Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA EN
TECNOLOGÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Plataforma software de automatización de medidas de laboratorio.

Alumno: Agirre Ezama, Harkaitz
Director: de Diego Rodrigo, José Miguel

Curso: 2018-2019

Fecha: *12 de Noviembre de 2018*

Resumen

Resumen

Este documento describe el software para la automatización de medidas de laboratorio *HARKD*, el cual se compone de dos programas;

- *HARKDC* – Programa de terminal/consola que sirve de «backend», se comunica con los instrumentos de laboratorio y contiene una colección de algoritmos de medidas.
- *HARKDW* – Programa gráfico para realizar las medidas. Actúa como «frontend» y se comunica con *HARKDC*.

El software se ha desarrollado en el lenguaje C. Se ha utilizado un esquema de desarrollo modular por lo que añadir soporte para mas pruebas y maquinas es relativamente sencillo.

Summary

This document describes the laboratory measurement automation tool *HARKD*, which consists on two programs;

- *HARKDC* – A console based program that serves as «backend». It communicates with the laboratory instrumentation and contains the proceedings for carrying on the measurements.
- *HARKDW* – A graphical program to perform measurements. It acts as a «frontend» and uses *HARKDC* in the background.

This software has been developed in the C language. It uses a modular approach, so adding support for new tests and devices is relatively easy.

Laburpena

Dokumentu honek *HARKD* laborategiko neurketa automatizatze erraminta deskribatzen du. Bi programez osaturik dago;

- *HARKDC* – Kontsolan oinarritutako programa, «backend» modura jokatzen du. Laborategiko makinekin komunikatzen da eta neurketen nola egin daki.
- *HARKDW* – Neurketak egiten dituen programa grafikoa. «Frontend» jokabidea dauka, *HARKDC* erabiltzen du atzetik.

Software hau C hizkuntzan eraiki da. Eskema modular bat segitzen du, proba eta gailu berrientzako euskarria ematea erraza izan dadin.

Índice

Lista de tablas

Lista de figuras

1 INTRODUCCIÓN

Desde ya bien entrado el siglo XX las computadoras personales han suplantado a las personas en la realización de tareas repetitivas. La automatización de procesos es un método clave para mejorar la eficiencia en casi toda organización, la docencia no es una excepción.

En los laboratorios de la *Universidad del País Vasco* los alumnos realizan una cantidad considerable de experimentos que, en muchas ocasiones, necesitan de una cantidad aún mayor de medidas. La ejecución de estas medidas puede llegar a muy arduo y no mejora el entendimiento del problema por parte de los alumnos.

La automatización de las medidas mediante *HARKD* permite a los alumnos realizar experimentos repetibles, además de que les proporciona un mejor método para realizar experimentos de mayor complejidad.

El programa gráfico *HARKDW* hace uso de *HARKDC* para automatizar la medida de la curva característica de un conversor DC/DC. Este se va a usar en *La escuela de Ingenieros de Bilbao* en la asignatura de *Sistemas de alimentación* y de *Electrónica de potencia*, en este a los alumnos se les requiere diseñar un conversor DC/DC y medir su curva característica «$V-I$ y $\eta-I$».

Primero se explica el contexto del proyecto, se ponderan las alternativas barajadas y se expone un plan de acción. También se incluye el presupuesto del proyecto y los riesgos que en la ejecución de este se pueden ocasionar.

A continuación se presenta como se ha desarrollado la programación del software y se detallan las distintas abstracciones utilizadas mediante unos esquemas.

2 CONTEXTO

Actualmente la *Universidad del país vasco* consta de una gran variedad de instrumentos o maquinas en sus laboratorios. La mayoría de estos instrumentos utilizan distintos interfaces para comunicarse con el ordenador.

2.1 Instrumentos de medida.

El banco de medida típico de un laboratorio de electrónica consta de cuatro instrumentos indispensables: Osciloscopio, generador de funciones, fuente de alimentación y Multímetro. además del ordenador.

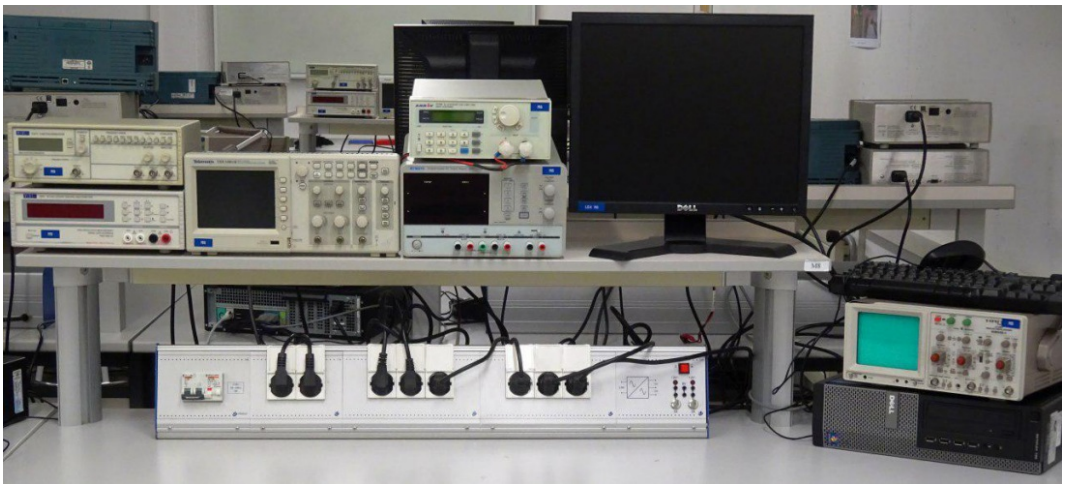


Figura 1. Mesa de medidas.

Osciloscopio

Permite visualizar la evolución en el tiempo de las señales de tensión. Este instrumento es solo de medida.

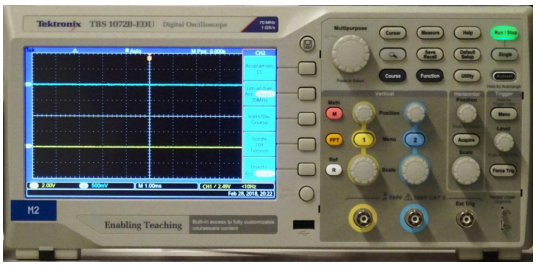


Figura 2. Osciloscopio.

Generador de funciones

Permite generar señales que cambian en el tiempo. Normalmente son capaces de generar señales básicas como triangular, sinusoidal y cuadradas. Las tensiones generadas se suelen introducir en las entradas de los circuitos electrónicos.

Figura 3. Generador de funciones.

Fuente de alimentación de CC ajustable

Genera una tensión constante para alimentar el circuito. Se puede ajustar la corriente máxima para poder experimentar con el circuito de forma segura.

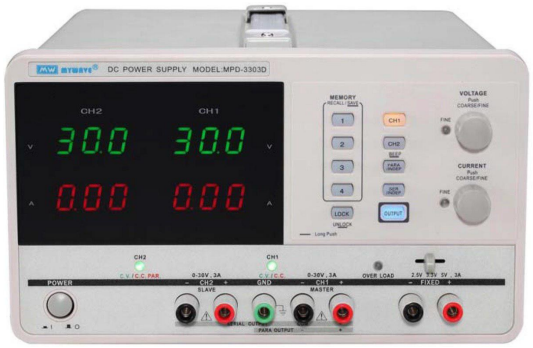


Figura 4. Fuente de alimentación de CC.

Multímetro

Un instrumento que se utiliza para medir resistencias, tensiones e intensidades y comprobar el correcto montaje del circuito.

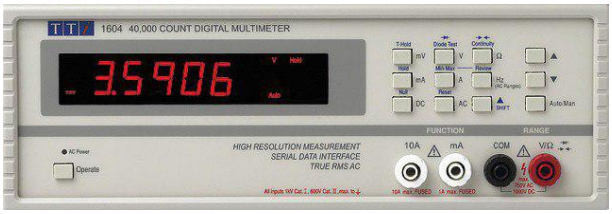


Figura 5. Multímetro.

Para el estudio de sistemas de alimentación (baterías, etc.) se utilizan otros instrumentos, como la carga electrónica.

Carga electrónica

Sirve para emular la carga que soporta un generador eléctrico.

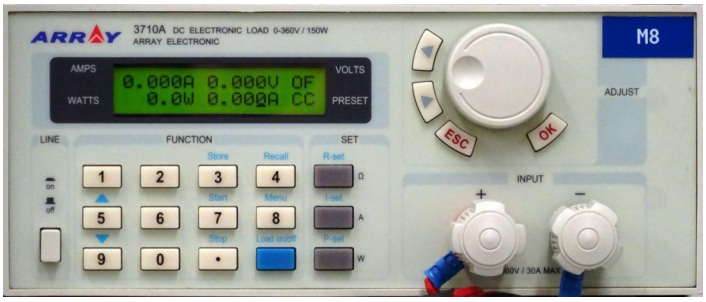


Figura 6. Carga electrónica.

2.2 Herramientas informáticas.

Cada estación de trabajo cuenta con un ordenador que dispone de una gran variedad de programas de ingeniería. En el contexto de este proyecto se hace uso de los siguientes programas:

Explorador de internet. Es la herramienta que los alumnos utilizan para descargar e instalar el programas. Los ordenadores constan de 3 exploradores donde elegir:

- Google Chrome.
- Mozilla Firefox.
- Microsoft Internet explorer.

Archivador de ficheros. Mediante esta herramienta se descomprimen el fichero de distribución del programa. El programa se distribuye en formato ZIP, los programas que lo soportan y están instalados son:

- 7z (https://www.7-zip.org/).
- Windows Explorer.

Aplicación de hojas de calculo. Se utiliza para procesar los datos obtenidos en el laboratorio con formulas y gráficos. El programa *HARKD* puede generar los resultados en formato *.xlsx*.

- Microsoft Excel.
- LibreOffice.

3 OBJETIVO.

Se requiere un programa que pueda ser utilizado para realizar medidas en el laboratorio de manera automática. Este programa debe ser fácilmente extensible, se necesita añadir soporte para muchos dispositivos. Si en el laboratorio se reemplaza un instrumento por otro, añadir soporte para este debe ser fácil.

Además de todo eso, los alumnos deben de poder diseñar sus pruebas con pocos conocimientos de programación, en el lenguaje de programación que quieran.

Para demostrar esta funcionalidad se requiere de un programa gráfico que se pueda utilizar en la asignatura de sistemas de alimentación para medir la curva característica de las fuentes de alimentación.

- Soportar múltiples instrumentos mediante un sistema modular de drivers.
- Un lenguaje batch para escribir pruebas personalizadas.
- Un programa gráfico para medir la curva característica de los sistemas de alimentación.
- Soporte para el formato de hojas de calculo *.xlsx*.
- Programa portable.
- Código simple y bien estructurado que facilite añadir funcionalidad al programa.

4 ALCANCE.

El alcance de este proyecto es el diseño, desarrollo, simulación e implementación de una arquitectura de software en los ordenadores del laboratorio. Principalmente está diseñado para su uso en la *Universidad del país vasco*.

4.1 FASES DE DESARROLLO DEL PROYECTO.

El desarrollo del proyecto se divide en 3 fases.

1. **Desarrollo de HARKDC.**

   Este es el componente que se comunica con los instrumentos del laboratorio. El cual se implementa en varias subfases.

   a) Programar el «core» del programa encargado del sistema de modularidad.

   b) Añadir soporte para la carga electrónica.

   c) Añadir soporte para la fuente de alimentación.

   d) Diseñar el lenguaje «batch».

   e) Crear una «prueba» para la medida de la curva característica de un conversor.

   f) Añadir soporte para la salida en formato Excel.

2. **Desarrollo de HARKDW.**

   Este es el programa gráfico que se encarga de hacer de interfaz entre el usuario y HARKD.

3. **Experimentación en el laboratorio.**

4.2 INSTRUMENTACIÓN A SOPORTAR POR HARKD.

4.2.1 MyWave MPD-3305D.

Fuente de alimentación de CC, multicanal programable. Contiene dos salidas programables. En estas se puede establecer la tensión de salida y medir la corriente.

Figura 7. MyWave MPD-3305D

Características importantes:

- 3 salidas independientes: 30V/3A (5A)x2, 2,5V/3,3V/5V/3Ax1 (fijo).
- Resolución mínima: 100mV, 10mA.
- Interfaz USB, controlador serie, protocolo basado en texto.

4.2.2 Array 3710A

Carga electrónica de CC programable.

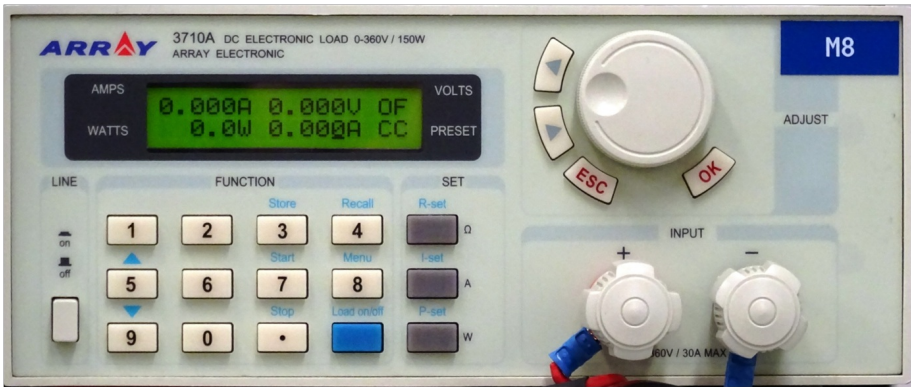


Figura 8. Array 3710A

Características importantes:

- Pagina web: http://www.array.sh/yq-3700e.htm
- 1 entrada programable: 0-30A
- Interfaz RS232, con conversor RS232-USB. Protocolo binario.

4.3 PRUEBAS A SOPORTAR POR HARKD.

4.3.1 Medida de la curva característica de un conversor DC/DC.

Esta prueba consiste en medir la curva caracteristica de un conversor DC/DC lo que implica medir la tensión de salida del conversor con distintas cargas.

Figura 9. Esquema de la prueba para medir la curva caracteristica.

Para realizar esta medida es necesario una fuente de alimentación y una carga electrónica, ambos programables. Los pasos a seguir para realizar la prueba se han especificado en el siguiente diagrama de flujo:

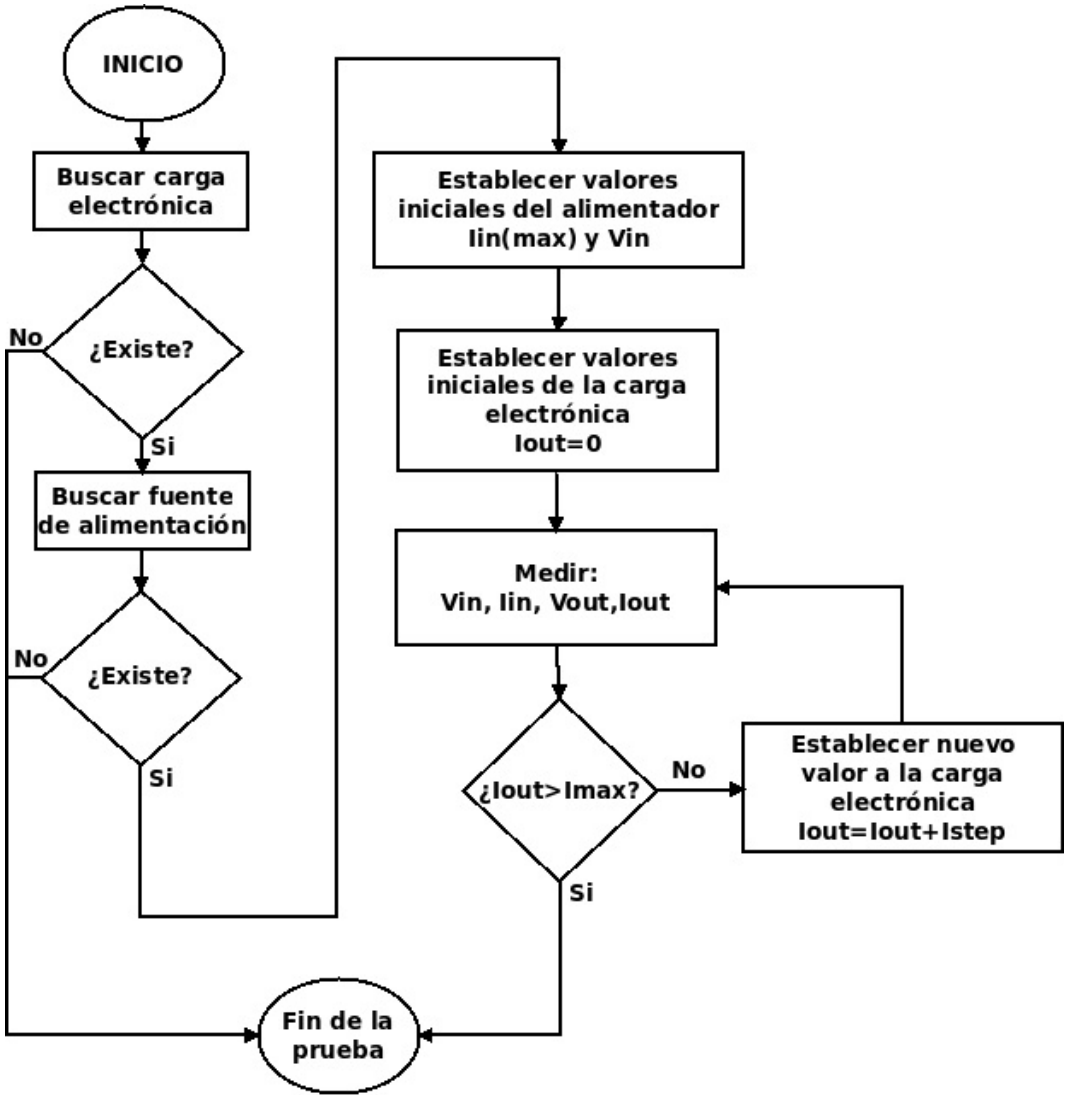


Figura 10. Diagrama de flujo de la prueba de trazado de curva caracteristica de un conversor DC/DC.

4.4 INTEGRACIÓN CON MICROSOFT EXCEL.

El programa guarda los resultados en un fichero excel para que el usuario pueda ver y analizar los resultados obtenidos por el software *HARKD*.

5 BENEFICIOS DE PROYECTO

El beneficio principal de este proyecto es mejorar la experiencia de los alumnos en el laboratorio. Se automatiza la ardua tarea de coger las medidas y rellenar tablas. Además los alumnos podrán preparar sus medidas de antemano y ejecutar las pruebas de forma reiterativa.

Otro gran beneficio es que los alumnos podrán ver el funcionamiento de sus diseños en una hoja excel, donde podrán sacar todos los datos específicos de sus proyectos. De esta forma también facilitara la labor del profesor a la hora de puntuar y ver el correcto funcionamiento de esos dispositivos.

6 ANÁLISIS DE ALTERNATIVAS

6.1 LENGUAJE DE PROGRAMACIÓN.

En el desarrollo de una plataforma software es necesario tomar varias decisiones. La mas importante es la selección del lenguaje de programación que se va a utilizar. Las alternativas son las siguientes.

6.1.1 Java.

Es un lenguaje de programación multipropósito concurrente y orientado a objetos. Esta diseñado específicamente para que tenga la menor cantidad de dependencias de implementación que sea posible.

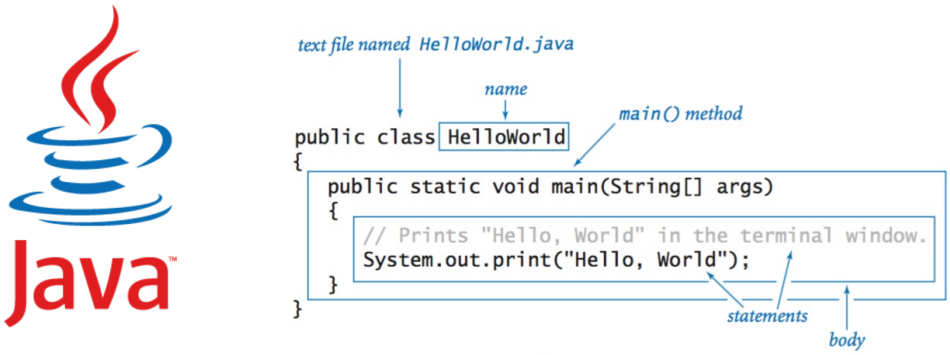


Figura 11. Java.

Esta pensado para que los desarrolladores «escriban una vez para correr en cualquier parte». Esto se consigue compilando el código a un código intermedio «bitcode» la cual la pueda interpretar o compilar la maquina virtual java «JVM».

Fue publicado por primera vez por *Sun Microsystems* como *Java 1.0* en 1996, hace ya mas de 20 años. Es un lenguaje por tanto de la época, con un estilo muy influenciado por *C/C++*. Se popularizo mucho gracias a su capacidad de integrarse en paginas web como applets.

En los últimos años (2019) Java a adquirido gran popularidad gracias a su adopción por parte de *Google* para desarrollo de aplicaciones móviles en Android.

Ventajas:

1. Puede correr en cualquier maquina que contenga JVM. JVM se ha diseñado para que sea fácilmente portable a muchísimas plataformas.
2. Un programa se puede empaquetar en un simple .JAR. No hay necesidad de instalación.
3. Gran cantidad de librerías disponibles en internet. Librerías disponibles para manejar puertos serie y archivos Excel.

4. Lenguaje muy conocido y fácil de utilizar por programadores inexpertos.
5. Gran cantidad de IDEs de desarrollo.
6. Librerías gráficas integradas en el JVM.

Desventajas:

1. Necesita de JVM para funcionar. Si bien este se distribuye por defecto en Linux y Android no es tal para Microsoft windows.
2. Muchas librerías de Java necesitan de librerías externas que se tienen que distribuir en ficheros distintos.
3. Java no esta pensado para desarrollar controladores de maquinas, no para trabajar a bajo nivel.

6.1.2 Lenguajes de scripting: Python, Perl y Tcl.

Son lenguajes de programación multipropósito, de alto nivel y dinámicos. Son fáciles de usar y la mayoría de la gente con formación técnica conoce por lo menos uno de ellos.

Figura 12. Lenguajes de scripting.

El primero en aparecer fue ***Perl*** en 1987. En un principio de diseño como lenguaje de scripting para UNIX para análisis de textos. El lenguaje se expandió rápidamente en los años ochenta y noventa hasta convertirse en en lenguaje de programación mas extendido en el área de servidores.

En el año 2000 *Perl* era el lenguaje mas utilizado para desarrollo de paginas web dinámicas, programación de redes, finanzas y bioinformática. Es un lenguaje muy flexible y potente.

En las ultimas dos décadas *Perl* ha sido gradualmente substituido por un lenguaje competidor, ***Python*** el cual tiene una sintaxis mas limpia y esta orientada a objetos. A diferencia de *Perl* hay varias implementaciones de Python aunque la mas popular es CPython. Es un lenguaje de programación moderno.

Finalmente esta ***Tcl***, el cual se desarrollo en la *Universidad de California, Berkeley* en 1988 por John Ousterhout. Este lenguaje de programación se diseño con la simplicidad en mente, y el interprete es muy pequeño. Consta de una libreria grafica integrada ***Tk***.

Se utiliza mucho en la industria como lenguaje de extensión, al ser fácilmente extensible en C. Sigue la misma filosofía que Java en cuanto a portabilidad se refiere.

Ventajas:

1. Pueden correr en cualquier maquina que contenga un interprete del lenguaje en cuestión. Los tres vienen preinstalados en Linux y Mac OS X pero no en Microsoft Windows.
2. Lenguajes dinámicos fáciles de utilizar por cualquier persona con un mínimo de formación técnica.
3. Traen librerías las gráficas integradas.

Desventajas:

1. Hay que instalar el interprete en Microsoft Windows. Ya que Windows no suele traerlos instalados por defecto.
2. Se tiene que distribuir el código a los clientes, puesto que el interprete necesita acceso al código fuente para ejecutar el programa.

6.1.3 C.

C es un lenguaje the propósito general imperativo que permite programación estructurada. Originalmente se desarrollo para utilizarlo en la reimplementación del sistema operativo UNIX.

```c
#include "../harkd.h"
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
harkd_r harkd_mpd_init(harkd_dev_obj_t *harkd,const char *port,const char *args[]) {
     char buffer[512];
     if(harkd_serial_open(harkd,port)==NULL)                goto io_error;
     if(harkd_serial_puts(harkd,"\n*idn?\n")!=HARKD_OK)     goto io_error;
     if(harkd_serial_gets(harkd,buffer,sizeof(buffer))==NULL) goto another_device;
     if(harkd_serial_puts(harkd,"OUT1\n"   )!=HARKD_OK)     goto io_error;
     if(harkd_serial_puts(harkd,"TRACK0\n" )!=HARKD_OK)     goto io_error;
     if(strcasecmp("SN:V1.81",buffer))                      goto another_device;
     return HARKD_OK;
io_error:
     return HARKD_ERR;
another_device:
     return HARKD_ERR;
}
```

Figura 13. Lenguaje de programación C.

El lenguaje C se diseño para que un compilador relativamente sencillo pudiera generar código maquina que se pudiera ejecutar en una maquina con o sin sistema operativo.

Hoy en día es el lenguaje de programación mas extendido, la mayoría de sistemas operativos; Linux,Windows, Mac OS, BSD, ... están implementados en este lenguaje. La mayoría de drivers, software de sistema y librerías están escritos en C. La maquina virtual Java y casi todos los interpretes están escritos en C.

En 1989 el *Instituto Nacional Estadounidense de estándares (ANSI)* estandarizo el lenguaje, el ultimo estándar se publico en 2018.

Ventajas:

1. Lenguaje de programación portable y potente. Permite el desarrollo de librerías.

2. Lenguaje estandarizado, gran variedad de compiladores compatibles.
3. Permite generar programas portables en un único ejecutable de forma nativa y natural.
4. Portable a casi todos los sistemas operativos y arquitecturas disponibles.
5. Acceso a rutinas del sistema operativo de forma directa.
6. Gran cantidad de librerías en internet.
7. Eficiente.

Desventajas:

1. Lenguaje complejo de usar. Se requiere de conocimientos avanzados de programación para poder programar en el.
2. No dispone de librerías gráficas portables que puedan utilizarse en todos los sistemas operativos.

7 SELECCIÓN DE LA SOLUCIÓN

Las distintas alternativas se ponderan en un cuadro comparativo para poder elegir la solución que mas se acerca a las necesidades.

7.1 PONDERACIÓN DE LAS CARACTERISTICAS FUNDAMENTALES

Para poder elegir el lenguaje de programación mas adecuado se van a considerar las siguientes caracteristicas fundamentales;

Caracteristica	**Ponderación %**
Portabilidad	10
Lenguaje estándar	20
Facilidad de instalación	40
Facilidad de programación	30

Tabla 1. Ponderación de las caracteristicas requieridas en el proyecto.

Alternativas:		Java	Python/Perl/Tcl	C
Portabilidad	10	3	3	3
Lenguaje estándar	20	2	1	3
Facilidad de instalación	40	1	1	3
Facilidad de programación	30	2	3	1
Total:		170	180	240

Tabla 2. Puntuación de las alternativas.

Como se puede observar en la tabla, la mejor alternativa es programar HARKD en el lenguaje C. Hay que resaltar todos los lenguajes de programación pueden interactuar con otro escrito en C, así pues la interfaz grafica sera en Tcl.

8 PLAN DE TRABAJO

8.1 ENFOQUE BÁSICO.

El trabajo se dividirá en **paquetes de trabajo** los cuales se dividirán a su vez en **tareas** a realizar. La duración de un paquete de trabajo es la suma del tiempo que las tareas requieren.

Como mecanismo de control se organizaran reuniones con el profesor mensualmente. En estas reuniones se hablara de como desarrollar el proyecto a lo largo de toda su ejecución.

Además de las reuniones se utilizaran medios informáticos como **Telegram** y **correo electrónico** para informar de avances y aclarar dudas que vayan surgiendo en el desarrollo del proyecto.

8.2 DESCRIPCIÓN DE PAQUETES.

A continuación se describen los **paquetes de trabajo principales** de los que se compone el proyecto. Mas tarde en la lista de tareas se describirán las horas necesarias y las tareas vinculadas a cada paquete.

P.1 - Software HARKD. Es la parte de programación del proyecto. El software *HARKD* esta dividido en dos programas, el segundo depende del primero.

- **P.1.3 - Programa HARKDC.** Se buscaran las librerías necesarias y se escribirá el programa.
  - **P.1.3.2 - Colección de drivers.** Se escribe el código que controlara los dispositivos.
  - **P.1.3.3 - Colección de pruebas.** Se escribe el código que concierne a las medidas.
- **P.1.4 - Programa HARKDW.** Se escribirá la interfaz grafica.

P.2 - Depuración en el laboratorio. Se comprobara el correcto funcionamiento del software en el laboratorio.

- **P.2.3 - Escritura del manual.** En el laboratorio se escribirá el manual que se distribuirá con el software.

P.3 - Empaquetado y distribución. Se empaquetara el software con el manual para distribuirlo en los laboratorios.

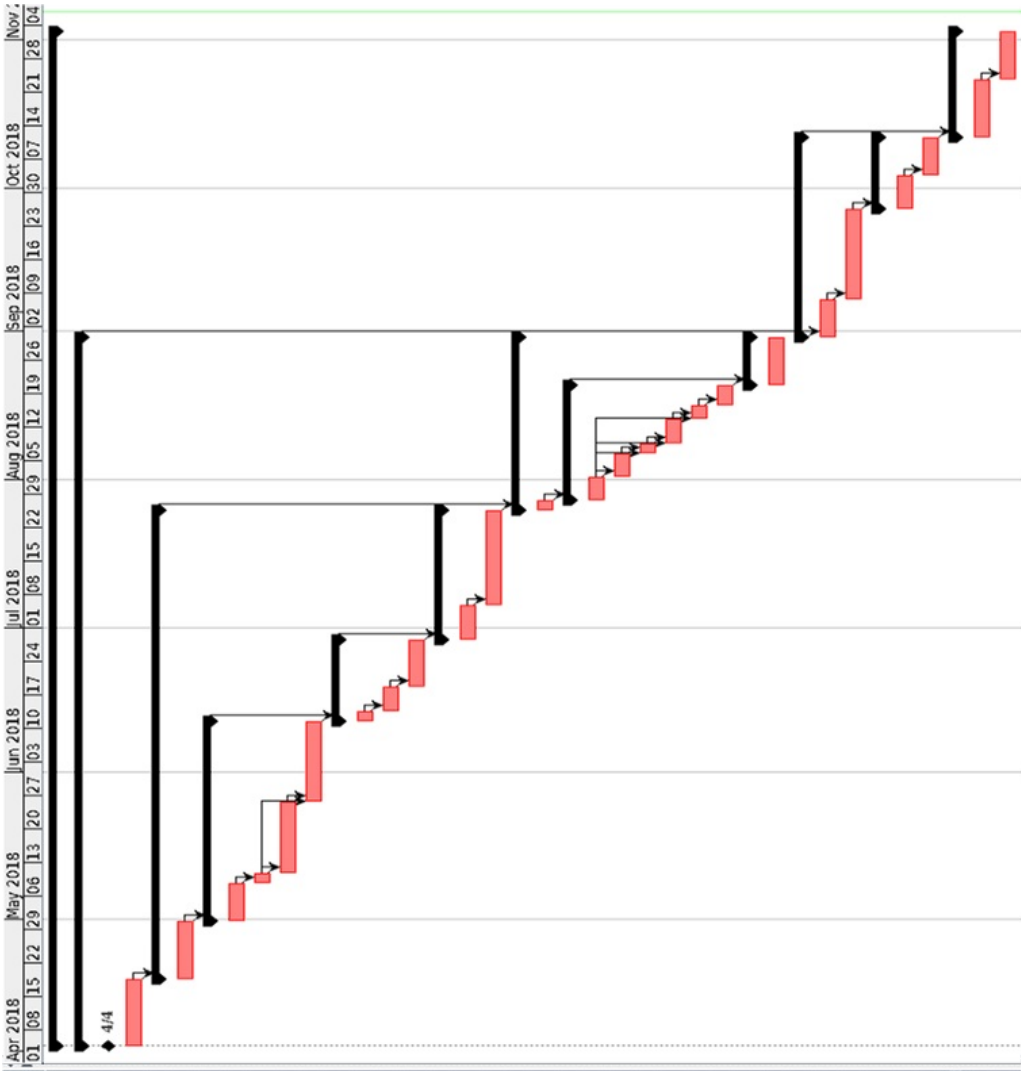
8.3 LISTA DE TAREAS.

	EDT	Name	Duration	.	Work
1	**0**	**⊟Total**	**228 days**		**456 hours**
2	**P.1**	**⊟Software HARKD**	**160 days**		**320 hours**
3	T.1.1	Comienzo del proyecto	0 days		0 hours
4	T.1.2	Diseño de la plataforma	16 days		32 hours
5	**P.1.3**	**⊟Programa HARKDC**	**104 days**		**208 hours**
6	T.1.3.1	Interfaz de comandos	12 days		24 hours
7	**P.1.3.2**	**⊟Colección de drivers**	**44 days**		**88 hours**
8	T.1.3.2.1	Manejador generico de drivers	8 days		16 hours
9	T.1.3.2.2	Driver de ejemplo	4 days		8 hours
10	T.1.3.2.3	Driver MPD-3305D	16 days		32 hours
11	T.1.3.2.4	Driver ARRAY-3710A	16 days		32 hours
12	**P.1.3.3**	**⊟Colección de pruebas**	**20 days**		**40 hours**
13	T.1.3.3.1	Manejador generico de pruebas	4 days		8 hours
14	T.1.3.3.2	Prueba de ejemplo	4 days		8 hours
15	T.1.3.3.3	Prueba DC/DC	12 days		24 hours
16	**P.1.3.4**	**⊟Soporte de salida en tabla**	**28 days**		**56 hours**
17	T.1.3.4.1	Formato CSV (Texto)	8 days		16 hours
18	T.1.3.4.2	Formato XLSX (Excel)	20 days		40 hours
19	**P.1.4**	**⊟Programa HARKDW**	**40 days**		**80 hours**
20	T.1.4.1	Comunicación con HARKDC	4 days		8 hours
21	**P.1.4.2**	**⊟Interfaz grafica**	**24 days**		**48 hours**
22	T.1.4.2.1	Boton de comienzo	4 days		8 hours
23	T.1.4.2.2	Tabla de resultados	4 days		8 hours
24	T.1.4.2.3	Boton de stop	4 days		8 hours
25	T.1.4.2.4	Botón de guardar en	4 days		8 hours
26	T.1.4.2.5	Botón de abrir en Excel.	4 days		8 hours
27	T.1.4.2.6	Imagen explicativa	4 days		8 hours
28	**P.1.4.3**	**⊟Menus para pruebas**	**12 days**		**24 hours**
29	T.1.4.3.1	Prueba DC/DC	12 days		24 hours
30	**P.2**	**⊟Depuración en laboratorio**	**44 days**		**88 hours**
31	T.2.1	Instalación.	8 days		16 hours
32	T.2.2	Corrección de errores.	20 days		40 hours
33	**P.2.3**	**⊟Escritura del manual**	**16 days**		**32 hours**
34	T.2.3.1	Manual HARKDC	8 days		16 hours
35	T.2.3.1	Manual HARKDW	8 days		16 hours
36	**P.3**	**⊟Empaquetado y distribución**	**24 days**		**48 hours**
37	T.3.1	Compilar versión definitiva	12 days		24 hours

HARKD - page1

Figura 14. Lista de tareas.

9 DIAGRAMA DE GANTT.



	EDT	Name	Duration	Start
1	0	⊟Total	228 days	**4/4/18 5:30 PM**
2	P.1	⊟Software HARKD	160 days	**4/4/18 5:30 PM**
3	T.1.1	Comienzo del proyecto	0 days	4/4/18 5:30 PM
4	T.1.2	Diseño de la plataforma	16 days	4/4/18 5:30 PM
5	P.1.3	⊟Programa HARKDC	104 days	**4/18/18 7:30 PM**
6	T.1.3.1	Interfaz de comandos	12 days	4/18/18 7:30 PM
7	P.1.3.2	⊟Colección de drivers	44 days	**4/30/18 7:30 PM**
8	T.1.3.2.1	Manejador generico de drivers	8 days	4/30/18 7:30 PM
9	T.1.3.2.2	Driver de ejemplo	4 days	5/8/18 5:30 PM
10	T.1.3.2.3	Driver MPD-3305D	16 days	5/10/18 7:30 PM
11	T.1.3.2.4	Driver ARRAY-3710A	16 days	5/25/18 6:30 PM
12	P.1.3.3	⊟Colección de pruebas	20 days	**6/11/18 5:30 PM**
13	T.1.3.3.1	Manejador generico de pruebas	4 days	6/11/18 5:30 PM
14	T.1.3.3.2	Prueba de ejemplo	4 days	6/13/18 7:30 PM
15	T.1.3.3.3	Prueba DC/DC	12 days	6/18/18 6:30 PM
16	P.1.3.4	⊟Soporte de salida en tabla	28 days	**6/28/18 6:30 PM**
17	T.1.3.4.1	Formato CSV (Texto)	8 days	6/28/18 6:30 PM
18	T.1.3.4.2	Formato XLSX (Excel)	20 days	7/5/18 7:30 PM
19	P.1.4	⊟Programa HARKDW	40 days	**7/25/18 5:30 PM**
20	T.1.4.1	Comunicación con HARKDC	4 days	7/25/18 5:30 PM
21	P.1.4.2	⊟Interfaz grafica	24 days	**7/27/18 7:30 PM**
22	T.1.4.2.1	Boton de comienzo	4 days	7/27/18 7:30 PM
23	T.1.4.2.2	Tabla de resultados	4 days	8/1/18 6:30 PM
24	T.1.4.2.3	Boton de stop	4 days	8/6/18 5:30 PM
25	T.1.4.2.4	Botón de guardar en	4 days	8/8/18 7:30 PM
26	T.1.4.2.5	Botón de abrir en Excel.	4 days	8/13/18 6:30 PM
27	T.1.4.2.6	Imagen explicativa	4 days	8/16/18 5:30 PM
28	P.1.4.3	⊟Menus para pruebas	12 days	**8/20/18 7:30 PM**
29	T.1.4.3.1	Prueba DC/DC	12 days	8/20/18 7:30 PM
30	P.2	⊟Depuración en laboratorio	44 days	**8/30/18 7:30 PM**
31	T.2.1	Instalación.	8 days	8/30/18 7:30 PM
32	T.2.2	Corrección de errores.	20 days	9/7/18 5:30 PM
33	P.2.3	⊟Escritura del manual	16 days	**9/26/18 6:30 PM**
34	T.2.3.1	Manual HARKDC	8 days	9/26/18 6:30 PM
35	T.2.3.2	Manual HARKDW	8 days	10/3/18 7:30 PM
36	P.3	⊟Empaquetado y distribución	24 days	**10/11/18 5:30 PM**
37	T.3.1	Compilar versión definitiva	12 days	10/11/18 5:30 PM
38	T.3.2	Instalar y verificar	12 days	10/23/18 5:30 PM

Figura 15. Diagrama de GANTT.

10 METODOLOGÍA

La primera acción a la hora de diseñar un programa de ordenador es dibujar un diagrama de bloques donde se vea la interacción entre los distintos objetos que la componen. A medida que se escribe el programa el diseño puede variar, pero ayuda a la hora de tener las ideas claras.

10.1 DIAGRAMA DE BLOQUES DE LA PLATAFORMA HARKD.

Este es un diagrama de bloques de llamadas, donde los distintos elementos solo se comunican con los elementos superior e inferior. Por ejemplo, el modulo «prueba DC/DC» si quiere obtener un dato de la fuente de alimentación se comunica con este a trabes de «src/harkd-core.c».

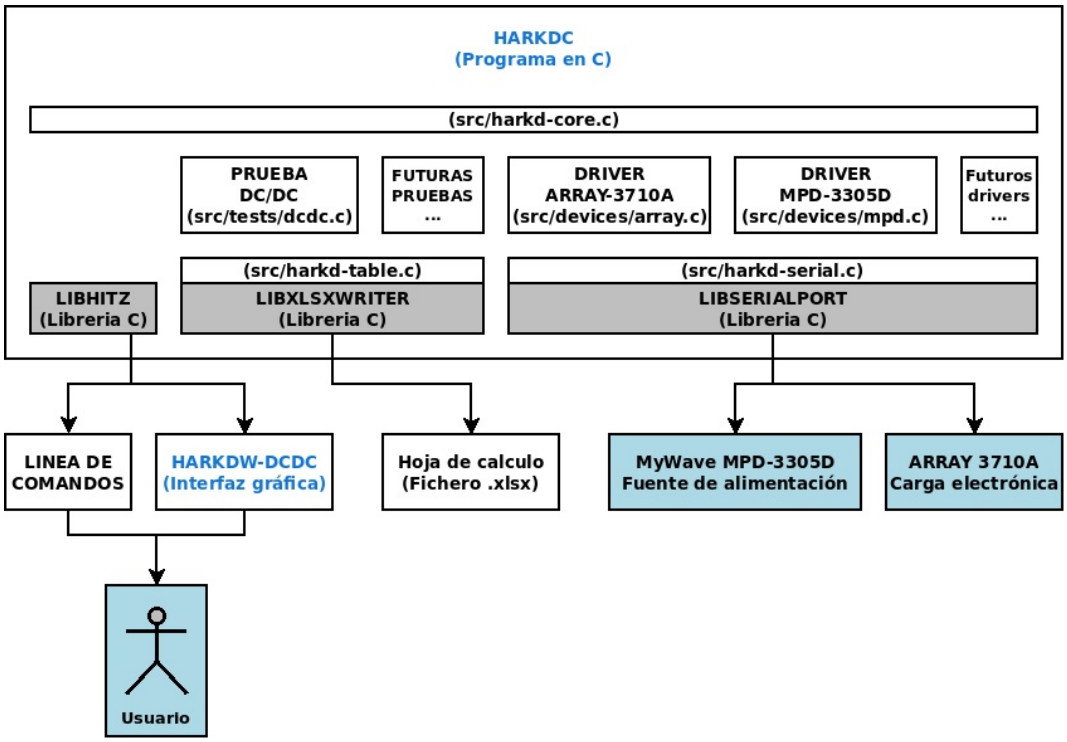


Figura 16. Diagrama de bloques del programa.

10.2 EXPLICACIÓN DE LOS COMPONENTES.

A continuación se explican los distintos objetos que componen el programa *HARKD*.

10.2.1 HARKDC

Este es el programa central que hace la mayoría de las tareas necesarias para la realización las medidas del laboratorio (backend). Es un programa de terminal además de libreria, tiene interfaz de comandos interactiva.

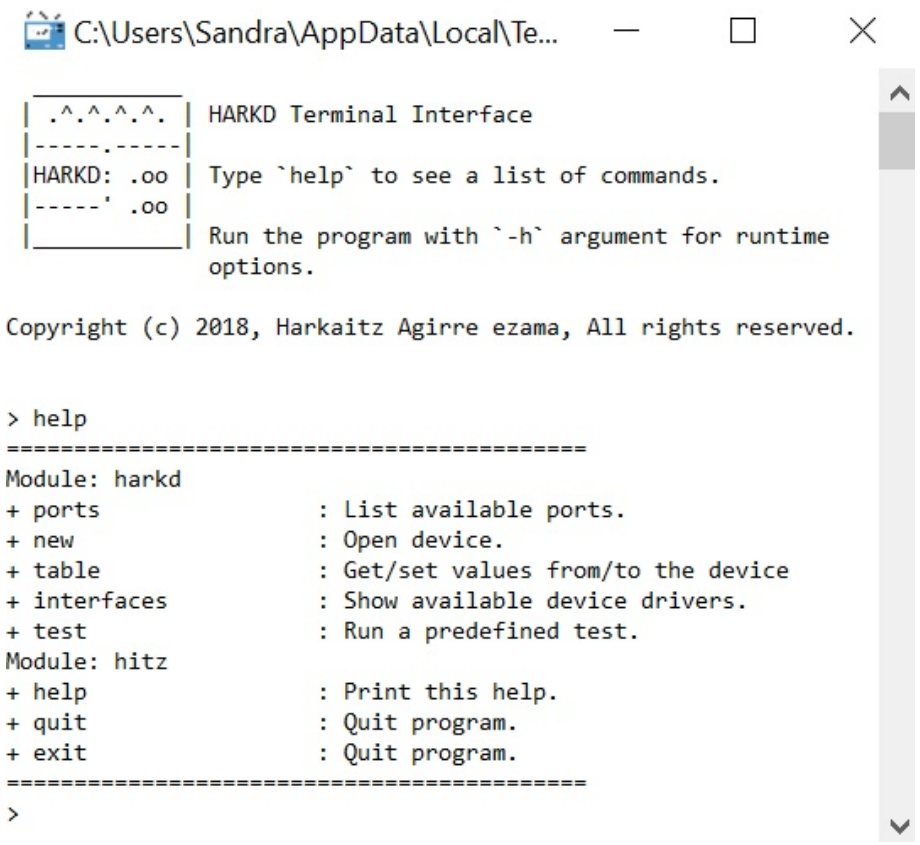


Figura 17. HARKDC.

El estilo de programación utilizado es el modular. Un «modulo» se define rellenando una estructura con punteros de función «métodos» y datos «parámetros». Hoy en día este es el estilo de programación mas popular en C, ejemplos son; «Kamailio SIP», «Kernel Linux», «JVM», ...

En el caso de *HARKDC* se consta de dos conjuntos de módulos; los **controladores de dispositivo** y las **pruebas**. Los primeros se encargan de la comunicación con el dispositivo, los segundos implementan los procedimientos de laboratorio para coger medidas.

Esta separación entre pruebas y controladores permite que, por ejemplo, quien esta implementando un controlador no se tenga que preocupar de las pruebas que con ella se van a realizar. Y viceversa, quien esta escribiendo una prueba no necesita saber del protocolo de comunicación del dispositivo.

Lista de dispositivos:

Los drivers de los dispositivos están escritos en la carpeta «src/devices». Cada driver consta de un archivo C donde se definen sus «métodos». A continuación se explica el procedimiento a seguir para implementar un controlador de dispositivo.

1. Renombra el fichero *src/devices/example.c*, este controlador no soporta ningún dispositivo, solo es una plantilla.
2. Reemplaza la palabra *example* a ser posible por el nombre del modelo del dispositivo a soportar, por ejemplo *mi_driver*.
3. Modifica el código dentro de los métodos definidos. Por ejemplo, en el caso de asignar valor es necesario modificar el método *harkd_mi_driver_set*.
4. Finalmente especifica los nombres de salidas y entradas en la estructura *HARKD_DEVICE_MI_DRIVER*.
5. Especifica en esa misma estructura las opciones soportadas por el dispositivo.

Una vez creado el driver, recompila y abre el programa, veras como el comando *interfaces* lista el driver creado. Se pueden utilizar las funciones definidas en «src/harkd-serial.c» para controlar el puerto serie.

Lista de pruebas:

Las pruebas están escritas en la carpeta «src/tests». Igual que los controladores de dispositivo cada prueba consta de un archivo C donde se definen sus «métodos». Los pasos a seguir para añadir una prueba son las siguientes:

1. Se renombra el fichero *src/tests/example.c*, esta prueba no hace nada útil, solo es una plantilla.
2. Se reemplaza la palabra *example* por otro nombre.
3. Define la prueba dentro de la función *harkd_mi_prueba_test*.
4. Especifica las opciones que la prueba soporta en la estructura *HARKD_MI_PRUEBA_TEST*.

Soporte para multiples salidas:

En el fichero «harkd-table.c» se han definido varias funciones para que las pruebas puedan guardar sus resultados en los formatos *CSV (Texto)* y *XLSX (Excel)*.

10.2.2 HARKDW

Este es el programa gráfico que interactúa con el usuario final (frontend). Por ahora solo soporta una prueba; Medir las curvas caracteristicas de fuentes de alimentación.

Este programa recibe los valores del usuario y los pasa a *HARKDC*. Es el backend quien se encarga de ejecutar todas las acciones.

HARKDW se ha implementado usando el lenguaje de programación *Tcl/Tk*, y se le han añadido las siguientes funcionalidades;

1. Botones de «Comienzo», «Stop» y «Restart». Es importante que la prueba se pueda parar a toque de botón por si algo falla.

Figura 18. Comienzo, stop y restart.

2. Botón de «Guardar en». El usuario puede guardar el resultado en un fichero si lo desea.

Figura 19. Boton «Guardar en»

3. Botón de «Abrir resultados». El usuario puede abrir el resultado en el programa de hojas de calculo por defecto.

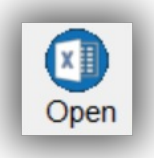


Figura 20. Boton «Abrir resultados»

4. Botón de «Ayuda». El usuario puede leer la documentación en linea.

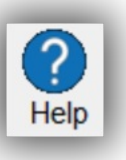


Figura 21. Boton de «ayuda»

5. Logotipo de la universidad. Al pulsar se abre la página web oficial de la universidad.

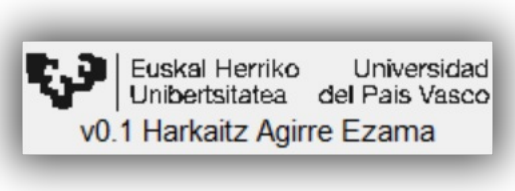


Figura 22. Logotipo UPV/EHU.

6. Icono del programa. Al pulsarlo se abre la pagina web de descargas oficial de *HARKD*. El diseño de los iconos los ha realizado la arquitecta *Sandra Elena Endolz Nava*, «sandraendolznava@gmail.com»

Figura 23. Logotipos UPV/EHU y icono del programa.

10.3 LIBRERÍAS.

Una libreria en C es un conjunto de código ya compilado al que pueden llamar programas externos. El único caso en el que un programa escrito en C opera con el «hardware» directamente se da en microcontroladores y DSPs, en la mayoría de sistemas este se comunica con el «hardware» a trabes de las librerías del sistema.

Desgraciadamente las librerías de sistema cambian dependiendo del sistema operativo utilizado, y si bien existe un estándar (POSIX), el sistema operativo de sobremesa mas popular (Microsoft Windows) no lo trae instalado por defecto.

Es por eso que en vez de utilizar las librerías de sistema de un sistema operativo en concreto se ha decidido utilizar una libreria intermedia *libserialport*. Para la salida en formato Excel se ha utilizado la libreria *libxlsxwriter*.

10.3.1 LIBSERIALPORT.

Libserialport (A veces abreviado como «sp») es una libreria compartida multiplataforma escrita en C que se hace cargo de los detalles específicos del sistema operativo a la hora de escribir programas que hagan uso del puerto serie.

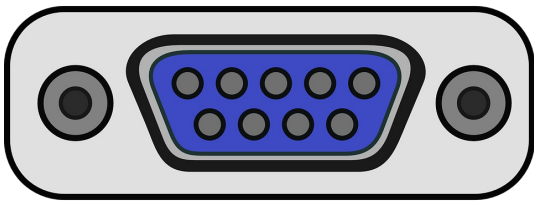


Figura 24. Conector puerto serie.

Sistemas operativos soportados:

1. Linux.
2. Mac OS X.
3. FreeBSD.
4. Windows.
5. Android.

Documentacion del proyecto:

Documentación de la API. http://sigrok.org/api/libserialport/unstable/index.html

Pagina web oficial y descarga. https://sigrok.org/wiki/Libserialport

10.3.2 LIBXLSXWRITER.

Es una libreria C para creación de archivos XLSX Excel. Es una libreria muy completa y bien documentada.

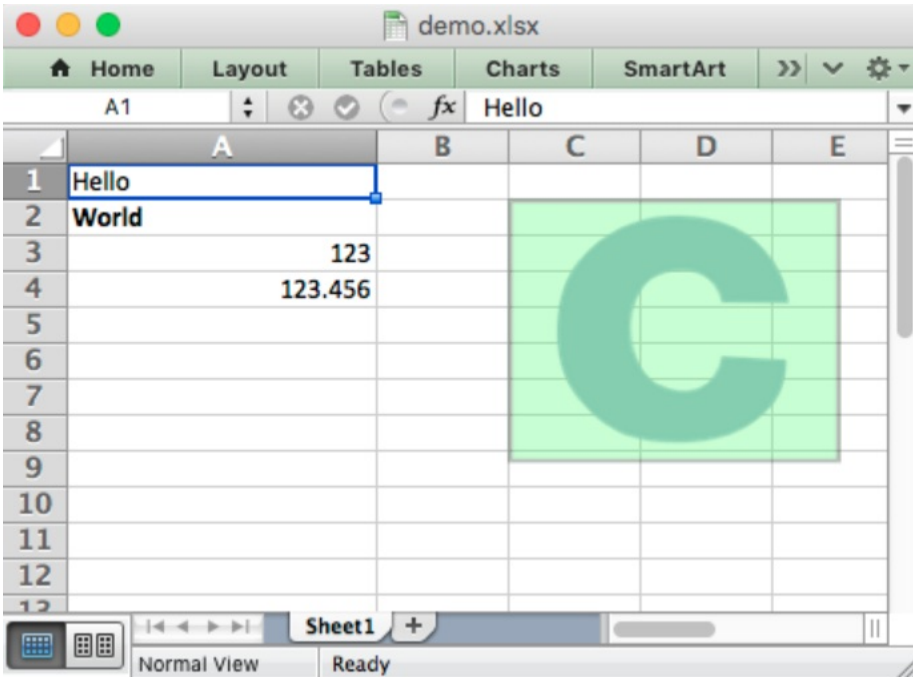


Figura 25. Libxlsxwriter

Funcionalidades:

- 100% compatible con los ficheros Excel XLSX.
- Formateado completo de excel.
- Se pueden juntar celdas, definir nombres, etc.
- Gráficos.
- Formulas.
- Imágenes.
- Optimización de memoria con archivos grandes.
- Portable.
- Su única dependencia es zlib.

Documentación de la libreria:

Documentación de la API. http://libxlsxwriter.github.io/

Pagina web oficial y descarga. https://github.com/jmcnamara/libxlsxwriter

10.4 MEDIOS.

10.4.1 GNU/Emacs.

Editor de textos libre, extensible, personalizable y libre. Es muy popular entre programadores y usuarios técnicos. Soporta coloreado y indentación automática para gran cantidad de lenguajes de programación, además de corrector al vuelo «flycheck» y analizador semántico «semantic».

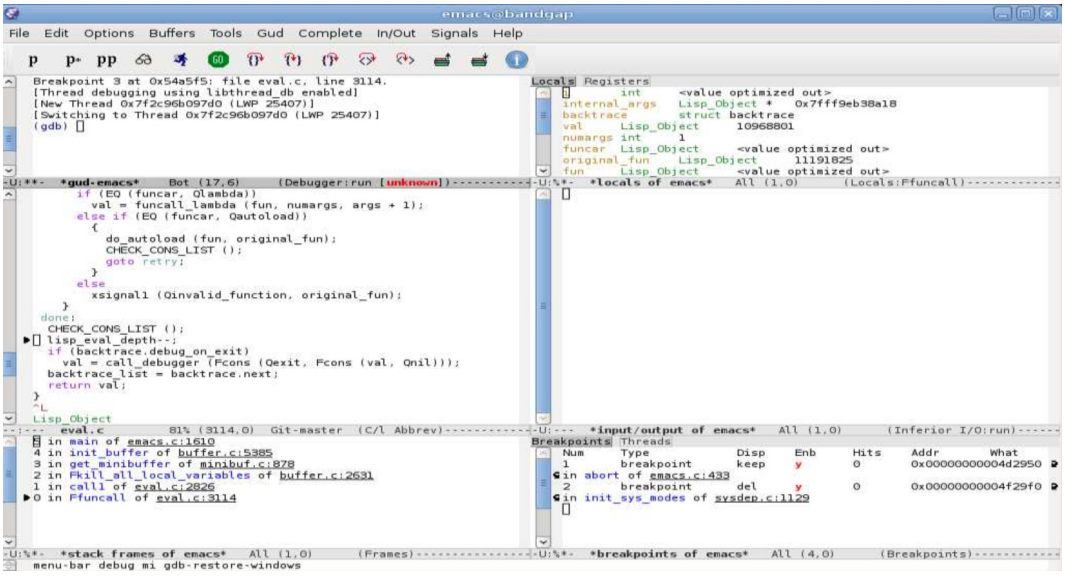


Figura 26. GNU/Emacs

Caracteristicas principales:

- Es software libre.
- Portable, funciona en la mayoría de sistemas operativos.
- Extensible, es un editor de textos programable.
- Soporte para una decena de lenguajes de programación.
- Comprueba los errores de sintaxis al momento «Flycheck».
- Analizador semántico «Semantic»
- Repositorio de módulos integrado.
- Editor de texto mas utilizado en programación en C.

Documentacion del programa:

Pagina web oficial. https://www.gnu.org/software/emacs/

Versión para Microsoft Windows. http://ergoemacs.org/emacs/which_emacs.html

Documentación. https://www.gnu.org/software/emacs/manual/

10.4.2 FreeWrap.

Este programa convierte scripts TCL/TK en un único fichero ejecutable. Para ello FreeWrap empaqueta todo el interprete TCL, sus librerías y el script en un ejecutable.

La aplicación *HARKDW* se ha empaquetado utilizando este programa.

Pagina web oficial. http://freewrap.sourceforge.net/

Versión para Microsoft Windows.
https://sourceforge.net/projects/freewrap/files/freewrap/freeWrap%206.64/

10.4.3 GCC (GNU Compiler Collection).

Es el compilador de C utilizado en el proyecto. Es software libre y soporta el estándar ANSI C99 para el cual se ha escrito el programa. Su nombre deriva de (GNU Compiler Collection) y soporta mas de un lenguaje, una veintena de arquitecturas y casi todos los sistemas operativos.

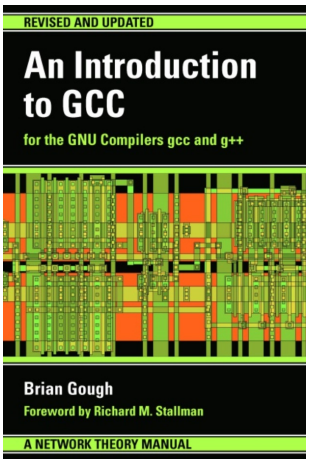


Figura 27. Manual de introducción a GCC.

Una de las caracteristicas mas a resaltar es su capacidad de compilar para otros sistemas distintos a la maquina compiladora (compilación cruzada). Gracias a esta funcionalidad se puede compilar un programa para distintos sistemas operativos al momento.

Pagina web oficial. https://gcc.gnu.org/

Descarga para Microsoft Windows. https://www.msys2.org/ (Incluye CMake)

10.4.4 CMake

Es el entorno de compilación utilizado en el proyecto. Un entorno de compilación se encarga de dar las ordenes necesarias al compilador C para generar los ejecutables o las librerías necesarias.

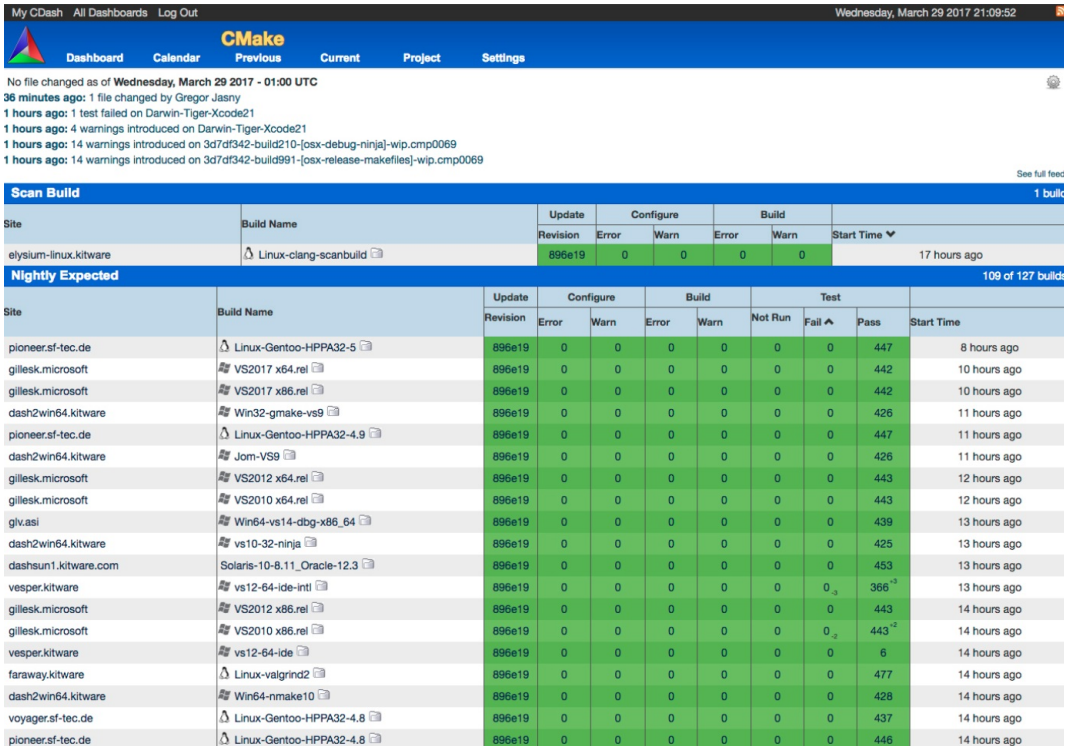


Figura 28. CMake.

Durante mucho tiempo el entorno de compilación mas utilizado ha sido Autotools pero este esta siendo reemplazado poco a poco por esta herramienta. Al día de hoy es el entorno mas soportado por los IDEs.

Pagina web oficial. https://cmake.org/

Documentación. https://cmake.org/documentation/

11 PRESUPUESTO

El presupuesto estimado para la realización del proyecto se ha desglosado en cuatro partes; Horas internas, amortizaciones/inversiones, subcontrataciones, gastos.

11.1 HORAS INTERNAS.

En este apartado se desglosan los costes de recursos humanos. Los costes horarios se han obtenido del BOPV (Boletín Oficial del País Vasco) de Marzo de 2011.

Cargo	Coste horario (€/h)	Horas	Coste (€)
Programador	41,1	456	18.741,6
Director	78,6	40	3.144
Técnicos de laboratorio	31,0	10	310
Subtotal			22.195,6

Tabla 3. Presupuesto, horas internas.

11.2 AMORTIZACIONES/INVERSIONES.

En este otro apartado se establecen los gastos derivados de la amortización de los materiales así como las inversiones realizadas.

Activo	Coste (€)	Vida útil (h)	Coste horario (€/h)	Tiempo de uso (h)	Coste (€)
Ordenador	1200	12000	0,1	406	40,6
Portátil	300	8000	0,0375	50	1,875
Carga Electrónica	400	12000	0,0333	100	3,33
Fuente de alimentación	1000	12000	0,0833	100	8,33
Subtotal					54,14

Tabla 4. Presupuesto, amortizaciones/inversiones.

11.3 SUBCONTRATACIONES.

En este proyecto no se ha subcontratado a nadie.

11.4 GASTOS.

En esta partida se describen los costes de los productos empleados únicamente en este proyecto y que no se reutilizaran.

Concepto	Coste unitario (€)	Multiplicador	Coste (€)
Material de oficina	—-	—	60
Electricidad	0,13€/kWh	230kWh	30
Subtotal			90

Tabla 5. Presupuesto, gastos.

11.5 PRESUPUESTO COMPLETO.

Concepto	Coste (€)
Horas internas	22.195,6
Amortizaciones/inversiones	54,14
Subcontrataciones	0
Gastos	90
Costes indirectos	200
Subtotal	**22.539,74€**

Tabla 6. Presupuesto completo.

12 ANÁLISIS DE RENTABILIDAD

Este proyecto se enmarca en el ámbito de la docencia, y su carácter es exclusivamente altruista. Además puesto que el proyecto se ha diseñado para poder extendido su código fuente debe de ser accesible por el publico en general.

Esto no significa que no se pueda obtener beneficios del mismo, puesto que podría ser útil para los departamentos de I+D+I de muchas empresas. Por tanto se va a comercializar con dos licencias distintas.

La primera es la **licencia académica**, la cual podrá ser utilizada única y exclusivamente en el ámbito educativo, ya bien sea en la universidad o por usuarios individuales, sin coste alguno para el usuario.

La segunda es la **licencia comercial con soporte**, la cual podrá ser utilizada en ámbitos comerciales, y tendrá un coste de 30€/mes.

12.1 Derechos con licencia academica.

1. Se puede acceder al código fuente.
2. No tiene coste alguno.
3. Únicamente puede emplearse para fines didácticos.

12.2 Licencia comercial con soporte.

1. Se puede acceder al código fuente.
2. Tiene un coste de 30 €/mes.
3. Se puede utilizar para fines comerciales.

12.3 Posible rentabilidad.

Para poder rentabilizar el proyecto el proyecto se necesitarían recaudar 22540€. Si un cliente adquiriera la licencia comercial por un año se obtendrían 360€, por tanto se necesitan 63 empresas que compren por un año.

13 ANÁLISIS DE RIESGOS

A continuación se describen los posibles riesgos del proyecto. En el ámbito del software podemos definir los siguientes riesgos.

13.1 El programa binario deja de funcionar.

Después de una actualización del sistema operativo, o del cambio de alguna libreria del sistema el programa puede quedar en un estado en el que el programa no pueda arrancar.

Solución: Recompilar el programa.

13.2 El programa da errores de permisos.

Por algun error del administrador del sistema el programa puede no tener permisos para acceder a las interfaces serie o USB. También puede darse el caso de que los permisos cambien después de una actualización.

Solución: Cambiar los permisos.

13.3 El programa no detecta los dispositivos.

Se ha actualizado algun controlador de dispositivo de Windows y el interfaz USB o serie no es accesible.

Solución: Instalar los drivers necesarios.

13.4 Cambio de algun instrumento por otro no soportado.

El laboratorio a substituido un instrumento por otro no soportado aún por HARKD. Por tanto el programa no detectara el dispositivo.

Solución: Añadir el controlador necesario a HARKD

14 CONCLUSIONES

Este software se utilizara en un principio el los laboratorios de electrónica de *la Escuela de Ingeniería de Bilbao (UPV/EHU)* pero su utilidad va mas allá. Gracias a su diseño modular y extensibilidad puede ser utilizado en cualquier laboratorio para automatizar la toma de medidas.

Todos los requerimientos y objetivos establecidos al comienzo del proyecto se han satisfecho en su totalidad. La plataforma software desarrollada a llegado a un punto de madurez suficiente para instalarlo en producción, y se utilizara en clase el curso 2018/2019.

La mejora en la calidad de las clases sera sustancial ya que el tiempo que se empleara en tomar las medidas en laboratorio se reduce considerablemente.

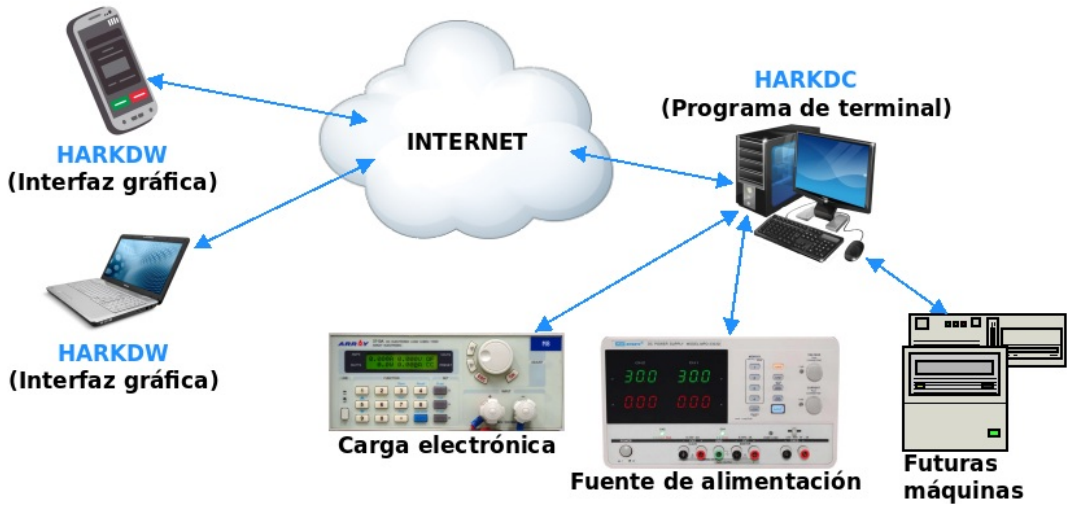


Figura 29. Futuro de HARKD.

Es un software con vistas al futuro, gracias a la separación de «frontend» (HARKDW) y «backend» (HARKDC) se podría crear una aplicación móvil que controlara los instrumentos ya que la tecnología avanza a marchas forzadas y los móviles están reemplazando a los ordenadores personales.

15 FUENTES DE INFORMACIÓN/BIBLIOGRAFÍA

1. GCC [Último acceso 8 noviembre 2018]. https://gcc.gnu.org/onlinedocs/
2. Libserialport [Último acceso 8 noviembre 2018]. https://sigrok.org/wiki/Libserialport
3. Libxlsxwriter [Último acceso 8 noviembre 2018]. https://github.com/jmcnamara/libxlsxwriter
4. UPV/EHU «Universidad del País Vasco» [Último acceso 8 noviembre 2018]. https://www.ehu.eus/es/
5. Pagina web oficial de Array. [Último acceso 8 noviembre 2018]. http://www.array.sh/yq-3700e.htm
6. Protocolo de comunicaciones carga electrónica Array 371XA. [Último acceso 8 noviembre 2018]. http://www.array.sh/download/Communication%20protocol%20for%20electronic%20load.pdf
7. Pagina del Wellzion. [Último acceso 8 noviembre 2018]. http://www.wellzion.com/Products/Programmable_Power_Supply/30V_5A_/MPD-3305D/
8. Java [Último acceso 8 noviembre 2018]. https://www.oracle.com/es/java/
9. Tcl [Último acceso 8 noviembre 2018]. https://www.tcl.tk/
10. Python [Último acceso 8 noviembre 2018]. https://docs.python.org/3/
11. Perl [Último acceso 8 noviembre 2018]. https://perldoc.perl.org/
12. Licencia LGPL [Último acceso 8 noviembre 2018]. https://doc.qt.io/qt-5.11/lgpl.html
13. Qt [Último acceso 8 noviembre 2018]. https://www.qt.io/download
14. BOPV (Boletín oficial del País Vasco) [Último acceso 8 noviembre 2018]. https://www.euskadi.eus/y22-bopv/es/bopv2/datos/Ultimo.shtml
15. CMake [Último acceso 8 noviembre 2018]. https://cmake.org/documentation/
16. GNU/Emacs [Último acceso 8 noviembre 2018]. https://www.gnu.org/software/emacs/manual/
17. Editor de textos TeXmacs. [Último acceso 8 noviembre 2018]. http://www.texmacs.org/tmweb/home/welcome.en.html