



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería
Informática**

**título del TFG
Documentación Técnica**



Presentado por Álvaro Ruifernández Palacios
en Universidad de Burgos — 14 de enero
de 2019

Tutor: Jesús Enrique Sierra García

Índice general

Índice general	I
Índice de figuras	III
Índice de tablas	IV
Apéndice A Plan de Proyecto Software	1
A.1. Introducción	1
A.2. Planificación temporal	2
A.3. Estudio de viabilidad	11
Apéndice B Especificación de Requisitos	17
B.1. Introducción	17
B.2. Objetivos generales	17
B.3. Catalogo de requisitos	17
B.4. Especificación de requisitos	18
Apéndice C Especificación de diseño	25
C.1. Introducción	25
C.2. Diseño de datos	25
C.3. Diseño procedimental	26
C.4. Diseño arquitectónico	27
Apéndice D Documentación técnica de programación	29
D.1. Introducción	29
D.2. Estructura de directorios	29
D.3. Manual del programador	31

D.4. Compilación, instalación y ejecución del proyecto	32
D.5. Pruebas del sistema	32
Apéndice E Documentación de usuario	33
E.1. Introducción	33
E.2. Requisitos de usuarios	33
E.3. Instalación	33
E.4. Manual del usuario	33

Índice de figuras

C.1. Diagrama de secuencia del sistema	26
C.2. Diagrama de clases del sistema	27
D.1. Interfaz gráfica de Jupyter	32
D.2. Interfaz gráfica de Spyder	32

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En este apartado se especificará la planificación seguida en cuanto a tiempo se refiere para realizar cada una de las tareas que han conformado este proyecto, permitiendo gracias a esta planificación cumplir con los objetivos que se han establecido para el sistema desarrollado.

Debido a que gran parte de este proyecto se ha basado en la investigación (ya que, como se ha explicado en otros puntos, se desconocían algunos de los procedimientos a realizar), la planificación temporal no ha podido ser tan estricta como debería ser en un proyecto cuya organización se basa en la metodología ágil conocida como SCRUM como es este caso. Otro aspecto que ha impedido seguir esa planificación ha sido la aparición de una gran cantidad de contratiempos surgidos durante cada una de las fases de este proyecto, como se describirá más adelante.

En cuanto al estudio de viabilidad, la parte económica ha sido también un aspecto difícil de calcular ya que se ha necesitado probar una serie de implementaciones, tanto hardware como software, hasta dar con la idónea para cumplir los requisitos del proyecto.

A.2. Planificación temporal

Como bien se ha descrito en la introducción, en este apartado se especificará el desarrollo del proyecto descrito a partir de los periodos de tiempo que han llevado a desarrollar cada parte, o como se conoce en terminología SCRUM, cada uno de los sprints.

Sprint 0: 05/10/2017 - 22/11/2017

Este es el sprint inicial, donde se explica y establece los conceptos sobre los que se va a trabajar y los elementos que se van a emplear para ello. Se plante la realización de este trabajo basandose en el sistema operativo Ubuntu, en su versión 14.04 LTS, para desarrollar el código usando la aplicación conocida como Eclipse en su versión 4.8 (conocida como Photon) y el lenguaje de programación C++. Además se plantearon las librerías a utilizar para este proceso y la utilización del dispositivo MTX-GTW para la ejecución remota del sistema en este dispositivo.

Tareas

- Creación de la máquina virtual con el sistema operativo antes mencionado.
- Configuración de Eclipse con las correspondientes librerías.
- Preparación de los diagramas que servirán como guía para el desarrollo del sistema.
- Búsqueda de cables de alimentación para cada uno de los elementos.
- Comienzo de investigación sobre la realización de ejecuciones remotas.
- Creación del repositorio.

Backlog

En este sprint inicial se plantea cada uno de los aspectos del proyecto (requisitos, objetivos...) así como el planteamiento de la organización con el tutor en cuanto a reuniones que se iban a realizar. Además se empezó a organizar la estructura del programa con la realización de los diagramas de clases.

Investigación

Se comienza la investigación a cerca de las ejecuciones remotas de programas para poder realizarlo con el MTX-GTW. Tambien se comienza a investigar a cerca del manejo de sockets y lecturas de puertos utilizando C++ para poder recoger los datos del láser y enviar mensajes por este mismo puerto para comunicarse con el láser (con la dificultad de que esta escucha también será remota).

Sprint 1: 22/11/2017 - 16/01/2018

En este sprint se plantean los cambios a realizar en el diagrama creado en el anterior sprint, se plantea el uso de \LaTeX para la realización de la memoria del proyecto y la creación del diagrama de secuencia para mejor comprensión del funcionamiento del sistema. Se comprueba también el estado de la máquina virtual y se detecta la falta de algunas librerías necesarias.

Tareas

- Creación del diagrama de secuencia.
- Mejora del diagrama de clases según la revisión realizada.
- Solucionar la ausencia de librerías antes mencionado.
- Realización de pruebas de conexión con el MTX-GTW.
- Continuación de búsqueda de material hardware.

Backlog

En este sprint se termina de crear y modificar los diagramas que servirán de base a la posterior creación del código del sistema, se pretende configurar Eclipse con las librerías que faltaban del sprint anterior y se implementa el programa que imprima por pantalla "Hola mundo" realizando esta ejecución en el dispositivo remoto. Mientras se realizan estos procesos se prosigue con la búsqueda del cable de alimentación del láser ya que el cable del MTX-GTW ya ha sido encontrado.

Investigación

En este sprint se realizan varias tareas de investigación. En primer lugar, se deben encontrar las librerías que se necesitan para incluirlas en la configuración del proyecto en Eclipse. Esta tarea resulta costosa debido a que algunas de las librerías que se habían acordado utilizar estaban desactualizadas y no se podían encontrar y se desconocía la organización de ficheros y el funcionamiento de las nuevas versiones con lo que se hacía complicado la configuración para algunas de las tareas futuras.

La otra tarea de investigación realizada fue la búsqueda de los diferentes cables de alimentación y esto es debido a la diferencia entre ambos ya que el dispositivo MTX-GTW necesita ser alimentado con una corriente alterna de 12 V. Debido a que, en la actualidad, la mayoría de los cargadores de los diferentes aparatos electrónicos cotidianos no suministran más de 7 voltios se necesitó fabricar un cargador a base de un transformador que suministrase la corriente deseada unido a dos cable los cuales irán conectados al cabezal del aparato debido a que tampoco posee un cabezal convencional. Por otro lado, el láser se han utilizado dos cableados distintos ya que se debe conectar tanto a la red eléctrica para alimentarse como al dispositivo antes mencionado. Para la primera de las conexiones descritas se ha empleado un adaptador a corriente alterna ya que este láser necesita alimentarse con 24 V de corriente alterna. Para la segunda conexión se necesita un cable con cabezal VGA y el otro extremos sin cabezal para poder conectar solo los cables que sean necesarios ya que solo se necesitan algunos de los pines de esta conexión para el envío y recepción de datos.

Pruebas

En este sprint se comienzan a realizar las primeras pruebas para la conexión del PC con el MTW-GTW para configurar de forma correcta esta conexión con el fin de tenerla de forma definitiva como conexión de ambos sistemas en el proyecto final.

Sprint 2: 16/01/2018 - 01/02/2018

Una vez se han creado los diagramas para aclarar en funcionamiento del sistema final, se plantean las tareas para este nuevo sprint. En este caso, una vez se posee una versión estable del programa que realiza las pruebas se empieza a plantear el inicio del desarrollo del código del sistema, así como la

creación de una cuenta en la web Trello (mencionada en puntos anteriores) para la creación del panel con las tareas a realizar con lo que se consigue una version del panel de tareas para la organización de cada sprint.

Tareas

- Paso del proyecto de pruebas al repositorio para comenzar en el desarrollo del sistema
- Creación de cuenta y tablón en Trello.
- Creacion de las primeras clases.

Backlog

En este sprint se ha comenzado con el desarrollo del sistema ya en el repositorio para mantener un control de las versiones que se van realizando. Tambien se comienza a dejar constancia de cada una de las tareas que se han plantado en los sprint, dividiéndolas en diferentes categorias segun se encuentren pendientes de realizar, en curso o si ya han sido realizadas.

Investigación

Se retoma la investigación a cerca de los sockets en el lenguaje de C++ para plantear la conexión con el láser con el objetivo de organizar la estructura final de la comunicación lógica de los diferentes elementos físicos que componen el sistema.

Sprint 3: 01/02/2018 - 20/05/2018

Este es uno de los sprints más largos debido a que ha sido en el que han surgido la mayoría de problemas. En este punto, al no poder plantear nuevas tareas debido al bloqueo de las ya presentes del anterior sprint, las reuniones se fueron separando unas de otras en el tiempo.

Tareas

- Solución de los problemas de Ubuntu y Eclipse.
- Continuación de la creación de las distintas clases y procedimientos.

Backlog

En este Sprint, como se ha dicho anteriormente, surgen la mayor parte de las incidencias que impiden el avance en el proyecto por un largo periodo de tiempo. Una vez en proceso de creación del sistema, Eclipse comenzó a dar errores en cuanto a la comprensión de comandos básicos del lenguaje C++. Junto con esta clase de problemas surgió el inconveniente del fallo completo de Eclipse, ya que en la máquina virtual donde se estaba realizando todo el proceso, la aplicación dejó de funcionar, permitiendo verse en pantalla únicamente la barra de tareas. Para solucionar este fallo se realizaron algunas investigaciones sin éxito.

Investigación

Las investigaciones realizadas en este periodo de tiempo fueron varias aunque todas con el mismo objetivo, solucionar fallos.

En un inicio, los fallos en cuanto al lenguaje de programación. Para solucionar este tipo de fallo se realizó un análisis exhaustivo de la configuración completa del proyecto, incluyendo partes que no se habían visto hasta ese momento, por si se podría haber producido algún efecto en cascada que hubiese desencadenado el error (dicho análisis fue realizado de forma conjunta con el profesor). Tras comprobar que ese no era el problema se comenzó una búsqueda por diversas páginas web para encontrar soluciones de otros usuarios que hubiesen tenido el mismo problema. Al realizar esta búsqueda, no solo se descubrió que el número de personas con este error era muy reducido, sino que además se pudo observar que las soluciones que se les ofrecía para solucionarlo o no funcionaban o no eran útiles para el problema que se planteaba en nuestro caso. Como conclusión a este problema se decidió crear otro proyecto y dejar para más adelante la configuración realizada con anterioridad.

En cuanto al problema de la visualización de Eclipse, la situación era similar a la del anterior problema. En este caso el número de personas con el mismo problema era mayor pero la solución que se daba en la mayoría de los casos era la eliminación de los recursos temporales generados al arrancar Eclipse para reiniciarlo y que los crease de nuevo, intentando solventar así los fallos que hayan podido aparecer durante el arranque. Al no conseguir solucionarlo de esta manera, se optó por la opción más drástica, extraer los cambios ya realizados en el proyecto creado para evitar el primer problema,

eliminar la máquina virtual y comenzar la instalación y configuración de aplicaciones desde el principio.

Este método no funcionó, por lo que se decidió tomar un camino alternativo para el proyecto.

Sprint 4: 20/05/2018 - 13/06/2018

Tras el periodo anterior de cambios de máquina virtual y reconfiguraciones se tomó la decisión de cambiar el proyecto. A partir de este sprint el proyecto sería realizado en Windows 10 (sistema anfitrión del PC) para evitar problemas con los servicios de virtualización. Debido a esa decisión se hacia incompatible la utilización del MTX-GTW, ya que este aparato es incompatible con Windows, por lo que la ejecución del sistema se realizaría en el PC. Este aspecto solucionó otro de los problemas anteriores, las librerías. Al no tener que realizar ninguna ejecución remota, el proyecto solo iba a necesitar utilizar las librerías que ya vienen por defecto en C++, sin necesidad de descargar ninguna otra. En este periodo se comienza las primeras interacciones con el láser

Tareas

- Configuración del proyecto y desarrollo del código en Windows.
- Investigación sobre programas de manejo del láser.
- Cambios importantes en la memoria.

Backlog

En este periodo de tiempo se consigue la realización de gran parte del código que forma el sistema software ya que no se producen errores como los del anterior sprint. También se instala y empieza a investigar el funcionamiento del programa que se encuentra adjunto al láser en un CD presente en la misma caja del aparato. Junto con este programa, tambien se descarga e instala el programa RealTerm, el cual se usará para enviar y recibir mensajes por el puerto USB, ya que el láser tiene la posibilidad de conectarse al PC por esta vía.

Investigación

En este aspecto se empieza a investigar la configuración del programa RealTerm antes mencionado, dejando para más adelante la programación de la parte restante del sistema, para centrarse en averiguar como conseguir la conexión con el láser además de conocer como realizar el intercambio de información.

También se investiga la forma de configurar del programa del láser (UAM Project Designer) para poder ver el funcionamiento de este programa y la visualización por pantalla de las lecturas del láser, lo cual se podrá tomar como guía en la representación del resultado de la ejecución del sistema objetivo. Esto se consigue gracias a la configuración de los puertos USB de PC para que permitan instalar drivers no firmados (como son los del propio láser, los cuales también se tuvieron que buscar por internet para evitar la dependencia con el CD) con lo que se pudiese reconocer al láser para que el programa funcionase correctamente. Este buen comportamiento también se produjo ya que el tutor facilitó un fichero de configuración básica el cual fue enviado vía USB al propio láser, el cual lo conserva en memoria desde entonces.

Sprint 5: 13/06/2018 - 19/11/2018

En este sprint, cuya duración se debe a la aparición del último de los problemas de gran relevancia para el proyecto y la presencia del periodo de verano lo cual supone una mayor dificultad para las reuniones. En este sprint se analizan los resultados del ciclo anterior, observando las lecturas del láser por primera vez desde el inicio del proyecto. El problema antes mencionado surge al intentar crear el socket lógico en C++ tras conocer como crear los mensajes y provocar la comunicación entre los dos aparatos y conseguir así implementar una de las partes más importantes del proyecto. No se consigue (entre el alumno y el profesor) que el socket conecte el PC con el láser con lo que se dedica este periodo a solucionar este problema

Tareas

- Solucionar falta de comunicación con el láser desde el PC.
- Cambio de lenguaje del proyecto

Backlog

En este periodo se realizan algunas investigaciones con resultados nada satisfactorios y, tras no conseguir el objetivo principal de este sprint, se decide cambiar de lenguaje. En este caso se pasa todo el código existente a Python. Este lenguaje, al ser más usado, posee un soporte mucho mayor, especialmente por parte de los usuarios que lo usan para desarrollar su código, por lo que la traducción del código se realiza de forma rápida y sencilla. Gracias a la utilización de este lenguaje se consigue configurar el socket y tenerlo plenamente operativo de forma muy sencilla, con lo que en el periodo final de este sprint se consigue solucionar el problema inicial y provocar un gran avance.

Investigación

En este periodo la primera investigación, como ya se ha comentado anteriormente, es la solución del socket en el lenguaje de C++, sin resultado satisfactorio. A parte de esta, la investigación principal se centró en la creación de sockets en Python lo cual se consiguió de forma muy sencilla gracias a la ayuda proporcionada por la propia web del lenguaje y los usuarios que poseían este problema en diferentes foros web.

Pruebas

Se realizan las primeras pruebas de comunicación y obtención de respuestas por parte del láser que da como resultado la implementación del procedimiento que permite extraer la lectura y conservar esos datos en el sistema para su posterior análisis.

Sprint 6: 19/11/2018 - 11/12/2018

En este periodo de tiempo se desarrolla el análisis de los datos recibidos por el láser (descripción del análisis en el punto 4 de la memoria). Con todos nuevos procedimientos desarrollados, se crea la primera versión plenamente operativa del sistema software que conforma el objetivo del proyecto.

Tareas

- Crear procedimientos de análisis de datos.
- Crear representación gráfica del análisis.
- Integración de comentarios para mejor comprensión del código-

Backlog

Como se ha explicado antes se crea la primera versión operativa del proyecto, con interacción del usuario al introducir las coordenadas de las áreas que desea analizar (procedimiento el cual posee prevención de errores). Se añaden también comentarios al código para facilitar su lectura. Además de esto se implementa la mayor parte del código de forma modular para facilitar su escalabilidad futura.

Investigación

En este aspecto, este es el sprint en el que más se investiga ya que es en el que se desarrolla el comportamiento principal del sistema, por lo que se debe investigar la forma de hacerlo lo más eficiente y a la vez correcto.

Pruebas

En este sprint se han realizado pruebas con diferentes objetos a diferentes distancias para comprobar el correcto funcionamiento del programa. Tras algunas pruebas se demuestra que el programa es capaz de dibujar en la gráfica su entorno con mayor precisión cuanto más cerca estén los objetos que detecta.

Sprint 7: 11/12/2018 - 04/01/2019

En este sprint se pretende mejorar el comportamiento del sistema haciendo que funciones de forma continua, es decir, sin tener la necesidad de recargar el programa cada vez que se desea analizar una nueva lectura. Además de esto, se comienza el proceso de creación de la versión final de la memoria.

Tareas

- Creación de comportamiento continuo.
- Creación de versión final de la memoria.

Backlog

Trascurrido este sprint no se ha conseguido la implementación continua así que se ha devuelto el código a su forma funcional. Por la parte de la memoria se produce un avance muy significativo en cuanto a su versión final se refiere.

Investigación

Esta parte cuenta con la investigación en cuanto a las pruebas realizadas con otros comandos del láser usados para poder desarrollar el funcionamiento continuo, aunque, como ya se ha explicado en puntos anteriores, no se ha conseguido, teniendo que volver a la forma más funcional y estable que se tenía hasta el momento.

A.3. Estudio de viabilidad

En el desarrollo de este proyecto también se ha de estudiar si el sistema desarrollado es rentable y cumple con la legalidad vigente para poder instalarlo en los AGVs de cualquier empresa, es decir, se ha de estudiar su viabilidad, tanto económica como legal.

Viabilidad económica

En este caso se analizarán los costes de cada aspecto del proyecto los cuales tengan una importancia en este aspecto.

Coste de personal

En este caso se debe de hacer una estimación a cerca del coste que supone el trabajo del programador a lo largo de toda la duración del proyecto. Debido a que algunos periodos han sido vacacionales (en cuanto al calendario laboral estándar) y otros no se ha avanzado en el proyecto, se estimará un trabajo de 10 meses. Teniendo como salario (también obtenido de una estimación aproximada) de 9 euros la hora y una jornada laboral estándar de 6 horas al día y trabajando 20 días cada mes el suelo se calcularía como:

$$9^{euros}/hora * 4^{horas}/dia = 36^{euros}/dia$$

$$36^{euros}/dia * 20^{dias}/mes = 720^{euros}/mes$$

$$720^{euros}/mes * 10^{meses} = 7200^{euros}$$

Con esto se estima el coste de personal en 7200 euros.

Coste de materiales

En este caso se ha de calcular tanto el coste del sistema hardware final como el de los materiales que han sido empleados a lo largo del proyecto para el desarrollo de las investigaciones y las pruebas de otras formas de

implementación del sistema. Sabiendo esto y estimando Los costes de material de pruebas en 100 euros, el precio del láser en 400 euros, el del PC en 600 euros y 15 euros del cableado de alimentación y conexión entre los dos elementos finales del sistema, estimando una duración de 6 años para los finales y 6 meses para los materiales de prueba se pueden realizar los siguientes cálculos:

$$(400\text{euros}/(12^{\text{meses}}/\text{año} * 6\text{años})) * 6\text{meses} = 33,33\text{eurosLaser}$$

$$(600\text{euros}/(12^{\text{meses}}/\text{año} * 6\text{años})) * 6\text{meses} = 49,99\text{eurosPC}$$

$$(15\text{euros}/(12^{\text{meses}}/\text{año} * 6\text{años})) * 6\text{meses} = 1,25\text{eurosCables}$$

$$33,33\text{eurosLaser} + 49,99\text{eurosPC} + 1,25\text{eurosCables} + 100\text{eurosPruebas} = 184,57\text{eurosMaterial}$$

Con esto se estima el coste de material en 184.57 euros.

Costes software

También hay que tener en cuenta los costes en cuanto a los programas y aplicaciones que se han empleado. En este caso el único coste a tener en cuenta es la licencia del sistema operativo final, Windows 10, con un valor de 120 euros con una vida útil de 4 años. El resto de software empleado no ha tenido coste económico alguno (a excepción de el programa UAM Project Designer el cual ya se encuentra incluido en el coste del láser previamente tenido en cuenta). Por lo tanto los cálculos a realizar son:

$$(120\text{euros}/(12^{\text{meses}}/\text{año} * 4\text{años})) * 6\text{meses} = 15\text{eurosSO}$$

Con esto se estima el coste del software en 15 euros.

Costes totales

Con todos los cálculos anteriores se puede calcular de forma estimada (ya que, como bien se ha dicho, todos los cálculos se han realizado a través de estimaciones) el coste total del proyecto. Para ello, hay que hacer la suma de todos los costes y tener en cuenta los porcentajes correspondientes a la Cotización del desarrollado a la Seguridad Social. Se sabe que estos porcentajes son : 23.6 % de contingencias comunes, 5.5 % de desempleo y 0.6 % por la formación profesional. Estos porcentajes hacen un total del 29.7 %, con lo que los costes finales se calculan como: $7200 + (7200 * 0,297) = 9388,4\text{eurosDesarrollador}$
 $9388,4\text{eurosDesarrollador} + 15\text{eurosSO} + 184,57\text{eurosMaterial} = 9537,97\text{eurosTotales}$
 Con esto se estima el coste total en 9537.97 euros.

Beneficios

Después de conocer en cuanto se estima el coste del sistema para la compañía, en este apartado se calculará el beneficio que este software proporciona. Debido a la orientación de este, el software se ha de distribuir instalado en los AGVs en los cuales se va a emplear.

Se estima un total de 500 vehículos en los cuales este programa va a ser instalado, con un precio de 40 euros por cada uno de estos vehículos cada año. Con esto el beneficio sería de:

$$500 \text{vehículos} * 40^{\text{euros}} / \text{vehículo} = 20000^{\text{euros}} / \text{año} \text{de beneficio}$$

Con estos beneficios se estimará también el tiempo que e tiempo en el que la compañía será capaz de recuperar la inversión es de:

$$9537,97^{\text{euros}} \text{Totales} / 20000^{\text{euros}} / \text{año} = 0,48 \text{años}$$

La compañía recuperará la inversión realizada en este proyecto en poco menos de 6 meses.

Viabilidad legal

Para poder comprobar que el sistema implementado no incumple ninguna ley o norma vigente hemos de analizar dos aspectos: las licencias software (comprobando que todos los programas y aplicaciones utilizados posean licencia) y la legislación en cuanto a los AGV en los que irá instalado el software.

Licencias software

Para comenzar esta sección, se van a investigar las licencias que poseen los distintos programas utilizados para el desarrollo de este proyecto.

- Numpy: licencia BSD.
- Matplotlib: licencia BSD.
- Jupyter Notebook: licencia BSD.
- Spyder: licencia BSD.
- RealTerm: licencia BSD.
- Eclipse: licencia EPL.

Por la compatibilidad con la mayoría de los programas utilizados y por ser además una de las licencias adecuadas para un proyecto Python según su guía, se ha decidido que para este proyecto se va a escoger una licencia BSD (o Berkeley Software Distribution). Se ha escogido este tipo de licencias debido a la protección que ofrece, tanto al código del programa como al creador del mismo. A pesar de tener ciertos aspectos restrictivos, también posee algunas ventajas importantes:

- Esta licencia permite que otras versiones a crear por usuarios ajenos al desarrollo inicial puedan crear sus propias versiones con distintos tipos de licencia, e incluso, crear versiones y distribuirlas como software libre.
- Se permite realizar casi cualquier acción que el usuario quiera realizar sobre el software, incluyendo el uso este para productos propietarios, es decir, productos con licencias más restrictivas.
- Es una de las de las licencias que más se acerca a la definición de software libre, debido a la libertad ofrecida al usuario con respecto al software adquirido.

A parte de estas características, como ya se ha dicho con anterioridad, esta licencia también posee ciertas características no tan favorables al usuario o al software:

- Se debe incluir el reconocimiento del origen del software en cualquier anuncio o distribución realizada aunque no es obligatorio la colocación del nombre de los autores.
- No se incluye ninguna restricción en esta licencia que ayude a que las versiones sucesivas del software sean libres con lo que, como se ha comentado anteriormente, se podrían asignar licencias mucho más restrictivas a cualquiera de esas versiones.

En cuanto a la licencia de Eclipse, se ha tomado la decisión de no considerarla para tomar la decisión anterior debido a que solo se empleó esta aplicación para las fases iniciales del proyecto, por lo cual no tiene relevancia en el funcionamiento del sistema final.

Legislación de sobre la robótica

A nivel nacional no existe ningún tipo de legislación a cerca de los robots más allá de las leyes presentes para otras creaciones (Ley de Orgánica de Protección de Datos, Ley de la Propiedad Intelectual...), pero a nivel europeo se están empezando a desarrollar una serie de normativas en lo que se refiere a diferentes aspectos relacionados con la robótica las cuales, si en un futuro son transformadas en leyes, pueden afectar al desarrollo de este tipo de sistemas:

- Proteger a todo ser humano de los daños causados por un robot.
- Respetar el derecho de una persona a negarse a ser atendido por un robot (relacionado sobre todo con el cuidado de personas mayores o la robótica aplicada al sector sanitario).
- Hacer que los robots respeten la autonomía de las decisiones tomadas por seres humanos.
- Protección de datos con respecto a los robots: es decir, hacer que estos sistemas respeten la privacidad del ser humano, o lo que es lo mismo, asegurar el derecho a la intimidad.
- Permitir al un ser humano gestionar los datos recopilados por el robot sobre si mismo para, como la anterior normativa, asegurar la protección de datos personales.
- Evitar la creación de vínculos emocionales con robots: sobre todo referido a las personas mayores, niños o personas con discapacidad. Esta ley pretende evitar el aislamiento de un ser humano del resto de la sociedad, es decir, evitar la sustitución de seres humanos por robots.
- Igualdad de acceso a las mejoras en la robótica tanto para los usuarios profesionales como los que no lo son.
- Restringir al ser humano mejoras en la robótica que permita la creación de los denominados cyborgs, lo que se augura que podría causar la extinción de la raza humana.
- Necesidad de implantar un sistema de registro, responsabilidad y cobro de impuestos para los robots: ya que al realizar un trabajo que copia al de un ser humano, se necesitaría tener un registro equivalente.

- Implantación de un botón de emergencia para revertir determinadas decisiones tomadas por el robot o desconectarlo directamente. Esta normativa viene impulsada por el objetivo de preservar la seguridad del ser humano.

En resumen, las reglas que se están desarrollando con las nuevas tecnologías desarrolladas es la de orientarlas a la ayuda del ser humano evitando daños al mismo o perjuicios con respecto al resto de la sociedad.

Apéndice B

Especificación de Requisitos

B.1. Introducción

En este apartado se presentan las funcionalidades básicas que debe cumplir el sistema desarrollado para este proyecto. En este caso se van a diferenciar dos tipos: los objetivos generales, los cuales son de obligado cumplimiento, y el resto de requisitos que se mostrarán en un catálogo debido a que no es necesario que se cumplan todos.

B.2. Objetivos generales

Para el sistema software desarrollado en este proyecto, se fijan los siguientes objetivos generales:

- Establecer una conexión con el láser.
- Intercambiar mensajes para obtener los datos de lectura.
- Implementar un analizador de dichos datos.
- Implementar una representación gráfica para la visualización del análisis.

B.3. Catalogo de requisitos

Para la realización de este catálogo se diferenciará entre dos tipos de requisitos: los requisitos funcionales y los requisitos no funcionales.

Requisitos funcionales

- R1: Conectar con el láser.
- R2: Enviar mensaje.
- R3: Recoger datos.
- R4: Traducir datos.
- R5: Incorporar áreas al sistema.
- R6: Representar gráficamente los datos del análisis.
- R7: Impresión de las áreas con objetos detectados.

Requisitos no funcionales

- El sistema ha de detectar de la forma más precisa posible la distancia a la que se encuentra cada uno de los puntos detectados, ya se encuentren dentro o fuera de los campos de área establecidos.
- El software necesita estar conectado al láser para poder realizar la comunicación.
- Al ser una primera versión de este proyecto, es decir, al poder ser un sistema extensible y escalable, se ha codificado de forma modular.

B.4. Especificación de requisitos

R1: Conectar con el láser.

Versión 1.0

Autor Álvaro Ruifernández Palacios

Descripción El sistema tiene que conseguir crear conexión con el láser para poder interactuar con él y conseguir la información de sus lecturas.

Precondición Ha tenido que comenzar la ejecución del programa principal.

Secuencia Normal los pasos a seguir son:

1. El usuario ejecuta el programa principal.
2. Se crea el objeto de tipo Socket.

3. Se establece la dirección IP y el número de puerto.

Postcondición Al enviar el mensaje se recibirá una respuesta dependiendo de dicho mensaje

Excepciones El láser no esté físicamente conectado al PC.

Importancia Alta

Comentarios Sin comentarios

R2: Enviar mensaje.

Versión 1.0

Autor Álvaro Ruifernández Palacios

Descripción Se crea el mensaje para, después de haberlo codificado, enviarlo a través del socket para que este devuelva la información de lectura del entorno, es decir, se crea un mensaje de petición de datos dirigido al láser.

Precondición Se tiene que haber configurado el socket previamente (se tiene que haber cumplido R1)

Secuencia Normal Los pasos de este requisito son:

1. Se crea una lista con cada uno de los caracteres que forman el mensaje codificados según corresponda (explicado en el punto 3 de la memoria).
2. Esta lista es transformada en una array de bytes o bytearray.
3. Este bytearray es enviado al láser a través del socket mencionado previamente.

Postcondición El láser enviará la información de lectura por el mismo puerto por el que se envió la petición.

Excepciones Si el mensaje no es configurado de forma correcta (fallo en la codificación, falta de alguno de los campos...) el láser devolverá un mensaje de error.

Importancia Alta

Comentarios Sin comentarios.

R3: Recoger datos.

Versión 1.0

Autor Álvaro Ruifernández Palacios

Descripción Recoger y mantener en el sistema los datos recibidos de las lecturas del láser tras el paso del mensaje correcto correspondiente para su posterior análisis.

Precondición Se debe haber enviado el mensaje correcto, es decir, se debe haber cumplido pre y postcondiciones de R2.

Secuencia Normal Los pasos de este requisito son:

1. El láser comienza la transmisión de información.
2. La variable empieza a encadenar las parte de la cadena que forma los datos recibidos.
3. A la vez que esto, otra variable recoge la cantidad de información obtenida para recoger solo la información deseada.
4. Cuando la variable anterior llega al máximo preestablecido por el programador, se cierra el socket quedando la primera de las variables con la información de lectura deseada.

Postcondición La información de lectura que realizó el láser queda almacenada en el programa principal para posterior análisis.

Excepciones Si se interrumpe la conexión con el láser, la ejecución del programa queda en un bucle infinito al no haber completado la longitud máxima marcada por la segunda de las variables descritas en la enumeración.

Importancia Alta

Comentarios Sin comentarios.

R4: Traducir datos.

Versión 1.0

Autor Álvaro Ruifernández Palacios

Descripción Se utiliza la variable que almacena las lecturas del láser para extraer los datos y transformarlos en una colección de puntos para tenerlos el un formato amigable para el sistema.

Precondición Se debe haber cumplido R3, con lo que el sistema tiene almacenado la información de lectura a analizar.

Secuencia Normal Los pasos a seguir en este requisito son:

1. Se divide la cadena de lectura según los caracteres de inicio y final de los mensajes.

2. Se eliminan de la lista creada de la división todas las cadenas vacías.
3. Se extrae a una variable la segunda de las posiciones de la lista de cadenas (ya que la primera es información de confirmación).
4. Se elimina la información irrelevante y se divide la cadena resultante en tantos datos como pasos da el láser en cada lectura.
5. Se asigna a cada dato un ángulo de una lista previamente configurada de la misma longitud que cantidad de datos de lectura
6. Se traduce cada par de datos a valores cartesianos (ya que la asignación anterior creaba valores polares).

Postcondición El sistema posee una lista con las coordenadas cartesianas de cada una de las lecturas.

Excepciones Si los cálculos de traducción de puntos fallan, se guardarían datos erróneos en el sistema.

Importancia Alta

Comentarios Sin comentarios.

R5: Incorporar áreas al sistema.

Versión 1.0

Autor Álvaro Ruifernández Palacios

Descripción Introducir en el sistema el número y las coordenadas de los límites de las áreas que el usuario desea analizar para comprobar si existen objetos.

Precondición El sistema debe estar ejecutándose sin ningún tipo de fallo.

Secuencia Normal Los pasos a seguir son:

1. El sistema muestra un mensaje para que el usuario introduzca el número de áreas a introducir.
2. Tras haber introducido el número correctamente se muestra una sucesión de mensajes para introducir cada uno de los límites de cada una de las áreas.

Postcondición El sistema guarda los datos a cerca de las áreas a analizar para conocer si existen o no objetos en ellas.

Excepciones Si los datos son introducidos de manera incorrecta, serán

registrados incorrectamente por lo que causará errores en el análisis.

Importancia Media

Comentarios Existe la opción de no querer introducir áreas (introduciendo 0 como número de áreas a analizar) si el usuario solo quiere observar las lecturas del láser. Esta forma de funcionamiento del programa puede ser empleada para comprobar el buen funcionamiento del sistema de traducción.

R6: Representar gráficamente los datos del análisis.

Versión 1.0

Autor Álvaro Ruifernández Palacios

Descripción Como bien se describe en el título, se representan gráficamente las coordenadas tanto de las lecturas traducidas como de las coordenadas de las áreas introducidas por el usuario.

Precondición Ejecución del programa principal sin ningún fallo.

Secuencia Normal Los pasos a ejecutar son:

1. Se divide cada uno de los puntos (sea cual sea su origen) en dos listas, una con las coordenadas pertenecientes al eje de abscisas y la otra con las correspondientes al eje de ordenadas. A la hora de almacenar los datos separados en listas si que se tiene en cuenta su procedencia.
2. Se introducen en la gráfica los datos de lectura del láser en color azul.
3. Se introducen en la gráfica los datos de los límites de cada una de las áreas en color rojo.

Postcondición Se muestran todos los datos a cerca de los puntos que posee el sistema en ese momento.

Excepciones Se puede producir un error a la hora de representar si los datos de las áreas se encuentran mal introducidos.

Importancia Media

Comentarios Sin comentarios

R7: Impresión de las áreas con objetos detectados.

Versión 1.0

Autor Álvaro Ruifernández Palacios

Descripción Se imprimen los mensajes de las áreas, es decir, el sistema

dice si se ha encontrado algún punto dentro de estas áreas, lo que significaría que hay un objeto en alguna de ellas.

Precondición Los datos de las áreas deben de estar correctamente introducidos y no haber aparecido ningún mensaje de error que hubiese detenido la ejecución.

Secuencia Normal Los pasos a seguir son:

1. Se comprueba si las coordenadas de alguno de los puntos de lectura se encuentran entre los límites de alguna de las áreas.
2. En caso positivo aparece un mensaje con la posición de ese punto.
3. En caso negativo aparece un mensaje diciendo que no se encontró ningún objeto.

Postcondición El usuario es informado de si en las áreas que deseaba analizar se ha encontrado o no algún objeto, indicando la posición del objeto encontrado en caso de que así fuese.

Excepciones Se pueden producir errores dependiendo de la corrección con la que se hayan introducido las coordenadas de las áreas.

Importancia Alta

Comentarios Sin comentarios.

Apéndice C

Especificación de diseño

C.1. Introducción

En esta sección se describirá el proceso de diseño de cada uno de los aspectos del proyecto que forman la base para conseguir alcanzar los objetivos propuestos para este proyecto.

C.2. Diseño de datos

En este proyecto se emplea un único tipo de datos, las coordenadas cartesianas, las cuales se dividen en dos grupos dependiendo de su procedencia:

- Puntos de lectura: grupo formado por las lecturas que provienen de las lecturas del láser y cuya traducción de formato cadena (formato de llegada al sistema) a coordenadas cartesianas ya ha sido descrita en apartados anteriores.
- Límites de áreas: grupo formado por los límites de las áreas introducidas por el usuario, las cuales no necesitan traducción ya que son introducidas directamente como coordenadas cartesianas.

Ambos grupos se tratan de la misma forma dentro del sistema, ya que se crean listas de instancias de la clase *Punto*, creada en este proyecto.

C.3. Diseño procedimental

Para este apartado se ha desarrollado el siguiente diagrama de secuencia, en el que se describe el proceso de ejecución del sistema. Como se puede observar en la imagen del diagrama, la única interacción que tiene el usuario es la introducción del número de áreas y las coordenadas de cada uno de sus límites.

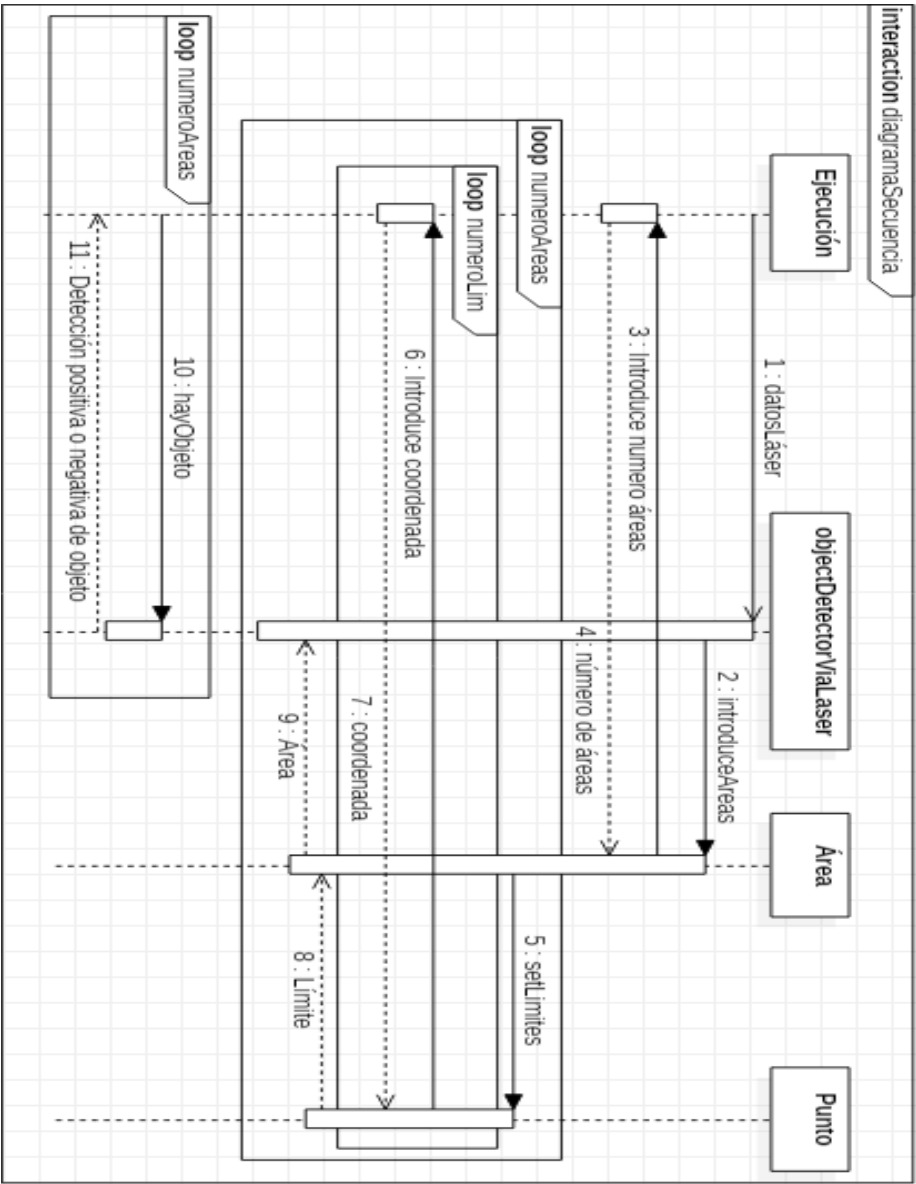


Figura C.1: Diagrama de secuencia del sistema

C.4. Diseño arquitectónico

En cuanto a la arquitectura del proyecto se destaca su simplicidad. Estos es debido a que la base del proyecto es la creación de un proceso, por lo que no posee clases como interfaces u otros elementos a dividir en varias clases.

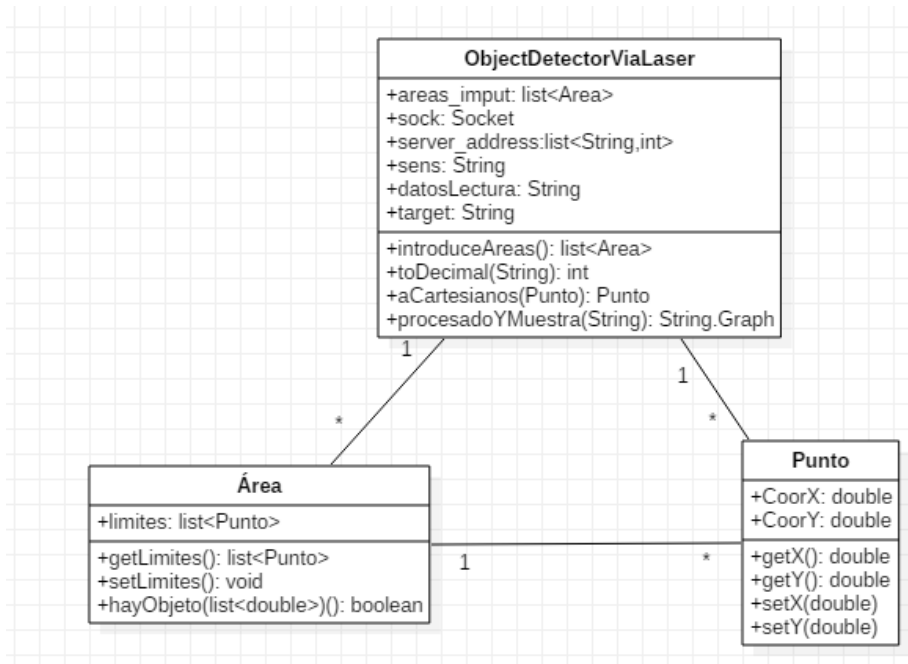


Figura C.2: Diagrama de clases del sistema

Apéndice D

Documentación técnica de programación

D.1. Introducción

En este apartado se mostrarán todas las herramientas que necesita un programador para poder utilizar y trabajar en este proyecto. Como primer paso, antes de cualquier herramienta, se necesita acceso al proyecto.

Este proyecto es accesible a través de la URL: https://github.com/arp0054/TFG_ObjDetectorViaLaser

D.2. Estructura de directorios

Debido a las diferentes formas de implementación que este proyecto ha tenido a lo largo de su desarrollo, se pueden observar una gran variedad de directorios:

- DataSheets: en este directorio se encuentran los documentos técnicos y el protocolo de comunicación utilizado por el láser, los cuales han servido para obtener una gran cantidad de datos que han ayudado al desarrollo del proyecto (dirección IP y número de puerto por defecto, formas de encriptación y desencriptación de los datos...).
- Diagramas: en este caso el nombre es muy descriptivo, en este directorio se encuentran los diagramas de clases y de secuencia, tanto los iniciales,

APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

desarrollados para las primeras fases del proyecto, como los finales. Estos diagramas se distinguen por estar separados en los subdirectorios Iniciales y Finales".

- Dispositivo: también perteneciente a las fases iniciales, en él aparecen el manual de usuario del MTX-GTW, con detalladas explicaciones a cerca de sus necesidades eléctricas así como todas las aplicaciones que el hardware de este dispositivo es capaz de realizar.
- Instalación: en este caso, aparecen tanto imágenes como documentos a cerca de como instalar las librerías las cuales permitían la ejecución remota en el sistema descrito en el punto anterior así como otras utilidades necesarias para el desarrollo del proyecto en su primera versión (útiles para su uso con C++)
- Memoria: como su mismo nombre indica, este es el directorio en el que se guardan todos los archivos e imágenes que conforman la memoria de este proyecto, así como su sección de anexos.
- Python: en este directorio se almacenan todos los archivos del sistema codificados en el lenguaje de programación con el mismo nombre, los cuales forman la versión actual y funcional del proyecto. Hay dos archivos con el mismo nombre `ObjectDetectorViaLaser` aunque con distinta extensión, dependiendo si es un archivo para usar con Jupyter Notebook (extensión ".ipynb"), el cual se utilizó primero para desarrollo y después solo para alguna prueba, o para utilización en Spyder(extensión ".py"). De la utilización de estos archivos aparece la generación automática de los dos subdirectorios pertenecientes a esta parte del repositorio.
- TFG_1718: de nuevo un directorio de las primeras versiones, ya que este alberga todas las clases, ya sean funcionales o librerías, que fueron usadas en las primeras versiones para codificar el sistema objetivo en el lenguaje de programación C++ .
- URG_USB_Driver_ en: este directorio almacena simplemente el driver utilizado para la conexión con el láser vía USB.

Como se ha podido observar, algunos de los directorios pertenecen a las primeras versiones. Esto viene dado por la motivación ya expuesta en el punto 7 de la memoria, donde una de las

D.3. Manual del programador

En esta sección se explica como se configura el entorno de trabajo para este proyecto.

Python

Como primer paso para esta configuración se debe descargar el lenguaje de programación que se va a emplear, en este caso Python. Para este proyecto no se especifica ninguna versión en especial pero para su desarrollo se ha empleado la versión número 3.6.3.

Para su descarga visitar: <https://www.python.org/downloads/release/python-363/>

Después de descargar e instalar este lenguaje, el cual tiene incorporado el manejador de paquetes *pip*, se descargan los módulos o paquetes que se emplean sobre todo en la representación gráfica con los siguientes comandos:

- `pip install matplotlib.`
- `pip install numpy`

Estas bibliotecas son las que permiten preparar y representar los datos a mostrar de forma gráfica tras haber pasado por el proceso de análisis y traducción.

IDE

Para poder modificar o crear nuevo código en este proyecto se necesita un entorno de desarrollo integrado o IDE. Aunque Python ya tiene el suyo propio, se ha optado por el uso de otros debido a que son más amigables y algunos de ellos se han empleado a lo largo del grado. Estos entornos son: Jupyter Notebook y Spyder.

Para descargar ambos entornos se debe instalar la aplicación de Anaconda Navigator, la cual incorpora a ambos. Para descargarlo hay que visitar la página web: <https://anaconda.org/anaconda/anaconda-navigator>

La principal diferencia entre ambos es que Jupyter utiliza una interfaz web, mientras que Spyder posee su propia interfaz.

APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

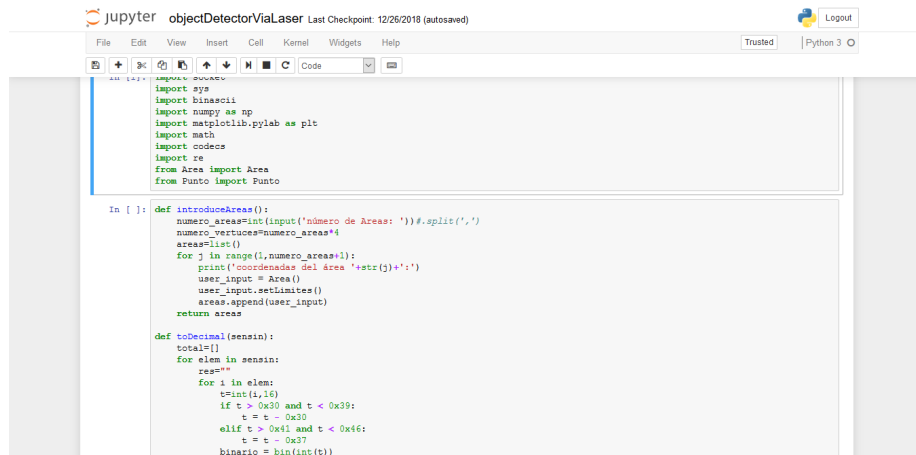


Figura D.1: Interfaz gráfica de Jupyter

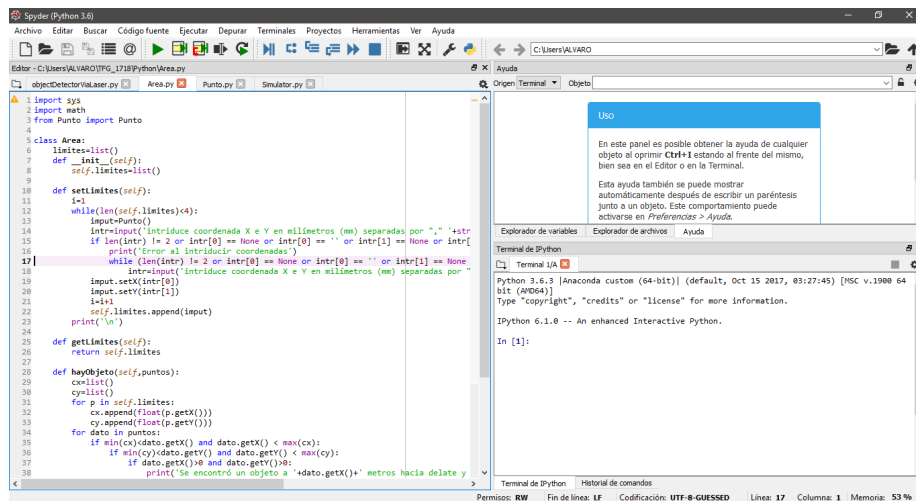


Figura D.2: Interfaz gráfica de Spyder

D.4. Compilación, instalación y ejecución del proyecto

D.5. Pruebas del sistema

Apéndice E

Documentación de usuario

- E.1. Introducción
- E.2. Requisitos de usuarios
- E.3. Instalación
- E.4. Manual del usuario