UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

AUTOMATIZACIÓN DE MEDIDAS PARA SISTEMAS DE COMUNICACIÓN POR LUZ VISIBLE MEDIANTE CÉLULAS ORGÁNICAS FOTOVOLTAICAS

TRABAJO FIN DE GRADO

Junio - 2022

AUTOR: Jorge García Brea

DIRECTOR: Pablo Corral González

RESUMEN

En la realización de este Trabajo Fin de Grado se pretende en primer lugar, mediante la herramienta de script de un osciloscopio digital (WaveForms), automatizar la toma de medidas a distintas frecuencias, de células orgánicas fotovoltaicas consiguiendo reducir considerablemente el tiempo dedicado a esta tarea, obteniendo los datos del comportamiento de cada célula de manera más eficiente.

Por otro lado, se buscará mediante la herramienta de programación MATLAB obtener un script para representar el diagrama de ojo de cada una de las medidas obtenidas anteriormente. De igual forma, buscaremos mediante otro script también en MATLAB calcular ciertos valores de calidad de la señal como son el factor de calidad Q, la tasa de error de bit (BER) y la relación señal a ruido (SNR), reflejando todos estos valores en una tabla para así poder realizar de una forma más sencilla un análisis de la calidad de cada señal.

Palabras clave: Célula, Script, Matlab, SNR, Q, BER.

ABSTRACT

In this Final Degree Project, firstly we will search using a script tool of a digital oscilloscope (WaveForms), to automate the taking of measurements at different frequencies of organic photovoltaic cells, achieving a significant disminution of the time dedicated to this task, obtaining the data of the behavior of each cell in a more efficient way.

On the other hand, we will search using the MATLAB programming tool to obtain a script to represent the eye diagram of each of the measures obtained previously. Similarly, we will search by another script also in MATLAB calculate certain values of signal quality such as the quality factor Q, the bit error rate (BER) and the signal to noise ratio (SNR), exporting all these values in a table, to be able to carry out a simple analysis of the quality of each signal.

Key words: Cell, Script, Matlab, SNR, Q, BER.

ÍNDICE

RESUMEN **3**

ABSTRACT **5**

ÍNDICE **7**

ÍNDICE DE FIGURAS **11**

ABREVIATURAS **15**

1. INTRODUCCIÓN Y OBJETIVOS **17**

1.1 MOTIVACIÓN 17

1.2 ESTADO DEL ARTE 18

1.3 OBJETIVOS A CONSEGUIR 20

1.4 ESTRUCTURA DEL TRABAJO 20

1.4.1 Apartados del trabajo 21

1.4.2 Secuenciación de los trabajos 22

2. MATERIAL Y MÉTODOS **25**

2.1 HARDWARE 25

2.1.1 Células orgánicas fotovoltaicas 25

2.1.2 Analog Discovery 2 27

2.2 SOFTWARE 29

2.2.1 WaveForms 29

2.2.2 MATLAB 31

2.3 FUNDAMENTOS TEÓRICOS 32

2.3.1 Comunicación con luz visible (VLC) 32

2.3.2 Relación señal a ruido (SNR) 33

2.3.3 Factor de calidad Q 34

2.3.4 Tasa de error de bit (BER) 37

2.3.5 Diagrama de ojo 38

2.3.6 Filtro Kalman 39
2.4 MONTAJE ELÉCTRICO PARA TOMA DE MEDIDAS 40
2.5 PROGRAMACIÓN DE LA AUTOMATIZACIÓN DE LA TOMA DE MEDIDAS EN EL OSCILOSCOPIO DIGITAL 42
2.5.1 Comprobaciones iniciales y declaración de variables de la célula . 44
2.5.2 Declaraciones y cálculos previos al bucle 46
2.5.3 Bucle for 49
2.5.4 Exportación de las medidas realizadas 51
2.5.4 Diagrama de flujo del script 52
2.6 REPRESENTACIÓN DIAGRAMA DE OJO 54
2.6.1 Diagrama de flujo del script 57
2.7 PROGRAMACIÓN CÁLCULO SNR, Q y BER 59
2.7.1 Obtención vector con archivos de medidas 59
2.7.2 Inicio bucle for y cálculo factor Q 61
2.7.2.1 Obtención valores máximos para nivel alto y nivel bajo 62
2.7.2.2 Obtención valores de σ1 y σ0 63
2.7.3 Corrección de posibles errores para el cálculo 66
2.7.3.1 Error falta de máximos en el archivo de histograma 66
2.7.3.2 Error falta valor para la interpolación de σ1 y σ0 67
2.7.4 Cálculo SNR y BER 68
2.7.5 Creación y exportación de tabla de resultados 68
2.7.6 Diagrama de flujo script del script 69
3. RESULTADOS 71
3.1 RESULTADOS OBTENIDOS PROGRAMACIÓN TOMA DE MEDIDAS OSCILOSCOPIO DIGITAL 71
3.1.1 Medición a frecuencias superiores a 200 KHz 79
3.2 RESULTADOS OBTENIDOS REPRESENTACIÓN DIAGRAMA DE OJO ... 82

3.3 RESULTADOS OBTENIDOS PROGRAMACIÓN CÁLCULO SNR, Q y BER 87

4. CONCLUSIONES Y FUTURAS LÍNEAS DE INVESTIGACIÓN 93

4.1 CONCLUSIONES DEL PROYECTO..... 93

4.2 LÍNEAS FUTURAS..... 95

5. ANEXOS..... 97

5.1 ANEXO 1: CÓDIGO COMPLETO WAVEFORMS DE AUTOMATIZACIÓN TOMA DE MEDIDAS 97

5.2 ANEXO 2: CÓDIGO COMPLETO MATLAB REPRESENTACIÓN DIAGRAMA DE OJO 101

5.3 ANEXO 3: CÓDIGO COMPLETO MATLAB CÁLCULO SNR, Q Y BER 103

5.4 ANEXO 4: CÓDIGO COMPLETO WAVEFORMS DE AUTOMATIZACIÓN TOMA DE MEDIDAS, AÑADIENDO POSIBILIDAD A FRECUENCIAS ALTAS (500 KHz, 1 MHz y 2 MHz). 108

5.5 ANEXO 5: INFORME DEL PROYECTO A PARTIR DEL DIAGRAMA DE GANTT 115

6. REFERENCIAS BIBLIOGRÁFICAS..... 119

ÍNDICE DE FIGURAS

Figura 1. Diagrama de Gantt del TFG. 23
Figura 2. Célula fotovoltaica orgánica encapsulada. 26
Figura 3. Portaplacas empleado para medir las células. 26
Figura 4. Soporte del portaplacas con distancias. 27
Figura 5. Osciloscopio Analog Discovery 2. 28
Figura 6. Esquema conexiones osciloscopio digital Analog Discovery 2. 28
Figura 7. Ejemplo medición realizada en laboratorio de células fabricadas 30-03-2022. 30
Figura 8. Captura de pantalla entorno programación scripts de WaveForms. .. 31
Figura 9. Captura de pantalla del software Matlab. 32
Figura 10. Espectro luz visible en espectro electromagnético. 33
Figura 11. Representación cálculo factor de calidad Q. 36
Figura 12. Tasa de error de bit (BER) vs factor Q en lineal. 37
Figura 13. Diagrama de ojo simple (arriba) y con distorsión (abajo). 38
Figura 14. Proceso filtro Kalman. 39
Figura 15. Diagrama sistema lineal discreto filtro Kalman. 39
Figura 16. Montaje eléctrico para toma de medidas mediante Fritzing. 41
Figura 17. Montaje eléctrico para toma de medidas en el laboratorio. 42
Figura 18. Captura ventana Ayuda (Help) WaveForms. 43
Figura 19. Captura parte de código valores iniciales célula. 44
Figura 20. Captura parte código comprobación conexión y ventana de salida. 45
Figura 21. Captura parte código previo a bucle for. 46
Figura 22. Captura de pantalla de parámetros a ajustar en osciloscopio. 47
Figura 23. Captura parte código cálculo pico-pico y offset. 48
Figura 24. Captura parte código inicio bucle y ajuste de frecuencia. 49
Figura 25. Captura parte código ajuste base de tiempos. 49
Figura 26. Captura parte código ajuste Range en función tensión pico-pico. ... 50
Figura 27. Captura parte código ajuste offset. 51
Figura 28. Captura parte código exportar datos osciloscopio. 51
Figura 29. Captura parte código exportar datos histograma y medidas. 52

Figura 30. Diagrama de flujo script automatización toma de datos WaveForms. 53
Figura 31. Captura parte código MATLAB obtener archivos datos osciloscopio. 54
Figura 32. Captura parte código MATLAB crear carpeta resultados. 55
Figura 33. Captura parte código MATLAB obtener valor de tensiones. 55
Figura 34. Captura parte código MATLAB obtener valor de frecuencia. 56
Figura 35. Captura parte código MATLAB representación diagrama de ojo. 56
Figura 36. Captura parte código MATLAB exportación resultados. 57
Figura 37. Diagrama de flujo script representación diagrama de ojo en Matlab. 58
Figura 38. Histograma ejemplo señal senoidal 3,3 V. 59
Figura 39. Histograma ejemplo medición célula fotovoltaica. 60
Figura 40. Captura parte código MATLAB obtener archivos datos histograma. 60
Figura 41. Captura parte código MATLAB obtener vectores datos histograma. 61
Figura 42. Captura parte código MATLAB obtener máximos histograma. 62
Figura 43. Captura parte código MATLAB obtener V0. 62
Figura 44. Captura parte código MATLAB obtener V1. 63
Figura 45. Captura parte código MATLAB obtener voltajes al 60%. 63
Figura 46. Captura parte código MATLAB obtener valores cercanos. 64
Figura 47. Representación σ1 y σ0 en el cálculo de Q. 64
Figura 48. Captura parte código MATLAB obtener valores cercanos por arriba y por abajo. 64
Figura 49. Captura parte código MATLAB interpolación valores V0 al 60%. 65
Figura 50. Captura parte código MATLAB valor σ0. 65
Figura 51. Captura parte código MATLAB cálculo factor Q. 65
Figura 52. Captura parte código MATLAB error al menos dos máximos. 66
Figura 53. Captura parte código MATLAB situación error al menos un valor cercano por arriba. 67
Figura 54. Captura parte código MATLAB situación error al menos un valor cercano por abajo. 67
Figura 55. Captura parte código MATLAB cálculo fórmula SNR y BER. 68
Figura 56. Captura parte código MATLAB creación tabla de resultados. 68

Figura 57. Captura parte código MATLAB exportación tabla de resultados. 69
Figura 58. Diagrama de flujo script obtención valor SNR, Q y BER en Matlab. 70
Figura 59. Captura parte código posible solución base de tiempos osciloscopio. 71
Figura 60. Osciloscopio sin ajustar la base de tiempos. 72
Figura 61. Osciloscopio una vez ajustada la base de tiempos. 72
Figura 62. Captura parte código posible solución Range osciloscopio. 73
Figura 63. Captura parte código posible solución Range osciloscopio sin emplear filtro Kalman. 74
Figura 64. Captura ejemplo ejecución Script automatización toma de medidas. 75
Figura 65. Captura ejemplo archivos exportados del osciloscopio. 76
Figura 66. Captura ejemplo archivo de datos histograma exportado. 76
Figura 67. Ejemplo histograma exportado en formato .png. 77
Figura 68. Captura ejemplo archivo de datos de osciloscopio exportado..... 77
Figura 69. Ejemplo osciloscopio exportado en formato .png. 78
Figura 70. Captura ejemplo archivo de datos de medidas exportado. 79
Figura 71. Ejemplo captura osciloscopio mediciones realizadas a célula de perovskita (Arriba: Frecuencia 1 KHz. Abajo: Frecuencia 500 KHz). 80
Figura 72. Captura parte código comprobación tensión pico-pico mediciones a frecuencias altas. 81
Figura 73. Captura ejemplo archivos exportados osciloscopio a 500 KHz, 1 MHz y 2 MHz. 81
Figura 74. Representaciones diagrama de ojo (primeras pruebas con errores). 82
Figura 75. Representaciones diagrama de ojo (primeras pruebas con errores) (2). 83
Figura 76. Captura ejemplo exportación diagramas de ojo. 84
Figura 77. Representación diagrama de ojo para señal a 1 KHz y 0 cm. 84
Figura 78. Representación diagrama de ojo para señal a 1 KHz y 10 cm. 85
Figura 79. Representación diagrama de ojo para señal a 200 KHz y 5 cm. 85
Figura 80. Representación diagrama de ojo para señal a 10 KHz y 3 cm. 86

Figura 81. Ejemplo histograma obtenido sin dos valores máximos. 87
Figura 82. Resultado osciloscopio ejemplo histograma obtenido sin dos valores máximos. 88
Figura 83. Ejemplo fila de tabla de resultados con falta de máximos. 88
Figura 84. Ejemplo datos archivo histograma. 89
Figura 85. Representación interpolación lineal. 90
Figura 86. Valor obtenido con Matlab interpolación ejemplo. 91
Figura 87. Ejemplo ejecución script en Matlab. 91
Figura 88. Captura ejemplo archivo exportado de tabla resultados. 92
Figura 89. Captura ejemplo tabla de resultados cálculos Matlab. 92
Figura 90. Gráfica comparación tiempo dedicado al proceso de toma de medidas de una célula. 94

ABREVIATURAS

Abreviatura	Significado
LiFi	Tecnología comunicación inalámbrica (*Light Fidelity*)
SNR	Relación señal a ruido (*Signal to noise ratio*)
V	Voltios
BER	Tasa de error de bit (*Bit error rate*)
AWGN	Ruido blanco Gaussiano aditivo (*Additive white Gaussian noise*)
Erfc	Función error complementario
VLC	Comunicación con luz visible (*Visible light communication*)
KHz	Kilohercio
cm	Centímetro
LED	Diodo emisor (*Light-emitting diode*)
KF	Filtro Kalman
dB	Decibelios
mV	Milivoltio
AD2	Analog Discovery 2
nm	Nanómetro

1. INTRODUCCIÓN Y OBJETIVOS

1.1 MOTIVACIÓN

Las comunicaciones inalámbricas se definen como el tipo de comunicación en la que no se emplea un medio de propagación físico, se utiliza la modulación de ondas electromagnéticas, propagadas por el espacio comunicando un extremo (emisor) con el otro extremo (receptor).

Este tipo de comunicaciones presentan una serie de ventajas claras, en la actualidad se tiende a eliminar los cables en todos los tipos de comunicación, lo que nos permite una mayor movilidad, proporcionando una flexibilidad del sistema en general. Por otro lado, las redes inalámbricas nos proporcionan una mayor accesibilidad de todos los equipos y usuarios a los recursos de la red. En cuanto a los costes de mantenimiento y configuración son menos elevados que en una red cableada y nos permite una mayor escalabilidad.

También hay que tener en cuenta que hay desventajas a la hora de emplear las comunicaciones inalámbricas, la principal es la seguridad, ya que resulta más difícil controlar el acceso a la red, lo que puede conllevar problemas de robo de información, entre otros. Otra desventaja es la menor velocidad de transmisión de datos en comparación con redes cableadas.

La tecnología LiFi es un ejemplo de este tipo de comunicaciones inalámbricas, más en concreto la transmisión de datos mediante luz visible (VLC) es la tecnología que empleamos en este proyecto, la explicaremos con más detalle a lo largo del TFG.

Por otro lado, en la industria y más en concreto en el ámbito de las tecnologías se está buscando la automatización de procesos, que consiste en convertir un procedimiento que antes se ejecutaba de manera manual, para que este se realice de una manera autómata. A nivel empresarial resulta muy interesante al conseguir generar una mayor eficiencia en los procesos, ahorrando tiempo, esfuerzo y con total seguridad dinero. Esto mismo, pero a un nivel más bajo es lo que se busca con este Trabajo Fin de Grado, conseguir automatizar una toma de medidas que hasta este momento se realizaba de forma manual.

1.2 ESTADO DEL ARTE

El hecho de realizar un proceso manual de forma repetitiva, junto con la necesidad de avanzar a mayor velocidad en las investigaciones, hacen de la automatización de la técnica una necesidad. Algunas de las investigaciones existentes sobre este tema ya hacen ver la posibilidad de ajustar los aparatos de laboratorio, en nuestro caso el osciloscopio digital para realizar las medidas de una forma más eficiente.

La automatización en los laboratorios de investigación tiene varias décadas, muestra de ello podemos ver como las empresas más potentes de instrumentación y control, han dedicado los últimos años a desarrollar herramientas que permiten automatizar numerosos procesos, una de estas empresas es *National Instruments* con la plataforma *LabView*. Es una herramienta gráfica e intuitiva de programación a través de la cual el usuario es capaz de mediante bloques generar un algoritmo a seguir por el programa. La mayor desventaja de esta plataforma es que es un software propietario, es decir, se requiere de una licencia para su uso.

Hay más herramientas similares a las mencionadas, pero a la hora de elegir como desarrollar la automatización en este proyecto, nos interesa trabajar con los elementos ya empleados con anterioridad en el laboratorio de la Universidad Miguel Hernández, como es el osciloscopio digital *Analog Discovery* 2 y su software *WaveForms.*

Un Trabajo Fin de Grado que aborda este tema es: "Automatización de medidas de eficiencia energética" [8] desarrollado por Gabriela Barreto, Natalia Da Rosa y Jorge Freire en el año 2021; en el cual los autores buscan desarrollar un software que automatiza la toma de valores eléctricos y fotométricos en la Universidad de Uruguay. Nuestra necesidad de automatizar un proceso de medición es la misma que en ese TFG, a diferencia que en el nuestro se busca la medición de células fotovoltaicas orgánicas y no los parámetros eléctricos de diodos LED. Por otro lado, en nuestro caso emplearemos la herramienta del software del osciloscopio digital, mientras que el trabajo mencionado emplea una herramienta propia de su universidad.

Otro Trabajo Fin de Grado que trata sobre un tema similar al nuestro es: "Automatización de una fuente de corriente-voltaje y un osciloscopio digital en el entorno de programación *LabView*" [9] desarrollado por Alejandro Moreno Martín en el año 2019 en la Universidad Carlos III de Madrid; en el cual se busca desarrollar la automatización de dos equipos de laboratorio para hacer que las mediciones sean más sencillas, para ello hace uso de la plataforma *LabView* que hemos mencionado al comienzo de este apartado. A diferencia de nuestro caso, que buscamos un script de automatización para un caso en concreto (células fotovoltaicas orgánicas), este otro proyecto fija el objetivo en hacer más intuitivo y cómodo mediciones más básicas y de un nivel más general.

Por otro lado, en cuanto a la parte que también trataremos en este TFG de la obtención de los parámetros de calidad de la señal y del diagrama de ojo, encontramos en el artículo del congreso ICOCN (International Conference on Optical Communications and Networks) del año 2017: "Visible Light Communication Using a Solar-Panel Receiver", [10] en el que los autores muestran un caso de comunicación mediante luz visible, aportando datos de la tasa de error de bit (BER) de en torno a 1.6883×10^{-3}. La tasa de error de bit es un parámetro que en este trabajo calcularemos mediante Matlab a partir de las medidas realizadas en el osciloscopio.

Por último, otro trabajo interesante que trata sobre las comunicaciones mediante luz visible es el artículo de investigación del año 2017: "Performance enhancement technique of visible light communications using passive photovoltaic cell", [11] en el que los autores realizan una descripción de la investigación donde evalúan experimentalmente el factor de calidad Q de un sistema VLC, con receptor de célula solar, aportando resultados de este parámetro para distintas señales en el espacio libre. Además, relacionan estos resultados con la apertura de los diagramas de ojo de las señales. Tanto el diagrama de ojo como el factor Q son conceptos que desarrollaremos y aplicaremos en este TFG.

1.3 OBJETIVOS A CONSEGUIR

En este Trabajo Fin de Grado como hemos comentado anteriormente, el primer objetivo que buscamos es conseguir automatizar el proceso de toma de medidas a distintas frecuencias, de la respuesta de una célula orgánica fotovoltaica empleando un osciloscopio digital (Analog Discovery 2), que nos permite a través de su propio software (WaveForms) mediante una ventana de script, programar una serie operaciones, consiguiendo con esto reducir de manera notable el tiempo dedicado a obtener los datos necesarios de cada célula.

Por otro lado, buscaremos mediante el software MATLAB, realizar en primer lugar un script que nos permita a partir de los datos exportados del osciloscopio digital representar el diagrama de ojo para cada una de las mediciones y exportar dicha representación para futuras consultas. En segundo lugar, se buscará mediante otro script a partir de los datos de las mediciones, calcular los valores de tensión para el nivel bajo y para el nivel alto, el valor de la relación señal a ruido (SNR), el factor de calidad Q y la tasa de error de bit (BER), mostrando todos estos valores en una tabla que se exportará.

Como objetivo final, se buscará analizar los distintos resultados obtenidos a partir de los códigos de programación que se han llevado a cabo a lo largo del proyecto, valorando el logro de los objetivos marcados.

Es muy importante tener en cuenta la importancia de llevar a cabo los distintos objetivos de forma ordenada, ya que es de vital importancia tener el código de automatización del osciloscopio digital correctamente optimizado, para que los datos que obtengamos a partir de él sean correctos y no hagamos posteriormente cálculos con mediciones incorrectas que no llevarían a nada.

1.4 ESTRUCTURA DEL TRABAJO

En este apartado explicaremos en primer lugar, la estructura de apartados en los que se va a desarrollar el TFG, y por otro lado realizaremos una breve explicación de los trabajos realizados a lo largo del tiempo, mediante un diagrama de Gantt.

1.4.1 Apartados del trabajo

Este Trabajo de Fin de Grado está organizado en 6 capítulos, que se explican de manera breve a continuación:

- **Capítulo 1:** Es el apartado actual, estamos haciendo una introducción al trabajo explicando en primer lugar los motivos por los cuales se ha elegido este, realizando una breve explicación del estado actual de la técnica sobre el tema que hemos planteado, los objetivos que se pretenden conseguir, la estructura del documento y mediante un diagrama de Gantt exponer los distintos pasos realizados a lo largo del proyecto.
- **Capítulo 2**: En este apartado se explicarán todo el material empleado para el desarrollo del trabajo, tanto a nivel de hardware como a nivel de software. También se explicarán los distintos fundamentos teóricos que han sido necesarios implementar en el desarrollo del proyecto. Por último, se describirán los métodos empleados y una descripción de los trabajos desarrollados para la obtención de los distintos resultados.
- **Capítulo 3:** En este apartado se analizarán los resultados obtenidos en las mediciones, mediante gráficos, tablas y otros elementos que nos permitan visualizar los resultados de una manera más detallada y cómoda. Reflejando los resultados que se han ido obteniendo a lo largo del TFG, previos a obtener los deseados.
- **Capítulo 4**: En este apartado, una vez obtenidos los resultados se expondrán las conclusiones a todo el trabajo, valorando la obtención de los objetivos planteados al comienzo y mencionando posibles mejoras a futuro.
- **Capítulo 5:** En este apartado se adjuntarán los distintos anexos necesarios para complementar el trabajo: códigos de programación completos, informe diagrama de Gantt, etc.
- **Capítulo 6**: Por último, en este apartado se añadirán todas las referencias bibliográficas empleadas a lo largo de Trabajo Fin de Grado.

1.4.2 Secuenciación de los trabajos

Para tener una idea general de los trabajos que se han realizado, se desarrolla un diagrama de Gantt del TFG, que es una herramienta muy útil para planificar proyectos, al proporcionarnos una vista general de las tareas necesarias para la finalización de estos. En el diagrama de Gantt se debe mostrar: la fecha de inicio, la fecha de finalización, que tareas son necesarias en el desarrollo del proyecto y su distribución en el tiempo.

Para realizar esta representación se ha empleado el software *GanttProject*, [31] que es un programa de código abierto que se emplea para la programación y gestión de proyectos, también nos proporciona la posibilidad de exportar informes, donde se puede ver información importante como: el tiempo dedicado a cada tarea, el diagrama de Gantt y la utilización de los recursos. De igual forma, nos proporciona el diagrama PERT, que es una herramienta similar al diagrama de Gantt, que nos permite analizar cada tarea necesaria para completar el proyecto.

En 5.5 ANEXO 5: INFORME DEL PROYECTO A PARTIR DEL DIAGRAMA DE GANTT se adjunta el informe completo que nos proporciona la herramienta *GanttProject*, donde podemos ver con detalle todas las tareas realizadas, el tiempo que ha sido necesario para llevarlas a cabo, la duración total del TFG, etc.

El diagrama de Gantt de este proyecto es el siguiente (*siguiente página*):

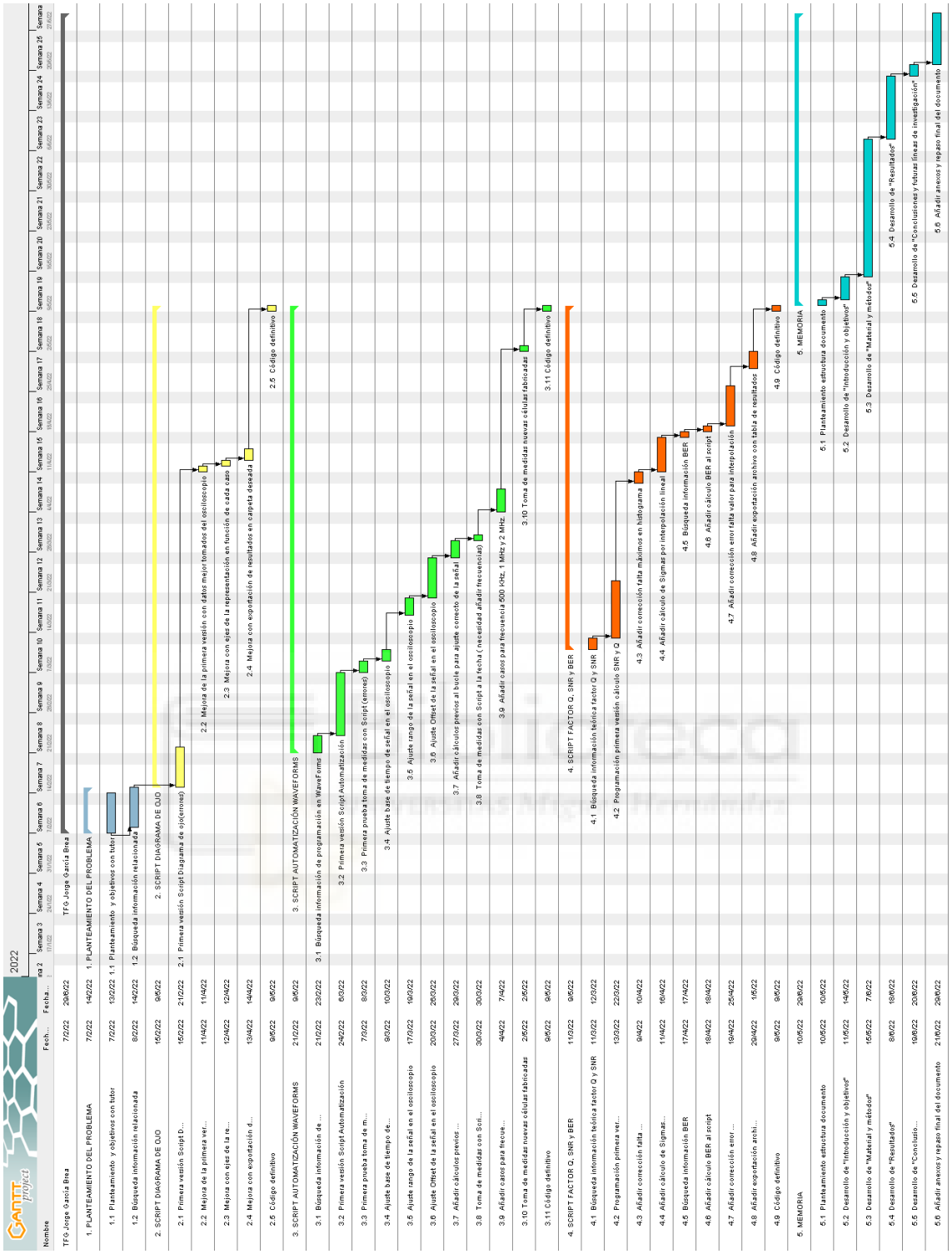


Figura 1. Diagrama de Gantt del TFG.

Por el tamaño de la imagen, para poder visualizarlo de una mejor manera se adjunta el enlace al diagrama: https://drive.google.com/file/d/1qZ5Jxk8RT-DweXif0qXLtltbS4qBCXE6/view?usp=sharing.

2. MATERIAL Y MÉTODOS

En este apartado vamos a describir todos los equipos y materiales empleados para la realización del Trabajo Fin de Grado, tanto a nivel de hardware como a nivel de software. Más adelante se describirán los métodos y conceptos teóricos que se han utilizado en el procedimiento experimental. Por último, se explicará también paso a paso la programación de los distintos scripts necesarios para el proyecto.

2.1 HARDWARE

En primer lugar, vamos a enumerar los elementos de hardware necesarios a lo largo del TFG.

2.1.1 Células orgánicas fotovoltaicas

El elemento principal sobre el que gira toda la investigación del trabajo, son las células fotovoltaicas orgánicas fabricadas en la Universidad Miguel Hernández de Elche.

Son celdas solares en las que sus capas están compuestas por elementos orgánicos, [17] la gran mayoría de las células que se han empleado en este Trabajo de Fin de Grado están compuestas por polímeros orgánicos como el polímero poli (3, 4 –etilendioxiotiofeno)-poli (estireno sulfonato), PEDOT:PSS, que es un polímero resultante de la mezcla de dos ionómeros. Un ionómero es un polímero que se caracteriza por tener unidades repetitivas que no están balanceadas eléctricamente y no tienen carga neta. Los dos ionómeros que lo componen son el PEDOT con carga positiva y el PSS con carga negativa.[12]

Es muy sensible a la radiación ultravioleta, a temperaturas altas y a la humedad, lo que causa una rápida degradación. La mayoría de las células además del PEDOT:PSS, están compuestas por P3HT:PCBM (cadena polimérica y fullereno), entre otros elementos.

Tanto la composición como la fabricación de las células fotovoltaicas no son de interés en este trabajo, ya que los objetivos de este son mejorar la toma de datos

y agilizar las mediciones, obteniendo resultados de calidad. Son parte del hardware empleado a lo largo del trabajo, pero no motivo de estudio.

Las células fotovoltaicas orgánicas que se han empleado tienen el siguiente aspecto:

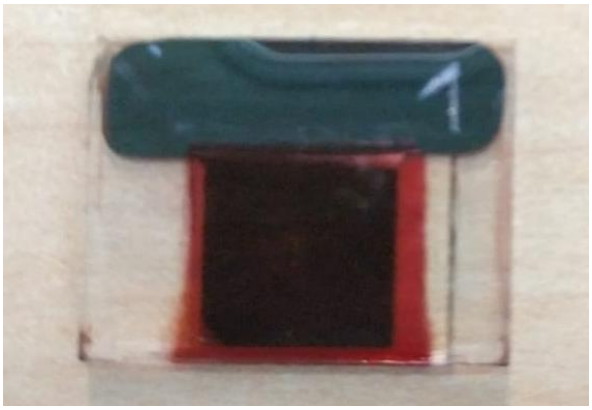


Figura 2. Célula fotovoltaica orgánica encapsulada.

Es importante mencionar que las células han sufrido una evolución a lo largo del tiempo en su fabricación, en un primer lugar se elaboraban sin encapsular, en la actualidad mediante la encapsulación se busca evitar la oxidación que hace que se degrade con gran velocidad, una siguiente fase de mejora será la fabricación en cámaras de vacío.

Para realizar las mediciones, en el laboratorio se dispone de un portaplacas fabricado mediante una impresora 3D, con el tamaño justo de las células y con las conexiones para cada ánodo (Ánodo 1 a Ánodo 6) y otra para el cátodo común. En la siguiente imagen podemos ver la célula sobre el mencionado portaplacas:

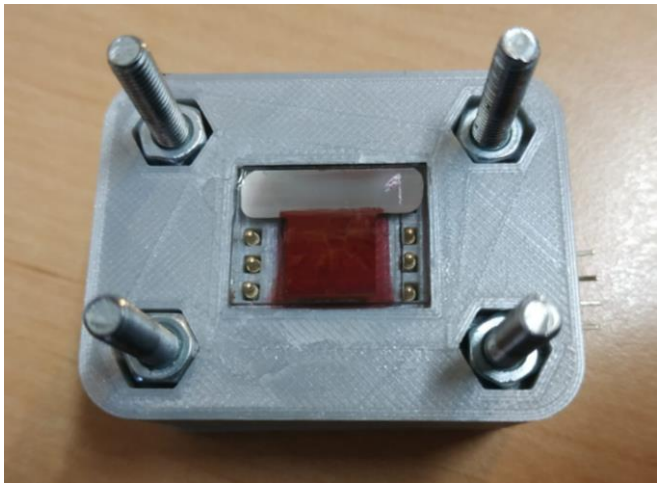


Figura 3. Portaplacas empleado para medir las células.

Este portaplacas se sitúa en un soporte, para realizar las mediciones está marcado con las distancias en centímetros, como podemos ver en la siguiente imagen:

Figura 4. Soporte del portaplacas con distancias.

2.1.2 Analog Discovery 2

El Analog Discovery 2 es un osciloscopio USB que permite a los usuarios medir, generar, registrar y controlar circuitos de señal mixta de todo tipo. Cuenta con un entorno software muy completo que expondremos en el siguiente apartado, sobre el que se basa gran parte del trabajo realizado.[27]

Figura 5. Osciloscopio Analog Discovery 2.

El dispositivo Analog Discovery 2 cuenta con entradas y salidas tanto analógicas como digitales. Para nuestras mediciones, emplearemos los dos canales del osciloscopio, por un lado, en el Canal 1 para generar en el emisor una señal sinusoidal y en el Canal 2, para medir en el receptor (célula fotovoltaica) la señal recibida, generando archivos con medidas, histograma y capturas de la pantalla del osciloscopio.

Las conexiones del dispositivo las podemos apreciar en el siguiente esquema que nos proporciona el propio fabricante:

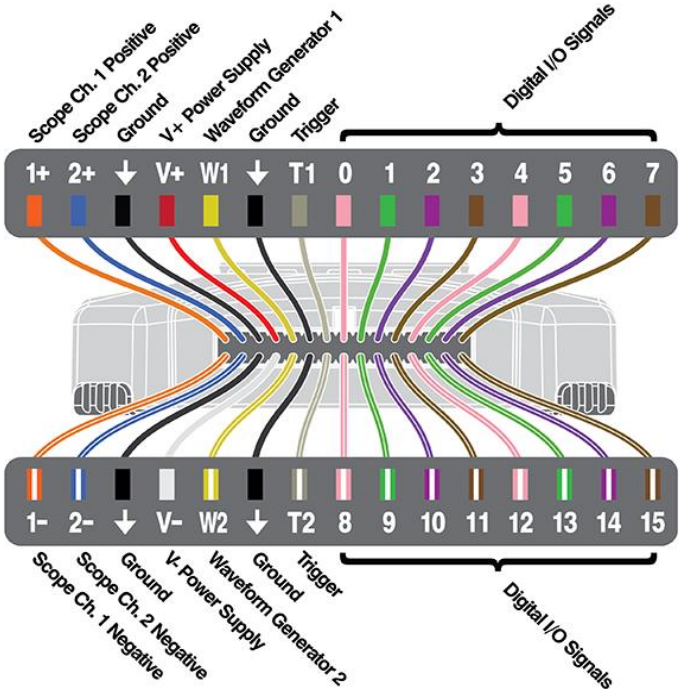


Figura 6. Esquema conexiones osciloscopio digital Analog Discovery 2.

Podemos observar los pines del Canal 1, positivo y negativo, de igual manera para el Canal 2, el nodo común (Ground), los pines de alimentación y del generador de señal, que son los que emplearemos en el transcurso del proyecto.

2.2 SOFTWARE

En este apartado vamos a enumerar los elementos de software que se han empleado a lo largo de este TFG.

2.2.1 WaveForms

Es el software del osciloscopio digital Analog Discovery 2 que hemos mencionado anteriormente. Lo emplearemos a lo largo del trabajo para realizar las distintas mediciones con las células y obtener los datos buscados.

Una vez iniciado el programa debemos conectar el dispositivo, en nuestro caso es el Analog Discovery 2, pero se puede utilizar el software sin conectar ningún dispositivo, nos permite emplear un dispositivo "DEMO". El programa cuenta con algunos códigos de ejemplo que pueden ayudar a entender el funcionamiento de este.

El programa dispone de distintas opciones como son, el osciloscopio (Scope), el generador de señales (Wavegen), el analizador de espectros (Spectrum), el voltímetro (Voltmeter), la interfaz de programación (Script), entre otras funciones. A lo largo del proyecto vamos a emplear, el generador de señales, el osciloscopio y cobrará una gran importancia la interfaz de programación, con la que se nos permitirá automatizar las mediciones a distintas frecuencias, alcanzando uno de los objetivos de este trabajo.

La interfaz tiene el siguiente aspecto una vez iniciado el osciloscopio:

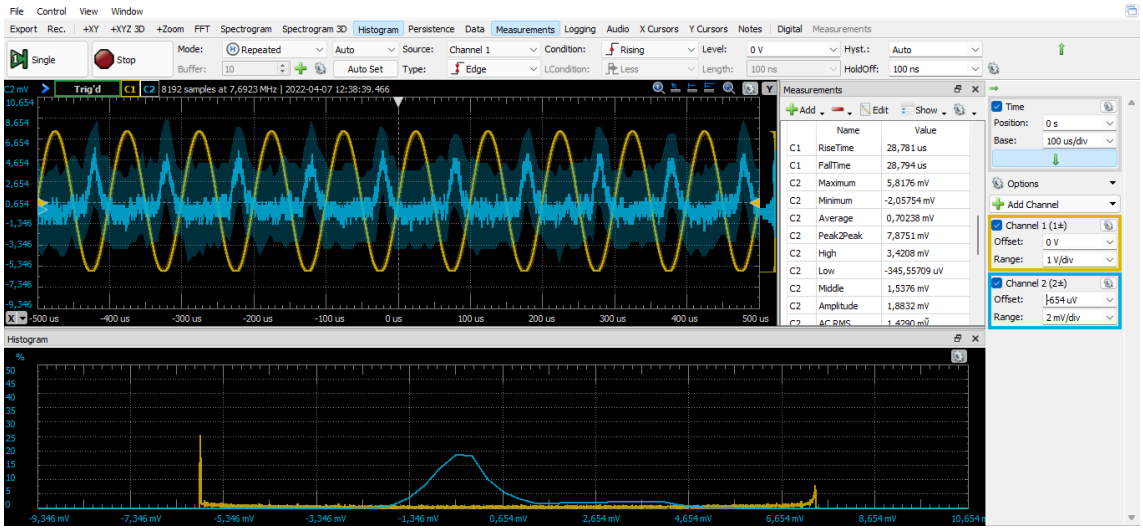


Figura 7. Ejemplo medición realizada en laboratorio de células fabricadas 30-03-2022.

Es interesante tener en cuenta como el software nos permite realizar exportación de los datos que nos interesa almacenar, en nuestro caso como ya explicaremos más adelante, resultará interesante obtener los siguientes archivos:

- Archivo de imagen (.png) y de datos (.csv) del osciloscopio.
- Archivo de imagen (.png) y de datos (.csv) del histograma.
- Archivo de datos (.csv) de las medidas de ambos canales del osciloscopio.

Posteriormente con estos datos almacenados trabajaremos con ellos para obtener los valores de calidad de la señal, el nivel de señal a ruido, la representación del diagrama de ojo, etc.

Como hemos mencionado anteriormente, el software WaveForms cuenta con la posibilidad de mediante una interfaz de programación en lenguaje JavaScript, desarrollar códigos que nos permitan realizar una serie de mediciones de manera más eficiente.

El entorno de programación del software tiene el siguiente aspecto:

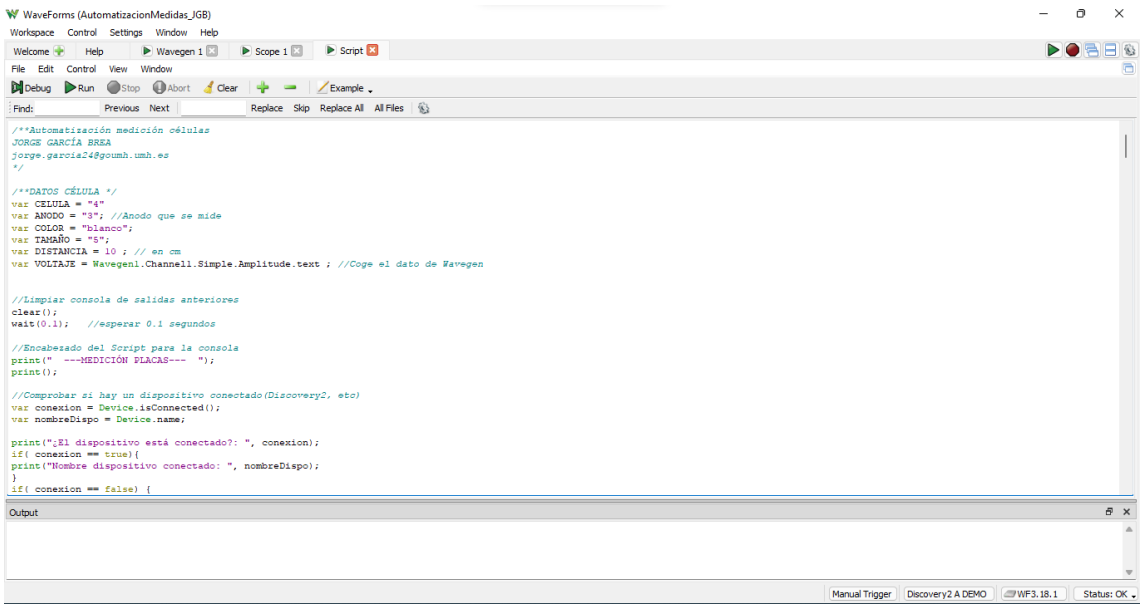


Figura 8. Captura de pantalla entorno programación scripts de WaveForms.

2.2.2 MATLAB

Matlab es un programa desarrollado por MathWorks, empresa estadounidense líder en desarrollo de software para cálculo técnico. Matlab está basado en un lenguaje de programación propio (lenguaje M), es muy completo ya que nos permite realizar múltiples cálculos en muy poco tiempo, así como representar todo tipo de gráficas tanto en 2D como en 3D, también nos facilita trabajar con funciones predefinidas en el programa lo que hace mucho más cómodo utilizar la herramienta. [29]

Matlab nos ofrece la posibilidad de importar archivos de datos *csv* o de otros formatos, pudiendo realizar cálculos y representaciones con valores obtenidos de fuentes externas, esto resulta muy interesante para nuestro TFG.

También nos permite exportar los resultados obtenidos por el programa, tanto de gráficas en formato de imagen (.png) como de los datos de los distintos cálculos que se realicen, en una tabla en formato *csv*.

Es una herramienta ampliamente utilizada en carreras universitarias técnicas como es el caso del Grado en Ingeniería de Tecnologías de Telecomunicación.

El aspecto del software es el siguiente:

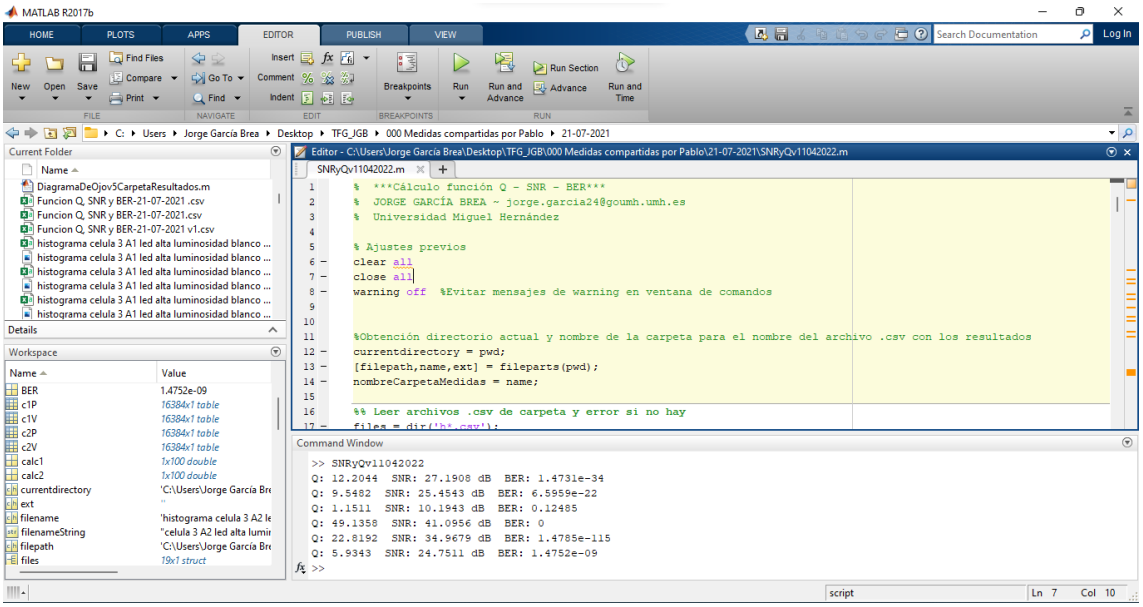


Figura 9. Captura de pantalla del software Matlab.

2.3 FUNDAMENTOS TEÓRICOS

En este apartado se van a explicar los distintos conceptos teóricos que resultan de vital importancia entender en el desarrollo del TFG.

2.3.1 Comunicación con luz visible (VLC)

La comunicación con luz visible, conocida por VLC del inglés *Visible light communications*, es una tecnología de comunicación inalámbrica en la que el espectro visible se modula para transmitir datos. Es una tecnología de corto alcance debido a la distancia de propagación de los diodos emisores (LED).[18]

En el espectro electromagnético el rango de la luz visible abarca entre los 400 nm y los 750 nm de longitud de onda, para unas frecuencias comprendidas entre $4x10^{14}$ Hz y $7.5x10^{14}$ Hz. En VLC se emplean diodos emisores LED ya que resulta muy sencillo modular su intensidad de corriente, siendo muy eficientes y proporcionando una larga vida útil.

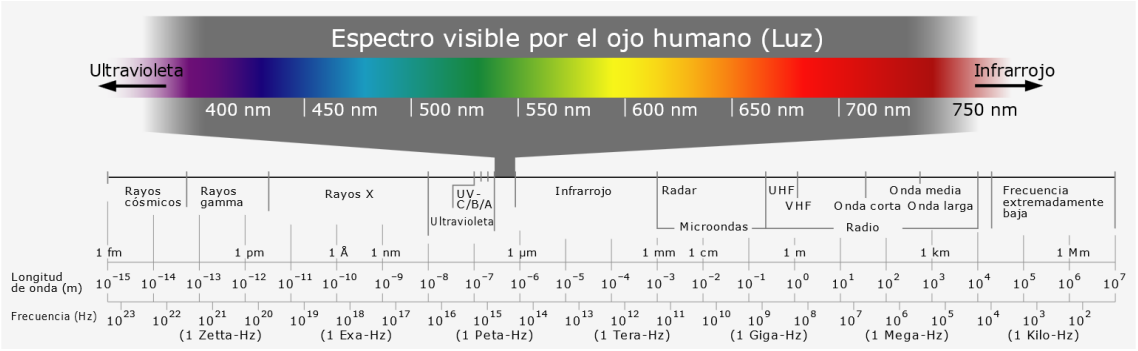


Figura 10. Espectro luz visible en espectro electromagnético.

2.3.2 Relación señal a ruido (SNR)

La relación señal a ruido proporciona una medida de calidad de una señal en un sistema determinado, depende del nivel de señal y del nivel de ruido. Tiene un impacto directo sobre el rendimiento del sistema.[13]

El nivel de señal lo podemos definir como el producto de la tensión por la corriente:

$$S = v * i \tag{1}$$

Empleando la ley de Ohm podemos realizar una sustitución, expresando la ecuación en función de una variable R idéntica para todo el cálculo:

$$S = v^2/R = i^2 * R \tag{2}$$

Por otro lado, el nivel de ruido lo definimos como aquella señal que nos produce interferencias, la cual no queremos. El nivel de ruido lo podemos calcular matemáticamente mediante las siguientes ecuaciones:

$$N = \frac{1}{T}\int_0^T i^2 * R \; dt = \; {\sigma_i}^2 * R$$
$$N = \frac{1}{T}\int_0^T v^2 * \frac{1}{R} \; dt = \; {\sigma_v}^2 * \frac{1}{R} \tag{3}$$

Donde N es el nivel de ruido, T es el periodo de integración y σ es la potencia cuadrática media.

Una vez obtenidos los niveles de señal y de ruido (*S* y *N*) de forma independiente, volvemos a la relación entre ellos, que resulta muy importante ya que está directamente relacionada con la tasa de error de bit (BER) de la que hablaremos en próximos apartados. De los resultados de los cálculos previos podemos expresar SNR como lo siguiente:

$$SNR = \frac{S}{N} = \frac{v^2/R}{{\sigma_v}^2/R} = \frac{v^2}{{\sigma_v}^2}$$
$$SNR = \frac{S}{N} = \frac{i^2 * R}{{\sigma_i}^2 * R} = \frac{i^2}{{\sigma_i}^2} \quad (4)$$

A lo largo de este trabajo emplearemos la primera expresión para el cálculo de la relación señal a ruido (SNR).

2.3.3 Factor de calidad Q

En los sistemas de comunicación digital contamos con dos posibles niveles de señal, un nivel bajo (0) y un nivel alto (1). Lo que nos llevaría a tener dos valores de SNR asociados cada uno de ellos a un nivel de la señal. Para poder obtener una medida general de ambos, vamos a emplear el factor Q, que nos permite combinar ambas SNR proporcionándonos un valor general de la calidad del sistema.

Para llegar a la expresión matemática del factor de calidad Q, que emplearemos en este trabajo, debemos tener en cuenta una serie de conceptos. Asumimos que el ruido blanco gaussiano (AWGN) es la primera causa de error a la hora de la toma de decisión entre el nivel bajo o el nivel alto de la señal. La función de probabilidad para una señal v(t) con AWGN se puede expresar de la siguiente manera:[13]

$$Prob[v(t)] = \frac{1}{\sqrt{2\pi{\sigma_x}^2}} * e^{-\frac{1}{2}*\left(\frac{v(t)-Vs}{\sigma_x}\right)^2} \quad (5)$$

Donde Vs es el nivel de tensión enviado, es decir, será por un lado la tensión para el nivel alto (1) y por otro lado la tensión para el nivel bajo (0) y σ es la desviación estándar del ruido.

Asumiendo que Vs puede tomar dos niveles de tensión: V0 para el nivel bajo y V1 para el nivel alto, la probabilidad de cometer un error a la hora de decidir el nivel la podemos expresar como:

$$P_{error} = Prob[v(t) > \gamma \mid Vs = V0] * Prob[Vs = V0] + Prob[v(t) < \gamma \mid Vs = V1] * Prob[Vs = V1] \quad (6)$$

Donde P_{error} es la probabilidad de cometer error y P[x|y] es la probabilidad condicionada. Asumimos a partir de ahora que la probabilidad de enviar V0 y V1 es la misma, por lo que Prob[Vs=V0] y Prob[Vs=V1] es igual a 0,5. Asumiendo esto, la expresión que habíamos obtenido anteriormente en la ecuación (6) se reduce de la siguiente manera:

$$P_{error} = Prob[v(t) > \gamma \mid Vs = V0] * 0.5 + Prob[v(t) < \gamma \mid Vs = V1] * 0.5 \quad (7)$$

$$P_{error} = \frac{1}{2}\int_{-\infty}^{\gamma} Prob[V(t), \sigma 0]\, dt + \frac{1}{2}\int_{\gamma}^{\infty} Prob[V(t), \sigma 1]\, dt \quad (8)$$

De esta expresión podemos observar como el error viene determinado por dos factores: la desviación estándar (σ0 y σ1) y el nivel de tensión de cada nivel (V0 y V1).

La probabilidad de error la podemos simplificar de la siguiente manera:

$$P_{error} = \frac{1}{2} Erf\left[\frac{V1 - \gamma}{\sigma 1}\right] + \frac{1}{2} Erf\left[\frac{\gamma - V0}{\sigma 0}\right] \quad (9)$$

Donde *Erf* la definimos como la función error, que viene determinada por la siguiente ecuación:

$$Erf(x) = \frac{1}{\sqrt{2\pi}}\int_{x=\gamma}^{\infty} e^{\left(\frac{-x^2}{2}\right)} dx \quad (10)$$

Todo lo comentado hasta este punto, lo podemos observar de forma gráfica y más sencilla en la siguiente figura:

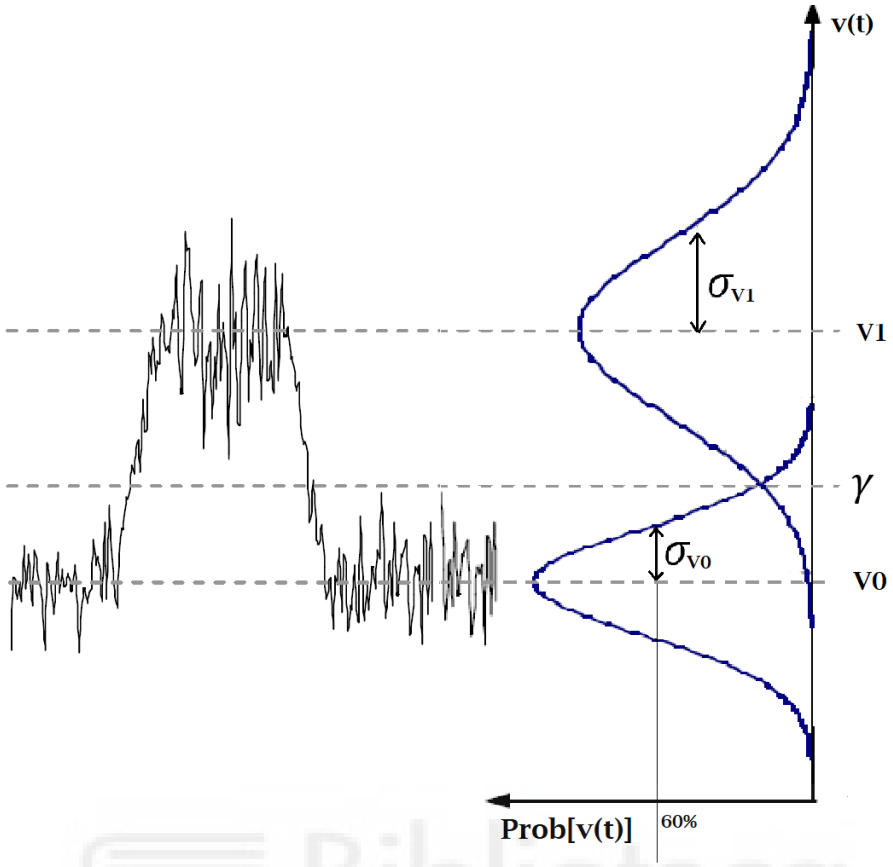


Figura 11. Representación cálculo factor de calidad Q.

Con lo explicado anteriormente y a partir de la definición del factor Q, podemos obtener el factor Q en función de los valores de tensión y de la desviación estándar de cada nivel:

$$Q = \frac{V1 - \gamma}{\sigma 1} = \frac{\gamma - V0}{\sigma 0} \tag{11}$$

Sustituyendo esta definición de Q en la ecuación (9) obtenemos que:

$$Perror = \frac{1}{2} Erf[Q] + \frac{1}{2} Erf[Q] = Erf[Q] \tag{12}$$

$$\gamma = \frac{V1 * \sigma 0 + V0 * \sigma 1}{\sigma 1 + \sigma 0} \tag{13}$$

Sustituyendo, obtenemos la siguiente expresión para el cálculo del factor Q:

$$Q = \frac{V1 - V0}{\sigma 1 + \sigma 0} \tag{14}$$

Esta será la expresión del factor de calidad Q que se empleará a lo largo de este TFG.

2.3.4 Tasa de error de bit (BER)

En las comunicaciones digitales la tasa de error de bit (Bit Error Rate - BER), se emplea para indicar la efectividad de un receptor a la hora de recibir los datos transmitidos por el emisor. La BER es la relación de bits erróneos con respecto a todos los bits recibidos en una transmisión de datos, normalmente resulta ser del orden de 10 elevado a un número negativo.[30]

Para obtener la tasa de error de bit en este trabajo vamos a calcularla a partir del factor de calidad Q que hemos explicado en el anterior apartado, mediante la siguiente ecuación que relaciona directamente el factor de calidad con la BER:[17]

$$BER = \frac{1}{2} * Erfc\left(\frac{Q}{\sqrt{2}}\right) \tag{15}$$

Donde *Erfc* es la función error complementaria, siendo 1-*Erf* con *Erf* definido en la ecuación (10).

En la siguiente figura podemos ver la relación de la tasa de error de bit frente al factor de calidad Q: [6]

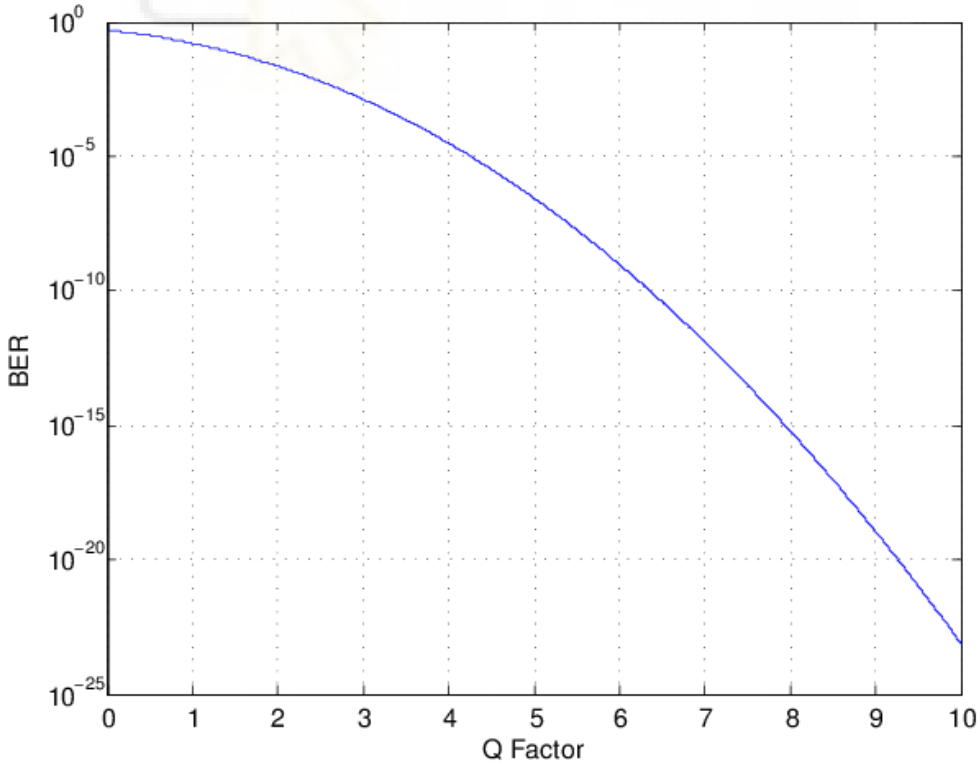


Figura 12. Tasa de error de bit (BER) vs factor Q en lineal.

Como se observa hay una relación directa entre ambas variables, cuando se incrementa el valor de Q, el valor de BER se reduce y al revés. Para una mejor calidad de la señal buscaremos siempre el mayor valor de Q posible, es decir, la menor tasa de error de bit.

2.3.5 Diagrama de ojo

El diagrama de ojo es una herramienta útil para analizar el comportamiento de las señales en comunicaciones digitales. Nos proporciona una evaluación rápida del rendimiento del sistema y nos da información del ruido con el que contamos en el canal.

Es una representación de la superposición de las señales que se reciben en el receptor (Canal 2 del osciloscopio). Es una herramienta que con tan solo un vistazo nos proporciona varios tipos de información: niveles de señalización y de ruido de cada nivel, relación señal a ruido, posibles distorsiones, errores de sincronía, entre otros.

En la siguiente figura se muestra un diagrama de ojo simple sin ninguna distorsión ni ruido y otro diagrama de ojo donde se incluye ruido y errores de sincronía:[5]

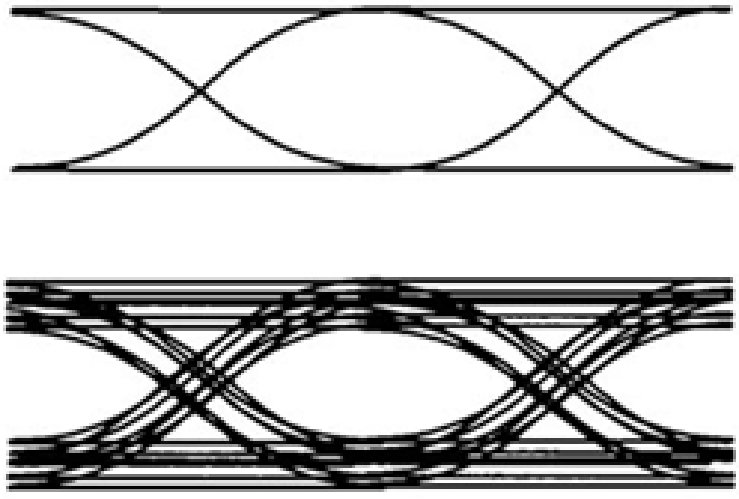


Figura 13. Diagrama de ojo simple (arriba) y con distorsión (abajo).

Para este trabajo emplearemos el software Matlab para representar a partir de los resultados exportados del osciloscopio, el diagrama de ojo para cada uno de los casos.

2.3.6 Filtro Kalman

El filtro Kalman es un método empleado para estimar un estado de un sistema, cuando no se puede medir directamente. El algoritmo consta de dos etapas: predicción y actualización. En la primera etapa, el algoritmo produce una estimación previa del estado y en la etapa de actualización se realizan nuevas estimaciones, en base a las medidas junto con la estimación anterior y la covarianza del error.[27]

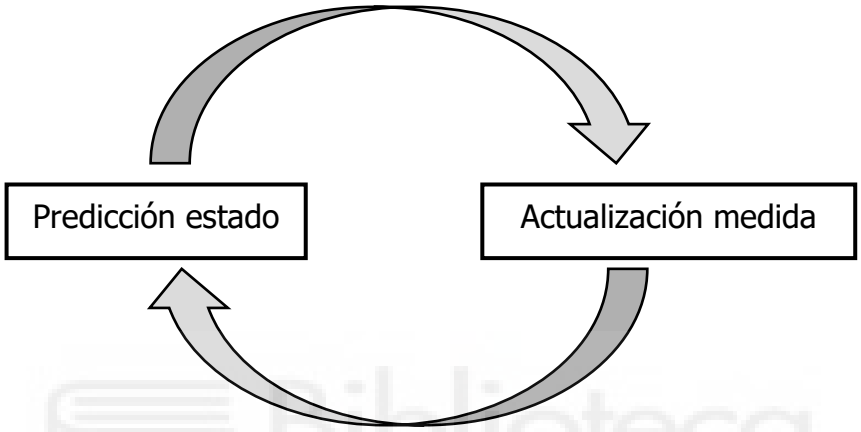


Figura 14. Proceso filtro Kalman.

De una forma resumida, en un filtro Kalman considerando un sistema lineal discreto como el de la figura 15. Es importante entender el concepto de estado (X_k) que se define como el conjunto mínimo de datos suficiente para describir el comportamiento del sistema, es decir, es el número mínimo de datos necesarios para predecir el valor futuro del sistema.[14]

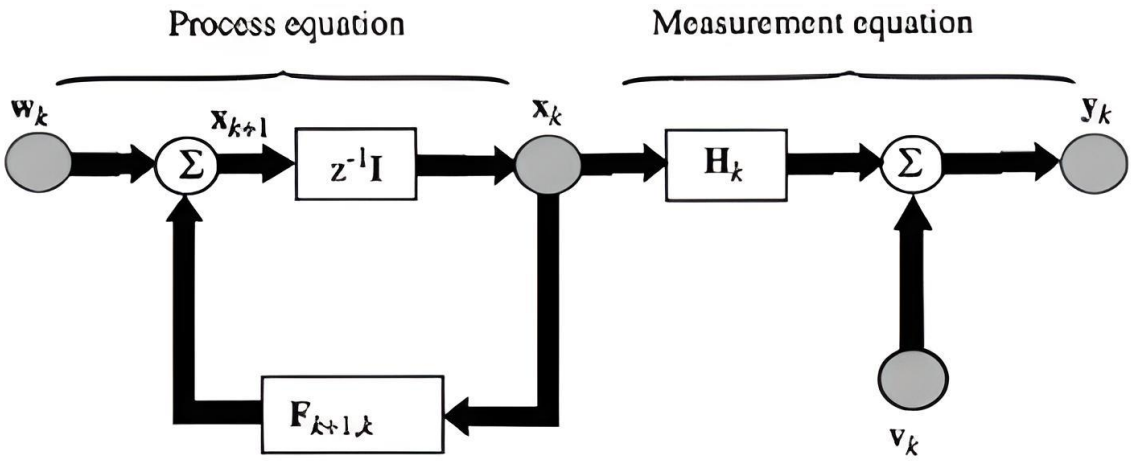


Figura 15. Diagrama sistema lineal discreto filtro Kalman.

En lenguaje matemático el diagrama anterior se puede expresar por la siguiente ecuación a la entrada, en la parte de la ecuación de proceso *(Process equation):*

$$X_{k+1} = F_{k+1,k} * X_k + W_k \quad (16)$$

Donde $F_{k+1,k}$ es la matriz de transición para pasar del estado X_k al X_{k+1}, el proceso cuenta con ruido W_k que se asume blanco y gaussiano.

Por otro lado, poseemos la siguiente ecuación para la salida, en la parte de ecuación de medida *(Measurement equation)*:

$$Y_k = H_k * X_k + V_k \quad (17)$$

Donde Y_k es el valor observado en el tiempo k-ésimo, H_k es la matriz de medidas y V_k es ruido blanco y gaussiano.

Por último, en este TFG se va a emplear la idea del filtro Kalman para una interpretación sencilla del mismo, para la obtención de un valor futuro de una variable conociendo la tendencia que siguen y el valor anterior de dicha variable, resolviendo así uno de los problemas que surgen para automatizar la toma de medidas en el osciloscopio digital.

2.4 MONTAJE ELÉCTRICO PARA TOMA DE MEDIDAS

En este apartado vamos a explicar y analizar el montaje eléctrico necesario para tomar las medidas de cada una de las células fotovoltaicas en el laboratorio.

Para tener una imagen visual del montaje se emplea una representación gráfica del mismo, utilizando la herramienta de diseño electrónico Fritzing, [32] que nos permite de una manera sencilla mediante su interfaz dinámica crear esquemas de circuitos.

Para este caso el montaje necesario resulta bastante sencillo, como podemos ver en la siguiente imagen:

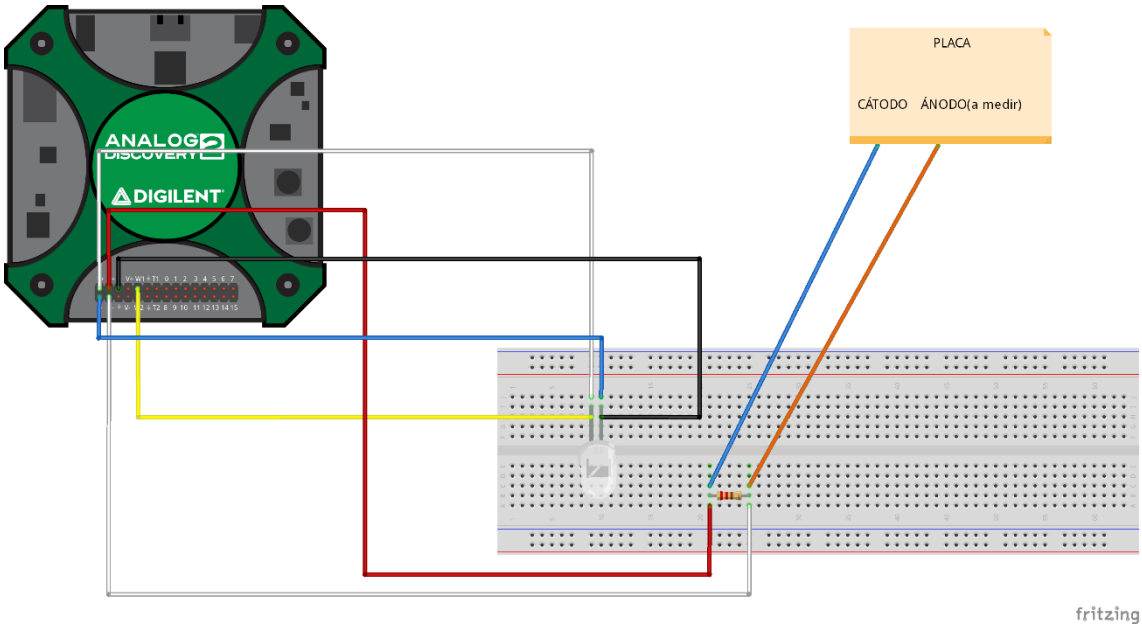


Figura 16. Montaje eléctrico para toma de medidas mediante Fritzing.

Como observamos tenemos por un lado la parte de emisión, que en nuestro caso es el diodo LED que empleamos para emitir la señal de la comunicación VLC, este se alimenta mediante el generador de señales W1 del osciloscopio AD2 y la otra pata del LED la conectamos a tierra (GND).

Por otro lado, tenemos la parte del receptor en la comunicación VLC, que en nuestro caso se trata de la célula fotovoltaica, que cuenta con el cátodo común y con los seis ánodos que se deben medir. Se realiza la conexión a una resistencia de valor 1 KΩ con la que se busca normalizar el valor obtenido y como el osciloscopio lleva una resistencia interna de 1 MΩ conseguimos que la corriente circule por los bornes del canal 2 del osciloscopio donde estamos midiendo.

Por último, debemos conectar los canales 1 y 2 del osciloscopio para observar la señal emitida y recibida respectivamente, en el caso del Canal 1 la conexión positiva va al pin de la pata del LED emisor, donde hemos conectado el generador de señales W1 y la conexión negativa a la otra pata del LED.

En el caso del Canal 2, la conexión positiva se enlaza con el pin de la resistencia que hemos conectado al cátodo común de la célula y la conexión negativa al pin de la resistencia que hemos conectado con el ánodo de la célula.

De esta manera obtenemos el montaje eléctrico necesario para realizar las mediciones a cada una de las células, lo único a tener en cuenta es que dependiendo del ánodo que queramos medir se debe conectar al pin del portaplacas que corresponda.

El montaje eléctrico en el laboratorio queda de la siguiente manera:

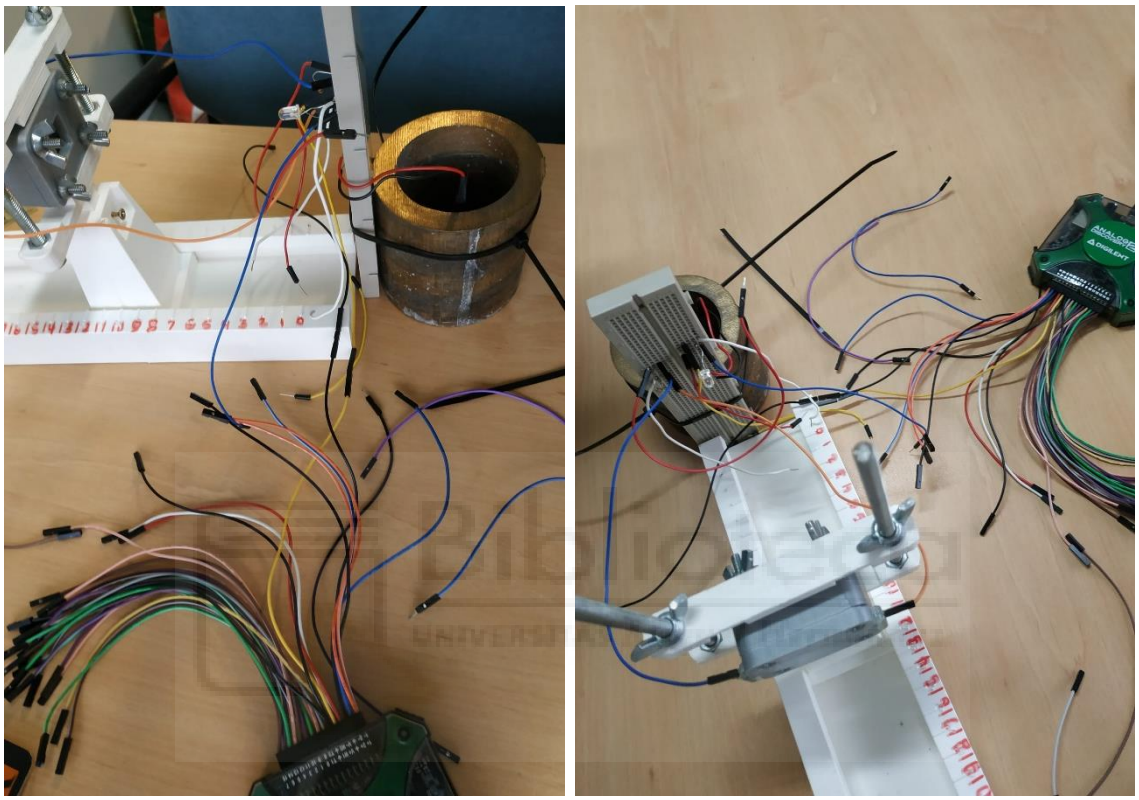


Figura 17. Montaje eléctrico para toma de medidas en el laboratorio.

2.5 PROGRAMACIÓN DE LA AUTOMATIZACIÓN DE LA TOMA DE MEDIDAS EN EL OSCILOSCOPIO DIGITAL

Como hemos indicado al comienzo del trabajo, uno de los objetivos principales del proyecto es conseguir automatizar en todo lo posible la toma de medidas con el osciloscopio digital, consiguiendo así reducir el tiempo dedicado a esta tarea y obtener mediciones de mayor calidad. En este apartado vamos a explicar el proceso para la programación del script que nos permitirá realizar esto.

El osciloscopio digital empleado en este TFG (Analog Discovery 2) como hemos comentado anteriormente, cuenta con un software que nos deja entre otras

múltiples funciones mediante una ventana de programación, escribir scripts para automatizar ciertas funciones, como la toma de datos, exportar archivos, etc. Esto es lo que emplearemos para nuestro caso.

El editor de scripts del software permite crear, ejecutar y depurar scripts que nos van a permitir controlar todos los instrumentos de los que dispone la herramienta.

En primer lugar, es interesante destacar que la ventana de scripts del software WaveForms está basado en el lenguaje JavaScript, que es el lenguaje de programación interpretado más popular, empleado para páginas web, pero también muy usado en entornos fuera de navegador, como es nuestro caso. Está basado en el estándar ECMA-262, que define el lenguaje de programación.

Es de vital importancia apoyarse en el manual del software para conocer las funciones y los objetos que se pueden emplear a la hora de escribir los códigos de automatización, para ello hay podemos entrar a la página web del fabricante (https://digilent.com/reference/software/waveforms/waveforms-3/reference-manual) o también en la ventana de Ayuda (Help) del programa, donde tenemos la misma información, como vemos en la siguiente imagen. Apoyándonos en este manual comenzamos a desarrollar el código.

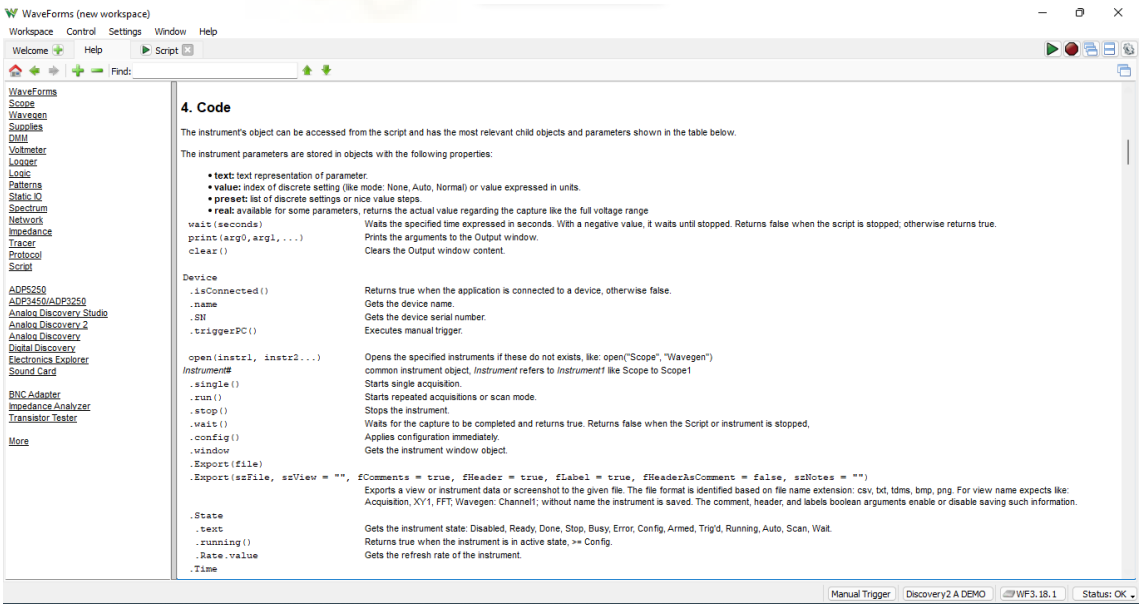


Figura 18. Captura ventana Ayuda (Help) WaveForms.

2.5.1 Comprobaciones iniciales y declaración de variables de la célula

En primer lugar, al inicio del script hay que tener en cuenta que debemos realizar una declaración de ciertos valores iniciales que corresponden a la célula fotovoltaica sobre la que se van a realizar las distintas medidas: número de célula, el ánodo que se va a medir, la distancia a la que se está midiendo, el voltaje con el que se está alimentando, el color y el tamaño del LED. Estos valores se deberán ajustar previo a las mediciones, sustituyendo al inicio del código los valores que correspondan, es importante ya que estas variables que se definen son las empleadas posteriormente para nombrar a los archivos que exportaremos con los resultados de las distintas mediciones. En la siguiente imagen podemos ver un ejemplo de valores para esta parte del código:

```
/**DATOS CÉLULA */
var CELULA = "1"
var ANODO = "6"; //Ánodo que se mide
var COLOR = "blanco";
var TAMAÑO = "5";
var DISTANCIA = 0 ; // en cm
var VOLTAJE = Wavegen1.Channel1.Simple.Amplitude.text; //Dato de Wavegen
```

Figura 19. Captura parte de código valores iniciales célula.

En la última línea de la figura anterior podemos ver un ejemplo de las sentencias con las que nos permite el software trabajar, en este caso mediante *Wavegen1.Channel1.Simple.Amplitude.text* obtenemos el valor de la tensión que se ha definido en el generador de señales (*Wavegen*) de WaveForms y la almacenamos en la variable *VOLTAJE*.

A lo largo del código se hace uso de sentencias como la anterior, que por un lado nos permiten escribir cierto valor que nos interese en el generador de señales o en el osciloscopio, o por otro lado, obtener el valor que tiene cierto parámetro y guardarlo en una variable.

Antes de iniciar lo que sería el bloque principal del código, se realizan una serie de comprobaciones para asegurarnos de que tenemos el osciloscopio digital conectado correctamente, y en caso de que no fuera así se nos indicaría por la ventana de salida (Output). Para esto vamos a emplear la función

Device.IsConnected() que nos devuelve *true* si el programa está conectado a algún dispositivo, o *false* en caso contrario.

A esta altura del código vamos a definir unas líneas que van a hacer que, a la hora de ejecutar y realizar las medidas, el script resulte más cómodo de entender, añadiendo un encabezado a la ventana de salida y una línea en la que se nos indica la célula que se está midiendo y más concretamente que ánodo, permitiendo hacer un seguimiento cuando se está en el proceso de realizar las mediciones. Todo lo mencionado anteriormente se puede observar en las siguientes líneas de código:

```
//Encabezado del Script para la consola
print("  ---MEDICIÓN PLACAS---  ");
print();

//Comprobar si hay un dispositivo conectado(Discovery2, etc)
var conexion = Device.isConnected();
var nombreDispo = Device.name;

print("¿El dispositivo está conectado?: ", conexion);
if( conexion == true){
print("Nombre dispositivo conectado: ", nombreDispo);
}
if( conexion == false) {
print("No hay dispositivo conectado");
}
print();   //Espacio para ordenar la salida por pantalla


//CÉLULA y ANODO que se está midiendo
print("Célula: " ,CELULA,"-","Ánodo: ",ANODO);
print();
```

Figura 20. Captura parte código comprobación conexión y ventana de salida.

Una vez se ha realizado la definición de valores iniciales y la comprobación de la correcta conexión del dispositivo, entramos en el bloque principal del código que lo podemos dividir en dos partes: por un lado, las declaraciones y cálculos previos al bucle *for* y por otro lado el bucle *for*.

2.5.2 Declaraciones y cálculos previos al bucle

Antes de entrar en el bucle for que será la parte del código que nos permita realizar las distintas mediciones para cada una de las frecuencias, hay que hacer algunas declaraciones de variables y cálculos previos.

En primer lugar, debemos definir un array de frecuencias que será el que posteriormente recorreremos en el bucle, en esta parte inicializamos el generador de señales (Wavegen1) para asegurarnos de que está en funcionamiento mediante *Wavegen1.run()* y establecemos la frecuencia inicial en el generador a 1 KHz, ajustando las unidades a KHz para los siguientes saltos de frecuencia, como podemos ver en la siguiente imagen:

```
//Definición de VARIABLES iniciales
//Array con frecuencias a medir
var frecuencias =[1,2,5,10,20,50,100,200] //en KHz

//Tamaño array frecuencias
var n = frecuencias.length;

//Iniciamos Wavegen
Wavegen1.run()
//Establecemos unidades de frecuencia en KHz
Wavegen1.Channel1.Simple.Frequency.text = '1 khz';

//Iniciamos osciloscopio
Scope1.run()
```

Figura 21. Captura parte código previo a bucle for.

Posteriormente, de igual forma que hemos hecho con el generador de señales, debemos iniciar el osciloscopio mediante la función *Scope1.run()* y realizar una serie de cálculos iniciales, para ajustar los valores necesarios para observar bien la señal recibida en el osciloscopio.

Previo a entrar en el código de programación, vamos a mencionar brevemente los distintos valores que pueden hacer que la señal no se observe como es debido en el osciloscopio, si estos no están correctamente ajustados.

Estos valores que debemos ajustar de manera correcta son: la base de tiempos para el osciloscopio (Base), el offset y el rango (Range) para cada canal.

Son los parámetros que observamos en la parte derecha del osciloscopio, como vemos en la siguiente imagen:

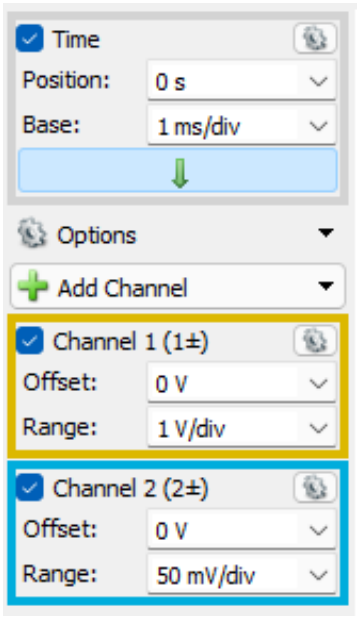


Figura 22. Captura de pantalla de parámetros a ajustar en osciloscopio.

La base de tiempos es el valor de segundos por división en la pantalla del osciloscopio, es un factor de escala. Por ejemplo, si el valor al que está configurado es 1 ms/div, cada división horizontal del osciloscopio representa 1 ms, ajustando el valor de la base de tiempo podemos visualizar más o menos intervalos de tiempo. Hay que tener en cuenta que conforme aumentamos la frecuencia, la base de tiempos debe ir reduciéndose para observar la señal de forma correcta, por lo tanto, en nuestro código tenemos la base de tiempos directamente relacionada a la frecuencia.

El offset nos permite controlar la señal en el eje vertical, hasta el punto de la pantalla del osciloscopio donde nos interese, normalmente se busca que esté centrada. El offset es la diferencia de voltaje entre el eje del osciloscopio y la tensión de la señal.

El rango de tensión (Range) nos permite controlar la escala del eje vertical de la pantalla del osciloscopio, al igual que ocurría en la base de tiempos, los valores voltios/div son factores de escala. Por ejemplo, si se configura como 2 V/div significa que cada una de las diez divisiones del osciloscopio representa 2 Voltios y toda la pantalla puede mostrar hasta 20 V.

Para ajustar los valores del osciloscopio que acabamos de mencionar, emplearemos por un lado el valor de la tensión pico-pico para ajustar el Range y, por otro lado, calcularemos el valor medio para ajustar el offset. Para conseguir los valores iniciales necesarios previos a entrar al bucle, se utilizan las siguientes líneas de código:

```
//CALCULO INICIAL (busco offset inicial y range inicial)
Scope1.Time.Base.text = '1000';

Scope1.Channel1.Offset.text = '0 V';
Scope1.Channel1.Range.text = '1 V/div';

wait(0.2) //NECESARIO PARA CORRECTO FUNCIONAMIENTO
var PicoPicoCh2 = Scope1.Channel2.measure('Peak2Peak');

//Para cálculo del offset
var high = Scope1.Channel2.measure('High');      //valor máximo
var low = Scope1.Channel2.measure('Low');        //valor mínimo
var amp = Scope1.Channel2.measure('Amplitude'); //valor amplitud

var offset1 = - ( (high + low)/2);    //Offset para señal centrada en 0
var offset = offset1 + amp ;          //Valor OFFSET para señal sobre 0
```

Figura 23. Captura parte código cálculo pico-pico y offset.

Podemos observar cómo para el canal 1 se definen valores iniciales de offset y de Range estáticos, ya que al generar para todas las mediciones una señal senoidal de 3,3V esos valores no variarán a lo largo del código.

Una vez ajustados estos parámetros para el canal 1, mediante *wait(0.2)* se añade una pequeña parada para que en la siguiente línea se calcule de forma correcta la tensión pico-pico del canal 2, si no se añadiera esa pausa en el código, la medición de la tensión pico-pico sería errónea ya que al osciloscopio no se le ha dado margen a ajustar los valores definidos en las líneas anteriores. El valor de la tensión pico-pico se almacena en una variable que posteriormente dentro del bucle for, se empleará para ajustar el Range del canal 2.

Por último, se calcula de la misma manera el valor máximo, el valor mínimo y la amplitud de la señal en el canal 2, con los valores obtenidos se realiza un simple cálculo del valor medio para corregir el offset que podamos tener.

2.5.3 Bucle for

En este punto llegamos a la parte principal del código, donde vamos a realizar las distintas mediciones en cada una de las frecuencias y exportar dichos datos.

En primer lugar, se define el bucle for que inicia en 0 y se recorrerá tantas veces como frecuencias a medir tengamos en el array que hemos definido anteriormente. Mediante la siguiente sentencia ajustamos el valor de la frecuencia en el generador de señal: *Wavegen1.Channel1.Simple.Frequency.text = frequency;*

Para facilitar la toma de medidas se añade código para imprimir por pantalla la frecuencia a la que se está midiendo, es decir, cada vez que se recorra el bucle aparecerá por pantalla para que frecuencia se está midiendo. Esto lo podemos observar en la siguiente parte del código:

```
//Bucle para cambiar la frecuencia y realizar medidas
for(var j = 0 ; j < n ; j++) {
    var frequency = frecuencias[j];
    Wavegen1.Channel1.Simple.Frequency.text = frequency;//Cambiar frec
    print("Distancia:  ",DISTANCIA,"cm  ->","Frecuencia:  ",frequency,"
KHz");  //Imprime por pantalla la distancia y la frecuencia una a una
```

Figura 24. Captura parte código inicio bucle y ajuste de frecuencia.

Como hemos comentado anteriormente, la base de tiempos del osciloscopio debe ir cambiando a medida que varían las frecuencias y como se ha indicado, en nuestro caso van directamente relacionadas.

Para ajustar la base de tiempos se crea un array con los valores ideales para cada una de las frecuencias, de manera que conforme vamos recorriendo el array de frecuencias también vamos a recorrer este array y ajustando el valor en el osciloscopio.

```
//Array con Bases de tiempos
var timeBase =[1000,500,200,100,50,20,10,5]; //en us/div
var base = timeBase[j];
Scope1.Time.Base.text = base;  //cambiar base de tiempos del osciloscopio
```

Figura 25. Captura parte código ajuste base de tiempos.

Una vez tenemos la base de tiempos, nos faltaría ajustar el Range y el offset del Canal 2 para visualizar la señal de manera correcta en el osciloscopio.

Como hemos introducido anteriormente, empleamos el valor de la tensión pico-pico para calcular el Range, por tanto, debemos medir dicho valor para cada frecuencia.

Es importante tener en cuenta que el código almacena el valor de la tensión pico-pico en una variable, pero esa variable no actualiza su valor hasta la siguiente medición, es decir, no se puede emplear el valor actual si no que debemos predecir un valor a futuro, mediante una interpretación del filtro de Kalman que hemos explicado en 2.3.6 Filtro Kalman.

Conociendo la tendencia de caída de la tensión pico-pico al aumentar la frecuencia, podemos suponer mediante el valor almacenado en la variable *PicoPicoCh2* obtenido en la medición anterior, cual es el valor del Range para visualizar de manera correcta la señal en el osciloscopio.

Para esto, creamos intervalos mediante sentencias if, como podemos observar en la siguiente imagen:

```
//Ajustar Canal 2 (Range)
    //Medida Tensión Pico-Pico
var PicoPicoCh2 = Scope1.Channel2.measure('Peak2Peak');

    //Range en función tensión pico-pico --FILTRO  Kalman--
    if((PicoPicoCh2 <= 0.004)){
        Scope1.Channel2.Range.text= '1 mV/div';
    }
    if((PicoPicoCh2 > 0.004) && (PicoPicoCh2 <= 0.011)){
        Scope1.Channel2.Range.text= '2 mV/div';
    }
    if((PicoPicoCh2 > 0.011) && (PicoPicoCh2 <= 0.025)){
        Scope1.Channel2.Range.text= '5 mV/div';
    }
    if((PicoPicoCh2 > 0.025) && (PicoPicoCh2 <= 0.050)){
        Scope1.Channel2.Range.text= '10 mV/div';
    }
    if((PicoPicoCh2 > 0.050) && (PicoPicoCh2 <= 0.08)){
        Scope1.Channel2.Range.text= '20 mV/div';
    }
    if(PicoPicoCh2 > 0.08 ){
        Scope1.Channel2.Range.text= '50 mV/div';
    }
```

Figura 26. Captura parte código ajuste Range en función tensión pico-pico.

Dependiendo del intervalo en el que esté el valor de la tensión pico-pico (*PicoPicoCh2*) medido en la anterior iteración, se ajustará el valor del Range del canal 2.

Por otro lado, el valor del offset del canal 2 se ajusta de la misma manera que se ha hecho antes del bucle for, midiendo el valor máximo, el mínimo y la amplitud, y a partir de ellos calcular el valor medio y sumándolo a la amplitud, obteniendo la tensión de offset necesaria para centrar la señal en la pantalla del osciloscopio.

```
//Canal 2 (Offset)
var high = Scope1.Channel2.measure('High');      //valor máximo
var low = Scope1.Channel2.measure('Low');        //valor mínimo
var amp = Scope1.Channel2.measure('Amplitude'); //valor amplitud

var offset1 = - ( (high + low)/2);     //Offset señal centrada en 0
var offset = offset1 + amp ;           // OFFSET señal sobre 0

Scope1.Channel2.Offset.value = offset; //Establecer valor OFFSET
```

Figura 27. Captura parte código ajuste offset.

2.5.4 Exportación de las medidas realizadas

Por último, una parte fundamental de este trabajo es poder exportar las distintas medidas que se han realizado, para ello en la parte final del bucle for tenemos las líneas que nos van a permitir exportar los archivos que necesitamos.

Como hemos mencionado al principio del TFG, los ficheros necesarios son: por un lado, los archivos en formato .csv y en .png del osciloscopio y del histograma y un archivo .csv con las medidas de ambos canales del osciloscopio. Para esto, hacemos uso de la función *Scope.Export()*. En el caso de los datos del osciloscopio se hace de la siguiente manera:

```
//EXPORTAR datos OSCILOSCOPIO .csv y .png
Scope1.wait()
    //Osciloscopio en CSV
Scope1.Export("~/Desktop/DescargaWaveForms/osciloscopio          celula
"+CELULA+" A"+ANODO+" led alta luminosidad "+COLOR+" "+TAMAÑO+" mm
"+VOLTAJE+" " +frecuencias[j]+ " KHz " +DISTANCIA+ " cm.csv")
    //Osciloscopio en PNG
Scope1.Export("~/Desktop/DescargaWaveForms/osciloscopio          celula
"+CELULA+" A"+ANODO+" led alta luminosidad "+COLOR+" "+TAMAÑO+" mm
"+VOLTAJE+" " +frecuencias[j]+ " KHz " +DISTANCIA+ " cm.png")
```

Figura 28. Captura parte código exportar datos osciloscopio.

Podemos observar cómo primero debemos indicar la ruta donde queremos que se nos guarde el archivo, en este caso ponemos por defecto el escritorio del ordenador en una carpeta llamada "DescargaWF". Después, debemos escribir el nombre que queremos que tenga el archivo, como necesitamos identificar y ordenar cada medida usamos las variables que hemos declarado al principio del

script y también el valor de la frecuencia que se está midiendo, para nombrar a cada archivo.

Para exportar los datos del histograma y de las medidas se realiza de la misma forma, indicando al final de cada sentencia el archivo que se quiere descargar: "Histogram" o "Measurements".

```
//Exportar datos HISTOGRAMA .csv y .png
    //Histograma en CSV
Scope1.Export("~/Desktop/DescargaWaveForms/histograma celula "+CELULA+"
A"+ANODO+" led alta luminosidad "+COLOR+" "+TAMAÑO+" mm "+VOLTAJE+" "
+frecuencias[j]+ " KHz " +DISTANCIA+ " cm.csv", "Histogram")
    //Histograma en PNG
Scope1.Export("~/Desktop/DescargaWaveForms/histograma celula "+CELULA+"
A"+ANODO+" led alta luminosidad "+COLOR+" "+TAMAÑO+" mm "+VOLTAJE+" "
+frecuencias[j]+ " KHz " +DISTANCIA+ " cm.png", "Histogram")

//Exportar datos MEDIDAS canal1 y canal2 .csv
    /**HAY QUE SELECCIONAR LAS MEDIDAS QUE NECESITAMOS EN EL OSCILOSCOPIO**/
Scope1.Export("~/Desktop/DescargaWaveForms/medidas  celula  "+CELULA+"
A"+ANODO+" led alta luminosidad "+COLOR+" "+TAMAÑO+" mm "+VOLTAJE+" "
+frecuencias[j]+ " KHz " +DISTANCIA+ " cm.csv", "Measurements")
```

Figura 29. Captura parte código exportar datos histograma y medidas.

Cabe destacar que el software WaveForms no permite mediante script abrir o cerrar ventanas en el osciloscopio, por lo que es necesario previo a ejecutar el script abrir la ventana del histograma en el osciloscopio para que se exporten los datos. De igual forma ocurre con las medidas que se quieran exportar, al iniciar el osciloscopio debemos indicar las medidas del canal 1 y canal 2 que queremos que posteriormente se exporten en el archivo de datos .csv. No supone un gran inconveniente ya que únicamente se debe realizar al iniciar el programa, una vez configurado, el script funciona de manera correcta.

Por tanto, mediante el anterior código que hemos ido explicando paso a paso somos capaces de resolver el objetivo principal de este proyecto, pudiendo exportar todos los archivos de mediciones a distintas frecuencias de una manera eficiente.

2.5.4 Diagrama de flujo del script

Por último, todo el desarrollo de la programación del script se puede entender paso a paso de una forma más visual y resumida empleando un diagrama de flujo de su funcionamiento. Para ello hemos empleado la herramienta

diagrams.net [34] que es un software de dibujo de gráficos multiplataforma gratuito de acceso online (https://app.diagrams.net), con una interfaz muy sencilla, como podemos observar en la siguiente imagen.

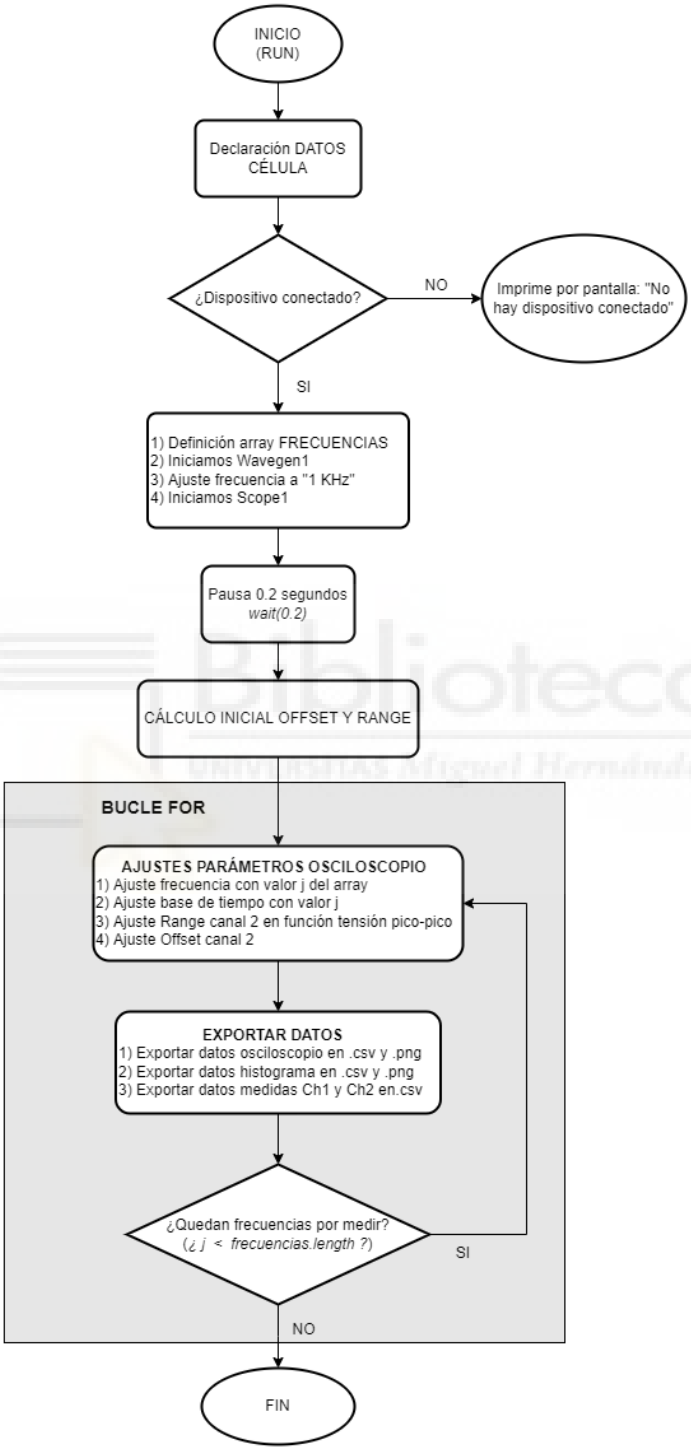


Figura 30. Diagrama de flujo script automatización toma de datos WaveForms.

El código completo en WaveForms está incluido en 5.1 ANEXO 1: CÓDIGO COMPLETO WAVEFORMS DE AUTOMATIZACIÓN TOMA DE MEDIDAS.

2.6 REPRESENTACIÓN DIAGRAMA DE OJO

Una vez obtenidos los archivos con las medidas realizadas en el osciloscopio, es interesante poder realizar una representación del diagrama de ojo, que como hemos explicado anteriormente nos permite de una manera sencilla y visual observar el comportamiento de las señales.

Para obtener dicha representación empleamos Matlab, que nos da la posibilidad de trabajar con los archivos de datos .csv que hemos exportado del osciloscopio.

En primer lugar, en el script de Matlab debemos leer los archivos de los datos de osciloscopio, ya que para realizar la representación vamos a emplear los valores de tensión del canal. Creamos un vector (*files*) con los archivos de datos de osciloscopio que tengamos en la carpeta, más adelante recorreremos este vector para ir representando el diagrama de ojo de cada archivo.

Para evitar posibles errores, añadimos una condición para carpetas que no contengan archivos de datos de osciloscopio, se nos indicaría por la ventana de comandos.

```
%% Leer archivos csv de carpeta
files = dir('o*.csv');
table_data = cell(1, numel(files));
if( numel(files)==0)
    error= 'Error: No hay archivos ".csv" para representar en la
carpeta.';
    disp(error)
else
end
```

Figura 31. Captura parte código MATLAB obtener archivos datos osciloscopio.

Antes de entrar en el bucle que recorrerá el vector de archivos y con el que representaremos los diagramas de ojo, para archivar los resultados vamos a crear una carpeta, haciendo que al ejecutar el script mediante la función *uigetdir()* nos solicite donde queremos que se cree dicha carpeta, haciendo que sea más sencillo de manejar el código.

```
%% Crear carpeta para guardar resultados
currentdirectory = pwd;
[filepath,name,ext] = fileparts(pwd);
nombreCarpetaMedidas = name;

newFolder1 = strcat("Diagramas de ojo ", nombreCarpetaMedidas);
newFolder = convertStringsToChars(newFolder1);

%Te pide seleccionar DONDE crear la carpeta para descargar resultados
route = uigetdir('*.*');

mkdir(fullfile(route,newFolder));
```

Figura 32. Captura parte código MATLAB crear carpeta resultados.

Observamos como todas las carpetas de resultados se nombrarán como "Diagramas de ojo " seguido del nombre de la carpeta donde están los archivos de medidas, de esta forma mantenemos los resultados correctamente ordenados. Esto lo hacemos quedándonos con la parte del nombre de la ruta del directorio en que estamos, empleando la función *fileparts().*

Una vez creada la carpeta donde se van a exportar los resultados de las representaciones, entramos en el bucle donde vamos a recorrer todos los archivos de datos y a representar el diagrama de cada uno de ellos.

Es un bucle for que irá recorriendo el vector (*files*) que hemos creado con anterioridad al leer los archivos de datos de la carpeta.

Para la representación del diagrama de ojo necesitamos los valores de tensión, para ello en el bucle debemos obtener los datos de las tensiones de cada archivo .csv, estos valores corresponden a las columnas 2 y 3 del archivo exportado del osciloscopio.

Hay que tener en cuenta que para poder operar con los valores es necesario pasar la tabla a vector. Esto lo hacemos mediante las siguientes líneas de código:

```
%% Datos obtenidos osciloscopio
%Obtenemos datos
v1 = table_data{1, i}  (:,2);     %Canal 1
v2 = table_data{1, i}  (:,3);     %Canal 2


%Pasar tabla a vector (para poder operar con los valores)
A = table2array(v1);    %Canal 1
B = table2array(v2);    %Canal 2
```

Figura 33. Captura parte código MATLAB obtener valor de tensiones.

Con el código anterior tenemos almacenados en los vectores A y B los valores de las tensiones del canal 1 y el canal 2, respectivamente.

Para una correcta representación del diagrama de ojo, debemos obtener el valor del periodo T de la señal para ajustar el eje horizontal de la representación, en función de este valor.

Obtenemos el valor de T a partir del valor de la frecuencia, que la conseguimos a partir del nombre de cada archivo, que como hemos explicado en el anterior apartado viene declarado automáticamente desde el script del osciloscopio.

Una vez extraído el valor de la frecuencia del nombre del archivo, es importante tener en cuenta que al igual que sucedía con los valores de tensión, hay que pasar el valor de formato *string* a *double* para poder operar con él.

```
%% Obtener   frecuencia
parte1 = strfind(filename, 'KHz');
parte2 = strfind(filename, 'V');
strFrecu = filename(parte2 + 1: parte1-1);

frecuVal=str2double(strFrecu);  %Valor de la frecuencia sin unidades
frecu = frecuVal*1000;          %Valor de la frecuencia en Hz

 %% Definiciones
T= (1/frecu) ;  %Periodo en funcion de la frecuencia
```

Figura 34. Captura parte código MATLAB obtener valor de frecuencia.

Al contar con el valor del periodo T, haciendo uso de la función *eyediagram(x,n,period,offset)* de Matlab, donde *x* es la señal de entrada que queremos representar, *n* es el número de muestras por traza, *period* es el periodo por traza y *offset* es el valor de tensión offset que queramos en la representación.

En nuestro caso, la señal de entrada que nos interesa representar es el vector de valores de tensión del canal 2, con el número de muestras igual a las que se obtienen del osciloscopio y el periodo es el que hemos obtenido el valor anteriormente.

```
%% Representación diagrama de ojo
eyediagram(B, 16384, T, offset);  %Canal 2
% eyediagram(A, 16384, T, offset); %Canal 1
```

Figura 35. Captura parte código MATLAB representación diagrama de ojo.

Una vez representados los diagramas de ojo, es necesario que seamos capaces de exportar los resultados para un futuro análisis o posibles consultas. Para ello guardamos cada una de las representaciones en formato .png en la carpeta que hemos creado al principio del código, con el nombre "Diagrama de Ojo-" seguido del nombre del archivo de datos que se está representando, de esta manera archivamos de una forma ordenada los resultados.

```
%% Descargar png diagrama de ojo
carp = strcat(route,'\', newFolder);
saveas(gca,fullfile(carp, strcat('Diagrama de Ojo- ',filename,'.png')));
```

Figura 36. Captura parte código MATLAB exportación resultados.

Con la exportación de las representaciones se cierra el bucle for, con el que hemos recorrido el vector con todos los archivos de datos del osciloscopio.

2.6.1 Diagrama de flujo del script

Por último, todo el desarrollo de la programación del script se puede entender paso a paso de una forma más visual y resumida empleando un diagrama de flujo de su funcionamiento.

Para ello, al igual que en el anterior script, hemos empleado la herramienta *diagrams.net* [34], obteniendo el siguiente diagrama de flujo:

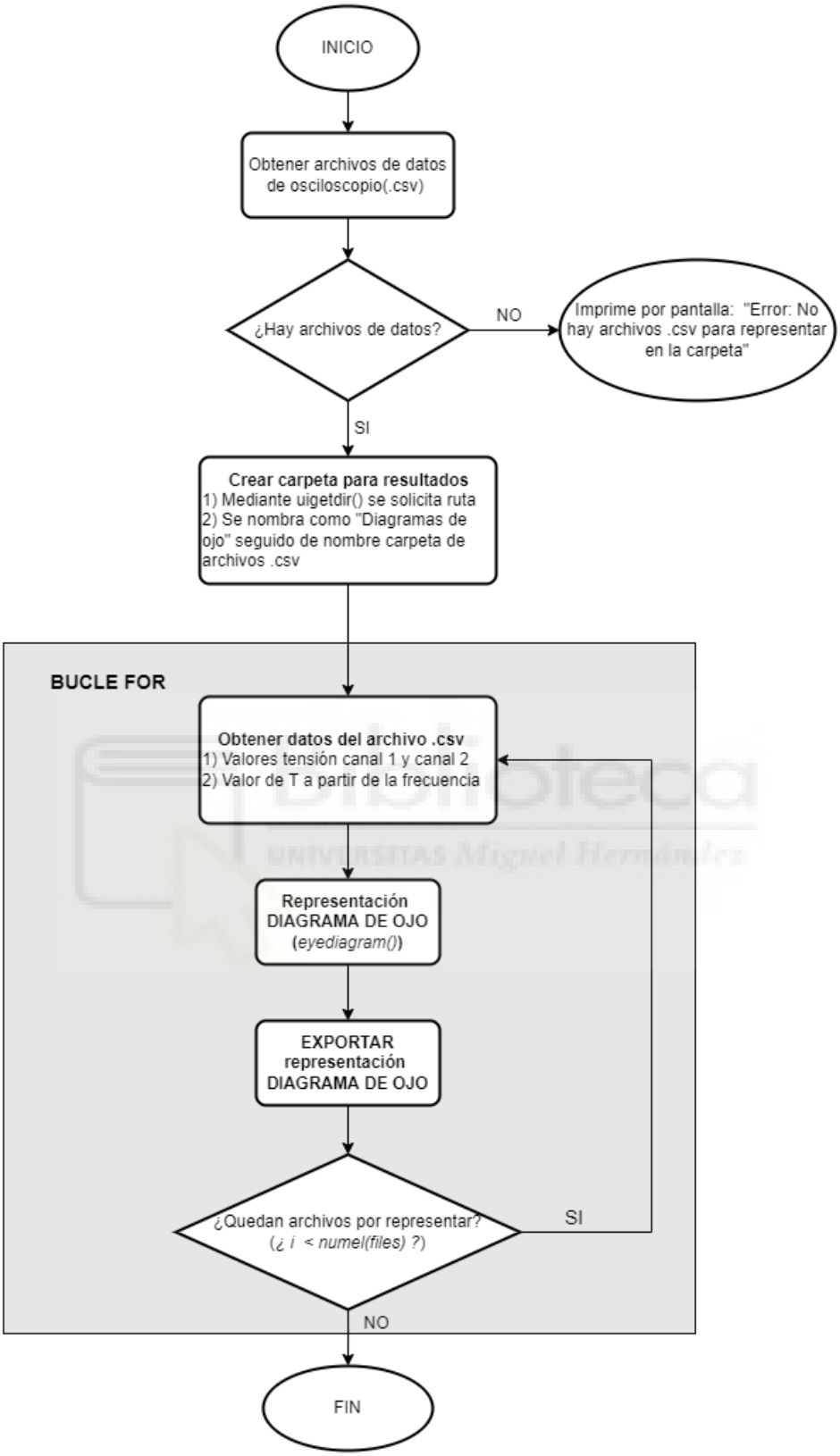


Figura 37. Diagrama de flujo script representación diagrama de ojo en Matlab.

El código completo en MATLAB está incluido en 5.2 ANEXO 2: CÓDIGO COMPLETO MATLAB REPRESENTACIÓN DIAGRAMA DE OJO.

2.7 PROGRAMACIÓN CÁLCULO SNR, Q y BER

Como hemos mencionado anteriormente otro de los objetivos de este TFG es calcular una serie de parámetros de calidad de la señal, a partir de los datos que se obtienen del osciloscopio.

Para este cálculo se emplea el software Matlab que como en el caso de la representación del diagrama de ojo, nos permite trabajar de una forma cómoda con los archivos de datos.

Para obtener los valores emplearemos los conceptos explicados en los fundamentos teóricos del apartado 2.3 de este documento, más en concreto en los apartados 2.3.2 Relación señal a ruido (SNR), 2.3.3 Factor de calidad Q y 2.3.4 Tasa de error de bit (BER).

2.7.1 Obtención vector con archivos de medidas

En primer lugar, de igual forma que sucedía en el script del diagrama de ojo, debemos leer los archivos de datos en formato .csv que tenemos en la carpeta. Esta vez buscamos los archivos del histograma del osciloscopio ya que serán los ficheros donde tenemos la información necesaria para el cálculo.

El histograma es una representación gráfica de la distribución del voltaje, nos muestra el número de veces que cierta señal ha tenido ese valor de tensión expresado en porcentajes. Por ejemplo, la siguiente imagen es el histograma de una señal senoidal de 3,3 V de amplitud:

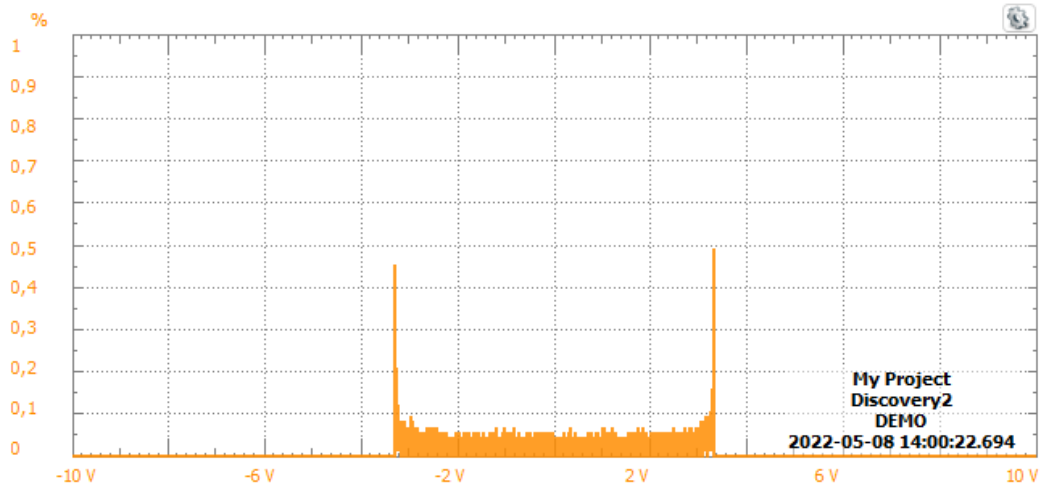


Figura 38. Histograma ejemplo señal senoidal 3,3 V.

Podemos ver como al ser una señal senoidal en la representación del histograma tenemos dos picos tanto en -3,3 V y en 3,3 V. En el eje horizontal tenemos los distintos niveles de tensión y en el eje vertical en porcentaje la cantidad de veces que se tiene cada voltaje.

Para el caso de un histograma de una medición real realizada a lo largo de este proyecto, observamos la siguiente imagen:

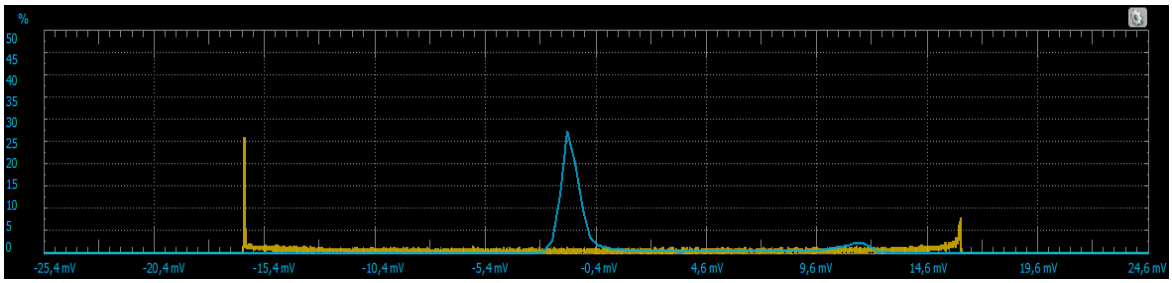


Figura 39. Histograma ejemplo medición célula fotovoltaica.

Donde podemos ver como en el canal 1 (amarillo) tenemos una señal senoidal con la que alimentamos el LED a 3,3 V y en el canal 2 (azul) tenemos la respuesta de la célula fotovoltaica en el ánodo que se estaba midiendo, donde podemos distinguir claramente como la gran mayoría de la tensión se sitúa en torno a 0 V al tratarse de una señal rectificada. Pero también se puede apreciar otro pico sobre los 10 mV, cada uno de estos máximos corresponde al 0 y al 1 de la señal binaria y son los valores que buscaremos y necesitamos para calcular el factor Q, SNR y la BER. Todo esto lo analizaremos con más detalle en el siguiente apartado del trabajo (3. RESULTADOS).

Una vez explicado brevemente el concepto de histograma retomamos el script de Matlab, como hemos comentado antes debemos buscar en la carpeta los archivos de histograma, creando un vector que más adelante recorreremos en el bucle.

```
%% Leer archivos .csv de carpeta y error si no hay
files = dir('h*.csv');
table_data = cell(1, numel(files));
if( numel(files)==0)
    error= 'Error: No hay archivos ".csv" para representar en la carpeta.';
    disp(error)
else
end
```

Figura 40. Captura parte código MATLAB obtener archivos datos histograma.

2.7.2 Inicio bucle for y cálculo factor Q

Una vez contamos con el vector con los archivos, entramos en el bucle for que es la parte del código que vamos a repetir para cada uno de los archivos de ese vector creado.

Hay que tener en cuenta que el archivo de histograma cuenta con cuatro columnas: voltaje del canal 1, porcentaje del canal 1, voltaje del canal 2 y porcentaje del canal 2. Estas cuatro columnas son necesarias para realizar los cálculos, por lo que lo primero que debemos realizar en el bucle for, es pasar cada una de las columnas de la tabla de datos a vectores para poder operar con ellos.

```
for i=1:numel(files)
    filename = files(i).name;
    table_data{i} = readtable(filename);

%% Datos obtenidos por columnas del archivo
  %CANAL 1
c1V= table_data{1, i} (:,1);    %Tensiones V canal 1
c1P= table_data{1, i} (:,2);    %Porcentajes canal 1
  %CANAL 2
c2V = table_data{1, i} (:,3);     %Tensiones V canal 2
c2P = table_data{1, i} (:,4);     %Porcentajes canal 2

  %Pasar tabla a vector, para poder trabajar con los valores
vectorc1V = table2array(c1V);    %Tension canal 1
vectorc1P = table2array(c1P);    %Porcentaje canal 1
vectorc2V = table2array(c2V);    %Tension canal 2
vectorc2P = table2array(c2P);    %Porcentaje canal 2
```

Figura 41. Captura parte código MATLAB obtener vectores datos histograma.

Una vez tenemos cada una de las columnas en vectores, debemos fijarnos en que queremos calcular en primer lugar el factor de calidad Q y como hemos explicado en la ecuación 14 depende de V1, V0, σ1 y σ0, por tanto, debemos calcular dichos valores.

En primer lugar, supondremos un código para un caso ideal, en el que los valores de tensión en el nivel 0 (V0) y nivel 1 (V1) se obtienen de manera correcta. Pero más adelante trabajaremos con los distintos casos que llevan a error y como conseguir filtrar dichos errores.

2.7.2.1 Obtención valores máximos para nivel alto y nivel bajo

Para obtener los valores máximos del histograma, es decir, los picos que corresponde al 0 y 1 de la señal, empleamos la función *findpeaks()* de Matlab. Nos interesa esos valores de máximos locales en el vector de porcentaje de canal 2, ya que es el eje vertical en el histograma.

Es importante tener en cuenta que con la función *findpeaks()* del vector porcentajes, obtenemos los máximos locales y nos devuelve los valores de los porcentajes, pero a nosotros nos interesan los voltajes correspondientes a esos porcentajes, es decir, debemos relacionar el vector de porcentajes con el vector de voltajes. Para relacionar ambos vectores, empleamos el número de celda que es igual en ambos vectores, en el código lo identificamos como locV0 y locV1.

```
%% Obtener picos máximos porcentajes Canal 2

[valmaximosc2P , locmaximosc2P]=findpeaks(vectorc2P,'SortStr','descend');
```

Figura 42. Captura parte código MATLAB obtener máximos histograma.

En primer lugar, lo hacemos para el nivel 0 (V0) que resulta más sencillo al corresponder al primer máximo que se haya encontrado, ya que al ser una señal rectificada el valor de voltaje más repetido es siempre en torno a 0V. Identificamos en tres variables: el valor de V0, el porcentaje al que corresponde ese voltaje y el número de celda que ocupa en el vector.

```
%Valor máximo en el 0 y su porcentaje
V0 = vectorc2V(locmaximosc2P(1,1) , 1);

V0porcent = valmaximosc2P(1,1);
locV0 = find(vectorc2V == V0);
```

Figura 43. Captura parte código MATLAB obtener V0.

Para el cálculo de V1 se sigue un razonamiento un poco diferente, ya que para el nivel 1 tenemos el máximo del histograma con mayor voltaje. Por tanto, se busca en el vector de tensiones del canal 2 (vectorc2V) cual es el máximo voltaje para el mayor porcentaje, asignando el resultado a V1. Al igual que con V0 se identifican tres variables que serán necesarias más adelante: el valor de V1 obtenido, el porcentaje y el número de celda.

```
%Cálculos para obtener el valor de v1
valoresVOLTmaximosc2P = vectorc2V(locmaximosc2P(:,1) , 1);

%Valor máximo en el 1 y su porcentaje
V1 = max(valoresVOLTmaximosc2P);

locV1 = find(vectorc2V == V1);
V1porcent = vectorc2P(locV1   , 1);
```

Figura 44. Captura parte código MATLAB obtener V1.

2.7.2.2 Obtención valores de σ1 y σ0

Una vez obtenidos los valores de V0 y V1 debemos calcular σ1 y σ0, con lo que tendríamos todas las variables necesarias para el cálculo del factor de calidad Q. Para obtener el valor de σ1 y σ0 debemos conseguir el valor que se obtiene del voltaje, a nivel alto para σ1 y a nivel bajo para σ0, con respecto al 60% del máximo, que en nuestro caso son los valores que hemos obtenido anteriormente V0 y V1. Esto guarda relación directa con lo que hemos explicado en 2.3.3 Factor de calidad Q en el apartado de fundamentos teóricos del TFG y más en concreto lo podemos observar en la Figura 10.

Para obtener el voltaje que corresponde al 60% de los máximos, debemos tener en cuenta que nuestro vector de datos tiene valores de mediciones que se han hecho en el osciloscopio, por lo que no vamos a encontrar directamente el valor exacto del voltaje deseado, para ello empleamos una interpolación de datos con la que buscaremos mediante el valor más cercano por arriba y el más cercano por abajo un valor más exacto.

En primer lugar, definimos unas variables con los valores de voltaje a 60% ideales, pero sabiendo que no vamos a encontrar el mismo valor en el vector, sobre estos valores de las variables son los que buscaremos los más cercanos por arriba y por abajo:

```
%% CÁLCULOS PARA SIGMA0 Y SIGMA1

%Valor porcentajes a 60% del máximo
V0porcent60 = 0.6 * V0porcent;
V1porcent60 = 0.6 * V1porcent;
```

Figura 45. Captura parte código MATLAB obtener voltajes al 60%.

Comenzamos por el nivel bajo, debemos buscar en primer lugar en el vector de porcentajes del canal 2, el valor más cercano a *V0porcent60* para ello empleamos la función *knnsearch()* de Matlab.

```
%Valor para 0
%Valores más cercanos al buscado
calc1 = knnsearch(vectorc2P, V0porcent60, 'k', 500);
vecinosV0porcent60 = vectorc2P(  calc1(1, find(calc1 >= locV0)) ,1) ;
```

Figura 46. Captura parte código MATLAB obtener valores cercanos.

De los posibles valores que obtengamos nos interesa quedarnos con los que estén a la derecha de V0, ya que podríamos calcular los valores de σ0 tanto para la izquierda de V0 como para la derecha, en nuestro caso trabajaremos a la derecha del máximo.

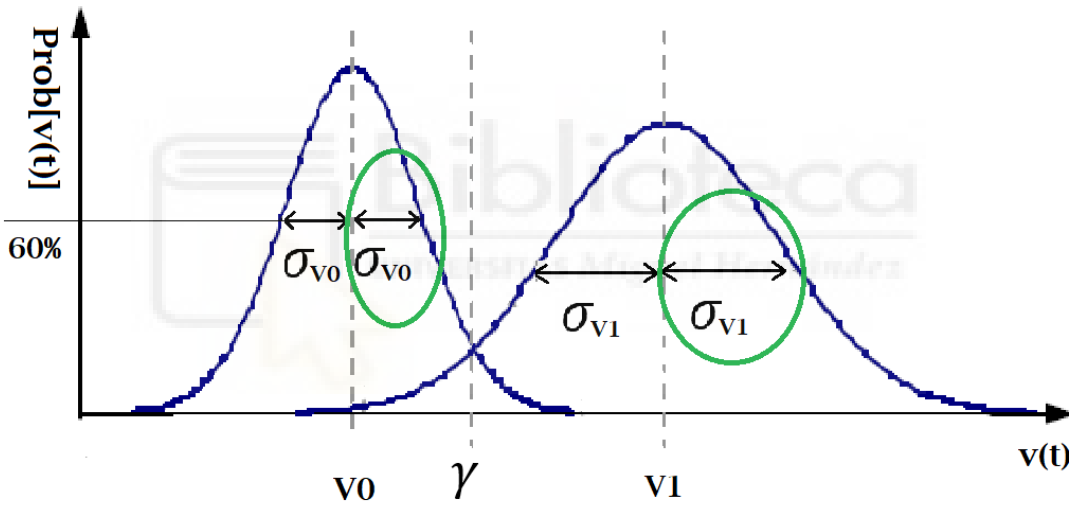


Figura 47. Representación σ1 y σ0 en el cálculo de Q.

Del vector de valores cercanos al voltaje al 60%, debemos obtener el más cercano por arriba y el más cercano por abajo para poder hacer la interpolación, para ello obtenemos por un lado el que es mayor o igual y su posición en el vector y de igual forma para el menor o igual.

```
vecinosV0porcent60ARRIBA=vecinosV0porcent60(find(vecinosV0porcent60  >=
V0porcent60,1),1);
locV0porcent60ARRIBA = min(find(vectorc2P == vecinosV0porcent60ARRIBA));
valorVOLTvecinosV0ARRIBA = vectorc2V(locV0porcent60ARRIBA , 1);
vecinosV0porcent60ABAJO = vecinosV0porcent60(find( vecinosV0porcent60 <
V0porcent60, 1 ), 1);
locV0porcent60ABAJO = min(find((vectorc2P == vecinosV0porcent60ABAJO)));
valorVOLTvecinosV0ABAJO = vectorc2V(locV0porcent60ABAJO , 1);
```

Figura 48. Captura parte código MATLAB obtener valores cercanos por arriba y por abajo.

Una vez tenemos el valor del voltaje y la posición en el vector, tanto del valor por arriba como por abajo, podemos realizar mediante la función *interp1()* una interpolación lineal que nos devuelve los valores buscados. Esta función necesita los valores de las columnas de porcentajes del histograma (eje vertical) y de la columna de voltaje (eje horizontal) e indicar el valor que se quiere interpolar en nuestro caso *V0porcent60.*

```
interpolacionV0a60=interp1([vecinosV0porcent60ABAJO vecinosV0porcent60ARRIBA],
[valorVOLTvecinosV0ABAJO valorVOLTvecinosV0ARRIBA], V0porcent60);
```

Figura 49. Captura parte código MATLAB interpolación valores V0 al 60%.

Una vez obtenido el valor del voltaje que corresponde al 60% del máximo podemos obtener el valor de σ0 mediante la diferencia de V0 y el que hemos obtenido en la interpolación.

```
%VALORES SIGMA
sigmaV0 = ( V0 - interpolacionV0a60 );
sigmaV1 = ( V1 - interpolacionV1a60 );
```

Figura 50. Captura parte código MATLAB valor σ0.

De igual forma que hemos hecho para el nivel bajo V0 hacemos con el nivel alto V1, obteniendo en primer lugar los valores más cercanos a la variable que hemos calculado previamente V1porcent60, a partir de ahí seleccionamos un valor cercano por arriba y otro por abajo, para acabar realizando la interpolación y obtener el valor de σ1 de la misma manera que hemos explicado para σ0.

Una vez obtenidos los valores de voltaje del nivel bajo V0, del nivel alto V1 y de σ0 y σ1, mediante la ecuación 14 podemos calcular el valor del factor de calidad Q, expresándolo también en decibelios.

```
%% VALOR Q
Q = abs((V1-V0)./(sigmaV1+sigmaV0));
Qdb = 20*log10(Q);  %valor de Q en dBs
```

Figura 51. Captura parte código MATLAB cálculo factor Q.

2.7.3 Corrección de posibles errores para el cálculo

Hay que tener en cuenta que como hemos comentado, hasta aquí era un script en el que archivo de medidas es ideal y no hay ninguna circunstancia que lleve a error, la realidad es que hay ciertas posibles situaciones que hacen que se obtengan errores, por tanto, debemos ser capaces de filtrar esas situaciones. A continuación, vamos a comentar una a una las circunstancias que se pueden dar a lo largo del anterior script y como se han solucionado.

2.7.3.1 Error falta de máximos en el archivo de histograma

En primer lugar, al principio del código cuando buscamos máximos en el vector de porcentajes (eje vertical) se puede dar el caso que no se obtengan dos máximos al menos, por lo que no podríamos realizar cálculos ya que necesitamos mínimo un máximo para V0 y otro para V1. Esta situación se da en casos de histogramas donde no hay picos claros, donde hasta visualmente resulta difícil distinguir el voltaje del nivel alto del voltaje del nivel bajo. Para solucionar esto vamos a filtrar mediante una sentencia if los casos donde no haya al menos dos máximos, indicando en las variables que falta un máximo escribiendo "Falta max".

```
numeroDeMaximos = length(valmaximosc2P);  %número de máximos en histograma

%Condición: Si no hay al menos dos máximos, no podemos tener V0 y V1
if numeroDeMaximos < 2
        V0='Falta max';
        V1='Falta max';
        Q='Falta max';
        Qdb='Falta max';
        SNR='Falta max';
        BER='Falta max';
else
```

Figura 52. Captura parte código MATLAB error al menos dos máximos.

De esta manera al exportar los resultados en la tabla sabremos que para esa medición en el histograma no se puede obtener V0 y V1.

Esta situación se da la mayoría de los casos para mediciones a frecuencias muy altas donde las señales en el osciloscopio se observan muy distorsionadas y en el histograma no se distinguen los máximos de los niveles de señal.

2.7.3.2 Error falta valor para la interpolación de σ1 y σ0

Por otro lado, otra situación en la que puede ocurrir un error es cuando necesitamos calcular los valores cercanos al voltaje al 60%, ya que se puede dar el caso que el vector de valores cercanos por arriba o cercanos por abajo, *vecinosV0porcent60ARRIBA* y *vecinosV0porcent60ABAJO* respectivamente, no tengan ningún valor. Debido en la mayoría de los casos a situaciones extremas que hacen que únicamente haya un único valor cercano, no pudiendo obtener mediante el script uno por arriba y otro por abajo.

Para solucionar esto, se toma como valor cercano por arriba el único valor cercano que se haya encontrado y para el valor por abajo utilizamos el siguiente que tengamos en el vector. Conociendo la tendencia del histograma sabemos que el siguiente valor será menor que el anterior y por tanto el vecino más cercano por abajo. Con esta solución podríamos calcular la interpolación para el nivel V0 al 60%.

```
if length(vecinosV0porcent60ARRIBA) < 1

   locV0porcent60ARRIBA=min(find(vectorc2P==vecinosV0porcent60ARRIBA));
   valorVOLTvecinosV0ARRIBA = vectorc2V(locV0porcent60ARRIBA  , 1);
   vecinosV0porcent60ARRIBA = vectorc2P( locV0porcent60ARRIBA  ,1);
```

Figura 53. Captura parte código MATLAB situación error al menos un valor cercano por arriba.

```
if length(vecinosV0porcent60ABAJO) < 1

    vecinosV0porcent60ABAJO = vecinosV0porcent60;
    locV0porcent60ABAJO=min(find((vectorc2P==vecinosV0porcent60ABAJO)));
    valorVOLTvecinosV0ABAJO = vectorc2V(locV0porcent60ABAJO + 1  , 1);
    vecinosV0porcent60ABAJO =  vectorc2P(locV0porcent60ABAJO + 1  , 1);
```

Figura 54. Captura parte código MATLAB situación error al menos un valor cercano por abajo.

Para el nivel de señal alto (V1) seguimos el mismo razonamiento anterior, ya que la situación que lleva a error es la misma.

2.7.4 Cálculo SNR y BER

Una vez hemos filtrado los casos en los que podríamos tener errores debemos calcular el resto de las variables siguiendo lo explicado en el apartado 2.3.2 Relación señal-ruido SNR y 2.3.4 Tasa de error de bit (BER).

Para el caso de la SNR, en la ecuación 4 la hemos relacionado directamente con el valor de V1 y σ1, y para el caso de la VER, en la ecuación 15 la relacionamos mediante la función de error complementario con el valor de Q, que ya hemos calculado anteriormente.

```
%% VALOR SNR
SNR = 10*log10(((V1).^2)./((sigmaV1).^2)); %en dB

%% VALOR BER
BER = (1/2)*erfc(Q/(sqrt(2)));
```

Figura 55. Captura parte código MATLAB cálculo fórmula SNR y BER.

2.7.5 Creación y exportación de tabla de resultados

Por último, debemos crear una tabla con los resultados obtenidos para cada archivo que recorre el bucle for. La tabla contendrá 7 columnas: nombre del archivo, nivel bajo V0, nivel alto V1, factor Q, factor Q en dB, SNR en dB y la BER. Para construir la tabla almacenamos cada una de las variables en vectores, que posteriormente serán cada una de las columnas de la tabla y finalmente identificamos cada columna por su nombre.

```
%% Crear tabla con resultados  NOMBRE , V0, V1, Q, Q(dB), SNR(dB), BER
filenameString = string(strName);
vectorSolu(i,1)= filenameString;    %Nombre archivo
vectorSolu(i,2)= V0;                %Valor V0
vectorSolu(i,3)= V1;                %Valor V1
vectorSolu(i,4)= Q;                 %Valor Q
vectorSolu(i,5)= Qdb;               %Valor Q(dB)
vectorSolu(i,6)= SNR;               %Valor SNR(dB)
vectorSolu(i,7)= BER;               %Valor BER

%Cambiamos los array a una tabla y añadimos nombre a cada columna
TablaSolu = array2table(vectorSolu);
TablaSolu.Properties.VariableNames{'vectorSolu1'} = 'Archivo';
TablaSolu.Properties.VariableNames{'vectorSolu2'} = 'V0';
TablaSolu.Properties.VariableNames{'vectorSolu3'} = 'V1';
TablaSolu.Properties.VariableNames{'vectorSolu4'} = 'Q';
TablaSolu.Properties.VariableNames{'vectorSolu5'} = 'Q_dB';
TablaSolu.Properties.VariableNames{'vectorSolu6'} = 'SNR_dB';
TablaSolu.Properties.VariableNames{'vectorSolu7'} = 'BER';
```

Figura 56. Captura parte código MATLAB creación tabla de resultados.

Como es lógico, una vez obtenida la tabla con todos los resultados la exportamos en formato .csv con el nombre "Función Q, SNR y BER-" seguido del nombre de la carpeta de datos que se está midiendo, para tener los resultados ordenados.

```
%% Exportar tabla SNR y Q en formato .csv
writetable(TablaSolu    ,    strcat('Factor    Q,    SNR    y    BER-    ',
nombreCarpetaMedidas,'.csv'));
```

Figura 57. Captura parte código MATLAB exportación tabla de resultados.

2.7.6 Diagrama de flujo script del script

Por último, todo el desarrollo de la programación del script se puede entender paso a paso de una forma más visual y resumida empleando un diagrama de flujo de su funcionamiento.

Para ello, al igual que en los anteriores scripts, hemos empleado la herramienta *diagrams.net* [34], obteniendo el siguiente diagrama de flujo (*siguiente página*):

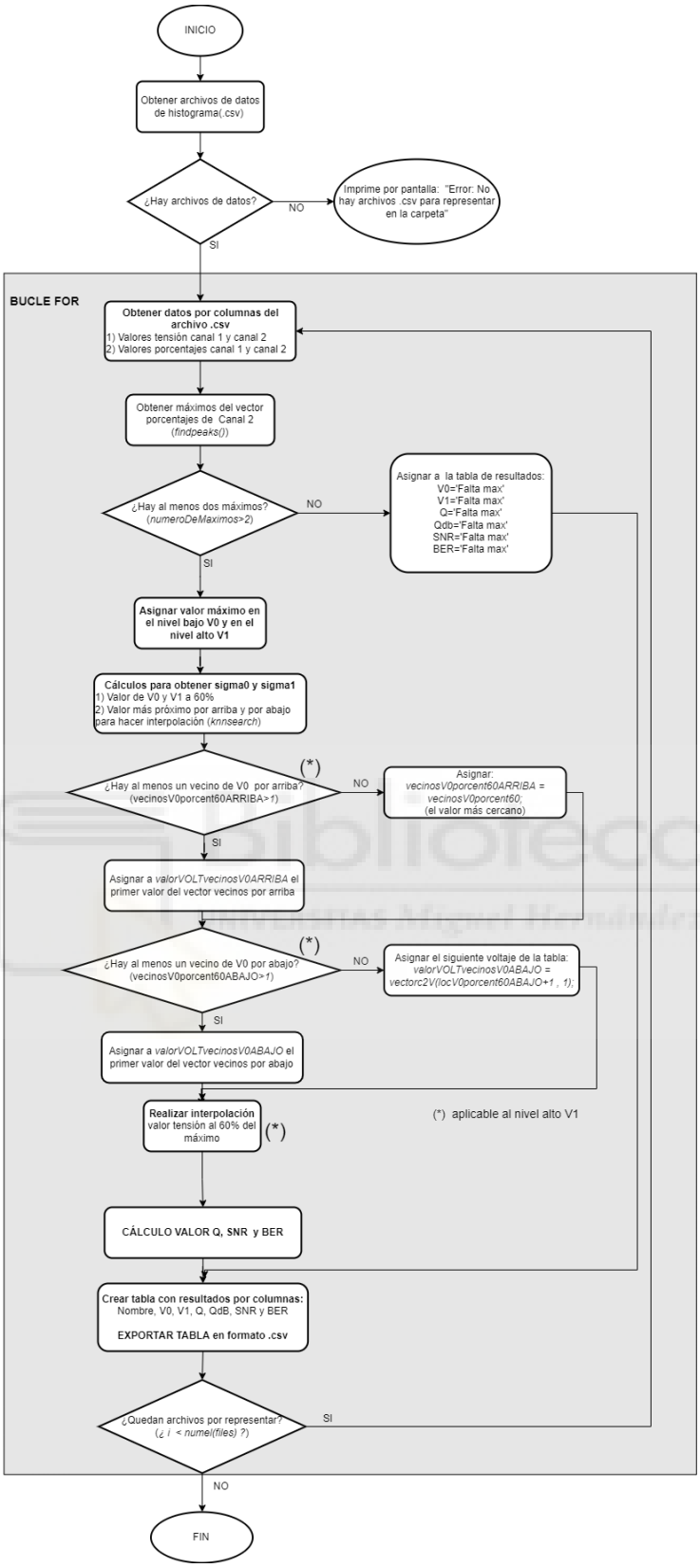


Figura 58. Diagrama de flujo script obtención valor SNR, Q y BER en Matlab.

El código completo en MATLAB está incluido en [*5.3 ANEXO 3: CÓDIGO COMPLETO MATLAB CÁLCULO SNR, Q Y BER*](#)*.*

3. RESULTADOS

En este apartado vamos a mostrar y comentar los resultados obtenidos a lo largo del TFG para cada uno de los trabajos realizados que hemos explicado en el apartado anterior.

Expondremos para cada una de las partes paso a paso los distintos resultados que se fueron obteniendo hasta llegar a los códigos finales, que son los que hemos comentado anteriormente.

3.1 RESULTADOS OBTENIDOS PROGRAMACIÓN TOMA DE MEDIDAS OSCILOSCOPIO DIGITAL

En este apartado vamos a comentar los distintos resultados obtenidos paso a paso hasta obtener el script final de automatización de la toma de medidas con el osciloscopio digital.

Al principio, al escribir un primer script en el que se cambiaba la frecuencia mediante el bucle for recorriendo el array, no teníamos en cuenta la base de tiempos en el osciloscopio por lo que a la hora de ir variando las frecuencias la base de tiempos estaba estática y no se podía observar correctamente las señales en el osciloscopio.

Esto lo solucionamos como hemos comentado en la Figura 22 creando un array de bases de tiempo inversamente relacionado con las frecuencias. En un primer intento de resolver el problema de la base de tiempos pretendemos relacionar directamente la base de tiempos como la inversa de la frecuencia.

```
//BASE DE TIEMPOS EN FUNCION DE LA FRECUENCIA
var inv = 1/frequency;
Scope1.Time.Base.text = inv;
```

Figura 59. Captura parte código posible solución base de tiempos osciloscopio.

Con esta solución, el valor de la variable *inv* no se ajustaba hasta la siguiente iteración del bucle, es decir, la variable siempre coge el valor de la iteración anterior. Por tanto, siendo esta una solución posible, pero a priori menos

eficiente, se prefiere crear el array con bases de tiempos que se recorre al mismo tiempo que el array de frecuencias.

En las siguientes imágenes se muestra en primer lugar una captura del osciloscopio sin realizar el ajuste de la base de tiempos, y en segundo lugar, tras crear el array de base de tiempos en el script, ambas capturas para una frecuencia de 50 KHz.

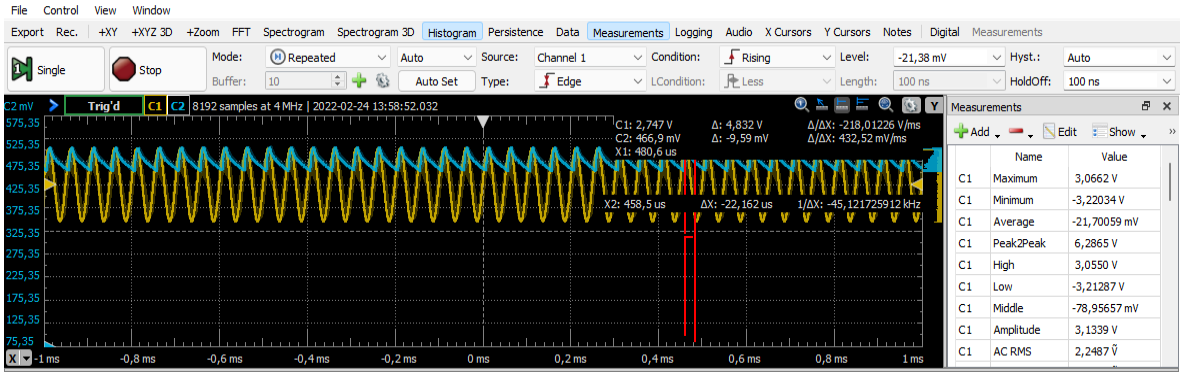


Figura 60. Osciloscopio sin ajustar la base de tiempos.

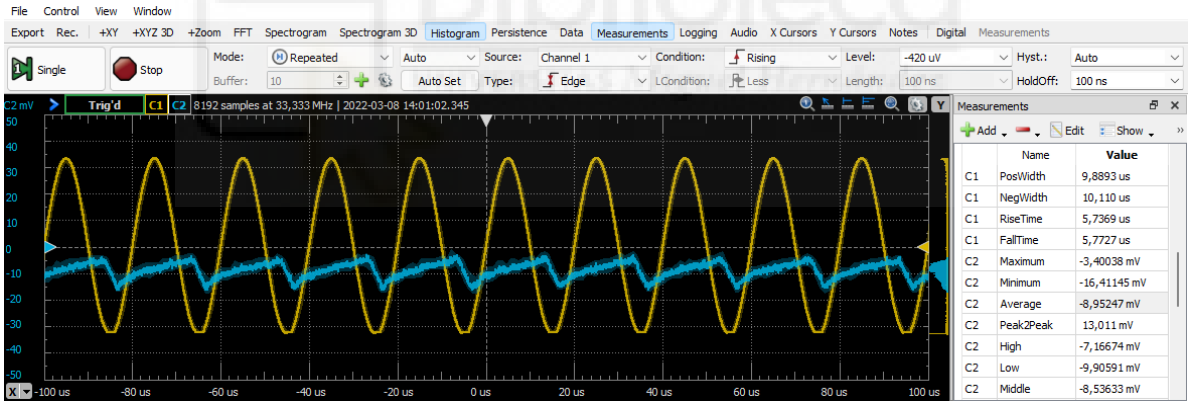


Figura 61. Osciloscopio una vez ajustada la base de tiempos.

En la primera imagen podemos ver como al no estar ajustada la base de tiempos en función de la frecuencia, a una frecuencia de 50KHz las señales no se ven de manera correcta en la pantalla del osciloscopio. En cambio, en la segunda imagen vemos como la señal se ve de manera correcta.

Una vez solucionado el problema de la base de tiempos, otra variable que nos damos cuenta de que es necesario ajustar en función de cada medición es el rango de valores de tensión (Range) del canal 2 del osciloscopio, ya que si no

está correctamente ajustado las señales se saldrán de la pantalla del osciloscopio o se verán muy pequeñas.

En una primera intención tras analizar los datos de algunas mediciones, se puede observar cómo hay que ajustar el Range la mayoría de las ocasiones en función de la distancia a la que se esté midiendo las células fotovoltaicas, por tanto, en un primer lugar, se pretende ajustar el valor del rango en función del valor que se indique en la variable *DISTANCIA* del principio del código. Añadiendo la siguiente parte de código al script de WaveForms.

```
//Canal 2
//Por defecto
Scope1.Channel2.Range.text= '50 mV/div';

//RANGE Canal 2
if(DISTANCIA < '5'){
      Scope1.Channel2.Range.text= '50 mV/div';
}
if((DISTANCIA < '4') && (DISTANCIA < '8')){
      Scope1.Channel2.Range.text= '10 mV/div';
}
if(DISTANCIA < '7'){
      Scope1.Channel2.Range.text= '5 mV/div';
}
```

Figura 62. Captura parte código posible solución Range osciloscopio.

Tras analizar la posible solución y los resultados obtenidos con esta, se concluye rápidamente que esta no es una posible solución ya que las mediciones obtenidas no son correctas y, por otro lado, nos damos cuenta de que debemos ajustar el valor de Range en función de un valor medido de la señal, para de esta forma que se ajuste de una forma más exacta, no dependiendo de un valor externo a cada medición como es la distancia.

Por tanto, se debe buscar un valor medible en la señal para ajustar en función de ese valor el Range del Canal 2. Al tratarse del rango de tensión que es una variable para ajustar la escala del eje vertical del osciloscopio (V/div) decidimos emplear una medida de tensión, más en concreto el valor pico-pico de la señal en el canal 2 para realizar el ajuste en función de ese valor.

Hay que tener en cuenta que al realizar la medición de la tensión pico-pico el valor de dicha tensión se almacena en la variable pasada la iteración del bucle.

Para solucionar esto empleamos una interpretación del filtro Kalman como ya hemos comentado anteriormente en este TFG.

```
//Canal 2 (Range)
//Medida Tensión Pico-Pico y ajustar Range en función de pico-pico

var PicoPicoCh2 = Scope1.Channel2.measure('Peak2Peak');

    if((PicoPicoCh2 <= 0.0039)){
        Scope1.Channel2.Range.text= '1 mV/div';
    }
    if((PicoPicoCh2 > 0.0039) && (PicoPicoCh2 <= 0.007)){
        Scope1.Channel2.Range.text= '2 mV/div';
    }
    if((PicoPicoCh2 > 0.007) && (PicoPicoCh2 <= 0.012)){
        Scope1.Channel2.Range.text= '5 mV/div';
    }
    if((PicoPicoCh2 > 0.012) && (PicoPicoCh2 <= 0.025)){
        Scope1.Channel2.Range.text= '10 mV/div';
    }
    if((PicoPicoCh2 > 0.025) && (PicoPicoCh2 <= 0.058)){
        Scope1.Channel2.Range.text= '20 mV/div';
    }
    if(PicoPicoCh2 > 0.058 ){
        Scope1.Channel2.Range.text= '50 mV/div';
    }
```

Figura 63. Captura parte código posible solución Range osciloscopio sin emplear filtro Kalman.

Con el anterior código al analizar los resultados nos damos cuenta de lo que ya hemos comentado, el valor de las variables se actualiza en la siguiente iteración del bucle, por lo que los ajustes que se realizan del rango de tensión del osciloscopio no eran correctos.

Otro problema a la hora de obtener los resultados esperados surge con ciertas mediciones en las que es necesario ajustar el valor del offset, que como hemos comentado en el apartado anterior se soluciona empleando el valor máximo y el mínimo de la señal, haciendo el valor medio y sumándole la amplitud de la señal para dejar la señal centrada en el osciloscopio.

Una vez ajustados todos estos parámetros en el osciloscopio observamos como la toma de medidas se agiliza de manera notable obteniendo los resultados esperados en un tiempo muy inferior a lo que se tardaba antes.

Recordamos que con el script en el osciloscopio digital se esperaba ir realizando saltos en frecuencia para poder medir el comportamiento de las células fotovoltaicas, y exportar los archivos de resultados obtenidos de las mediciones.

Por un lado, necesitamos los archivos de datos en formato .csv del osciloscopio, histograma y medidas de ambos canales, y por otro lado, las capturas en formato .png del osciloscopio y del histograma.

Una vez ejecutamos el script en *WaveForms*, conforme van avanzando las frecuencias vamos observando lo siguiente por la ventana de salida (Output).

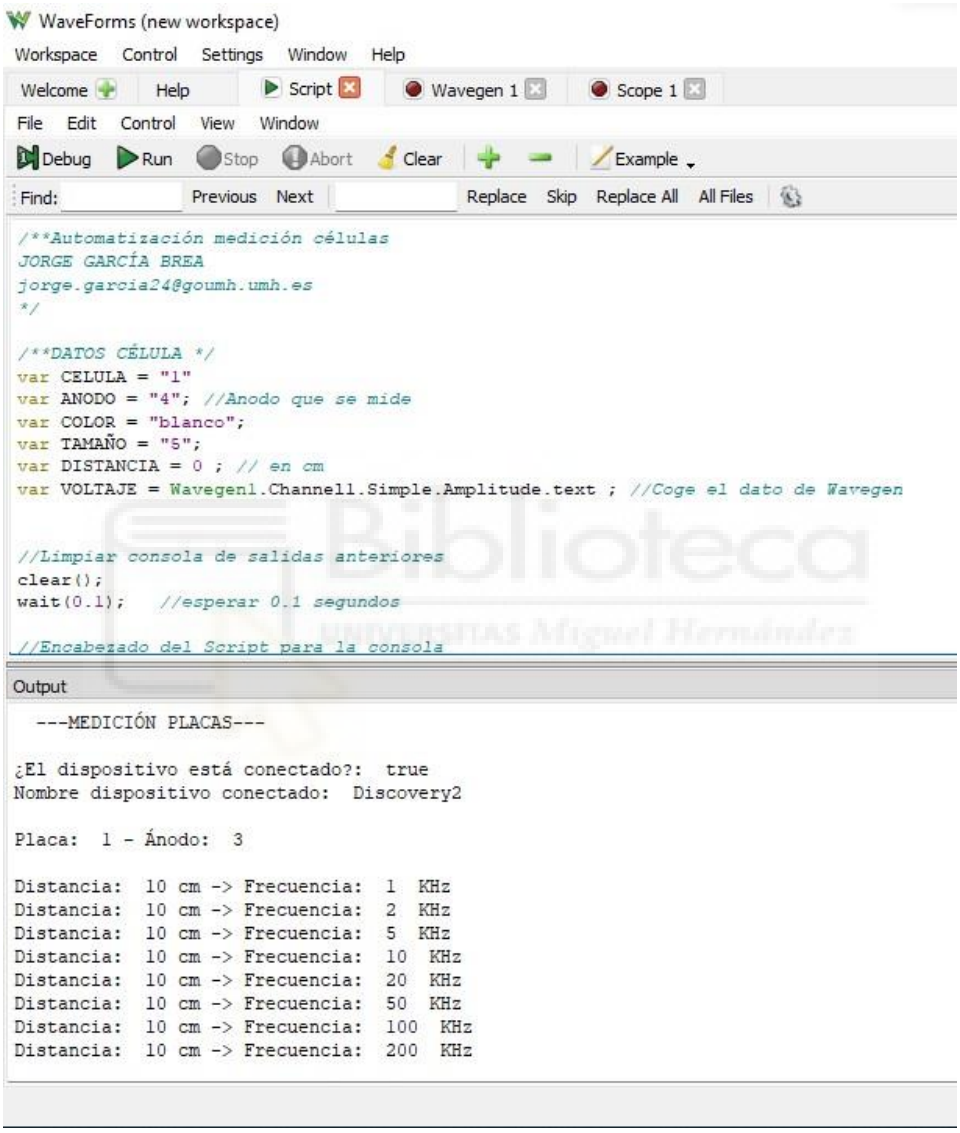
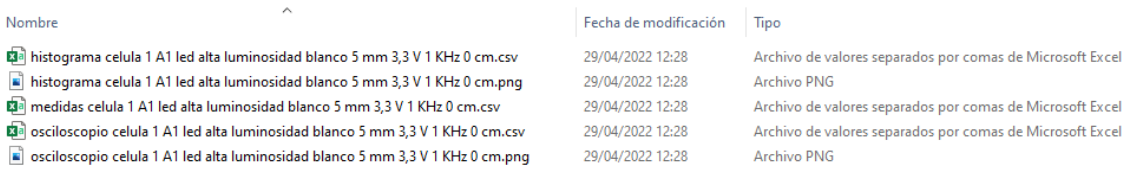


Figura 64. Captura ejemplo ejecución Script automatización toma de medidas.

En la anterior imagen vemos como se nos indica la célula y ánodo que se está midiendo y una a una van apareciendo las frecuencias que va recorriendo el bucle.

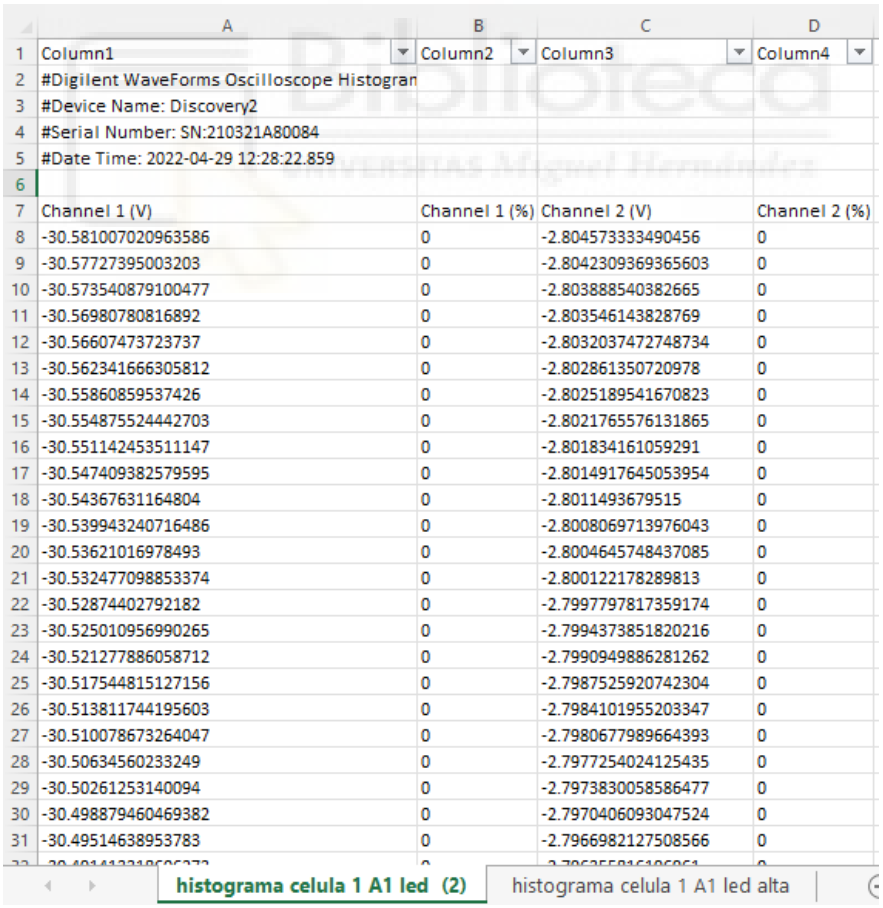
Este script nos descarga los archivos deseados con el formato que nosotros le hemos indicado anteriormente para poder archivarlos de forma correcta, a continuación, vemos un ejemplo de la exportación de archivos.



Nombre	Fecha de modificación	Tipo
histograma celula 1 A1 led alta luminosidad blanco 5 mm 3,3 V 1 KHz 0 cm.csv	29/04/2022 12:28	Archivo de valores separados por comas de Microsoft Excel
histograma celula 1 A1 led alta luminosidad blanco 5 mm 3,3 V 1 KHz 0 cm.png	29/04/2022 12:28	Archivo PNG
medidas celula 1 A1 led alta luminosidad blanco 5 mm 3,3 V 1 KHz 0 cm.csv	29/04/2022 12:28	Archivo de valores separados por comas de Microsoft Excel
osciloscopio celula 1 A1 led alta luminosidad blanco 5 mm 3,3 V 1 KHz 0 cm.csv	29/04/2022 12:28	Archivo de valores separados por comas de Microsoft Excel
osciloscopio celula 1 A1 led alta luminosidad blanco 5 mm 3,3 V 1 KHz 0 cm.png	29/04/2022 12:28	Archivo PNG

Figura 65. Captura ejemplo archivos exportados del osciloscopio.

Vamos a ir explicando uno a uno que contiene cada uno de los archivos, en primer lugar, el archivo de datos del histograma nos muestra cuatro columnas: tensión del canal 1, porcentaje de canal 1, tensión del canal 2 y porcentaje del canal 2. Este archivo será el que emplearemos en Matlab para los distintos cálculos de los parámetros de calidad de la señal.



	A	B	C	D
1	Column1	Column2	Column3	Column4
2	#Digilent WaveForms Oscilloscope Histogran			
3	#Device Name: Discovery2			
4	#Serial Number: SN:210321A80084			
5	#Date Time: 2022-04-29 12:28:22.859			
6				
7	Channel 1 (V)	Channel 1 (%)	Channel 2 (V)	Channel 2 (%)
8	-30.581007020963586	0	-2.804573333490456	0
9	-30.57727395003203	0	-2.8042309369365603	0
10	-30.573540879100477	0	-2.803888540382665	0
11	-30.56980780816892	0	-2.803546143828769	0
12	-30.56607473723737	0	-2.8032037472748734	0
13	-30.562341666305812	0	-2.802861350720978	0
14	-30.55860859537426	0	-2.8025189541670823	0
15	-30.554875524442703	0	-2.8021765576131865	0
16	-30.551142453511147	0	-2.801834161059291	0
17	-30.547409382579595	0	-2.8014917645053954	0
18	-30.54367631164804	0	-2.8011493679515	0
19	-30.539943240716486	0	-2.8008069713976043	0
20	-30.53621016978493	0	-2.8004645748437085	0
21	-30.532477098853374	0	-2.800122178289813	0
22	-30.52874402792182	0	-2.7997797817359174	0
23	-30.525010956990265	0	-2.7994373851820216	0
24	-30.521277886058712	0	-2.7990949886281262	0
25	-30.517544815127156	0	-2.7987525920742304	0
26	-30.513811744195603	0	-2.7984101955203347	0
27	-30.510078673264047	0	-2.7980677989664393	0
28	-30.50634560233249	0	-2.797725402412543 5	0
29	-30.50261253140094	0	-2.7973830058586477	0
30	-30.498879460469382	0	-2.7970406093047524	0
31	-30.49514638953783	0	-2.7966982127508566	0

histograma celula 1 A1 led (2) | histograma celula 1 A1 led alta

Figura 66. Captura ejemplo archivo de datos histograma exportado.

En cuanto al histograma en formato de imagen en las exportaciones obtenemos lo siguiente, que al fin y al cabo es la representación gráfica de todos los datos que tenemos en el archivo .csv anterior.

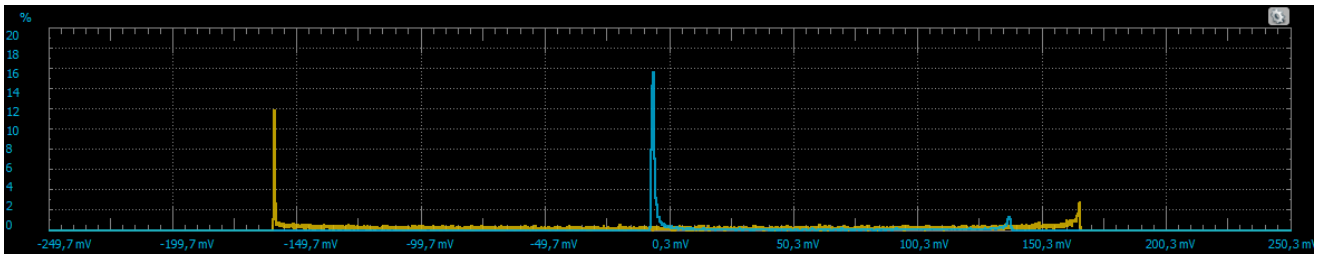


Figura 67. Ejemplo histograma exportado en formato .png.

Para el caso de los archivos del osciloscopio, obtenemos algo similar a lo anterior, por un lado, tenemos el archivo de datos con tres columnas: tiempo, tensión del canal 1 y tensión del canal 2. Este es el archivo que se emplea para representar el diagrama de ojo mediante Matlab.

	A	B	C	D	E
1	#Digilent WaveForms Oscilloscope Acquisition				
2	#Device Name: Discovery2				
3	#Serial Number: SN:210321A80084				
4	#Date Time: 2022-04-29 12:28:22.857				
5	#Sample rate: 800000Hz				
6	#Samples: 8192				
7	#Trigger: Source: Channel 1 Type: Edge Condition: Rising Level: 0 V Hyst.: Auto HoldOff: 100 ns				
8	#Channel 1: Range: 1 V/div Offset: 0 V Attenuation: 1 X Sample Mode: Average				
9	#Channel 2: Range: 50 mV/div Offset: -300 uV Attenuation: 1 X Sample Mode: Average				
10	#Wavegen Channel 1: Running				
11	#Mode: Simple				
12	#Type: Sine				
13	#Frequency: 1 kHz				
14	#Period: 1 ms				
15	#Amplitude: 3,3 V				
16	#Offset: 0 V				
17	#Symmetry: 50 %				
18	#Phase: 0 °				
19					
20	Time (s)	Channel 1 (V)	Channel 2 (V)		
21	-0.005119375000000001	-22,5446479232665	0.007187167100445769		
22	-0.005118125	-22,3206636673732	0.006502373992654512		
23	-0.005116875	-22,1340101207954	0.0061599774387585725		
24	-0.005115625	-21,9473565742178	0.005475184330967626		
25	-0.005114375	-21,7607030276400	0.005475184330967626		
26	-0.005113125000000005	-21,5367187717468	0.0044479946692807405		
27	-0.005111875	-21,3127345158535	0.004105598115385111		
28	-0.005110625000000001	-21,1260809692758	0.004105598115385111		
29	-0.005109375	-20,9767581320135	0.004105598115385111		
30	-0.005108125	-20,7154431668046	0.0030784084536980704		
31	-0.005106875	-20,5287896202270	0.0027360118998025974		

osciloscopio celula 1 A1 led al

Figura 68. Captura ejemplo archivo de datos de osciloscopio exportado.

Como ocurría en el histograma, la captura en formato .png del osciloscopio es la representación gráfica de los datos anteriores.

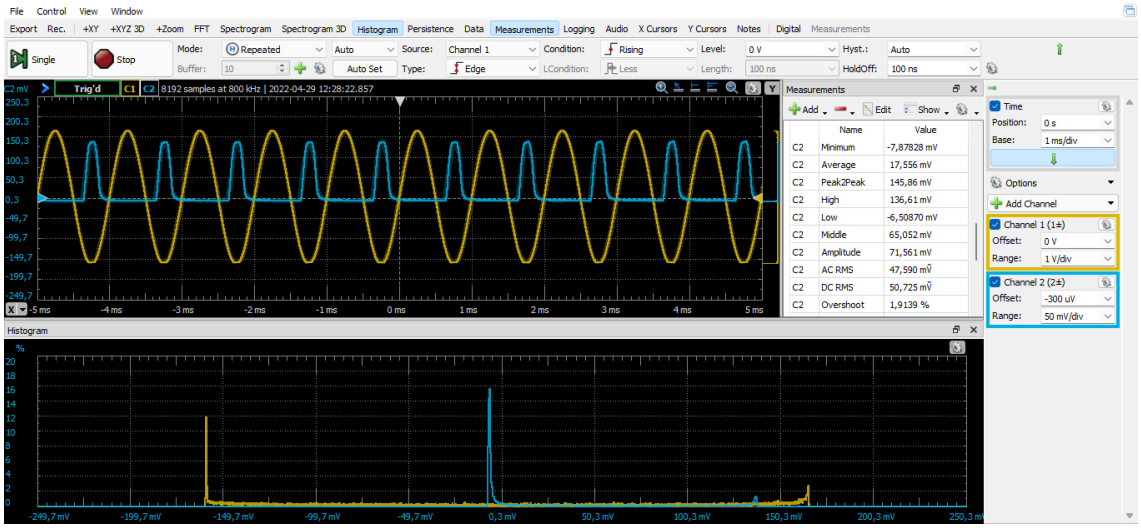
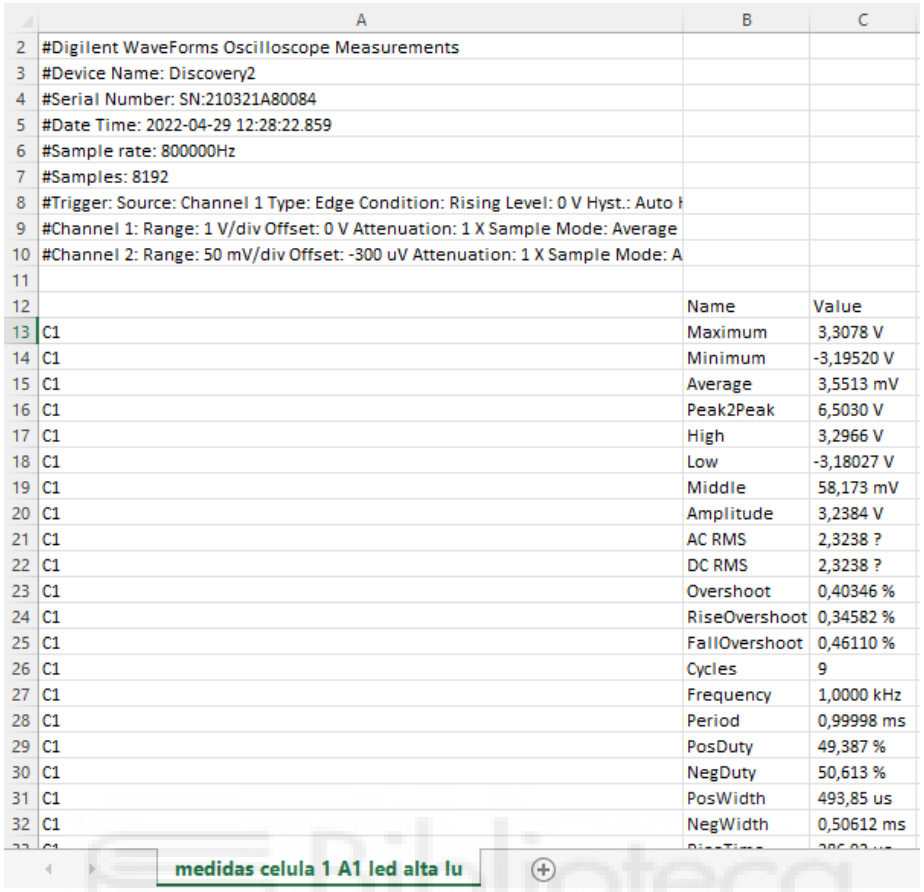


Figura 69. Ejemplo osciloscopio exportado en formato .png.

Aunque el archivo de datos y la imagen represente lo mismo, es importante exportar ambos archivos ya que de las imágenes podemos tener una idea directa de que datos hemos obtenido en la medición, pero no podemos trabajar con las imágenes para representar el diagrama de ojo o calcular parámetros de la señal, de igual forma que tampoco podemos hacernos un idea rápida y visual de cómo ha ido la medición con tan solo el archivo de datos, ya que es una tabla con muchas muestras. Por ese motivo es importante obtener los resultados en ambos formatos.

Por último, el archivo de datos de medidas de ambos canales resulta muy importante ya que en él tenemos los parámetros característicos de ambos canales, como son las tensiones máximas y mínimas, tensión pico-pico, amplitud, frecuencia, tiempo de subida y tiempo de bajada, entre otros.

El archivo es una tabla de 3 columnas: canal al que se refiere la medida, nombre de la medida y valor de esta.



	A	B	C
2	#Digilent WaveForms Oscilloscope Measurements		
3	#Device Name: Discovery2		
4	#Serial Number: SN:210321A80084		
5	#Date Time: 2022-04-29 12:28:22.859		
6	#Sample rate: 800000Hz		
7	#Samples: 8192		
8	#Trigger: Source: Channel 1 Type: Edge Condition: Rising Level: 0 V Hyst.: Auto H		
9	#Channel 1: Range: 1 V/div Offset: 0 V Attenuation: 1 X Sample Mode: Average		
10	#Channel 2: Range: 50 mV/div Offset: -300 uV Attenuation: 1 X Sample Mode: A		
11			
12		Name	Value
13	C1	Maximum	3,3078 V
14	C1	Minimum	-3,19520 V
15	C1	Average	3,5513 mV
16	C1	Peak2Peak	6,5030 V
17	C1	High	3,2966 V
18	C1	Low	-3,18027 V
19	C1	Middle	58,173 mV
20	C1	Amplitude	3,2384 V
21	C1	AC RMS	2,3238 ?
22	C1	DC RMS	2,3238 ?
23	C1	Overshoot	0,40346 %
24	C1	RiseOvershoot	0,34582 %
25	C1	FallOvershoot	0,46110 %
26	C1	Cycles	9
27	C1	Frequency	1,0000 kHz
28	C1	Period	0,99998 ms
29	C1	PosDuty	49,387 %
30	C1	NegDuty	50,613 %
31	C1	PosWidth	493,85 us
32	C1	NegWidth	0,50612 ms

Figura 70. Captura ejemplo archivo de datos de medidas exportado.

Como observamos los resultados obtenidos para todos los archivos exportados son los que esperábamos, por un lado, tanto el osciloscopio como el histograma están bien ajustados en función de las señales y, por otro lado, en los archivos de datos exportados tenemos todos los parámetros que necesitamos.

3.1.1 Medición a frecuencias superiores a 200 KHz

Una vez analizados los resultados obtenidos para algunas de las células, nos damos cuenta de que para algunos de los casos el comportamiento de la célula hace que resulte interesante medir a frecuencias superiores a 200 KHz.

Uno de estos casos son las células fabricadas con perovskita, las cuales tienen una mayor tensión pico-pico a frecuencias más altas. Un ejemplo de este comportamiento lo podemos observar en los siguientes resultados de una célula de perovskita fabricada el 30 de marzo de 2022, la cual a una frecuencia de 1 KHz y 0 cm de distancia se obtiene una tensión pico-pico en el Canal 2 de 7.1903 mV, en cambio, a una frecuencia de 500 KHz y a la misma distancia, se obtiene

una tensión pico-pico de 24.653 mV, para una señal que a pesar de tener algo de ruido se pueden distinguir de manera correcta los pulsos.

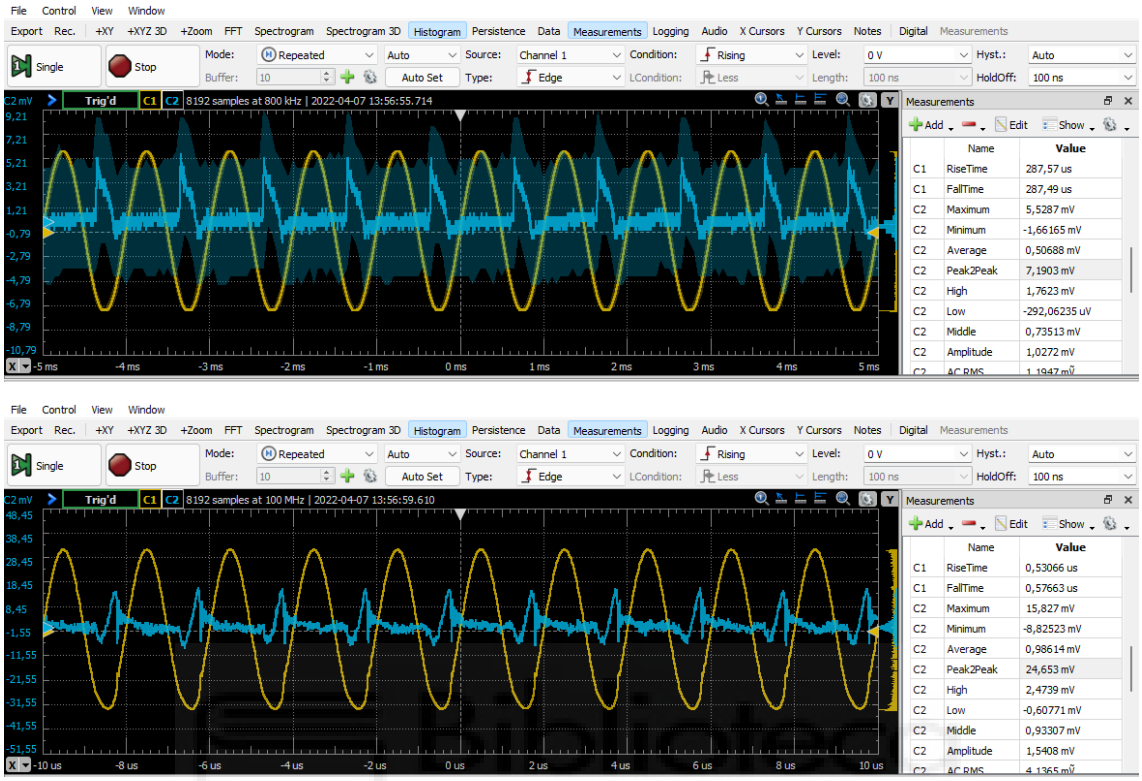


Figura 71. Ejemplo captura osciloscopio mediciones realizadas a célula de perovskita (Arriba: Frecuencia 1 KHz. Abajo: Frecuencia 500 KHz).

Por esta razón decidimos extender el código implementado y que ya hemos explicado, para estos casos en los que nos interesa obtener mediciones a frecuencias altas.

Para ello, una vez finalizado el bucle con el que en el código anterior recorríamos todas las frecuencias de 1 KHz a 200 KHz, vamos a añadir una condición para que en función del valor de la tensión pico-pico en la última frecuencia medida (200 KHz) se realicen mediciones a 500 KHz, 1 MHz y 2 MHz.

Consideramos que el valor pico-pico mínimo que nos interesa para obtener medidas en esas frecuencias es 7 mV, ya que un valor inferior a este voltaje no es de nuestro interés.

Para esta comprobación empleamos una condición if, y en el caso en que se cumpla, se crea un array de frecuencias que se recorrerá al igual que sucedía

antes, ajustando los parámetros de la señal en el osciloscopio y exportando los archivos con las mediciones.

```
//MEDIDAS DE 500K, 1M, 2M
if(PicoPicoCh2 > 0.007){

//Array con frecuencias a medir
var frecuencias2 =[500,1000,2000] //en KHz
```

Figura 72. Captura parte código comprobación tensión pico-pico mediciones a frecuencias altas.

Podemos comprobar como esta ampliación del script realiza su función, al exportar los archivos necesarios con las mediciones de la célula a estas nuevas frecuencias que nos interesan.

Nombre	Tipo
histograma celula 3 A1 led alta luminosidad blanco 5 mm 3,3 V 500 KHz 0 cm.csv	Archivo de valores separados por comas de Microsoft Excel
histograma celula 3 A1 led alta luminosidad blanco 5 mm 3,3 V 500 KHz 0 cm.png	Archivo PNG
histograma celula 3 A1 led alta luminosidad blanco 5 mm 3,3 V 1000 KHz 0 cm.csv	Archivo de valores separados por comas de Microsoft Excel
histograma celula 3 A1 led alta luminosidad blanco 5 mm 3,3 V 1000 KHz 0 cm.png	Archivo PNG
histograma celula 3 A1 led alta luminosidad blanco 5 mm 3,3 V 2000 KHz 0 cm.csv	Archivo de valores separados por comas de Microsoft Excel
histograma celula 3 A1 led alta luminosidad blanco 5 mm 3,3 V 2000 KHz 0 cm.png	Archivo PNG
medidas celula 3 A1 led alta luminosidad blanco 5 mm 3,3 V 500 KHz 0 cm.csv	Archivo de valores separados por comas de Microsoft Excel
medidas celula 3 A1 led alta luminosidad blanco 5 mm 3,3 V 1000 KHz 0 cm.csv	Archivo de valores separados por comas de Microsoft Excel
medidas celula 3 A1 led alta luminosidad blanco 5 mm 3,3 V 2000 KHz 0 cm.csv	Archivo de valores separados por comas de Microsoft Excel
osciloscopio celula 3 A1 led alta luminosidad blanco 5 mm 3,3 V 500 KHz 0 cm.csv	Archivo de valores separados por comas de Microsoft Excel
osciloscopio celula 3 A1 led alta luminosidad blanco 5 mm 3,3 V 500 KHz 0 cm.png	Archivo PNG
osciloscopio celula 3 A1 led alta luminosidad blanco 5 mm 3,3 V 1000 KHz 0 cm.csv	Archivo de valores separados por comas de Microsoft Excel
osciloscopio celula 3 A1 led alta luminosidad blanco 5 mm 3,3 V 1000 KHz 0 cm.png	Archivo PNG
osciloscopio celula 3 A1 led alta luminosidad blanco 5 mm 3,3 V 2000 KHz 0 cm.csv	Archivo de valores separados por comas de Microsoft Excel
osciloscopio celula 3 A1 led alta luminosidad blanco 5 mm 3,3 V 2000 KHz 0 cm.png	Archivo PNG

Figura 73. Captura ejemplo archivos exportados osciloscopio a 500 KHz, 1 MHz y 2 MHz.

El código completo en WaveForms para estos casos está incluido en *5.4 ANEXO 4: CÓDIGO COMPLETO WAVEFORMS DE AUTOMATIZACIÓN TOMA DE MEDIDAS, AÑADIENDO POSIBILIDAD A FRECUENCIAS ALTAS (500 KHz, 1MHz y 2MHz).*

3.2 RESULTADOS OBTENIDOS REPRESENTACIÓN DIAGRAMA DE OJO

En este apartado vamos a comentar los distintos resultados obtenidos paso a paso hasta obtener el script final para la representación del diagrama de ojo, a partir de las medidas exportadas del osciloscopio digital.

En un primer lugar, para la representación del diagrama de ojo empleamos archivos de osciloscopio en formato .csv de medidas a células realizadas antes de automatizar la toma de datos, al ejecutar el script con estos archivos de medidas obtenemos como resultado representaciones raras.

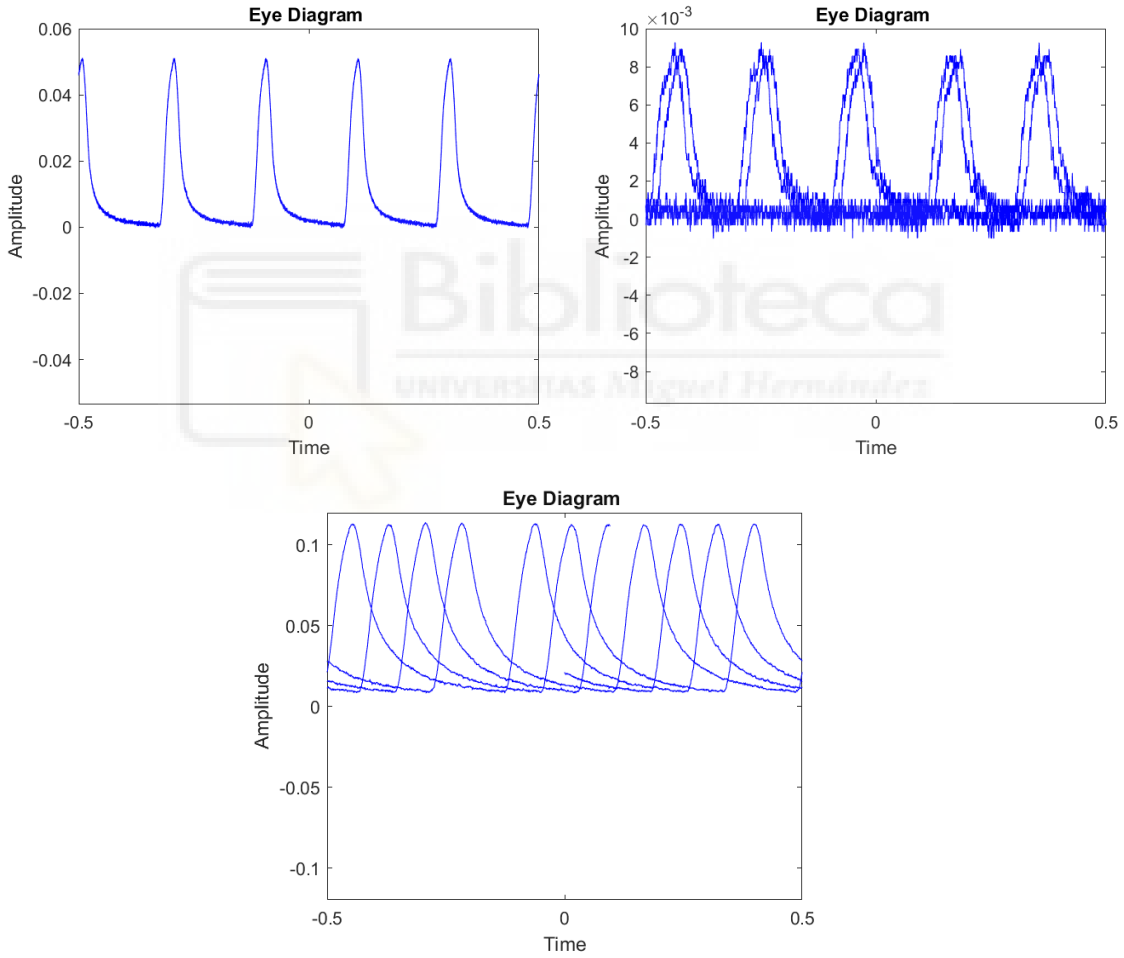


Figura 74. Representaciones diagrama de ojo (primeras pruebas con errores).

En el diagrama de arriba a la izquierda podemos ver como el resultado es el esperado, donde vemos los distintos picos de la señal y podemos distinguir el nivel alto del nivel bajo. Pero también se obtienen otros casos como la imagen de la derecha, una representación donde se ve la señal repetida con un

desplazamiento temporal, o el caso del diagrama de abajo donde faltan partes de la señal.

Tras analizar en el laboratorio estos resultados, concluimos que puede deberse a que a la hora de tomar los datos no estuviese ajustada la base de tiempos de manera correcta, es decir, que al exportar los datos en el osciloscopio, en pantalla hubiera demasiados periodos de la señal, haciendo que los valores obtenidos no sean del todo correctos y por ello a la hora de representar el diagrama de ojo no salga el resultado esperado.

Cabe destacar que los errores vienen cuando hay muchos períodos, ya que cuando se representan entre 6 a 10 periodos la representación del diagrama de ojo es correcta, en cambio cuando hay más empiezan a aparecer los errores.

Por tanto, estos errores se esperan resolver al ajustar de manera correcta la base de tiempos en el osciloscopio para la toma de medidas, de ahí la importancia de todo lo explicado en apartados anteriores en el script de automatización.

Una vez se tomaron medidas empleando el código de automatización, surgieron algunos problemas en Matlab a la hora de representar, como por ejemplo en ciertos diagramas de ojo el eje Y no se ajustaba de manera correcta, por lo que se apreciaba únicamente una línea en el diagrama de ojo.

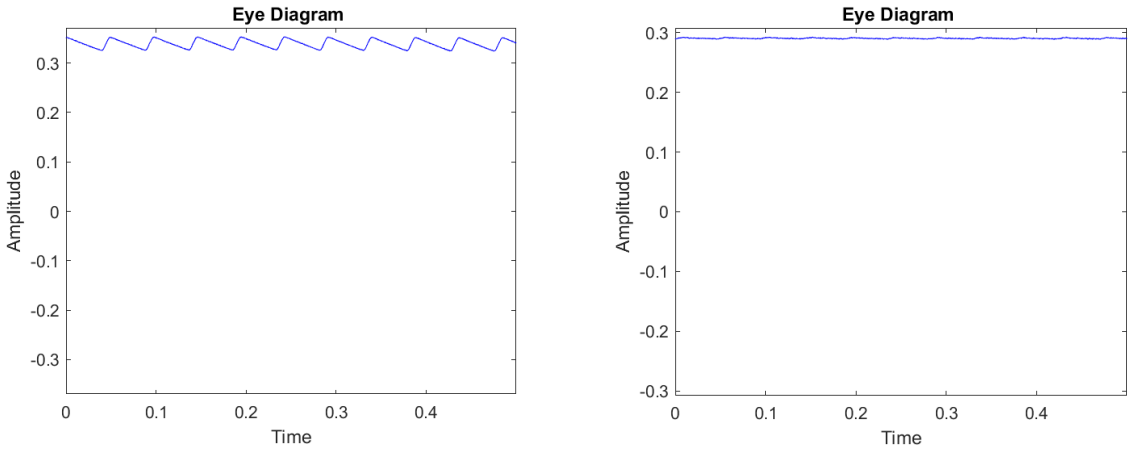


Figura 75. Representaciones diagrama de ojo (primeras pruebas con errores) (2).

Para solucionar esto configuramos el eje Y de manera automática en Matlab para que se ajuste a los valores de cada señal.

Como hemos comentado, el script en Matlab nos permite descargar de forma ordenada en una carpeta todas las representaciones de los diagramas para poder consultarlas más adelante, un ejemplo de esta exportación es la siguiente.

Nombre	Fecha	Tipo
Diagrama de Ojo-osciloscopio celula 1 A1 led alta luminosidad blanco 5 mm 3,3 V 1 KHz 0 cm.csv.png	31/03/2022 18:40	Archivo PNG
Diagrama de Ojo-osciloscopio celula 1 A1 led alta luminosidad blanco 5 mm 3,3 V 1 KHz 1 cm.csv.png	31/03/2022 18:40	Archivo PNG
Diagrama de Ojo-osciloscopio celula 1 A1 led alta luminosidad blanco 5 mm 3,3 V 1 KHz 2 cm.csv.png	31/03/2022 18:40	Archivo PNG
Diagrama de Ojo-osciloscopio celula 1 A1 led alta luminosidad blanco 5 mm 3,3 V 1 KHz 3 cm.csv.png	31/03/2022 18:40	Archivo PNG
Diagrama de Ojo-osciloscopio celula 1 A1 led alta luminosidad blanco 5 mm 3,3 V 1 KHz 4 cm.csv.png	31/03/2022 18:40	Archivo PNG
Diagrama de Ojo-osciloscopio celula 1 A1 led alta luminosidad blanco 5 mm 3,3 V 1 KHz 5 cm.csv.png	31/03/2022 18:40	Archivo PNG
Diagrama de Ojo-osciloscopio celula 1 A1 led alta luminosidad blanco 5 mm 3,3 V 1 KHz 6 cm.csv.png	31/03/2022 18:40	Archivo PNG
Diagrama de Ojo-osciloscopio celula 1 A1 led alta luminosidad blanco 5 mm 3,3 V 1 KHz 7 cm.csv.png	31/03/2022 18:40	Archivo PNG
Diagrama de Ojo-osciloscopio celula 1 A1 led alta luminosidad blanco 5 mm 3,3 V 1 KHz 8 cm.csv.png	31/03/2022 18:40	Archivo PNG
Diagrama de Ojo-osciloscopio celula 1 A1 led alta luminosidad blanco 5 mm 3,3 V 1 KHz 9 cm.csv.png	31/03/2022 18:40	Archivo PNG
Diagrama de Ojo-osciloscopio celula 1 A1 led alta luminosidad blanco 5 mm 3,3 V 1 KHz 10 cm.csv.png	31/03/2022 18:40	Archivo PNG

Figura 76. Captura ejemplo exportación diagramas de ojo.

Tal y como hemos comentado anteriormente los archivos se nombran como "Diagrama de Ojo-" y el nombre del archivo .csv del osciloscopio para poder identificar a que medición corresponde cada diagrama.

Los diagramas de ojo que se obtienen dependen de cada uno de los archivos que se pasan para representar. Algún ejemplo de representación para algunos casos extremos, en primer lugar, para el caso ideal de una frecuencia de 1 KHz y a una distancia de 0 cm obtenemos una representación, donde podemos ver claramente la señal con los picos en el nivel alto a 80 mV y en el nivel bajo a 0 V, sin apenas ruido en la señal.

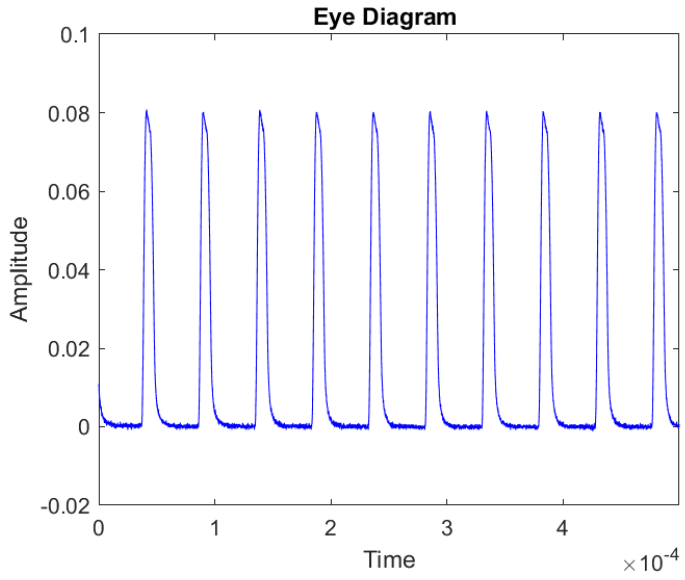


Figura 77. Representación diagrama de ojo para señal a 1 KHz y 0 cm.

Por otro lado, para el caso de una medida realizada también a 1 KHz, pero a una distancia de 10 cm, vemos claramente como la señal tiene mucho ruido y los niveles no están tan definidos, es más difícil apreciar la tensión para el nivel alto y para el nivel bajo.

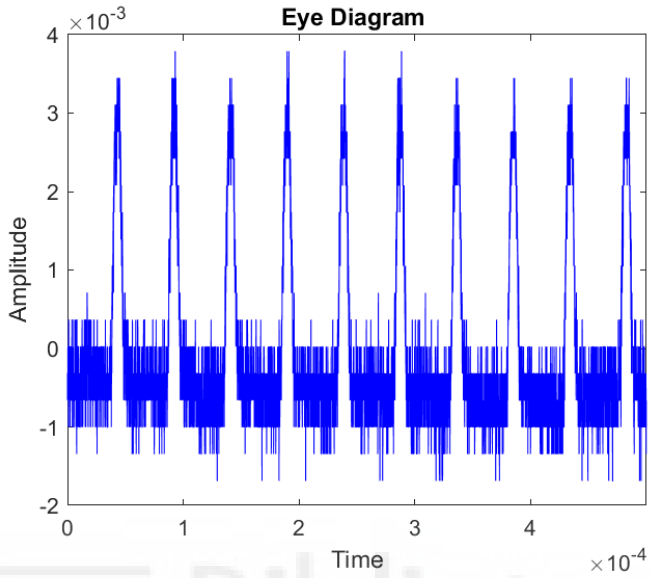


Figura 78. Representación diagrama de ojo para señal a 1 KHz y 10 cm.

Otro de los casos extremos a la hora de representar el diagrama de ojo es para casi todas las mediciones a una frecuencia de 200KHz, al obtener una señal con mucho ruido en el receptor en el osciloscopio, los datos exportados nos proporcionan unas representaciones donde no podemos observar casi nada.

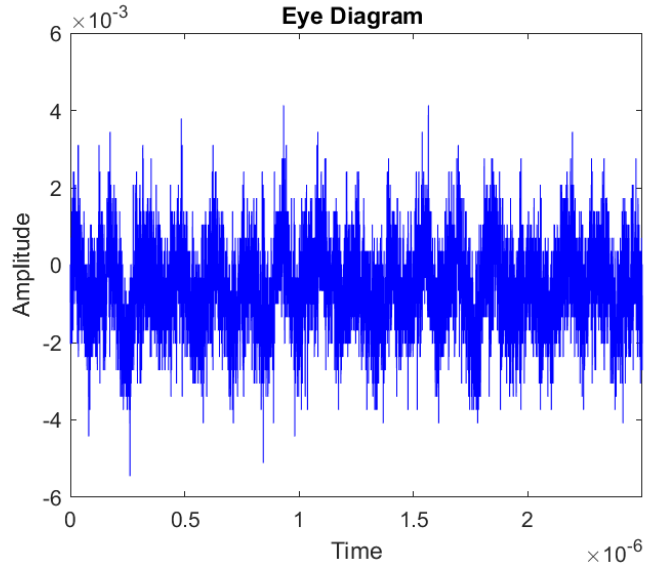


Figura 79. Representación diagrama de ojo para señal a 200 KHz y 5 cm.

Para casos no tan extremos de frecuencia o de distancia, obtenemos representaciones donde podemos observar de un simple vistazo, como se comporta la señal para ese caso y los valores de tensión para el nivel alto y bajo.

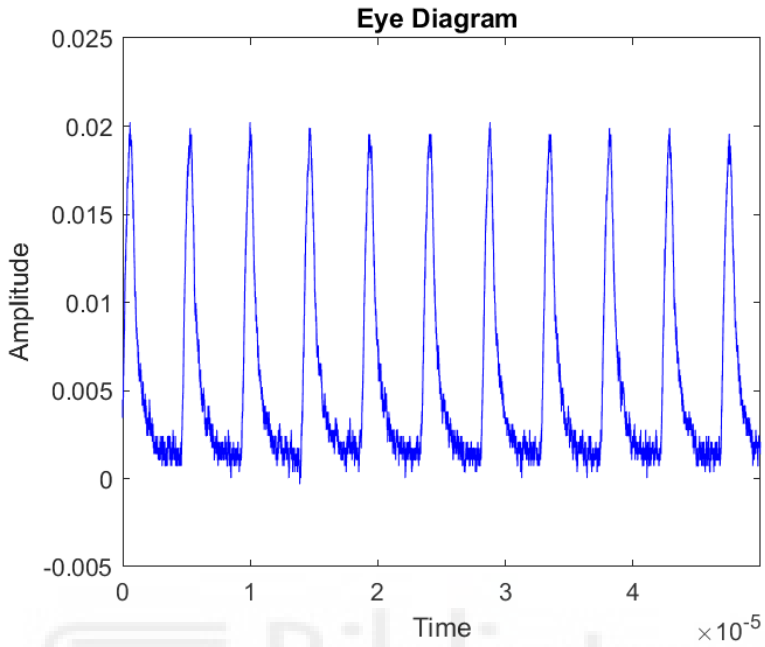


Figura 80. Representación diagrama de ojo para señal a 10 KHz y 3 cm.

Para este caso de una medición a una frecuencia de 10 KHz y una distancia de 3 cm, podemos ver como ya no es el caso ideal de la Figura 77, si no que observamos como la señal tiene un poco de ruido y también cuenta con un pequeño offset, el nivel alto se encuentra sobre 20 mV y el nivel bajo sobre 2 mV. Este análisis del diagrama de ojo se podría hacer para cada uno de los resultados, obteniendo rápidamente una idea del comportamiento de la señal.

3.3 RESULTADOS OBTENIDOS PROGRAMACIÓN CÁLCULO SNR, Q y BER

En este apartado vamos a comentar los distintos resultados obtenidos paso a paso hasta obtener el script final para el cálculo de la SNR, factor de calidad Q y la BER, a partir de las medidas exportadas del osciloscopio digital.

En un primer lugar, siguiendo el planteamiento expuesto en el apartado 2.7 PROGRAMACIÓN CÁLCULO SNR, Q y BER que se basa en los fundamentos teóricos explicados en el apartado 2.3 se planteó un código en Matlab suponiendo un caso ideal, en el que se obtienen los picos máximos para cada nivel de señal y en que obtenemos los resultados de σ0 y σ1 de forma directa, lo que supone una situación muy lejos de la realidad.

El primer inconveniente de este primer código es el caso en el que en el archivo de histograma del osciloscopio no obtenemos al menos dos picos máximos, ya que nuestro cálculo se basa en un valor V1 y un valor V0. Para solucionar este primer problema, como ya hemos comentado antes se añade una condición en el caso en que no haya al menos dos máximos indicaremos en la tabla que "Falta Max". Tras analizar resultados obtenidos y ver en qué casos ocurría esto, observamos como es en situaciones de una señal recibida muy distorsionada, normalmente a altas frecuencias (100 kHz y 200 KHz) por lo que no supone un motivo de preocupación, ya que son situaciones que ni visualmente somos capaces de distinguir máximos en el histograma. Un ejemplo de situación en la que no podríamos obtener al menos dos máximos es la siguiente:

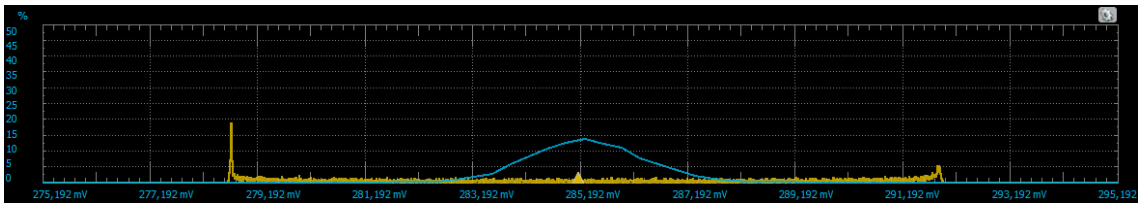


Figura 81. Ejemplo histograma obtenido sin dos valores máximos.

Podemos observar cómo sí que se podría obtener un valor para el nivel de señal V0, pero para el nivel V1, la señal está tan distorsionada que no hay otro máximo. Es un histograma de una medición a una frecuencia de 200 KHz y 10 cm de

distancia, es decir, el caso más extremo que nosotros medimos. Si nos vamos al resultado obtenido en el osciloscopio, podemos ver como la señal recibida tiene una tensión pico-pico muy baja y claramente está muy distorsionada, por eso recogemos ese resultado en el histograma.

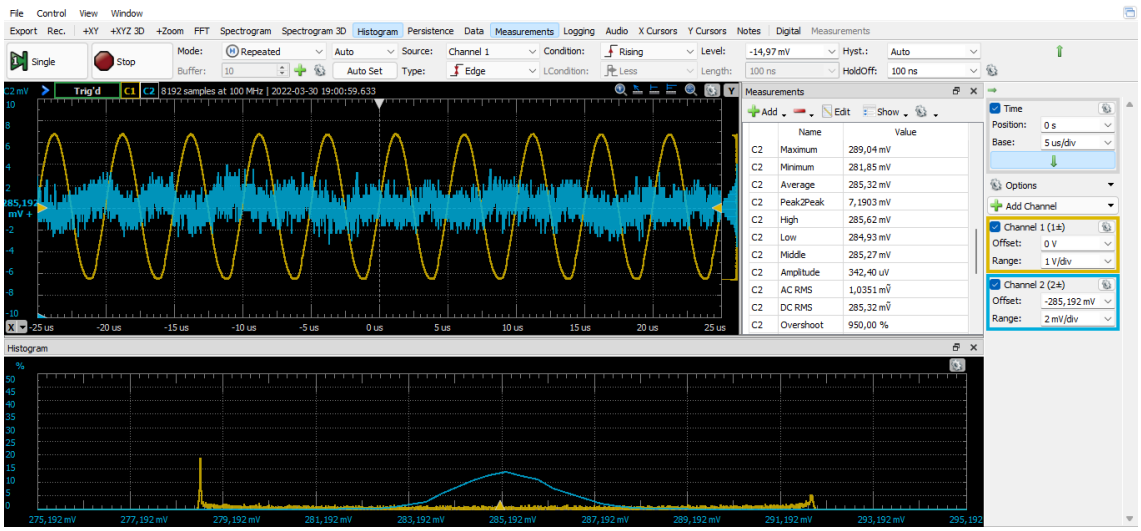


Figura 82. Resultado osciloscopio ejemplo histograma obtenido sin dos valores máximos.

Una vez añadida la condición en el código, vemos como al obtener la tabla con los resultados, para el caso en el que suceda esta situación, nos aparecerá lo siguiente:

324	celula 6 A6 led alta luminosidad blanco 5 mm 3,3 V 200 KHz 10 cm	Falta max	Falta max	Falta max	Falta max	Falta max	Falta max

Figura 83. Ejemplo fila de tabla de resultados con falta de máximos.

Otro de los problemas que surgen al considerar un caso ideal para el primer código es a la hora de obtener σ0 y σ1, ya que para obtener estos valores necesitamos el valor de la tensión al 60% del valor máximo V0 y V1 como ya hemos explicado.

Para obtener estos valores es importante tener en cuenta que, en el archivo de datos del histograma, tenemos unos valores de tensión que es muy complicado que sean exactamente los mismos que buscamos para el 60% de las tensiones máximas. Es decir, una vez nosotros calculamos el 60% de V0 es prácticamente imposible encontrar exactamente el mismo valor en el archivo del histograma, para solucionar esto se plantean varias soluciones: una de ellas, seleccionar el

valor de tensión más próximo al que buscamos, pero debemos tener en cuenta que, aunque obtenemos un archivo de datos con 16384 filas, con valores desde -30V a 30V realmente las tensiones en las que nosotros tenemos valores en el histograma son muy pocas filas.

Por ejemplo, tomamos como muestra un archivo de una medición a una célula a una frecuencia de 1 KHz y una distancia de 0 cm, en la que obtenemos el nivel de tensión para V0 con un porcentaje de 15.5625% en el histograma, por lo que deberíamos buscar para el 60% un valor de 9.3375% este valor como podemos ver en la siguiente imagen, no lo podemos obtener exactamente de forma directa en la tabla. Es importante observar como en las filas entre las que está ese valor los saltos son muy grandes, por lo que seleccionar directamente el valor más cercano al que queremos, nos llevaría a cometer un error bastante grande.

	Channel1V	Channel1	Channel2V	Channel2
	Number	Number	Number	Number
8175	-0.08555058109338616	0.0375	-0.00753588471706626	1.9875
8176	-0.08181751016183147	0.05	-0.007193488163170632	7.9125
8177	-0.0780844392302768	0.05	-0.006851091609275004	14.175
8178	-0.07435136829872212	0.0125	-0.006508695055379375	15.5625
8179	-0.07061829736716743	0.0125	-0.006166298501483746	11.1625
8180	-0.06688522643561276	0.05	-0.005823901947588118	7.0125
8181	-0.06315215550405807	0.025	-0.005481505393692489	3.15
8182	-0.05941908457250339	0.0375	-0.00513910883979686	2.5

Figura 84. Ejemplo datos archivo histograma.

Para el caso anterior, si escogiéramos directamente el valor más cercano al que buscamos (9.3375%) escogeríamos el 11.1625%, que al trabajar en valores tan pequeños realmente arrastraríamos un error muy grande.

Por esta razón, la solución que finalmente se adopta para este problema es realizar una interpolación lineal de los datos, obteniendo un valor de tensión conociendo los valores extremos de un intervalo, mediante una recta.

En nuestro caso, los valores extremos del intervalo serán por un lado el valor más próximo por arriba del buscado y el valor más próximo por abajo. Para los datos de ejemplo anteriores, como buscamos un porcentaje de 9.3375%

escogeríamos como valores del intervalo al 11.1625% por arriba y al 7.0125% por abajo. Realizando la interpolación lineal, obtenemos un valor de tensión para el porcentaje buscado mucho más exacto. Mediante la siguiente gráfica podemos ver como es el cálculo de la interpolación lineal.

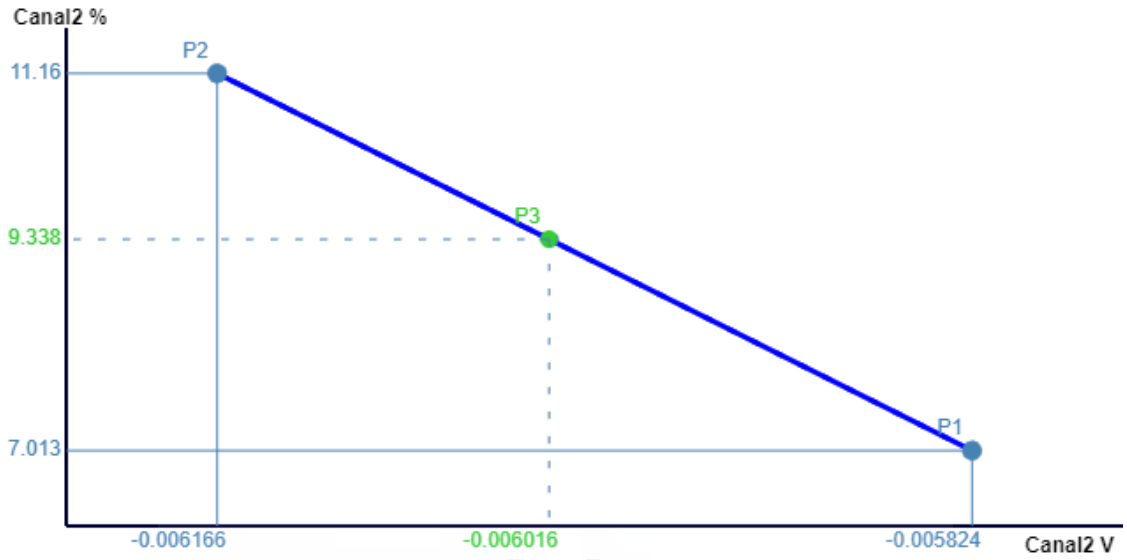


Figura 85. Representación interpolación lineal.

Observamos como a partir del intervalo de valores conocidos (en azul) obtenemos el valor que buscamos (en verde). Comprobamos como el valor obtenido matemáticamente coincide con el que obtenemos a partir del código de Matlab.

$$V_{60\%} = V_{ProxArriba} + (PorcentBuscado - PorcentAbajo) * \frac{(V_{ProxAbajo} - V_{ProxArriba})}{(PorcentArriba - PorcentAbajo)} \quad (18)$$

Para nuestro caso de ejemplo, tenemos:

$$\boldsymbol{V_{60\%}} = -0.005824 + (9.3375 - 7.0125) * \frac{\left(-0.006166 - (-0.005824)\right)}{(11.1625 - 7.0125)} = \boldsymbol{-0.0060157\ V} \quad (19)$$

Con el código obtenemos el mismo valor, como podemos comprobar en la variable de la interpolación:

```
interpolacionV0a60 =

   -0.00601572652296338
```

Figura 86. Valor obtenido con Matlab interpolación ejemplo.

Este planteamiento nos sirve completamente para el caso del nivel de tensión en V0 y también en el nivel de tensión V1.

Esta solución para el cálculo de σ0 y σ1 conlleva unos posibles errores similares al del comienzo, ya que para poder realizar la interpolación lineal debemos ser capaces de encontrar un valor próximo por arriba y otro valor próximo por abajo. Hay situaciones en las que se puede dar el caso de no obtener mediante el código de programación valores cercanos o por arriba o por abajo, para ello hay que considerar una serie de situaciones como hemos comentado en el apartado 2.7.3.2 Error falta valor para la interpolación de σ1 y σ0 para obtener el valor más exacto posible.

Una vez hemos obtenido el script definitivo sin errores, somos capaces de obtener una tabla con resultado de todos los valores que nos interesan. Al ejecutar el script en una carpeta de medidas, podemos ver en la ventana de comandos de Matlab como se van realizando los cálculos.

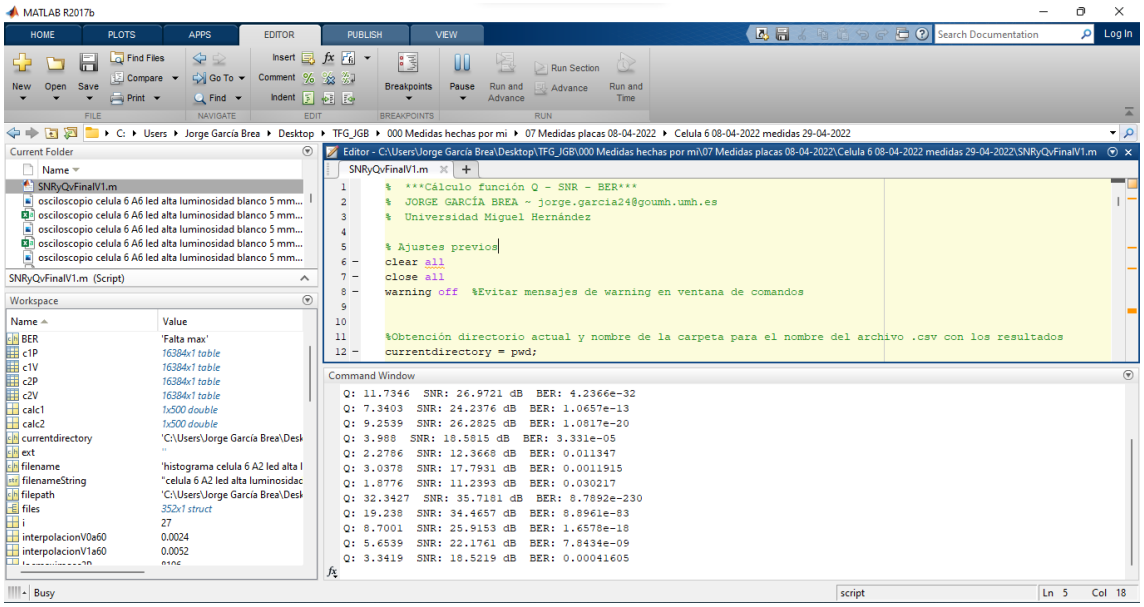


Figura 87. Ejemplo ejecución script en Matlab.

Todos estos cálculos se van almacenando tal y como ya hemos comentado en una tabla, que al finalizar la ejecución del script se exporta tal y como nos interesa en la misma carpeta de medidas, en un archivo de tipo .csv con el nombre "Función Q, SNR y BER-" seguido del nombre de la carpeta de la célula, para poder archivar los resultados de forma correcta.

Figura 88. Captura ejemplo archivo exportado de tabla resultados.

El archivo contiene una tabla con las columnas que nos interesa: nombre del archivo, nivel V0, nivel V1, factor Q, factor Q en dB, SNR y BER.

A	B	C	D	E	F	G
Archivo	V0	V1	Q	Q_dB	SNR_dB	BER
celula 6 A6 led alta luminosidad blanco 5 mm 3,3 V 1 KHz 0 cm	-2.9908e-05	0.039346	3.852	11.7138	26.5425	5.8575e-05
celula 6 A6 led alta luminosidad blanco 5 mm 3,3 V 1 KHz 1 cm	-0.0048235	0.026335	9.6269	19.6697	29.5357	3.079e-22
celula 6 A6 led alta luminosidad blanco 5 mm 3,3 V 1 KHz 2 cm	-0.0020575	0.017801	23.5286	27.4319	34.2106	1.039e-122
celula 6 A6 led alta luminosidad blanco 5 mm 3,3 V 1 KHz 3 cm	-0.0027156	0.0096107	13.1498	22.3784	25.5964	8.5346e-40
celula 6 A6 led alta luminosidad blanco 5 mm 3,3 V 1 KHz 4 cm	-0.00031881	0.0082411	10.4284	20.3644	23.4388	9.195e-26
celula 6 A6 led alta luminosidad blanco 5 mm 3,3 V 1 KHz 5 cm	0.001024	0.0071872	7.9984	18.06	24.9425	6.303e-16
celula 6 A6 led alta luminosidad blanco 5 mm 3,3 V 1 KHz 6 cm	-0.00034556	0.0041056	8.8871	18.9752	24.2709	3.1363e-19
celula 6 A6 led alta luminosidad blanco 5 mm 3,3 V 1 KHz 7 cm	-3.1605e-06	0.0034208	7.1786	17.1208	23.7416	3.5204e-13
celula 6 A6 led alta luminosidad blanco 5 mm 3,3 V 1 KHz 8 cm	0.001024	0.0037632	4.7983	13.6218	25.2187	7.9991e-07
celula 6 A6 led alta luminosidad blanco 5 mm 3,3 V 1 KHz 9 cm	-3.1605e-06	0.0017088	2.0501	6.2356	9.5699	0.020176
celula 6 A6 led alta luminosidad blanco 5 mm 3,3 V 1 KHz 10 cm	0.00068163	0.0020512	1.2974	2.2618	10.3823	0.097239
celula 6 A6 led alta luminosidad blanco 5 mm 3,3 V 2 KHz 0 cm	-0.00066121	0.042138	74.9939	37.5005	42.7181	0
celula 6 A6 led alta luminosidad blanco 5 mm 3,3 V 2 KHz 1 cm	-0.0035555	0.028972	41.498	32.3605	40.0339	0
celula 6 A6 led alta luminosidad blanco 5 mm 3,3 V 2 KHz 2 cm	-0.0016082	0.017224	24.8086	27.8921	30.9811	3.6173e-136
celula 6 A6 led alta luminosidad blanco 5 mm 3,3 V 2 KHz 3 cm	-0.0015814	0.0097177	13.8653	22.8386	25.211	5.1383e-44
celula 6 A6 led alta luminosidad blanco 5 mm 3,3 V 2 KHz 4 cm	0.00036598	0.0085835	8.1768	18.2517	25.2513	1.4574e-16
celula 6 A6 led alta luminosidad blanco 5 mm 3,3 V 2 KHz 5 cm	0.00099728	0.0064756	8.2558	18.3351	23.5162	7.5473e-17
celula 6 A6 led alta luminosidad blanco 5 mm 3,3 V 2 KHz 6 cm	2.3587e-05	0.0041323	7.6786	17.7056	24.784	8.0435e-15
celula 6 A6 led alta luminosidad blanco 5 mm 3,3 V 2 KHz 7 cm	-0.00031881	0.0024204	3.5383	10.9758	13.1996	0.00020139
celula 6 A6 led alta luminosidad blanco 5 mm 3,3 V 2 KHz 8 cm	0.00099728	0.0030517	2.9659	9.4431	16.7856	0.001509
celula 6 A6 led alta luminosidad blanco 5 mm 3,3 V 2 KHz 9 cm	-0.00066121	0.0013932	2.2543	7.0603	11.3762	0.012088

Funcion Q, SNR y BER-Celula 6 0 | Hoja1

Figura 89. Captura ejemplo tabla de resultados cálculos Matlab.

Como observamos el resultado obtenido de la ejecución del script de Matlab es el esperado, en este caso el cálculo de los parámetros que hemos mencionado y su exportación en una tabla, que nos pueden resultar de gran utilidad a la hora de analizar la calidad de la señal recibida.

4. CONCLUSIONES Y FUTURAS LÍNEAS DE INVESTIGACIÓN

En este apartado vamos a comentar las conclusiones que podemos extraer de este TFG una vez finalizado, analizando el logro de los distintos objetivos que se planteaban en él. Por otra parte, introduciremos unas posibles líneas de investigación para mejorar a futuro.

4.1 CONCLUSIONES DEL PROYECTO

Al comienzo de este TFG hemos establecido los objetivos que se buscaban, era en primer lugar, la automatización del proceso de toma de medidas a distintas frecuencias, de la respuesta de una célula orgánica fotovoltaica empleando un osciloscopio digital y, por otro lado, mediante el software Matlab obtener un script para representar el diagrama de ojo de cada una de las mediciones. Por último, también mediante Matlab obtener una tabla con los distintos parámetros de calidad de la señal.

En primer lugar, hemos conseguido entender los conceptos teóricos claves sobre los que giran todo el proyecto: la comunicación con luz visible (VLC), las placas fotovoltaicas orgánicas, los parámetros de la señal como SNR, BER o el factor Q y la representación del diagrama de ojo.

Hemos sido capaces de llevar a cabo la automatización de la toma de medidas a distintas frecuencias con el software del osciloscopio digital, siendo esto el principal objetivo de este TFG, ya que supone un gran avance para el estudio de la respuesta de las células fotovoltaicas orgánicas. De esta forma, reducimos de una manera muy considerable el tiempo que se dedica a la medición y exportación de los datos. Además, conseguimos que los datos exportados sean más exactos al ajustar la señal de manera correcta en el osciloscopio, lo que harán que los cálculos que se realicen con esos datos sean de mayor calidad.

En cuanto a la reducción del tiempo en el proceso de medición y exportación de los datos que hemos comentado anteriormente, podemos observar en el siguiente diagrama como hemos conseguido una disminución de un 98.5%:

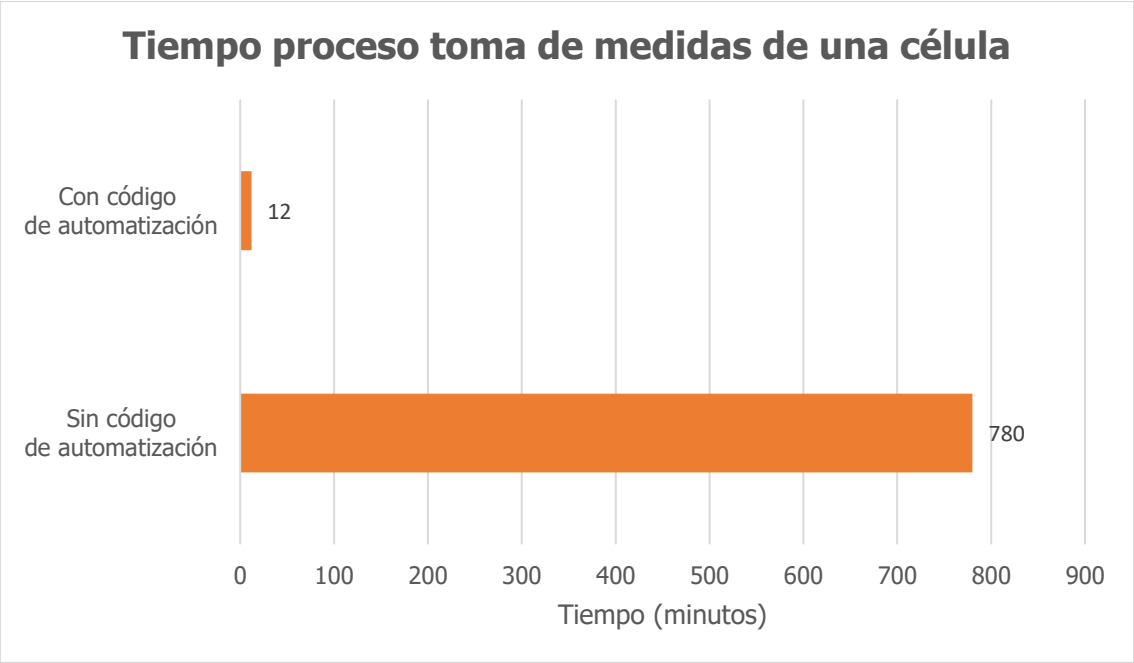


Figura 90. Gráfica comparación tiempo dedicado al proceso de toma de medidas de una célula.

Por otro lado, hemos sido capaces de llevar a cabo también otro de los objetivos del TFG, consiguiendo mediante un script en Matlab representar y exportar los diagramas de ojos de las distintas mediciones que se obtienen del osciloscopio. Pudiendo obtener de un vistazo rápido una idea de la calidad de la señal recibida en nuestra célula.

De igual forma, hemos conseguido el último objetivo del proyecto, desarrollando un código mediante el cual somos capaces de calcular parámetros de calidad de la señal que resultan de gran interés en la investigación. Cabe destacar, que este código supone una mejora con respecto a otros que se empleaban con anterioridad, ya que conseguimos seleccionar los máximos reales para los niveles de tensión V0 y V1. Esto sucede al trabajar con un código creado por y para este tipo de mediciones, y no ser una función predeterminada del software Matlab, las cuales ponen una serie de restricciones a los valores de entrada que hacen que los resultados finales no sean lo más exactos posible.

Podemos concluir, que la automatización de los procesos ya sean las mediciones de las células, la representación del diagrama de ojo o el cálculo de los parámetros, hacen avanzar la investigación a una mayor velocidad, dedicando el tiempo y esfuerzo en tareas intelectuales y no en procesos mecánicos.

4.2 LÍNEAS FUTURAS

Una vez hemos finalizado el proyecto y analizado los objetivos conseguidos, se abren posibles líneas de investigación futura.

En primer lugar, se cuenta con la posibilidad de avanzar en la investigación realizando la comunicación mediante luz visible en otros medios, como puede ser el agua.

Por otro lado, una posibilidad que se abre a partir de este TFG es emplear redes neuronales en lugar del filtro de Kalman, para predecir el estado futuro de ciertas variables. El filtro de Kalman, como hemos visto, es una solución buena empleando la tendencia de los datos, pero mediante las redes neuronales podríamos ser capaces de "enseñar" al código para que pueda prever el estado futuro de una manera más eficiente.

Otra posible línea de investigación a raíz de agilizar la toma de medidas es, conseguir mejorar los receptores haciendo que se pueda transmitir a frecuencias más altas, lo que conllevaría una mayor tasa binaria y permitiría transmitir imágenes empleando la tecnología Lifi.

Por último, en el mismo sentido que lo anterior, realizando mejoras en el material de fabricación de los receptores que se emplean, se podrían obtener mejores resultados.

5. ANEXOS

5.1 ANEXO 1: CÓDIGO COMPLETO WAVEFORMS DE AUTOMATIZACIÓN TOMA DE MEDIDAS

```
/**Automatización medición células
JORGE GARCÍA BREA
jorge.garcia24@goumh.umh.es
*/

/**DATOS CÉLULA */
var CELULA = "CEL"
var ANODO = "A"; //Ánodo que se mide
var COLOR = "blanco";
var TAMAÑO = "X";
var DISTANCIA = D ; // en cm
var VOLTAJE = Wavegen1.Channel1.Simple.Amplitude.text ; //Coge el dato de
Wavegen


//Limpiar consola de salidas anteriores
clear();
wait(0.1);   //esperar 0.1 segundos

//Encabezado del Script para la consola
print("  ---MEDICIÓN CÉLULAS---  ");
print();

//Comprobar si hay un dispositivo conectado(Discovery2, etc)
var conexion = Device.isConnected();
var nombreDispo = Device.name;

print("¿El dispositivo está conectado?: ", conexion);
if( conexion == true){
print("Nombre dispositivo conectado: ", nombreDispo);
}
if( conexion == false) {
print("No hay dispositivo conectado");
}
print();   //Espacio para ordenar la salida por pantalla


//CÉLULA y ANODO que se está midiendo
print("Célula: " ,CELULA,"-","Ánodo: ",ANODO);
print();


//Definición de VARIABLES iniciales
//Array con frecuencias a medir
var frecuencias =[1,2,5,10,20,50,100,200] //en KHz

frecu0=1; //frecuencia inicio (KHz)
```

```
frecumax= 200; //frecuencia máxima a la que queremos medir (KHz)

//Tamaño array frecuencias
var n = frecuencias.length;

//Iniciamos Wavegen
Wavegen1.run()
//Establecemos unidades de frecuencia en KHz
Wavegen1.Channel1.Simple.Frequency.text = '1 khz';

//Iniciamos osciloscopio
Scope1.run()


//CALCULO INICIAL (busco offset inicial y range inicial)
Wavegen1.Channel1.Simple.Frequency.text = '1 kHz';
Scope1.Time.Base.text = '1000';

Scope1.Channel1.Offset.text = '0 V';
Scope1.Channel1.Range.text = '1 V/div';

wait(0.2) //NECESARIO PARA CORRECTO FUNCIONAMIENTO, o si no coge valor
anterior del picopico
var PicoPicoCh2 = Scope1.Channel2.measure('Peak2Peak');

//Para cálculo del offset
var high = Scope1.Channel2.measure('High');     //medida valor máximo(High)
var low = Scope1.Channel2.measure('Low');       //medida valor mínimo(Low)
var amp = Scope1.Channel2.measure('Amplitude'); //medida valor
amplitud(Amplitude)

var offset1 = - ( (high + low)/2);      //Cálculo Offset para señal centrada en 0
var offset = offset1 + amp ;           //Valor OFFSET para señal sobre 0




//Bucle para cambiar la frecuencia y realizar medidas en cada una de ellas
for(var j = 0 ; j < n ; j++) {



    var frequency = frecuencias[j];
    Wavegen1.Channel1.Simple.Frequency.text = frequency; //Cambiar frecuencia
    print("Distancia: ",DISTANCIA,"cm ->","Frecuencia: ",frequency," KHz"); //Imprime
por pantalla la distancia y la frecuencia una a una (comprobación de las frecuencias
que toma)
```

```
//Ajustar TIME BASE señal osciloscopio
Scope1.Time.Position.text= '0 s';   //Ajustar por defecto POSITION a 0 seg
Scope1.Time.Base.text = '1 us/div'; //Establecer unidades de base de tiempos a useg

//Array con Bases de tiempos
var timeBase =[1000,500,200,100,50,20,10,5]; //en us/div
var base = timeBase[j];
Scope1.Time.Base.text = base;       //cambiar base de tiempos del osciloscopio




//Ajustar valores Canal 1 (constantes)
   //Canal 1 (Offset)
Scope1.Channel1.Offset.text = '0 V';
   //Canal 1 (Range)
Scope1.Channel1.Range.text = '1 V/div';



//Ajustar valores Canal 2 (variables)
   //Canal 2 (Range)
   //Medida Tensión Pico-Pico

var PicoPicoCh2 = Scope1.Channel2.measure('Peak2Peak');


   //Ajustar Range en función de tensión pico-pico (PicoPicoCh2)   --FILTRO  Kalman--
   if((PicoPicoCh2 <= 0.004)){
      Scope1.Channel2.Range.text= '1 mV/div';
   }
   if((PicoPicoCh2 > 0.004) && (PicoPicoCh2 <= 0.011)){
      Scope1.Channel2.Range.text= '2 mV/div';
   }
   if((PicoPicoCh2 > 0.011) && (PicoPicoCh2 <= 0.025)){
      Scope1.Channel2.Range.text= '5 mV/div';
   }
   if((PicoPicoCh2 > 0.025) && (PicoPicoCh2 <= 0.050)){
      Scope1.Channel2.Range.text= '10 mV/div';
   }
   if((PicoPicoCh2 > 0.050) && (PicoPicoCh2 <= 0.08)){
      Scope1.Channel2.Range.text= '20 mV/div';
   }
   if(PicoPicoCh2 > 0.08 ){
      Scope1.Channel2.Range.text= '50 mV/div';
   }
```

```
    //Canal 2 (Offset)
//Calculo valor Offset

var high = Scope1.Channel2.measure('High');     //medida valor máximo(High)
var low = Scope1.Channel2.measure('Low');       //medida valor mínimo(Low)
var amp = Scope1.Channel2.measure('Amplitude'); //medida valor
amplitud(Amplitude)


var offset1 = - ( (high + low)/2);      //Cálculo Offset para señal centrada en 0
var offset = offset1 + amp ;            //Valor OFFSET para señal sobre 0

Scope1.Channel2.Offset.value = offset; //Establecer valor OFFSET



//EXPORTAR datos OSCILOSCOPIO .csv y .png
Scope1.wait()
    //Osciloscopio en CSV
Scope1.Export("~/Desktop/DescargaWF/osciloscopio celula "+CELULA+"
A"+ANODO+" led alta luminosidad "+COLOR+" "+TAMAÑO+" mm "+VOLTAJE+" "
+frecuencias[j]+ " KHz " +DISTANCIA+ " cm.csv")
    //Osciloscopio en PNG
Scope1.Export("~/Desktop/DescargaWF/osciloscopio celula "+CELULA+"
A"+ANODO+" led alta luminosidad "+COLOR+" "+TAMAÑO+" mm "+VOLTAJE+" "
+frecuencias[j]+ " KHz " +DISTANCIA+ " cm.png")



//Exportar datos HISTOGRAMA .csv y .png
    //Histograma en CSV
Scope1.Export("~/Desktop/DescargaWF/histograma celula "+CELULA+"
A"+ANODO+" led alta luminosidad "+COLOR+" "+TAMAÑO+" mm "+VOLTAJE+" "
+frecuencias[j]+ " KHz " +DISTANCIA+ " cm.csv", "Histogram")
    //Histograma en PNG
Scope1.Export("~/Desktop/DescargaWF/histograma celula "+CELULA+"
A"+ANODO+" led alta luminosidad "+COLOR+" "+TAMAÑO+" mm "+VOLTAJE+" "
+frecuencias[j]+ " KHz " +DISTANCIA+ " cm.png", "Histogram")



//Exportar datos MEDIDAS canal1 y canal2 .csv

    /**HAY QUE SELECCIONAR LAS MEDIDAS(measurements) QUE NECESITAMOS EN
EL OSCILOSCOPIO*/

Scope1.Export("~/Desktop/DescargaWF/medidas celula "+CELULA+" A"+ANODO+"
led alta luminosidad "+COLOR+" "+TAMAÑO+" mm "+VOLTAJE+" "
+frecuencias[j]+ " KHz " +DISTANCIA+ " cm.csv", "Measurements")


}
```

5.2 ANEXO 2: CÓDIGO COMPLETO MATLAB REPRESENTACIÓN DIAGRAMA DE OJO

```
%  ***Representación DIAGRAMA DE OJO***
%  JORGE GARCÍA BREA ~ jorge.garcia24@goumh.umh.es
%  Universidad Miguel Hernández

%% Ajustes previos
clear all
close all
warning off  %Evitar mensajes de warning en ventana de comandos



%% Leer archivos csv de carpeta
files = dir('o*.csv');
table_data = cell(1, numel(files));
if( numel(files)==0)
    error= 'Error: No hay archivos ".csv" para representar en la carpeta.';
    disp(error)
else
end

%% Crear carpeta para guardar resultados
currentdirectory = pwd;
[filepath,name,ext] = fileparts(pwd);
nombreCarpetaMedidas = name;

newFolder1 = strcat("Diagramas de ojo ", nombreCarpetaMedidas);
newFolder = convertStringsToChars(newFolder1);

%Te pide seleccionar DONDE crear la carpeta para descargar resultados
route = uigetdir('*.*');

mkdir(fullfile(route,newFolder));




for i=1:numel(files)
    filename = files(i).name;
    table_data{i} = readtable(filename);


%% Obtener   frecuencia
parte1 = strfind(filename, 'KHz');
parte2 = strfind(filename, 'V');
strFrecu = filename(parte2 + 1: parte1-1);

frecuVal=str2double(strFrecu);  %Valor de la frecuencia sin unidades
frecu = frecuVal*1000;          %Valor de la frecuencia correctamente en Hz
```

```
 %% Definiciones
T= (1/frecu) ;  %Periodo en funcion de la frecuencia
offset= 0;

%% Datos obtenidos osciloscopio
%Obtenemos datos
%tiempos= data(:,1); %Tiempo
v1 = table_data{1, i} (:,2);     %Canal 1
v2 = table_data{1, i}  (:,3);     %Canal 2


%Pasar tabla a vector (para poder operar con los valores)
A = table2array(v1);    %Canal 1
B = table2array(v2);    %Canal 2

%% Representación diagrama de ojo
eyediagram(B, 16384, T, offset);  %Canal 2
% eyediagram(A, 16384, T, offset); %Canal 1

ylim('auto');

%% Descargar png diagrama de ojo
%carp='C:\Users\Jorge García Brea\Desktop\TFG_JGB\11 DescargaMatlab';
carp = strcat(route,'\', newFolder);
saveas(gca,fullfile(carp, strcat('Diagrama de Ojo- ', filename,'.png')));

clc
display(i);
close

end

close  all  %Cerrar figuras
```

5.3 ANEXO 3: CÓDIGO COMPLETO MATLAB CÁLCULO SNR, Q Y BER

```
%  ***Cálculo factor Q - SNR - BER***
%  JORGE GARCÍA BREA ~ jorge.garcia24@goumh.umh.es
%  Universidad Miguel Hernández

% Ajustes previos
clear all
close all
warning off  %Evitar mensajes de warning en ventana de comandos


%Obtención directorio actual y nombre de la carpeta para el nombre del archivo .csv
con los resultados
currentdirectory = pwd;
[filepath,name,ext] = fileparts(pwd);
nombreCarpetaMedidas = name;

%% Leer archivos .csv de carpeta y error si no hay
files = dir('h*.csv');
table_data = cell(1, numel(files));
if( numel(files)==0)
    error= 'Error: No hay archivos ".csv" para representar en la carpeta.';
    disp(error)
else
end


%% Bucle for
for i=1:numel(files)
    filename = files(i).name;
    table_data{i} = readtable(filename);

 %% Datos obtenidos por columnas del archivo
 %CANAL 1
c1V= table_data{1, i} (:,1);    %Tensiones V canal 1
c1P= table_data{1, i} (:,2);    %Porcentajes canal 1
%CANAL 2
c2V = table_data{1, i} (:,3);     %Tensiones V canal 2
c2P = table_data{1, i} (:,4);     %Porcentajes canal 2

%Pasar tabla a vector, para poder trabajar con los valores
vectorc1V = table2array(c1V);    %Tension canal 1
vectorc1P = table2array(c1P);    %Porcentaje canal 1
vectorc2V = table2array(c2V);    %Tension canal 2
vectorc2P = table2array(c2P);    %Porcentaje canal 2

%% Obtener picos máximos porcentajes Canal 2

[valmaximosc2P , locmaximosc2P]=findpeaks(vectorc2P,'SortStr','descend');
%máximos locales y su localizacion en la tabla ordenados de mayor a menor
```

```
numeroDeMaximos = length(valmaximosc2P);  %número de maximos en el
histograma del canal 2

%Condición: Si no hay al menos dos máximos, no podemos tener V0 y V1, por tanto
FALTA MAX
if numeroDeMaximos < 2
        V0='Falta max';
        V1='Falta max';
        Q='Falta max';
        Qdb='Falta max';
        SNR='Falta max';
        BER='Falta max';
else

%Valor máximo en el 0 y su porcentaje
V0 = vectorc2V(locmaximosc2P(1,1) , 1);
V0porcent = valmaximosc2P(1,1);
locV0 = find(vectorc2V == V0);

%Calculos para obtener el valor de v1
valoresVOLTmaximosc2P = vectorc2V(locmaximosc2P(:,1) , 1);

%Valor máximo en el 1 y su porcentaje
V1 = max(valoresVOLTmaximosc2P);

locV1 = find(vectorc2V == V1);
V1porcent = vectorc2P(locV1  , 1);


%% CÁLCULOS PARA SIGMA0 Y SIGMA1

%Valor porcentajes a 60% del máximo
V0porcent60 = 0.6 * V0porcent;
V1porcent60 = 0.6 * V1porcent;


%Busco valor más cercano por arriba y valor más cercano por abajo, para hacer la
interpolación

%Valor para 0
%Valores más cercanos al buscado
calc1 = knnsearch(vectorc2P, V0porcent60, 'k', 500);
vecinosV0porcent60 = vectorc2P(  calc1(1, find(calc1 >= locV0)) ,1) ;

vecinosV0porcent60ARRIBA = vecinosV0porcent60( find( vecinosV0porcent60 >=
V0porcent60 , 1)  , 1);

if length(vecinosV0porcent60ARRIBA) < 1
    locV0porcent60ARRIBA = min(find(vectorc2P == vecinosV0porcent60ARRIBA));
    valorVOLTvecinosV0ARRIBA = vectorc2V(locV0porcent60ARRIBA  , 1);
    vecinosV0porcent60ARRIBA = vectorc2P( locV0porcent60ARRIBA  ,1);
```

```
else

    locV0porcent60ARRIBA = min(find(vectorc2P == vecinosV0porcent60ARRIBA));
    valorVOLTvecinosV0ARRIBA = vectorc2V(locV0porcent60ARRIBA , 1);
end

vecinosV0porcent60ABAJO = vecinosV0porcent60( find( vecinosV0porcent60 <
V0porcent60 , 1 )  , 1);
if length(vecinosV0porcent60ABAJO) < 1
    vecinosV0porcent60ABAJO = vecinosV0porcent60;
    locV0porcent60ABAJO = min(find((vectorc2P == vecinosV0porcent60ABAJO)));
    valorVOLTvecinosV0ABAJO = vectorc2V(locV0porcent60ABAJO + 1 , 1);
    vecinosV0porcent60ABAJO =  vectorc2P(locV0porcent60ABAJO + 1 , 1);
else

locV0porcent60ABAJO = min(find((vectorc2P == vecinosV0porcent60ABAJO)));
valorVOLTvecinosV0ABAJO = vectorc2V(locV0porcent60ABAJO , 1);
end

if (vecinosV0porcent60ABAJO ==  vecinosV0porcent60ARRIBA)
    interpolacionV0a60 = vectorc2V(locV0porcent60ABAJO , 1);
else
    interpolacionV0a60 = interp1 ([vecinosV0porcent60ABAJO
vecinosV0porcent60ARRIBA],[valorVOLTvecinosV0ABAJO
valorVOLTvecinosV0ARRIBA], V0porcent60);
end

%Valor para nivel 1
calc2 = knnsearch(vectorc2P, V1porcent60, 'k', 500);
vecinosV1porcent60 = vectorc2P(  calc2(1, find(calc2 >= locV1)) ,1) ;

vecinosV1porcent60ARRIBA = vecinosV1porcent60( find( vecinosV1porcent60 >=
V1porcent60 , 1)  , 1);
if length(vecinosV1porcent60ARRIBA) < 1
vecinosV1porcent60ARRIBA = vecinosV1porcent60;
locV1porcent60ARRIBA = max(find(vectorc2P == vecinosV1porcent60ARRIBA));
valorVOLTvecinosV1ARRIBA = vectorc2V(locV1porcent60ARRIBA , 1);

else

    locV1porcent60ARRIBA = max(find(vectorc2P == vecinosV1porcent60ARRIBA));
    valorVOLTvecinosV1ARRIBA = vectorc2V(locV1porcent60ARRIBA , 1);
end

vecinosV1porcent60ABAJO = vecinosV1porcent60( find( vecinosV1porcent60 <=
V1porcent60 , 1 )  , 1);
if length(vecinosV1porcent60ABAJO) < 1

    vecinosV1porcent60ABAJO = vecinosV1porcent60(1,1);
    locV1porcent60ABAJO = max(find(vectorc2P == vecinosV1porcent60ABAJO));
    valorVOLTvecinosV1ABAJO = vectorc2V(locV1porcent60ABAJO + 1 , 1);
    vecinosV1porcent60ABAJO =  vectorc2P(locV1porcent60ABAJO + 1 , 1);
```

```
else

    locV1porcent60ABAJO = max(find(vectorc2P == vecinosV1porcent60ABAJO));
    valorVOLTvecinosV1ABAJO = vectorc2V(locV1porcent60ABAJO , 1);
end

if (vecinosV1porcent60ABAJO ==  vecinosV1porcent60ARRIBA)
    interpolacionV1a60 = vectorc2V(locV1porcent60ABAJO , 1);
else
    interpolacionV1a60 = interp1 ([vecinosV1porcent60ABAJO
vecinosV1porcent60ARRIBA],[valorVOLTvecinosV1ABAJO
valorVOLTvecinosV1ARRIBA], V1porcent60);
end



%VALORES SIGMA
sigmaV0 = ( V0 - interpolacionV0a60 );
sigmaV1 = ( V1 - interpolacionV1a60 );

%% VALOR Q
Q = abs((V1-V0)./(sigmaV1+sigmaV0));
Qdb = 20*log10(Q);  %valor de Q en dBs

%% VALOR SNR
SNR = 10*log10(((V1).^2)./((sigmaV1).^2)); %en dB

%% VALOR BER
BER = (1/2)*erfc(Q/(sqrt(2)));

%% Mostrar por pantalla resultados
SOL = ['Q: ', num2str(Q),'  SNR: ', num2str(SNR),' dB','  BER: ', num2str(BER)  ];
disp(SOL);



end %Fin condicion numeroDeMaximos al menos 2


%% Crear tabla con resultados  NOMBRE , V0, V1, Q, Q(dB), SNR(dB), BER
parte1 = strfind(filename, 'celula');
parte2 = strfind(filename, 'cm');
strName = filename(parte1 : parte2 + 1);


filenameString = string(strName);
vectorSolu(i,1)= filenameString;    %Nombre archivo
vectorSolu(i,2)= V0;                %Valor V0
vectorSolu(i,3)= V1;                %Valor V1
vectorSolu(i,4)= Q;                 %Valor Q
```

```
vectorSolu(i,5)= Qdb;              %Valor Q(dB)
vectorSolu(i,6)= SNR;              %Valor SNR(dB)
vectorSolu(i,7)= BER;              %Valor BER


%Cambiamos los array a una tabla y añadimos nombre a cada columna
TablaSolu = array2table(vectorSolu);
TablaSolu.Properties.VariableNames{'vectorSolu1'} = 'Archivo';
TablaSolu.Properties.VariableNames{'vectorSolu2'} = 'V0';
TablaSolu.Properties.VariableNames{'vectorSolu3'} = 'V1';
TablaSolu.Properties.VariableNames{'vectorSolu4'} = 'Q';
TablaSolu.Properties.VariableNames{'vectorSolu5'} = 'Q_dB';
TablaSolu.Properties.VariableNames{'vectorSolu6'} = 'SNR_dB';
TablaSolu.Properties.VariableNames{'vectorSolu7'} = 'BER';


%% Exportar tabla SNR y Q en formato .csv
writetable(TablaSolu , strcat('Factor Q, SNR y BER- ', nombreCarpetaMedidas,'.csv'));



end   %Fin bucle for (de lectura de todos los archivos)

disp("----------------FIN----------------"); %Mostrar por pantalla que ha finalizado la
ejecución del script
```

5.4 ANEXO 4: CÓDIGO COMPLETO WAVEFORMS DE AUTOMATIZACIÓN TOMA DE MEDIDAS, AÑADIENDO POSIBILIDAD A FRECUENCIAS ALTAS (500 KHz, 1 MHz y 2 MHz).

```
/**Automatización medición células CASO 500K, 1M, 2M
JORGE GARCÍA BREA
jorge.garcia24@goumh.umh.es
*/

/**DATOS CÉLULA */
var CELULA = "CEL"
var ANODO = "A"; //Ánodo que se mide
var COLOR = "blanco";
var TAMAÑO = "X";
var DISTANCIA = D ; // en cm
var VOLTAJE = Wavegen1.Channel1.Simple.Amplitude.text ; //Coge el dato de
Wavegen


//Limpiar consola de salidas anteriores
clear();
wait(0.1);   //esperar 0.1 segundos

//Encabezado del Script para la consola
print("  ---MEDICIÓN PLACAS---  ");
print();

//Comprobar si hay un dispositivo conectado(Discovery2, etc)
var conexion = Device.isConnected();
var nombreDispo = Device.name;

print("¿El dispositivo está conectado?: ", conexion);
if( conexion == true){
print("Nombre dispositivo conectado: ", nombreDispo);
}
if( conexion == false) {
print("No hay dispositivo conectado");
}
print();   //Espacio para ordenar la salida por pantalla


//CÉLULA y ANODO que se esta midiendo
print("Célula: " ,CELULA,"-","Ánodo: ",ANODO);
print();


//Definición de VARIABLES iniciales
//Array con frecuencias a medir
var frecuencias =[1,2,5,10,20,50,100,200] //en KHz
```

```
frecu0=1; //frecuencia inicio (KHz)
frecumax= 200; //frecuencia máxima a la que queremos medir (KHz)
//Tamaño array frecuencias
var n = frecuencias.length;

//Iniciamos Wavegen
Wavegen1.run()
//Establecemos unidades de frecuencia en KHz
Wavegen1.Channel1.Simple.Frequency.text = '1 khz';

//Iniciamos osciloscopio
Scope1.run()


//CALCULO INICIAL (busco offset inicial y range inicial)
Wavegen1.Channel1.Simple.Frequency.text = '1 kHz';
Scope1.Time.Base.text = '1000';

Scope1.Channel1.Offset.text = '0 V';
Scope1.Channel1.Range.text = '1 V/div';

wait(0.2) //NECESARIO PARA CORRECTO FUNCIONAMIENTO, o si no coge valor
anterior del picopico
var PicoPicoCh2 = Scope1.Channel2.measure('Peak2Peak');

//Para cálculo del offset
var high = Scope1.Channel2.measure('High');     //medida valor máximo(High)
var low = Scope1.Channel2.measure('Low');       //medida valor mínimo(Low)
var amp = Scope1.Channel2.measure('Amplitude'); //medida valor
amplitud(Amplitude)

var offset1 = - ( (high + low)/2);      //Cálculo Offset para señal centrada en 0
var offset = offset1 + amp ;           //Valor OFFSET para señal sobre 0




//Bucle para cambiar la frecuencia y realizar medidas en cada una de ellas
for(var j = 0 ; j < n ; j++) {



    var frequency = frecuencias[j];
    Wavegen1.Channel1.Simple.Frequency.text = frequency; //Cambiar frecuencia
    print("Distancia: ",DISTANCIA,"cm ->","Frecuencia: ",frequency," KHz"); //Imprime
por pantalla la distancia y la frecuencia una a una (comprobación de las frecuencias
que toma)


//Ajustar TIME BASE señal osciloscopio
Scope1.Time.Position.text= '0 s';   //Ajustar por defecto POSITION a 0 seg
```

```
Scope1.Time.Base.text = '1 us/div'; //Establecer unidades de base de tiempos a useg

//Array con Bases de tiempos
var timeBase =[1000,500,200,100,50,20,10,5]; //en us/div
var base = timeBase[j];
Scope1.Time.Base.text = base;       //cambiar base de tiempos del osciloscopio



//Ajustar valores Canal 1 (constantes)
   //Canal 1 (Offset)
Scope1.Channel1.Offset.text = '0 V';
   //Canal 1 (Range)
Scope1.Channel1.Range.text = '1 V/div';



//Ajustar valores Canal 2 (variables)
   //Canal 2 (Range)
   //Medida Tensión Pico-Pico

var PicoPicoCh2 = Scope1.Channel2.measure('Peak2Peak');


   //Ajustar Range en función de tensión pico-pico (PicoPicoCh2)   --FILTRO  Kalman--
   if((PicoPicoCh2 <= 0.004)){
      Scope1.Channel2.Range.text= '1 mV/div';
   }
   if((PicoPicoCh2 > 0.004) && (PicoPicoCh2 <= 0.011)){
      Scope1.Channel2.Range.text= '2 mV/div';
   }
   if((PicoPicoCh2 > 0.011) && (PicoPicoCh2 <= 0.025)){
      Scope1.Channel2.Range.text= '5 mV/div';
   }
   if((PicoPicoCh2 > 0.025) && (PicoPicoCh2 <= 0.050)){
      Scope1.Channel2.Range.text= '10 mV/div';
   }
   if((PicoPicoCh2 > 0.050) && (PicoPicoCh2 <= 0.08)){
      Scope1.Channel2.Range.text= '20 mV/div';
   }
   if(PicoPicoCh2 > 0.08 ){
      Scope1.Channel2.Range.text= '50 mV/div';
   }


   //Canal 2 (Offset)
//Calculo valor Offset

var high = Scope1.Channel2.measure('High');     //medida valor máximo(High)
var low = Scope1.Channel2.measure('Low');       //medida valor mínimo(Low)
```

```
var amp = Scope1.Channel2.measure('Amplitude'); //medida valor
amplitud(Amplitude)
//print (offset)

var offset1 = - ( (high + low)/2);      //Cálculo Offset para señal centrada en 0
var offset = offset1 + amp ;            //Valor OFFSET para señal sobre 0

Scope1.Channel2.Offset.value = offset; //Establecer valor OFFSET



//EXPORTAR datos OSCILOSCOPIO .csv y .png (de 1K a 200K)

Scope1.wait()
   //Osciloscopio en CSV
Scope1.Export("~/Desktop/DescargaWaveForms/osciloscopio celula "+CELULA+"
A"+ANODO+" led alta luminosidad "+COLOR+" "+TAMAÑO+" mm "+VOLTAJE+" "
+frecuencias[j]+ " KHz " +DISTANCIA+ " cm.csv")
   //Osciloscopio en PNG
Scope1.Export("~/Desktop/DescargaWaveForms/osciloscopio celula "+CELULA+"
A"+ANODO+" led alta luminosidad "+COLOR+" "+TAMAÑO+" mm "+VOLTAJE+" "
+frecuencias[j]+ " KHz " +DISTANCIA+ " cm.png")



//Exportar datos HISTOGRAMA .csv y .png
   //Histograma en CSV
Scope1.Export("~/Desktop/DescargaWaveForms/histograma celula "+CELULA+"
A"+ANODO+" led alta luminosidad "+COLOR+" "+TAMAÑO+" mm "+VOLTAJE+" "
+frecuencias[j]+ " KHz " +DISTANCIA+ " cm.csv", "Histogram")
   //Histograma en PNG
Scope1.Export("~/Desktop/DescargaWaveForms/histograma celula "+CELULA+"
A"+ANODO+" led alta luminosidad "+COLOR+" "+TAMAÑO+" mm "+VOLTAJE+" "
+frecuencias[j]+ " KHz " +DISTANCIA+ " cm.png", "Histogram")



//Exportar datos MEDIDAS canal1 y canal2 .csv

   /**HAY QUE SELECCIONAR LAS MEDIDAS(measurements) QUE NECESITAMOS EN
EL OSCILOSCOPIO**/


Scope1.Export("~/Desktop/DescargaWaveForms/medidas celula "+CELULA+"
A"+ANODO+" led alta luminosidad "+COLOR+" "+TAMAÑO+" mm "+VOLTAJE+" "
+frecuencias[j]+ " KHz " +DISTANCIA+ " cm.csv", "Measurements")


}//fin bucle for (frecuencias hasta200K)
```

```
//-------------------------------------------------------------------------------------//


//MEDIDAS DE 500K, 1M, 2M
if(PicoPicoCh2 > 0.007){

//Array con frecuencias a medir
var frecuencias2 =[500,1000,2000] //en KHz
//Tamaño array frecuencias
var n2 = frecuencias2.length;
//Iniciamos Wavegen
Wavegen1.run()
//Establecemos unidades de frecuencia en KHz
Wavegen1.Channel1.Simple.Frequency.text = '1 khz';
//Iniciamos osciloscopio
Scope1.run()


//Bucle para cambiar la frecuencia
for(var k = 0 ; k < n2 ; k++) {

    var frequency2 = frecuencias2[k];
    Wavegen1.Channel1.Simple.Frequency.text = frequency2; //Cambiar frecuencia
    print("Distancia: ",DISTANCIA,"cm ->","Frecuencia: ",frequency2," KHz");
//Imprime por pantalla la distancia y la frecuencia una a una (comprobación de las
frecuencias que toma)


//Ajustar TIME BASE señal osciloscopio
Scope1.Time.Position.text= '0 s';   //Ajustar por defecto POSITION a 0 seg
Scope1.Time.Base.text = '1 us/div'; //Establecer unidades de base de tiempos a useg

//Array con Bases de tiempos
var timeBase2 =['2 us/div','1 us/div','500 ns/div']; //en us/div
var base2 = timeBase2[k];
Scope1.Time.Base.text = base2;       //cambiar base de tiempos del osciloscopio

//Ajustar valores Canal 1 (constantes)
    //Canal 1 (Offset)
Scope1.Channel1.Offset.text = '0 V';
    //Canal 1 (Range)
Scope1.Channel1.Range.text = '1 V/div';


//Ajustar valores Canal 2 (variables)
    //Canal 2 (Range)
    //Medida Tensión Pico-Pico

var PicoPicoCh2 = Scope1.Channel2.measure('Peak2Peak');
```

```
    //Ajustar Range en función de tensión pico-pico (PicoPicoCh2)   --FILTRO  Kalman
(predecimos futuro valor, conociendo la tendencia)--
    if((PicoPicoCh2 <= 0.004)){
        Scope1.Channel2.Range.text= '1 mV/div';
    }
    if((PicoPicoCh2 > 0.004) && (PicoPicoCh2 <= 0.011)){
        Scope1.Channel2.Range.text= '2 mV/div';
    }
    if((PicoPicoCh2 > 0.011) && (PicoPicoCh2 <= 0.025)){
        Scope1.Channel2.Range.text= '5 mV/div';
    }
    if((PicoPicoCh2 > 0.025) && (PicoPicoCh2 <= 0.050)){
        Scope1.Channel2.Range.text= '10 mV/div';
    }
    if((PicoPicoCh2 > 0.050) && (PicoPicoCh2 <= 0.08)){
        Scope1.Channel2.Range.text= '20 mV/div';
    }
    if(PicoPicoCh2 > 0.08 ){
        Scope1.Channel2.Range.text= '50 mV/div';
    }



    //Canal 2 (Offset)
//Calculo valor Offset         --Predecimos FUTURO VALOR, conociendo la tendencia--

var high = Scope1.Channel2.measure('High');     //medida valor máximo(High)
var low = Scope1.Channel2.measure('Low');       //medida valor mínimo(Low)
var amp = Scope1.Channel2.measure('Amplitude'); //medida valor
amplitud(Amplitude)
//print (offset)

var offset1 = - ( (high + low)/2);      //Cálculo Offset para señal centrada en 0
var offset = offset1 + amp ;           //Valor OFFSET para señal sobre 0

Scope1.Channel2.Offset.value = offset; //Establecer valor OFFSET



//EXPORTAR datos OSCILOSCOPIO .csv y .png (de 500k,1M,2M)

Scope1.wait()
    //Osciloscopio en CSV
Scope1.Export("~/Desktop/DescargaWaveForms/osciloscopio celula "+CELULA+"
A"+ANODO+" led alta luminosidad "+COLOR+" "+TAMAÑO+" mm "+VOLTAJE+" "
+frecuencias2[k]+ " KHz " +DISTANCIA+ " cm.csv")
    //Osciloscopio en PNG
Scope1.Export("~/Desktop/DescargaWaveForms/osciloscopio celula "+CELULA+"
A"+ANODO+" led alta luminosidad "+COLOR+" "+TAMAÑO+" mm "+VOLTAJE+" "
+frecuencias2[k]+ " KHz " +DISTANCIA+ " cm.png")
```

```
//Exportar datos HISTOGRAMA .csv y .png
    //Histograma en CSV
Scope1.Export("~/Desktop/DescargaWaveForms/histograma celula "+CELULA+"
A"+ANODO+" led alta luminosidad "+COLOR+" "+TAMAÑO+" mm "+VOLTAJE+" "
+frecuencias2[k]+ " KHz " +DISTANCIA+ " cm.csv", "Histogram")
    //Histograma en PNG
Scope1.Export("~/Desktop/DescargaWaveForms/histograma celula "+CELULA+"
A"+ANODO+" led alta luminosidad "+COLOR+" "+TAMAÑO+" mm "+VOLTAJE+" "
+frecuencias2[k]+ " KHz " +DISTANCIA+ " cm.png", "Histogram")


//Exportar datos MEDIDAS canal1 y canal2 .csv

    /**HAY QUE SELECCIONAR LAS MEDIDAS(measurements) QUE NECESITAMOS EN
EL OSCILOSCOPIO**/


Scope1.Export("~/Desktop/DescargaWaveForms/medidas celula "+CELULA+"
A"+ANODO+" led alta luminosidad "+COLOR+" "+TAMAÑO+" mm "+VOLTAJE+" "
+frecuencias2[k]+ " KHz " +DISTANCIA+ " cm.csv", "Measurements")


}//fin bucle for (frecuencias 500k,1M,2M)


}//fin condicion if medicion 500k,1M,2M
```

5.5 ANEXO 5: INFORME DEL PROYECTO A PARTIR DEL DIAGRAMA DE GANTT

TFG_JGB 12 jun. 2022

UMH http://

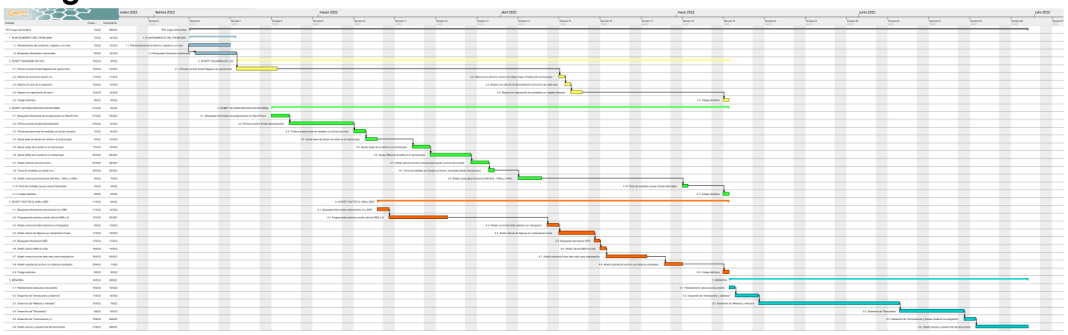
Encargado del proyecto	
Fechas de inicio y fin del proyecto	7 feb. 2022 - 30 jun. 2022
Progreso	0%
Tarea	39
Recursos	0

Cronograma pasos seguidos en el TFG

Tarea

Nombre	Fecha de inicio	Fecha de fin
TFG Jorge García Brea	7/2/22	29/6/22
1. PLANTEAMIENTO DEL PROBLEMA	7/2/22	14/2/22
1.1 Planteamiento del problema y objetivos con tutor	7/2/22	13/2/22
1.2 Búsqueda información relacionada	8/2/22	14/2/22
2. SCRIPT DIAGRAMA DE OJO	15/2/22	9/5/22
2.1 Primera versión Script Diagrama de ojo(errores)	15/2/22	21/2/22
2.2 Mejora de la primera versión con datos mejor tomados del osciloscopio	11/4/22	11/4/22
2.3 Mejora con ejes de la representación en función de cada caso	12/4/22	12/4/22
2.4 Mejora con exportación de resultados en carpeta deseada	13/4/22	14/4/22
2.5 Código definitivo	9/5/22	9/5/22
3. SCRIPT AUTOMATIZACIÓN WAVEFORMS	21/2/22	9/5/22
3.1 Búsqueda información de programación en WaveForms	21/2/22	23/2/22
3.2 Primera versión Script Automatización	24/2/22	6/3/22
3.3 Primera prueba toma de medidas con Script (errores)	7/3/22	8/3/22
3.4 Ajuste base de tiempo de señal en el osciloscopio	9/3/22	10/3/22
3.5 Ajuste rango de la señal en el osciloscopio	17/3/22	19/3/22
3.6 Ajuste Offset de la señal en el osciloscopio	20/3/22	26/3/22
3.7 Añadir cálculos previos al bucle para ajuste correcto de la señal	27/3/22	29/3/22
3.8 Toma de medidas con Script a la fecha (necesidad añadir frecuencias)	30/3/22	30/3/22
3.9 Añadir casos para frecuencia 500 KHz, 1 MHz y 2 MHz.	4/4/22	7/4/22
3.10 Toma de medidas nuevas células fabricadas	2/5/22	2/5/22
3.11 Código definitivo	9/5/22	9/5/22
4. SCRIPT FACTOR Q, SNR y BER	11/3/22	9/5/22
4.1 Búsqueda información teórica factor Q y SNR	11/3/22	12/3/22
4.2 Programación primera versión cálculo SNR y Q	13/3/22	22/3/22
4.3 Añadir corrección falta máximos en histograma	9/4/22	10/4/22
4.4 Añadir cálculo de Sigmas por interpolación lineal	11/4/22	16/4/22
4.5 Búsqueda información BER	17/4/22	17/4/22
4.6 Añadir cálculo BER al script	18/4/22	18/4/22
4.7 Añadir corrección error falta valor para interpolación	19/4/22	25/4/22
4.8 Añadir exportación archivo con tabla de resultados	29/4/22	1/5/22
4.9 Código definitivo	9/5/22	9/5/22
5. MEMORIA	10/5/22	29/6/22
5.1 Planteamiento estructura documento	10/5/22	10/5/22
5.2 Desarrollo de "Introducción y objetivos"	11/5/22	14/5/22
5.3 Desarrollo de "Material y métodos"	15/5/22	7/6/22
5.4 Desarrollo de "Resultados"	8/6/22	18/6/22
5.5 Desarrollo de "Conclusiones y futuras líneas de investigación"	19/6/22	20/6/22
5.6 Añadir anexos y repaso final del documento	21/6/22	29/6/22

Diagrama de Gantt



6. REFERENCIAS BIBLIOGRÁFICAS

[1] John G. Proakis. (2000). "Digital Communications". (4ª ed.). McGraw-Hill Education.

[2] Bernard Sklar. (2001). "Digital Communications: Fundamentals and Applications". (2ª ed.). Prentice Hall.

[3] John G. Proakis y Masoud Salehi. (2014). "Fundamentals of Communication Systems". (2ª ed.). Pearson.

[4] A. R. Ndjiongue, H. C. Ferreira y T. M. N. Ngatched. (2015). "Visible Light Communications (VLC) Technology". DOI: 10.1002/047134608X.W8267. Disponible en: https://www.researchgate.net/. [Último acceso: 2-05-2022].

[5] Gary Breed. (2005). "Analyzing signals using the eye diagram". Disponible en: https://www.highfrequencyelectronics.com/. [Último acceso: 26-04-2022].

[6] Houbing Song. (2012). "Model-Centric Approach to Discrete-Time Signal Processing for Dense Wavelength-Division Multiplexing Systems". Disponible en: https://www.researchgate.net/. [Último acceso: 22-04-2022].

[7] Mark Largent. (2012). "Optical receiver performance evaluation". Disponible en: https://www.researchgate.net/. [Último acceso: 16-03-2022].

[8] G. Barreto, N. Da Rosa y J. Freire. (2021). "Automatización de Medidas de Eficiencia Energética" Disponible en: https://www.colibri.udelar.edu.uy/. [Último acceso: 11-05-2022].

[9] A.Moreno Martín. (2019). "Automatización de una fuente de corriente-voltaje y un osciloscopio digital en el entorno de programación *LabView*". Disponible en: https://e-archivo.uc3m.es/. [Último acceso: 12-05-2022].

[10] R. Sarwar y otros. (2017). "Visible light communication using a solar-panel receiver". 16th International Conference on Optical Communications and Networks (ICOCN). DOI:

10.1109/ICOCN.2017.8121577. Disponible en: https://ieeexplore.ieee.org/. [Último acceso: 12-05-2022].

[11] Jhao-Ting Wu, Chi-Wai Chow, Yang Liu, Chin-Wei Hsu y Chien-Hung Yeh. (2017). "Performance enhancement technique of visible light communications using passive photovoltaic cell." Disponible en: https://www.sciencedirect.com/. [Último acceso: 12-05-2022].

[12] Fernando Rodríguez Mas. (2020). "Síntesis y caracterización de nanopartículas de CdS y PbS con recubrimiento mixto y su influencia en dispositivos optoelectrónicos".

[13] Maxim Integrated. (2008). "Optical Signal-to-Noise Ratio and the Q-Factor in Fiber-Optic Communication Systems". Disponible en: https://www.maximintegrated.com/. [Último acceso: 21-04-2022].

[14] Simon Haykin. (2001). "Kalman filtering and neural networks". John Wiley & Sons.

[15] Priya Shree Madhukar y L. B. Prasad. (2020). "State Estimation using Extended Kalman Filter and Unscented Kalman Filter". DOI: 10.1109/ICONC345789.2020.9117536.

[16] D. Yan, Q. Zhong y. Sui. (2014). "Spatial Kalman Filters and Spatial-Temporal Kalman Filters". 12th International Conference on Signal Processing (ICSP). DOI: 10.1109/ICOSP.2014.7015323.

[17] W. Freude, R. Schmogogrow, B. Nebedndahl, otros. (2012). "Quality metrics for optical signals: Eye diagram, Q-factor, OSNR, EVM and BER". 14th International Conference on Transparent Optical Networks (ICTON). DOI: 10.1109/ICTON.2012.6254380.

[18] L. Salamandra, N. Yaghoobi Nia, C. Fazolo, S. Maiello, G. Susanna, A. Pizzoleo, F. Matteocci, otros. (2019). "Perovskite photo-detectors (PVSK-PDs) for visible light communication". Disponible en: https://www.sciencedirect.com/. [Último acceso: 26-04-2022].

[19] S. Nurdah, O. Nur Samijayani, A. Syahriar, R. Ramdhani y R. Alamtaha. (2019). "Performance analisis of Q factor optical communication in free space optics and single mode fiber". Disponible en: https://repository.uai.ac.id/. [Último acceso: 19-04-2022].

[20] N. Y. Ko, G. Song, W. Youn, I. H. Choi y T. S. Kim. (2018). "Improvement of Extended Kalman Filter Using Invariant Extended Kalman Filter". 18th International Conference on Control, Automation and Systems (ICCAS).

[21] Emilio G. Moreno. (1999). "Automatización de procesos industriales". Disponible en: https://www.lalibreria.upv.es/. [Último acceso: 7-05-2022].

[22] Z. Hao, B. Li y X. Dang. (2019). "An improved Kalman filter positioning method in NLOS environment". DOI: 10.23919/JCC.2019.12.006.

[23] P. J. Hargrave. (1989) "A tutorial introduction to Kalman filtering". IEE Colloquium on Kalman Filters: Introduction, Applications and Future Developments.

[24] W. Pietruszkiewicz. (2010). "A comparison of nonlinear Kalman filtering applied to feed-forward neural networks as learning algorithms". IEEE 9th International Conference on Cyberntic Intelligent Systems. DOI: 10.1109/UKRICIS.2010.5898137.

[25] A. I. Ilieş, G. Chindriş y D. Pitică. (2020). "A Comparison between State of Charge Estimation Methods: Extended Kalman Filter and Unscented Kalman Filter". IEEE 26th International Symposium for Design and Technology in Electronic Packaging (SIITME). DOI: 10.1109/SIITME50350.2020.9292232.

[26] California Scientific. Inc. (2018). "Bit Error Rate vs. Q-Factor". Disponible en: http://www.californiascientific.com/. [Último acceso: 16-03-2022].

[27] Digilent. (s.f.). "Analog Discovery 2: 100MS/s USB Oscilloscope, Logic Analyzer and Variable Power Supply". Disponible en: https://digilent.com/. [Último acceso: 18-04-2022].

[28] Digilent. (s.f.). "Analog Discovery 2 Reference Manual". Disponible en: https://digilent.com/. [Último acceso: 14-06-2022].

[29] MathWorks. (s.f.). "MATLAB Documentation". Disponible en: https://es.mathworks.com/help/matlab/. [Último acceso: 9-05-2022].

[30] RF Café- Tech News & Resources. (s.f.). "Bit Error Rate". Disponible en: https://www.rfcafe.com/. [Último acceso: 21-04-2022].

[31] GanttProject. (2022). Disponible en: https://www.ganttproject.biz/. [Último acceso: 14-06-2022].

[32] Fritzing. (s.f.). "Tutorials - Fritzing". Disponible en: https://fritzing.org/learning/. [Último acceso: 18-05-2022].

[33] KalmanFilter. (s.f.). Disponible en: https://www.kalmanfilter.net/. [Último acceso: 10-04-2022].

[34] Diagrams.net. (s.f.). "Flowchart maker and online diagram software". Disponible en: https://app.diagrams.net/. [Último acceso: 9-06-2022].