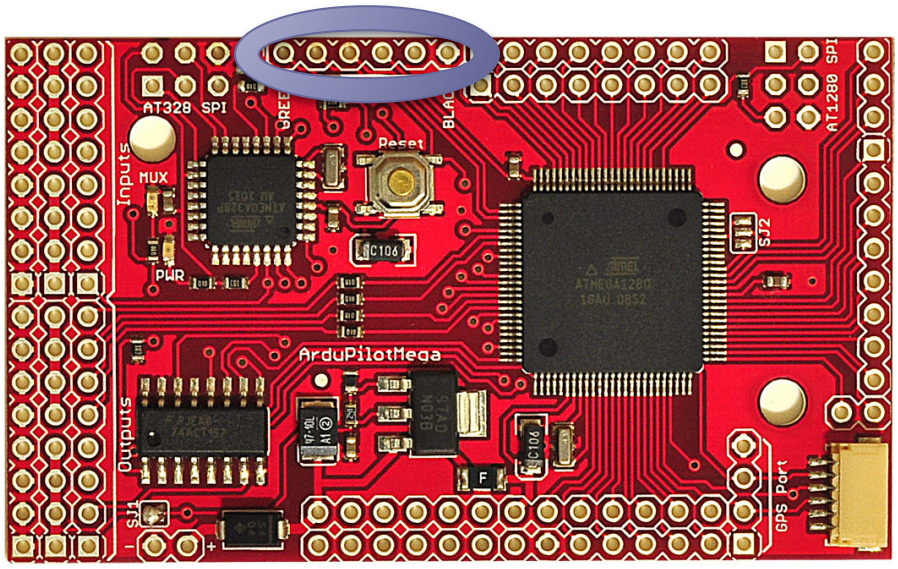
Los pines que dispone son:

DTR,RXI

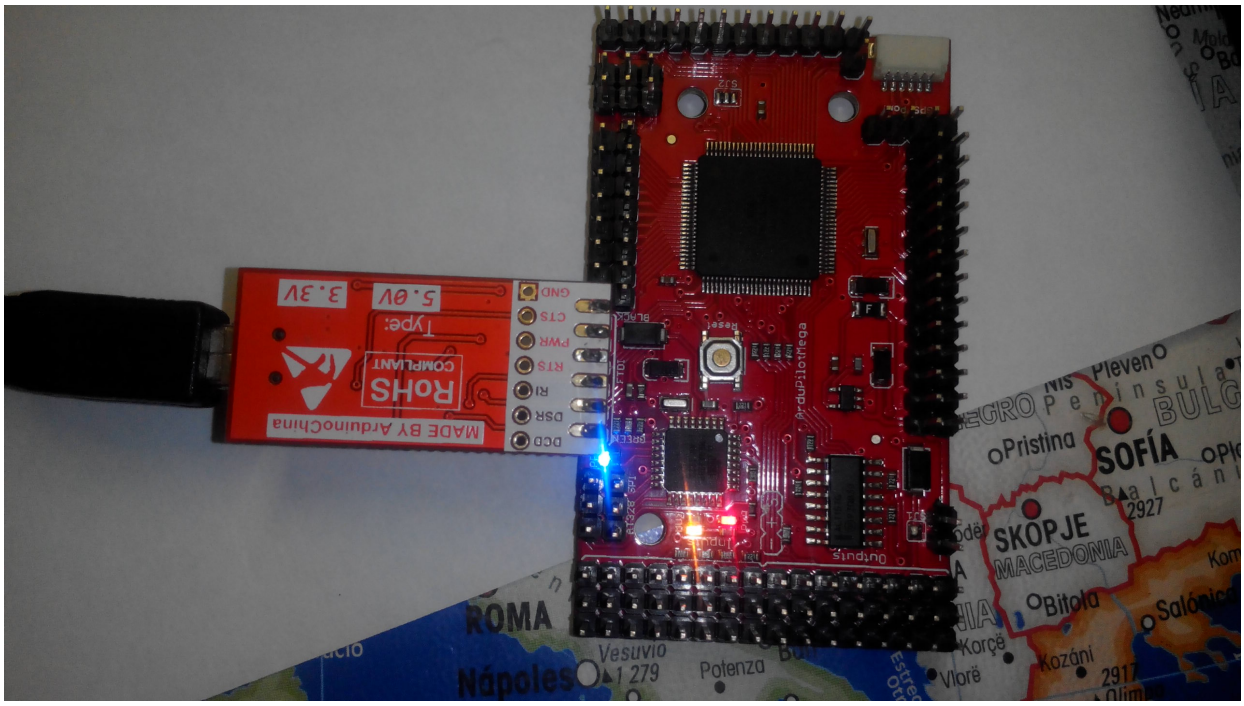
TXO,PWR

CTS,GND

Aqui es donde va
conectado el FTDI



I de esta forma es como tiene que ir acoplado el FTDI:



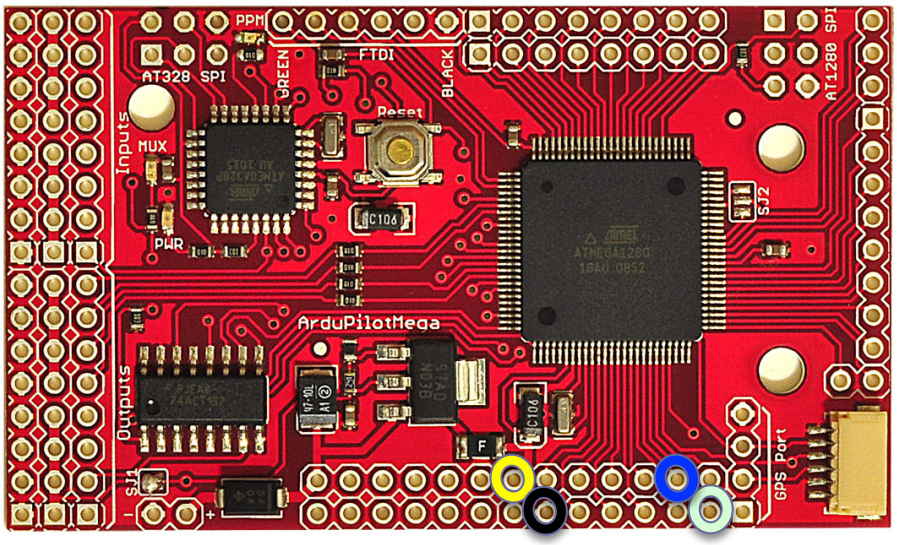
Una vez lo tengamos montado probaremos un programa sencillito para comprobar que las soldaduras están bien y que todo funciona correcto, tanto enviar el programa al Ardupilot, como desde el Ardupilot mandarle mensajes al monitor serie.

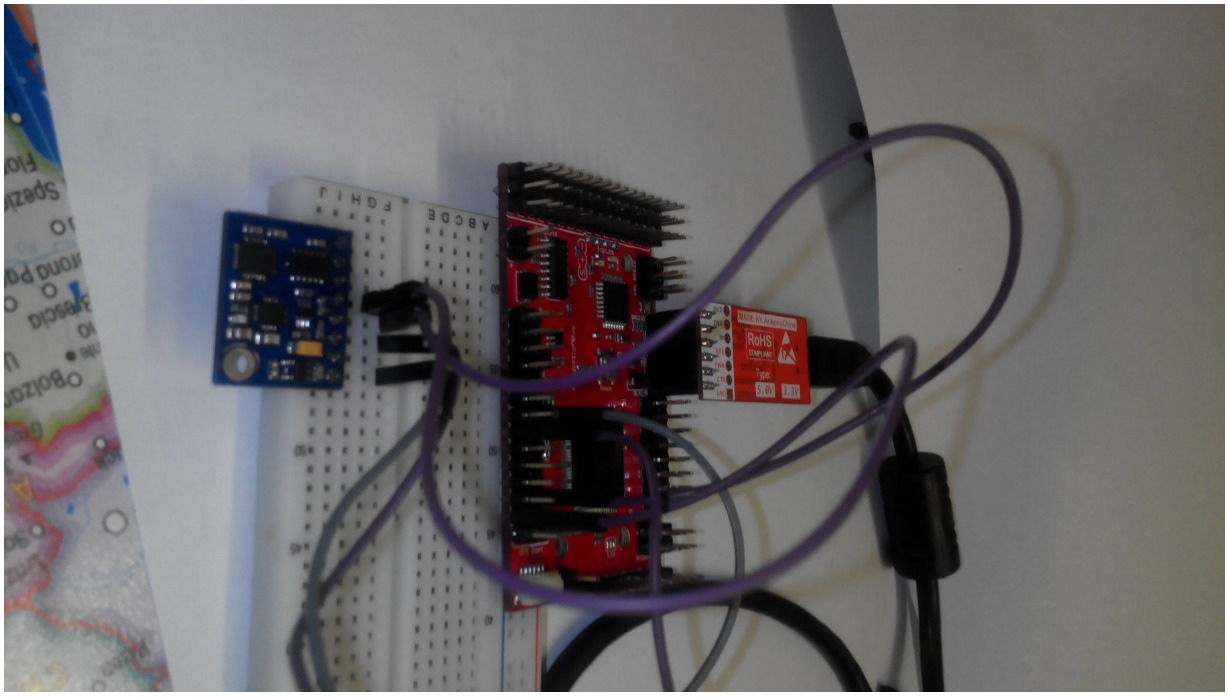
8.2.2. GY-85

Este hardware se lo añadimos porque se nos rompió el IMU de Ardupilot, con este módulo solo usaremos 4 pines que son Vin, GND, SDA, SCL.

Primero tendremos que soldar las patillas del módulo, después conectarlo al Ardupilot en sus pines correspondientes.

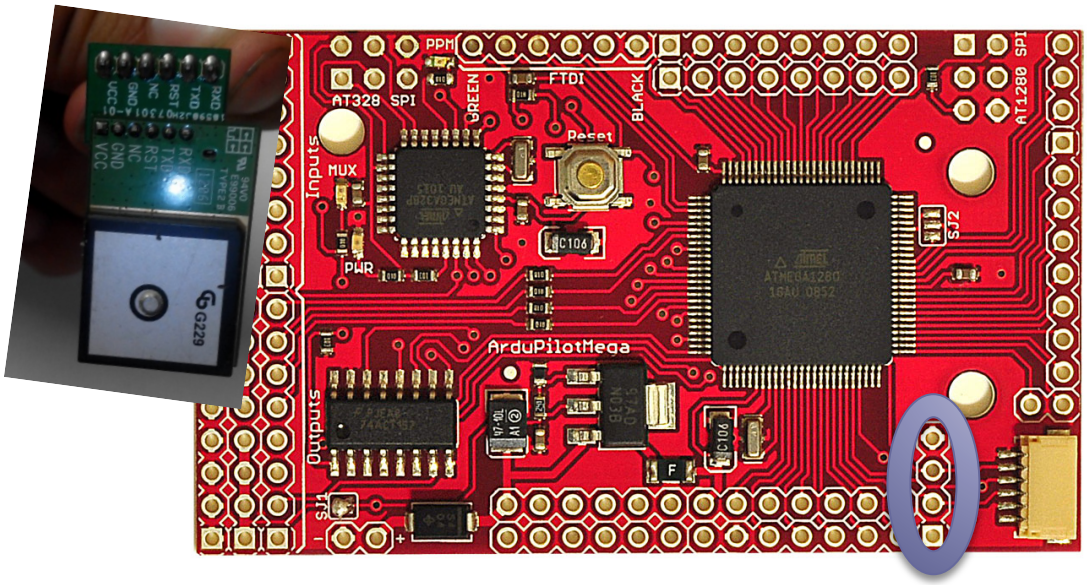
VIN GND SDA SCL



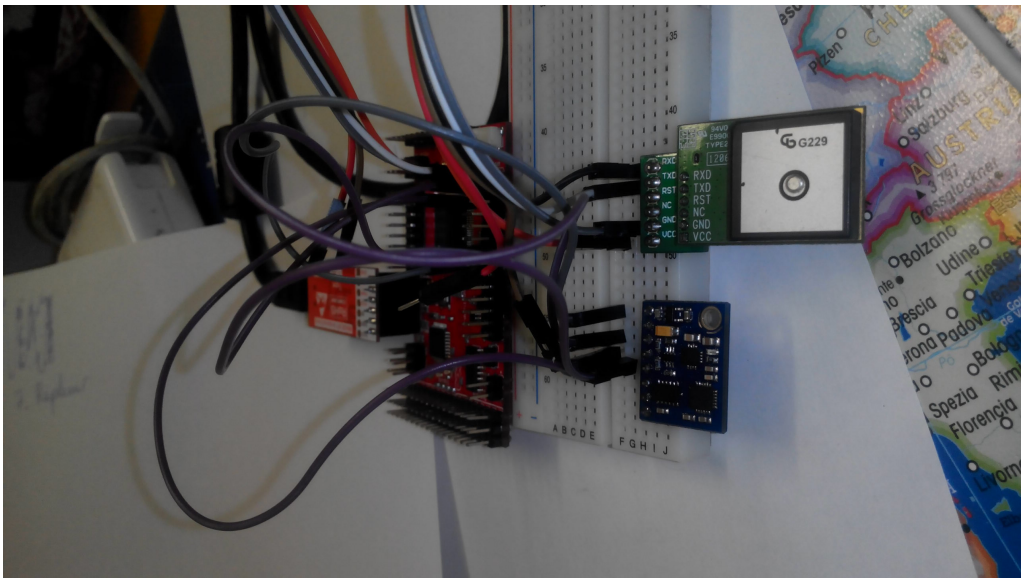


8.2.3. GPS

El módulo GPS es muy fácil de conectar a nuestro Ardupilot, ya que de este módulo solo necesitamos 4 pines que son Vcc , GND , TX y RX que son los de escritura y lectura, para entablar conversación con el Gps.



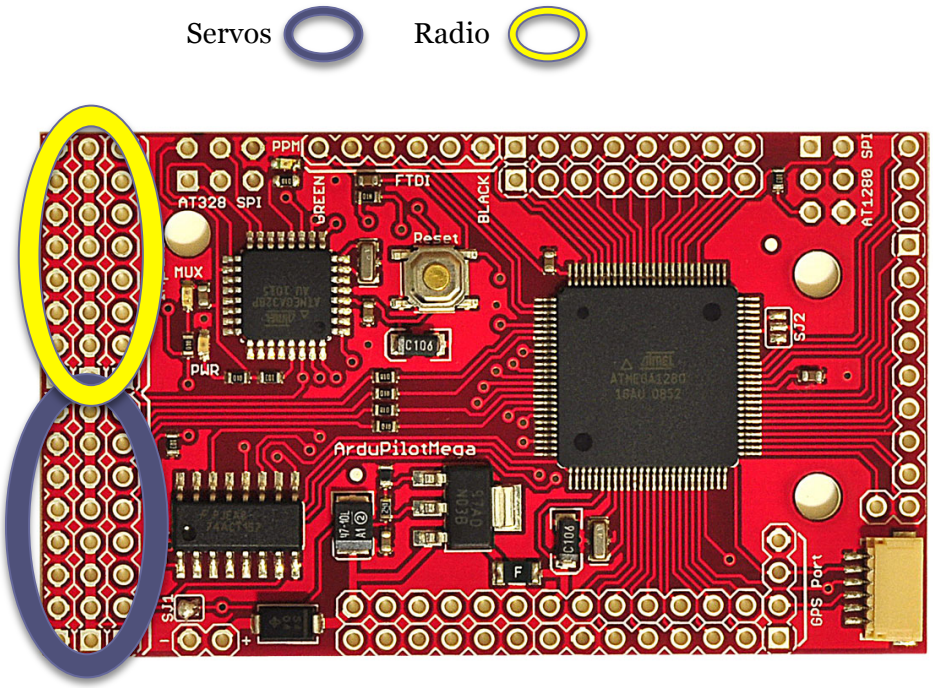
Aquí es donde va conectado el GPS



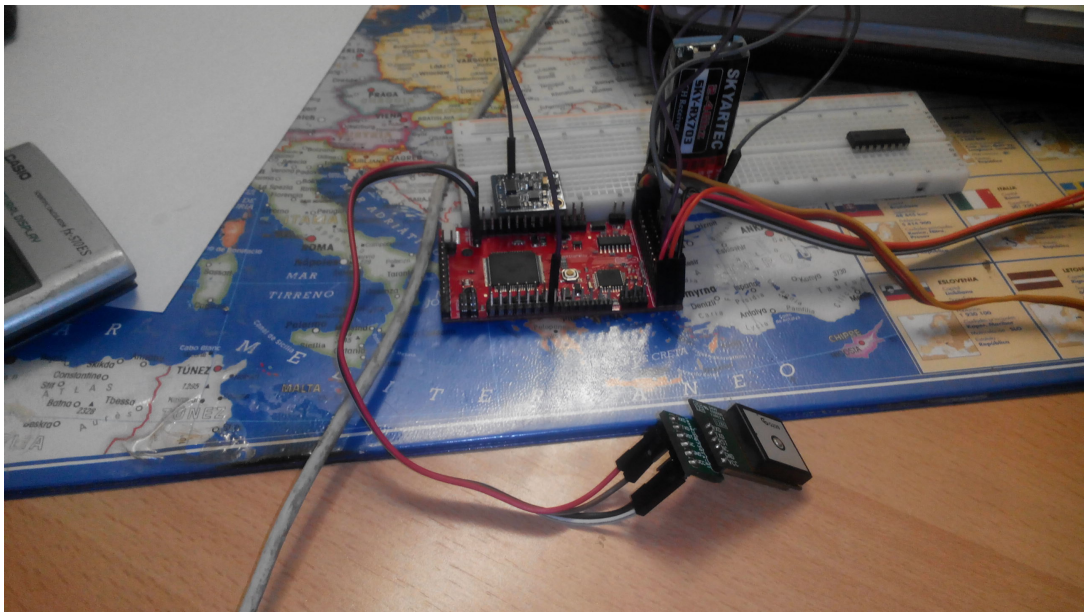
Como podemos comprobar necesitamos demasiado espacio para colocar todo de esta forma más adelante mostraremos una pequeña solución.

8.2.4. Servos y radio

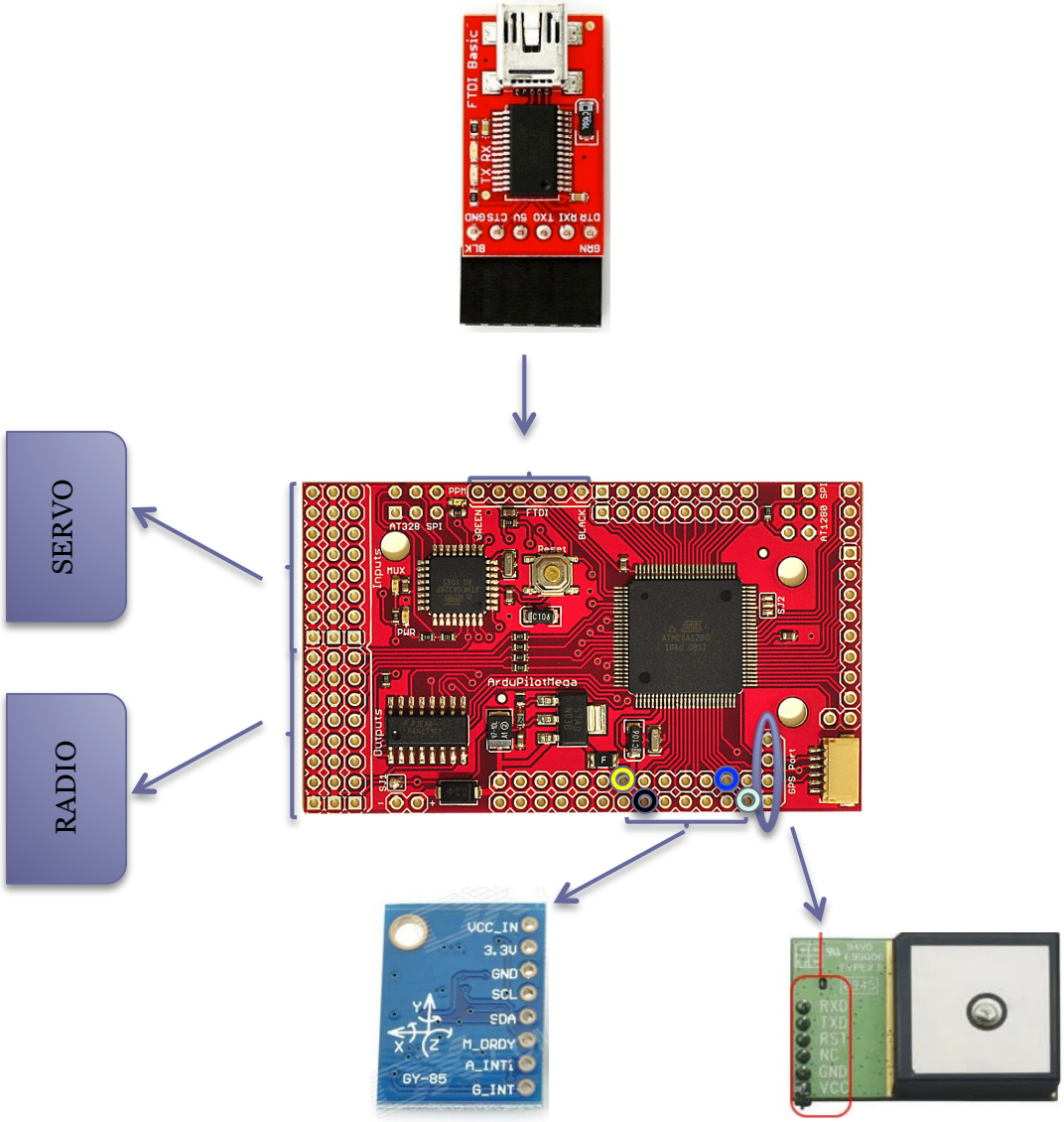
Los servos disponen de tres entradas, dos son Vin y GND y el otro controla la señal, es decir le envía pulsos para realizar los movimientos, estos van conectados tal que asi:



Nosotros usaremos 3 servos por lo tanto usaremos las 3 primeras salidas, y las entradas de los pines de la radio conectaremos el receptor de esta para poder entablar conversación con el Ardupilot, aquí podemos ver el Ardupilot con todo conectado.



8.2.5. Diagrama de conexiones

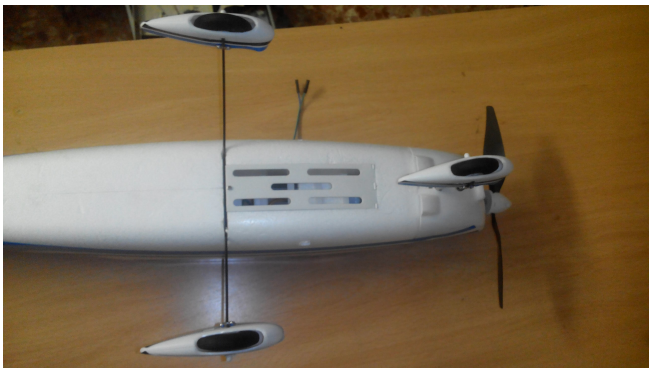


8.3. Montaje del avión

Primero cogemos la cabina y el motor brushless y los juntamos con los tornillos, una vez cogido fuerte, podemos cerrar la tapa para que cubre el motor, y poner la hélice.

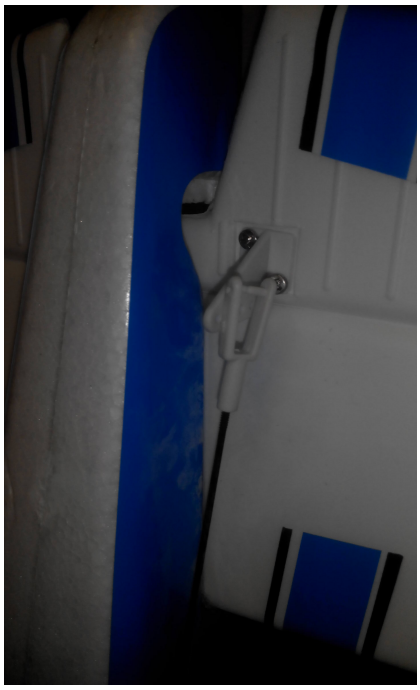
Ahora podemos montar el Puente de aterrizaje que es fijo, este puente es bastante sencillo.

Aquí irán sujetas las ruedas traseras



Aquí podemos observar como están las ruedas enganchadas al corcho de la maqueta, y en la siguiente foto como se apoya sobre ellas, le da un toque de suspensión.

Por último pegamos la cola y la aleta a la maqueta, también unimos las pestañas que se pueden ver en la segunda foto que son para unir el movimiento de estas con los servos.

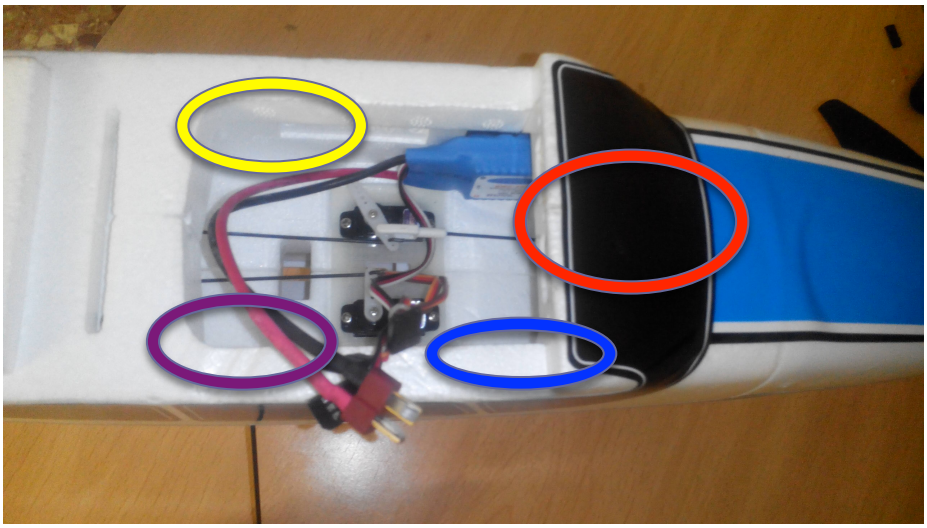


Y después de poner el hardware, ya atornillaremos las alas.

8.4. Acoplar el hardware en la maqueta

Primero designaremos donde queremos poner cada componente, y después acoplaremos el Ardupilot.

En la parte superior de la cabina colocaremos el Ardupilot y el receiver el compás/acelerómetro y el GPS y los pegaremos con velcro.



9. Processing

9.1. Dado con acelerómetro y compás

En este código que vamos a mostrar, podemos decir que establece conversación processing y Ardupilot, es decir processing pregunta y Ardupilot contesta, es distinto a lo utilizado en el GPS ya que solo escribía Ardupilot y processing lo mostraba.

```
import processing.serial.*;
Serial myPort;
float val;
void setup(){
  size(600, 400, P3D);
  myPort = new Serial(this, "COM17", 9600);
  smooth();
  frameRate(60);
  stroke(20,11,36);
}
int i=0;
float x=0;
float y=0;
void draw(){
  background(0);
  pushMatrix();   //Dibujando el objeto 3D
  translate( width/2, height/2); //Lo situamos en medio de la pantalla
  rotateX(radians(x));   //Rotamos en X
  rotateZ(radians(y));   //Rotamos en Y
  fill(48,218,37);     //Color de relleno de la figura
  box(100, 70, 180);   //Dimensiones de la figura
  popMatrix();
 print(x+"   "+y);
}
void serialEvent(Serial port) {
 String input = port.readStringUntil('*');
  if (input != null) {
   float[] vals = float(splitTokens(input));
   x =vals[0];
    y =vals[1];
    println("esta es x   :  " + x + "esta es y   :   " + y);
  }}
```

9.2. GPS

```
import processing.serial.*;
PImage map;
PImage flecha;
float lat, lon;
double mapCenterLat =0.0;
double mapCenterLon =-0.0;
float x,y=0;

public void setup() {
  size(750,450);
  new Serial(this, "/dev/tty.usbserial-AD029ZAH", 9600);
  map=loadImage("mapa.jpg");
  flecha=loadImage("Flecha.png");
  image(map,0,0);
 }
 public void draw(){
    if(x!=0||y!=0){
     image(flecha,y,x);
     }
}
void serialEvent(Serial port) {
 String input = port.readStringUntil('*');
  if (input != null) {
    float[] vals = float(splitTokens(input));
    println( "Receiving:" + input+"    "+vals[0]+"    "+vals[1]);
    lat =vals[0];
    lon =vals[1];
    mapCenterLat=lat;
    mapCenterLon=lon;
    //calculo aproximado a escala en el mapa
    x=672-((lat-39.15)*100000);
    y=670+((lon+0.43)*100000);
    println("esta es x  :  " + x);
  }
}
```

Declaramos las variables necesarias, PImage declara una imagen

Declaramos el serial para que pueda leer la información del GPS

Leemos lo que nos mandan por el puerto serie, y lo gestionamos

10. Pruebas

10.1. Compás y Acelerómetro

Las pruebas que vamos a realizar con el compás, vamos a crear un programita sencillo con processing para poder simular una brújula, y poder comprobar que realmente funciona bien el compás.

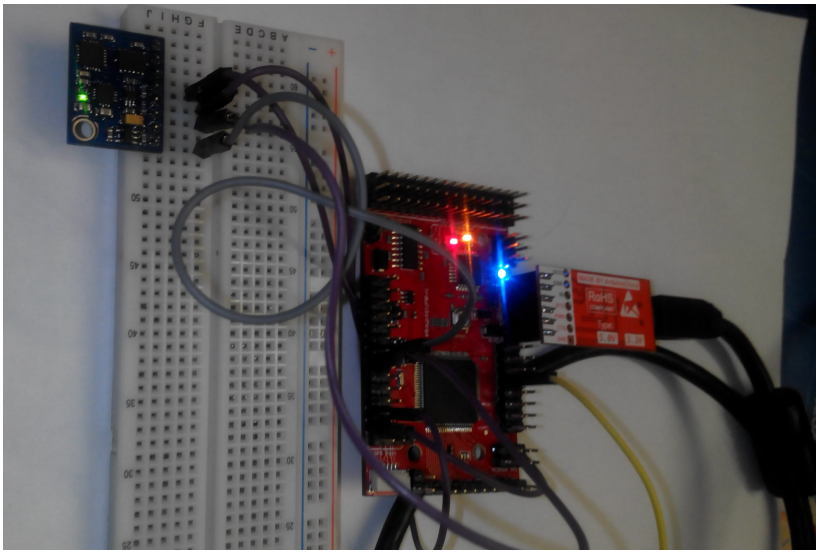
Primero conectamos el Ardupilot al PC mediante el programador FTDI, y conectamos el GY-85 al Ardupilot:

Vin → 5V

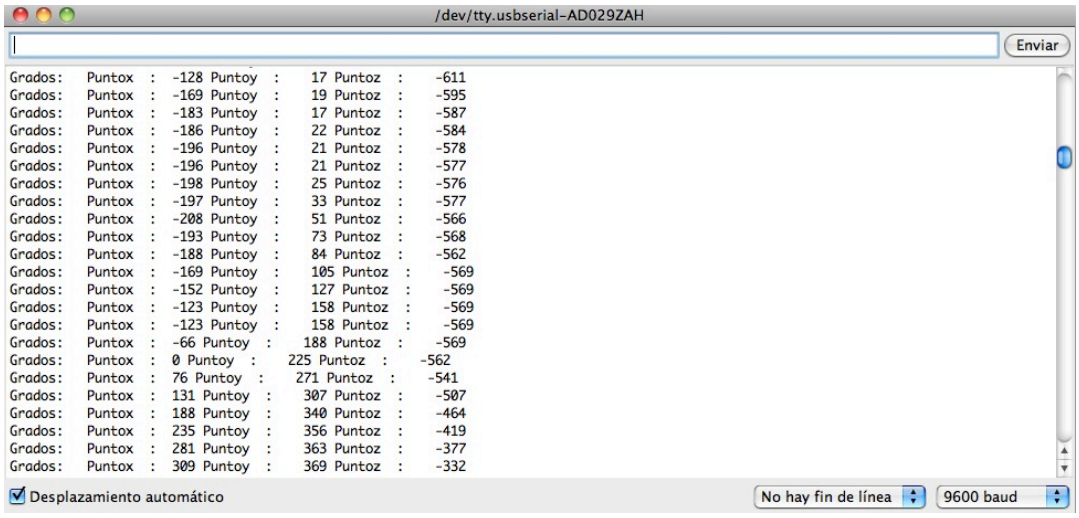
GND → GND

SDA → SDA

SCL → SCL



Podemos ver lo que saca por el monitor serie:



10.2. Simulación

10.2.1. Simulación compás y acelerómetro

Mediante el compás y el código de processing mostrado en el punto 9.1 simulamos una brújula real, el compás le envía mediante el puerto serie los grados respecto al norte, y hace girar el fotograma que simula la brújula, funciona perfectamente y casi sin error.

Hemos realizado dos pruebas y las dos satisfactorias, eso si tener cuidado al conectar SDA y SCL ya que si no se conectan en los pines correctos no funcionará bien.

Video de la simulación del acelerómetro/compás:

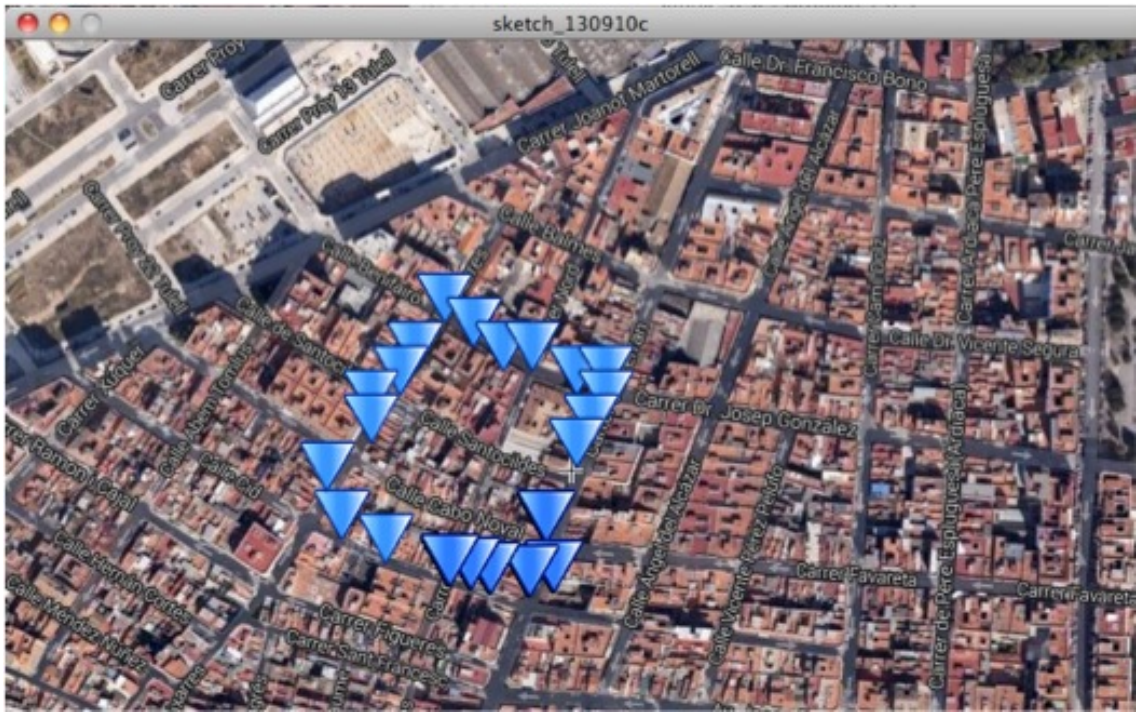
http://www.youtube.com/watch?v=dSpaem8Lg88&feature=youtu.be

10.2.2. Simulación GPS

Mediante el compás y el código de processing mostrado en el punto 9.2 simulamos los datos que nos devuelve el GPS en un pequeño mapa sacado de google maps.

La prueba a sido realizada dando la vuelta a la manzana de mi casa, como se puede comprobar no es exacto, pero es bastante preciso.

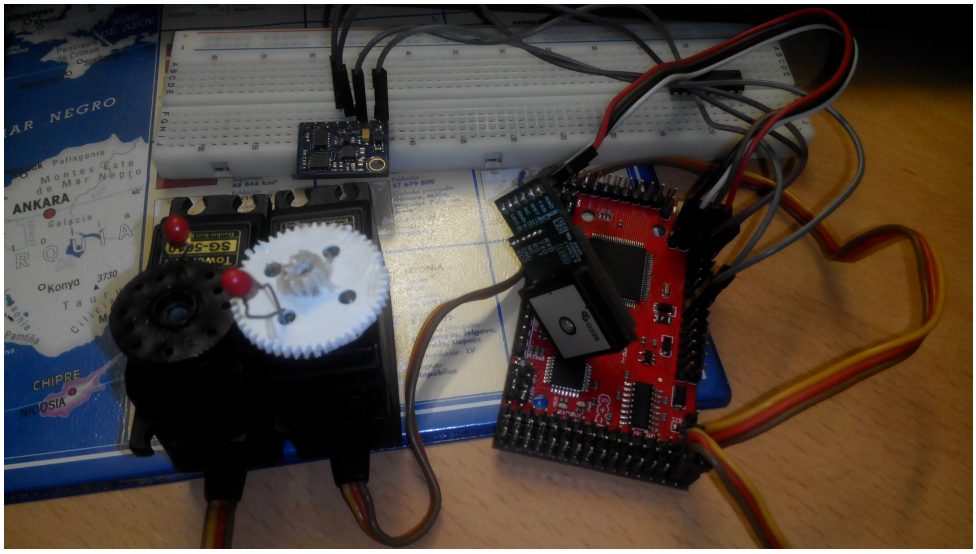
Lo que sorprende es la rápida conexión del GPS con los satélites.



10.3. Servos y motores

Después de lo costoso que fue configurar los servomoteres, porque en casi ningún sitio ponía como moverlos desde los pines out, hasta que encontramos un foro que nos ayudo bastante, ya que nos explicaron los puertos que había que usar y como gestionarlos, sin este foro no lo habríamos conseguido. Ver bibliografía.

El código de este programa esta en el punto 7, ya que nos costo de realizar y de entender hemos creído oportuno ponerlo.



También realizamos un video, ya que en una foto no podemos apreciar su movimiento.

Video en youtube:

http://www.youtube.com/watch?v=vIDpvmdAYNs&feature=youtu.be

10.4. Vuelo

Las pruebas de vuelo han sido las más complicadas de realizar, primero por los problemas del hardware, ya que casi nunca a funcionado a la primera, y segundo por el parte meteorológico, ya que probamos a volar una vez con un poco de viento y nos volvimos con la cabina del motor suelta y sin hélice, pero pudimos repararlo todo.

Otra prueba que hicimos era mandar el avión muy lejos, para probar el cambio a modo automático, al principio no funcionaba porque no habíamos conectado bien el receiver, gracias a la ayuda de un compañero pudimos solucionarlo.

Y la ultima prueba la definitiva que fue cuando hizo más o menos el recorrido que le habíamos marcado y regreso, eso si el aterrizaje y el despegue lo tuvimos que hacer manual, ya que no nos fiábamos de que no detectase bien el suelo, ya que donde hemos hecho las pruebas hay distintos tipos de superficie como césped, tierra y asfalto.

Aquí mostramos el recorrido en Google Maps que mandamos hacer al avión.

Este día el viento iba a 0m/s pero aún así había veces que sufríamos, porque el avión se desplazaba un poco del rumbo pero después continuaba bien.

11. Conclusiones y trabajos futuros

La conclusión que sacamos de este proyecto final de carrera, que cuando tenemos que programar hardware, existen muchos más problemas, ya que dependemos no solo de lo que se refiere al software, sino que también hay que tener en cuenta el cableado y el microprocesador. Por lo tanto hay que tener más pendiente y trabajar en plazos corto, ya que si tienes mucho código pero no lo has probado con el hardware y no funciona es mucho más difícil encontrar el error.

Podemos decir también que nos hemos quedado con ganas de realizar muchas más pruebas pero eso de depender del aire libre es lo pero, no se descarta seguir avanzando en el proyecto una vez entregado.

También puedo afirmar que la parte del proyecto que más me ha gustado ha sido poder comprobar todo aquello programado en real, aunque no siempre el hardware realice aquello esperado en el programa.

Con este proyecto hemos alcanzado diversos campos, y aún así nos hemos dejado muchos de lado, como es realizar una interfaz conjunta para gestionar la maqueta durante el vuelo, y hacer pruebas en parado para comprobar el GPS y el acelerómetro.

Como proyectos futuros tenemos lo que son añadirle una cámara, y añadirle sensores.

También como proyecto sería realizar una aplicación que recibiera todos los datos del Ardupiltot, y realizar una placa para poder juntar todos los componentes sin utilizar cables y evitar errores.

Hacer un vuelo memoria, que sería realizar un vuelo con el mando, que el Ardupilot almacenara todo el recorrido, y que después el avión realizase el circuito sin control remoto.

12. Bibliografía

Librerías Ardupilot:

http://code.google.com/p/ardupilotmega/source/browse/libraries/?name=ArduCopter-2.4.1 - libraries%2FAC_PID

Librería GY-85:

http://arduikyo.blogspot.nl/2013/08/hmc5883l-compass-arduino-processing.html

Simulador:

http://code.google.com/p/ardupilot-mega/wiki/BuildingWithMake

Foros aeromodelismo:

http://www.aeromodelismovirtual.com/showthread.php?t=2361&highlight=ardupilot

http://www.foro-aeromodelismo.com/viewtopic.php?f=41&t=5141

http://www.aeromodelismovirtual.com/showthread.php?t=562

http://tecnopinos.blogspot.com.es/

Páginas información general:

https://sites.google.com/site/cuadricoptero/home/arducopter/software/programando-ardupilot-mega/modo-cli

http://www.aerorobotica.com/

Estabilidad:

http://www.manualvuelo.com/PBV/PBV16.html

Control PID:

http://flyerballr4.wordpress.com/2010/04/14/siguientes-pasos/

Avión:

http://www.ebay.es/itm/4Ch-Cessna-182-RC-Plane-Brushless-version-RTF-2-4GHz-/261279812309?pt=UK_ToysGames_RadioControlled_JN&hash=item3cd57d8ad5&_uhb=1

Pines Ardupilot:

https://docs.google.com/spreadsheet/ccc?key=0AlDHmpARnagbdDJVanpBWTlhQnZuWEJ1dzJBRmdBblE&authkey=CKzJv_IP&hl=en&authkey=CKzJv_IP - gid=0