Contenido

[1 INFORMACIÓN DEL PROFESOR 2](#_Toc503264469)

[2 FUNDAMENTOS/CONTEXTO 2](#_Toc503264470)

[3 PROPÓSITOS GENERALES 2](#_Toc503264471)

[4 PROPÓSITOS ESPECÍFICOS 3](#_Toc503264472)

[5 PRE-REQUISITOS Y MATERIAS SUBSECUENTES QUE HACEN USO DE COMPETENCIAS DE PROGRAMACIÓN 4](#_Toc503264473)

[6 TEMAS Y SUBTEMAS 4](#_Toc503264474)

[7 COMPETENCIAS A DESARROLLAR 5](#_Toc503264475)

[8 EVALUACIÓN 5](#_Toc503264476)

[9 SITUACIONES DE APRENDIZAJE 9](#_Toc503264477)

[1 PROYECTO INTEGRADOR / PRÁCTICA INTEGRADORA 14](#_Toc503264478)

[a DESARROLLO DEL PROYECTO INTEGRADOR 14](#_Toc503264479)

[2 BIBLIOGRAFÍA Y RECURSOS 17](#_Toc503264480)

[3 Atributos clave del ingeniero del futuro con los que contribuye este curso 18](#_Toc503264481)

[4 APÉNDICE 1. ASIGNATURAS QUE REQUIEREN COMPETENCIAS BÁSICAS DE ALGORITMOS Y PROGRAMACIÓN 19](#_Toc503264482)

[5 APÉNDICE 2. OBJETIVOS, REQUISITOS GENERALES Y RÚBRICAS PARA LA EVALUACIÓN DEL PROYECTO INTEGRADOR . 21](#_Toc503264483)

|  |  |
| --- | --- |
| **ASIGNATURA:** Algoritmos y Programación | **CREDITOS:** BCD 4 TIE 4 |
| **CLAVE DE ASIGNATURA:** ESI2232G | **HORARIO:** Martes y Jueves de 11:00 – 13:00  **SALÓN:** B110, T104 |
| **PROGRAMA ACADEMICO:** ISC, ISI, IE,LII, IFI, IBI, LAB, IES, LIC, LIM | **IDIOMA:** Español |
| **DEPARTAMENTO:** E.S.I. | **PERIODO ESCOLAR:** Otoño 2018 |

# INFORMACIÓN DEL PROFESOR

|  |  |
| --- | --- |
| **Grado, nombre: Ing. Alejandro Rizo** | |
| **Correo Electrónico:** [alejandrorizo@iteso.mx](mailto:alejandrorizo@iteso.mx) | **Sitio de Moodle:** |

# FUNDAMENTOS/CONTEXTO

Los programas computacionales representan soluciones algorítmicas para muchos problemas de la ingeniería relacionados con el procesamiento de la información proveniente de personas, equipos o repositorios de datos (números, textos, imágenes, sonidos, etc.).

Construir un programa computacional implica analizar el problema en cuestión, modelar una solución algorítmica, codificar la solución en un lenguaje de programación y realizar la depuración, documentación y pruebas requeridas (corrección, completitud, desempeño, etc.).

Es importante dotar a todos los alumnos de las ingenierías del ITESO de competencias para que se desenvuelvan con éxito resolviendo problemas profesionales con programas de cómputo que eviten iteraciones innecesarias, ahorren tiempo y esfuerzo, den precisión y claridad a la información que requieren, entre otros objetivos clave.

En este curso se ha escogido el lenguaje Python debido a:

* Su popularidad entre ingenieros
* Su semántica directa y su sintaxis clara lo convierten en un primer lenguaje fácilmente accesible
* Es adecuado para la programación estructurada, la orientada a objetos, así como para otros paradigmas modernos
* Las numerosas bibliotecas de funciones disponibles y los kits de herramientas de terceros que extienden la funcionalidad del lenguaje a prácticamente todos los dominios.

# PROPÓSITOS[[1]](#footnote-2) GENERALES

1. Que el alumno comprenda qué tipos de problemas de su profesión pueden solucionarse a través de aplicaciones computacionales y qué beneficios tienen estas soluciones (capacidad de las herramientas).
2. Que el alumno diseñe un algoritmo para la solución de un problema de su profesión, ya sea de corte ingenieril, científico, o de gestión, haciendo uso de diagramas, pseudocódigo u otras técnicas.
3. Que el alumno desarrolle un programa de cómputo (aplicación) correcto y funcional, haciendo uso de un lenguaje de programación de alto nivel, a través de un proceso de análisis, diseño, codificación, depuración y prueba.

# PROPÓSITOS[[2]](#footnote-3) ESPECÍFICOS

Que el alumno:

1. Diseñe soluciones a problemas de su profesión, a través de aplicaciones computacionales y cuya complejidad requiera:
   1. El uso de estructuras de secuencia, decisión y repetición.
   2. La manipulación de variables numéricas, de cadenas de texto y objetos más complejos como listas y archivos.
   3. La organización de la solución en módulos.
   4. La creación de funciones y el uso de bibliotecas de funciones ya existentes.
   5. El uso de datos en memoria interna y externa.
   6. La visualización de resultados de manera textual y gráfica.

Para ello realiza abstracción de datos y operaciones, haciendo uso de técnicas como el pseudocódigo y los diagramas de flujo.

1. Analice y especifique requisitos para los productos resultantes de una aplicación computacional considerando, entre otros, a los usuarios.
2. Codifique, extienda, re-use y depure código a partir de un diseño, empleando un lenguaje de programación de alto nivel (Python en este caso).
3. Revise código propio y de compañeros a partir de una lista de verificación de buenas prácticas.
4. Escribe documentos técnicos sencillos, correctos y completos para documentar una solución y para reportar los resultados de una práctica o proyecto.
5. Evalúe la efectividad del algoritmo programado desde sus casos más triviales hasta los más extremos.
6. Trabaje en equipo asumiendo diferentes roles y dando cuenta de manera oral de sus aprendizajes y de las decisiones técnicas y de gestión tomadas durante el proyecto o tareas de desarrollo.

# PRERREQUISITOS Y MATERIAS SUBSECUENTES QUE HACEN USO DE COMPETENCIAS DE PROGRAMACIÓN

|  |  |
| --- | --- |
| **Materia(s) previa(s) y competencias** | **Materia(s) siguiente(s)** |
| **Materias de prerrequisitos**: No tiene  **Competencias previas requeridas:** las básicas de análisis, síntesis, inducción y deducción, así como conocimientos de álgebra. | No necesariamente seriadas.  Ver **Apéndice 1. Materias que requieren competencias básicas de Algoritmos y Programación de Computadoras** |

# TEMAS Y SUBTEMAS

La siguiente lista no implica que los temas se verán estrictamente en ese orden.

1. Conceptos básicos de la programación de computadoras
   1. Algoritmo y sus características
   2. Lenguajes de programación y su clasificación
2. Metodologías para la solución de problemas computables
   1. Análisis del problema: identificación del dominio de los datos de entrada y de salida y otras características, identificación de datos extremos, restricciones, algoritmo de solución
   2. Pruebas de Escritorio
   3. Notaciones para la especificación de algoritmos: pseudocódigo y diagramas de flujo
   4. Uso de herramientas computacionales para ejecutar pseudocódigos y diagramas de flujo
   5. Sintaxis de lenguajes de programación de alto nivel para codificar algoritmos
3. Datos y estructuras secuenciales
   1. Constantes y variables: tipo, identificador (nombre) y valor
   2. Objetos
   3. Notación para entrada y salida de datos en pseudocódigo y diagramas de flujo. Sintaxis de las sentencias básicas en Python para entrada y salida de datos
   4. Operaciones con números, cadenas y conjuntos
   5. Desarrollo de algoritmos con estructuras secuenciales y codificación en Python
   6. Generación de casos de prueba de dominio de variables y requisitos de usuario para entradas y salidas.
4. Estructuras selectivas o de decisión
   1. Operaciones de relación y lógicas
   2. Algoritmos con estructuras selectivas simples
   3. Algoritmos con estructuras selectivas múltiples y anidadas
   4. Pseudocódigo, diagrama de flujo y codificación en Python de estructuras selectivas
   5. Generación de casos de prueba para las decisiones
5. Estructuras repetitivas (iteraciones)
   1. Contadores, acumuladores y centinelas (banderas)
   2. Algoritmos con estructuras repetitivas con control al inicio
   3. Algoritmos con estructuras repetitivas con control al final
   4. Operaciones sobre estructuras de datos iterables: listas, arreglos (vectores), diccionarios, tuplas y conjuntos
   5. Pseudocódigo, diagrama de flujo y codificación en Python de estructuras repetitivas
   6. Generación de casos de prueba para las iteraciones
6. Descomposición modular
   1. Código fuente, objeto y ejecutable
   2. Intérprete, compilador, enlazador
   3. Entornos de programación integrados
   4. Técnicas de diseño: bottom- up y top-down
   5. Módulos
   6. Especificación e implementación (codificación) de una función
   7. Llamados de funciones (métodos). Paso de parámetros
   8. Diagrama de descomposición modular
   9. Codificación de funciones en Python
   10. Uso de bibliotecas de funciones
   11. Fundamentos de programación orientada a objetos
7. Manipulación de archivos de texto
   1. Operaciones con archivos: apertura, lectura, escritura
   2. Manejo de datos en archivos CSV
8. Temas opcionales para los que escogen hacer proyecto.
   1. Datos estructurados en json
   2. Programación con interfaz gráfica
   3. Analítica y visualización de datos
   4. Lectura y escritura de puertos de dispositivos

# COMPETENCIAS A DESARROLLAR

1. Especificar algoritmos con pseudocódigo y diagramas de flujo como parte del proceso previo para programar o para ejecutar un proceso de manera ordenada y eficiente.
2. Comprender código escrito por otros.
3. Desarrollar código propio para solucionar problemas.
4. Aprender por sí solo otro lenguaje de programación.
5. Trabajar efectivamente en equipos para desarrollar y poner a punto una aplicación de software sencilla.

Las competencias toman en cuenta el contexto, son el resultado de un proceso de integración, están asociadas con criterios de ejecución o desempeño e implican responsabilidad.

# EVALUACIÓN

Las evaluaciones finales pueden ser ordinarias, extraordinarias o a título de suficiencia. Esta asignatura no tiene un Examen Final, sino que el profesor realizará la evaluación ordinaria final considerando el logro de los propósitos específicos y propósito general ya enunciados, así como las competencias derivadas de esos logros y las calificaciones de las evaluaciones parciales y el proyecto integrador.

El alumno que considere que ya cuenta con las competencias de este curso, puede darle de baja el primer día del curso y solicitar un examen a título de suficiencia con el Coordinador de Programa Educativo.

El alumno reprobado en la evaluación final ordinaria por no cumplir el requisito de haber aprobado los dos exámenes parciales o por no haber acreditado el 80% de asistencia a clases, puede realizar un examen extraordinario (si alcanza el 60% de asistencia a clases) en las fechas previstas en el calendario escolar del ITESO, pero deberá notificar su decisión al profesor para poder programar el mismo.

**Plazos de información y realimentación de resultados de evaluaciones**: Se indican en la plataforma Moodle que soporta este curso. Tanto los plazos (que no pueden exceder de seis días hábiles en el caso de las evaluaciones no finales y de tres días hábiles posteriores al fin del curso en el caso de las evaluaciones finales) como la realimentación son específicos para cada una de las evaluaciones.

**Lugares y tiempos de las evaluaciones**: Se indican en la plataforma Moodle que soporta este curso. Pueden ser el salón de clases, el laboratorio u otra instalación de la universidad que facilite el proceso.

**Comunicación y publicación de calificaciones**: todas se harán en la plataforma Moodle que soporta el curso. Las calificaciones finales se comunicarán personalmente.

**Impugnaciones y renuncias de calificación**: Ver Reglamento de evaluaciones académicas para alumnos de licenciatura [http://www.iteso.mx/documents/2624322/0/Reglamento+de+evaluaciones+acad%C3%A9micas+para+los+estudiantes+de+licenciatura.pdf/faf8a50e-4411-4f8c-adfd-6097ef8f3881](http://www.iteso.mx/documents/2624322/0/Reglamento+de+evaluaciones+académicas+para+los+estudiantes+de+licenciatura.pdf/faf8a50e-4411-4f8c-adfd-6097ef8f3881)

**Reglamento de alumnos** <http://www.iteso.mx/documents/2624322/0/Reglamento+de+alumnos.pdf/99178436-f50d-46ea-b976-e305532664e3>

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Peso en la calificación final** | **Tipo de evaluación parcial** | | | | | | | |
| **20%** | **Examen parcial I** | | | | | Examen de reposición para reprobados del examen I y II en semana 14 o para subir calificación sin renunciar a la que ya se tiene | | |
| Se evalúan los temas ... | | | | |
| **20%** | **Examen parcial II** | | | | |
| Se evalúan los temas ... | | | | |
| **30%** | **Prácticas de laboratorio, tareas, ejercicios en clase, exámenes rápidos y torneo 1 de programación y autoevaluacion.** | | | | | | | |
| En el caso de las prácticas se considera reprobado el que no asista | | | | | | | |
| **Tareas y prácticas de laboratorio** | | | | | | | |
| • | Se describirán los entregables de cada tarea en la plataforma Moodle | | | | | | |
| • | Se entregarán a través de buzones de Moodle en formato ZIP que contiene archivo fuente (\*.py o \*.psc) y reporte de práctica (\*.doc) | | | | | | |
|  | • | Las tareas son individuales o grupales (como se especifique) | | | | | | |
|  | • | Se calificará con cero la evaluación en la que se compruebe deshonestidad académica en la primera ocasión y se hará una amonestación verbal no pública. En caso de reincidencia de deshonestidad académica se analizará el caso para determinar sanción superior.  Así se efectuará el redondeo para obtener la calificación final de prácticas, tareas y concurso 1:  … 6.49 a 6 ; 6.5 … 7.49 a 7 y así sucesivamente | | | | | | |
|  | * • | Por omisión (si en su momento no se indica), las tareas serán evaluadas siguiendo esta rúbrica: | | | | | | |
| **Característica** | | **Descripción** | | | | **Ponderación** |
| Presentación | | Claridad y corrección de la redacción y calidad de la documentación del código | | | | 20% |
| Completitud, corrección, eficiencia | | El programa hace todo lo que se pide y no tiene errores. El algoritmo utilizado es el adecuado; no incluye más de lo necesario. | | | | 50% |
| Pruebas | | Se incluyen pantallas con la mínima cantidad de corridas necesarias para demostrar que el programa cumple con los requisitos. | | | | 30% |
| **20%** | **Proyecto integrador** | | | | Se realiza en equipos de tres alumnos que deben tener al menos dos de cuatro roles principales: analista-diseñador; codificador-depurador-documentador; probador, administrador del proyecto (seguimiento, comunicación, integración).  Los detalles de los entregables se especifican en el curso de Moodle que soporta este curso | | | |
| **10%** | **Torneos de programación.** | | | | | | | |
|  | • Se evalúa que el algoritmo entregue los resultados correctos en cada caso de prueba. Se realizan dos concursos durante el periodo escolar. | | | | | | | |
| **Torneo** | | **Descripción** | | | | **Ponderación** | |
| Torneo I | | Participación individual. | | | | Porcentaje definido por el profesor, dentro del apartado “Practicas de laboratorio” | |
| Torneo II | | Participación en equipo. | | | | 10/100 puntos sobre la calificación final. | |

|  |
| --- |
| **Necesario para aprobar el curso** |
| * Aprobar los dos exámenes parciales. El que no cumpla este requisito tiene opción de reponerlos en la semana 14 (con 80% de asistencia) o tomar el examen extraordinario (al menos 60% de asistencia). * No tener una sanción que impida acreditar el curso, ya sea por más de 6 faltas u otra causa. |

|  |
| --- |
| **Consideraciones adicionales sobre horario y convivencia** |
| El lenguaje para el trato entre estudiantes y entre estudiantes y profesor será cordial y de respeto.  Colaboraremos para que no haya distracciones e interrupciones que atenten contra el objetivo común.  Como señala el reglamento, la clase comienza 10 min después de la hora en punto. Pasado dicho tiempo se pasa lista y se coloca falta. Después de los primeros 10 minutos se podrá entrar al salón o laboratorio.  La participación activa en clase es parte de la asistencia, a consideración del profesor solicitará entregar el o los producto(s) de la o las actividad(es) de clase para mantener la asistencia.  Una vez en clase, si el alumno necesita retirarse, deberá informarlo al profesor.  Las notificaciones de enfermedad, accidente u otros eventos personales se notificarán a alejandrorizo@iteso.mx para poder atender de manera personalizada la situación y que el alumno no pierda el curso. |
| **Uso de dispositivos tecnológicos y software** |
| • **Dentro de la sesión**, sólo se permite el uso de computadora **de manera exclusiva** para realizar las actividades indicadas en la misma. |

# SITUACIONES DE APRENDIZAJE

|  |  |  |  |
| --- | --- | --- | --- |
| **1. Diseñe soluciones a problemas de su profesión a través de aplicaciones computacionales y cuya complejidad requiera:**   * **El uso de estructuras de secuencia, decisión y repetición.** * **La manipulación de variables numéricas, de cadenas de texto y objetos más complejos como listas.** * **La organización de la solución en módulos.** * **La creación de funciones y el uso de bibliotecas de funciones ya existentes.** * **El uso de datos en memoria interna y externa.** * **La visualización de resultados de manera textual y gráfica.** | | | |
| **Aprendizajes Esperados** | **Actividades en clase** | **Actividades independientes** | **Productos e indicadores de evaluación** |
| Encuentra algoritmos para solucionar un problemas matemáticos y estadísticos sencillos | Análisis de casos  Análisis de algoritmos que requieren tratamiento matemático o estadístico sencillos | Análisis de casos  Lectura de textos  Análisis de algoritmos | Diseños en pseudocódigo, con diagramas de flujo y otra técnica de especificación |
| Diseña códigos que incluyan los 6 elementos de complejidad descritos. | En **laboratorio**:  Ejecutar prácticas previamente diseñadas para:   1. Diseñar soluciones con PSEint u otra herramienta para pseudocódigo y diagramas de flujo 2. Conocer, comprender y aplicar estructuras de secuencia, decisión y repetición, en pseudocódigo, con diagramas de flujo y en Python. 3. Emplear diferentes tipos de datos para realizar operaciones aritméticas, de concatenación, de conjuntos, etc., así como empleando las funciones incorporadas (*built-in*) más comunes. 4. Comprender cómo se deben organizar los programas y cómo se relacionan sus partes.   En el **salón**:  Diseño de soluciones con requisitos mencionados usando diagramas de flujo y pseudocódigo. | 1. Estudio de temas en libros y otras fuentes en Internet 2. Solución de ejercicios a mano y en computadora para elaborar diagramas de flujo, escribir algoritmos en pseudocódigo y codificar en Python. 3. Prácticas en lenguaje Python. 4. Corridas a mano de algoritmos | Reportes de prácticas de laboratorio  Soluciones en los concursos  Entregables del proyecto (para los que escogieron esta opción) |
| Determina dominio de datos de entrada y salida; selecciona tipo de datos y estructura de los mismos; define operaciones sobre datos; diseña de algoritmos con pseudocódigo y diagramas de flujos; especifica pre y post condiciones de un proceso. | En el **salón**:  Dado un problema: identificar entradas y salidas de datos, dominio de los mismos, tipos más adecuados en el lenguaje Python; determinar prerrequisitos de entrada y post requisitos (qué se debe cumplir a la salida); determinar el algoritmo que resuelve el problema, identificando las operaciones que se deben realizar sobre los datos, su secuencia, así como las posibles situaciones a considerar para la toma de decisiones dentro del algoritmo, si se requiriera; escribir el algoritmo en pseudocódigo; dibujar el algoritmo con diagrama de flujo.  En el **laboratorio**:  Analizar casos típicos de problemas y sus soluciones; codificar algoritmos escritos en pseudocódigo o especificados con diagramas de flujo; comprobar que se cumplan pre y post condiciones; depurar código. | 1)Leer textos sobre diseño de algoritmos, diseño de entradas y salidas, así como sobre determinación de pre y post condiciones.  2) Resolver ejercicios de tarea.  3) Redactar reportes técnicos para entrega de tareas.  4) Asistir a asesorías  5) Aplicar lo aprendido al proyecto (si escogieron esa opción de proyecto integrador)   1. 6) Revisiones de pares. | Pseudocódigos, diagramas de flujo, especificación de pre y post condiciones, reportes de tareas, soluciones de ejercicios  Este tema es objeto fundamentalmente del primer examen parcial. |

|  |  |  |  |
| --- | --- | --- | --- |
| **2. Analice y especifique requisitos para los productos resultantes de una aplicación computacional considerando, entre otros, a los usuarios** | | | |
| **Aprendizajes Esperados** | **Actividades en clase** | **Actividades independientes** | **Productos e indicadores de evaluación** |
| Identifica requisitos que debe cumplir una aplicación de software sencilla.  Elabora listas de requisitos funcionales y no funcionales a partir de la descripción de un problema por escrito o derivada de entrevistas a clientes y usuarios. | En el **salón**:  Especificación de requisitos funcionales y no funcionales claros, completos y verificables a partir de un problema dado por escrito. | 1)Leer textos sobre requisitos de programas.  2)Especificar requisitos funcionales y no funcionales (claros, completos y verificables) de problemas de tarea y el proyecto (para los que escogieron esa opción) | Documento con Especificación de requisitos de problemas (en forma de lista) |

|  |  |  |  |
| --- | --- | --- | --- |
| **3. Diseñe casos de prueba para un algoritmo y para un programa.** | | | |
| **Aprendizajes Esperados** | **Actividades en clase** | **Actividades independientes** | **Productos e indicadores de evaluación** |
| Genera casos de prueba que verifiquen que el algoritmo o código: considera los cursos normales de eventos en la secuencia, así como los alternativos y excepciones. | En el **salón**:  Revisión de códigos que no consideran todos los casos y su corrección.  Redacción de casos para ser probados después e codificados, fundamentalmente aquellos que “desbordan” variables, no consideran restricciones matemáticas o de la vida real. | 1)Leer textos sobre casos de prueba  2) Redactar casos de prueba a partir de requisitos o de análisis de cursos normales o alternativos de eventos. | Documento con Especificación Reportes de tareas sobre casos de prueba  Caos de prueba para el proyecto (si procede) |

|  |  |  |  |
| --- | --- | --- | --- |
| **4. Codifique, extienda, re-use y depure código a partir de un diseño, empleando un lenguaje de programación de alto nivel (Python en este caso)** | | | |
| **Aprendizajes Esperados** | **Actividades en clase** | **Actividades independientes** | **Productos e indicadores de evaluación** |
| Codifica y depura código en Python, auxiliándose de la ayuda y otros recursos en línea. Demuestra habilidades en el uso del Entorno de Desarrollo Integrado de Python.  Extiende código de terceros.  Re-usa código propio o de terceros. | En el **laboratorio**:  Codificar en Python y depurar empleando un Entorno de Desarrollo Integrado. | 1)Leer textos sobre codificación y depuración en Python.  2)Codificar y depurar los problemas de tarea, el proyecto (si se escogió) y ejercicios de libros de texto sugeridos.  3) Consultar en blogs profesionales u otros espacios en Internet, sobre soluciones a los problemas. | Código en Python depurado |

|  |  |  |  |
| --- | --- | --- | --- |
| **5. Revise código propio y de compañeros a partir de una lista de verificación de buenas prácticas** | | | |
| **Aprendizajes Esperados** | **Actividades en clase** | **Actividades independientes** | **Productos e indicadores de evaluación** |
| Encuentra anomalías en un código sin ejecutarlo, empleando una lista de verificación (revisión estática).  Reporta anomalías a pares  Corrige anomalías que le reporten. | En el **salón:**  Revisar código propio de ejercicios que oriente el profesor, guiándose por una lista de verificación y listar anomalías encontradas. | 1)Leer textos sobre revisiones estáticas.  2)Revisar códigos propios y de compañeros.  3) Corregir código a partir de reportes de compañeros  Reportar a compañeros anomalías encontradas (en tareas y proyecto). | Reporte de anomalías encontradas después de una revisión estática.  Código propio corregido a partir de reportes de anomalías de compañeros. |

|  |  |  |  |
| --- | --- | --- | --- |
| **6. Escribe documentos técnicos sencillos, correctos y completos para documentar una solución y para reportar los resultados de una práctica o proyecto.** | | | |
| **Aprendizajes Esperados** | **Actividades en clase** | **Actividades independientes** | **Productos e indicadores de evaluación** |
| Aplica conocimientos de expresión escrita y manejo de información a la redacción técnica para comunicar resultados de un trabajo de manera clara, precisa, completa.  Documenta correctamente las referencias a las fuentes de información empleadas. | En el **laboratorio**:  Consulta de documentos técnicos.  En el **salón:**  Estudio de ejemplos de documentación de diseño de algoritmos y codificación. | 1)Leer textos sobre calidad e documentación técnica y estilo.  2) Documentar el proyecto (si escogió esa opción) y demás tareas que exijan documentación técnica.  3) Escribir reportes sobre su aprendizaje | Documentos técnicos para especificación de código.  Documentos del proyecto integrador  Reportes de aprendizaje |

|  |  |  |  |
| --- | --- | --- | --- |
| **7. Pruebe código usando casos de prueba previamente diseñados.** | | | |
| **Aprendizajes Esperados** | **Actividades en clase** | **Actividades independientes** | **Productos e indicadores de evaluación** |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **8. Trabaje en equipo asumiendo diferentes roles y dando cuenta de manera oral de sus aprendizajes y de las decisiones técnicas y de gestión tomadas durante el proyecto o tareas de desarrollo.** | | | |
| **Aprendizajes Esperados** | **Actividades en clase** | **Actividades independientes** | **Productos e indicadores de evaluación** |
|  |  |  |  |

# PROYECTO INTEGRADOR / PRÁCTICA INTEGRADORA

Contribuye con todos los propósitos, pero de manera más importante con los propósitos específicos 1, 5 y 7.

En todos los casos se participa en un equipo de tres alumnos. Se desarrolla un proyecto de programación asociado con la ingeniería, la ciencia o los negocios, de preferencia asociado a un proyecto de otra asignatura que estén tomando. Existen opciones con puntuación extra (de manera que puedan subir algunos puntos en la calificación de este apartado) sobre los resultados generales alcanzados al final del semestre)

# DESARROLLO DEL PROYECTO INTEGRADOR

En el **Apéndice 2** se encuentran los objetivos, requisitos generales y rúbricas para la evaluación del proyecto integrador.

**Inicio del proyecto**: semana 5

**Fin del proyecto:** semana 16 (con la presentación en clase de resultados)

**Pasos**

1. Definir el alcance del proyecto, precisando los principales entregables.
2. Seleccionar metodología de trabajo y planear al menos una revisión parcial y una iteración para mejorar y corregir anomalías detectadas en el avance.
3. Hacer diagrama de descomposición del trabajo incluyendo tareas técnicas, de gestión, de revisiones y de evaluación. Incluir asesorías, autoestudio y documentación de resultados.
4. Asignar roles a los participantes y otros involucrados. Todos los integrantes tendrán dos roles, siendo uno de ellos el de desarrollador

|  |  |
| --- | --- |
| **Rol** | **Responsabilidades** |
| Desarrollador (Analista-diseñador-programador) | Especifica requisitos funcionales y no funcionales del programa; analiza los datos de entrada y de salida (tipo, estructura); diseña los algoritmos y los datos, así como la interfaz de usuario y con dispositivos; diseña los casos e prueba; documenta sus análisis y diseños; codifica en Python lo diseñado. |
| Administrador del proyecto | Coordina a los integrantes y garantiza la integración de los mismos, así como de los productos que cada uno desarrolla; da seguimiento a los avances; convoca a juntas de seguimiento; garantiza la buena comunicación entre los involucrados; facilita los acuerdos; vela por el cumplimiento en tiempo y calidad. |
| Control de calidad (revisor-probador) | Realizan las revisiones y pruebas necesarias para que todos los entregables del proyecto tengan la calidad requerida, tanto código como documentación de análisis y diseño. |

1. Planificar las actividades en 12 semanas comenzando en la semana 4 de clases.
2. Ejecutar y controlar los avances del proyecto.
3. Cierre y presentación de resultados

**Tipos de proyectos**

**De preferencia multidisciplinarios y que empleen analítica y visualización.**

1. Empleando datos de [http://www.fao.org/faostat/en/#country/138](http://www.fao.org/faostat/en/" \l "country/138) Mostrar información relevante sobre el estado de la agricultura y los alimentos en México. FAOSTAT es de la Food and Agricultural Organization of United Nations.
2. Agua. Por ejemplo, procesando y visualizando datos sobre el uso del agua en México <http://mexico.globalwaterwatch.org/datos-de-agua/ver-datos/> Datos de química, bacteria, caudal, sólidos. Parece que no hay de Jalisco
3. <https://opensource.com/life/14/9/visualize-nanotechnology-3D-open-source-software> Visualización 3d con Tomviz. ¿Hay imágenes producidas en el laboratorio de nano que pueden manipular en Python los alumnos?
4. Juego educativo <https://opensource.com/article/17/10/python-101?utm_campaign=intrel>
5. Proyectos de hidrología:

* Análisis de series de tiempo de precipitación en la ciudad
* Relación entre temperatura y precipitación. Correlaciones múltiples para determinar características del suelo

1. Proyectos de correlación de variables para determinar la calidad del aire (ozono, pm10, pm2.5, óxidos de nitrógeno)
2. Procesar datos de equipos de laboratorio. Algunas ideas para proyectos en el área de la ingeniería química, ambiental y biotecnología que pueden implementarse con Python y usando la plataforma Arduino[[3]](#footnote-4) son:

* Diseño y control de un medidor de humedad para suelos, basado en sensores de capacitancia.
* Control de humedad y temperatura de un invernadero casero por medio de sensores de humedad relativa y temperatura.
* Construcción de un colorímetro de filtro para medir concentraciones de soluciones, utilizando una fotorresistencia como sensor.
* Construcción de un turbidímetro para medir concentraciones de soluciones, utilizando una fotorresistencia como sensor.
* Medición y control de flujo en tuberías utilizando sensores de nivel, rotámetros y sensores de presión.
* Construcción y calibración de un termómetro digital.
* Demostración de las leyes de los gases utilizando sensores de presión y adquisición de datos automatizada.

Hay muchas otras posibilidades. En estas dos páginas web pueden encontrar gran cantidad de sensores para Arduino:

<https://www.taloselectronics.com/categoria-producto/sensores/>

<http://www.labelectronica.com/sensores-transductores_qO9XtOcxSM>

# BIBLIOGRAFÍA Y RECURSOS

El material de estudios fundamental es el primero que aparece en la tabla que se muestra a continuación. Se listan otros recursos que podrán ayudar en el aprendizaje.

|  |  |  |
| --- | --- | --- |
| **Título / Tipo de recurso** | **Autor** | **Editorial-Año /No. Referencia biblioteca ITESO/acceso** |
| Starting Out with Python | Tony Gaddis | Pearson, 2017, Cuarta Edición |
| How To Think Like a Computer Scientist. Learning with Python. | Peter Wentworth, Jeffrey Elkner, Allen B. Downey, and Chris Meyers | <http://openbookproject.net/thinkcs/python/english3e/> |
| Tutoriales para aprender Python |  | <http://learnpython.org/> |
| Learn how to program in Python by building a simple dice game |  | <https://opensource.com/article/17/10/python-101?utm_campaign=intrel> |
| Beginner's Guide to Python |  | <https://wiki.python.org/moin/BeginnersGuide> |
| PEP 8 -- Style Guide for Python Code | Guido van Rossum <guido at python.org>, Barry Warsaw <barry at python.org>, Nick Coghlan <ncoghlan at gmail.com> | <https://www.python.org/dev/peps/pep-0008/> |
| Python 3.6.0 documentation. The Python Tutorial. |  |  |
| Metodología de la programación – algoritmos, diagramas de flujo y programas. / libro | Cairó, Osvaldo. | Alfaomega, 2015. / 005. 1 CAI |
| Python 3 al descubierto 2da edición. /libro | Arturo Fernández Montoro | Alfaomega, 2013. / 005. 133 FER |
| Introduction to Programming Using Python, Boston, EUA : Pearson, 2013, / libro | Liang, Y. Daniel | Prentice Hall, 2013. / 005. 133 LIA / 5 ejemplares en Biblioteca |
| Introduction to Programming in Python. An interdisciplinary approach / libro | Robert Sedgewick Kevin Wayne and Robert Dondero | Princeton University, Addison Wesley, 2015  En biblioteca |
| **Automate the Boring Stuff with Python - Practical Programming for Total Beginners** | Al Sweigart | <https://automatetheboringstuff.com/>  online course on Udemy.com covers most of the content of the book |
| Beginning Programming with Python for dummies | John Paul Mueller |  |
| Fundamentals of Python®: Data Structures  Se puede leer en línea entrando por la Biblioteca ITESO | Lambert, Kenneth | Cengeage Learning 2014 |

# Atributos clave del ingeniero del futuro[[4]](#footnote-5) con los que contribuye este curso

* Analítico y práctico
* Minucioso y orientado a los detalles en el diseño
* Creativo e innovador;
* Comunicativo;
* Conocedor de la aplicación de las ciencias y las matemáticas;
* Capaz de proporcionar liderazgo: capacidad para efectuar cambios en estrategias, tácticas, políticas y procedimientos en proyectos y otros roles;
* Actitud profesional y positiva;
* Consciente y cumplidor de las leyes, regulaciones, estándares y códigos relevantes;
* Licenciado como ingeniero profesional y conocedor de la ética de la ingeniería y los códigos de conducta profesional aplicables; y
* Dedicado al aprendizaje permanente
* Si llegaste a este punto, solicita tu mazapán antes de la sesión del miércoles 24/ene a Alejandro Rizo.

# APÉNDICE 1. ASIGNATURAS QUE REQUIEREN COMPETENCIAS BÁSICAS DE ALGORITMOS Y PROGRAMACIÓN

|  |  |
| --- | --- |
| **INGENIERÍA** | **ASIGNATURAS QUE REQUIEREN COMPETENCIAS DE ALGORITMOS Y PROGRAMACIÓN** |
| 1. ALIMENTOS |  |
| 1. AMBIENTAL | 1. Sensores remotos y sistemas de información geográfica 2. Modelación de sistemas ambientales 3. Geoestadística 4. Algoritmos de solución numérica 5. Ingeniería eléctrica 6. Proyectos de aplicación profesional 7. Procesos de tratamiento de agua y control de efluentes 8. Gestión de cuencas y territorio |
| 1. BIOTECNOLOGÍA | 1. **Simulación de bioprocesos**: Se usa Aspen, Comsol, Matlab y Simulink. 2. **Cinética química y biológica**: para la obtención y procesamiento de datos experimentales que miden velocidad de reacción química, enzimática o biológica. 3. **Ingeniería de biorreactores**: útil para la modelación y diseño de biorreactores. 4. **Biosensores y control analítico:** para la programación de dispositivos de instrumentación y control. |
| 1. CIVIL | Simulación matemática, Simulación estructural, Hidráulica II, Análisis estructural, Elemento finito aplicado a la Ingeniería Civil, Proyectos estructurales, Ingeniería sísmica y mampostería, algunos proyectos de Aplicación profesional (PAP) |
| 1. ELECTRÓNICA |  |
| 1. EMPRESAS DE SERVICIOS |  |
| 1. FINANCIERA |  |
| 1. INDUSTRIAL |  |
| 1. MECÁNICA | **Elementos Finitos:** Resolución de sistemas de ecuaciones, soluciones iterativas, uso eficiente de algoritmos para llenado y resolución de matrices de gran tamaño (decenas o inclusive centenas de filas y columnas de una matriz). **Cinemática de mecanismos:** Resolución de sistemas de ecuaciones, incluyendo números complejos. Manejo de variables que crecen y decrecen de tamaño dentro de ciclos.  **Instrumentación y control de procesos:** Se utiliza programación para el estudio del comportamiento dinámico de sistemas que están descritos por ecuaciones diferenciales en función del tiempo.  **PAP:** Necesidad de elaborar algoritmos que descarguen/carguen información automática de sitios o archivos externos a Python/Matlab en variables para su manipulación. De igual forma, se utilizan estos programas frecuentemente para comunicarse con hardware para detectar condiciones de mecanismos o condiciones atmosféricos, y controlar subsistemas como motores, frenos y transmisiones. |
| 1. NANOTECNOLOGÍA |  |
| 1. QUÍMICA | 1. **Simulación de procesos químicos**: impacta de manera directa, pues se usa Aspen, Comsol, Matlab y Simulink para simular sistemas químicos. 2. **Cinética química y biológica**: se requiere la obtención y procesamiento de datos producto de experimentos de laboratorio donde se miden propiedades físicas o químicas relacionadas con la velocidad de reacción. 3. **Reactores químicos**: el modelado matemático de los reactores se facilita mucho si se cuenta con conocimientos de programación. 4. **Instrumentación y control de procesos:** la programación es útil para estudiar el comportamiento dinámico de sistemas que están descritos por ecuaciones diferenciales en función del tiempo. 5. **Proyecto de aplicación profesional**: hemos tenido casos donde es necesario controlar instrumentos de laboratorio y adquirir datos por medio de programas escritos en Python o Matlab. |
| 1. SEGURIDAD Y REDES | **Programación Estructurada, 2) Programación Orientada a Objetos, 3) Diseño de Bases de Datos y 4) Desarrollo de Aplicaciones y Servicios Web.**  Otras materias que requieren competencias de la materia son:  5) **Seguridad en Redes, 6)** **Administración de Redes, 7) Ethical Hacking, 8) Análisis de Señales de Comunicación, 9) Redes para Sistemas Embebidos, 10) Internet de las Cosas y 11) Computo en la Nube -** Las materias requieren que los alumnos implementen diferentes soluciones en algún lenguaje de programación, por lo tanto, el alumno requiere tener los fundamentos de programación y la capacidad de plantear problemas como algoritmos. |
| 1. SISTEMAS COMPUTACIONALES | Programación Estructurada, Programación Orientada a Objetos, Programación con Memoria Dinámica, Lenguajes Formales, Diseño de Bases de Datos, Fundamentos de Sistemas Digitales, Estrategias Algorítmicas, Programación de Aplicaciones de Escritorio, Gestión de Bases de Datos, Fundamentos de Microprocesadores y Microcontroladores, Desarrollo de Aplicaciones y Servicios Web, Bases de Datos para Apoyar la Toma de Decisiones, Aprendizaje Máquina, Programación de Dispositivos Móviles, Internet de las Cosas (IoT), Diseño de Software, Fundamentos de Sistemas Operativos, Cómputo en la Nube. |

# APÉNDICE 2. OBJETIVOS, REQUISITOS GENERALES Y RÚBRICAS PARA LA EVALUACIÓN DEL PROYECTO INTEGRADOR.

1. **Objetivos del proyecto**

* Aplicar los conocimientos y competencias en la solución de un problema profesional sencillo
* Que demuestren que pueden adquirir nuevos conocimientos y competencias de programación a través del autoaprendizaje y empleando fuentes diversas de información
* Que comprendan mejor la importancia de tener competencias de desarrollo de algoritmos y programación para su vida profesional
* Que desarrollen competencias de trabajo en equipo y de comunicación oral y escrita

1. **Requisitos generales del proyecto y sus entregables**

* El código generado debe tener un grado de complejidad suficiente como para incluir los 6 elementos descritos en el Propósito Específico 1 de este curso
* Debe dar crédito a colaboradores, de existir, que no sean alumnos del curso o miembros del equipo
* La solución debe ser original, aunque contenga parte de código elaborado por terceros.
* El entregable final debe contener los archivos de código fuente y ejecutable, presentación en Power Point, así como los documentos en Word que especifiquen requisitos, diseño, y conclusiones.

1. **Rúbricas**

**Evaluación técnica asociada a los propósitos específicos 2, 11 y 12**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Elemento de evaluación/Escala** | **No aceptable (2)** | **Cumple con el mínimo (3)** | **Adecuado (4)** | **Excelente (5)** | **Recomendaciones del evaluador** |
| a. Uso de estructuras de secuencia, decisión y repetición | Ausencia de estructuras de decisión o de repetición en el programa resultante o no saber dar cuenta del uso de cualquiera de las tres estructuras. | Se usan las estructuras de manera correcta y el alumno sabe explicar cómo operan. El alumno sabe explicar el uso de variables y operaciones en las estructuras de secuencia y repetición. | Además de usarlas de manera correcta, se usan de manera eficiente y clara. Se emplean comentarios. Los nombres de variables ayudan a la comprensión del código. | Además de lo anterior, contemplan todos los casos de cursos alternativos de eventos y manejo de errores. Demuestran un alto dominio del uso de las estructuras de iteración en Python. |  |
| b. Manipulación de variables numéricas, de cadenas de texto y objetos más complejos como listas | Ausencia de alguno de los tipos de datos mencionados o no saber dar cuenta del manejo de datos en el proyecto. |  |  |  |  |
| c. Organización de la solución en módulos |  |  |  |  |  |
| d. Creación de funciones y el uso de bibliotecas de funciones ya existentes |  |  |  |  |  |
| e. Uso de datos en memoria interna y externa | No manipulan archivos. | Manipulan datos en memoria interna y externa, pero |  | Además de lo anterior, demuestra tener un buen dominio de estructuras iterables como matrices, tuplas, listas, arreglos, etc., así como las operaciones con archivos y el manejo de excepciones durante una sesión de trabajo con archivos, de manera que no se corrompan los mismos por falta de energía u otra excepción. |  |
| f. Visualización de resultados de manera textual y gráfica. Incluye mensajes, ayudas e instrucciones para el usuario. | La implementación de las salidas, debido a su diseño o codificación, no presenta resultados correctos, completos y claros. | Las salidas del programa son completas, claras y correctas, ya sean para un usuario o dispositivo. | Además del anterior, tiene un buen diseño de interfaz hombre- máquina (distribución espacial, uso de colores, fuentes, etc.). |  |  |

**Evaluación asociada a roles asumidos en el equipo de trabajo y decisiones de gestión del proyecto asociadas al propósito específico 2**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Elemento de evaluación/Escala** | **No aceptable (2)** | **Cumple con el mínimo (3)** | **Adecuado (4)** | **Excelente (5)** | **Recomendaciones del evaluador** |
| Cumplimiento de las responsabilidades de su rol | Incumplió con responsabilidades de su rol que provocaron que otro integrante tuviera que asumirlas o provocaron que no se concluyera el trabajo | Cumplió con las responsabilidades de su rol, aunque sus múltiples errores provocaron retrasos o re-trabajo. | Cumplió con sus responsabilidades sin provocar retrasos o re-trabajos significativos | Cumplió con sus responsabilidades de manera eficiente, concluyó en tiempo y forma y pudo dedicar tiempo a colaborar con otros para obtener mejores resultados, sobre todo en revisiones, correcciones y documentación. |  |
| Papel en la toma de decisiones | Se limitó a realizar tareas técnicas de manera individual sin interactuar con otros |  |  |  |  |

**Evaluación de la calidad de la presentación oral asociada al Propósito Específico 2**

* 1. Claridad en la comunicación oral durante las presentaciones técnicas
  2. Claridad en la comunicación escrita de presentaciones
  3. Completitud de la información requerida para conceptualizar el Trabajo de Obtención de Grado.
  4. Corrección de los elementos técnicos presentados

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Elemento de evaluación/Escala** | **No aceptable (2)** | **Cumple con el mínimo (3)** | **Adecuado (4)** | **Excelente (5)** | **Recomendaciones del evaluador** |
| Claridad en la comunicación oral durante las presentaciones técnicas. Buen uso del tiempo. | El tiempo no fue suficiente, no logró que la audiencia comprendiera de qué trata el trabajo y qué productos entregará. Problemas de expresión oral (cantinfleo, tiene cacofonías, divaga, repite) | Usó el tiempo de manera adecuada y se comprendió someramente en qué consiste el proyecto y sus productos. Expresión oral con problemas que puede mejorar con la práctica | Usó el tiempo de manera adecuada, comunicó adecuadamente en qué consiste el proyecto y sus productos. Expresión oral satisfactoria. Lee poco y usa lo escrito como soporte sintético del mensaje a transmitir. | Sintético, claro, fluido. Perfectamente comprensible el proyecto y sus productos. Convenció sin dejar dudas de la pertinencia de su propuesta. |  |
| Claridad en la comunicación escrita de presentaciones | Presentación poco clara. Problemas de ortografía o redacción recurrentes. Uso inadecuado de gráficas o textos no propios sin referencia. Diapositivas abigarradas. | Uno o dos errores de redacción u ortografía. Secuencia clara y anomalías menores en citas, tablas o gráficos | Sin errores de ortografía y redacción, pero con pequeñas mejoras en elementos como orden, apoyo audiovisual, etc.. | No hay recomendaciones de mejora en ortografía, redacción, ni en secuencia y uso de recursos visuales. Los textos son sintéticos. Excelentes apoyos audiovisuales para la mejor comprensión del producto, la competencia u otro. |  |
| Completitud y calidad de la información requerida para conceptualizar el Trabajo de Obtención de Grado | Dejó de abordar dos o más elementos de la Hoja de Concepto, sus abordajes fueron erróneos o hay falta de sustento en dos o más afirmaciones. | Dejó de abordar un tema de la Hoja de Concepto, su abordaje fue erróneo o hay aseveraciones sin sustento. | Abordó todos los temas y aunque no hay errores, si hay falta de sustento en alguna afirmación, lo que no implica falta e rigor consistente en el trabajo. | Todos los temas bien planteados y argumentados con rigor. |  |

1. Los propósitos se identifican más con el constructivismo y los objetivos se identifican más con el conductismo. Los objetivos tienen metas estrictas y definidas a las que se tiene que llegar después de varias acciones. El propósito es una intención del proceso al irse construyendo el aprendizaje. Para la asignatura establecemos propósitos; para algunos entregables como el proyecto integrador, establecemos objetivos. “componente que da intencionalidad y direccionalidad a la planificación. Son una explicación de la aspiración que orienta el quehacer educativo y que expresan los aspectos deseables o que se quieren conseguir. Los propósitos nos permiten tener claro el desde dónde, para qué, el qué, el cómo del proceso pedagógico; el sentido y finalidad que se pretende alcanzar” <http://www.efdeportes.com/efd170/construir-objetivos-propositos-o-competencias.htm> [↑](#footnote-ref-2)
2. Están redactados en función de las competencias que se pretenden desarrollar “Las competencias se encuentran estrechamente ligadas a conocimientos sólidos, ya que su realización implica la incorporación y la movilización de conocimientos específicos, por lo que no hay competencias sin conocimientos. Una competencia implica un saber hacer (habilidades) con saber (conocimiento), así como la valoración de las consecuencias de ese hacer (valores y actitudes)” (Programas de Estudio, 2009:12). Refiere Monereo (2001:15) que demostrar competencia en algún ámbito de la vida conlleva resolver problemas de cierta complejidad, encadenando una serie de estrategias de manera coordinada.”

   “Para Coll (2007:20) el concepto de competencia pone el acento en el hacer y en el saber hacer, es decir, en la movilización o aplicación del conocimiento, subrayando de este modo la importancia de la funcionalidad de los aprendizajes escolares. No basta con adquirir unos conocimientos, retenerlos y memorizarlos, ni siquiera con memorizarlos comprensivamente; además, hay que movilizarlos, hay que utilizarlos cuando la situación y las circunstancias lo requieran. “<http://www.efdeportes.com/efd170/construir-objetivos-propositos-o-competencias.htm>

   Se redactan en 3ra persona del singular; presente; subjuntivo. Es lo correcto para un propósito. [↑](#footnote-ref-3)
3. **Arduino** (**Genuino** a nivel internacional hasta octubre 2016), es una compañía de [hardware libre](https://es.wikipedia.org/wiki/Hardware_libre) y una comunidad tecnológica que diseña y manufactura [placas de desarrollo](https://es.wikipedia.org/wiki/Circuito_impreso) de hardware, compuestas por [Microcontroladores](https://es.wikipedia.org/wiki/Microcontrolador), elementos pasivos y activos. Por otro lado las placas son programadas a través de un [entorno de desarrollo](https://es.wikipedia.org/wiki/Entorno_de_desarrollo) (IDE), el cual compila el código al modelo seleccionado de placa.

   Arduino se enfoca en acercar y facilitar el uso de la electrónica y programación de sistemas embebidos en proyectos multidisciplinarios.[1](https://es.wikipedia.org/wiki/Arduino" \l "cite_note-Wiring-1)​[2](https://es.wikipedia.org/wiki/Arduino" \l "cite_note-2)​Toda la plataforma, incluyendo sus componentes de hardware (esquemáticos) y Software, son liberados con licencia de [código abierto](https://es.wikipedia.org/wiki/Código_abierto) que permite libertad de acceso a ellos.[3](https://es.wikipedia.org/wiki/Arduino" \l "cite_note-3)​

   El [*hardware*](https://es.wikipedia.org/wiki/Hardware) consiste en una placa de circuito impreso con un microcontrolador, usualmente [Atmel AVR](https://es.wikipedia.org/wiki/AVR), puertos digitales y analógicos de [entrada/salida](https://es.wikipedia.org/wiki/Entrada/salida),[4](https://es.wikipedia.org/wiki/Arduino" \l "cite_note-Arduino-4)​ los cuales pueden conectarse a placas de expansión (shields), que amplían los funcionamientos de la placa Arduino. Asimismo, posee un puerto de conexión USB desde donde se puede alimentar la placa y establecer comunicación con el computador.

   Por otro lado, el *software* consiste en un entorno de desarrollo ([IDE](https://es.wikipedia.org/wiki/Ambiente_de_desarrollo_integrado)) basado en el entorno de [*processing*](https://es.wikipedia.org/wiki/Processing) y [lenguaje de programación](https://es.wikipedia.org/wiki/Lenguaje_de_programación) basado en Wiring, así como en el [cargador de arranque](https://es.wikipedia.org/wiki/Cargador_de_arranque) (*bootloader*) que es ejecutado en la placa.[4](https://es.wikipedia.org/wiki/Arduino" \l "cite_note-Arduino-4)​ El microcontrolador de la placa se programa mediante un computador, usando una comunicación serie mediante un convertidor de niveles RS-232 a TTL. <https://es.wikipedia.org/wiki/Arduino> [↑](#footnote-ref-4)
4. tomado de <https://www.nspe.org/sites/default/files/resources/nspe-body-of-knowledge.pdf> [↑](#footnote-ref-5)