**Capítulo 1**

**Costes**

* 1. **Costes Hardware**

Este proyecto incluye la construcción de un cuadricóptero desde cero y la generación de una API para permitir el control del drone a través de un ordenador.

* + 1. Costes para el cuadricóptero

Tabla 1. Costes cuadricóptero.

|  |  |  |  |
| --- | --- | --- | --- |
| Componente | Cantidad | Precio | Total |
| Frame “*Diatone FPV 250L”* | 1 | 15,63 € | 15,63 € |
| *Controladora “Afro Flight Naze32 Acro”* | 1 | 27,59 € | 27,59 € |
| Motores *“DYS BE1806-2300kv Brushless Multirotor Motor 2S~3S”* | 4 | 8,86 € | 35,44 € |
| Hélices “*Gemfan 5x3”* 2pcs | 2 | 1,21 € | 2,42 € |
| Hélices *“HobbyKing 5x3”* 6pcs | 1 | 2,33 € | 2,33 € |
| ESC “*Afro ESC BEC UltraLite 12Amp”* | 4 | 9,19 € | 36,76 € |
| Batería “*ZIPPY Compact 2450mAh 3S 35C LiPo”* | 1 | 19,95 € | 19,95 € |
| Conectores “*XT-60”* | 1 | 3,67 € | 3,67 € |
| *Transmisor control “433 MHz Kit Telemetry 100 mW v1.0”* | 1 | 26,56 € | 26,56 € |
| Cámara “*Mobius C2”* | 1 | 62,99 € | 62,99 € |
| Cable *“Mobius ActionCam A/V Out Cable Set”* | 1 | 4,49 € | 4,49€ |
| Transmisor *“BOSCAM Micro 5,8 GHz TS351+RC305”* | 1 | 34,12 € | 34,12 € |
| Buzzer *“HobbyKing Lipoly Low Voltage Alarm”* | 1 | 1,85 € | 1,85 € |
| TOTAL | **20** | **-** | **273,8 €** |

* + 1. Otros costes hardware

Tabla 2. Otros costes hardware.

|  |  |  |  |
| --- | --- | --- | --- |
| Componente | Cantidad | Precio | Total |
| *Emisora “Futaba T6J”* | 1 | 164,68 € | 164,68 € |
| *Emisora-USB “E-Sky RadioTransmitter”* | 1 | 20,00 € | 20,00 € |
| Portátil *“Toshiba Satellite Pro L-300”* | 1 | 100,00 € | 100,00 € |
| Portátil “*MSI GE72 6QD”* | 1 | 1.179,00 € | 1.179,00 € |
| TOTAL | **4** | **-** | **1.463,68 €** |

El coste total en hardware para este proyecto ha sido de **1.737,48** €. Sin embargo el coste de la construcción del drone es de **273,8** €, y los demás costes pueden suplirse con otros componentes que sirvan para la causa más baratos.

* 1. **Costes de personal basados en COCOMO**

Entre los distintos métodos de estimación de costes de desarrollo de software, el modelo COCOMO (COnstructive COst MOdel) desarrollado por Barry M. Boehm, se engloba en el grupo de los modelos algorítmicos que tratan de establecer una relación matemática la cual permite estimar el esfuerzo y tiempo requerido para desarrollar un producto.

Por un lado COCOMO define tres modos de desarrollo o tipos de proyectos:

* **Orgánico:** proyectos relativamente sencillos, menores de 50 KDLC líneas de código, en los cuales se tiene experiencia de proyectos similares y se encuentran en entornos estables.
* **Semi-acoplado:** proyectos intermedios en complejidad y tamaño (menores de 300 KDLC), donde la experiencia en este tipo de proyectos es variable, y las restricciones intermedias.
* **Empotrado:** proyectos bastante complejos, en los que apenas se tiene experiencia y se engloban en un entorno de gran innovación técnica. Además se trabaja con unos requisitos muy restrictivos y de gran volatilidad.

Y por otro lado existen diferentes modelos que define COCOMO:

* **Modelo básico:** Se basa exclusivamente en el tamaño expresado en LDC.
* **Modelo intermedio:** Además del tamaño del programa incluye un conjunto de medidas subjetivas llamadas conductores de costes.
* **Modelo avanzado:** Incluye todo lo del modelo intermedio además del impacto de cada conductor de coste en las distintas fases de desarrollo.

Para nuestro caso el modelo intermedio será el que usaremos, dado que realiza las estimaciones con bastante precisión y es suficiente para nuestro proyecto.

Las estimaciones de COCOMO se basan en lo siguiente:

* **E = Esfuerzo = a KLDC e \* FAE** (persona x mes)
* **T = Tiempo de duración del desarrollo = c Esfuerzo d** (meses)
* **P= Personal = E/T** (personas)

Por tanto lo primero que tenemos que calcular son los **KLDC**, o KiloLíneas De Código. Para ello se debe calcular la siguiente fórmula:

**KLDC** = (**PF** \* **LDCporPF**) / 1000

Siendo los **PF** los puntos de función. Los puntos de función representan una métrica de la funcionalidad que brinda el producto software.

Componentes:

* **EI:** Procesos en los que se introducen datos y actualizan archivos internos.
* **EO:** Procesos en los que se envían datos al exterior de la aplicación.
* **EQ:** Procesos consistentes en la que la entrada no modifican archivos ni derivados.
* **ILF:** Grupos de datos relacionados entre sí.
* **EIF:** Grupos de datos que se mantienen externamente.

Estos valores los podemos ver en la siguiente tabla. Con esto se pueden calcular los puntos de función sin ajustar según pesos en nuestro proyecto, que se puede ver la siguiente tabla:

Tabla 3. Puntos de función sin ajustar.

|  |  |  |  |
| --- | --- | --- | --- |
| Componente | Valor | Peso | Total |
| EI | *14* | *3* | 42 |
| EO | *91* | *4* | 364 |
| EQ | *62* | *3* | 186 |
| ILF | *4* | *7* | 28 |
| EIF | *3* | *5* | 15 |
| **TOTAL** |  |  | **635** |

Para ajustar dicho cálculo se debe averiguar primero el factor de ajuste, valorando del 0 al 5 las características generales del sistema. Según el nivel de influencia:

Tabla 4. Niveles de influencia.

|  |  |
| --- | --- |
| **Grado** | **Descripción** |
| **0** | **No está presente o no influye** |
| **1** | **Influencia Mínima** |
| **2** | **Influencia Moderada** |
| **3** | **Influencia promedio** |
| **4** | **Influencia significativa** |
| **5** | **Influencia fuerte** |

Tabla 5. Características de un sistema.

|  |  |
| --- | --- |
| Características generales de un sistema | Valor |
| Comunicación de datos | 5 |
| Procesamiento distribuido | 0 |
| Performance | 5 |
| Equipamiento | 3 |
| Volumen de transacciones | 0 |
| Entrada de datos Online | 0 |
| Interfaz de usuario | 3 |
| Actualización Online | 4 |
| Procesamiento complejo | 2 |
| Reusabilidad | 5 |
| Facilidad de implementación | 0 |
| Facilidad de operación | 3 |
| Múltiples locales | 0 |
| Facilidad de cambios | 2 |
| **Total** | **31** |

**Factor de ajuste:** (31 \* 0.01) + 0.65 = **0.96**.

Por tanto el valor de **PF** ajustado sería **635 \* 0.96** = **609.6**.

Ahora podemos calcular los **KLDC** pero para ello necesitamos también las líneas de código por punto de función que vienen determinado según el lenguaje de programación:

|  |  |
| --- | --- |
|  | LDC/PF |
| **Ensamblador** | 320 |
| **C** | 150 |
| **COBOL** | 105 |
| **Pascal** | 91 |
| **C++** | 64 |
| **C#** | 54 |

Tabla 6. LDC/PF según lenguaje.

Como tenemos en el lenguaje C#, que es el utilizado, usaremos su valor. Finalmente el valor de **KLDC** = **609.6 \* 54** / **1000** = **32,91**.

Sin embargo esta estimación no es del todo válida para usarla como indicadora del número de líneas de código, pero con los pesos asignados en los puntos de función, podemos ver una estimación de complejidad.

Aquí vemos las líneas de código reales de nuestro proyecto, que son muchas menos, según archivo.

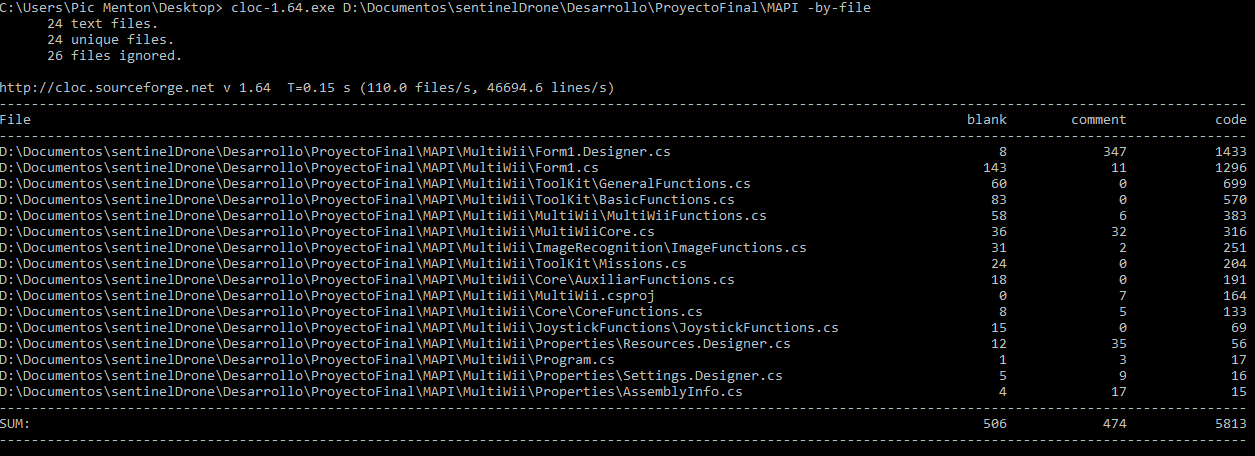


Figura 1. Líneas de código obtenidas mediante el programa “Cloc”.

En total son **5813** líneas de código, 586 líneas en blanco y 474 líneas de comentarios.

Aun así como nuestro **KLDC** es menor que 50, nuestro proyecto se considera orgánico. Aun así cogeremos el valor de 5813, sin contar los pesos. Es decir **KLDC** = **5,8**.

Por tanto y siguiendo la tabla cogeremos los valores adecuados para los coeficientes de COCOMO:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Proyecto Software | a | e | c | d |
| Orgánico | 3,2 | 1,05 | 2,5 | 0,38 |
| Semi-acoplado | 3,0 | 1,12 | 2,5 | 0,35 |
| Empotrado | 2,8 | 1,20 | 2,5 | 0,32 |

Tabla 7. Coeficientes de COCOMO.

Por otro lado también hemos de hallar la variable **FAE**, Factor de Ajuste de Esfuerzo, la cual se obtiene mediante la multiplicación de los valores evaluados en los diferentes 15 conductores de coste que se observan en la siguiente tabla:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Conductores de coste | VALORACIÓN | | | | | |
| ***Muy bajo*** | ***Bajo*** | ***Nominal*** | ***Alto*** | ***Muy***  ***alto*** | ***Extr. alto*** |
| Fiabilidad requerida del software | 0,75 | 0,88 | **1.00** | 1,15 | 1,40 | - |
| Tamaño de la base de datos | **-** | 0,94 | 1.00 | 1,08 | 1,16 | - |
| Complejidad del producto | 0,70 | 0,85 | 1.00 | **1,15** | 1,30 | 1,65 |
| Restricciones del tiempo de ejecución | - | - | **1.00** | 1,11 | 1,30 | 1,66 |
| Restricciones del almacenamiento principal | **-** | - | 1.00 | 1,06 | 1,21 | 1,56 |
| Volatilidad de la máquina virtual | **-** | 0,87 | 1.00 | 1,15 | 1,30 | - |
| Tiempo de respuesta del ordenador | - | 0,87 | 1.00 | 1,07 | **1,15** | - |
| Capacidad del analista | 1,46 | 1,19 | **1.00** | 0,86 | 0,71 | - |
| Experiencia en la aplicación | 1,29 | 1,13 | 1.00 | **0,91** | 0,82 | - |
| Capacidad de los programadores | 1,42 | 1,17 | 1.00 | 0,86 | **0,70** | - |
| Experiencia en S.O. utilizado | 1,21 | 1,10 | 1.00 | **0,90** | - | - |
| Experiencia en el lenguaje de programación | **1,14** | 1,07 | 1.00 | 0,95 | - | - |
| Prácticas de programación modernas | 1,24 | 1,10 | 1.00 | **0,91** | 0,82 | - |
| Utilización de herramientas software | 1,24 | 1,10 | **1.00** | 0,91 | 0,83 | - |
| Limitaciones de planificación del proyecto | 1,23 | 1,08 | 1.00 | **1,04** | 1,10 | - |

Tabla 8. Conductores de coste para el FAE.

**FAE:** 1 \* 1,15 \* 1 \* 1,15 \* 1 \* 0.91 \* 0.7 \* 0,9 \* 1,14 \* 0,91 \* 1 \* 1,04 = **0.818**.

**Justificación de los valores:**

*Atributos de software*

* **Fiabilidad requerida del software:** Si se producen fallos en las funciones de la API cualquier drone puede acabar rompiéndose. (Nominal)
* **Tamaño de la base de datos:** No tenemos base de datos. (Nada)
* **Complejidad del producto:** La aplicación debe poder trabajar en paralelo, realizar procesamiento de imágenes en tiempo de ejecución y enviar y recibir datos por puerto serie. (Alto)

*Atributos de hardware*

* **Restricciones del tiempo de ejecución:** En los requerimientos se exige buen rendimiento ya que sino el hardware estaría actuando bien. (Nominal)
* **Restricciones del almacenamiento principal:** No hay restricciones al respecto. (Nada)
* **Volatilidad de la máquina virtual:** No se usarán. (Nada)
* **Tiempo de respuesta del ordenador:** Deberá ser interactivo con el usuario y actuar de forma rápida. (Muy Alto)

*Atributos del personal*

* **Capacidad del analista:** No hemos realizado proyectos similares, pero hemos sido capaces de analizar el problema de forma bastante buena. (Nominal)
* **Experiencia en la aplicación:** No hemos tenido ninguna experiencia con drones anterior. (Bajo)
* **Capacidad de los programadores:** No hemos estado en proyectos similares. (Bajo)
* **Experiencia en S.O. utilizado:** Con Windows 8, experiencia alta. (Alto)
* **Experiencia en el lenguaje de programación:** Ninguna. (Muy Bajo)

*Atributos del proyecto*

* **Prácticas de programación modernas:** Se usarán prácticas de programación mayormente convencional (Alto).
* **Utilización de herramientas software:** Se usarán herramientas estándar que no exigirán apenas formación, de las cuales se tiene cierta experiencia (Nominal).
* **Limitaciones de planificación del proyecto:** Existen limitaciones en el proyecto debido al tiempo y al objetivo. (Alto)

Por tanto los cálculos serán los siguientes:

***Cálculo del esfuerzo del desarrollo:***

**E** = a KLDC e \* FAE = 3,2 \* (**5.8**)^1,05 \* **0.818**= **16,57 personas/mes**

***Cálculo tiempo de desarrollo:***

**T** = c Esfuerzo d = 2,5 \* (**16,57**)^0,38 = **7,3 meses**

***Productividad:***

**PR** = LDC/Esfuerzo = 5813/16,57 = **350,815 LDC/personas mes**

***Personal promedio:***

**P** = E/T = 16.57/7,3 = **2,27 personas**

Según estas cifras será necesario un equipo de 3 personas trabajando alrededor de 7 meses, pero puesto que somos trabajando dos personas, se realizará en el periodo de 9 meses. No hay que olvidar que a este proyecto software también se suma el trabajo realizado en la construcción del drone.

* 1. **Costes de producción**

Los costes de producción no tienen mucho sentido para un proyecto que trata de crear una API abierta, sin embargo, teniendo en cuenta el coste de personal y el coste hardware, podemos estimar un coste de producción de la construcción de un drone, y una aplicación realizada con esta API.

Para ello ponemos el precio base de un cuadricóptero como el nuestro, reducimos el 30% de coste debido a que, en caso de ser producido a gran escala, cada componente saldría más barato, esto es:

Precio del drone en producción: 273,8€ \* 0.7 = **191,66** €.

A este precio le sumamos el de la construcción estimada del drone, que, al haber realizado un manual de montaje, no costará demasiado. Automatizando dichos pasos el coste por drone de producción podría ser de unos 20 €, para pagar la luz de las máquinas que deben montarlo.

Y suponemos que el coste de realizar una aplicación que valdrá para muchos drones puede ser de unos 500 €, con lo que se le podría suma un poco más a la producción.

El total estimado de producción por cuadricóptero con una aplicación realizada por nuestra API sería de **250** €.

* 1. **Precio final del producto**

Debido a que el coste de producción estimado es de 250 €, el precio final del producto, teniendo en cuenta desplazamientos, mantenimiento y actualizaciones, podría ser de unos **350** €.

Aun así recordemos que el proyecto en sí no tiene precio como producto, pues lo más importante es la generación de la API y que cualquier drone que utilice MultiWii puede utilizarse con dicha API.