UNIVERSIDAD DE VALLADOLID

ESCUELA TECNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIONES

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS ESPECÍFICAS DE LAS TELECOMUNICACIONES

MENCION EN SISTEMAS ELECTRÓNICOS

Diseño de un equipo de caracterización y test de dispositivos discretos

---

Autor:

Julián Arambarri García

Tutor:

Jesús Arias Álvarez

Agradecimientos

A mi familia que es la que me ha dado la oportunidad de estar hoy aquí, agradecer la paciencia que han tenido conmigo y que siempre han sido un ejemplo a seguir de esfuerzo y dedicación.

A mi tutor Jesús Arias Álvarez por todos los conocimientos que me ha transmitido a lo largo de estos años gracias a su dedicación y por haberme guiado y apoyado a lo largo de este proyecto.

A la escuela de Ingenieros de Telecomunicaciones de la UVA en especial a los profesores del departamento de electrónica por todo el conocimiento que me han transmitido sobre un mundo que me apasiona.

A mis amigos por todos los buenos momentos que hemos pasado y seguiremos pasando.

Resumen

Sin duda la invención del transistor ha sido el causante del increíble desarrollo de toda la tecnología y las telecomunicaciones en los últimos 100 años. Con el desarrollo de este proyecto se pretende ser capaz de caracterizar los principales tipos de transistores que se encuentran en la actualidad.

Para ello se desarrolla con un equipo que sea capaz de detectar que componente se esta analizando, obtener sus curvas características y mostrar todos estos datos obtenidos mediante un servidor Web a cualquier ordenador o Smartphone con acceso a internet.

Abstract

Without any doubt, the invention of the transistor has been the cause of the incredible development of all technology and telecommunications in the last 100 years. With the development of this project, the aim is to be able to characterize the main types of transistors that are currently found.

With this objetive in mind, a product is developed that is capable of detecting which component is being analyzed, obtaining its characteristic curves and displaying all this data through a web server to any computer or Smartphone with internet access.

Índice general

Resumen **4**
1 Introducción **9**
1.1 Objetivos 9
1.2 Fases del proyecto 9
1.3 Vista general del proyecto 10
2 ESPECIFICACION TECNICA DEL PROYECTO **11**
2.1 Estudio de mercado 11
2.2 Estudio de especificaciones 12
2.3 Estudio del sistema y selección de componentes 14
2.3.1 Microprocesador 15
2.3.2 DAC 16
2.3.3 POTENCIOMETRO 17
2.3.4 AMPLIFICADORES 18
2.3.5 MULTIPLEXOR 19
2.3.6 MEMORIA 20
2.3.7 ALIMENTACIÓN 20
2.3.8 OTROS 21
3 Desarrollo Hardware **22**
3.1 Diseño esquemático 22
3.1.1 Circuito módulo ESP-12F 28
3.1.2 Alimentación 28
3.1.3 DAC y amplificador 29
3.1.4 POTENCIOMETRO 29
3.1.5 Multiplexor 30
3.1.6 Memoria y leds 30
3.2 DISEÑO DE LA PCB 31
3.3 FABRICACIÓN PCB 36
3.4 MONTAJE DE COMPONENTES 36
3.5 Verificación del hardware 38
3.6 Errores de diseño 39
3.7 Bill of materials 41
3.8 Consumo eléctrico 43
3.9 Fuente de alimentación casera 44
4 Desarrollo Software **46**

4.1 Introducción al desarrollo software 46
4.2 Entorno de trabajo 46
4.3 Interconexiones 49
4.4 Funciones básicas 51
4.5 WiFi 52
4.6 Servidor Web 53
4.7 Desarrollo principal 55
4.7.1 Detección 57
4.7.2 Obtención de gráficas 57
4.7.3 Servidor web 58
5 Herramientas para la evaluación del producto 60
5.1 Monitor serie 60
5.2 Instrumentos electrónicos 61
5.3 GNUPLOT 61
5.4 Firefox 64
6 Resultados 66
6.1 Mecanizado del producto 66
6.2 Verificación del producto 69
6.3 Funcionamiento final 75
6.4 Futuras iteraciones 75
7 Documentación 77
7.1 Bibliografía 77
7.2 Apéndice: código del ESP-12F 79

Índice de figuras

Figura 2.1: Esquema lógico de un equipo que obtiene las curvas de un transistor. 12
Figura 2.2: Boceto conceptual del producto. 14
Figura 2.3: Bloques que forman el ESP-12F. 15
Figura 2.4: Módulo y pines ESP-12F. 16
Figura 2.5: Esquema lógico del MCP4728A0. 17
Figura 2.6: Esquema lógico del AD5263BRU200. 18
Figura 2.7: Circuito integrado MAX44243AUD+. 19
Figura 3.1: Hoja 1 del esquemático, módulo ESP8266. 23
Figura 3.2: Hoja 2 del esquemático, batería y alimentación 3.3V. 24
Figura 3.3: Hoja 3 del esquemático, electrónica analógica: DAC, amplificadores y potenciómetro digital. 25
Figura 3.4: Hoja 4 del esquemático, electrónica analógica: Multiplexor de entrada y buffer. 26
Figura 3.5: Hoja 5 del esquemático, alimentación de 12V y miscelánea. 27
Figura 3.6: pines Boot ESP-12F. 28
Figura 3.7: Huellas generadas. 32
Figura 3.8: Cara superior de la PCB diseñada. 34
Figura 3.9: Cara inferior de la PCB diseñada. 35
Figura 3.10: PCB con los componentes soldados. 37
Figura 3.11: PCB conversor USB-serie con los componentes soldados. 38
Figura 3.12: Una PCB boca arriba y otra boca abajo. 39
Figura 3.13: Conexionado de la UART. 40
Figura 3.14: Solución conexionado TX y RX. 40
Figura 3.15: Transistores rotados 120º para lograr la asignación de *pad* correcta. 41
Figura 3.16: Bill Of Materials. 42
Figura 3.17: Estimación del consumo y la autonomía 43
Figura 3.18: Fuente de alimentación lineal. 44
Figura 3.19: Fuente de alimentación lineal de la que se disponía. 45
Figura 3.20: Esquemático LM2595. 45
Figura 4.1: Extensión de PlataformIO. 47
Figura 4.2: Estructura de las carpetas del proyecto. 48
Figura 4.3: Archivo plarafforio.ini y salida del terminal después de usar la herramienta *Devices*. 48
Figura 4.4: Campos que componen las tramas del protocolo I2C. 50

Figura 4.5: Modos de funcionamiento de la interfaz WiFi del ESP-12F. 52
Figura 4.6: Código que permite conectarse al WiFi. 53
Figura 4.7: Inicialización del servidor Web. 54
Figura 4.8: Diagrama del programa principal. 56
Figura 5.1: Monitor serie mostrando datos del flujo del programa. 60
Figura 5.2: Gráfica realizada de los datos obtenidos de un JFET tipo N mediante GNUPlot. 62
Figura 5.3: Gráfica en 3 dimensiones del JFET tipo N donde se muestra que la Vgs permanece constante en cada grafo. 63
Figura 5.4: Gráficas de un transistor BJT tipo P, en la primera imagen se muestra el resultado final y en las otras dos imágenes en 3 dimensiones se observa como la tercera variable Ibase permanece constante. 63
Figura 5.5: Herramienta y consola de Firefox para desarrolladores Web. 65
Figura 6.1: PCB anclada a la carcasa inferior y batería pegada a la PCB. 66
Figura 6.2: Orificios que permiten la pulsación de los botones. 67
Figura 6.3: Imágenes del producto final. 68
Figura 6.4: Pagina web mostrada al conectarnos al servidor del producto. 70
Figura 6.5: Pagina Web mientras el equipo se encuentra midiendo, 1ºJFET tipo P y 2º BJT NPN. 71
Figura 6.6: Herramienta de red de Firefox donde se observa el retardo de algunas peticiones. 72
Figura 6.7: Medida completada de un JFET tipo P. 73
Figura 6.8: Medida completada de un BJT NPN. 74
Figura 6.9: Caracterización completada de un diodo Zener. 74

1 Introducción

1.1 Objetivos

Continuamente en los distintos diseños electrónicos que se realizan hoy en día se emplean transistores de instrumentación los cuales pueden realizar infinidad de funciones. Debido al desarrollo de los microprocesadores y la nanotecnología cada vez es más difícil poder verlos con nuestros propios ojos y por tanto poder comprobar que se cumplen los fundamentos teóricos de cada tipo de transistor.

Mediante los conocimientos y las habilidades desarrollados a lo largo de estos años de estudio se pretende diseñar un sistema autónomo e inteligente que sea capaz de distinguir el tipo de transistor que se pone a prueba, así como sus características y las gráficas correspondientes. Por lo tanto, se espera que el dispositivo sea capaz de realizar medidas en un gran abanico de tipos de transistores y enviar estas medidas y gráficas mediante un protocolo WiFi a un servidor web para que se puedan comprobar desde un Smartphone o un ordenador en los laboratorios de instrumentación.

1.2 Fases del proyecto

A continuación, con el fin de comprender de una manera más sencilla como se ha completado el proyecto se ha divido en apartados:

1. Introducción
2. Especificación técnica del proyecto
3. Desarrollo Hardware: diseño y construcción
4. Diseño Software
5. Herramientas para la evaluación del producto
6. Resultados
7. Documentación

1.3 Vista general del proyecto

Introducción: como se ha podido ver este apartado muestra el principal objetivo del proyecto y se describen brevemente los apartados que se van a seguir.

Especificación técnica del proyecto: como su propio nombre indica se establece una serie de objetos o retos que se desea que el equipo desarrollado cumpla y se empieza a trazar un plan para cumplirlos. Además, se realiza un estudio de mercado de para ver si existe algún producto similar.

Hardware: en este capítulo se realizan diversos diseños hasta obtener el diseño electrónico final junto con su captura esquemática para poder encargar la producción de la PCB. También se trata el tema de la selección de componentes, el montaje de la PCB con dichos componentes y su verificación.

Diseño Firmware: en este apartado se narra el proceso de realización del firmware el cual sea realizado en C++ empleando librerías de Arduino, así como los objetivos que este debe de cumplir y los diagramas de flujo que emplea para cumplirlos.

Herramientas para la evaluación del producto: en este apartado se describen los diferentes métodos que se han empleado para desarrollar y conseguir un correcto funcionamiento del equipo final.

Evaluación del producto: el este quinto capitulo se realizan una serie de pruebas al equipo para comprobar si se ha logrado cumplirlo objetivos que se establecieron. Estas pruebas comprobaran el funcionamiento en circunstancias normales de funcionamiento y en circunstancias en las que intencionadamente el usuario haya cometido un error, y de esta manera comprobar la robustez del equipo

Resultados: en este apartado se realizan una serie de pruebas al equipo para comprobar si se ha logrado cumplir los objetivos que se establecieron. Además, se muestran unas conclusiones y se ofrece una serie de pautas para un desarrollo futuro en una nueva iteración del equipo.

Documentación: por último, en este capitulo se muestra toda la bibliografía empleada y el código desarrollado

2 ESPECIFICACIÓN TÉCNICA DEL PROYECTO

2.1 Estudio de mercado

Se ha realizado una búsqueda de equipos similares que permitan realizar alguna de las tareas que es espera realizar con el equipo electrónico que se ha desarrollado.

En el aspecto comercial prácticamente no existen equipos a la venta con características o funciones similares, los pocos que existen son difíciles de encontrar y tienen una documentación muy limitada. Algunos de los equipos que podemos encontrar son el QT-2A, CH-012 y CA-4810A. Estos equipos no son capaces de diferenciar entre transistores, ni distinguir sus terminales. Además, salvo los equipos con un precio muy elevado como el CA-4810, estos necesitan un osciloscopio para mostrar la salida en el modo X-Y.

Fuera del campo comercial se pueden encontrar diversos proyectos individuales que han realizado un equipo electrónico similar [1], son más numerosos que los comerciales. Estos equipos al igual que los que se comercializan siguen sin ser capaces de distinguir entre transistores ni terminales. Para mostrar las curvas de los transistores algunos siguen utilizando un osciloscopio, otros emplean un pequeño panel LCD y por último otros, los más complejos se conectan a un ordenador mediante una interfaz USB a un programa instalado en el ordenador. Estos últimos son los más completos. Por último, hay que añadir que los equipos que se encuentran solo muestran gráficas y no nos muestran información adicional como la ganancia del transistor.

En la documentación encontrada de estos equipos la cual no es abundante se puede observar que todos siguen aproximadamente el mismo esquema lógico. Como se puede observar en la figura 2.1, disponen de dos DACs, la salida de uno se conecta al colector del transistor bajo prueba, la del otro DAC a la base, el tercer terminal del transistor, el emisor en este caso, va conectado a la tierra del equipo (en caso de ser un transistor bipolar NPN, en el resto de casos la idea principal sería la misma). Entre los dos DACs y los terminales del transistor se encuentra una resistencia de valor fijo. Se fija una tensión en el DAC conectado a la puerta, y mediante la resistencia se obtiene una corriente de base continua. Una vez fijada la corriente de base estos equipos realizan un barrido de tensión en el colector, mediante el DAC que está conectado a la resistencia

que a su vez va conectado al colector. Por último, estos equipos realizan medidas mediante dos ADC y repiten este proceso cambiando la corriente de base tantas veces como gráficas deseen obtener. A partir de aquí los equipos muestran estas gráficas de distinta manera, normalmente en la pantalla de un osciloscopio.

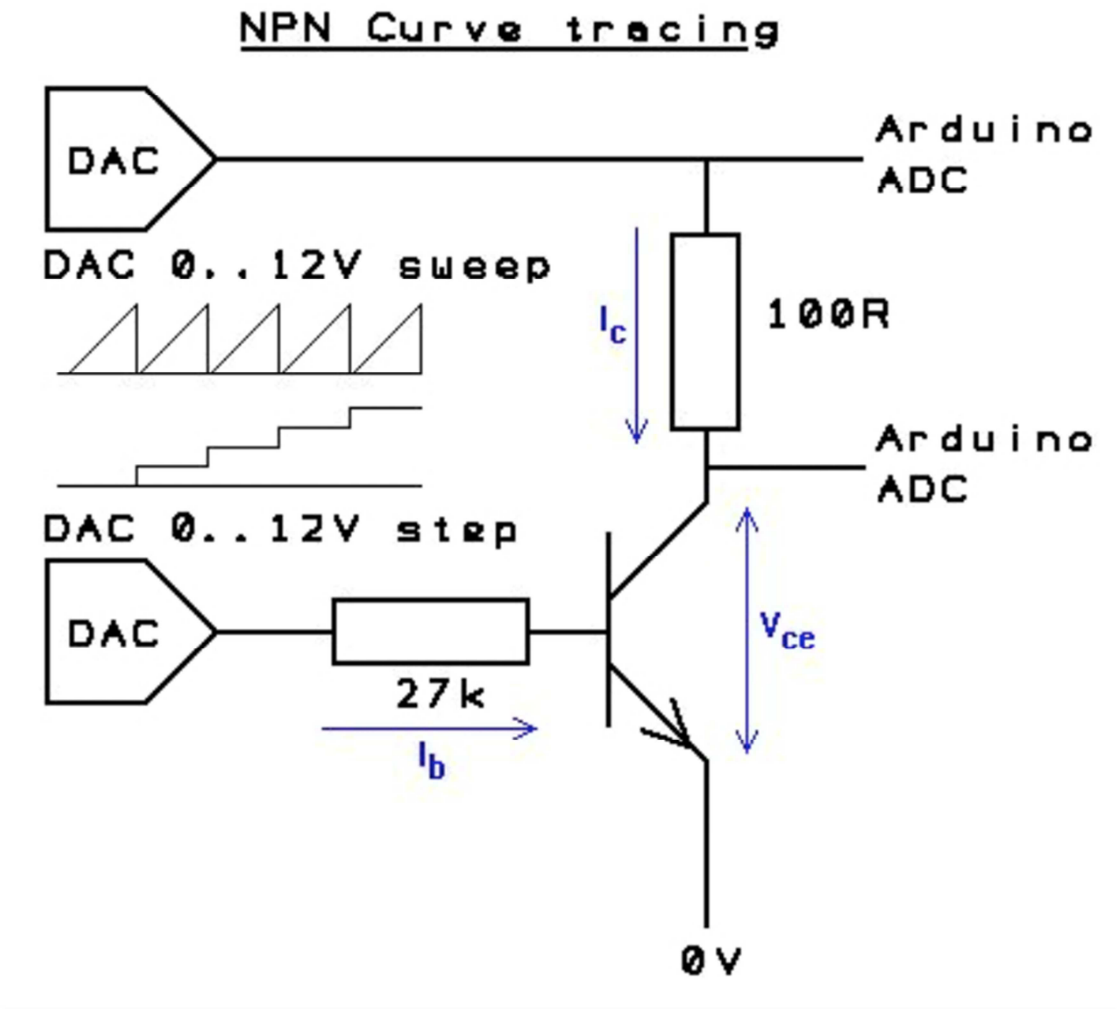


Figura 2.1: Esquema lógico de un equipo que obtiene las curvas de un transistor.

2.2 Estudio de especificaciones

Al igual que si se tratara de un proyecto que se desarrollase para un cliente se plantean una serie de especificaciones que se espera que el producto final cumpla. Algunas de estas especificaciones se han nombrado anteriormente pero ahora se van a ver de manera más detallada.

En primer lugar, el equipo que se desarrolla debe ser lo más 'inteligente' posible, de manera que sea capaz de distinguir entre diferentes tipos de transistores, es decir que sea

capaz de identificar si estos son bipolares, MOSFET, JFET o diodos. Una vez detectados debe saber identificar de qué tipo de unión se trata ya sea NPN o PNP. A esto se añade que los terminales del componente bajo test se puedan colocar como se desee en las 3 entradas del equipo, de esta manera el usuario no tendrá que conocer de ante mano cual es cada terminal. Será el propio producto el que detectará cual es cada uno de los 2 (en caso de diodos) o 3 terminales, y este indicará al usuario que terminal está conectado a cada entrada.

Una vez que el producto sea capaz de realizar todo lo anterior, este debe ser capaz de obtener la ganancia y las curvas características con la mayor resolución posible que correspondan a cada transistor.

Los datos y gráficas que se hayan obtenido se deberán mostrar mediante un servidor Web que genere el propio equipo, a este servidor Web se podrá acceder mediante la conectividad WiFi, por lo que se podrá acceder a los resultados mediante un ordenador o un Smartphone. Esto obliga al equipo a contar con una interfaz WiFi. Sería interesante añadir alguna funcionalidad de interacción con el equipo en el servidor Web de forma que por ejemplo permita empezar una medida. La interfaz gráfica del servidor web debe ser lo más sencilla e ilustrativa posible.

Se debe tener en cuenta el aspecto de la robustez procurando que esta sea lo mayor posible. Se entiende por robustez que el dispositivo sea capaz de detectar cuando no es capaz de identificar un componente porque se encuentra fuera de su abanico o cuando se hayan colocado mal los terminales del dispositivo a analizar en las entradas del equipo. En caso de que se detecte un error se procederá a parar la medida e informar al usuario mediante el servidor web. De esta manera se evita que realicen mediciones que muestren falsos resultados.

Para comenzar la medida el equipo debe de disponer de un interruptor físico que tras accionarlo realizará todo el proceso, a esto sería interesarle añadir un interruptor virtual que le permite al usuario comenzar la medida desde el servidor web.

Se deben incorporar uno o dos leds de manera que se establezca unos patrones luminosos que permitan al usuario identificar en qué estado se encuentra la máquina.

Se desea que este producto sea portable, es decir que a la hora de utilizarlo no sea necesario conectarlo a una fuente de alimentación o a la red eléctrica, para ello debe

contar con una batería que teniendo en cuenta el consumo eléctrico de los componentes le permita una autonomía razonable. Como era de esperar el equipo debe incorporar un sistema que le permita recargar esa batería. El producto debe tener en la medida de lo posible unas medidas lo más contenidas posibles. Por último, se debe incorpora un interruptor de encendido y apagado que permita conectar y desconectar la corriente eléctrica al equipo.

2.3 Estudio del sistema y selección de componentes

En esta sección una vez que se tienen claras las especificaciones se procede a desarrollar el hardware que sea capaz de cumplirlas. Lo que se muestra a continuación en la figura 2.2 es un boceto conceptual de los diferentes bloques con los que contará el equipo.

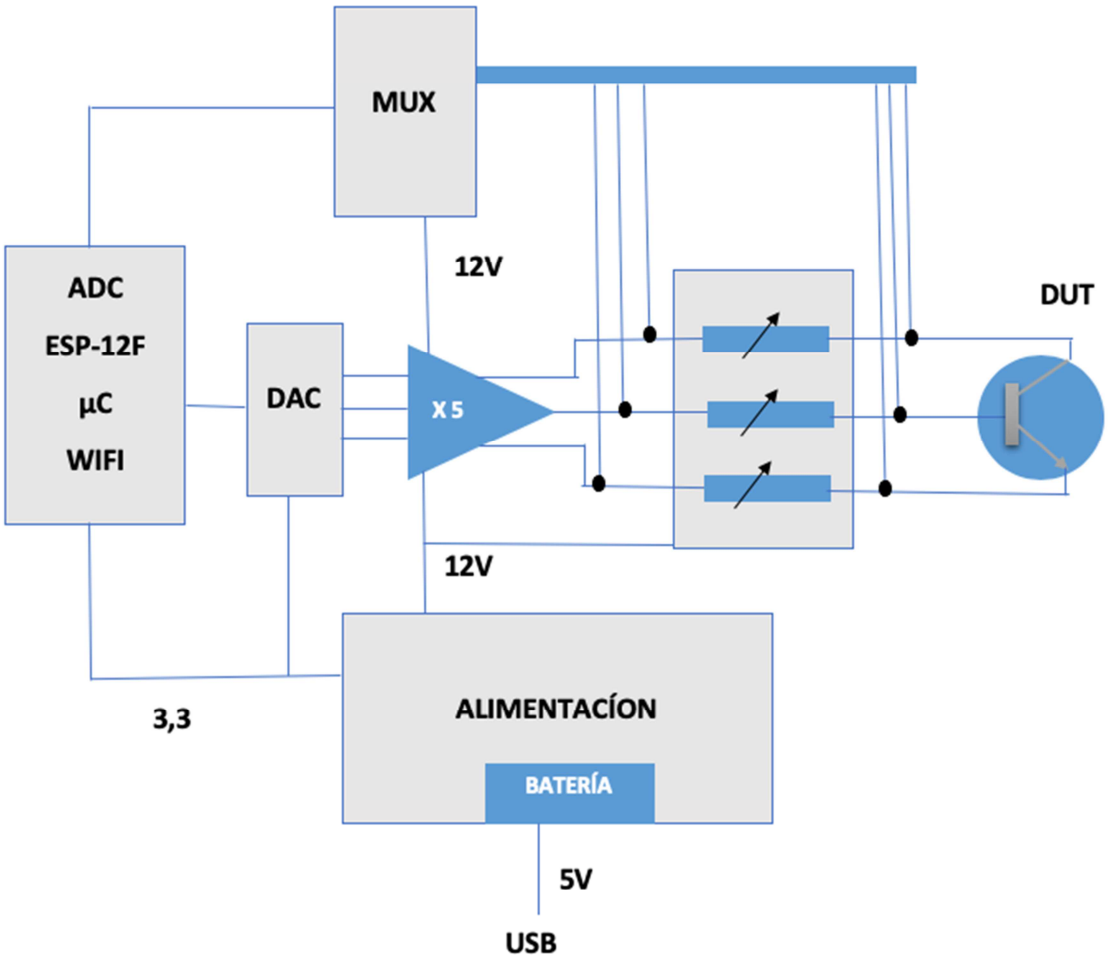


Figura 2.2: Boceto conceptual del producto.

2.3.1 Microprocesador

Como microprocesador que se encargue de orquestar todo el funcionamiento del producto se ha escogido el módulo ESP-12F [2]. La principal razón por la que se ha escogido este SOC es que incorpora un módulo WiFi y un microcontrolador fabricado por Espressif. Está diseñado por Ai-Thinker Technology, cuenta con 36kB de memoria RAM, también cuenta con una memoria *flash* SPI donde se pueden almacenar programas, puede funcionar a unas frecuencias de reloj de 80 MHz y 160MHz y soporta RTOS (*real time operating system*). Este procesador de 32 bits incorpora un módulo WiFi que soporta los estándares IEEE802.11 b/g/n e implementa una pila de protocolos TCP/IP completa para la comunicación.

Este módulo nos da la posibilidad de utilizar numerosas interfaces de comunicación ya conocidas, entre ellas I2C, SPI, UART y GPIO, además cuenta con un conversor analógico-digital (CAD) con una resolución de 10 bits. I2C será el protocolo seleccionado para interconectar los distintos componentes hardware del equipo. El microprocesador permite ser programado desde un entorno Arduino gracias al desarrollo de la comunidad, dentro de este entorno podemos encontrar librerías que facilitan la programación del protocolo I2C.

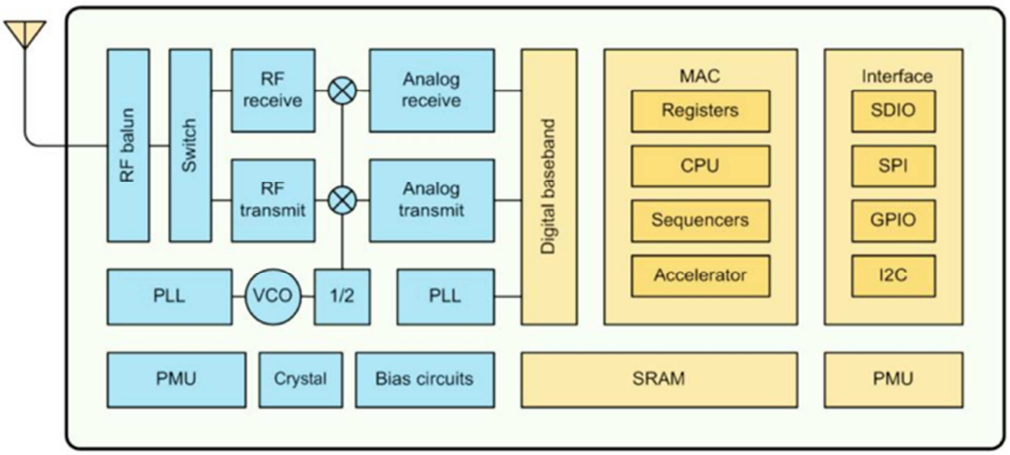


Figura 2.3: Bloques que forman el ESP-12F.

Para programar el chip se utilizará la comunicación serie de la UART, para lograr esta comunicación puesto que la salida del ordenador desde el que se programa es USB

se utilizará un conversor que transforme la comunicación USB en asíncrona. El chip encargado de realizar este proceso será el CY7C65213 del fabricante Cypress Semiconductor. La PCB en la que se encuentra este puente de interfaz no será la PCB del equipo final. Está pensado de esta manera para que un solo CY7C65213 pueda programar numerosos ESP-12F. Esto en caso de una producción de varias unidades permitiría abaratar costes y reducir el tamaño del producto.

Por último, hay que destacar que los principales atractivos de este módulo son que incorpora un módulo WiFi, permite utilizar un entorno Arduino, no tiene un consumo elevado, su rendimiento debería ser suficiente, y es bastante económico ya que se puede encontrar por un precio inferior a 3 euros.

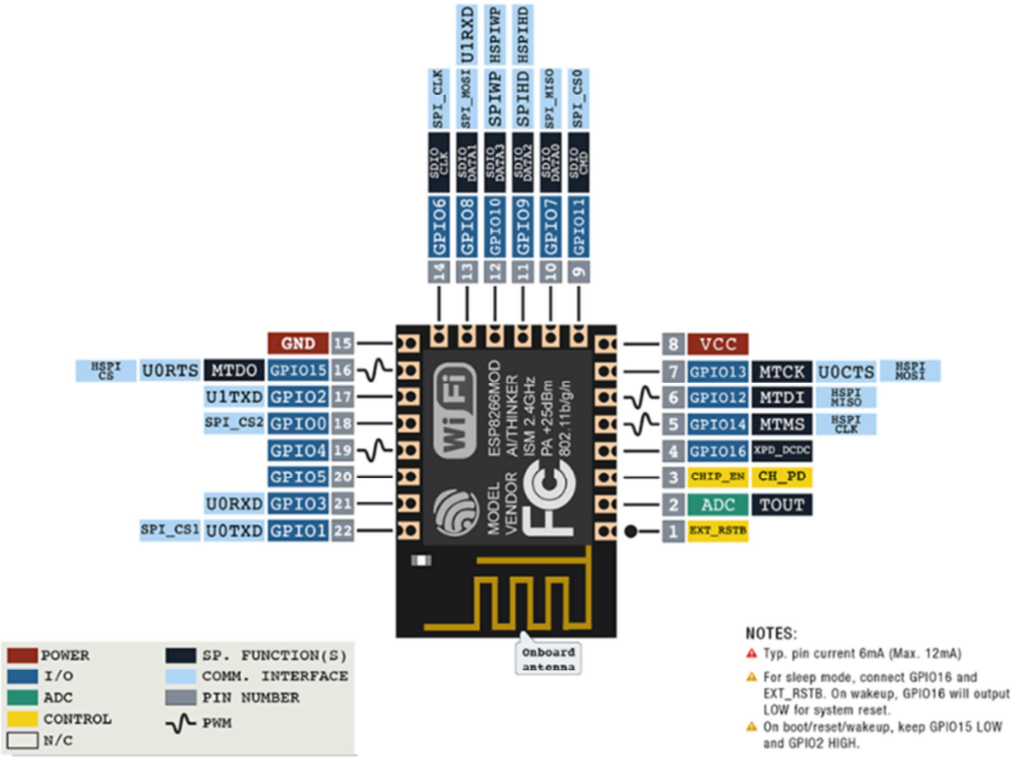


Figura 2.4: Módulo y pines ESP-12F.

2.3.2 DAC

Teniendo en cuenta que cualquier tipo de transistor puede estar colocado en cualquier posición y el equipo tiene que ser capaz de responder, se decide incorporar 3 DAC uno por cada terminal. En vez de incorporar 3 DAC de manera individual se escoge el chip MCP4728A0 del fabricante Microchip [3] el cual cuenta con 4 DACs.

Cada uno cuenta con 12 bits de precisión, puede emplearse como voltaje de referencia la tensión de alimentación o un voltaje de referencia interno de 2.048V, esta última configuración será la escogida además cuenta con la opción de doblar la tensión de salida mediante un amplificador.

Este chip además dispone de interfaz serie I2C, como ya se indicó anteriormente va a ser el protocolo I2C el utilizado para comunicar el componente con el microprocesador ESP-12F. A pesar de que no se va a emplear cuenta con una memoria interna no volátil EEPROM que permite almacenar los valores de salida de los DAC y recuperarlos cuando se vuelve a iniciar el componente. En la figura 2.5 se muestra un esquema del funcionamiento del DAC.

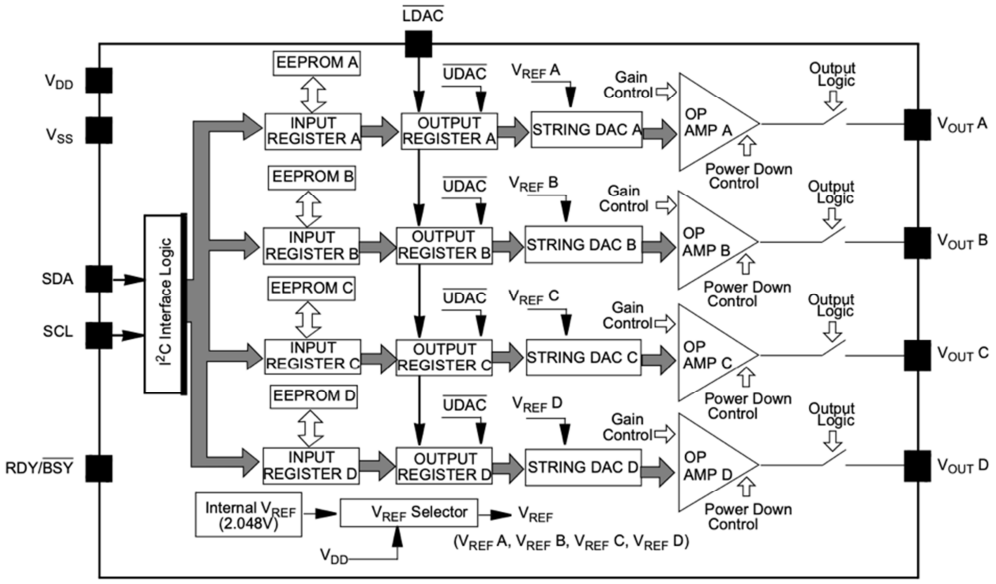


Figura 2.5: Esquema lógico del MCP4728A0.

2.3.3 POTENCIOMETRO

Para además de controlar las tensiones y tener un mayor rango de libertad en lo que respecta a las corrientes que circulan por los terminales en vez de emplear una resistencia entre el DAC y su terminal correspondiente se empleará un potenciómetro digital. Al igual que ocurre con el DAC en vez de tener 3 chips para 3 potenciómetros

se opta por seleccionar un chip que incorpora 4 potenciómetros digitales, el AD5263BRU200 del fabricante Analog Devices [4].

Este componente, que utilizamos como resistencia variable, cuenta con 8 bits de resolución o 256 posiciones que van desde 120Ω a 200kΩ. Permite operar hasta con tensiones de 15v a diferencia de la mayoría de los potenciómetros digitales del mercado que no permiten tensiones tan altas, este aspecto es fundamental ya que tendrá una tensión de alimentación de 12v. La justificación de las alimentaciones escogidas se desarrollará más adelante. Por último, estas 256 posiciones de los cuatro potenciómetros son programables mediante SPI o I2C que es la interfaz que se utilizará. En la siguiente figura 2.6 se muestra el esquema lógico de este componente.

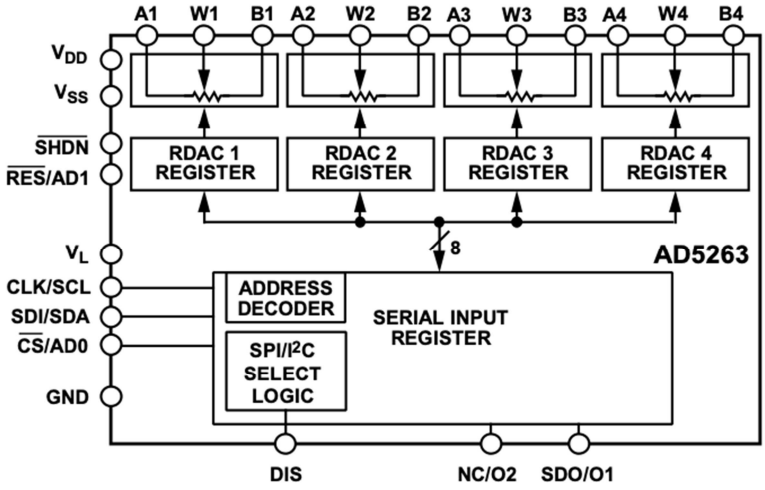


Figura 2.6: Esquema lógico del AD5263BRU200.

2.3.4 AMPLIFICADORES

De los DAC obtenemos una salida máxima en tensión de 2,048V, se desea obtener unos rangos de medida mayores, mayor tensión máxima y por tanto mayor intensidad máxima que pueda ser aplicada al DUT (*device under test*). Se decide ampliar la tensión máxima en un factor 5 hasta alcanzar aproximadamente los 10V.

Se selecciona el amplificador operacional MAX44243AUD+ [5] del fabricante Maxim Integrated, figura 2.7. Cuenta con 4 amplificadores de alta precisión y salida rail to rail integrados en el mismo componente. De estos amplificadores 3 irán destinados a amplificar la salida de los DAC en un factor 5 mediante una realimentación positiva.

Está alimentado a una tensión de 12V con lo que se cubre las tensiones de salida máxima que se pueden llegar a obtener que son 10V. Además, cuenta con 5Mhz de producto ganancia por ancho de banda (GBWP) para tener unos transitorios lo más rápidos posibles. Para esta realimentación se usarán resistencias de tolerancia inferior al 0,1% ya que al tratarse de un equipo dedicado a realizar mediciones es vital contar con la mayor precisión posible por lo que se trata de evitar un error en el factor de amplificación. El amplificador restante se usará como seguidor de tensión (buffer) antes de la entrada del ADC que incorpora el ESP-12F.

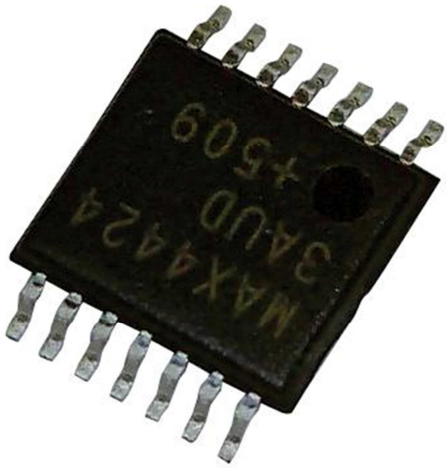


Figura 2.7: Circuito integrado MAX44243AUD+.

2.3.5 MULTIPLEXOR

Solo se dispone de un convertidor analógico digital, el de 10 bits de resolución que integra el ESP-12F, pero es necesario realizar medidas de 6 señales distintas, estas se encuentran en cada extremo de los 3 potenciómetros digitales que se van a utilizar. Estas 6 señales o puntos nos van a permitir conocer en todo momento no solo la tensión sino también la corriente que circula por cada uno de los tres terminales del componente a caracterizar.

Para seleccionar que canal se quiere medir en el ADC se opta por emplear el multiplexor CD4051BM96 del fabricante Texas Instruments [6]. Este es un multiplexor analógico que cuenta con 8 canales de entrada y uno de salida, al ser analógico también se puede emplear como un demultiplexador, es decir los canales pueden funcionar como

entrada o salida indistintamente. La selección de canal está controlada por 3 entradas digitales. El voltaje de estas entradas para transmitir un estado lógico positivo dependerá del valor que tome la alimentación del componente. El máximo voltaje de las señales que van a atravesar el multiplexor es aproximadamente 10V, al tratarse de un multiplexor analógico la tensión de alimentación debe ser superior a la máxima de entrada de alguno de sus canales, se selecciona una tensión de alimentación de 12V.

Como ya se ha indicado el voltaje para que la señal de selección se considere un estado lógico positivo depende de la alimentación. Esta alimentación que será aproximadamente 12V es muy superior a la tensión máxima de salida que pueden suministrar los pines digitales GPIO del microprocesador ESP-12F que es 3,3V. Para solucionar este problema de las tensiones de los 3 canales de selección se añaden al circuito 3 elevadores de nivel que permiten con la tensión de salida de los pines del microprocesador en alta conseguir un nivel de tensión en los pines de selección del multiplexor prácticamente igual a la alimentación.

2.3.6 MEMORIA

Se desea contar con una memoria que permita almacenar datos en caso de que sea necesario, como valores de las medidas que se tomen, gráficas, calibraciones de las resistencias de los potenciómetros digitales y otros valores que se deseen almacenar.

El ESP-12F cuenta con una memoria interna *flash* de 4MB conecta por medio de SPI. Puesto que la documentación sobre como acceder a esta memoria y escribir en ella es difícil de encontrar y la existente no es del todo a aclaratoria se decide añadir una memoria EEPROM al producto. La memoria seleccionada es la M24C64 [7], cuenta con 64-Kbit de almacenamiento y una interfaz I2C.

2.3.7 ALIMENTACIÓN

Como núcleo de la alimentación se encuentra una batería de litio de 800 mAh, ya que como se ha detallado anteriormente se desea que el equipo cuente con autonomía eléctrica, la capacidad de la batería será discutida más adelante.

La batería va a contar con una tensión nominal de 3,7V, pero como se puede ver en las hojas de datos de los componentes y se ha nombrado anteriormente las tensiones de alimentación de los componentes del producto van a ser 3,3V y 12V. El

microprocesador ESP-12F, el DAC cuádruple y la memoria EEPROM se alimentarán a 3,3V mientras que el potenciómetro cuádruple ADC5263BRU200, el amplificador operacional cuádruple MAX44243AUD+ y el multiplexor CD4051BM96 irán alimentados a 12V.

Se selecciona el MCP1725-3302ESN para pasar de la alimentación de 3,7V a 3,3V. El MPC1725 es un regulador lineal en el que la tensión de salida puede ser ajustable mediante una serie de resistencias o fija como es en el caso del modelo MCP1725-3302ESN.

Para obtener la alimentación de 12V a partir de una de tensión nominal de la batería se emplea el FAN5331SX, un regulador conmutado de tipo *Boost* de salida variable en función de su referencia interna y las resistencias que se le conecten.

Para cargar la batería se añade el cargador de batería MCP73831T-2ACI/OT que es un controlador de gestión de carga lineal. Está idealmente preparado para aplicaciones que incorpora un puerto USB para la carga. Por está razón se ha incorporado un terminal USB micro de tipo B hembra. Este tipo de terminal a pesar de no ser el nuevo estándar USB de carga para los Smartphones, como lo es el USB tipo C, es el más extendido actualmente y es relativamente sencillo encontrar un cargador con este tipo de adaptador

2.3.8 OTROS

Además de estos componentes principales se han empleado otros componentes como son resistencias condensadores y bobinas los cuales se han empleado para diferentes tareas como divisores de tensión, realimentaciones, desacoplos, etc. Hay que destacar que alguna de las resistencias empleadas han sido seleccionas con una tolerancia menor al 0,1% para las partes más críticas relacionadas con la señal a medir.

Por último, se han introducido al diseño un T-Block de 3 entradas al que por medio de unos cocodrilos o una regleta de inserción se conectan los 3 terminales del transistor a medir, un pulsador para comenzar la medida, un interruptor para conectar o desconectar la alimentación y unos diodos leds.

3 Desarrollo Hardware

3.1 Diseño esquemático

Para realizador todo el diseño relacionado con la parte hardware del proyecto se ha empleado la herramienta software Proteus en su versión 8.9. En el caso de este proyecto se van a emplear algunas de sus principales utilidades como son la realización del esquemático, la creación de nuevos componentes que no están disponibles dentro de las librerías de Proteus, el diseño de la PCB y la creación de un listado de materiales, el *Bill of materials* (B.O.M.)

El esquemático del circuito está formado por todos los componentes que permiten el correcto funcionamiento del producto. En la versión de Proteus que se ha empleado los componentes que se pueden consideran poco habituales o no estándar no se encuentran dentro de sus librerías. Esto supone que ha sido necesario el diseño esquemático de dichos componentes, así como de su huella para la PCB. Esto requiere que ambos diseños cuenten con el mismo número de pines que el componente original y el caso de la huella para la PCB que su diseño siga fielmente las dimensiones. A continuación, se muestra el esquemático realizado y se procederá a un análisis de cada sección.

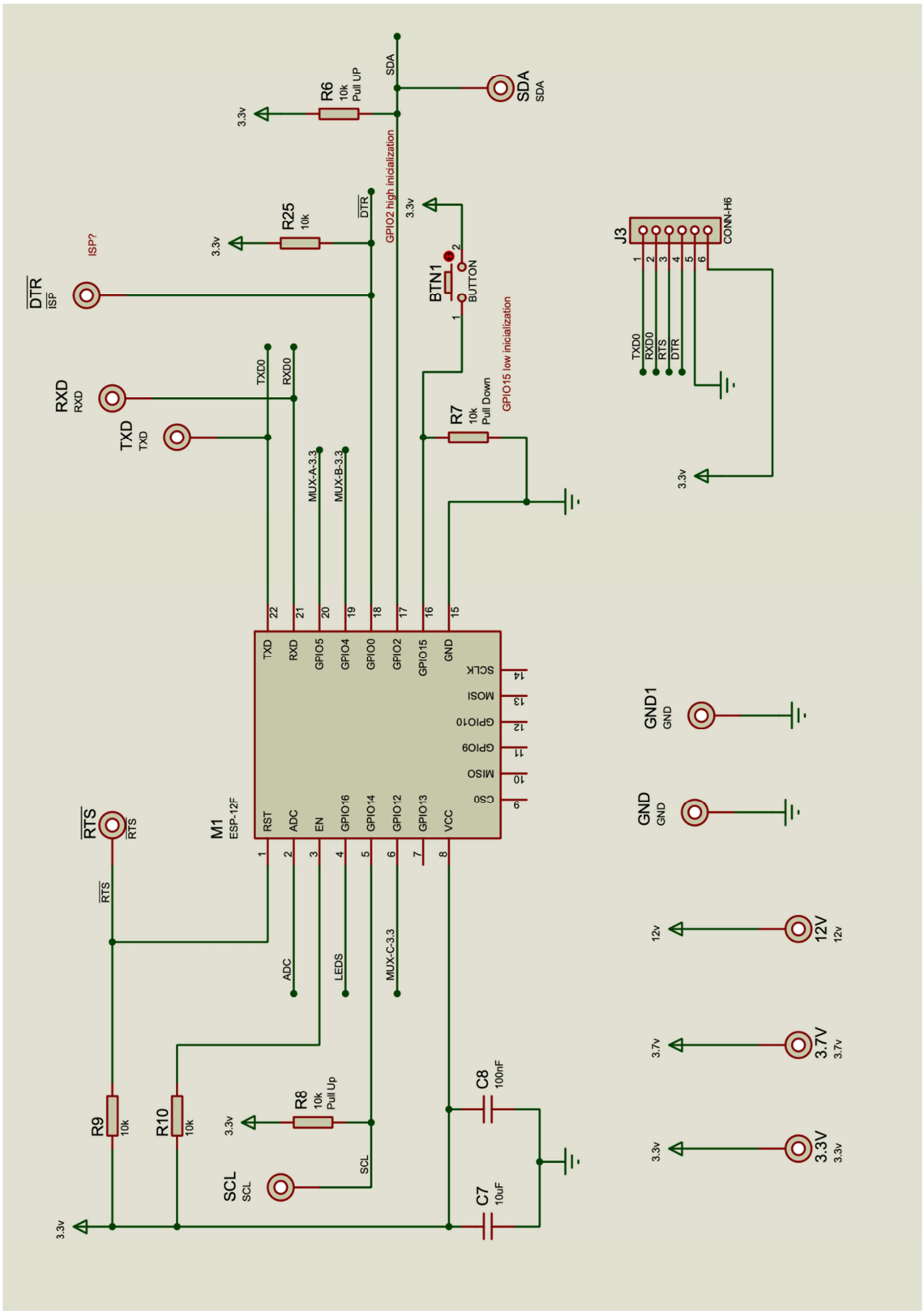


Figura 3.1: Hoja 1 del esquemático, módulo ESP-12F.

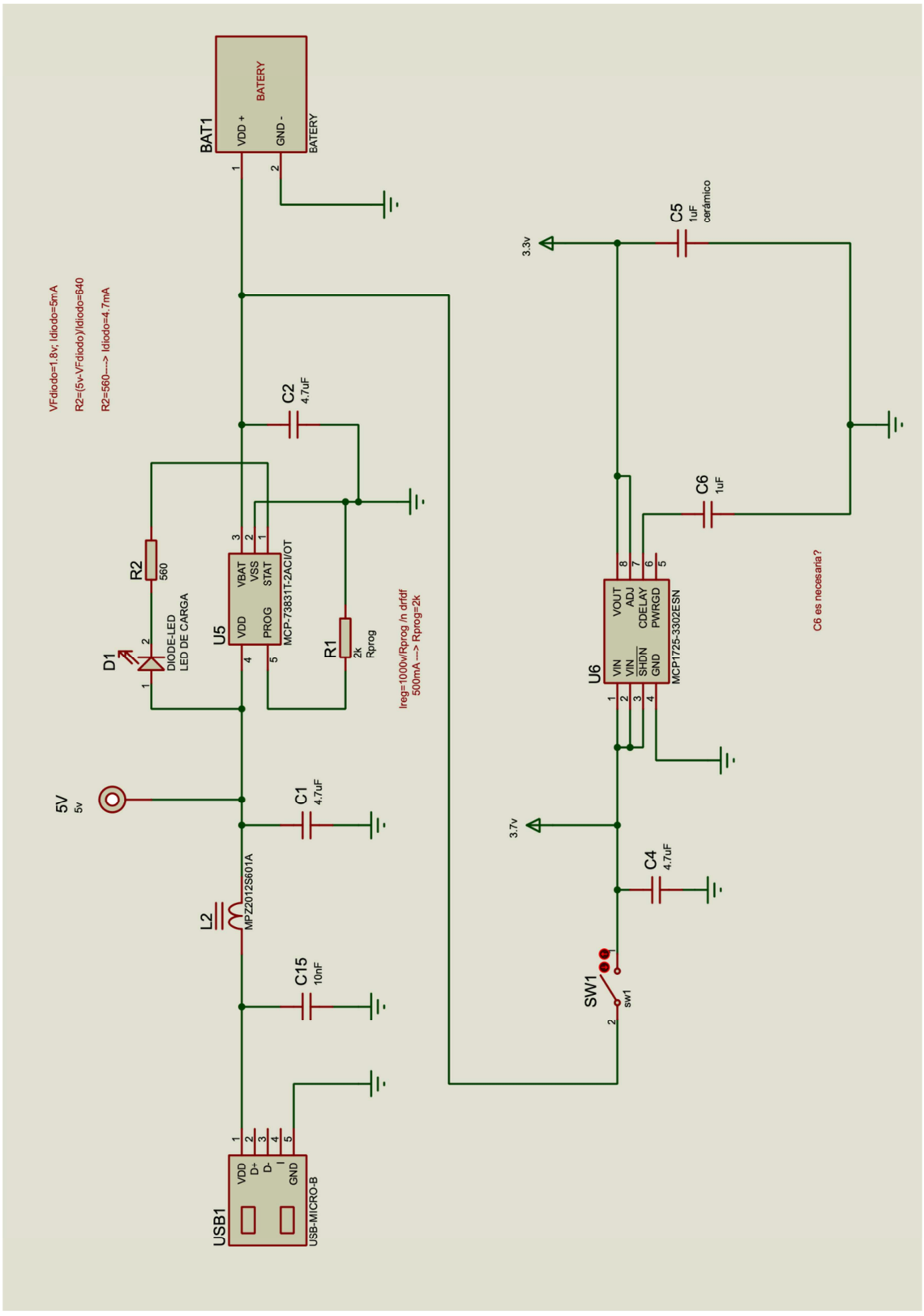


Figura 3.2: Hoja 2 del esquemático, batería y alimentación 3.3V.

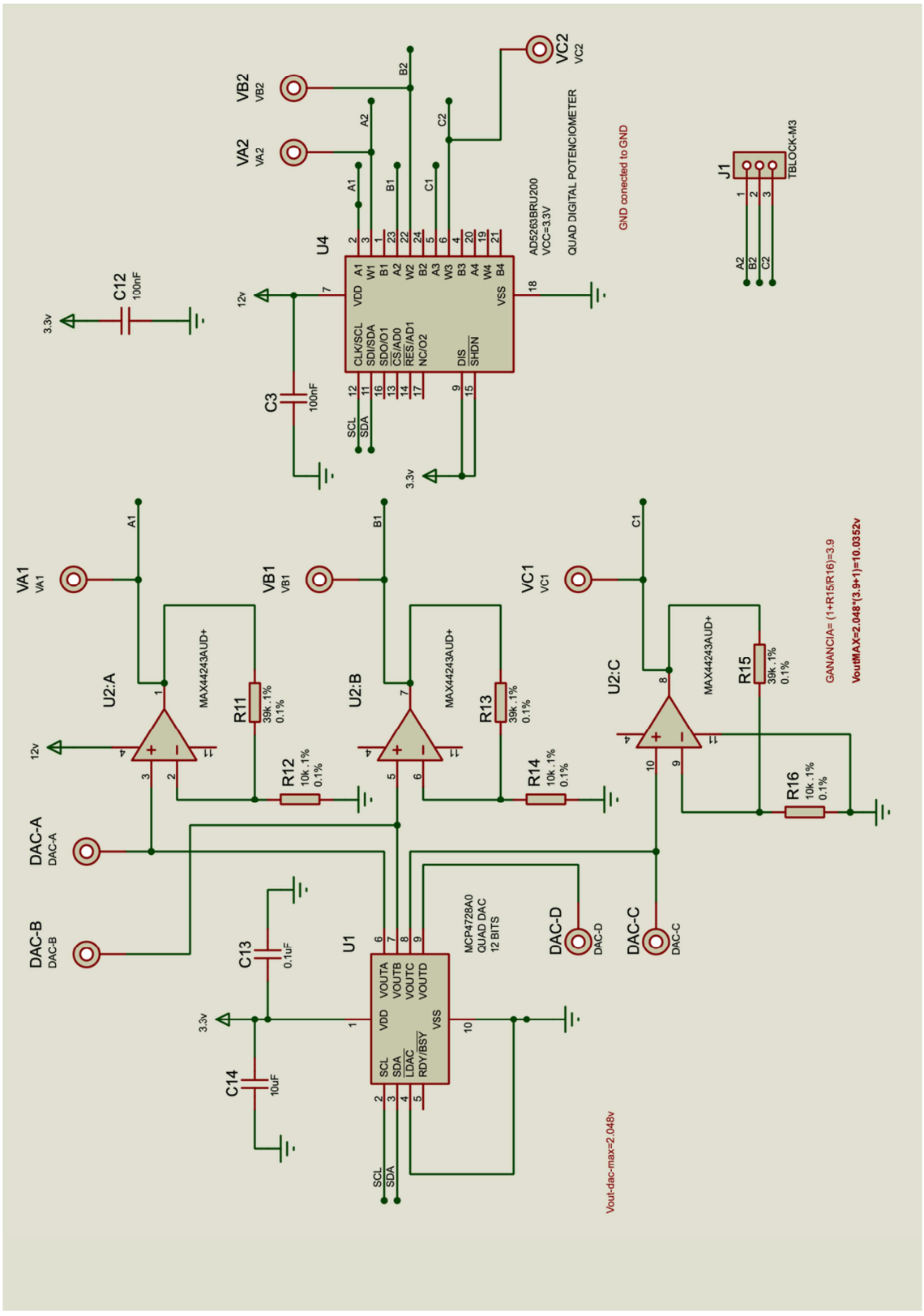


Figura 3.3: Hoja 3 del esquemático, electrónica analógica: DAC, amplificadores y potenciómetro digital.

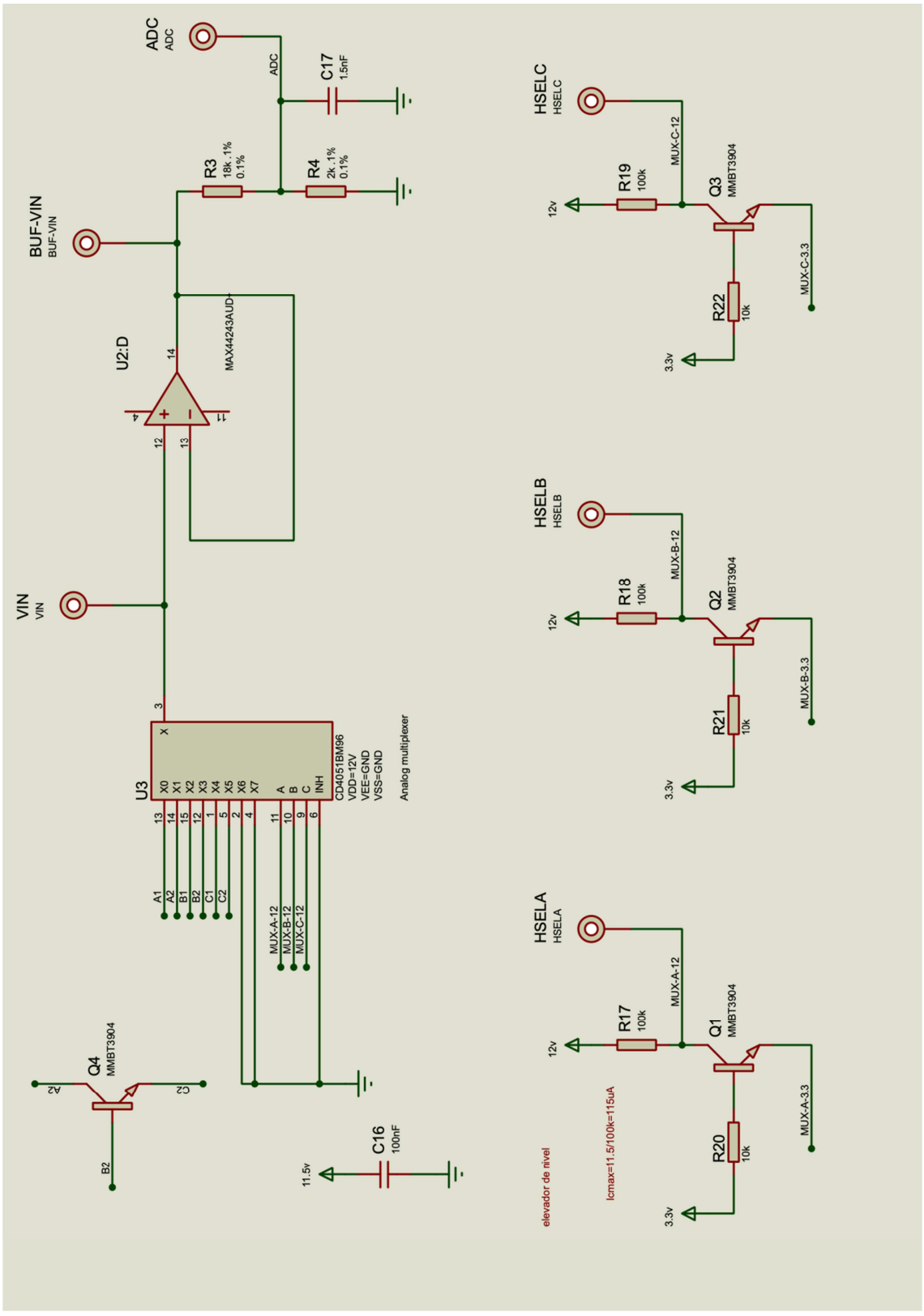


Figura 3.4: Hoja 4 del esquemático, electrónica analógica: Multiplexor de entrada y buffer.

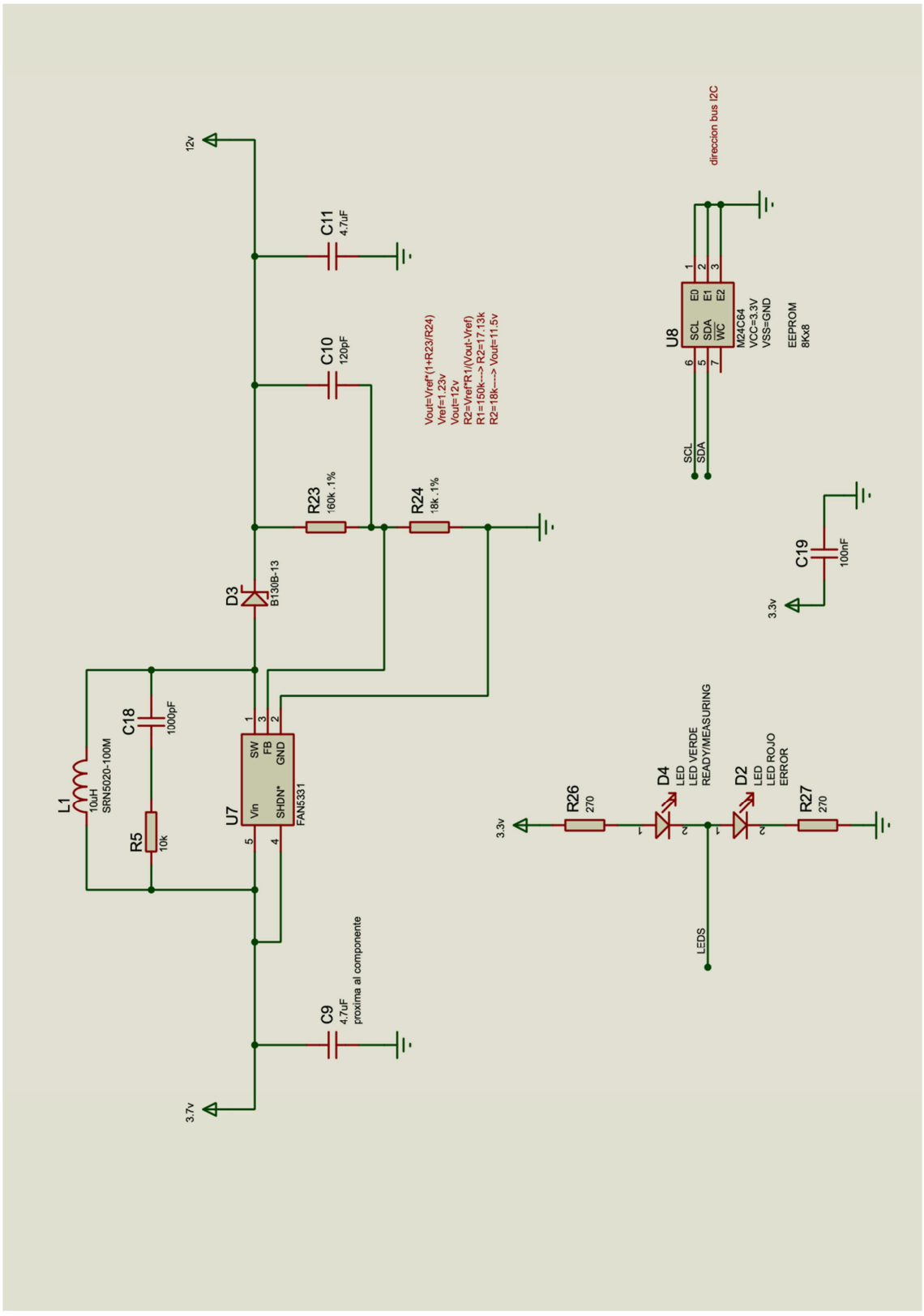


Figura 3.5: Hoja 5 del esquemático, alimentación de 12V y miscelánea.

3.1.1 Circuito módulo ESP-12F

Como se puede ver en la figura 3.1 al módulo se le proporciona la alimentación necesaria de 3,3V y se le añaden dos condensadores de desacoplo. Ciertos pines del micro han de tener un valor concreto, como se ve en la figura 3.6, para la inicialización del microcontrolador, estos son GPIO0, GPIO2 y GPIO15

PIN MODE FOR			
	GPIO15	GPIO0	GPIO2
UART BOOT	0	0	1
FLASH BOOT	0	1	1

Figura 3.6: pines Boot ESP-12F.

Los terminales de RESET, DTR conectados a un Pull-Up se conectarán a tierra por medio del circuito externo USB-serie que gestiona el CY7C65213, al igual que los pines de TXD y RXD.

Para su normal funcionamiento se reservan los pines de propósito general GPIO2 y GPIO14 para las líneas del protocolo I2C de SDA y SCL respectivamente. Otros tres pines, el GPIO4, GPIO5 Y GPIO12 se emplearán para seleccionar uno de los canales del multiplexor. El GPIO15 irá asociado a un interruptor de manera que se pueda detectar su pulsación y el GPIO16 en función de sus tres posibles estados nos permitirá controlar dos leds. Por último, se tiene la entrada del ADC que irá conectada a la salida del multiplexor y permitirá realizar las medidas de tensión que se deseen.

3.1.2 Alimentación

En las figuras 3.2 y 3.5 nos encontramos el esquema de la alimentación. En el podemos observar una ferrita que filtra la alimentación que proviene del USB-micro-b, también encontramos el cargador de la batería U5 que cuenta con un led para indicar que se encuentra cargando, el regulador lineal U6 y el elevador Boost U7 que nos permiten pasar de la tensión nominal de la batería a unos 12V. La tensión de la batería

en función del estado de carga puede ser variable, pero con estos chips obtenemos unas tensiones de 3,3V y 12V estables

3.1.3 DAC y amplificador

En la izquierda de la figura 3.3 podemos observar el DAC U1 conectado a la alimentación de 3,3V y a las líneas del bus I2C que seleccionan la tensión de sus 4 salidas. Solo 3 salidas de DAC serán empleadas, estas irán conectadas a 3 de las 4 entradas del amplificador operacional U2, el cual está realimentado positivamente ya que es más cómodo trabajar con tensiones positivas. Esta realimentación se realiza por medio de 2 resistencias de alta precisión para asegurarse de una buena precisión en la tensión de estos 3 canales. El cuarto amplificador se utiliza como BUFFER a la salida del multiplexor.

3.1.4 POTENCIOMETRO

En la parte derecha de la figura 3.3 podemos ver el potenciómetro U4, este recibe las señales del amplificador como entradas por los pines A, y tras pasar por la resistencia que se ha seleccionado gracias al bus I2C la señal sale por los terminales W. La señal de salida de estos tres terminales es la que irá directamente conectada al DUT. En este caso se ha decidido añadir un TBLOCK conectado a esos terminales que facilite la conexión de los transistores o diodos.

Para que el producto realice correctamente su función es necesario saber la tensión de los terminales del dispositivo que se va a caracterizar para ello las 3 salidas de los terminales W se conectan al multiplexor U3. Además, es necesario medir las corrientes que circulan por esos terminales, para ello las 3 entradas en los terminales del potenciómetro se conectan también al multiplexor. Sabiendo la diferencia de potencial que hay en una de las resistencias del potenciómetro y conociendo el valor de esta resistencia podemos calcular la corriente que circula por el terminal.

3.1.5 Multiplexor

En la figura 3.4 se muestra el multiplexor que recibe como entrada las señales nombradas anteriormente, la selección de unas de esas 6 entradas se realiza por medio de 3 entradas de selección. Las tensiones digitales que habilitan o deshabilitan estas entradas de selección depende de la tensión de alimentación del circuito. Al estar este alimentado a 12V será necesario una tensión cercana a esa tensión de alimentación para considerar una entrada en alta. Los tres pines de selección del microprocesador que se nombraron anteriormente solo pueden dar una salida hasta 3,3V. Para solucionar esto se añaden 3 elevadores de tensión formados cada uno principalmente por un transistor Q1, Q2 o Q3, uno por cada entrada de selección. De esta manera si una de las señales del microprocesador se pone en alta obtenemos en la entrada de selección una tensión cercana a 12V y si la señal del micro se pone en baja, en la entrada del multiplexor tendremos una tensión cercana a 0V.

A la salida del multiplexor tenemos un amplificador del U2 funcionando como Buffer ya comentado, a continuación, tenemos un divisor de tensión que nos permite disminuir la tensión en un factor 10. Esto se debe a que la tensión máxima de la señal a medir puede llegar a ser de hasta 10V pero el ADC del micro tiene un fondo de escala de 1V. De esta manera sea cual sea la tensión que se va a medir estará comprendida entre 0 y 1V, luego se reescalará mediante software. El divisor de tensión está formado por las resistencias R3 y R4 que son de baja tolerancia de manera que la precisión de la medida sea distorsionada lo menos posible y las tensiones que se están midiendo sean lo más fieles posibles a la realidad.

3.1.6 Memoria y leds

Finalmente, en la figura 3.5 podemos ver la memoria EEPROM direccionada mediante el bus I2C y alimentada a 3,3V. Tenemos 2 diodos leds con dos resistencias para limitar la corriente que circula por los diodos. Entre los dos diodos encontramos la señal del pin GPIO16. Mediante el estado de este pin del micro se puede encender uno de los dos leds o dejar estos prácticamente apagados. En caso de que la señal LEDS del micro este en alta solo D2 se encenderá, si la señal está en baja será en este caso D1 el que se encenderá mientras D2 permanece apagado, y por último si el GPIO16 se encuentra en alta impedancia ambos leds estarán prácticamente apagados.

3.2 DISEÑO DE LA PCB

Una vez completado el proceso de diseño y revisión del esquemático, se procede a comenzar con el diseño del circuito impreso, para esta tarea como ya se ha indicado se continuará utilizando la herramienta de software Proteus.

Antes de comenzar con el posicionamiento de los componentes sobre la PCB hay que asegurarse de que se cuenta con todas las huellas. Huella hace referencia a los *pads* y la serigrafía que sirve como referencia a la hora de posicionar los componentes sobre la PCB fabricada. Es vital que estas huellas sigan fielmente las dimensiones del componente para que los pines del componente encajen perfectamente con los *pads* y de esta manera con el trazado del circuito. De esta forma se consigue conectar el chip con el resto de los componentes del circuito. En caso de que la huella no coincidiera con el numero de pines, los *pads* de la huella no estuvieran asignados correctamente a los pines o las dimensiones no fueran correctas podría llegar a ser necesario un rediseño de la PCB. Por último, es conveniente añadir a las huellas que se creen una serigrafía que ayude a su posicionamiento, como por ejemplo indicando una esquina de referencia para los componentes simétricos.

Para la generación de las huellas se ha empleado también Proteus basándose en los datos mecánicos que se encuentran dentro de las hojas de datos de los componentes. En el caso del interruptor y el botón se ha empleado un calibre para obtener las dimensiones y diseñar la huella ya que no se disponía de una hoja de especificaciones. Una vez diseñada la huella de los componentes que no las tenían, se ha asignado correctamente el patillaje a los *pads* de la huella.

Las huellas que ha habido que generar son las del módulo WiFi ESP-12, el micro-USB tipo B, el interruptor de la batería SW1, y un botón BTN1, estas se pueden observar en dicho orden en la figura 3.7.

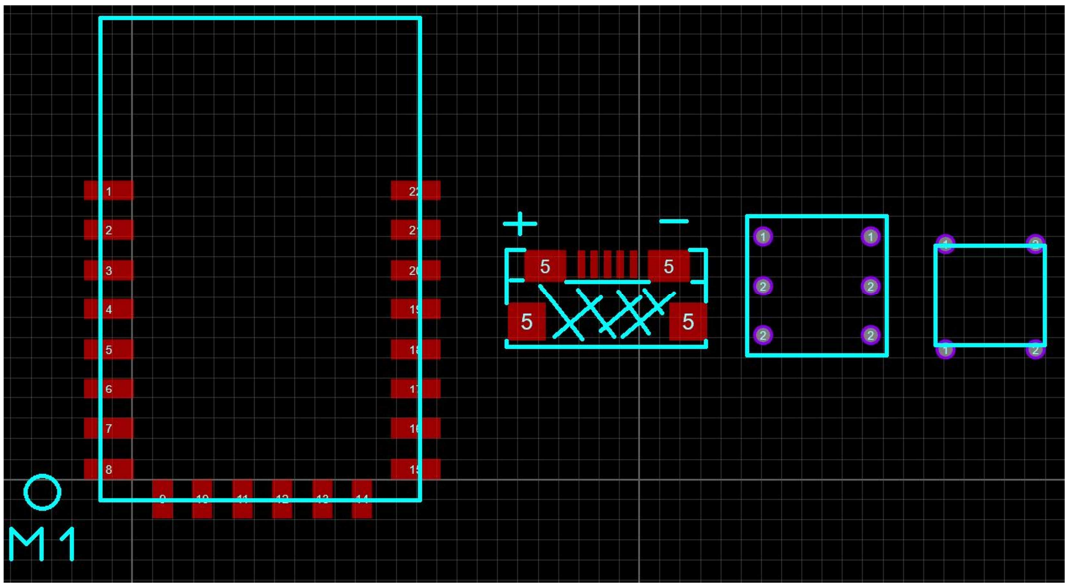


Figura 3.7: Huellas generadas.

Una vez se han creado las huellas que faltaban se ha procedido a la colocación de los componentes en el *layout* del circuito impreso. Originalmente se parte de una PCB de 2 capas con unas dimensiones de 10cm x 10cm, dimensiones estándar para el fabricante asiático de tarjetas, a partir de estas dimensiones y número de capas el precio comienza a aumentar. A la hora de colocar los componentes se va a intentar cumplir el mayor número posible de reglas de diseño sobre compatibilidad electromagnética [8]. Algunas de estas normas son:

- Las pistas deben de ser lo más cortas y estar lo más separadas posibles de manera que se eviten acoplamientos capacitivos, o acoplamientos por impedancia común.
- Las pistas no deben formar un área grande para evitar que los bucles de corriente generen acoplamiento magnético. Tanto esta norma como la anterior favorecen a disminuir las emisiones de EMI. Las pistas actúan como antenas, si se consigue que estas radien menos también se conseguirá que las emisiones recibidas del exterior o de otras partes del circuito afecten menos a esta antena o pista y por tanto a todo el circuito.
- Los condensadores de desacoplo deben estar lo más cerca posible del componente a proteger, en especial los condensadores más pequeños en caso de que se posicionen más de uno.
- Procurar en la medida de lo posible separar la electrónica analógica de la digital.

- En caso de que sea posible incluir uno o varios planos de masa de manera que se consiga un camino de retorno para la corriente lo más corto posible y con una impedancia común muy baja. Hay que destacar que una señal puede retornar por varios caminos en función de su frecuencia.

Cuando se comienza a colocar los componentes el primero en posicionarse es el módulo ESP-12F. Siguiendo una documentación de recomendaciones sobre su posicionamiento, este se coloca en la esquina superior izquierda, con parte del módulo sobresaliendo de la PCB, esta parte que sobresale es la antena. De esta manera lo que se consigue es que es que la antena WiFi que emite a 2,4Ghz este lo más separada posible del resto de componentes y el plano de masa. Con esto se consigue reducir en cierta medida el acoplamiento que puede haber entre ellos. Además, al estar la antena lo más separada posible del plano de masas, su emisión radiada se verá menos apantallada por el plano de masa y la PCB en si. De esta manera será posible comunicarse con el módulo a una distancia algo mayor.

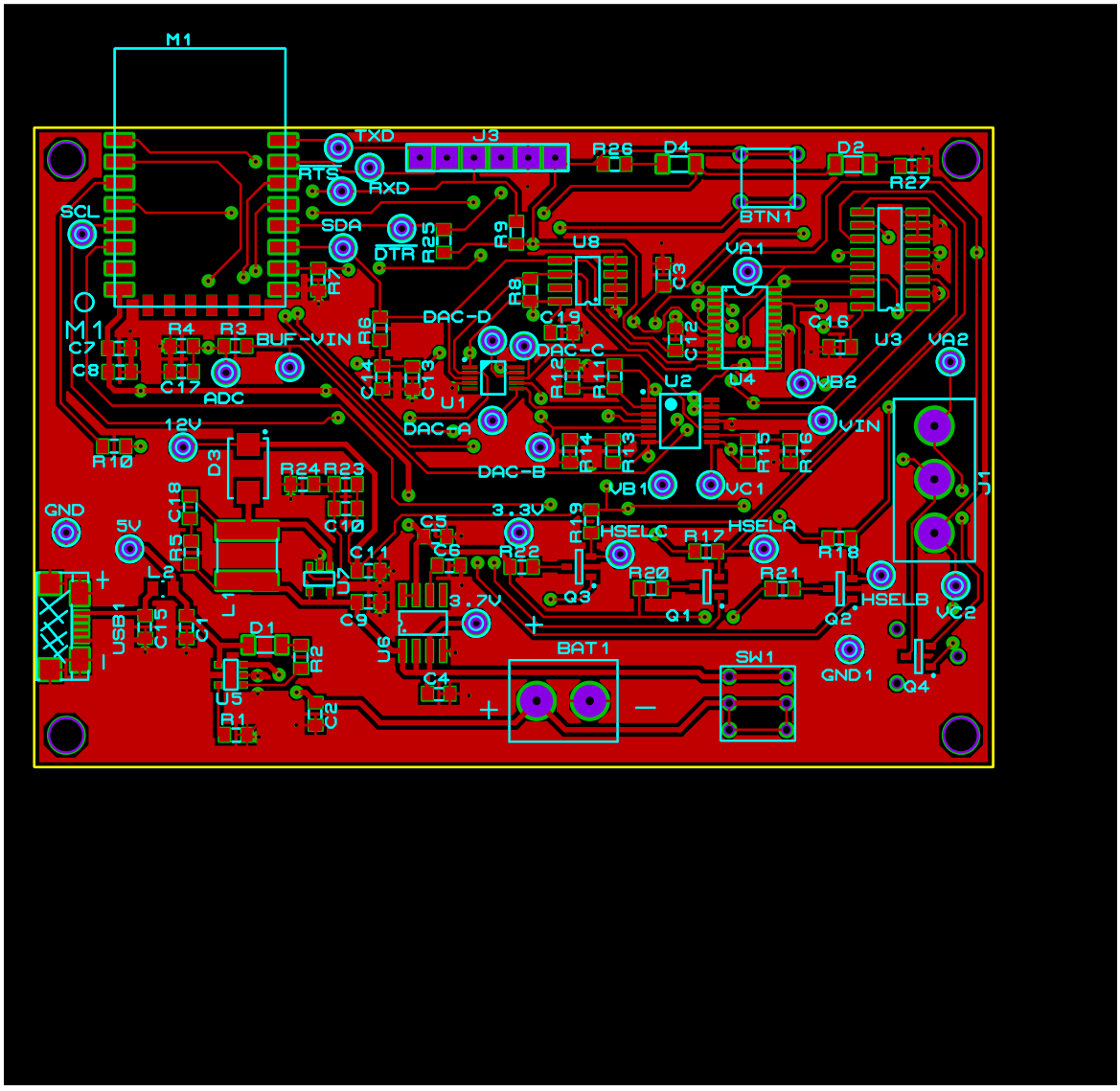
Todos los componentes relacionados con la alimentación se conectan juntos al ser estos completamente analógicos, se posicionan en la parte inferior izquierda junto al conector micro-USB-B. El resto de los componentes se colocan por la PCB restante intentando que se ocupe el menor espacio posible y que su posicionamiento ayude al rutado posterior de manera que se cumplan el mayor número posible de normas de compatibilidad electromagnética. Se consiguió reducir el tamaño del circuito impreso de 10cm x 10cm a 9cm x 6cm.

Una vez completado el posicionamiento se procede al rutado, en esta fase se sigue la estrategia de tratar de rutar el mayor número posible de pistas por la cara superior. Esta estrategia tiene como objetivo que en la cara inferior tengamos un plano de masas con el menor número de cortes posibles. Con esto conseguimos disminuir el camino de retorno de las señales y que haya puntos del plano de masa donde puedan producirse picos de corriente y por tanto se produzca una diferencia de potencial que afecte a la referencia de la tierra.

Para el proceso de rutado no se ha empleado la opción de autorutado que tiene Proteus ya que el rutado que se obtiene dista mucho de ser óptimo y en muchos casos es incapaz de realizarlo correctamente. Con el rutado manual pese a que lleve más tiempo se consigue un rutado mejor desde el punto de vista de la compatibilidad

electromagnética y permite al diseñador seguir la estrategia nombrada anteriormente. Una vez acaba el rutado se añadió otro plano de masas en la capa superior.

Hay que destacar que en el diseño de la PCB se han incluido serigrafías que ayudan a posicionar los componentes, como la etiqueta de ese componente y una referencia para ayudar a orientarlo cuando se vaya a montar en la PCB. Además, se han incluido puntos de *test* junto con la serigrafía de que señal permite medir ese punto de test. Esto será muy útil a la hora de verificar el hardware y depurar el software ya que se han incluido puntos de prueba de todas las señales del circuito. A continuación, en la figura 3.8 y en la figura 3.9 se muestran la cara superior e inferior de la PCB diseñada.



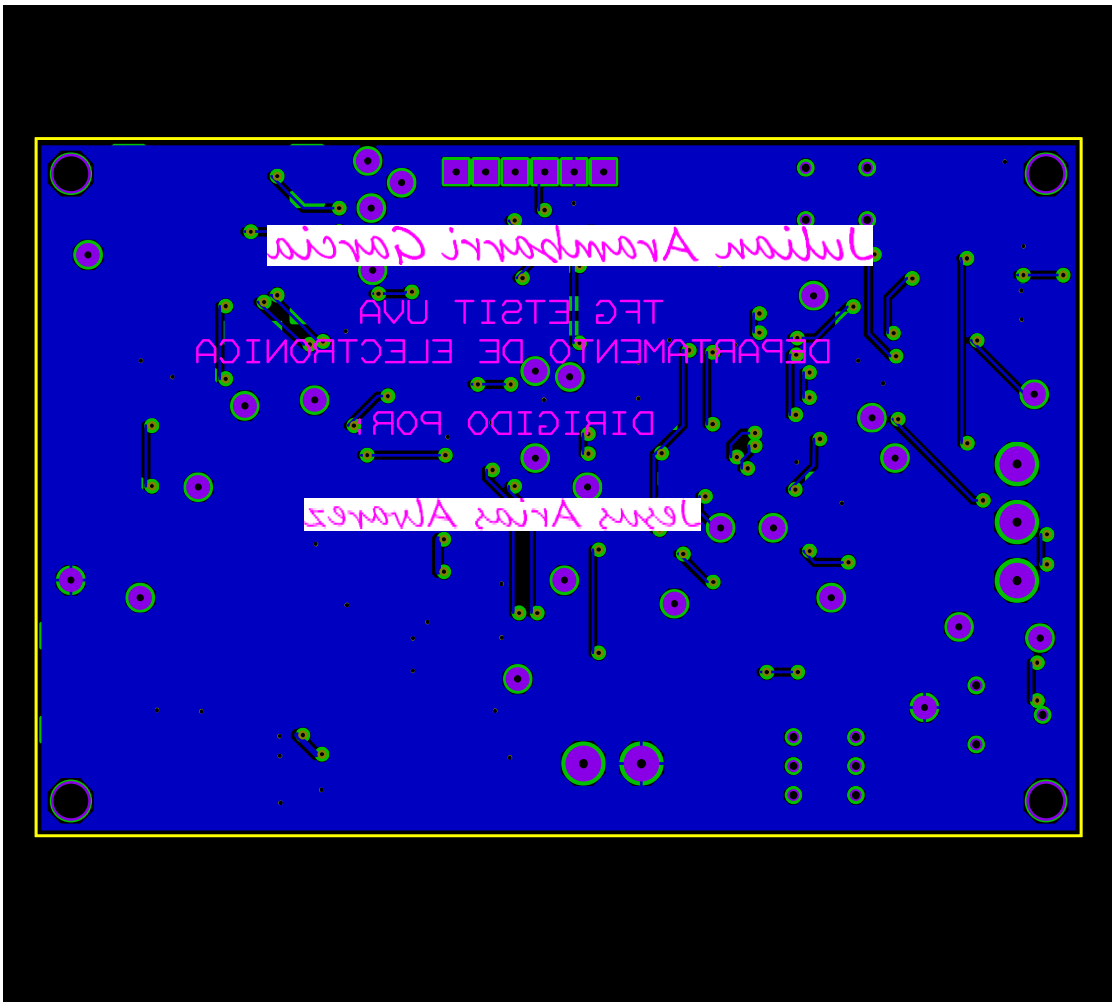


Figura 3.9: Cara inferior de la PCB diseñada.

Por último, se han incluido unas serigrafías con el nombre del tutor académico, del autor, de la escuela, el nombre del grado y una serigrafía con el escudo de la ETSIT UVA. Después se han generado los ficheros Gerber y estos se han enviado al fabricante de PCBs seleccionado, estos ficheros contienen toda la información que se necesita para la fabricación de las placas de circuito impreso.

3.3 FABRICACIÓN PCB

La fabricación de la placa se ha llevado acabo por una empresa asiática llamada PCBWay [9]. Tiene su fábrica situada en Shenzhen, en China. En su página web podemos encontrar las distintas opciones, acabados y precios que tienen en función del circuito que se quiera fabricar. Además de sus múltiples opciones de PCBs permiten en caso de que se quiera el montaje de los componentes sobre al PCB. En este caso se ha encargado un pedido de 5 PCBs diseñadas anteriormente.

Una de las principales ventajas de este fabricante es su bajo precio y una calidad alta teniendo en cuenta el coste. El pedido tardo en llegar menos de una semana desde su encargo.

3.4 MONTAJE DE COMPONENTES

El montaje de la PCB se realizó en el laboratorio de proyectos de la ETSIT de la Universidad de Valladolid. Esta aula cuenta con soldadores de precisión, microscopios, estaño, flux y otras herramientas que ayudan al montaje y depuración de PCBs.

Cuando se va a comenzar el montaje es aconsejable establecer una estrategia y organizarse. Para esto es buena idea decidir por qué zona o por qué tipo de componentes se va a empezar a soldar. Por ejemplo, empezar a soldar todos los componentes relacionados con la alimentación y comprobar si esta funciona antes de continuar con el posicionamiento del resto de componentes. Es conveniente en todo momento contar con el esquemático, el B.O.M. y el *layout* de la PCB impresos de manera que se sepa que componente se está montando y apuntar cuales se han posicionado y cuáles no.

En los circuitos integrados es de vital importancia fijarse en la orientación y el sentido de los componentes ya que estos suelen ser simétricos y se pueden cometer errores. Es recomendable ayudarse de la indicación de la serigrafía de la PCB y de las marcas que suelen incluir los fabricantes en sus componentes para reconocer la orientación. En la mayoría de los casos cuando se orienta mal un componente este se ve alimentado con tensiones negativas por lo que se estropea y es necesario sustituirlo. Durante el montaje se colocó mal la memoria EEPROM y al depurar el hardware con la alimentación conectada este se estropeó y hubo que modificarlo por un circuito integrado nuevo bien orientado.

Es muy importante asegurarse de que los terminales o patillas de todos los componentes que se sueldan quedan bien unidos a los *pads*. Para esto es recomendable ayudarse de los microscopios y pinzas para comprobar la unión. En el caso de los chips es muy conveniente ayudarse de flux a la hora de soldar sus patillas ya que ayuda a eliminar óxidos de los *pads*, permite concentrar el calor para que el estaño de distribuya de manera uniforme en la unión, y ayuda a que el componente no se mueva durante la soldadura de la primera patilla. En la figura 3.10 se observa la PCB una vez que se han soldado todos sus componentes.

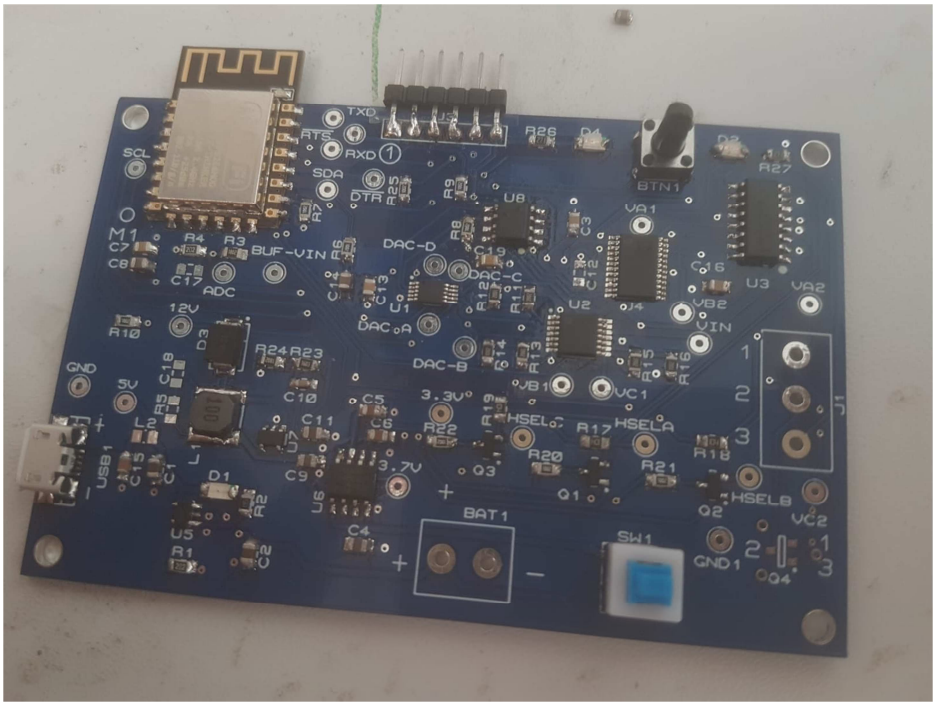


Figura 3.10: PCB con los componentes soldados.

Además de montar la PCB que se diseñó, hubo que montar la pequeña PCB, como se muestra en la figura 3.11, que facilitó el departamento de electrónica de la universidad que permite programar el microprocesador mediante un convertidor USB-serie.

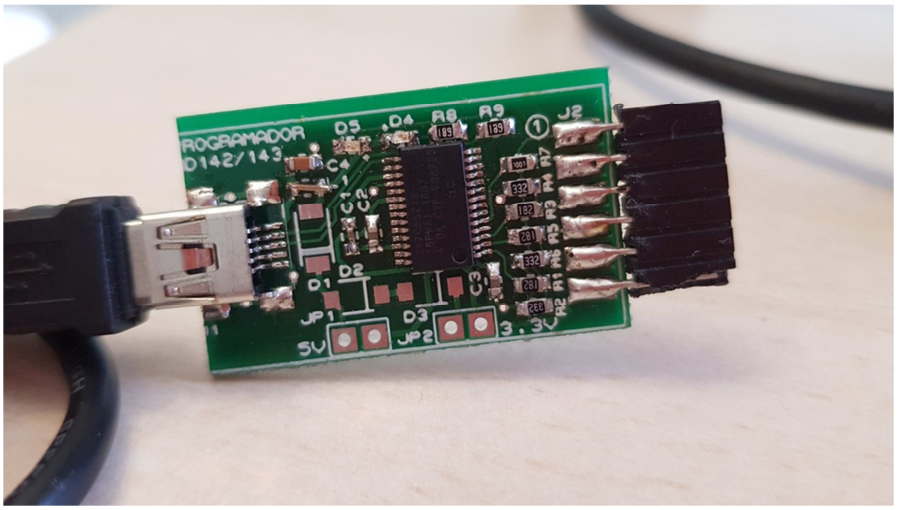


Figura 3.11: PCB conversor USB-serie con los componentes soldados.

3.5 Verificación del hardware

Para comprobar el correcto funcionamiento del hardware en primer lugar se alimentó al equipo a 3,7v mediante una fuente de alimentación ya que aun no se disponía de la batería. Una vez alimentado se comprobaron las tensiones de alimentación del circuito para comprobar que fueran correctas.

En esta primera fase de verificación del hardware se encontró un fallo en la alimentación de 3,3v. Este fallo se debía a la orientación errónea de la memoria EEPROM como se comentó anteriormente. Una vez instalada la nueva memoria correctamente se comprobó que la alimentación llegaba a cada componente correctamente.

A continuación, se comprobó que la PCB del conversor USB a bus serie era detectada por el ordenador al conectarlo a uno de sus USBs.

Posteriormente se conectó esta pequeña PCB a la PCB de equipo mediante un anclaje tipo peine que se instaló durante el montaje. Una vez que se tiene conectado el ordenador al equipo y se es capaz de programar el microprocesador se pueden crear rutinas software que prueben y verifiquen el funcionamiento de los diferentes componentes del equipo. Durante este proceso se encontró una serie de errores de diseño, no obstante, se pudieron solucionar sin necesidad de realizar un nuevo diseño y pedido de PCBs.

3.6 Errores de diseño

Durante el proceso de verificación del funcionamiento del hardware se encontraron los siguientes errores:

- Lo primero que se detectó podría no considerarse un error, no obstante, en una futura iteración del producto sería conveniente modificarlo. Se trata de que los pines que interconectan la PCB del equipo de medida con el módulo programador USB-UART estaban colocados de manera contraria en una placa respecto la otra. Es decir, solo se logra un correcto conexionado si cunado se conecta una placa está boca arriba y la otra boca abajo como se muestra en la figura 3.12.

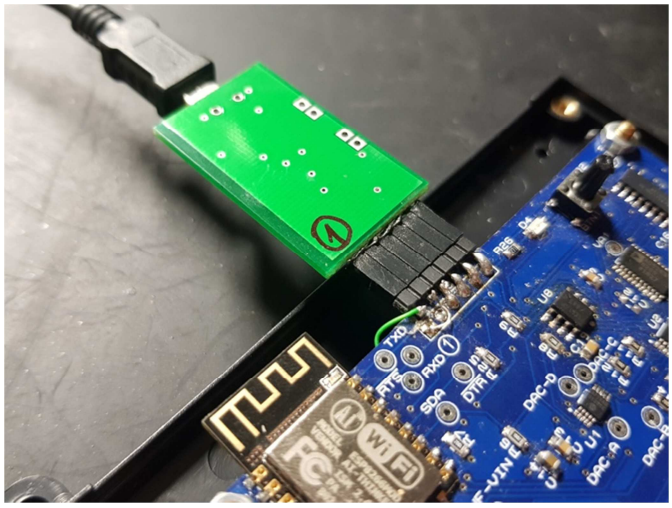


Figura 3.12: Una PCB boca arriba y otra boca abajo.

- El segundo error se encontró al no poder programar el microprocesador. El diseño que se había realizado conectaba el pin de la interfaz serie del microprocesador TX al terminal del convertidor CY7C65213 TX, y el terminal RX al RX. Para el correcto funcionamiento del protocolo UART el pin TX del primer dispositivo, el que transmite, debe estar conectado al pin RX del segundo dispositivo, el que recibe. De la misma manera el pin RX del primer dispositivo debe estar conectado al TX del segundo dispositivo como se muestra en la figura 3.13.

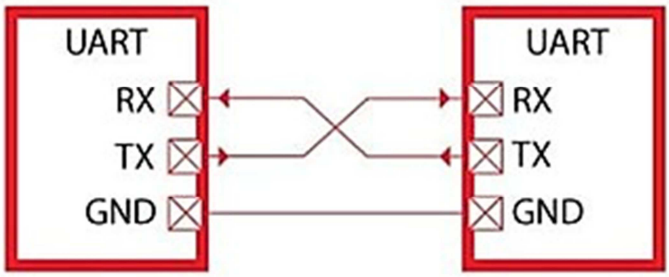


Figura 3.13: Conexionado de la UART.

Este error se solucionó realizando unos cortes en el peine de pines de la PCB que permitieron intercambiar el terminal RX por el TX y viceversa. El resultado se muestra en la figura 3.14.

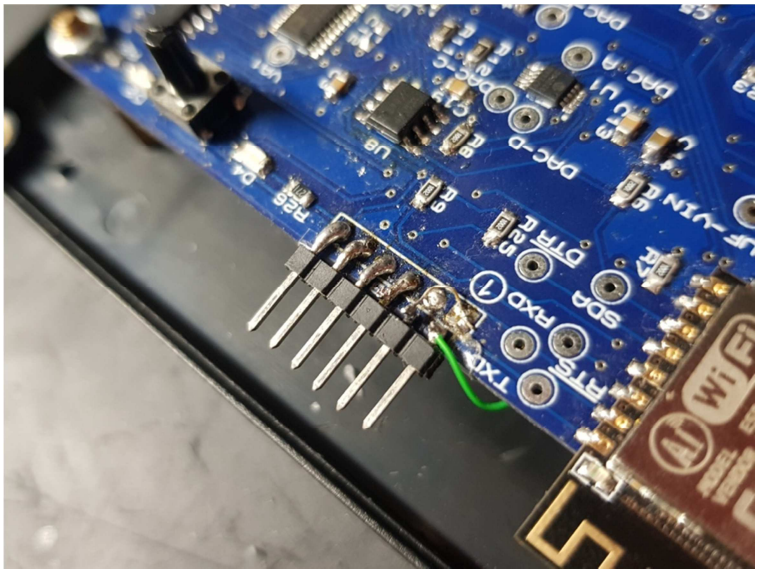


Figura 3.14: Solución conexionado TX y RX.

- El último error se encontró mediante rutinas software que permitían comprobar el funcionamiento de las distintas partes del equipo. No se era capaz de seleccionar el canal deseado del multiplexor, siempre estaba seleccionado el mismo. Este error causado por los convertidores de nivel era el producto de una mala asignación de pines en la huella de los 3 transistores MMBT3904. Los terminales del transistor bipolar base, colector y emisor estaban conectador a los *pads* que correspondían al emisor, base y colector respectivamente. Afortunadamente este fallo se pudo solucionar como se

muestra en la figura 3.15 rotando los transistores 120 grados en sentido horario de manera que los pines del transistor encajasen con su *pad* correspondiente.

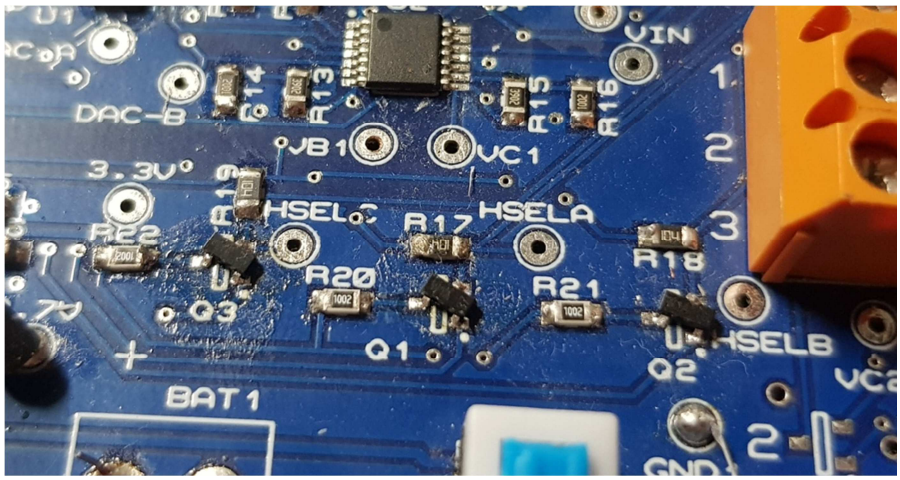


Figura 3.15: Transistores rotados 120º para lograr la asignación de *pad* correcta.

3.7 Bill of materials

Todos los componentes utilizados para la fabricación de la PCB se recogen en un listado de materiales denominado B.O.M. por sus siglas en ingles las cuales hacen referencia a *bill of materials*, figura 3.16. En este documento se muestra el nombre de todos los componentes, la cantidad de estos que se necesitan para una PCB, la referencia que tienen dentro de la PCB, su valor en caso de que sea necesario como es el caso de las resistencias y condensadores, el fabricante, el número de referencia del fabricante, el encapsulado y por último su precio.

Esto permite a la empresa calcular de una forma más simple los costos asociados a la fabricación de las PCBs y facilita realizar los pedidos al suministrador de componentes. Hay que recordar que para la estimación del precio de un producto real que se va a sacar al mercado, al coste de los componentes habría que sumar el precio de las PCBs, el tiempo de trabajo de los diseñadores y numerosas variables más que puedan afectar al precio del producto. Los precios que se muestra en el bill of material se reducirían en gran medida en algunos casos superando el 40% de ahorro si en lugar de comprar pocos componentes se compran a gran escala.

Bill Of Materials for v1

Design Title v1
Author
Document Number
Revision
Assembly Variant excl. BOM
Design Created lunes, 21 de octubre de 2019
Design Last Modified lunes, 2 de diciembre de 2019
Total Parts In Design 68

1 Modules

Quantity	References	Value	Stock Code	PCB Package	Unit Cost	encapsulado
1	M1	ESP-12F		ESP-12F	€2,00	EP-12 MICRO
Sub-totals:					€2,00	

19 Capacitors

Quantity	References	Value	Stock Code	PCB Package	Unit Cost	encapsulado
5	C1-C2,C4,C9,C11	4.7uF	Digikey 478-1000-1-ND	0805		
4	C3,C8,C12,C16	100nF	Digikey 478-1000-1-ND	0805		
2	C5-C6	1uF	Digikey 478-1000-1-ND	0805		
2	C7,C14	10uF	Digikey 478-1000-1-ND	0805		
1	C10	120pF	Digikey 478-1000-1-ND	0805		
1	C13	0.1uF	Digikey 478-1000-1-ND	0805		
1	C15	10nF	Digikey 478-1000-1-ND	0805		
1	C17	1.5nF	Digikey 478-1000-1-ND	0805		
1	C18	1000pF	Digikey 478-1000-1-ND	0805		
1	C19	100nF		0805		
Sub-totals:					€0,00	

27 Resistors

Quantity	References	Value	Stock Code	PCB Package	Unit Cost	encapsulado
1	R1	2k		0805		
1	R2	560		0805		
2	R3,R24	18k .1%	2483962	0805	€0,39	
1	R4	2k .1%	1160154	0805	€0,44	
10	R5-R10,R20-R22,R25	10k		0805		
3	R11,R13,R15	39k .1%	2483993	0805	€0,36	
3	R12,R14,R16	10k .1%	2483951	0805	€0,42	
3	R17-R19	100k		0805		
1	R23	160k .1%	2337639	0805	€0,16	
2	R26-R27	270		0805		
Sub-totals:					€3,72	

8 Integrated Circuits

Quantity	References	Value	Stock Code	PCB Package	Unit Cost	encapsulado
1	U1	MCP4728A0	2764386	SOP50P490X95-10	€1,72	MSOP
1	U2	MAX44243AUD+	2510892	SOP65P640X120-14	€4,63	TSSOP
1	U3	CD4051BM96	3121046	SO16	€0,50	SOIC-16
1	U4	AD5263BRU200	1438439	TSSOP24	€7,14	TSSOP-24
1	U5	MCP-73831T-2ACI/OT	1332158	SOT23-5	€0,50	SOT23-5
1	U6	MCP1725-3302ESN	1439383	SO8	€0,47	SOIC-8
1	U7	FAN5331	1858084	SOT23-5	€0,87	SOT23-5
1	U8	M24C64	9882685	SO8	€0,25	SOIC-8
Sub-totals:					€16,08	

3 Transistors

Quantity	References	Value	Stock Code	PCB Package	Unit Cost	encapsulado
3	Q1-Q3	MMBT3904	1757935	SOT95P230X109-3	€0,10	SOT-23
Sub-totals:					€0,30	

4 Diodes

Quantity	References	Value	Stock Code	PCB Package	Unit Cost	encapsulado
1	D1	DIODE-LED		1206	€0,00	1206
2	D2,D4	LED		1206	€0,00	1206
1	D3	B130B-13	1858651	DIOM5336X240	€0,34	B130LB
Sub-totals:					€0,34	

6 Miscellaneous

Quantity	References	Value	Stock Code	PCB Package	Unit Cost	encapsulado
1	BAT1	BATERY		BAT900MAH	€3,50	
1	J1	TBLOCK-M3		TBLOCK-M3		
1	J3	CONN-H6		CONN-SIL6	€0,00	

Figura 3.16: Bill Of Materials.

3.8 Consumo eléctrico

Es muy importante realizar una estimación del consumo eléctrico del equipo electrónico de manera que sepamos antes de fabricarlo si este va a cumplir con las especificaciones propuestas. Una estimación del caso más desfavorable en el que nuestro equipo tenga un gran consumo nos permitirá saber si la fuente de alimentación que se va a utilizar sea interna o externa es suficiente y en caso de que se cuente con una batería saber de cuanto tiempo de uso aproximado se va a disponer.

A continuación, en la figura 3.17 se recogen los voltajes y las corrientes máximas consumidas por cada componente de manera que se realice una aproximación de las horas de utilización de las que se va a disponer con la batería del equipo de 800 mAh a 3,7V.

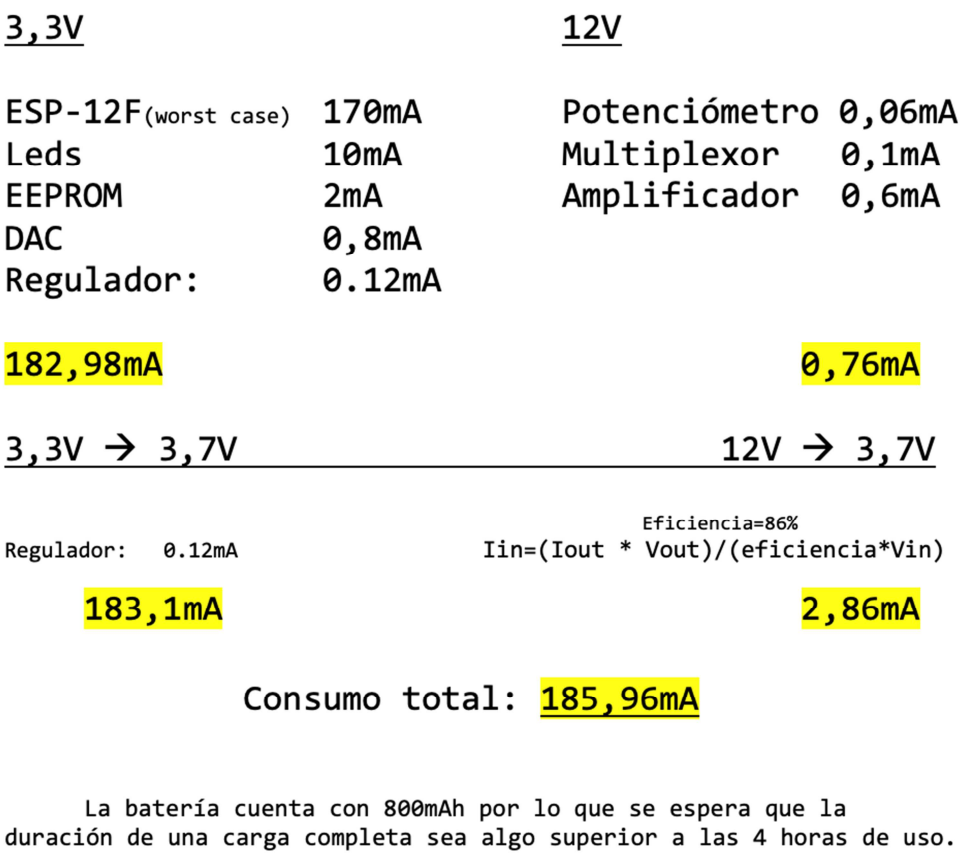


Figura 3.17: Estimación del consumo y la autonomía.

3.9 Fuente de alimentación casera

Debido al confinamiento provocado por la expansión del COVID-19, deje de tener a mi disposición la fuente de alimentación del laboratorio de la universidad. No contaba en esa fecha con la batería encargada a un proveedor asiático, ni tenía la bobina L2 conectada al circuito que me hubiera permitido alimentar el equipo mediante el USB.

Los voltajes habituales de las fuentes de alimentación comerciales son 5V ,12V, 24V. Especialmente la de 24v que es un tipo de fuente de alimentación muy usada en el entorno industrial. Disponía de una fuente de alimentación lineal de 24w que podía suministrar 1A a 24V. Las fuentes lineales (figura3.18) no son fuentes eficientes, pero bajo esas circunstancias era lo único que tenía.

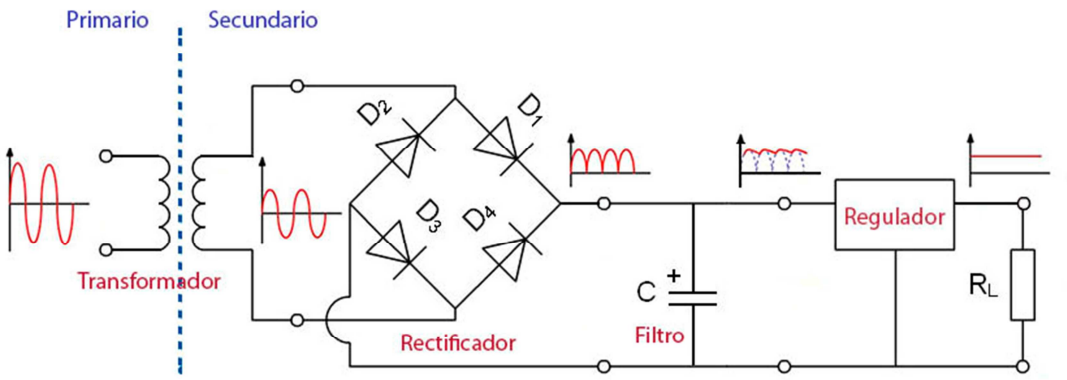


Figura 3.18: Fuente de alimentación lineal.

Básicamente existen dos tipos de fuentes de alimentación, las lineales, que utilizan un transformador para disminuir el nivel de tensión en la red eléctrica al nivel necesario en nuestro circuito y las fuentes conmutadas que utilizan circuitos basados en bobinas y transistores trabajando en conmutación para reducir o aumentar la tensión. Las ventajas de la fuente de alimentación lineal es su sencillez y que generan menor ruido electromagnético, las desventajas son su mayor tamaño y su menor eficiencia (disipan más energía en forma de calor que las fuentes conmutadas).

Desmontando la fuente industrial, estudie su comportamiento y saque el esquema analizando los componentes que llevaba. Como se ve en el esquema adjunto en la figura 3.19 la fuente es extremadamente sencilla.

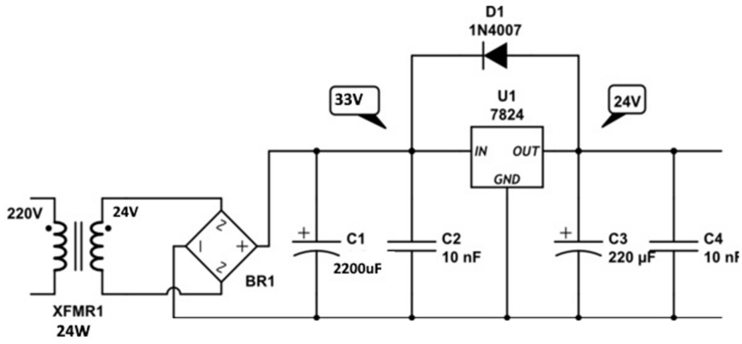


Figura 3.19: Fuente de alimentación lineal de la que se disponía.

En este momento ya disponia de 24V estabilizados con lo cual solo me quedaba bajar los 24V a los 3.7V que necesitaba para alimentar mi placa. Para cual compre un módulo conmutado para reducir el voltaje. Al final me decidí por el módulo LM2596 [10] un convertidor DC a DC (stepdown regulado) ya montado en un módulo con un coste de 0,75€ y con unas buenas carcteristicas. Entre sus especificaciones destaca por una eficiencia superior al 92%, corriente de salida máxima de 2A (3A con disipador), un voltaje de entrada comprendido entre 4 y 40V, un voltaje de salida comprendeido entre 1,23 y 35V y por último un dropout mínimo de 2V. Como se puede observar en el esquemático de la figura 3.20 ajustando VR1 se puede variar la tensión de salida OUT+. Conectado finalmente estas dos fuentes de alimentación se pudo ajustar el potenciometro VR1 hasta que en la salida en la que se medía con un multimetro se obtuvo 3,7V.

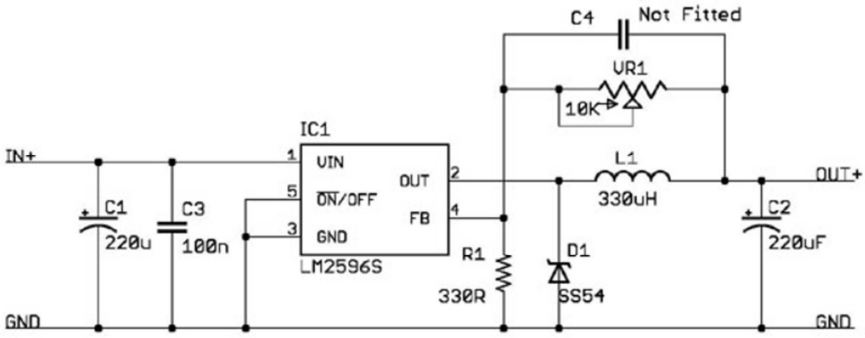


Figura 3.20: Esquemático LM2595.

4 Desarrollo Software

4.1 Introducción al desarrollo software

En los apartados anteriores se ha detallado toda la estructura del hardware del proyecto, una vez se consigue desarrollar el hardware se procede a implementar el software que consiga en primer lugar comprobar que los diferentes bloques funcionan como se espera y conseguir que el producto final realice la función esperada de la forma más optima posible.

El desarrollo del software sin duda es la parte más larga del proyecto y más difícil de estimar en duración. Pese a que no haya sido este el caso se recomienda que se comience a desarrollar el software, al menos una estructura básica durante la creación del hardware con el fin de paralelizar el trabajo.

4.2 Entorno de trabajo

Para el desarrollo del código de este micro se recurren a unas librerías que permiten que el micro sea programado mediante la plataforma Arduino, la cual emplea un lenguaje de alto nivel muy similar a C++.

Se comenzó empleando IDE de Arduino, el cual permite de manera muy sencilla crear programas, compilarlos, cargarlos en el dispositivo y monitorizar el funcionamiento mediante un terminal serie. Para que este entorno funcione con el microcontrolador ESP-12F utilizado, es necesario instalar un *plugin* creado por la comunidad de desarrolladores [11]. Esta instalación es muy sencilla y una vez completada permite configurar diferentes opciones relacionadas con la frecuencia de reloj del equipo, la frecuencia de la memoria SPI *flash* a la hora de subir el código, el puerto de programación y numerosos ajustes más. Es conveniente detenerse a configurar algunos de estos parámetros para que ajustar la frecuencia de reloj a la más alta posible, y que el código que se ha desarrollado una vez compilado tarde considerablemente menos en cargarse en el equipo.

Pese a que el entorno de Arduino es muy sencillo de utilizar, puede llegar a ser demasiado simple, y a medida que el código va creciendo es complicado situar partes del código. En su lagar se ha empleado el IDE de Visual Studio Code desarrollado por Microsoft. Este editor de código fuente gratuito es el más utilizado actualmente por los desarrolladores de todo el mundo. La principal ventaja de Visual Studio Code es que cuenta con numerosas extensiones que ayudan a la hora de visualizar el código y los errores para diferentes lenguajes de programación.

La extensión que se ha empleado ha sido PlataformIO (figura 4.1), la cual cuenta con casi un millón de descargas dentro de Visual Studio Code [12]. Mediante esta extensión tenemos numerosas funcionalidades, compilar el código, subirlo al dispositivo, monitorizarlo e incluso ver el código en ensamblador que se genera entre otras.

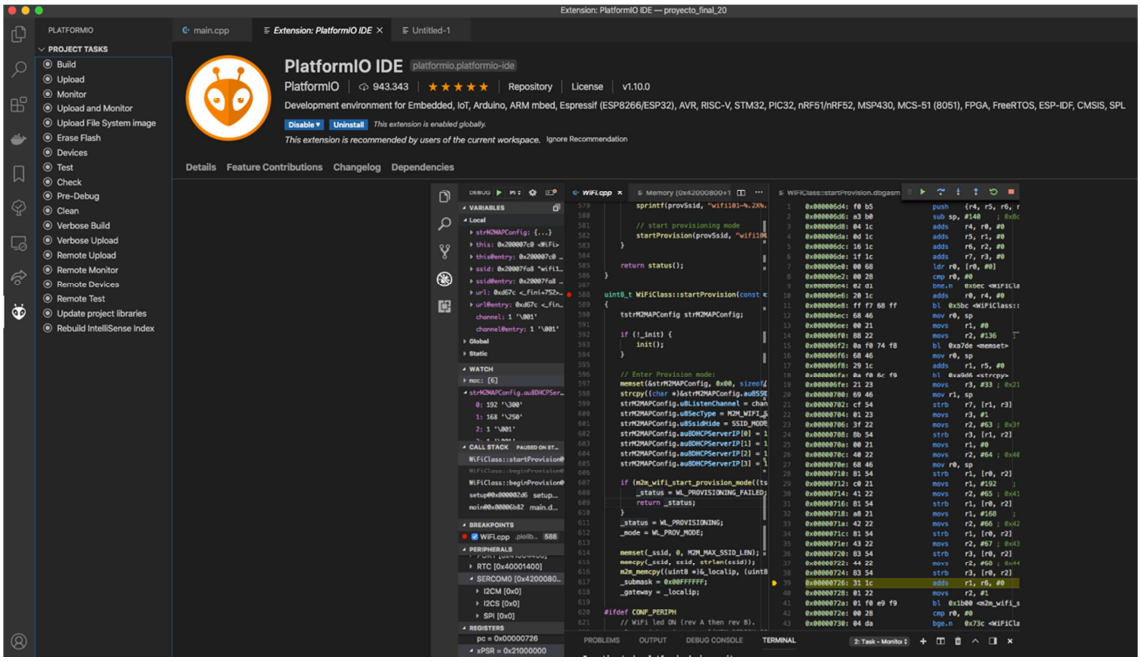


Figura 4.1: Extensión de PlataformIO.

Esta extensión nos exige que el fichero del código este en la carpeta SRC dentro de un proyecto como se muestra en la siguiente figura 4.2.

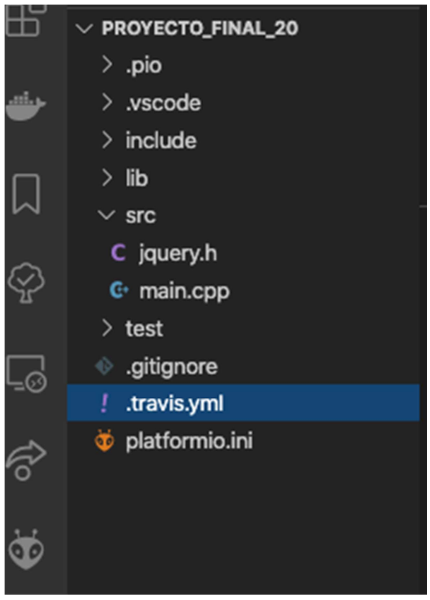


Figura 4.2: Estructura de las carpetas del proyecto.

Por último, es necesario que dentro del fichero plataformio.ini (figura 4.3) se asignen ciertos parámetros como el microcontrolador que se está utilizando, la frecuencia del monitor y el puerto de programación, al igual que en el IDE de Arduino. La extensión cuenta con la funcionalidad *Devices* que nos permite comprobar que puertos están disponibles para subir el código al equipo.

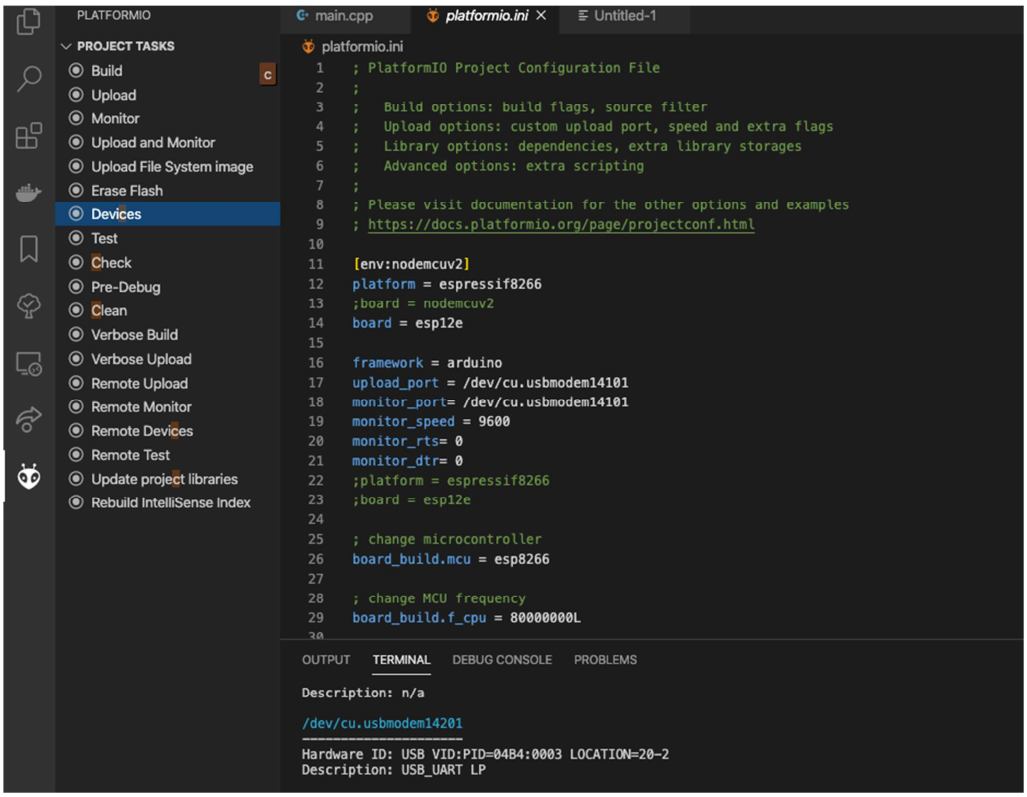


Figura 4.3: Archivo plaraforio.ini y salida del terminal después de usar la herramienta *Devices*.

4.3 Interconexiones

Antes de comenzar con el diseño del programa principal se comprueba el funcionamiento de los diferentes componentes interconectados con el microcontrolador por medio del protocolo I2C.

I2C también conocido como TWI (*Two Wire Interface*), es un protocolo síncrono que utiliza dos líneas, la de reloj (SCL) y otra para datos (SDA), por lo que el maestro (puede haber más de uno) y los esclavos envían datos por la misma línea. Este protocolo fue desarrollado por Philips en la década de los 80s para poder interconectar componentes dentro de los televisores de la época.

Los mensajes de I2C están organizados en tramas (figura 4.4), estas tramas están divididas a su vez en campos de bits:

1. **Condición de inicio**: la línea de datos (SDA) cambia de un nivel lógico alto a uno bajo antes de que la señal de reloj cambie también de estado lógico, pero esta vez de estado bajo a alto.
2. **Dirección**: en este campo de 7 o 10 bits se envía la dirección del esclavo con la que el maestro se quiere comunicar, solo el esclavo con esta dirección contestará.
3. **Bit lectura/escritura**: mediante este bit el maestro indica al esclavo si quiere realizar una operación de lectura (nivel lógico alto) o una operación de escritura (nivel lógico bajo)
4. **Bit ACK/NACK**: después de cada sección el equipo que ha recibido los bits anteriores envía un bit de ACK en caso de que el mensaje se haya recibido y reconocido correctamente para que se sigan enviando mensajes, o un bit de NACK en el caso contrario.
5. **Datos**: en las siguientes secciones de la trama se enviarán o recibirán datos de 8 bits e irán seguidas cada una de un bit de ACK o NACK.
6. **Condición de paro**: por último, cuando se va a terminar la comunicación la línea de datos (SDA) pasa de un nivel lógico bajo a alto después de que la línea de reloj (SCL) haya pasado de bajo a alto.

Figura 4.4: Campos que componen las tramas del protocolo I2C.

En este proyecto se va a emplear un solo maestro el ESP-12f y 3 esclavos que van a ser el potenciómetro digital AD5263, el DAC MCP4728, y la memoria EEPROM serie M24C64. Existen 4 tipos de velocidades estándar en las que puede trabajar I2C, estas van desde los 100kbps (modo estándar) hasta los 5Mbps (modo ultra *fast*). Puesto que no es crítico obtener una velocidad muy elevada ni se van a enviar una cantidad muy elevada de datos se opta por la velocidad estándar de 100kbps.

El microprocesador empleado no cuenta con unas líneas dedicadas para el protocolo I2C, pero dentro de las librerías de Arduino diseñadas por la comunidad existe una librería llamada Wire.h que cuenta con una serie de código que facilita mucho el trabajo del desarrollador mediante una serie de funciones. La librería Wire.h se puede encontrar dentro del repositorio de Arduino en GitHub (herramienta de control de versiones de Microsoft). Wire.h recibe este nombre debido al otro nombre por el que se le conoce a I2C, TWI (*Two Wire Interface*), está implementa el protocolo por medio de *bit bangin*, el cual es un método de transmisión de datos mediante software cuando no se dispone de hardware dedicado. Las funciones utilizadas de esta librería han sido:

1. **Wire.begin(int sda, int scl)**: con esta función al comienzo del programa seleccionamos que dos pines del microcontrolador queremos que se ocupen de las líneas de datos y reloj.
2. **Wire.beginTransmission(byte address)**: nos permite iniciar la comunicación teniendo solo que centrarnos en enviar correctamente la dirección del esclavo, esta función se encarga de realizar la condición de inicio.
3. **Wire.write(byte value)**: esta función nos permite enviar datos al esclavo. Los datos que se le envían al esclavo han sido obtenidos de las hojas de especificaciones de los distintos componentes. En estos documentos se muestran las diferentes formas de trabajar de los chips, así como sus direcciones y como configurarlos para obtener el resultado que se espera de ellos.

4. **Wire.endTransmision( )**: Por último se cierra la comunicación entre el maestro y el esclavo mediante esta rutina.

4.4 Funciones básicas

El proceso del desarrollo del software se comenzó realizando unas funciones que permiten comprobar el correcto funcionamiento del equipo y que a la hora de realizar el programa principal se pueda llamar a estas funciones de más bajo nivel y el programador se pueda centrar solo en el flujo principal. Las funciones básicas son:

1. **float mideTerminal (byte canal)**: a esta función le tenemos que pasar el número de 1 de los 6 canales del multiplexor para medir las señales de ese canal mediante el ADC del microprocesador. La función nos devuelve una variable de tipo *float* con el voltaje en voltios que se ha medido. Se realizan varias medidas para promediar el valor y eliminar ruido. La selección del canal del multiplexor se realiza mediante 3 pines ajustados como salida del micro que mediante los convertidores de nivel indicados anteriormente permiten escoger el canal del multiplexor analógico.
2. **float measureI (byte canal1, byte canal2, float POTresist)**: esta función emplea a la anterior para calcular la corriente circula en uno de los terminales del transistor mediante la ley de Ohm. Como variables de entrada recibe dos canales del multiplexor que y la resistencia de uno de los potenciómetros
3. **void POTset (byte iPOT, byte datPOT)**: mediante las funciones de la librería Wire.h comentadas anteriormente, esta función con los parámetros de entrada indicados selecciona uno de los cuatro potenciómetros mediante la variable iPOT, y ajusta su resistencia al valor indicado por datPOT.
4. **void DACset (byte iDAC, int valorDAC)**: esta función es muy similar a la anterior y mediante las variables de entrada selecciona uno de los DACs y ajusta la salida de este a la tensión que se indique.

4.5 WiFi

Antes de comenzar a explicar el funcionamiento del programa principal es importante recordar que contamos con una interfaz WiFi en el módulo ESP-12F, esto es lo que va a permitir mostrar los resultados y las medidas realizados por el producto mediante un servidor Web. El interfaz wifi de la que se dispone tiene varios modos de funcionamiento como se muestra en la figura 4.5.

Parámetro del método *WiFi.mode()*	**Modo de funcionamiento**
WIFI_AP	El módulo ESP-12F genera su propia red WiFi permitiendo conectarse a otros dispositivos.
WIFI_STA	El ESP-12F es capaz de conectarse a redes WiFi existentes.
WIFI_AP_STA	Modo dual: funcionan a la vez el modo STATION y el modo AP
WIFI_OFF	No se genera señal WiFi

Figura 4.5: Modos de funcionamiento de la interfaz WiFi del ESP-12F.

En función del entorno en el que nos encontremos puede ser más interesante un modo de funcionamiento u otro. El método final que se ha decidido seleccionar ha sido WIFI_STA, el cual se conecta al WiFi de mi casa o a la que se le haya indicado en el código. Gracias a esta solución no es necesario estar próximo al equipo para conectarse, desde cualquier parte de la casa con acceso a internet seremos capaces de conectarnos al equipo y realizar o comprobar medidas.

Además, en mi *router* domestico he realizado un redireccionamiento de puertos de manera que se puedan comprobar y empezar medidas del equipo desde cualquier parte del mundo siempre que se disponga de acceso a internet, mi IP publica y el puerto abierto al servidor del producto.

Este redireccionamiento de puertos ha sido posible porque mi suministrador de internet aun ofrece una IP publica para los *routers* de sus clientes y no una privada debido a la tecnología CGNAT. En la actualidad cada vez más operadoras emplean CGNAT (*Carrier Grade Network Address Translation*), una tecnología que genera un NAT con varios *routers* de clientes, es decir un NAT dentro de otro NAT. Esto provoca que el *router* del cliente tenga una IP privada a la que sería mucho más complicado

acceder, de esta manera se reducen las IP públicas en uso. Existe escasez de direcciones públicas que se debe a el número limitado de direcciones IP que facilita IPv4 para todos los dispositivos que existen en la actualidad. Se espera que CGNAT sea una solución temporal y existe una iniciativa por parte de la Unión Europea para que CGNAT se deje de utilizar cuando el uso de IPv6 se expanda de manera masiva.

Para las funcionalidades de red utilizaremos la librería ESP8266WiFi.h la cual incluye las funciones necesarias para que el microprocesador utilizado se conecte al WiFi que se le indique y se hace cargo de los protocolos 802.11 y TCP/IP. Este dispositivo solo es capaz de conectarse a versiones del estándar WiFi en las que al menos alguna de sus bandas trabaje en los 2.4GHz, no es compatible con la banda de 5GHz como como la que utiliza 802.11ac.

En la función **void setup ( )** la cual solo se ejecuta una vez al comienzo de la ejecución en todos los programas de Arduino, se llama (figura 4.6) a la función **void init_internet ( )** donde se emplean las funciones necesarias de la librería ESP8266WiFi.h para conectarse al WiFi.

```
void init_internet()
{

  WiFi.mode(WIFI_STA);
  WiFi.begin("vodafone0DC8", "PijusMagnificus");
  Serial.println("Connecting");

  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println();
  Serial.print("Connected, IP address: ");
  Serial.println(WiFi.localIP());

  // Mostrar mensaje de exito y dirección IP asignada
  Serial.println();
  Serial.print("Conectado a:\t");
  Serial.println(WiFi.SSID());
  Serial.print("IP address:\t");
  Serial.println(WiFi.localIP());

  return;
}
```

Figura 4.6: Código que permite conectarse al WiFi.

4.6 Servidor Web

La conexión a internet es fundamental para que los usuarios puedan visualizar el servidor web que el microcontrolador genera. Para gestionar un servidor HTTP que recibe y atiende peticiones también se cuenta con una librería ESP8266WebServer.h.

Esta librería permite crear el servidor web y definir distintas URIs a las que los clientes harán peticiones HTTP por el puerto 80.

Al igual que con la inicialización del WiFi, el servidor Web lo inicializo en la función **void setup ( )** al llamar a la función **void init_server( )**, que emplea las funciones de la librería ESP8266WebServer.h como se muestra en la figura 4.7. De esta manera desde que se inicia el dispositivo está en funcionamiento, y el usuario puede conectarse mediante un navegador desde su ordenador o Smartphone.

```
void init_server()
{

  server.on("/", HandleRoot);
  server.on("/U", ActualizaGrafico);
  server.on("/BOTON", IniciaProceso);
  server.on("/JQ.js", []() {
    server.send(200, "text/javascript", JQ, sizeof(JQ) - 1);
  });
  // Ruteo para URI desconocida
  server.onNotFound(HandleNotFound);
  // Iniciar servidor

  server.begin();
  Serial.println("HTTP server started");

  return;
}
```

Figura 4.7: Inicialización del servidor Web.

A lo largo de todo el código desarrollado se pueden encontrar llamadas a la función **handleClient( )**, esta función nos permite responder a las peticiones de los clientes. De esta manera mientras el usuario está conectado al servidor la página web que este visualiza se irá actualizando con las gráficas y la información de los transistores detectados. Además se podrá interactuar con un botón desde la página web que le permita iniciar la medida (la medida también puede iniciarse desde un botón físico del equipo).

La página web está programada en HTML (*HyperText Markup Language*), un lenguaje muy utilizado en sus diferentes versiones para la elaboración de páginas web con estructuras sencillas [13]. El código de HTML realizado incorpora una función en Javascript que permite mostrar las gráficas mediante la librería Flot. Esta librería en de código abierto que se puede encontrar en GitHub [14] permite realizar multitud te tipos de gráficas con diferentes opciones. Por medio de una API que incorpora podemos actualizar los datos de las gráficas y las leyendas. Todos los datos que se suministren han de estar en formato JSON (JavaScript *Objet Notation*), destaca por ser ligero y

rápido. Cuando se comienza a utilizar este formato es fácil equivocarse sobre todo cuando se trata de un texto largo como es el caso. Puede ser de mucha utilidad usar alguna página web que encuentre errores o ayude a dar este tipo de formato.

4.7 Desarrollo principal

En primer lugar antes de comenzar con el hilo principal se incluyen las librerías nombradas anteriormente, se definen las funciones que se van a utilizar, se definen las variables globales y por último se ejecuta la función **void setup ( )**.

Es después de esto cuando comienza el programa principal en el **void loop ( )**, esta función es donde se encuentra el programa a ejecutar y como su propio nombre indica se trata de un bucle. Este bucle es infinito y solo terminará con el apagado del equipo o si sucediese un error en el código que hiciese saltar una excepción, en ese caso el equipo se reiniciaría.

El funcionamiento lógico del programa como se muestra a continuación (figura 4.8) es relativamente sencillo. En primer lugar, se espera la pulsación del botón físico situado en la PCB o del botón del servidor web. Cuando esto sucede el equipo trata de identificar de qué tipo de componente discreto está conectado (transistor bipolar PNP, transistor MOSFET tipo N, diodo Zener, etc). Una vez detectado el componente se obtienen sus gráficas y su ganancia o transconductancia y todos estos datos se envían al servidor web para que el usuario pueda visualizarlos.

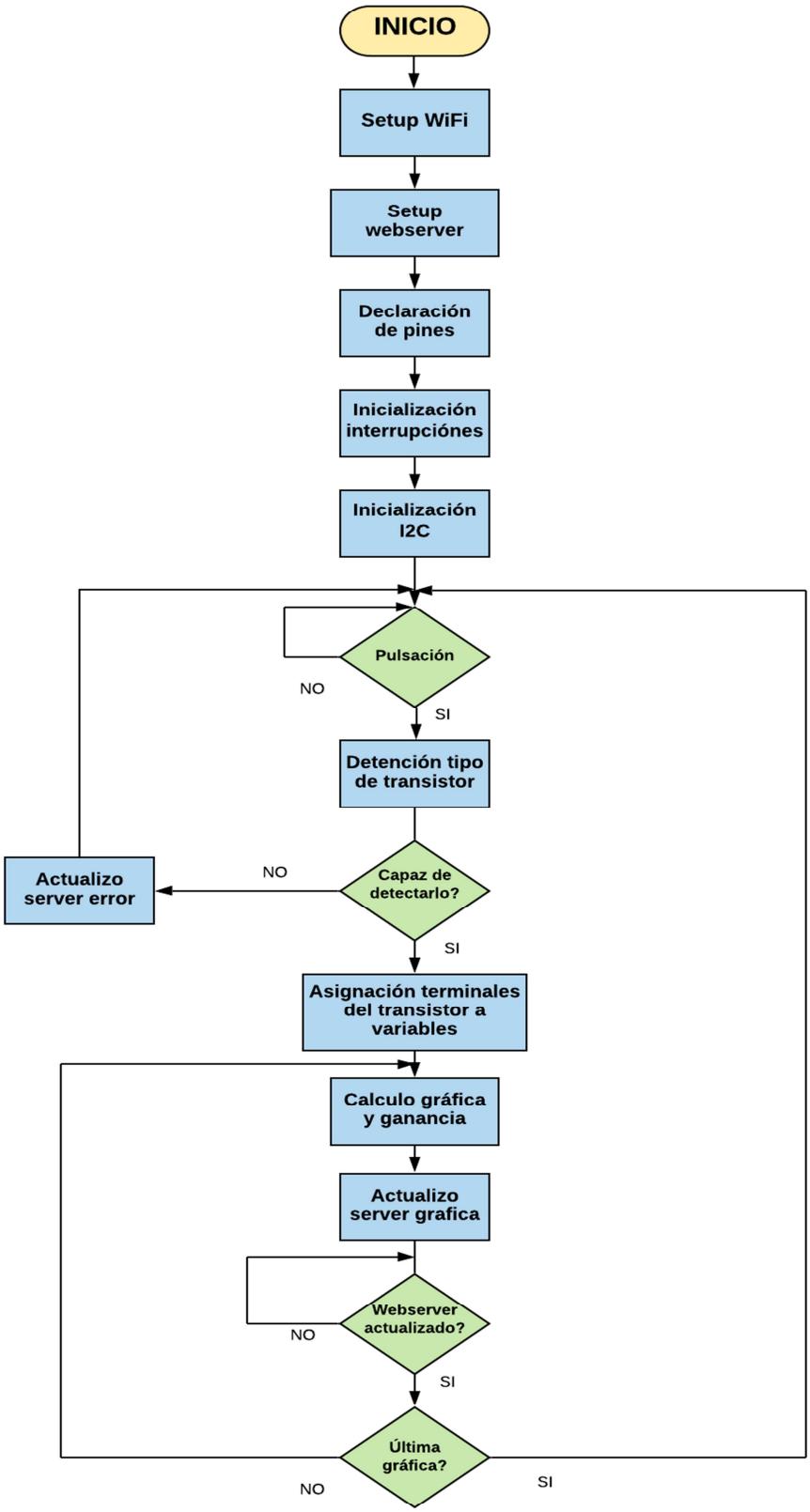


Figura 4.8: Diagrama del programa principal.

4.7.1 Detección

Una de las mayores complicaciones del proyecto es que no existe ninguna norma sobre cómo han de conectarse los componentes discretos en el equipo. Se dispone de 3 terminales sobre los que se conecta el componente a analizar. Estos componentes pueden disponer de 3 patillas (como es el caso de los transistores) o de 2 (como los diodos). El equipo es capaz de analizar transistores, BJT, MOSFET, y JFET de tipo N y P, y diodos estándar y Zener. Esto da un total de 8 componentes distintos, y cada componente se puede colocar de 6 maneras distintas, por lo que el equipo ha de ser lo suficientemente inteligente como para distinguir entre 48 situaciones diferentes. El problema no son solo el número de situaciones en las que se tenga que detectar, sino que al no colocar los transistores en una posición fija es difícil establecer una serie de reglas de detección excluyentes, es decir programar una serie de condiciones que solo un tipo de transistor cumple en una determinada posición, y ninguno de los demás tipos la cumpla sea cual sea la posición en la que se ha colocado.

A la hora de detectar el componente, se pasan todos los test de manera que en caso de que un componente se detecte como 2 o más posibles tipos de transistor o diodo, de un error ya que el equipo no es capaz de obtener con certeza de cual se trata. De la misma manera si el equipo al pasar todas las pruebas de detección no ha sido capaz de detectar ninguno, nos mostrará también un mensaje de error.

4.7.2 Obtención de gráficas

Cuando se ha detectado el componente de manera correcta y como están dispuestos sus terminales, se procede a obtener la gráfica característica del componente. Estas gráficas que normalmente se muestran como unas curvas en el eje X (voltaje Vds o Vce) y el eje Y (intensidad Id o Ic) tienen un tercer parámetro asociado a ellas que siempre es constante (Vgs o Ibase). Este tercer parámetro no es otra cosa que el eje Z, pero las gráficas características se muestran siempre solo con el eje X y Y por simplicidad ya que la Z de cada curva es constante a lo largo de ella.

Conociendo esto han de obtenerse las gráficas características de los transistores y diodos, esto requiere un estudio del comportamiento de cada uno de ellos, para lo cual también es útil el estudio necesario para la detección. Este estudio nos permite

determinar como polarizar los diferentes componentes de manera correcta y obtener unas gráficas con ejes X e Y que puedan ser de utilidad para el usuario.

En primer lugar independientemente del componente a analizar (siempre que no sea un diodo ya que en este caso no hay un tercer parámetro) se fija una corriente de base (BJT) o una tensión Vgs (MOSFET y JFET) ajustando convenientemente la tensión de salida de los DACs y la resistencia de los potenciómetros. Una vez fijada esta referencia se aumenta escalonadamente la diferencia de tensión Vce o Vds, aumentando o disminuyendo la tensión de los DACs según el caso. El problema que se encuentra a medida que se van aumentando estas diferencias de tensión y por tanto las corrientes asociadas a ellas, es que el parámetro Z, es decir la Ibase o la Vgs se modifica. Esto se debe a la naturaleza del diseño y los principios físicos de cada tipo de transistor. Es estrictamente necesario que para que las gráficas sean fieles a la realidad este tercer parámetro se mantenga constante a lo largo de todas las medidas de dicha curva. Es por eso que en cada variación de uno de los DACs a medida que se aumenta la diferencia Vds o Vce, se comprueba que este parámetro (Vgs o Ibase) se sigue manteniendo constante y en caso se haya modificado se realice un ajuste en el DAC que esté conectado a la base o puerta del transistor hasta que se consiga que este parámetro vuelva a ser igual que el inicial. Una vez que se ha obtenido una gráfica completa, se modifica la referencia de Ibase o Vgs y siguiendo el mismo patrón se realiza otra gráfica.

4.7.3 Servidor web

Cuando se detecta que tipo de componente es y como están colocados sus terminales esta información se envía a la página web respondiendo a las peticiones realizadas por los clientes. Cada vez que se obtienen todos los datos de una de las curvas, estos datos se encapsulan en una variable de tipo *String* en formato JSON y el servidor web las envía a los clientes para que gracias a la librería anteriormente comentada puedan visualizar las gráficas en sus navegadores.

El servidor web además de recibir los datos X e Y para formar el grafo recibe por parte del usuario el parámetro de referencia Z de cada una de las gráficas que se ha obtenido, esto permite que se muestre una leyenda con la Vgs o Ibase que corresponde a cada gráfica según el color. También con cada gráfica se envían el máximo y el mínimo

de cada eje para que el fondo de escala del eje X e Y de la gráfica se escalen según sea el transistor para analizar

El servidor como ya se ha comentado permite empezar la medida con un botón desde el navegador. Además, en el programa se ha implementado un indicador que nos muestra si el producto está listo para comenzar una medida o se encuentra midiendo. Esto ayuda al usuario a saber cuándo ha terminado la toma de valores, de esta manera se evita que el usuario retire el componente pensando que la medida ha terminado. A media que se toman las gráficas se ha diseñado una rutina que permite obtener unos valores aproximados de la ganancia o la transconductancia del componente. Por último, el programa incorpora una funcionalidad que muestra una imagen que corresponde al símbolo del componente analizado.

5 Herramientas para la evaluación del producto

5.1 Monitor serie

Para ir comprobando en cada etapa el funcionamiento del equipo y buscar fuentes de posibles errores principalmente se ha empleado el monitor serie que incorpora Visual Studio Code para Arduino. Gracias a esta funcionalidad podemos comprobar el valor de las variables de programa dentro del producto desde la pantalla del ordenador (figura 5.1). Esto ha sido empleado para comprobar valores de tensiones y corrientes que suponíamos correctas cuando algo no funcionaba o para comprobar que el flujo del programa era el esperado.

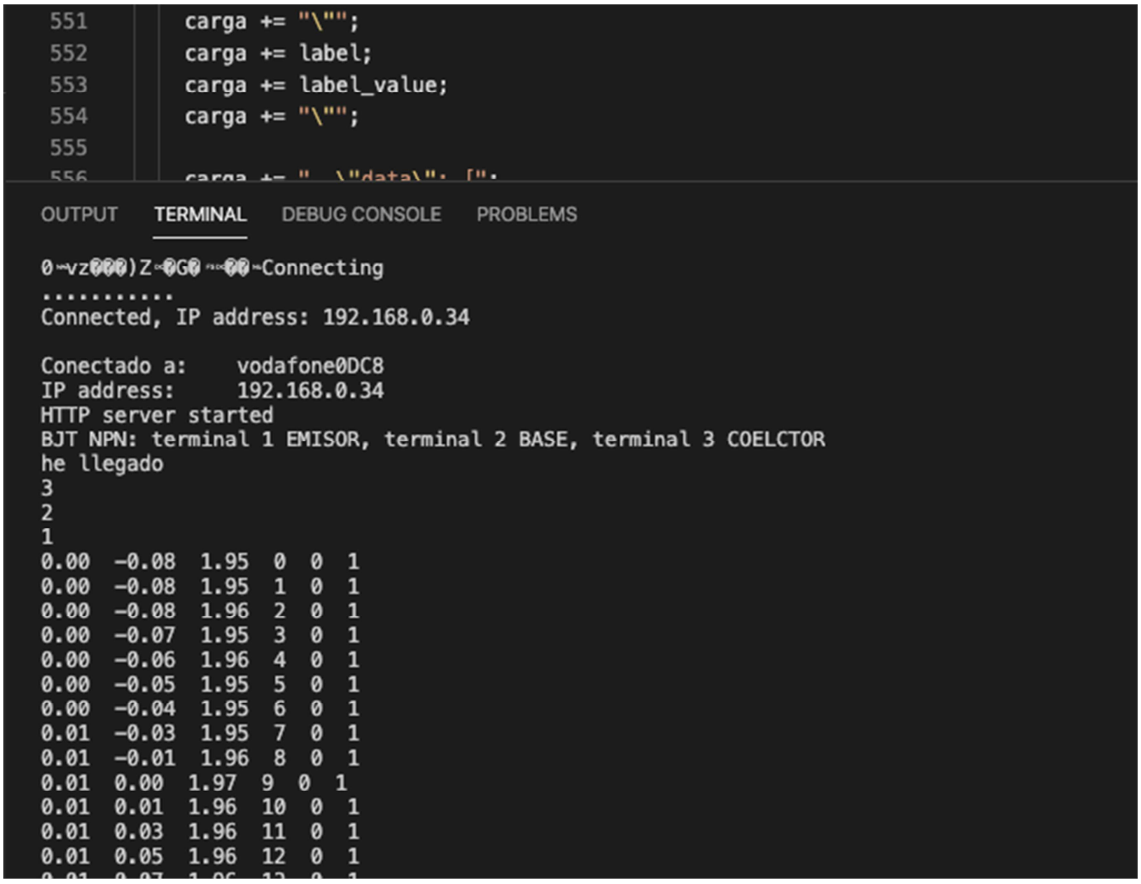


Figura 5.1: Monitor serie mostrando datos del flujo del programa.

Empleando el monitor serie imprimiendo mensajes sobre el componente detectado también se ha conseguido encontrar los casos en los que un transistor colocado en una

determinada posición en los 3 terminales de medida no conseguía ser detectado o era detectado como otro tipo de componente.

5.2 Instrumentos electrónicos

El multímetro ha sido de gran utilidad a la hora de comprobar que las medidas realizadas por el ADC eran correctas, comprobar el funcionamiento de los DACs y comprobar la resistencia real que existe entre los extremos de los potenciómetros en sus distintas posiciones. Es importante destacar que una herramienta como el multímetro puede ser de gran ayuda y podría considerarse imprescindible, no obstante, es muy importante tener cuidado a la hora de utilizarlo. Cuando se van a realizar medidas de intensidad hay que recordar que la medida no se realiza de la misma manera que una medida de tensión. Las medidas de tensión son las más habituales y puede darse el caso en el que vayamos a realizar una medida de corriente y los terminales del polímetro los coloquemos de la misma manera que si hiciésemos una medida de tensión en vez de cortar el circuito y hacer que la corriente circule por el polímetro. En caso de que se cometa este error se puede producir un cortocircuito que puede tener consecuencias terribles para los componentes hardware.

Otra herramienta utilizada ha sido un osciloscopio digital del laboratorio que permitió detectar el error de conexiones entre el puerto serie del microprocesador y el convertidos de USB a UART.

Las medidas tanto con el polímetro como con el osciloscopio se pudrieron realizar de forma sencilla gracias a que la PCB contaba con puntos de test serigrafiados con el nombre de la señal que se puede medir en ellos. De esta manera se ahorró tiempo en la medida a la hora de encontrar las señales y lo más importante se evitó el riesgo de producir cortocircuitos cuando se toca más de una patilla de los componentes pequeños con uno de los terminales del polímetro.

5.3 GNUPLOT

Si se quieren comprobar valores puntuales y no una gran cantidad de ellos estos métodos anteriores son muy útiles. En el caso de la toma de valores para las gráficas los datos numéricos que se recogen son muy numerosos, esto dificulta saber si las gráficas

obtenidas son correctas cuando no se tiene diseñado el servidor Web todavía y no se pueden visualizar ahí.

Para solucionar el problema de visualizar los valores obtenidos de las gráficas se optó por emplean GNUPlot, un programa que nos permite generar gráficas en 2 o 3 dimensiones con numerosas funcionalidades en el ajuste de datos. Este programa gratuito escrito en C y compatible con la mayoría de los sistemas operativos, como su propio nombre indica pertenece a la colección de programas del proyecto GNU apoyado por la *Free Software Foundation*. Empleando este programa y su línea de comandos ha sido posible mostrar los valores obtenidos de manera gráfica. Para ello en el programa principal del equipo se imprimían cada punto definido por X (Vds o Vce), Y (Id o Ic) y Z (Vgs o Ibase) separados por saltos de línea y las coordenadas de cada uno (X,Y,Z) se paradas por espacios. Recogiendo estos datos del monitor serie y pegándolos en un archivo de texto, gnuplot mediante su línea de comandos permite visualizar la gráfica en 2 dimensiones (X, Y) o en 3 dimensiones (X, Y, Z). A continuación, se muestran diferentes figuras en 2 y 3 dimensiones obtenidas mediante GNUPlot.

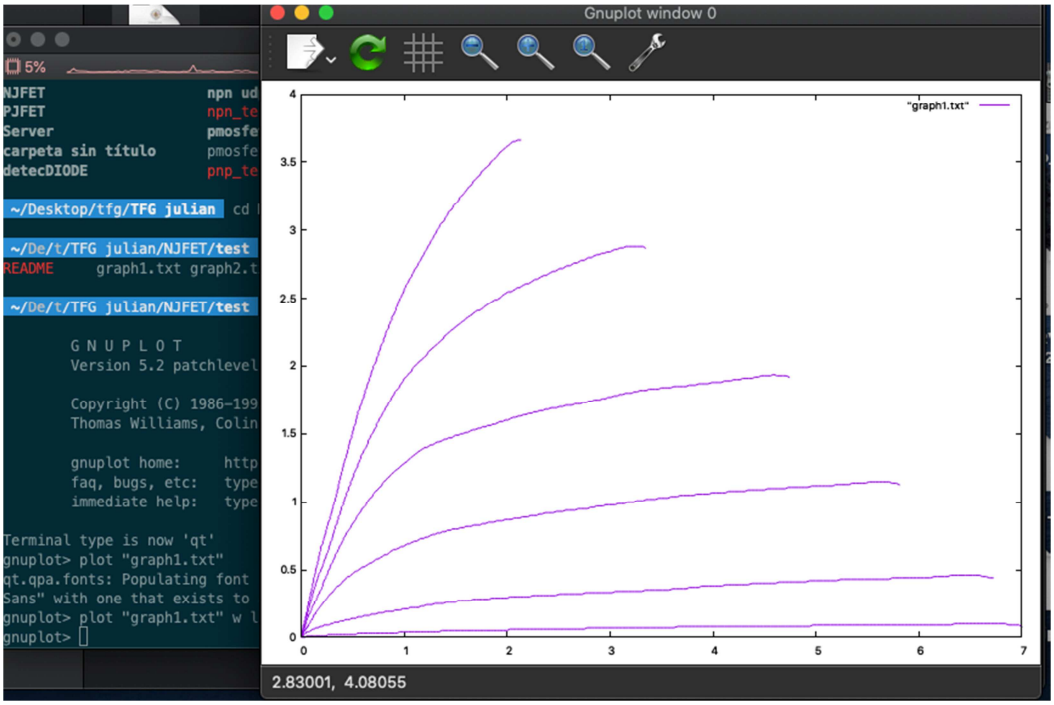


Figura 5.2: Gráfica realizada de los datos obtenidos de un JFET tipo N mediante GNUPlot.

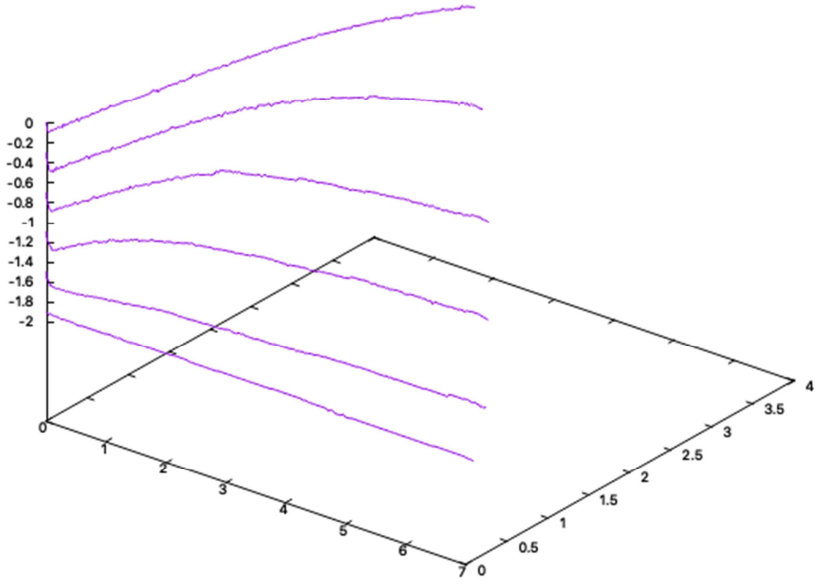


Figura 5.3: Gráfica en 3 dimensiones del JFET tipo N donde se muestra que la Vgs permanece constante en cada grafo.

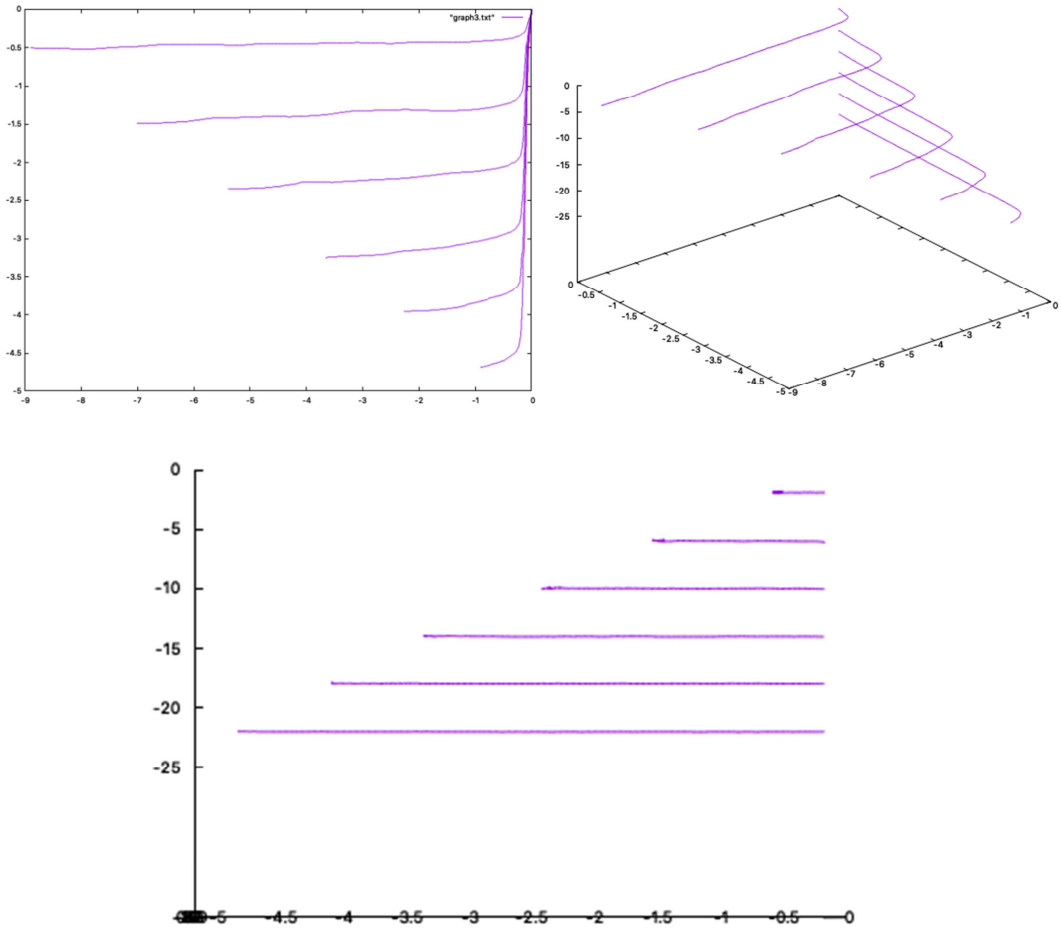


Figura 5.4: Gráficas de un transistor BJT tipo P, en la primera imagen se muestra el resultado final y en las otras dos imágenes en 3 dimensiones se observa como la tercera variable Ibase permanece constante.

Gracias a este programa sin la necesidad del servidor web se ha observado que existía gran cantidad de ruido. En un primer lugar se pensó que este ruido podría ser causado por alguna interferencia electromagnética producida fuera del equipo. Se acabó llegando a la conclusión de que este ruido al ser periódico (y no deberse a la red eléctrica) era producido por el microprocesador, el cual afectaba a la medida del ADC. Se cree que este ruido puede deberse a que el voltaje de referencia interno usado por el ADC del microprocesador no sea estable. Cuando el ESP-12F pasa de recibir a transmitir información ya que el consumo de corriente cambia de 50mA a 170mA.

Por último, esta herramienta nos ha servido para comprobar visualmente que la variable Z en cada gráfica permanecía constante a lo largo de ella dentro de un pequeño margen de histéresis como se ha podido observar en las figuras anteriores.

5.4 Firefox

Una vez programado el servidor web se han depurado sus errores y comprobado su funcionamiento mediante este programa. La selección de este navegador web gratuito no ha sido al azar, ha sido escogido porque implementa varias herramientas para desarrolladores web muy útiles.

Con estas herramientas se puede inspeccionar el código HTML que hemos programado, ver si existen errores y modificarlo en tiempo real para comprobar cambios mucho más rápido. Dispone de una consola (figura 5.5) que nos permite ver las funciones que se están ejecutando, y comprobar el valor de ciertas variables y listas. Por último, entre otras muchas más funcionalidades la de red permite ver todas las respuestas del servidor a las peticiones del navegador, el tipo de respuesta, el retardo entre la petición y por último el contenido de las respuestas del servidor.

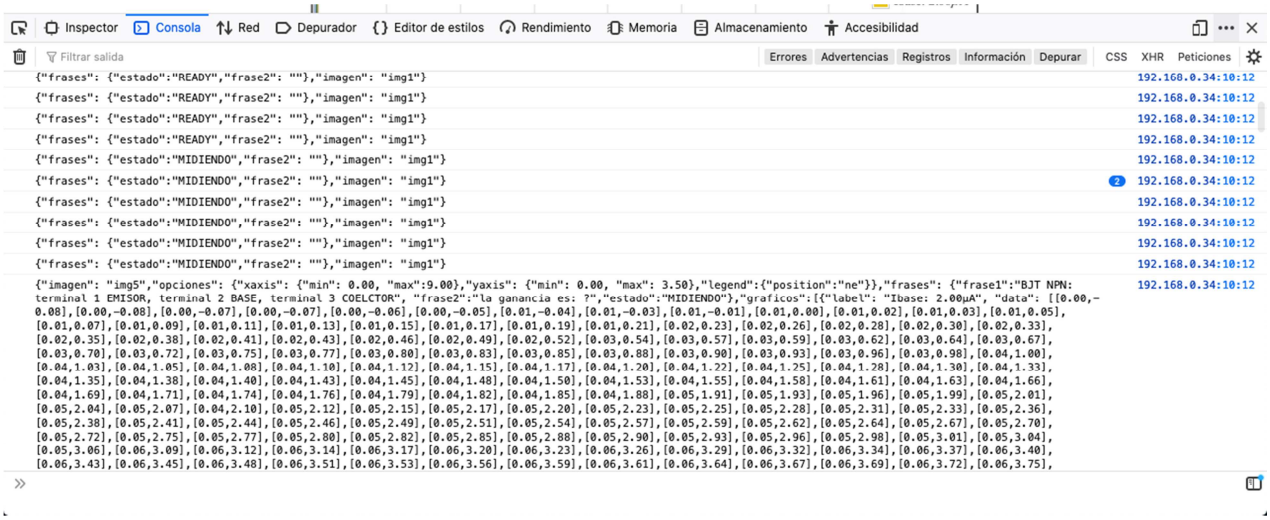


Figura 5.5: Herramienta y consola de Firefox para desarrolladores Web.

6 Resultados

6.1 Mecanizado del producto

Los productos que se comercializan habitualmente disponen de un recubrimiento exterior para proteger la electrónica. De esta manera se protegen los componentes internos de los usuarios y de distintos agentes externos. Esto permite proteger frente a cortocircuitos, golpes y otro tipo de situaciones que puedan dañar la electrónica interior.

Con este objetivo se seleccionó una caja rectangular de plástico negro y opaco por la cara inferior y transparente por la cara superior de unas dimensiones ligeramente superiores a la PCB. Se le realizaron 4 orificios por medio de un taladro en la cara inferior los cuales se han empleado para sujetar la PCB mediante 4 tornillos. Para evitar que la batería estuviera suelta y en caso de que se quisiera retirar la carcasa de plástico esta permaneciese unida a la PCB, se opto por pegar la batería a la PCB empleando un trozo de cinta de doble cara (figura 6.1).

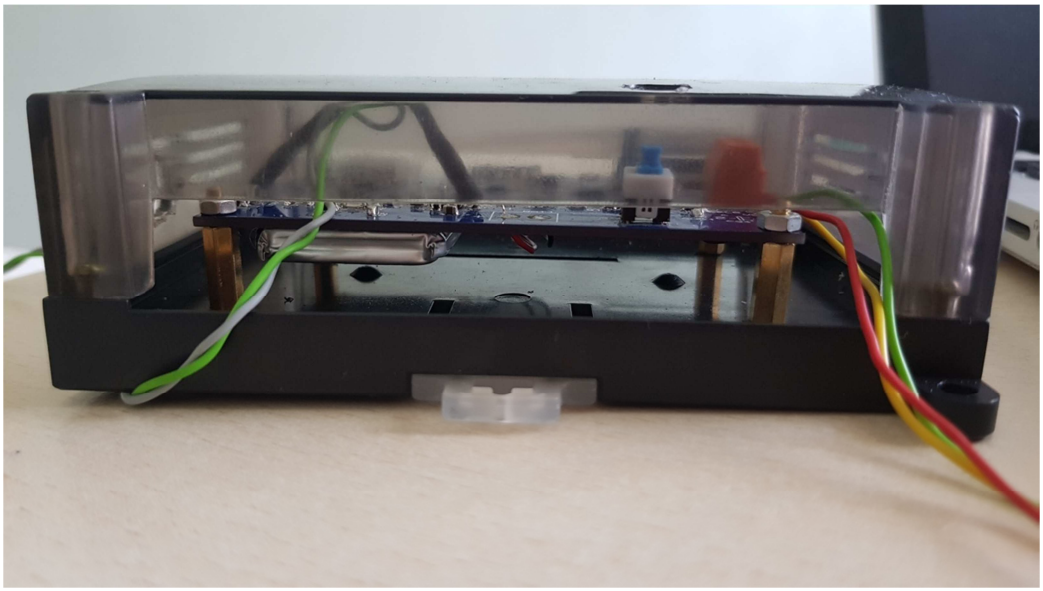


Figura 6.1: PCB anclada a la carcasa inferior y batería pegada a la PCB.

La carcasa superior dispone una abertura en dos de sus lados lo que permite reprogramar el equipo si se desea y sacar los 3 terminales que permiten conectar los componentes a medir. Para los botones se taladró un orificio para cada uno empleando un taladro de manera que se permita su pulsación empleando por ejemplo un bolígrafo (figura 6.2).

Figura 6.2: Orificios que permiten la pulsación de los botones.

Por último, para realizar una abertura que permitiera la carga de la batería se llevo acabo un último orificio empleando un soldador que permitiera quemar el plástico ya que en esa sección no era recomendable utilizar un taladro. El resultado final del producto se observa a continuación en la figura 6.3.

Figura 6.3: Imágenes del producto final.

6.2 Verificación del producto

Tras haber completado el desarrollo de los componentes hardware y software en este apartado se procede a detallar el funcionamiento final obtenido. Se sigue un típico proceso de utilización de este equipo como prueba.

En primer lugar cuando encendemos el equipo encontramos que los leds están apagados, hasta que el equipo no sea capaz de conectarse a la red Wifi el diodo led D2 no se encenderá. Esto se puede corroborar si conectamos la interfaz serie a nuestra computadora para poder emplear el monitor serie anteriormente comentado.

Una vez conectado accederemos con uno de nuestros navegadores a la dirección IP del equipo (192.168.0.34) en el caso en el que estemos en la misma red WiFi, si no es el caso podremos acceder por medio del redireccionamiento de puertos que también se ha comentado. Una vez nos conectemos con nuestro navegador al servidor web observaremos la página web programada (figura 6.4), que nos mostrará que está lista (*ready*) para comenzar la medida y que actualmente no se ha detectado ningún componente.

Figura 6.4: Pagina web mostrada al conectarnos al servidor del producto.

Una vez que hemos colocado el componente a analizar en los 3 terminales se puede comenzar la medida. Para esta tarea como ya se ha comentado se puede emplear tanto el botón físico situado sobre la PCB o el botón disponible en la página web. Una vez pulsado uno de los botones comienza la medida y la detección del componente. Podremos saber que la medida ha comenzado porque el led D2 se apaga a la vez que el D1 se enciende y en el servidor Web el indicador de READY de fondo verde pasará a indicarnos que está midiendo además de cambiar su fondo a azul (figura 6.5).

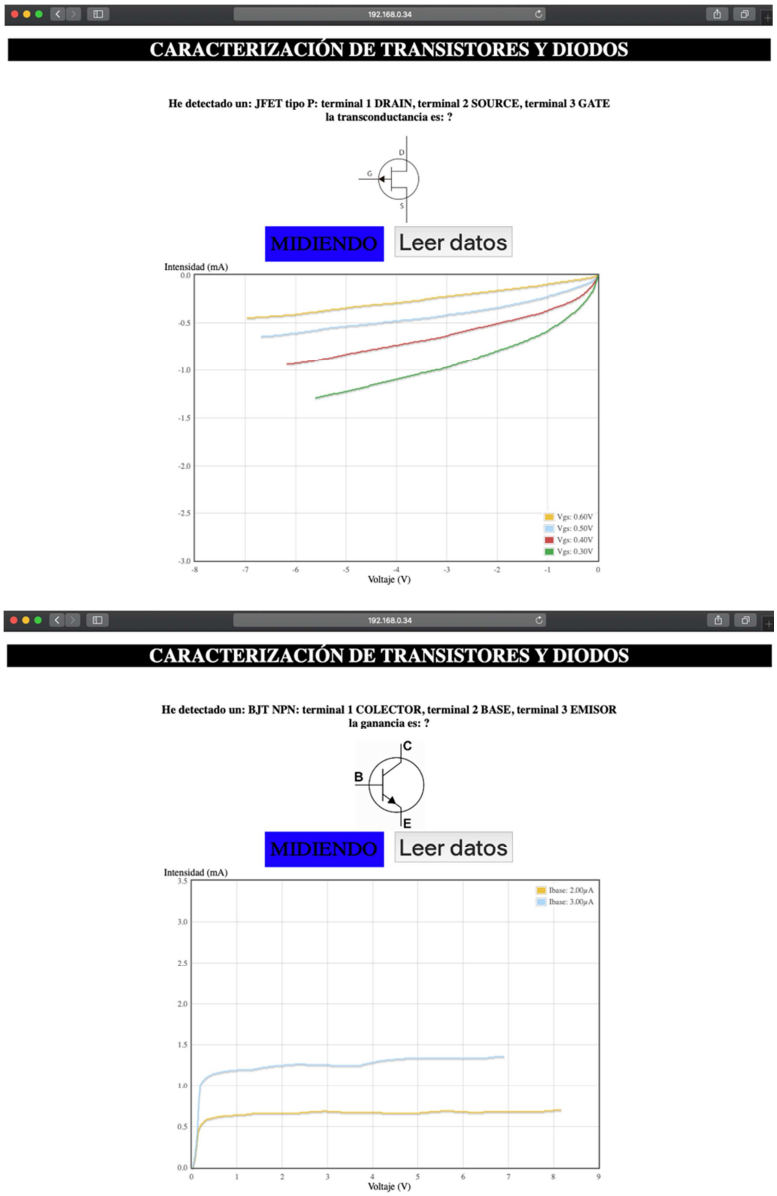


Figura 6.5: Pagina Web mientras el equipo se encuentra midiendo, 1ºJFET tipo P y 2º BJT NPN.

Este indicador puede tardar un máximo de un segundo en actualizarse, ya que por cómo está programado el navegador mandara peticiones al servidor web cada segundo. La elección de este tiempo de refresco de la página Web se debe a que si se disminuía este tiempo el ESP-12F no era capaz de contestar a todas las peticiones y muchas de estas se perdían o se recibían con mucho retraso como se puede observar la figura 6.6.

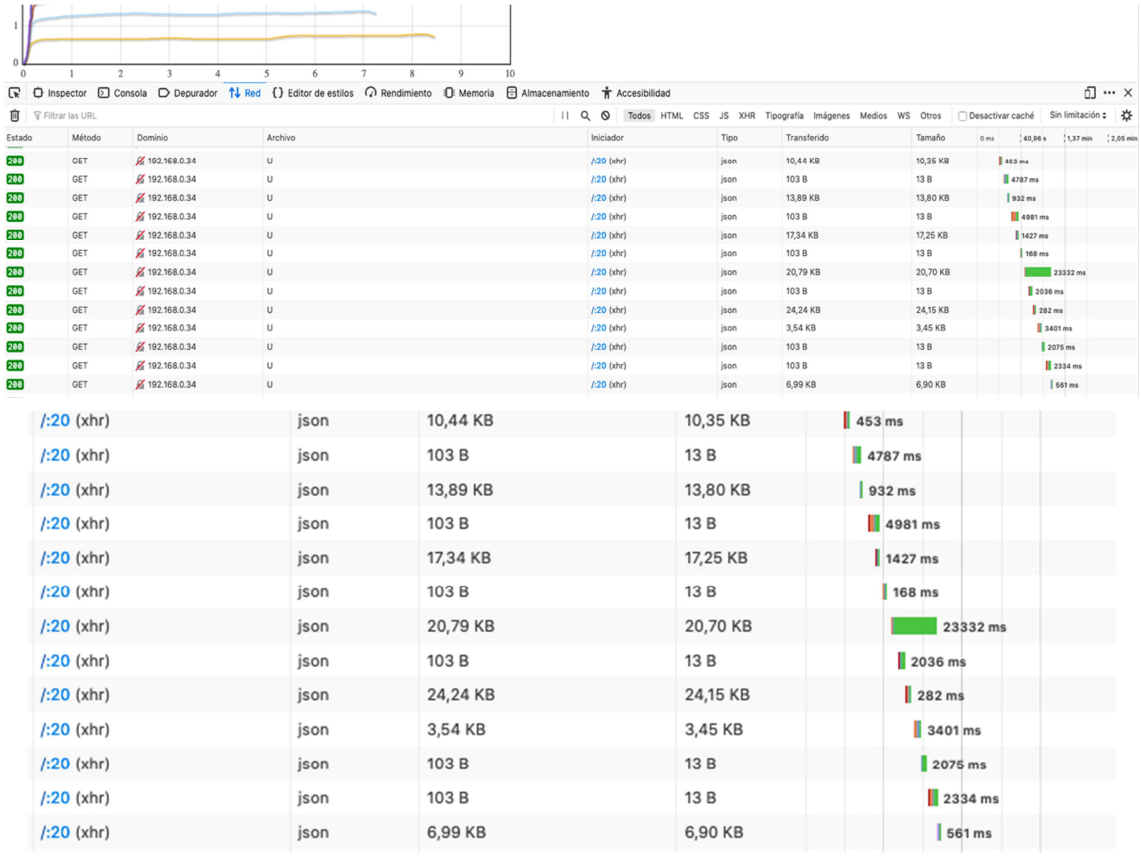


Figura 6.6: Herramienta de red de Firefox donde se observa el retardo de algunas peticiones.

La detección del componente es muy veloz, en un tiempo inferior a un segundo, esto se podría observar mediante el monitor serie, pero en caso de la página web no se mostrará hasta que se envíe la primera gráfica. En caso de no ser capaz de detectar el componente, el servidor nos mostrará un mensaje de error que muestra que no ha sido capaz de detectarlo.

Una vez el producto ha conseguido detectar el componente correctamente comenzará a obtener los puntos X, Y y Z para poder formar las gráficas en los navegadores web de los usuarios. Estos valores también se muestran por el terminal serie para que en caso de que el equipo esté conectado a un ordenador se puedan comprobar estos datos y extraerlos con facilidad. Los datos también se pueden visualizar desde el navegador web utilizando la consola de desarrolladores que incorpora. No obstante, los datos extraídos del monitor serie se pueden introducir en el GNUPlot en caso de que se desee, esto no es posible con los datos del navegador ya que están en formato JSON.

En función del tipo de componente se obtendrán más o menos curvas o gráficas, cada vez que se toman todos los valores de una de estas curvas estos datos se envían

respondiendo a las peticiones del navegador del usuario. El usuario desde el navegador podrá ver cómo van apareciendo las gráficas una a una en un intervalo aproximado de 12 segundos cada una. El proceso habrá terminado cuando en la página el indicador haya pasado de MIDIENDO a READY, esto nos indicará que se han obtenido todas las gráficas y que se puede retirar el componente, colocar otro o proceder a una nueva medida. A continuación, se muestra en las figuras 6.7, 6.8 y 6.9 el resultado obtenido tras completar la medida de alguno de los componentes.

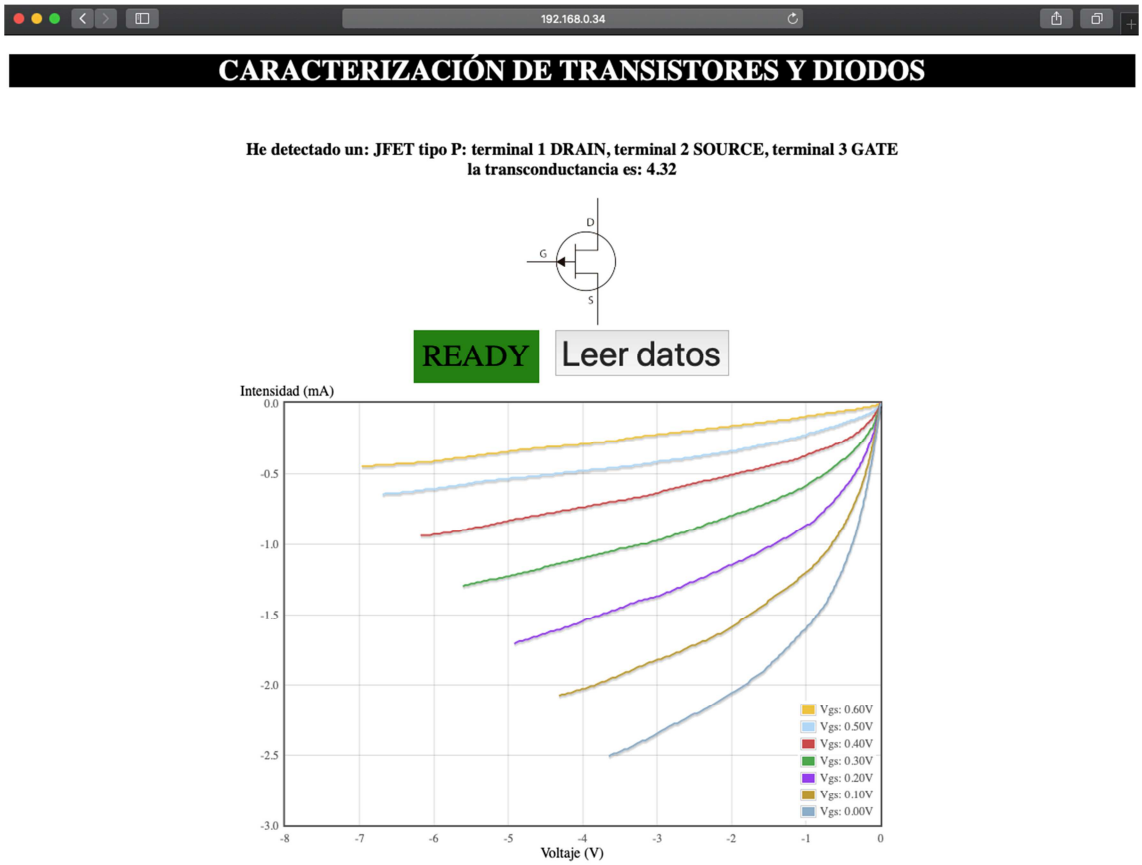


Figura 6.7: Medida completada de un JFET tipo P.

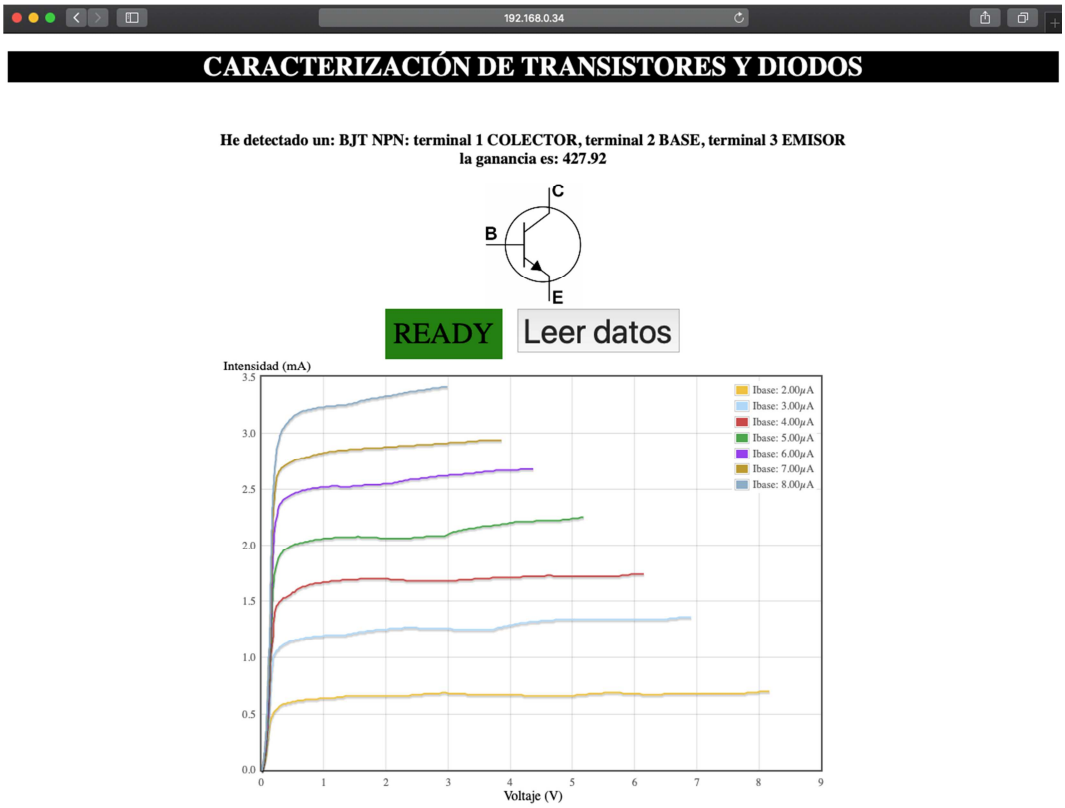


Figura 6.8: Medida completada de un BJT NPN.

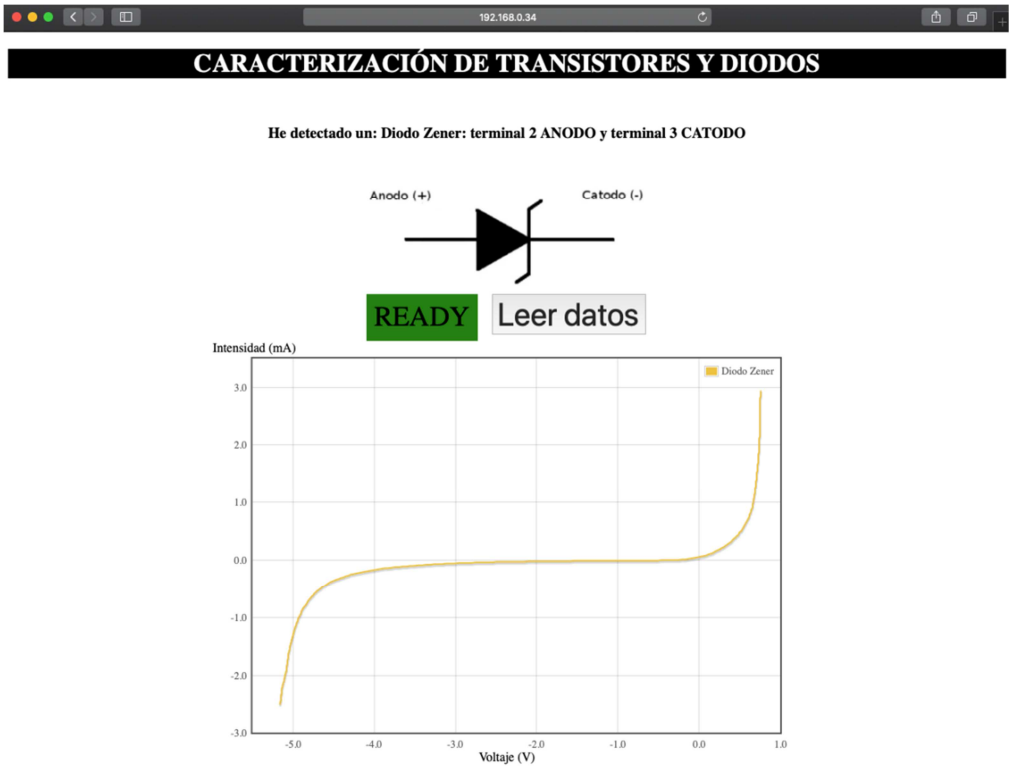


Figura 6.9: Caracterización completada de un diodo Zener.

6.3 Funcionamiento final

Finalmente, tras analizar diferentes componentes y visualizar los datos obtenidos de estos con un navegador web desde un ordenador o un Smartphone se ha comprobado el funcionamiento satisfactorio en la caracterización de lo componentes propuestos.

El análisis no es especialmente rápido, pero la calidad de las gráficas es buena y la interfaz del servidor es simple y muy funcional. Los leds del equipo ayudan a ver si el equipo se encuentra midiendo o está esperado a que le indiquemos que comience una medida, como el indicador de la página web.

Desde que el equipo arranca es necesario esperar unos segundos hasta que el equipo se conecta a la red WiFi. Una vez conectado es bastante robusto y no es habitual que se cuelgue salvo que se retire el componente mientras se realiza la medida.

6.4 Futuras iteraciones

Como se ha podido observar el equipo desarrollado tiene una serie de carencias debidas al hardware y al software por las que más que como un producto final se debería considerar como un prototipo. Habría que realizar nuevas iteraciones de prototipo con mejoras en el hardware y en el software que permitieran tener un equipo perfectamente funcional que se pudiera comercializar.

A continuación, en este apartado se va a desarrollar punto a punto las posibles mejoras que se consideran de mayor importancia y que permitirían solucionar los defectos del prototipo actual:

1. **Solucionar problemas hardware**: como se ha comentado ya a lo largo de esta memoria, durante la comprobación del hardware se encontraron 2 problemas. En primer lugar, rutar correctamente las líneas RXD y TXD de la interfaz serie del micro, y por último modificar la huella de los 3 transistores MMBT3904 ya que esta era errónea.
2. **Añadir un ADC externo**: puesto que el ADC del micro se ha comprobado que introduce gran cantidad de ruido sería conveniente incluir un ADC externo de mayor calidad que nos facilitara unas mediciones de mayor precisión y menor

incertidumbre. Sería aconsejable que este componente contase con una interfaz I2C para comunicarse con el micro

3. **Medida carga de la batería**: sería muy interesante contar con la posibilidad de medir el estado de la batería, de esta manera el usuario podría ver desde la página web el porcentaje restante y saber si es necesario cargar el equipo o saber cuando este está cargado al máximo de su capacidad. Para esto se podría incluir la tensión de salida de la batería como una entrada del multiplexor para que seleccionando ese canal el ADC pudiera medir la tensión de la batería de litio y de esta manera conocer su porcentaje de carga.
4. **Sistema de calibración del potenciómetro**: sería conveniente introducir un sistema que nos permite conocer el valor exacto de las 256 posiciones del potenciómetro para obtener medidas más precisas. Uno de los métodos que se podría emplear sería colocar en los terminales del equipo una resistencia de precisión de un valor conocido. De esta manera realizando medida se podría determinar la resistencia del potenciómetro junto con la de sus interruptores de manera automática y sencilla.
5. **Mejorar software de gráficas**: Sería muy útil incorporar en el servidor web la opción de que el usuario pueda escoger los rangos de medida sobre todo en lo referente al parámetro de referencia Z (Ibase o Vgs) en cada una de las gráficas.
6. **Mejora del servidor Web**: por último, se podría modificar el software de manera que más de un cliente a la vez pudiera estar conectado al servidor Web, ya que tal y como está programado en esta versión las gráficas solo se envían a uno de los clientes.

7 Documentación

7.1 Bibliografía

[1] *Elektor Article: Transistor Curve Tracer*. Url: https://www.elektormagazine.com/magazine/elektor-200902/18980/ https://archive.org/stream/ElektorMagazine/Elektornonlinear.ir2009-02#page/n25/mode/2up

[2] *ESP-12F WiFi Module. Verion 1.0*. Url: https://www.elecrow.com/download/ESP-12F.pdf

[3] *Microchip. MCP4728, 12-Bit, Quad Digital-to-Analog Converter*. Url: http://ww1.microchip.com/downloads/en/devicedoc/22187e.pdf

[4] *Analog Devices. Quad, 15 V, 256-Position, Digital Potentiometer.* Url: https://www.analog.com/media/en/technical-documentation/data-sheets/AD5263.pdf

[5] *Maxim Integrated. Low-Noise, Precision, Quad Op Amps.* Url: https://datasheets.maximintegrated.com/en/ds/MAX44241-MAX44246.pdf

[6] *Texas Instruments. CD4051 CMOS Single 8-Channel Analog Multiplexer.* Url: https://www.ti.com/lit/ds/symlink/cd4051b.pdf?HQS=TI-null-null-mousermode-df-pf-null-wwe&ts=1593622304758

[7] *STMicroelectronics. M24C64 64-Kbit serial I2C Bus EEPROM.* Url: https://www.st.com/resource/en/datasheet/m24c64-r.pdf

[8] *NXP. Application Note. Designing for Board Level Electromagnetic Compatibility. 10/2005*. Url: https://www.st.com/resource/en/datasheet/m24c64-r.pdf

[9] *PCBWay. PCB Instant Qoute. Estándar PCB*. Url: https://www.pcbway.com/orderonline.aspx

[10] *LM2596 DC-DC Adjustable PSU Mosule*. Url:
http://tpelectronic.ir/datasheets/20150123144301750.pdf

[11] *PromeTec. ESP8266 Para Arduino IDE*. Url:
https://www.prometec.net/esp8266-pluggin-arduino-ide/

[12] *PlataformIO. Professional collaborative platform embedded development*. Url:
https://platformio.org

[13] *MDN web docs. Structuring the web with HTML.* Url:
https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML

[14] *Flot. Attractive JavaScript plotting for Jquery.* Url:
https://www.flotcharts.org
https://github.com/flot/flot

7.2 Apéndice: código del ESP-12F

```
////////////////////////////////////////////////////////////////////////////
///////                TFG    Julian Arambarri  Garcia    Julio 2020
////////////////////////////////////////////////////////////////////////////

#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include "jquery.h"

#include <Ticker.h>
#include <Wire.h> //libreria I2C
#include <math.h>
ESP8266WebServer server(80);

//Ticker visu;
#define TERMINAL_A1 0
#define TERMINAL_A2 1
#define TERMINAL_B1 2
#define TERMINAL_B2 3
#define TERMINAL_C1 4
#define TERMINAL_C2 5

//////////////////////////DAC//////////////////////////7
#define direcDAC 0b1100000 //direccion DAC es programable 96 en decimal
#define comando 0b010      //c2 c1 c0
#define funcion 0b00       //w1 w0
#define trama1DAC (comando << 5) | (funcion << 3)
#define vref 0b1 // seleccion de voltaje de referencia, 1-> 2.048; 0->Vdd
#define pd 0     // esto se puede quitar impedancia de salida
#define gain 0   // ganancia x1, se se pone a 1-> x2
#define trama2DAC (vref << 7) | (pd << 5)

//////////////////////POT//////////////////////////////////
#define direcPOT 0b0101100

//////////////////DAC//////////////////////////
byte dacSelect[4] = {0b00, 0b01, 0b10, 0b11}; //seleccion de dac
int valor = 0;                          //pq no funciona con un char char ya que son 12 bits
byte iDACselect = 0;

//////////////////POT//////////////////////////
```

```
byte channelPOT[4] = {0b00, 0b01, 0b10, 0b11};
byte iPOTselect = 0;
byte datPOTresis = 0;

////////////////////////////////////////////
//////////////////////////////////////////     FUNCIONES
void DACset(byte iDAC, int valorDAC);
float POTvalor(int POTres);
float POTresist(int D);
void DACset(byte iDAC, int valorDAC);
void POTset(byte iPOT, byte datPOT);
float mideTerminal(byte canal);
float measureI(byte canal1, byte canal2, float POTresist);

///////////////detectar
void detectar_terminal();
void detectar_diodo();
void detectar_NPNbjt();
void detectar_PNPbjt();
void detectar_MOSFET();
void detectar_NJFET();
void detectar_PJFET();

void terminals();

void flip();
void ver_datos();

//////////graficas
void graficas();
void diode_graph();
void nmos_graph();
void pmos_graph();
void nbjt_graph();
void pbjt_graph();
void njfet_graph();
void pjfet_graph();
void diode_zener_graph();

//////// internet
void init_internet();

////////server
void init_server();
void ActualizaGrafico();
void HandleRoot();
void HandleNotFound();
void IniciaProceso();
void resetbuffer();
```

```
//////////////////ADC//////////////////////////la medida tarda un timepo de 103us  (106-3)
int analogInPin = A0; // ESP8266 Analog Pin ADC0 = A0

////////////////////////////////////////////
float VA1 = 0;
float VA2 = 0;
float VB1 = 0;
float VB2 = 0;
float VC1 = 0;
float VC2 = 0;
float Ibase = 0.0;
float Idrenador = 0.0;
float Iemisor = 0.0;
float Vce = 0;
float Vrc = 0.0;
float average = 0;
int Vg = 0;
float Vd = 0;
float Vds = 0;
float Vgset = 0;
float Vgs = 0;
int media = 0; //media de los valores finales leidos promediar 10
float VceMedia = 0.0;
float IbaseMedia = 0.0;
float IcolectorMedia = 0.0;
float IbSet = 0;
float V = 0;
float Vgate = 0;
float VgsMedia = 0;
float VdsMedia = 0;
float IdrenadorMedia = 0;
float Vdif1;
float Vdif2;
float Vdif3;
float Icolector;

float filtro = 0.09;
float Vdssal = 0;
float Vgssal = 0;
float Idrenadorsal = 0;
long countok = 0;

int Vcomp1 = 0;
int Vcomp2 = 0;
int Vcomp3 = 0;

/////detectar
int i = 0;
int Vb = 2000;
float Ia = 0;
float Ib = 0;
```

```
float Ic = 0;

//////////////////////////// asignación de transitor y de terminales
boolean NBJT = false;
boolean PBJT = false;
boolean NMOS = false;
boolean PMOS = false;
boolean NJFET = false;
boolean PJFET = false;
boolean DIODE = false;
boolean DIODE_ZENER = false;

//asignación de terminales
//bipolares
int select_colector;
int colector1;
int colector2;
int select_base;
int base1;
int base2;
int select_emisor;
int emisor1;
int emisor2;
//mosfet_jfet
int select_drain;
int drain1;
int drain2;
int select_gate;
int gate1;
int gate2;
int select_source;
int source1;
int source2;
//diodo

int anodo1 = 0;
int anodo2 = 0;
int select_anodo = 0;
int catodo1 = 0;
int catodo2 = 0;
int select_catodo = 0;

int ndetecciones = 0;
//BJT
int base = 0;
int colector = 0;
int emisor = 0;
//MOS y JFET
int gate = 0;
int drain = 0;
int source = 0;
```

```
//diodo
int anodo = 0;
int catodo = 0;

////////////////////////// graficas
//general
int Vbas = 0;
//mostrar graph
boolean flag = false;
boolean charge = true;
int chargeI = 0;
boolean endGraph = false;
String carga = "";
String carga2 = "";
int numero = 0;
String label = "";
String label_value = "";

//mosfet
float Vgs_set = 0;

//jfet
int Vsource_min = 0;

///////////ver datos
float Icolectorsal;
float Vcesal;
float Ibasesal;
float valorx[400];
float valory[400];

float x = 0;
float y = 0;
float z = 0;
float x_sal = 0;
float y_sal = 0;
float z_sal = 0;
String estado[2] = {"MIDIENDO", "READY"};
bool ready = true;

//grafica
String imagen = "";
float x_min = 0;
float x_max = 0;
float y_min = 0;
float y_max = 0;
String position = "";
String deteccion = "";
String gain_trans = "";

boolean start = false;
```

```
/// ganancia hfe y transconductancia

float hfe; ///  ganancia y tranconductancia
float gm;
float vgs_anterior;
float id_anterior;
float vgs_final;
float id_final;

float ib_final;
float ic_final;


////////////////////////////////////////////////////////////////////////////
///////                Setup  inicializamos
////////////////////////////////////////////////////////////////////////////

void setup()
{
  Serial.begin(9600);
  delay(10);

  init_internet();
  init_server();

  Wire.begin(2, 14); //Wire.begin(int sda, int scl)450KHz

  // flip the pin every 0.3s

  //
  //visu.attach(0.1, server.handleClient() );
  pinMode(5, OUTPUT);  //muxA
  pinMode(4, OUTPUT);  //muxB
  pinMode(12, OUTPUT); //muxc
  pinMode(16, OUTPUT); //leds
  //pinMode(15, INPUT);  //button
}

////////////////////////////////////////////////////////////////////////////
///////                loop  lazo basico
////////////////////////////////////////////////////////////////////////////

void loop()
{

  digitalWrite(16, HIGH);
  if (digitalRead(15) == HIGH)
  {

    while (digitalRead(15) == HIGH)
    {
```

```
    server.handleClient();
    delay(100);
  }
  ready = false;
  server.handleClient();

  digitalWrite(16, LOW);
  detectar_terminal();
  graficas();
  ready = true;
  start = false;
 }

 if (start)
 {

  ready = false;
  server.handleClient();

  digitalWrite(16, LOW);
  detectar_terminal();
  graficas();
  ready = true;
  start = false;
 }
 server.handleClient();
}

////////////////////////////////////////////////////////////////////////////
///////              Init internet  inicializamos internet
////////////////////////////////////////////////////////////////////////////
void init_internet()
{

 WiFi.mode(WIFI_STA);
 WiFi.begin("vodafone0DC8", "HAMVF92X8V2FY8");
 Serial.println("Connecting");

 while (WiFi.status() != WL_CONNECTED)
 {
  delay(500);
  Serial.print(".");
 }
 Serial.println();
 Serial.print("Connected, IP address: ");
 Serial.println(WiFi.localIP());

 // Mostrar mensaje de exito y dirección IP asignada
 Serial.println();
 Serial.print("Conectado a:\t");
 Serial.println(WiFi.SSID());
```

```
  Serial.print("IP address:\t");
  Serial.println(WiFi.localIP());

  return;
}

/////////////////////////////////////////////////////////////////////////////
///////               Init server  inicializamos server
/////////////////////////////////////////////////////////////////////////////

void init_server()
{

  server.on("/", HandleRoot);
  server.on("/U", ActualizaGrafico);
  server.on("/BOTON", IniciaProceso);
  server.on("/JQ.js", []() {
    server.send(200, "text/javascript", JQ, sizeof(JQ) - 1);
  });
  // Ruteo para URI desconocida
  server.onNotFound(HandleNotFound);
  // Iniciar servidor

  server.begin();
  Serial.println("HTTP server started");

  return;
}
/////////////////////////////////////////////////////////////////////////////
///////               Handle Root
/////////////////////////////////////////////////////////////////////////////
void HandleRoot()
{
  Serial.println("staaaaaaar1");
  String cuerpo = "<!DOCTYPE html><html>";
  cuerpo += "<head><meta charset='UTF-8'>";
  cuerpo += "</head>";
  cuerpo += "<body onload=\"update();setInterval('update()',1000);\">";
  cuerpo += "<h1 style='background-color:black;width:100%; text-align: center;
color:white;'>CARACTERIZACIÓN DE TRANSISTORES Y DIODOS</h1><br/>";
  cuerpo += "<h3 style='text-align: center;'>He detectado un:  <div style='display:inline-block'
id='frase1'></div><br/><div style='display:inline-block' id='frase2'></div></h3>";
  cuerpo += "<div style='text-align:center'>";
  cuerpo += "<div id='imagenes' style='height:150px;'>";
  cuerpo += "<img style='display:none;' src='https://1.bp.blogspot.com/-sF0u8XxSrTo/Xu3wj3D-
_rI/AAAAAAAAADU/KL6KoGwQeRwqKB42ZyE6y-MF8BMtCA04ACK4BGAsYHg/s232/diodo.png'
id='img1'/>";
  cuerpo += "<img style='display:none;' src='https://1.bp.blogspot.com/-iYOpNYfqt7U/Xu3wsqRR-
UI/AAAAAAAAADg/O0QkGmY1VmAycX28PrP1RbUTXrKdlnGugCK4BGAsYHg/s389/DIODO2.png'
id='img2'/>";
```

```
 cuerpo += "<img style='display:none;' src='https://1.bp.blogspot.com/-sfy6Vabf-
Xw/Xu3xAf411uI/AAAAAAAAAEM/tW2b-YtaeJsTNz5Wxizjgf6tO4m8MmgfgCK4BGAsYHg/N-MOS.png'
id='img3'/>";
 cuerpo += "<img style='display:none;' src='https://1.bp.blogspot.com/-
ZYFeMBGNsOs/Xu3xH8fTVHI/AAAAAAAAAEg/0KAHOVlNTGIKQxlATFNMiHLZSyOXPxKDgCK4BGAsY
Hg/P-MOS.png' id='img4'/>";
 cuerpo += "<img style='display:none;' src='https://1.bp.blogspot.com/-
HCpfOVwHYRM/Xu3wTwsgtAI/AAAAAAAAAC4/cxCj9JH726EtShZ-
IM_98UnzzyDb9xkvwCK4BGAsYHg/s150/BJT-NPN%2B%25281%2529.png' id='img5'/>";
 cuerpo += "<img style='display:none;' src='https://1.bp.blogspot.com/-p9dVy_BRvs0/Xu3weBZ-
rbI/AAAAAAAAADE/gUg2c3qUE_QTq5HFEDJopXLbDfbYih6ZACK4BGAsYHg/s150/BJT-
PNP%2B%25281%2529.png' id='img6'/>";
 cuerpo += "<img style='display:none;' src='https://1.bp.blogspot.com/-
IvWLiGACRXk/Xu3wzp1tPCI/AAAAAAAAADw/K6__g4loSloIC1fAf49PEx9Q0elBqw_FgCK4BGAsYHg/JFE
T-N.png' id='img7'/>";
 cuerpo += "<img style='display:none;' src='https://1.bp.blogspot.com/-
ZacqvrrTumM/Xu3w6WE0o9I/AAAAAAAAAD8/khyqY7JteNkjPKOJHaBgRikQ05W2yFpVACK4BGAsYHg/
JFET-P.png' id='img8'/>";
 cuerpo += "</div>";
 cuerpo += "<div style='padding:10px; font-size:35px;display:inline-block;'
id='estado'></div>     ";
 cuerpo += "<button style=' font-size:40px; display:inline-block;' onclick='Leedatos();'>Leer
datos</button></h2>";
 cuerpo += "</div>";
 cuerpo += "<div  style=\" margin: auto; width:745px;\"><div style='text-align:left;'>Intensidad (mA)</div>";
 cuerpo += "<div  style=\" margin: auto; width:700px;height:500px;\" id=\"flotcontainer\"></div>";
 cuerpo += "<div  style=\" margin: auto; width:700px;\"><div style='text-align:center;'>Voltaje (V)</div>";
 cuerpo += "<script src='/JQ.js'>;</script>";
 cuerpo += "<script>";
 cuerpo += "function Leedatos(){var xhttp;xhttp=new XMLHttpRequest();//creo peticion xhttp\n\
 xhttp.open('GET', '/BOTON', true);//genero http rewquest\n\
   xhttp.send();//envio http request\n\
   }";
 cuerpo += "function update(){\n\
 //console.log(\"update\");\n\
 var xhttp;xhttp=new XMLHttpRequest();//creo peticion xhttp\n\
 xhttp.onreadystatechange=function(){//evento cuando recibo la respuesta del servidor\n\
   if (xhttp.readyState == 4 && xhttp.status == 200)//si la peticion es correcta sigo\n\
   {\n\
     console.log(xhttp.responseText);//lo que recibo es un string, y lo imprimo por la consola del navegador
para debuggear\n\
     var json = JSON.parse(xhttp.responseText);//convierto el sting a json\n\
          for(i in json.frases){\n\
       if(document.getElementById(i)){\n\

if(i=='estado'){if(json.frases[i]=='READY'){document.getElementById(i).style.backgroundColor='green';}else
{document.getElementById(i).style.backgroundColor='blue';}}\n\
       document.getElementById(i).innerHTML=json.frases[i]\n\
       }\n\
     }\n\
     if(json.graficos){\n\
```

```
      var nombreimagenes=['img1','img2','img3','img4','img5','img6','img7','img8'];\n\
      for(i in nombreimagenes){\n\

if(json.imagen==nombreimagenes[i]){document.getElementById(nombreimagenes[i]).style.display='inline-
block';}\n\
      else{document.getElementById(nombreimagenes[i]).style.display='none';}\n\
      }\n\
      console.log(json);//madar el json por la consola\n\
        console.log('actualizo grafica');\n\
        plot.getOptions().xaxes[0].min=json.opciones.xaxis.min;\n\
        plot.getOptions().xaxes[0].max=json.opciones.xaxis.max;\n\
        plot.getOptions().yaxes[0].min=json.opciones.yaxis.min;\n\
        plot.getOptions().yaxes[0].max=json.opciones.yaxis.max;\n\
        plot.getOptions().legend.position=json.opciones.legend.position;\n\
        plot.setData(json.graficos);//metes los nuevos datos\n\
        plot.resize();\n\
        plot.setupGrid(plot);\n\
        plot.draw(plot);// refrescar los cambio\n\
      }\n\
    };\n\
  };\n\
    xhttp.open('GET', '/U', true);//genero http rewquest\n\
    xhttp.send();}//envio http request";
  cuerpo += "</script>";
  cuerpo += "<script>";
  cuerpo += "var placeholder = $(\"#flotcontainer\"); //quedate con este id para luego trabajar\n\
        var plot;\n\
        var plotop={xaxis: {min: 0, max:10},yaxis: {min: 0, max: 10},legend:{position:'se'}};";
  cuerpo += "$(document).ready(function(){  \n\
    var d2=[[1,1],[10,10]];\n\
    //d2=[{label:\"Foo\"},{label:\"PePe\"},{label:\"juan\"},{label:\"mikel\"}];\n\
    plot=$.plot(placeholder,d2,plotop);\n\
    });\n";
  cuerpo += "</script>";
  cuerpo += "</body>";
  cuerpo += "</html>";
  ;
  server.send(200, "text/html", cuerpo);
}

/*
    d2=[{label:\"Foo\",data:[[0, 0],[3, 3],[5, 5]]},{label:\"PePe\",data:[[1,9],[3, 3],[9, 2]]}];\n\

var plot = $.plot(placeholder, data, options)
placeholde es un objeto jquery sobre el que se va a plotear anchura altura
data [ [x1, y1], [x2, y2], ... ]  a null value for lines is interpreted as a line segment end, i.e. the points before
and after the null value are not connected.
label para leyendas
{
    label: "y = 3",
    data: [[0, 3], [10, 3]]
```

```
}
[ { label: "Foo", data: [ [10, 1], [17, -14], [30, 5] ] },
  { label: "Bar", data: [ [11, 13], [19, 11], [30, -7] ] }
]

// A null signifies separate line segments

    var d3 = [[0, 12], [7, 12], null, [7, 2.5], [12, 2.5]];

*/

/*


      for (i in document.getElementById('imagenes').children){//recorrer todas las imagenes\n\
        if(json.imagen==document.getElementById('imagenes').children[i].id){//si la imagen coincide con la
que mando\n\
        document.getElementById('imagenes').children[i].style.display='inline-block';\n\
        }\n\
        else if(document.getElementById('imagenes').children[i].id=='undefined'){}\n\
        else{\n\
        document.getElementById('imagenes').children[i].style.display='none';\n\
        }\n\


        */

////////////////////////////////////////////////////////////////////////////////
///////            Actualiza grafico  manda los valores a la pagina web
////////////////////////////////////////////////////////////////////////////////

void ActualizaGrafico()
{
  Serial.println("staaaaaaar2");

  if (flag == true)
  {

    if (numero == 1)
    {
      // carga += "{\"imagen\": \"img1\",\"opciones\": {\"xaxis\": {\"min\": 0, \"max\":10},\"yaxis\": {\"min\": 0,
\"max\": 7},\"legend\":{\"position\":\"se\"}},\"frases\": {\"frase1\":\"hola como estas\", \"frase2\":\"la ganancia
es\",\"estado\":\"" + estado[ready] + "\"},\"graficos\":[";

      // si es la primera grafica que muestro
      carga += "{\"imagen\": \"";
      carga += imagen;
      carga += "\",\"opciones\": {\"xaxis\": {\"min\": ";
      //minX
      carga += x_min;
      carga += ", \"max\":";
      //maxX
```

```
  carga += x_max;
  carga += "},\"yaxis\": {\"min\": ";
  //minY
  carga += y_min;
  carga += ", \"max\": ";
  //maxY
  carga += y_max;
  carga += "},\"legend\":{\"position\":\"";
  //posición labels
  carga += position;
  carga += "\"}},\"frases\": {\"frase1\":\"";
  carga += deteccion;
  carga += "\", \"frase2\":\"";
  carga += gain_trans;
  carga += "\",\"estado\":\"" + estado[ready] + "\"},\"graficos\":[";
  //Serial.print("carga 1: ");
  //Serial.println(carga);
}
else
{

  carga.remove(carga.length() - 2); // para que elimine la ultima "]]"
  carga += ',';
  Serial.println(numero);
}
carga += "{\"label\": ";
//  carga += "\"pepe\"";
carga += "\"";
carga += label;
carga += label_value;
carga += "\"";

carga += ", \"data\": [";

for (uint16_t i = 0; i < countok; i++) // que es esto de uint16_t
{

  carga += '[';
  carga += valorx[i];
  carga += ',';
  carga += valory[i];
  carga += "],";
  yield();
}

carga.remove(carga.length() - 1);
carga += "]}]}";
resetbuffer();
Serial.println("##############################");
Serial.println(numero);
Serial.println("##############################");
```

```
   server.send(200, "application/json", carga);
   //Serial.println(carga);
   delay(200);
   flag = false;
  }
  else
  {
   // Serial.println("flga falso");
   // carga += "[[]]";
   Serial.print("carga vacia");
   //Serial.println(carga);

   server.send(200, "application/json", "{\"frases\": {\"estado\":\"" + estado[ready] + "\",\"frase2\":
\""+gain_trans+"\"},\"imagen\": \"img1\"}");
   delay(200);
  }
}

////////////////////////////////////////////////////////////////////////////////
///////                    Inicia proceso  de medida
////////////////////////////////////////////////////////////////////////////////
void IniciaProceso()
{
  if (ready)
  {
   start = true;
   server.send(200, "text/plain", "ok");
  }
  else
  {
   server.send(200, "text/plain", "nok");
  }
}

void resetbuffer()

{

  for (uint16_t i = 0; i < 292; i++)

  {
   valorx[i] = 0;

   valory[i] = 0;
  }
  countok = 0;
}

void HandleNotFound()
{
```

```
  server.send(404, "text/plain", "Not found");
}

////////////////////////////////////////////////////////////////////////////
///////                    Graficas
////////////////////////////////////////////////////////////////////////////
void graficas()
{

  if (ndetecciones == 1)
  { // compruebo que he detectado que tipo de transitor y que no lo he confundido entre varios tipos

    if (DIODE)
      diode_graph();
    if (DIODE_ZENER)
      diode_zener_graph();
    if (NMOS)
      nmos_graph();
    if (PMOS)
      pmos_graph();
    if (NBJT)
      nbjt_graph();
    if (PBJT)
      pbjt_graph();
    if (NJFET)
      njfet_graph();
    if (PJFET)
      pjfet_graph();

    //poner a false y los puntos a 0
    ndetecciones = 0;
    NBJT = false;
    PBJT = false;
    NMOS = false;
    PMOS = false;
    DIODE = false;
    DIODE_ZENER = false;
    NJFET = false;
    PJFET = false;
  }
  else
  {
    ndetecciones = 0;
    Serial.println("no es posible detectar de que transistor se trata");
  }
  return;
}

////////////////////////////////////////////////////////////////////////////
///////                    Terminales
////////////////////////////////////////////////////////////////////////////
```

```
void terminals()
{
  if (DIODE | DIODE_ZENER)
  {

    if (anodo == 1)
    {
      anodo1 = 0; //terminal A1
      anodo2 = 1; // terminal A2
      select_anodo = 0;
    }
    if (anodo == 2)
    {
      anodo1 = 2; //terminal B1
      anodo2 = 3; //terminal B2
      select_anodo = 1;
    }
    if (anodo == 3)
    {
      anodo1 = 4; //terminal C1
      anodo2 = 5; //terminal C2
      select_anodo = 2;
    }
    if (catodo == 1)
    {
      catodo1 = 0; //terminal A1
      catodo2 = 1; // terminal A2
      select_catodo = 0;
    }
    if (catodo == 2)
    {
      catodo1 = 2; //terminal B1
      catodo2 = 3; //terminal B2
      select_catodo = 1;
    }
    if (catodo == 3)
    {
      catodo1 = 4; //terminal C1
      catodo2 = 5; //terminal C2
      select_catodo = 2;
    }
  }

  if (NMOS || PMOS || NJFET || PJFET)
  {

    if (drain == 1)
    {
      drain1 = 0; //terminal A1
      drain2 = 1; // terminal A2
```

```
  select_drain = 0;
}
if (drain == 2)
{
  drain1 = 2; //terminal B1
  drain2 = 3; //terminal B2
  select_drain = 1;
}
if (drain == 3)
{
  drain1 = 4; //terminal C1
  drain2 = 5; //terminal C2
  select_drain = 2;
}
if (gate == 1)
{
  gate1 = 0;
  gate2 = 1;
  select_gate = 0;
}
if (gate == 2)
{
  gate1 = 2;
  gate2 = 3;
  select_gate = 1;
}
if (gate == 3)
{
  gate1 = 4;
  gate2 = 5;
  select_gate = 2;
}
server.handleClient();
if (source == 1)
{
  source1 = 0;
  source2 = 1;
  select_source = 0;
}
if (source == 2)
{
  source1 = 2;
  source2 = 3;
  select_source = 1;
}
if (source == 3)
{
  source1 = 4;
  source2 = 5;
  select_source = 2;
}
```

```
}

if (NBJT || PBJT)
{
  if (colector == 1)
  {
    colector1 = 0; //terminal A1
    colector2 = 1; // terminal A2
    select_colector = 0;
  }
  if (colector == 2)
  {
    colector1 = 2; //terminal B1
    colector2 = 3; //terminal B2
    select_colector = 1;
  }
  if (colector == 3)
  {
    colector1 = 4; //terminal C1
    colector2 = 5; //terminal C2
    select_colector = 2;
  }
  if (base == 1)
  {
    base1 = 0;
    base2 = 1;
    select_base = 0;
  }
  if (base == 2)
  {
    base1 = 2;
    base2 = 3;
    select_base = 1;
  }
  if (base == 3)
  {
    base1 = 4;
    base2 = 5;
    select_base = 2;
  }
  server.handleClient();
  if (emisor == 1)
  {
    emisor1 = 0;
    emisor2 = 1;
    select_emisor = 0;
  }
  if (emisor == 2)
  {
    emisor1 = 2;
    emisor2 = 3;
```

```
    select_emisor = 1;
   }
   if (emisor == 3)
   {
    emisor1 = 4;
    emisor2 = 5;
    select_emisor = 2;
   }
  }
}
////////////////////////////////////////////////////////////////////////////////
///////                  analisis    diodo zener
////////////////////////////////////////////////////////////////////////////////

void diode_zener_graph()
{
  imagen = "img2";
  x_min = -5.5;
  x_max = 1;
  y_min = -3;
  y_max = 3.5;
  position = "ne";
  gain_trans = "";

  carga = ""; // vacio la cadena donde voy a almacenar los datos de la grafica
  numero = 0; // el contador de graficas a 0
  filtro = 0.05;
  label = "Diodo Zener";

  terminals();

  Serial.print(select_anodo);
  Serial.print(" ");
  Serial.print(" ");
  Serial.println(select_catodo);

  Serial.print(anodo1);
  Serial.print(" ");
  Serial.print(anodo2);
  Serial.print(" ");
  Serial.print(catodo1);
  Serial.print(" ");
  Serial.println(catodo2);

  POTset(select_anodo, 255);  //946 medidos anodo
  POTset(select_catodo, 255); //948 medidos catodo

  DACset(select_anodo, 0);  //anodo
  DACset(select_catodo, 0); // catodo

  charge = true;
```

```
 countok = 0;
 ++numero;
 server.handleClient();

 for (V = 4000; V > 0; V -= 23) // aplico una tensión negativa al diodo 900 para los diodos normales y 3700 para los zener
 {
   DACset(select_catodo, V);
   //catodo
   x = mideTerminal(anodo2) - mideTerminal(catodo2); // Vd tensión anodo catodo
   y = measureI(anodo1, anodo2, 946);              //Idiodo corriente del diodo entre la resistencia que esta el catodo

   //server.handleClient();

   ver_datos();
   server.handleClient();

   //meter los handel client
 }
 for (V = 0; V < 3000; V += 23) //aplico una tensión positva al diodo 1300 , 1000 para los schottky
 {
   DACset(select_anodo, V); //anodo

   x = mideTerminal(anodo2) - mideTerminal(catodo2); // tensión anodo catodo
   y = measureI(anodo1, anodo2, 946);              //corriente del diodo entre la resistencia que esta el catodo

   ver_datos();
   //server.handleClient();
   yield();
 }

 label_value = "";

 flag = true;

 while (flag == true) // hay que meterlo dentro de un while porque sino paraleliza el programa, mientas actualiza el web server sigue tomando datos.
 {
   server.handleClient();
   Serial.print("/");
 }

 Serial.println("");
 Serial.println("////////////////////////////////////////////////////// ");
 Serial.println("");
}

//////////////////////////////////////////////////////////////////////////////
///////                analisis    diodo normal
```

```
////////////////////////////////////////////////////////////////////////////////

void diode_graph()
{
  imagen = "img1";
  x_min = -2.4;
  x_max = 1;
  y_min = -0.5;
  y_max = 4;
  position = "ne";
  gain_trans = "";

  carga = ""; // vacio la cadena donde voy a almacenar los datos de la grafica
  numero = 0; // el contador de graficas a 0
  filtro = 0.05;
  label = "Diodo";

  terminals();
  server.handleClient();

  Serial.print(select_anodo);
  Serial.print(" ");
  Serial.print(" ");
  Serial.println(select_catodo);

  Serial.print(anodo1);
  Serial.print(" ");
  Serial.print(anodo2);
  Serial.print(" ");
  Serial.print(catodo1);
  Serial.print(" ");
  Serial.println(catodo2);

  POTset(select_anodo, 255);  //946 medidos anodo
  POTset(select_catodo, 255); //948 medidos catodo

  DACset(select_anodo, 0);  //anodo
  DACset(select_catodo, 0); // catodo

  charge = true;
  countok = 0;
  ++numero;

  for (V = 900; V > 0; V -= 14) // aplico una tensión negativa al diodo 900 para los diodos normales y 3700
para los zener
  {
    DACset(select_catodo, V);
    //catodo
    x = mideTerminal(anodo2) - mideTerminal(catodo2); // Vd tensión anodo catodo
    y = measureI(anodo1, anodo2, 946);              //Idiodo corriente del diodo entre la resistencia que esta
el catodo
```

```
   //server.handleClient();

   ver_datos();
   server.handleClient();

   //meter los handel client
  }
  for (V = 0; V < 3300; V += 14) //aplico una tensión positva al diodo 1300 , 1000 para los schottky
  {
   DACset(select_anodo, V); //anodo

   x = mideTerminal(anodo2) - mideTerminal(catodo2); // tensión anodo catodo
   y = measureI(anodo1, anodo2, 946);              //corriente del diodo entre la resistencia que esta el
catodo

   ver_datos();
   //server.handleClient();
   yield();
  }

  label_value = "";

  flag = true;

  while (flag == true) // hay que meterlo dentro de un while porque sino paraleliza el programa, mientas
actualiza el web server sigue tomando datos.
  {
   server.handleClient();
   Serial.print("/");
  }

  Serial.println("");
  Serial.println("////////////////////////////////////////////////////// ");
  Serial.println("");
}

//////////////////////////////////////////////////////////////////////////////
///////                analisis    Pjfet graph
//////////////////////////////////////////////////////////////////////////////

void pjfet_graph()
{
  imagen = "img8";
  x_min = -8;
  x_max = 0;
  y_min = -3;
  y_max = 0;
  position = "se";

  server.handleClient();
```

```
Vsource_min = 4000;
Vg = 4000;
filtro = 0.08;
carga = ""; // vacio la cadena donde voy a almacenar los datos de la grafica
numero = 0; // el contador de graficas a 0
label = "Vgs: ";

terminals();

POTset(select_drain, 255);  //946 medidos
POTset(select_gate, 255);   //948 medidos
POTset(select_source, 255); //948 medidos

Vsource_min = 4000;
Vg = 4000;
filtro = 0.03;

for (Vgs_set = 0.6; Vgs_set >=0; Vgs_set -= 0.1)
{
  vgs_anterior = z_sal;
  id_anterior = y_sal;

  DACset(select_drain, 4000);  //drain
  DACset(select_gate, 4000);   //gate
  DACset(select_source, 4000); //source

  charge = true;
  countok = 0;
  ++numero;

  for (V = 4000; V >= 0; V -= 20)
  {
    DACset(select_drain, V);
    if (countok == 0)
    {

      Vgs = mideTerminal(gate2) - mideTerminal(source2); //Vgs entre la gate y la fuente

      while (Vgs > Vgs_set + 0.01 || Vgs < Vgs_set - 0.01)
      { //la primera vez fijo con Vs para obtener la Vgs que quiero ya que Vg=0

        if (Vgs > Vgs_set + 0.01)
          Vsource_min += 5;
        if (Vgs < Vgs_set - 0.01)
          Vsource_min -= 3;

        DACset(select_source, Vsource_min); //fuente
        V = Vsource_min;
        DACset(select_colector, V); // drenador
        yield();
```

```
        Vgs = mideTerminal(gate2) - mideTerminal(source2); //vgs
        yield();
      }
    }
    else
    {
      Vgs = mideTerminal(gate2) - mideTerminal(source2); //Vgs entre la gate y la fuente
      while (Vgs > Vgs_set + 0.05 || Vgs < Vgs_set - 0.05)
      {

        yield();
        if (Vgs > Vgs_set + 0.05)
          Vg -= 3;
        if (Vgs < Vgs_set - 0.05)
          Vg += 5;
        DACset(select_gate, Vg);                        //aplico tension a la puerta
        Vgs = mideTerminal(gate2) - mideTerminal(source2); //Vgs entre la gate y la fuente
        yield();
      }
    }

    yield();

    x = mideTerminal(drain2) - mideTerminal(source2); //calculo Vce
    y = measureI(drain1, drain2, 946);
    z = mideTerminal(gate2) - mideTerminal(source2);

    ver_datos();
    server.handleClient();
    yield();
  }
  label_value = Vgs_set;
  label_value += "V";

  flag = true;
gain_trans = "la transconductancia es: ?";
  while (flag == true) // hay que meterlo dentro de un while porque sino paraleliza el programa, mientas actualiza el web server sigue tomando datos.
  {
    server.handleClient();
    Serial.print("/");
  }

  Serial.println("");
  Serial.println("////////////////////////////////////////////////////// ");
  Serial.println("");
 }

 vgs_final = z_sal;
 id_final = y_sal;
```

```
 gm = (fabs(id_final - id_anterior) / fabs(vgs_final - vgs_anterior));
 Serial.print("gm: ");
 Serial.println(gm);
 gain_trans = "la transconductancia es: "+(String)gm;
 /*
 while (flag == true) // hay que meterlo dentro de un while porque sino paraleliza el programa, mientas actualiza el web server sigue tomando datos.
 {
  server.handleClient();
  Serial.print("/");
 }
*/
 return;
}

////////////////////////////////////////////////////////////////////////////////
///////                analisis    Njfet graph
////////////////////////////////////////////////////////////////////////////////

void njfet_graph()
{

 imagen = "img7";
 x_min = 0;
 x_max = 8;
 y_min = 0;
 y_max = 4;
 position = "ne";


 Vsource_min = 0;
 carga = ""; // vacio la cadena donde voy a almacenar los datos de la grafica
 numero = 0; // el contador de graficas a 0
 filtro = 0.08;
 label = "Vgs: ";

 terminals();

 POTset(select_gate, 255);   //946 medidos gate
 POTset(select_drain, 255);  //948 medidos drain
 POTset(select_source, 255); //948 medidos source

 for (Vgs_set = 0; Vgs_set > (-2.1); Vgs_set -= 0.4)
 {

  vgs_anterior = z_sal;
  id_anterior = y_sal;

  DACset(select_gate, 0);   //gate
  DACset(select_drain, 0);  //drain
```

```
  DACset(select_source, 0); //source

  charge = true;
  countok = 0;
  ++numero;

  for (V = 0; V < 4095; V += 14) // fijo una Vmin porque la tensión del drenador no puede ser mas bajo que el de la fuente
  {
    DACset(select_drain, V); //drain
    if (countok == 0)
    {
      Vgs = mideTerminal(gate2) - mideTerminal(source2); //Vgs entre la gate y la fuente
      while (Vgs > Vgs_set + 0.05 || Vgs < Vgs_set - 0.05)
      { //la primera vez fijo con Vs para obtener la Vgs que quiero ya que Vg=0

        if (Vgs > Vgs_set + 0.05)
          Vsource_min += 5;
        if (Vgs < Vgs_set - 0.05)
          Vsource_min -= 3;

        DACset(select_source, Vsource_min); //fuente
        V = Vsource_min;
        DACset(select_drain, V); // drenador    igualo el valor del drenador para no tener Vds negativos }
        yield();

        Vgs = mideTerminal(gate2) - mideTerminal(source2); //vgs
      }
    }
    else /// el resto de veces como Vdrain aumenta Vs auementa y por tanto Vgs aumenta, voy manteniendo cte Vgs aumentado Vg
    {
      Vgs = mideTerminal(gate2) - mideTerminal(source2); //Vgs entre la gate y la fuente
      while (Vgs >= Vgs_set + 0.05 || Vgs <= Vgs_set - 0.05)
      {
        yield();
        if (Vgs > Vgs_set + 0.05)
          Vg -= 3;
        if (Vgs < Vgs_set - 0.05)
          Vg += 5;
        DACset(select_gate, Vg);                    //aplico tension a la puerta
        Vgs = mideTerminal(gate2) - mideTerminal(source2); //Vgs entre la gate y la fuente
      }
    }

    x = mideTerminal(drain2) - mideTerminal(source2); //calculo Vds Vdrain-Vsource
    y = measureI(drain1, drain2, 946);              // caludo Idrenador
    z = mideTerminal(gate2) - mideTerminal(source2);  //calculo Vgs Vgate-Vsource

    ver_datos();
    server.handleClient();
```

```
    yield();
   }

   label_value = Vgs_set;
   label_value += "V";

   flag = true;
gain_trans = "la transconductancia es: ?";
   while (flag == true) // hay que meterlo dentro de un while porque sino paraleliza el programa, mientas actualiza el web server sigue tomando datos.
   {
    server.handleClient();
    Serial.print("/");
   }

   Serial.println("");
   Serial.println("////////////////////////////////////////////////////// ");
   Serial.println("");
  }

  vgs_final = z_sal;
  id_final = y_sal;

  gm = (fabs(id_final - id_anterior) / fabs(vgs_final - vgs_anterior));
  Serial.print("gm: ");
  Serial.println(gm);
gain_trans = "la transconductancia es: "+(String)gm;
  /*
  while (flag == true) // hay que meterlo dentro de un while porque sino paraleliza el programa, mientas actualiza el web server sigue tomando datos.
  {
   server.handleClient();
   Serial.print("/");
  }
*/
  return;
}

////////////////////////////////////////////////////////////////////////////////
///////                analisis    Pmos graph
////////////////////////////////////////////////////////////////////////////////

void pmos_graph()
{
  imagen = "img4";
  x_min = -9.5;
  x_max = 0;
  y_min = -7;
  y_max = 0;
  position = "se";
```

```
terminals();

POTset(select_drain, 255);  //948 medidos drain
POTset(select_gate, 255);   //948 medidos gate
POTset(select_source, 255); //948 medidos source

Vg = 1366;  // valor de comienzo de Vgate para que vaya mas rápido
carga = ""; // vacio la cadena donde voy a almacenar los datos de la grafica
numero = 0; // el contador de graficas a 0
filtro = 0.1;
label = "Vgs: ";

for (Vgs_set = -2.25; Vgs_set >= -2.55; Vgs_set -= 0.05)
{

  vgs_anterior = z_sal;
  id_anterior = y_sal;

  DACset(select_drain, 0);     ///  el drenador a cero
  DACset(select_gate, 0);      // gate a 0
  DACset(select_source, 4000); // drenador a maxima tensión

  charge = true;
  countok = 0;
  ++numero;

  for (Vd = 4000; Vd >= 0; Vd -= 20)
  {

    DACset(select_drain, Vd); /// voy disminuyendo la tension del drenador para que auemente Vds

    Vgs = mideTerminal(gate2) - mideTerminal(source2); //calculo Vgs

    while (Vgs >= Vgs_set + 0.01 || Vgs <= Vgs_set - 0.01)
    {

      if (Vgs > (Vgs_set + 0.01))
        Vg -= 5;
      if (Vgs < (Vgs_set - 0.01))
        Vg += 3;

      DACset(select_gate, Vg);                    // vario la tension de la puerta para obetener la Vgs
deseada y mantenerla
      Vgs = mideTerminal(gate2) - mideTerminal(source2); //calculo Vgs
      yield();
    }

    x = (mideTerminal(drain2) - mideTerminal(source2)); // Vds drain - Vsource
    y = measureI(drain1, drain2, 948);                ////entre los dos terminales de la resistencia del drenador
    z = mideTerminal(gate2) - mideTerminal(source2);    //Vgs Vgate-Vsource
```

```
    ver_datos();
    server.handleClient();
    yield();
   }

   label_value = Vgs_set;
   label_value += "V";

   flag = true;
gain_trans = "la transconductancia es: ?";
   while (flag == true) // hay que meterlo dentro de un while porque sino paraleliza el programa, mientas actualiza el web server sigue tomando datos.
   {
    server.handleClient();
    Serial.print("/");
   }

   Serial.println("");
   Serial.println("////////////////////////////////////////////////////// ");
   Serial.println("");
  }

  vgs_final = z_sal;
  id_final = y_sal;

  gm = (fabs(id_final - id_anterior) / fabs(vgs_final - vgs_anterior));
  Serial.print("gm: ");
  Serial.println(gm);
gain_trans = "la transconductancia es: "+(String)gm;
  /*
  while (flag == true) // hay que meterlo dentro de un while porque sino paraleliza el programa, mientas actualiza el web server sigue tomando datos.
  {
   server.handleClient();
   Serial.print("/");
  }
*/
  return;
}

////////////////////////////////////////////////////////////////////////////////
///////                analisis    Nmos graph
////////////////////////////////////////////////////////////////////////////////

void nmos_graph()
{

  imagen = "img3";
  x_min = 0;
  x_max = 8;
```

```
y_min = 0;
y_max = 5;
position = "ne";

terminals();
Serial.print(select_drain);
Serial.print(" ");
Serial.print(select_gate);
Serial.print(" ");
Serial.println(select_source);

POTset(select_drain, 255);  //10 medidos drain
POTset(select_gate, 255);   //100k medidos gate
POTset(select_source, 255); //948 medidos source

Vg = 1366;  // valor de comienzo de Vgate para que vaya mas rápido
carga = ""; // vacio la cadena donde voy a almacenar los datos de la grafica
numero = 0; // el contador de graficas a 0
filtro = 0.03;
label = "Vgs: ";

////////////////////////////////////////7debug

for (Vgs_set = 1.95; Vgs_set <= 2.15; Vgs_set += 0.05)
{

  vgs_anterior = z_sal;
  id_anterior = y_sal;

  DACset(select_drain, 0);  //drain
  DACset(select_gate, 0);   //gate
  DACset(select_source, 0); //source

  charge = true;
  countok = 0;
  ++numero;

  for (Vd = 0; Vd <= 4000; Vd += 18)
  {

    DACset(select_drain, Vd); //drain

    Vgs = mideTerminal(gate2) - mideTerminal(source2); //calculo Vgs

    while (Vgs >= Vgs_set + 0.01 || Vgs <= Vgs_set - 0.01)
    {

      yield();
      if (Vgs > (Vgs_set + 0.01))
        Vg -= 3;
      if (Vgs < (Vgs_set - 0.01))
```

```
        Vg += 5;
       DACset(select_gate, Vg);
       Vgs = mideTerminal(gate2) - mideTerminal(source2); //calculo Vgs
     }

     x = mideTerminal(drain2) - mideTerminal(source2); // Vds drenador-fuente
     y = measureI(drain1, drain2, 948);               ////Idrenador entre los dos terminales de la resistencia del
drenador
     z = mideTerminal(gate2) - mideTerminal(source2);  // Vgs Vgate -Vsource

     ver_datos();
     server.handleClient();
     yield();
   }

   label_value = Vgs_set;
   label_value += "V";

   flag = true;
gain_trans = "la transconductancia es: ?";
   while (flag == true) // hay que meterlo dentro de un while porque sino paraleliza el programa, mientas
actualiza el web server sigue tomando datos.
   {
     server.handleClient();
     Serial.print("/");
   }

   Serial.println("");
   Serial.println("////////////////////////////////////////////////////// ");
   Serial.println("");
  }

  vgs_final = z_sal;
  id_final = y_sal;

  gm = (fabs(id_final - id_anterior) / fabs(vgs_final - vgs_anterior));
  Serial.print("gm: ");
  Serial.println(gm);
gain_trans = "la transconductancia es: "+(String)gm;
  /*
  while (flag == true) // hay que meterlo dentro de un while porque sino paraleliza el programa, mientas
actualiza el web server sigue tomando datos.
  {
    server.handleClient();
    Serial.print("/");
  }
*/
  return;
}

//////////////////////////////////////////////////////////////////////////////
```

```
///////                  analisis    Pbjt graph
////////////////////////////////////////////////////////////////////////////////

void pbjt_graph()
{
  imagen = "img6";
  x_min = -9;
  x_max = 0;
  y_min = -7;
  y_max = 0;
  position = "se";

  //borrar esto
  Serial.println("he llegado");
  Serial.println(colector);
  Serial.println(base);
  Serial.println(emisor);

  terminals();

  Vb = 1366;  // valor de comienzo de Vbase para que vaya mas rápido
  carga = ""; // vacio la cadena donde voy a almacenar los datos de la grafica
  numero = 0; // el contador de graficas a 0
  filtro = 0.09;
  label = "Ibase: ";

  POTset(select_colector, 255); //946 medidos colector
  POTset(select_base, 127);     //100k medidos base
  POTset(select_emisor, 255);   //948 medidos emisor

  for (IbSet = -2.0; IbSet > -23.0; IbSet -= 4.0) //IbSet = -2.0; IbSet > -23.0; IbSet -= 4.0
  {

    DACset(select_colector, 0);  //colector
    DACset(select_base, Vb);     //3,3vbase
    DACset(select_emisor, 4095); // emisor

    ++numero;
    charge = true;
    countok = 0;

    for (V = 4095; V >= 0; V -= 18)
    {

      DACset(select_colector, V); // colector

      Ibase = measureI(base1, base2, 100); //corriente de la base

      while (Ibase >= IbSet + 0.1 || Ibase <= IbSet - 0.1)
      {
```

```
      yield();

      if (Ibase > IbSet + 0.1)
        Vb -= 10;
      if (Ibase < IbSet - 0.1)
        Vb += 7;

      DACset(select_base, Vb); // base
      server.handleClient();
      Ibase = measureI(base1, base2, 100);
    }

    x = (mideTerminal(colector2) - mideTerminal(emisor2)); // calculo Vce Vcolector - Vemisor
    y = measureI(colector1, colector2, 946);               // calculo la corriente de drenador
    z = measureI(base1, base2, 100);                       // calculo la corriente de base

    ver_datos();
    server.handleClient();
    yield();
  }
  label_value = IbSet;
  label_value += "µA";
  flag = true;
  gain_trans = "la ganancia es: ?";
  while (flag == true) // hay que meterlo dentro de un while porque sino paraleliza el programa, mientas
actualiza el web server sigue tomando datos.
  {
    server.handleClient();
    Serial.print("/");
  }
  Serial.println("////////////////////////////////////////////////////// ");
 }
 ib_final = z_sal;
 ic_final = y_sal;

 hfe = (fabs(ic_final) / fabs(ib_final));
 // ib_final  son ua  multiplico por 1000
 hfe = hfe * 1000;
 Serial.print("hfe: ");
 Serial.println(hfe);
gain_trans = "la ganancia es: "+(String)hfe;
 /*
 while (flag == true) // hay que meterlo dentro de un while porque sino paraleliza el programa, mientas
actualiza el web server sigue tomando datos.
 {
  server.handleClient();
  Serial.print("/");
 }
*/
 return;
}
```

```
////////////////////////////////////////////////////////////////////////////////
///////                  analisis    Nbjt graph
////////////////////////////////////////////////////////////////////////////////

void nbjt_graph()
{
  imagen = "img5";
  x_min = 0;
  x_max = 9;
  y_min = 0;
  y_max = 3.5;
  position = "ne";

  //borrar esto
  Serial.println("he llegado");
  Serial.println(colector);
  Serial.println(base);
  Serial.println(emisor);

  terminals();

  Vb = 1366;  // valor de comienzo de Vbase para que vaya mas rápido
  carga = ""; // vacio la cadena donde voy a almacenar los datos de la grafica
  numero = 0; // el contador de graficas a 0
  filtro = 0.09;
  label = "Ibase: ";

  POTset(select_colector, 255); //946 medidos R colector
  POTset(select_base, 137);     //100k medidos R base
  POTset(select_emisor, 255);   //948 medidos R emisor

  for (IbSet = 2.0; IbSet < 9.0; IbSet += 1.0)
  {

    DACset(select_colector, 0); //tesión colector
    DACset(select_base, Vb);    //3,3vbase tensión base
    DACset(select_emisor, 0);   //tensión emisor

    ++numero; //saber el numero de la grafiaca
    charge = true;
    countok = 0;

    for (V = 0; V < 4000; V += 20)
    {
      DACset(select_colector, V); //voy aumentando la tensión del colector

      Ibase = measureI(base1, base2, 100); //corriente de base
      while (Ibase >= IbSet + 0.1 || Ibase <= IbSet - 0.1)
      {
        yield();
```

```
      if (Ibase > IbSet + 0.1)
        Vbas -= 10;
      if (Ibase < IbSet - 0.1)
        Vbas += 7;

      DACset(select_base, Vbas); //cambio tension de base
      Ibase = measureI(base1, base2, 100);
      server.handleClient();
    }

    x = mideTerminal(colector2) - mideTerminal(emisor2); //calculo Vce Vcolector-Vemiros
    y = measureI(colector1, colector2, 946);             //intensidad de colector
    z = measureI(base1, base2, 100);                     //calculo tension de base

    ver_datos();
    server.handleClient();
    yield();
   }
   label_value = IbSet;
   label_value += "µA";

   flag = true;
gain_trans = "la ganancia es: ?";
   while (flag == true) // hay que meterlo dentro de un while porque sino paraleliza el programa, mientas
actualiza el web server sigue tomando datos.
   {
    server.handleClient();
    Serial.print("/");
   }
   Serial.println("////////////////////////////////////////////////////// ");
 }

 ib_final = z_sal;
 ic_final = y_sal;

 hfe = (fabs(ic_final) / fabs(ib_final));
 // ib_final  son ua  multiplico por 1000
 hfe = hfe * 1000;
 Serial.print("hfe: ");
 Serial.println(hfe);
 gain_trans = "la ganancia es: "+(String)hfe;

 /*
 while (flag == true) // hay que meterlo dentro de un while porque sino paraleliza el programa, mientas
actualiza el web server sigue tomando datos.
 {
  server.handleClient();
  Serial.print("/");
 }
*/
```

```
  return;
}

////////////////////////////////////////////////////////////////////////////////
///////                    ver datos   filtramos  y almacenamos
////////////////////////////////////////////////////////////////////////////////

void ver_datos()
{
  if (charge == true)
  { ///cargo el filtro para los primeros valores

    x_sal = x;
    y_sal = y;
    z_sal = z;

    charge = false;
    return;
  }

  ////aplicamos filtro

  x_sal = x_sal + (filtro * (x - x_sal));
  y_sal = y_sal + (filtro * (y - y_sal));
  z_sal = z_sal + (filtro * (z - z_sal));

  valorx[countok] = x_sal;

  valory[countok] = y_sal;
  Serial.print(valorx[countok]);

  Serial.print("  ");

  Serial.print(valory[countok]);
  Serial.print("  ");

  Serial.print(z_sal);
  Serial.print("  ");

  Serial.print(countok);
  Serial.print("  ");

  Serial.print(start);
  Serial.print("  ");

  Serial.println(numero);

  ++countok;

  yield();
}
```

```
////////////////////////////////////////////////////////////////////////////
///////                 Detectamos terminales
////////////////////////////////////////////////////////////////////////////
void detectar_terminal()
{

  detectar_diodo();
  detectar_MOSFET();
  detectar_NPNbjt();
  detectar_PNPbjt();
  detectar_NJFET();
  detectar_PJFET();

  return;
}

////////////////////////////////////////////////////////////////////////////
///////                 Detectar diodo
////////////////////////////////////////////////////////////////////////////
void detectar_diodo()
{

  POTset(0, 255);
  POTset(1, 255);
  POTset(2, 255);

  ////////////////////////////////////// terminal A anodo

  DACset(0, 3000);
  DACset(1, 0);
  DACset(2, 0);

  VA2 = mideTerminal(TERMINAL_A2);
  VB2 = mideTerminal(TERMINAL_B2);
  VC2 = mideTerminal(TERMINAL_C2);

  DACset(0, 0);
  DACset(1, 0);
  DACset(2, 3000);

  Vcomp1 = int(VB2 * 10);
  Vcomp2 = int(VC2 * 10);

  if (Vcomp1 == 0)
  {

    Vdif1 = VA2 - VC2;

    if (Vdif1 > 0.5 & Vdif1 < 0.9)
    { ///esto se puede cumplir tb en los BJT PNP A podria ser el Colector y C la base
```

```
    DACset(0, 0);
    DACset(1, 3000); //// si es un BJT y aplicara a B al supuesto emisor tension la supuesta base C no
deberia estas a A
    DACset(2, 0);

    VA2 = mideTerminal(TERMINAL_A2);
    VC2 = mideTerminal(TERMINAL_C2);

    Vcomp1 = int(VA2 * 10); ///si es un dido ni A ni B deberian verse afectados
    Vcomp2 = int(VC2 * 10);

    if (Vcomp1 == 0 & Vcomp2 == 0)
    { //// Si se cumple esto me aseguro de que no es un

      DACset(0, 3000);
      DACset(1, 3000); ///aplico voltaje al terminal B para comprobar que no se trata de un NMOS
      DACset(2, 0);    /// si se trata de un NMOS la tensión de la fuente drenador bajaría y lo
                       /// habría detectado sería el diodo parasito del sustrato

      VA2 = mideTerminal(TERMINAL_A2);
      VC2 = mideTerminal(TERMINAL_C2);
      Vdif1 = VA2 - VC2;

      if (Vdif1 > 0.5 & Vdif1 < 0.9)
      { // si es un diodo la tensión ANODO CATODO seguira siendo la misma

        DACset(0, 0);
        DACset(1, 0); ///miro a ver si conduce en inversa para ver si es un diodo zenner
        DACset(2, 3500);

        VA2 = mideTerminal(TERMINAL_A2);
        Vcomp1 = int(VA2 * 10);

        if (Vcomp1 == 0)
        {

          Serial.println("Diodo: terminal 1 ANODO y terminal 3 CATODO");
          deteccion = "Diodo: terminal 1 ANODO y terminal 3 CATODO";
          DIODE = true;
          anodo = 1;
          catodo = 3;
          ++ndetecciones;
        }
        else
        {

          Serial.println("Diodo Zener: terminal 1 ANODO y terminal 3 CATODO");
          deteccion = "Diodo Zener: terminal 1 ANODO y terminal 3 CATODO";
          DIODE_ZENER = true;
          anodo = 1;
          catodo = 3;
```

```
          ++ndetecciones;
        }
       }
     }
   }
 }
 else if (Vcomp2 == 0)
 {

  Vdif1 = VA2 - VB2;

  if (Vdif1 > 0.5 & Vdif1 < 0.9)
  {

    DACset(0, 0);
    DACset(1, 0); //// si es un BJT y aplicara a B al supuesto emisor tension la supuesta base C no deberia estas a A
    DACset(2, 3000);

    VA2 = mideTerminal(TERMINAL_A2);
    VB2 = mideTerminal(TERMINAL_B2);

    Vcomp1 = int(VA2 * 10);
    Vcomp2 = int(VB2 * 10);

    if (Vcomp1 == 0 & Vcomp2 == 0)
    {

      DACset(0, 3000);
      DACset(1, 0);    ///aplico voltaje al terminal C para comprobar que no se trata de un NMOS
      DACset(2, 3000); /// si se trata de un NMOS la tensión de la fuente drenador bajaría y lo
                       /// habría detectado sería el diodo parasito del sustrato

      VA2 = mideTerminal(TERMINAL_A2);
      VB2 = mideTerminal(TERMINAL_B2);
      Vdif1 = VA2 - VB2;

      if (Vdif1 > 0.5 & Vdif1 < 0.9)
      { // si es un diodo la tensión ANODO CATODO seguira siendo la misma

        DACset(0, 0);
        DACset(1, 3500); ///miro a ver si conduce en inversa para ver si es un diodo zenner
        DACset(2, 0);

        VA2 = mideTerminal(TERMINAL_A2);
        Vcomp1 = int(VA2 * 10);

        if (Vcomp1 == 0)
        {

          Serial.println("Diodo: terminal 1 ANODO y terminal 2 CATODO");
```

```
        deteccion = "Diodo: terminal 1 ANODO y terminal 2 CATODO";
        DIODE = true;
        anodo = 1;
        catodo = 2;
        ++ndetecciones;
      }
      else
      {

        Serial.println("Diodo Zenner: terminal 1 ANODO y terminal 2 CATODO");
        deteccion = "Diodo Zenner: terminal 1 ANODO y terminal 2 CATODO";
        DIODE_ZENER = true;
        anodo = 1;
        catodo = 2;
        ++ndetecciones;
      }
    }
  }
 }
}

//////////////////////////////////////////// terminal B anodo

DACset(0, 0);
DACset(1, 3000);
DACset(2, 0);

VA2 = mideTerminal(TERMINAL_A2);
VB2 = mideTerminal(TERMINAL_B2);
VC2 = mideTerminal(TERMINAL_C2);

Vcomp1 = int(VA2 * 10);
Vcomp2 = int(VC2 * 10);

if (Vcomp1 == 0)
{

 Vdif1 = VB2 - VC2;

 if (Vdif1 > 0.5 & Vdif1 < 0.9)
 {

   DACset(0, 3000);
   DACset(1, 0);
   DACset(2, 0);

   VB2 = mideTerminal(TERMINAL_B2);
   VC2 = mideTerminal(TERMINAL_C2);

   Vcomp1 = int(VB2 * 10);
   Vcomp2 = int(VC2 * 10);
```

```
  if (Vcomp1 == 0 & Vcomp2 == 0)
  {

    DACset(0, 3000);
    DACset(1, 3000); ///aplico voltaje al terminal A para comprobar que no se trata de un NMOS
    DACset(2, 0);    /// si se trata de un NMOS la tensión de la fuente drenador bajaría y lo
                     /// habría detectado sería el diodo parasito del sustrato

    VB2 = mideTerminal(TERMINAL_B2);
    VC2 = mideTerminal(TERMINAL_C2);
    Vdif1 = VB2 - VC2;

    if (Vdif1 > 0.5 & Vdif1 < 0.9)
    { // si es un diodo la tensión ANODO CATODO seguira siendo la misma

      DACset(0, 0);
      DACset(1, 0); ///miro a ver si conduce en inversa para ver si es un diodo zenner
      DACset(2, 3500);

      VB2 = mideTerminal(TERMINAL_B2);
      Vcomp1 = int(VB2 * 10);

      if (Vcomp1 == 0)
      {

        Serial.println("Diodo: terminal 2 ANODO y terminal 3 CATODO");
        deteccion = "Diodo: terminal 2 ANODO y terminal 3 CATODO";
        DIODE = true;
        anodo = 2;
        catodo = 3;
        ++ndetecciones;
      }
      else
      {
        Serial.println("Diodo Zener: terminal 2 ANODO y terminal 3 CATODO");
        deteccion = "Diodo Zener: terminal 2 ANODO y terminal 3 CATODO";
        DIODE_ZENER = true;
        anodo = 2;
        catodo = 3;
        ++ndetecciones;
      }
    }
  }
}
else if (Vcomp2 == 0)
{

  Vdif1 = VB2 - VA2;
```

```
if (Vdif1 > 0.5 & Vdif1 < 0.9)
{
  DACset(0, 0);
  DACset(1, 0);
  DACset(2, 3000);

  VA2 = mideTerminal(TERMINAL_A2);
  VB2 = mideTerminal(TERMINAL_B2);

  Vcomp1 = int(VA2 * 10);
  Vcomp2 = int(VB2 * 10);

  if (Vcomp1 == 0 & Vcomp2 == 0)
  {

    DACset(0, 0);
    DACset(1, 3000); ///aplico voltaje al terminal C para comprobar que no se trata de un NMOS
    DACset(2, 3000); /// si se trata de un NMOS la tensión de la fuente drenador bajaría y lo
                     /// habría detectado sería el diodo parasito del sustrato

    VA2 = mideTerminal(TERMINAL_A2);
    VB2 = mideTerminal(TERMINAL_B2);
    Vdif1 = VB2 - VA2;

    if (Vdif1 > 0.5 & Vdif1 < 0.9)
    { // si es un diodo la tensión ANODO CATODO seguira siendo la misma

      DACset(0, 3500);
      DACset(1, 0); ///miro a ver si conduce en inversa para ver si es un diodo zenner
      DACset(2, 0);

      VB2 = mideTerminal(TERMINAL_B2);
      Vcomp1 = int(VB2 * 10);

      if (Vcomp1 == 0)
      {

        Serial.println("Diodo: terminal 2 ANODO y terminal 1 CATODO");
        deteccion = "Diodo: terminal 2 ANODO y terminal 1 CATODO";
        DIODE = true;
        anodo = 2;
        catodo = 1;
        ++ndetecciones;
      }
      else
      {
        Serial.println("Diodo Zener: terminal 2 ANODO y terminal 1 CATODO");
        deteccion = "Diodo Zener: terminal 2 ANODO y terminal 1 CATODO";
        DIODE_ZENER = true;
        anodo = 2;
        catodo = 1;
```

```
        ++ndetecciones;
       }
      }
     }
    }
  }

////////////////////////////////////////// terminal C anodo

DACset(0, 0);
DACset(1, 0);
DACset(2, 3000);

VA2 = mideTerminal(TERMINAL_A2);
VB2 = mideTerminal(TERMINAL_B2);
VC2 = mideTerminal(TERMINAL_C2);

Vcomp1 = int(VA2 * 10);
Vcomp2 = int(VB2 * 10);

if (Vcomp1 == 0)
{

  Vdif1 = VC2 - VB2;

  if (Vdif1 > 0.5 & Vdif1 < 0.9)
  {

    DACset(0, 3000);
    DACset(1, 0);
    DACset(2, 0);

    VB2 = mideTerminal(TERMINAL_B2);
    VC2 = mideTerminal(TERMINAL_C2);

    Vcomp1 = int(VB2 * 10);
    Vcomp2 = int(VC2 * 10);

    if (Vcomp1 == 0 & Vcomp2 == 0)
    {

      DACset(0, 3000);
      DACset(1, 0);    ///aplico voltaje al terminal A para comprobar que no se trata de un NMOS
      DACset(2, 3000); /// si se trata de un NMOS la tensión de la fuente drenador bajaría y lo
                       /// habría detectado sería el diodo parasito del sustrato

      VC2 = mideTerminal(TERMINAL_C2);
      VB2 = mideTerminal(TERMINAL_B2);
      Vdif1 = VC2 - VB2;

      if (Vdif1 > 0.5 & Vdif1 < 0.9)
```

```
    { // si es un diodo la tensión ANODO CATODO seguira siendo la misma

      DACset(0, 0);
      DACset(1, 3500); ///miro a ver si conduce en inversa para ver si es un diodo zenner
      DACset(2, 0);

      VC2 = mideTerminal(TERMINAL_C2);
      Vcomp1 = int(VC2 * 10);

      if (Vcomp1 == 0)
      {

        Serial.println("Diodo: terminal 3 ANODO y terminal 2 CATODO");
        deteccion = "Diodo: terminal 3 ANODO y terminal 2 CATODO";

        DIODE = true;
        anodo = 3;
        catodo = 2;
        ++ndetecciones;
      }
      else
      {
        Serial.println("Diodo Zener: terminal 3 ANODO y terminal 2 CATODO");
        deteccion = "Diodo Zener: terminal 3 ANODO y terminal 2 CATODO";

        DIODE_ZENER = true;
        anodo = 3;
        catodo = 2;
        ++ndetecciones;
      }
    }
  }
}
else if (Vcomp2 == 0)
{

  Vdif1 = VC2 - VA2;

  if (Vdif1 > 0.5 & Vdif1 < 0.9)
  {
    DACset(0, 0);
    DACset(1, 3000);
    DACset(2, 0);

    VA2 = mideTerminal(TERMINAL_A2);
    VC2 = mideTerminal(TERMINAL_C2);

    Vcomp1 = int(VA2 * 10);
    Vcomp2 = int(VC2 * 10);
```

```
    if (Vcomp1 == 0 & Vcomp2 == 0)
    {

      DACset(0, 0);
      DACset(1, 3000); ///aplico voltaje al terminal B para comprobar que no se trata de un NMOS
      DACset(2, 3000); /// si se trata de un NMOS la tensión de la fuente drenador bajaría y lo
                       /// habría detectado sería el diodo parasito del sustrato

      VA2 = mideTerminal(TERMINAL_A2);
      VC2 = mideTerminal(TERMINAL_C2);
      Vdif1 = VC2 - VA2;

      if (Vdif1 > 0.5 & Vdif1 < 0.9)
      { // si es un diodo la tensión ANODO CATODO seguira siendo la misma

        DACset(0, 3500);
        DACset(1, 0); ///miro a ver si conduce en inversa para ver si es un diodo zenner
        DACset(2, 0);

        VC2 = mideTerminal(TERMINAL_C2);
        Vcomp1 = int(VC2 * 10);

        if (Vcomp1 == 0)
        {

          Serial.println("Diodo: terminal 3 ANODO y terminal 1 CATODO");
          deteccion = "Diodo: terminal 3 ANODO y terminal 1 CATODO";

          DIODE = true;
          anodo = 3;
          catodo = 1;
          ++ndetecciones;
        }
        else
        {
          Serial.println("Diodo Zener: terminal 3 ANODO y terminal 1 CATODO");
          deteccion = "Diodo Zener: terminal 3 ANODO y terminal 1 CATODO";

          DIODE_ZENER = true;
          anodo = 3;
          catodo = 1;
          ++ndetecciones;
        }
      }
    }
   }
  }
}

////////////////////////////////////////////////////////////////////////////////
///////                  Detectar  MOSFET
```

```
////////////////////////////////////////////////////////////////////////////////

void detectar_MOSFET()
{

 ////////////////////detectar si es NMOS o PMOS ////////////////// detecto los ceros y luego cual es la guente

 i = 0;
 // voy a contar cuantas veces hay dos ceros cuando aplico una tension de 5 voltios, en caso de que haya 2 veces dos ceros sera NPN pq el transistor no conduce

 ////////////////////////ceros canal A
 POTset(0, 127);
 POTset(1, 127);
 POTset(2, 127);

 DACset(0, 2000);
 DACset(1, 0);
 DACset(2, 0);

 // VA2 = mideTerminal(TERMINAL_A2); //no me hace falta
 VB2 = mideTerminal(TERMINAL_B2);
 VC2 = mideTerminal(TERMINAL_C2);

 Vcomp1 = int(VB2 * 10);
 Vcomp2 = int(VC2 * 10);

 if (Vcomp1 == 0 & Vcomp2 == 0)
  i += 1;

 ////////////////////////////ceros canal B
 POTset(0, 127);
 POTset(1, 127);
 POTset(2, 127);

 DACset(0, 0);
 DACset(1, 2000);
 DACset(2, 0);

 VA2 = mideTerminal(TERMINAL_A2);
 VC2 = mideTerminal(TERMINAL_C2);

 Vcomp1 = int(VA2 * 10);
 Vcomp2 = int(VC2 * 10);

 if (Vcomp1 == 0 & Vcomp2 == 0)
  i += 1;

 ////////////////////////ceros canal C
 POTset(0, 127);
 POTset(1, 127);
```

```
POTset(2, 127);

DACset(0, 0);
DACset(1, 0);
DACset(2, 2000);

VA2 = mideTerminal(TERMINAL_A2);
VB2 = mideTerminal(TERMINAL_B2);

Vcomp1 = int(VA2 * 10);
Vcomp2 = int(VB2 * 10);

if (Vcomp1 == 0 & Vcomp2 == 0)
  i += 1;

////////////////////////////////////////// deteccion de NMOS ////////////////////////////
////////////////////////////////////////// deteccion de NMOS ////////////////////////////

if (i == 2)
{ /// en los BJT NPN tambien detecta 2 ceros

  ////////////////////// terminal A fuente (SOURCE)
  ////////////////////// terminal A fuente (SOURCE)

  POTset(0, 127);
  POTset(1, 127);
  POTset(2, 127);

  DACset(0, Vb);
  DACset(1, 0);
  DACset(2, 0);

  VA2 = mideTerminal(TERMINAL_A2);
  VB2 = mideTerminal(TERMINAL_B2);
  VC2 = mideTerminal(TERMINAL_C2);

  Vcomp1 = int(VB2 * 10);
  Vcomp2 = int(VC2 * 10);

  //probamos terminal B GATE

  if (Vcomp1 == 0 & Vcomp2 != 0)
  { //si se cumple posible terminal B(2) GATE

    Vdif1 = VA2 - VC2;

    if (Vdif1 >= 0.5)
    { // si se cumple tenemos las posiciones

      Serial.println("NMOS: terminal 1 SOURCE, terminal 2 GATE, terminal 3  DRAIN");
      deteccion = "NMOS: terminal 1 SOURCE, terminal 2 GATE, terminal 3  DRAIN";
```

```
    NMOS = true;
    source = 1;
    gate = 2;
    drain = 3;
    ++ndetecciones;
  }
}

// probamos que el terminal C es la GATE

if (Vcomp2 == 0 & Vcomp1 != 0)
{ //si se cumple posible terminal C(3) GATE

  Vdif1 = VA2 - VB2;

  if (Vdif1 >= 0.5)
  { // si se cumple tenemos las posiciones

    Serial.println("NMOS: terminal 1 SOURCE, terminal 2 DRAIN, terminal 3 GATE");
    deteccion = "NMOS: terminal 1 SOURCE, terminal 2 DRAIN, terminal 3 GATE";

    NMOS = true;
    source = 1;
    gate = 3;
    drain = 2;
    ++ndetecciones;
  }
}

//////////////////////terminal B fuente (SOURCE)

POTset(0, 127);
POTset(1, 127);
POTset(2, 127);

DACset(0, 0);
DACset(1, Vb);
DACset(2, 0);

VA2 = mideTerminal(TERMINAL_A2);
VB2 = mideTerminal(TERMINAL_B2);
VC2 = mideTerminal(TERMINAL_C2);

Vcomp1 = int(VA2 * 10);
Vcomp2 = int(VC2 * 10);

// probamos si el terminal A es la GATE

if (Vcomp1 == 0 & Vcomp2 != 0)
{ //si se cumple posible terminal A(1) GATE
```

```
  Vdif1 = VB2 - VC2;

  if (Vdif1 > 0.5)
  { // si se cumple tenemos las posiciones

    Serial.println("NMOS: terminal 1 GATE, terminal 2 SOURCE, terminal 3  DRAIN");
    deteccion = "NMOS: terminal 1 GATE, terminal 2 SOURCE, terminal 3  DRAIN";
    NMOS = true;
    source = 2;
    gate = 1;
    drain = 3;
    ++ndetecciones;
  }
}

// probamos si el terminal C es la GATE

if (Vcomp2 == 0 & Vcomp1 != 0)
{ //si se cumple posible terminal C(3) GATE

  Vdif1 = VB2 - VA2;

  if (Vdif1 >= 0.5)
  { // si se cumple tenemos las posiciones

    Serial.println("NMOS: terminal 1 DRAIN, terminal 2 SOURCE, terminal 3 GATE");
    deteccion = "NMOS: terminal 1 DRAIN, terminal 2 SOURCE, terminal 3 GATE";
    NMOS = true;
    source = 2;
    gate = 3;
    drain = 1;
    ++ndetecciones;
  }
}

//////////////////////terminal C fuente (SOURCE)

POTset(0, 127);
POTset(1, 127);
POTset(2, 127);

DACset(0, 0);
DACset(1, 0);
DACset(2, Vb);

VA2 = mideTerminal(TERMINAL_A2);
VB2 = mideTerminal(TERMINAL_B2);
VC2 = mideTerminal(TERMINAL_C2);

Vcomp1 = int(VA2 * 10);
```

```
  Vcomp2 = int(VB2 * 10);

  // probamos si el terminal A es la GATE

  if (Vcomp1 == 0 & Vcomp2 != 0)
  { //si se cumple posible terminal A(1) GATE

    Vdif1 = VC2 - VB2;

    if (Vdif1 >= 0.5)
    { // si se cumple tenemos las posiciones

      Serial.println("NMOS: terminal 1 GATE, terminal 2 DRAIN, terminal 3 SOURCE");
      deteccion = "NMOS: terminal 1 GATE, terminal 2 DRAIN, terminal 3 SOURCE";

      NMOS = true;
      source = 3;
      gate = 1;
      drain = 2;
      ++ndetecciones;
    }
  }

  //probamos si el terminal B es la GATE

  if (Vcomp2 == 0 & Vcomp1 != 0)
  { //si se cumple posible terminal A(1) GATE

    Vdif1 = VC2 - VA2;

    if (Vdif1 >= 0.5)
    { // si se cumple tenemos las posiciones

      Serial.println("NMOS: terminal 1 DRAIN, terminal 2 GATE, terminal 3 SOURCE");
      deteccion = "NMOS: terminal 1 DRAIN, terminal 2 GATE, terminal 3 SOURCE";

      NMOS = true;
      source = 3;
      gate = 2;
      drain = 1;
      ++ndetecciones;
    }
  }
}

////////////////////////////////////////////////////////////////////// deteccion PMOS ////////////////////////////////////////
////////////////////////////////////////////////////////////////////// deteccion PMOS ////////////////////////////////////////

if (i == 1)
{ //en los bjt PNP tambien detecta 1
```

```
  POTset(0, 127);
  POTset(1, 127);
  POTset(2, 127);

  // terminal 1 y 2  ;candidato a fuente C

  DACset(0, 1950);
  DACset(1, 1950);
  DACset(2, 0);

  VA1 = mideTerminal(TERMINAL_A1);
  VA2 = mideTerminal(TERMINAL_A2);
  VB1 = mideTerminal(TERMINAL_B1);
  VB2 = mideTerminal(TERMINAL_B2);
  VC2 = mideTerminal(TERMINAL_C2);

  Vcomp1 = int(VC2 * 10); //si Vcomp1 no es 0 es que Vc2 es la fuente SOURCE

  if (Vcomp1 != 0)
  {

    Vdif1 = VA2 - VC2; //la diferencia mayor tiene que estar entre la puerta y la fuente
    Vdif2 = VB2 - VC2; //la diferencia menor tiene que estar el drenador y la fuente tiene que ser del orden de 0,6

    if (Vdif1 > Vdif2)
    {

      // mediante los siguentes calculos impedimos que se detecte como un BJT PNP
      //si el terminal A es la puerta VA1 tiene que ser practicamente igual a VA2 puesto que no circula casi corriente por la puerta
      Vcomp1 = int(VA1 * 10);
      Vcomp2 = int(VA2 * 10);
      if (Vcomp1 == Vcomp2)
      {
        DACset(0, 0);
        DACset(1, 2000);
        DACset(2, 0);

        VA2 = mideTerminal(TERMINAL_A2);
        Vcomp3 = int(VA2 * 10);
        if (Vcomp3 == 0)
        { //compruebo que no es un jfetP

          // aplico tension solo a la fuente
          DACset(0, 0);
          DACset(1, 0);
          DACset(2, 3000);

          // mido tension drenador
          VB2 = mideTerminal(TERMINAL_B2);
```

```
          Vcomp1 = int(VB2 * 10);

          // aplico tension tambien a la puerta
          DACset(0, 3000);
          DACset(1, 0);
          DACset(2, 3000);

          //vuelvo a medir el drenador
          VB2 = mideTerminal(TERMINAL_B2);
          Vcomp2 = int(VB2 * 10);

          if (Vcomp1 != Vcomp2)
          { // si no es un diodo zener la tension habra cambiado al aplicar tension en la gate
            Serial.println("PMOS: terminal 1 GATE, terminal 2 DRAIN, terminal 3 SOURCE");
            deteccion = "PMOS: terminal 1 GATE, terminal 2 DRAIN, terminal 3 SOURCE";

            PMOS = true;
            source = 3;
            gate = 1;
            drain = 2;
            ++ndetecciones;
          }
        }
      }
    }
    else
    {

      // mediante los siguentes calculos impedimos que se detecte como un BJT PNP
      //si el terminal B es la puerta VB1 tiene que ser practicamente igual a VB2 puesto que no circula casi
corriente por la puerta
      Vcomp1 = int(VB1 * 10);
      Vcomp2 = int(VB2 * 10);
      if (Vcomp1 == Vcomp2)
      {
        DACset(0, 2000);
        DACset(1, 0);
        DACset(2, 0);

        VB2 = mideTerminal(TERMINAL_B2);
        Vcomp3 = int(VB2 * 10);
        if (Vcomp3 == 0)
        { //compruebo que no es un jfetP

          // aplico tension solo a la fuente
          DACset(0, 0);
          DACset(1, 0);
          DACset(2, 3000);

          // mido tension drenador
          VA2 = mideTerminal(TERMINAL_A2);
```

```
          Vcomp1 = int(VA2 * 10);

          // aplico tension tambien a la puerta
          DACset(0, 0);
          DACset(1, 3000);
          DACset(2, 3000);

          //vuelvo a medir el drenador
          VA2 = mideTerminal(TERMINAL_A2);
          Vcomp2 = int(VA2 * 10);

          if (Vcomp1 != Vcomp2)
          { // si no es un diodo zener la tension habra cambiado al aplicar tension en la gate

            Serial.println("PMOS: terminal 1 DRAIN, terminal 2 GATE, terminal 3 SOURCE");
            deteccion = "PMOS: terminal 1 DRAIN, terminal 2 GATE, terminal 3 SOURCE";

            PMOS = true;
            source = 3;
            gate = 2;
            drain = 1;
            ++ndetecciones;
          }
        }
      }
    }
  }

  // terminal 1 y 3  ; cadidate a fuente B
  DACset(0, 1950);
  DACset(1, 0);
  DACset(2, 1950);

  VA1 = mideTerminal(TERMINAL_A1);
  VA2 = mideTerminal(TERMINAL_A2);
  VB2 = mideTerminal(TERMINAL_B2);
  VC1 = mideTerminal(TERMINAL_C1);
  VC2 = mideTerminal(TERMINAL_C2);

  Vcomp1 = int(VB2 * 10); //si Vcomp1 no es 0 es que VB2 es la fuente SOURCE

  if (Vcomp1 != 0)
  {

    Vdif1 = VA2 - VB2; //la diferencia mayor tiene que estar entre la puerta y la fuente
    Vdif2 = VC2 - VB2; //la diferencia menor tiene que estar el drenador y la fuente tiene que ser del orden
de 0,6

    if (Vdif1 > Vdif2)
    {
```

```
      // mediante los siguentes calculos impedimos que se detecte como un BJT PNP
      //si el terminal A es la puerta VA1 tiene que ser practicamente igual a VA2 puesto que no circula casi
corriente por la puerta
      Vcomp1 = int(VA1 * 10);
      Vcomp2 = int(VA2 * 10);
      if (Vcomp1 == Vcomp2)
      {
        DACset(0, 0);
        DACset(1, 0);
        DACset(2, 2000);

        VA2 = mideTerminal(TERMINAL_A2);
        Vcomp3 = int(VA2 * 10);
        if (Vcomp3 == 0)
        { //compruebo que no es un jfetP

          // aplico tension solo a la fuente
          DACset(0, 0);
          DACset(1, 3000);
          DACset(2, 0);

          // mido tension drenador
          VC2 = mideTerminal(TERMINAL_C2);
          Vcomp1 = int(VC2 * 10);

          // aplico tension tambien a la puerta
          DACset(0, 3000);
          DACset(1, 3000);
          DACset(2, 0);

          //vuelvo a medir el drenador
          VC2 = mideTerminal(TERMINAL_C2);
          Vcomp2 = int(VC2 * 10);

          if (Vcomp1 != Vcomp2)
          { // si no es un diodo zener la tension habra cambiado al aplicar tension en la gate

            Serial.println("PMOS: terminal 1 GATE, terminal 2 SOURCE, terminal 3 DRAIN");
            deteccion = "PMOS: terminal 1 GATE, terminal 2 SOURCE, terminal 3 DRAIN";

            PMOS = true;
            source = 2;
            gate = 1;
            drain = 3;
            ++ndetecciones;
          }
        }
      }
    }
    else
    {
```

```
      //si el terminal C es la puerta VC1 tiene que ser practicamente igual a VC2 puesto que no circula casi
corriente por la puerta
      Vcomp1 = int(VC1 * 10);
      Vcomp2 = int(VC2 * 10);
      if (Vcomp1 == Vcomp2)
      {
        DACset(0, 2000);
        DACset(1, 0);
        DACset(2, 0);

        VC2 = mideTerminal(TERMINAL_C2);
        Vcomp3 = int(VC2 * 10);
        if (Vcomp3 == 0)
        { //compruebo que no es un jfetP

          // aplico tension solo a la fuente
          DACset(0, 0);
          DACset(1, 3000);
          DACset(2, 0);

          // mido tension drenador
          VA2 = mideTerminal(TERMINAL_A2);
          Vcomp1 = int(VA2 * 10);

          // aplico tension tambien a la puerta
          DACset(0, 0);
          DACset(1, 3000);
          DACset(2, 3000);

          //vuelvo a medir el drenador
          VA2 = mideTerminal(TERMINAL_A2);
          Vcomp2 = int(VA2 * 10);

          if (Vcomp1 != Vcomp2)
          { // si no es un diodo zener la tension habra cambiado al aplicar tension en la gate
            Serial.println("PMOS: terminal 1 DRAIN, terminal 2 SOURCE, terminal 3 GATE");
            deteccion = "PMOS: terminal 1 DRAIN, terminal 2 SOURCE, terminal 3 GATE";

            PMOS = true;
            source = 2;
            gate = 3;
            drain = 1;
            ++ndetecciones;
          }
        }
      }
    }
  }

  // terminal 2 y 3   ; cadidato  a fuente A
```

```
  DACset(0, 0);
  DACset(1, 1950);
  DACset(2, 1950);

  VA2 = mideTerminal(TERMINAL_A2);
  VB1 = mideTerminal(TERMINAL_B1);
  VB2 = mideTerminal(TERMINAL_B2);
  VC1 = mideTerminal(TERMINAL_C1);
  VC2 = mideTerminal(TERMINAL_C2);

  Vcomp1 = int(VA2 * 10); //si Vcomp1 no es 0 es que Vc2 es la fuente SOURCE

  if (Vcomp1 != 0)
  {

    Vdif1 = VB2 - VA2; //la diferencia mayor tiene que estar entre la puerta y la fuente
    Vdif2 = VC2 - VA2; //la diferencia menor tiene que estar el drenador y la fuente tiene que ser del orden de 0,6

    if (Vdif1 > Vdif2)
    {

      // mediante los siguentes calculos impedimos que se detecte como un BJT PNP
      //si el terminal B es la puerta VB1 tiene que ser practicamente igual a VB2 puesto que no circula casi corriente por la puerta
      Vcomp1 = int(VB1 * 10);
      Vcomp2 = int(VB2 * 10);
      if (Vcomp1 == Vcomp2)
      {
        DACset(0, 0);
        DACset(1, 0);
        DACset(2, 2000);

        VB2 = mideTerminal(TERMINAL_B2);
        Vcomp3 = int(VB2 * 10);
        if (Vcomp3 == 0)
        { //compruebo que no es un jfetP

          // aplico tension solo a la fuente
          DACset(0, 3000);
          DACset(1, 0);
          DACset(2, 0);

          // mido tension drenador
          VC2 = mideTerminal(TERMINAL_C2);
          Vcomp1 = int(VC2 * 10);

          // aplico tension tambien a la puerta
          DACset(0, 3000);
          DACset(1, 3000);
          DACset(2, 0);
```

```
        //vuelvo a medir el drenador
        VC2 = mideTerminal(TERMINAL_C2);
        Vcomp2 = int(VC2 * 10);

        if (Vcomp1 != Vcomp2)
        { // si no es un diodo zener la tension habra cambiado al aplicar tension en la gate

          Serial.println("PMOS: terminal 1 SOURCE, terminal 2 GATE, terminal 3 DRAIN");
          deteccion = "PMOS: terminal 1 SOURCE, terminal 2 GATE, terminal 3 DRAIN";

          PMOS = true;
          source = 1;
          gate = 2;
          drain = 3;
          ++ndetecciones;
        }
      }
    }
    else
    {

      // mediante los siguentes calculos impedimos que se detecte como un BJT PNP
      //si el terminal C es la puerta VC1 tiene que ser practicamente igual a VC2 puesto que no circula casi
corriente por la puerta
      Vcomp1 = int(VC1 * 10);
      Vcomp2 = int(VC2 * 10);
      if (Vcomp1 == Vcomp2)
      {

        DACset(0, 0);
        DACset(1, 2000);
        DACset(2, 0);

        VC2 = mideTerminal(TERMINAL_C2);
        Vcomp3 = int(VC2 * 10);
        if (Vcomp3 == 0)
        { //compruebo que no es un jfetP

          // aplico tension solo a la fuente
          DACset(0, 3000);
          DACset(1, 0);
          DACset(2, 0);

          // mido tension drenador
          VB2 = mideTerminal(TERMINAL_B2);
          Vcomp1 = int(VB2 * 10);

          // aplico tension tambien a la puerta
          DACset(0, 3000);
```

```
          DACset(1, 0);
          DACset(2, 3000);

          //vuelvo a medir el drenador
          VB2 = mideTerminal(TERMINAL_B2);
          Vcomp2 = int(VB2 * 10);

          if (Vcomp1 != Vcomp2)
          { // si no es un diodo zener la tension habra cambiado al aplicar tension en la gate

            Serial.println("PMOS: terminal 1 SOURCE, terminal 2 DRAIN, terminal 3 GATE");
            deteccion = "PMOS: terminal 1 SOURCE, terminal 2 DRAIN, terminal 3 GATE";

            PMOS = true;
            source = 1;
            gate = 3;
            drain = 2;
            ++ndetecciones;
          }
        }
      }
    }
  }
 }
}

////////////////////////////////////////////////////////////////////////////
///////                    Detectar  Bjt
////////////////////////////////////////////////////////////////////////////
void detectar_NPNbjt()
{

 ////////////////////////////////// detecto si es NPN ////////////////////////////////

 ////////////////////////////////////canal A base NPN

 POTset(0, 255); //minima R
 POTset(1, 242);
 POTset(2, 242);

 DACset(0, 2000);
 DACset(1, 0);
 DACset(2, 0);

 VA1 = mideTerminal(TERMINAL_A1);
 VA2 = mideTerminal(TERMINAL_A2);
 VB2 = mideTerminal(TERMINAL_B2);
 VC1 = mideTerminal(TERMINAL_C1);
 VC2 = mideTerminal(TERMINAL_C2);

 Vdif1 = VA2 - VB2;
```

```
 Vdif2 = VA2 - VC2;

 if (Vdif1 > 0.5 & Vdif2 > 0.5)
 {

  if (VB2 > 0.5 & VC2 > 0.5) //sabemos que es NPN y que la base es la A
  {
    DACset(0, 0);
    DACset(1, 2000);
    DACset(2, 0);

    VC2 = mideTerminal(TERMINAL_C2);
    Vcomp1 = int(VC2 * 10);

    if (Vcomp1 == 0)
    { ///hago esto para asegurarme de que no es un JFET tipo N y no conduce entre colector y drenador

      POTset(0, 127); // 100k en la base
      POTset(1, 242); //medimos la corriente de este terminal por eso le fijamos una R de aprox 10k
      POTset(2, 255);

      DACset(0, Vb);
      DACset(1, 4000);
      DACset(2, 0);

      Ib = measureI(TERMINAL_B1, TERMINAL_B2, 10);

      POTset(0, 127);
      POTset(1, 255);
      POTset(2, 242); //medimos la corriente de este terminal por eso le fijamos una R de aprox 10k

      DACset(0, Vb);
      DACset(1, 0);
      DACset(2, 4000);

      Ic = measureI(TERMINAL_C1, TERMINAL_C2, 10);

      // comparamos las corrientes la corriente de colector en un NPN tiene que ser mucho mayor ya que
la ganancia de colector es mucho mayor
      if (Ib > Ic)
      {
        Serial.println("BJT NPN: terminal 1 BASE, terminal 2 COLECTOR, terminal 3 EMISOR");
        deteccion = "BJT NPN: terminal 1 BASE, terminal 2 COLECTOR, terminal 3 EMISOR";

        NBJT = true;
        colector = 2;
        base = 1;
        emisor = 3;
        ++ndetecciones;
      }
      else
```

```
    {
      Serial.println("BJT NPN: terminal 1 BASE, terminal 2 EMISOR, terminal 3 COLECTOR");
      deteccion = "BJT NPN: terminal 1 BASE, terminal 2 EMISOR, terminal 3 COLECTOR";

      NBJT = true;
      colector = 3;
      base = 1;
      emisor = 2;
      ++ndetecciones;
    }
   }
  }
}

//////////////////////canal B base NPN
//////////////////////canal B base NPN

POTset(0, 242);
POTset(1, 255);
POTset(2, 242);

DACset(0, 0);
DACset(1, Vb);
DACset(2, 0);

VA2 = mideTerminal(TERMINAL_A2);
VB1 = mideTerminal(TERMINAL_B1);
VB2 = mideTerminal(TERMINAL_B2);
VC2 = mideTerminal(TERMINAL_C2);

Vdif1 = VB2 - VA2;
Vdif2 = VB2 - VC2;

if (Vdif1 > 0.5 & Vdif2 > 0.5)
{

 if (VA2 > 0.5 & VC2 > 0.5) //sabemos que es NPN y que la base es la B
 {

   DACset(0, 2000);
   DACset(1, 0);
   DACset(2, 0);

   VC2 = mideTerminal(TERMINAL_C2);
   Vcomp1 = int(VC2 * 10);

   if (Vcomp1 == 0)
   { ///hago esto para asegurarme de que no es un JFET tipo N y no conduce entre colector y drenador

     POTset(0, 242);
     POTset(1, 127);
```

```
      POTset(2, 255);
      DACset(0, 4000);
      DACset(1, Vb);
      DACset(2, 0);

      Ia = measureI(TERMINAL_A1, TERMINAL_A2, 10);

      POTset(0, 255);
      POTset(1, 127);
      POTset(2, 242);
      DACset(0, 0);
      DACset(1, Vb);
      DACset(2, 4000);

      Ic = measureI(TERMINAL_C1, TERMINAL_C2, 10);

      // comparamos las corrientes la corriente de colector en un NPN tiene que ser mucho mayor ya que la ganancia de colector es mucho mayor
      if (Ia > Ic)
      {
        Serial.println("BJT NPN: terminal 1 COLECTOR, terminal 2 BASE, terminal 3 EMISOR");
        deteccion = "BJT NPN: terminal 1 COLECTOR, terminal 2 BASE, terminal 3 EMISOR";

        NBJT = true;
        colector = 1;
        base = 2;
        emisor = 3;
        ++ndetecciones;
      }
      else
      {
        Serial.println("BJT NPN: terminal 1 EMISOR, terminal 2 BASE, terminal 3 COELCTOR");
        deteccion = "BJT NPN: terminal 1 EMISOR, terminal 2 BASE, terminal 3 COELCTOR";

        NBJT = true;
        colector = 3;
        base = 2;
        emisor = 1;
        ++ndetecciones;
      }
    }
  }
}

////////////////////////////////////canal C base NPN
////////////////////////////////////canal C base NPN

POTset(0, 242); //minima R
POTset(1, 242);
POTset(2, 255);
```

```
 DACset(0, 0);
 DACset(1, 0);
 DACset(2, Vb);

 VA2 = mideTerminal(TERMINAL_A2);
 VB2 = mideTerminal(TERMINAL_B2);
 VC1 = mideTerminal(TERMINAL_C1);
 VC2 = mideTerminal(TERMINAL_C2);

 Vdif1 = VC2 - VA2;
 Vdif2 = VC2 - VB2;

 if (Vdif1 > 0.5 & Vdif2 > 0.5)
 {

  if (VA2 > 0.5 & VB2 > 0.5) //sabemos que es NPN y que la base es la A
  {

    DACset(0, 2000);
    DACset(1, 0);
    DACset(2, 0);

    VB2 = mideTerminal(TERMINAL_B2);
    Vcomp1 = int(VB2 * 10);

    if (Vcomp1 == 0)
    { ///hago esto para asegurarme de que no es un JFET tipo N y no conduce entre colector y drenador

      POTset(0, 242);
      POTset(1, 255);
      POTset(2, 127);

      DACset(0, 4000);
      DACset(1, 0);
      DACset(2, Vb);

      Ia = measureI(TERMINAL_A1, TERMINAL_A2, 10);

      POTset(0, 127);
      POTset(1, 255);
      POTset(2, 242); //medimos la corriente de este terminal por eso le fijamos una R de aprox 10k

      DACset(0, 0);
      DACset(1, 4000);
      DACset(2, Vb);

      Ib = measureI(TERMINAL_B1, TERMINAL_B2, 10);

      // comparamos las corrientes la corriente de colector en un NPN tiene que ser mucho mayor ya que
la ganancia de colector es mucho mayor
      if (Ia > Ib)
```

```
      {
        Serial.println("BJT NPN: terminal 1 COLECTOR, terminal 2 EMISOR, terminal 3 BASE");
        deteccion = "BJT NPN: terminal 1 COLECTOR, terminal 2 EMISOR, terminal 3 BASE";

        NBJT = true;
        colector = 1;
        base = 3;
        emisor = 2;
        ++ndetecciones;
      }
      else
      {
        Serial.println("BJT NPN: terminal 1 EMISOR, terminal 2 COLECTOR, terminal 3 BASE");
        deteccion = "BJT NPN: terminal 1 EMISOR, terminal 2 COLECTOR, terminal 3 BASE";

        NBJT = true;
        colector = 2;
        base = 3;
        emisor = 1;
        ++ndetecciones;
      }
    }
  }
 }
}
void detectar_PNPbjt()
{

 ////////////////////////////////////////////////////////////////////// detecto si es  PNP ////////////////////////////////////
 ////////////////////////////////////////////////////////////////////// detecto si es  PNP ////////////////////////////////////

 ////////////////////////////////canal A base PNP
 ////////////////////////////////canal A base PNP

 POTset(0, 255); //minima R
 POTset(1, 247);
 POTset(2, 247);

 DACset(0, 0);
 DACset(1, 1300);
 DACset(2, 1300);

 VA2 = mideTerminal(TERMINAL_A2);
 VB2 = mideTerminal(TERMINAL_B2);
 VC2 = mideTerminal(TERMINAL_C2);

 Vdif1 = VA2 - VB2;
 Vdif2 = VA2 - VC2;

 if (Vdif1 < -0.5 & Vdif2 < -0.5)
 {
```

```
  if (VB2 < 1.5 & VC2 < 1.5) //sabemos que es PNP y que la base es la A
  {
    POTset(0, 127); // 100k en la base
    POTset(1, 242); //medimos la corriente de este terminal por eso le fijamos una R de aprox 10k
    POTset(2, 255);

    DACset(0, Vb);
    DACset(1, 0);    //supuesto colector
    DACset(2, 4000); //supuesto emisor

    Ib = measureI(TERMINAL_B1, TERMINAL_B2, 10);

    POTset(0, 127);
    POTset(1, 255);
    POTset(2, 242); //medimos la corriente de este terminal por eso le fijamos una R de aprox 10k

    DACset(0, Vb);
    DACset(1, 4000); //supuesto emisor
    DACset(2, 0);    //supuesto colector

    Ic = measureI(TERMINAL_C1, TERMINAL_C2, 10);

    // comparamos las corrientes, la corriente de colector en un PNP tiene que ser mucho mayor(en valor absoluto) ya que la ganancia de colector es mucho mayor
    if (Ib < Ic)
    {
      POTset(0, 255);
      POTset(1, 255);
      POTset(2, 255);

      DACset(0, 2000);
      DACset(1, 0);
      DACset(2, 3300);

      Vdif1 = mideTerminal(TERMINAL_A2) - mideTerminal(TERMINAL_A1);
      Vcomp2 = int(Vdif1 * 100);
      if (Vcomp2 != 0)
      { // compruebo que no es un jfetP

        Serial.println("PNP: terminal 1 BASE, terminal 2 COLECTOR, terminal 3 EMISOR");
        deteccion = "PNP: terminal 1 BASE, terminal 2 COLECTOR, terminal 3 EMISOR";

        PBJT = true;
        colector = 2;
        base = 1;
        emisor = 3;
        ++ndetecciones;
      }
    }
    else
```

```
    {

      DACset(0, 2000);
      DACset(1, 0);
      DACset(2, 3300);

      Vdif1 = mideTerminal(TERMINAL_A2) - mideTerminal(TERMINAL_A1);
      Vcomp2 = int(Vdif1 * 100);
      if (Vcomp2 != 0)
      { // compruebo que no es un jfetP cunado hay caida te tension en la resistencia de la base ya que
circula una corriente
        Serial.println("PNP: terminal 1 BASE, terminal 2 EMISOR, terminal 3 COLECTOR");
        deteccion = "PNP: terminal 1 BASE, terminal 2 EMISOR, terminal 3 COLECTOR";

        PBJT = true;
        colector = 3;
        base = 1;
        emisor = 2;
        ++ndetecciones;
      }
    }
  }
}

 ////////////////////////canal B base PNP
 ////////////////////////canal B base PNP

 POTset(0, 247);
 POTset(1, 255);
 POTset(2, 247);

 DACset(0, 1300);
 DACset(1, 0);
 DACset(2, 1300);

 VA2 = mideTerminal(TERMINAL_A2);
 VB2 = mideTerminal(TERMINAL_B2);
 VC2 = mideTerminal(TERMINAL_C2);

 Vdif1 = VB2 - VA2;
 Vdif2 = VB2 - VC2;

 if (Vdif1 < -0.5 & Vdif2 < -0.5)
 {

  if (VA2 < 1.5 & VC2 < 1.5) //sabemos que es PNP y que la base es la A
  {
    POTset(0, 242); //medimos la corriente de este terminal por eso le fijamos una R de aprox 10k
    POTset(1, 127); // 100k en la base
    POTset(2, 255);
```

```
    DACset(0, 0); //supuesto colector
    DACset(1, Vb);
    DACset(2, 4000); //supuesto emisor

    Ia = measureI(TERMINAL_A1, TERMINAL_A2, 10);

    POTset(0, 255);
    POTset(1, 127);
    POTset(2, 242); //medimos la corriente de este terminal por eso le fijamos una R de aprox 10k

    DACset(0, 4000); //supuesto emisor
    DACset(1, Vb);
    DACset(2, 0); //supuesto colector

    Ic = measureI(TERMINAL_C1, TERMINAL_C2, 10);

    // comparamos las corrientes, la corriente de colector en un PNP tiene que ser mucho mayor(en valor absoluto) ya que la ganancia de colector es mucho mayor
    if (Ia < Ic)
    {
      POTset(0, 255);
      POTset(1, 255);
      POTset(2, 255);

      DACset(0, 0);
      DACset(1, 2000);
      DACset(2, 3300);

      Vdif1 = mideTerminal(TERMINAL_B2) - mideTerminal(TERMINAL_B1);
      Vcomp2 = int(Vdif1 * 100);
      if (Vcomp2 != 0)
      { // compruebo que no es un jfetP

        Serial.println("PNP: terminal 1 COLECTOR, terminal 2 BASE , terminal 3 EMISOR");
        deteccion = "PNP: terminal 1 COLECTOR, terminal 2 BASE , terminal 3 EMISOR";

        PBJT = true;
        colector = 1;
        base = 2;
        emisor = 3;
        ++ndetecciones;
      }
    }
    else
    {

      DACset(0, 0);
      DACset(1, 2000);
      DACset(2, 3300);

      Vdif1 = mideTerminal(TERMINAL_B2) - mideTerminal(TERMINAL_B1);
```

```
      Vcomp2 = int(Vdif1 * 100);
      if (Vcomp2 != 0)
      { // compruebo que no es un jfetP

        Serial.println("PNP: terminal 1 EMISOR, terminal 2 BASE, terminal 3 COLECTOR");
        deteccion = "PNP: terminal 1 EMISOR, terminal 2 BASE, terminal 3 COLECTOR";

        PBJT = true;
        colector = 3;
        base = 2;
        emisor = 1;
        ++ndetecciones;
      }
    }
  }
}

//////////////////////canal C base PNP
//////////////////////canal C base PNP

POTset(0, 242); //minima R
POTset(1, 242);
POTset(2, 255);

DACset(0, 1300);
DACset(1, 1300);
DACset(2, 0);

VA2 = mideTerminal(TERMINAL_A2);
VB2 = mideTerminal(TERMINAL_B2);
VC2 = mideTerminal(TERMINAL_C2);

Vdif1 = VC2 - VA2;
Vdif2 = VC2 - VB2;

if (Vdif1 < -0.5 & Vdif2 < -0.5)
{

  if (VA2 < 1.5 & VB2 < 1.5) //sabemos que es PNP y que la base es la A
  {
    POTset(0, 242); //medimos la corriente de este terminal por eso le fijamos una R de aprox 10k
    POTset(1, 255);
    POTset(2, 127); // 100k en la base

    DACset(0, 0); //supuesto colector
    DACset(1, 4000);
    DACset(2, Vb); //supuesto emisor

    Ia = measureI(TERMINAL_A1, TERMINAL_A2, 10);

    POTset(0, 242);
```

```
    POTset(1, 255);
    POTset(2, 127); //medimos la corriente de este terminal por eso le fijamos una R de aprox 10k

    DACset(0, 4000); //supuesto emisor
    DACset(1, 0);
    DACset(2, Vb); //supuesto colector

    Ib = measureI(TERMINAL_B1, TERMINAL_B2, 10);

    // comparamos las corrientes, la corriente de colector en un PNP tiene que ser mucho mayor(en valor absoluto) ya que la ganancia de colector es mucho mayor
    if (Ia < Ib)
    {
      POTset(0, 255);
      POTset(1, 255);
      POTset(2, 255);

      DACset(0, 0);
      DACset(1, 3300);
      DACset(2, 2000);

      Vdif1 = mideTerminal(TERMINAL_C2) - mideTerminal(TERMINAL_C1);
      Vcomp2 = int(Vdif1 * 100);
      if (Vcomp2 != 0)
      { // compruebo que no es un jfetP

        Serial.println("PNP: terminal 1 COLECTOR, terminal 2 EMISOR , terminal 3 BASE");
        deteccion = "PNP: terminal 1 COLECTOR, terminal 2 EMISOR , terminal 3 BASE";

        PBJT = true;
        colector = 1;
        base = 3;
        emisor = 2;
        ++ndetecciones;
      }
    }
    else
    {

      DACset(0, 0);
      DACset(1, 3300);
      DACset(2, 2000);

      Vdif1 = mideTerminal(TERMINAL_C2) - mideTerminal(TERMINAL_C1);
      Vcomp2 = int(Vdif1 * 100);
      if (Vcomp2 != 0)
      { // compruebo que no es un jfetP
        Serial.println("PNP: terminal 1 EMISOR, terminal 2 COLECTOR , terminal 3 BASE");
        deteccion = "PNP: terminal 1 EMISOR, terminal 2 COLECTOR , terminal 3 BASE";

        PBJT = true;
```

```
        colector = 2;
        base = 3;
        emisor = 1;
        ++ndetecciones;
       }
      }
    }
  }
}
////////////////////////////////////////////////////////////////////////////
///////                    Detectar  JFET
////////////////////////////////////////////////////////////////////////////

void detectar_PJFET()
{
  POTset(0, 255);
  POTset(1, 255);
  POTset(2, 255);

  DACset(0, 2000);
  DACset(1, 0); // si aplico esta tension y no es la pue
  DACset(2, 0);

  VA2 = mideTerminal(TERMINAL_A2);
  VB2 = mideTerminal(TERMINAL_B2);
  VC2 = mideTerminal(TERMINAL_C2);

  if (VB2 > 0.5 || VC2 > 0.5)
  {

    Vcomp1 = int(VA2 * 10);
    Vcomp2 = int(VB2 * 10);

    if ((Vcomp1 - 1) <= Vcomp2) //
    {
      Vdif1 = VA2 - VC2;
      if (Vdif1 < 1.0 && Vdif1 > 0.4) //compruebo la diferencia entre el drenador y la gate debida a la unión
PN
      {

        DACset(0, 0);
        DACset(1, 3300);
        DACset(2, 2000);

        Vdif1 = mideTerminal(TERMINAL_C2) - mideTerminal(TERMINAL_C1);
        Vcomp2 = int(Vdif1 * 100);
        if (Vcomp2 == 0)
        {

          Serial.println("JFET tipo P: terminal 1 DRAIN, terminal 2 SOURCE, terminal 3 GATE");
          deteccion = "JFET tipo P: terminal 1 DRAIN, terminal 2 SOURCE, terminal 3 GATE";
```

```
        PJFET = true;
        source = 2;
        gate = 3;
        drain = 1;
        ++ndetecciones;
      }
    }
  }

  Vcomp2 = int(VC2 * 10);

  if ((Vcomp1 - 1) <= Vcomp2) //
  {

    Vdif1 = VA2 - VB2;
    if (Vdif1 < 1.0 && Vdif1 > 0.4) //compruebo la diferencia entre el drenador y la gate debida a la unión
PN
    {

      DACset(0, 0);
      DACset(1, 2000);
      DACset(2, 3300);

      Vdif1 = mideTerminal(TERMINAL_B2) - mideTerminal(TERMINAL_B1);
      Vcomp2 = int(Vdif1 * 100);
      if (Vcomp2 == 0)
      {
        Serial.println("JFET tipo P: terminal 1 DRAIN, terminal 2 GATE, terminal 3 SOURCE");
        deteccion = "JFET tipo P: terminal 1 DRAIN, terminal 2 GATE, terminal 3 SOURCE";

        PJFET = true;
        source = 3;
        gate = 2;
        drain = 1;
        ++ndetecciones;
      }
    }
  }
 }

 DACset(0, 0);
 DACset(1, 2000); // si aplico esta tension y no es la pue
 DACset(2, 0);

 VA2 = mideTerminal(TERMINAL_A2);
 VB2 = mideTerminal(TERMINAL_B2);
 VC2 = mideTerminal(TERMINAL_C2);

 if (VA2 > 0.5 || VC2 > 0.5)
 {
```

```
  Vcomp1 = int(VB2 * 10);
  Vcomp2 = int(VC2 * 10);

  if ((Vcomp1 - 1) <= Vcomp2) //
  {

    Vdif1 = VB2 - VA2;
    if (Vdif1<1.0 & Vdif1> 0.4) //compruebo la diferencia entre el drenador y la gate debida a la unión PN
    {

      DACset(0, 2000);
      DACset(1, 0);
      DACset(2, 3300);

      Vdif1 = mideTerminal(TERMINAL_A2) - mideTerminal(TERMINAL_A1);
      Vcomp2 = int(Vdif1 * 100);
      if (Vcomp2 == 0)
      {
        Serial.println("JFET tipo P: terminal 1 GATE, terminal 2 DRAIN, terminal 3 SOURCE");
        deteccion = "JFET tipo P: terminal 1 GATE, terminal 2 DRAIN, terminal 3 SOURCE";

        PJFET = true;
        source = 3;
        gate = 1;
        drain = 2;
        ++ndetecciones;
      }
    }
  }
 }
}


void detectar_NJFET()
{

 POTset(0, 127);
 POTset(1, 127);
 POTset(2, 127);

 DACset(0, 0);
 DACset(1, 0);
 DACset(2, 0);

 /// JFET tipo N  en el canal se comporta como una resistencia, esta R depende de Vgs canal de
deplexion
 /// no distingo drenador de fuente el cal conduce pero hay una R, condue al no ser que VGS<<0
 /// entre la gate y el canal unión PN
 /// quieza podría detectarlo como al mosfet N, pero que los terminales los tecte como hice con el MOS
tipo P
```

```
 // estategia:
 // voy a comprobar los terminales a la vez que determino si es un jfet
 // 1 la misma que en BJT NPN se aplico una tensión de 2v a la base D y S tendrán aprox 1,4 dif de 0,7 aprox

 ////////////////////////////////////////////////////////////////////
 ////////////// en caso de JFET el terminal A es la GATE  ///////////////////////
 ////////////////////////////////////////////////////////////////////

 DACset(0, 2000); // si aplico una tensión de 4.9 v al canal A tengo 2 v en el terminal de la gate
 DACset(1, 0);    // D y S a 0 v
 DACset(2, 0);

 VA2 = mideTerminal(TERMINAL_A2);
 VB2 = mideTerminal(TERMINAL_B2);
 VC2 = mideTerminal(TERMINAL_C2);

 Vdif1 = VA2 - VB2;
 Vdif2 = VA2 - VC2;

 if (Vdif1 > 0.5 & Vdif2 > 0.5)
 { // aqui doy por hecho que si se cumple las diferencia de Vdif1 respecto a Vdif2 son iguales

   if (VB2 > 0.5 & VC2 > 0.5) // PUEDE SER UN BJT NPN O UN JFET TIPO N
   {
     DACset(0, 0);
     DACset(1, 2000);
     DACset(2, 0);

     VC2 = mideTerminal(TERMINAL_C2);

     if (VC2 > 0.2)
     { // si tenemos tensión en ese canas el que conduce por lo tanto se trata de un jfet tipo y no de un bjt tipo N

       Serial.println("JFET tipo N: terminal 1 GATE, terminal 2 SOURCE, terminal 3 DRAIN");
       deteccion = "JFET tipo N: terminal 1 GATE, terminal 2 SOURCE, terminal 3 DRAIN";

       NJFET = true;
       source = 2;
       gate = 1;
       drain = 3;
       ++ndetecciones;
     }
   }
 }
 ////////////////////////////////////////////////////////////////////
 ////////////// en caso de JFET el terminal B es la GATE  ///////////////////////
 ////////////////////////////////////////////////////////////////////
```

```
 DACset(0, 0);    // si aplico una tensión de 4.9 v al canal A tengo 2 v en el terminal de la gate
 DACset(1, 2000); // D y S a 0 v
 DACset(2, 0);

 VA2 = mideTerminal(TERMINAL_A2);
 VB2 = mideTerminal(TERMINAL_B2);
 VC2 = mideTerminal(TERMINAL_C2);

 Vdif1 = VB2 - VA2;
 Vdif2 = VB2 - VC2;

 if (Vdif1 > 0.5 & Vdif2 > 0.5)
 { // aqui doy por hecho que si se cumple las diferencia de Vdif1 respecto a Vdif2 son iguales

   if (VA2 > 0.5 & VC2 > 0.5) // PUEDE SER UN BJT NPN O UN JFET TIPO N
   {
     DACset(0, 2000);
     DACset(1, 0);
     DACset(2, 0);

     VC2 = mideTerminal(TERMINAL_C2);

     if (VC2 > 0.2)
     { // si tenemos tensión en ese canas el que conduce por lo tanto se trata de un jfet tipo y no de un bjt
tipo N

       Serial.println("JFET tipo N: terminal 1 SOURCE, terminal 2 GATE, terminal 3 DRAIN");
       deteccion = "JFET tipo N: terminal 1 SOURCE, terminal 2 GATE, terminal 3 DRAIN";

       NJFET = true;
       source = 1;
       gate = 2;
       drain = 3;
       ++ndetecciones;
     }
   }
 }

 ////////////////////////////////////////////////////////////////////////
 ////////////// en caso de JFET el terminal C es la GATE  ///////////////////////
 ////////////////////////////////////////////////////////////////////////

 DACset(0, 0); // si aplico una tensión de 4.9 v al canal A tengo 2 v en el terminal de la gate
 DACset(1, 0); // D y S a 0 v
 DACset(2, 2000);

 VA2 = mideTerminal(TERMINAL_A2);
 VB2 = mideTerminal(TERMINAL_B2);
 VC2 = mideTerminal(TERMINAL_C2);

 Vdif1 = VC2 - VA2;
```

```
  Vdif2 = VC2 - VB2;

  if (Vdif1 > 0.5 & Vdif2 > 0.5)
  { // aqui doy por hecho que si se cumple las diferencia de Vdif1 respecto a Vdif2 son iguales

    if (VA2 > 0.5 & VB2 > 0.5) // PUEDE SER UN BJT NPN O UN JFET TIPO N
    {
      DACset(0, 2000);
      DACset(1, 0);
      DACset(2, 0);

      VB2 = mideTerminal(TERMINAL_B2);

      if (VC2 > 0.2)
      { // si tenemos tensión en ese canas el que conduce por lo tanto se trata de un jfet tipo y no de un bjt
tipo N

        Serial.println("JFET tipo N: terminal 1 SOURCE, terminal 2 DRAIN, terminal 3 GATE");
        deteccion = "JFET tipo N: terminal 1 SOURCE, terminal 2 DRAIN, terminal 3 GATE";

        NJFET = true;
        source = 1;
        gate = 3;
        drain = 2;
        ++ndetecciones;
      }
      ////////////// podría comprovar que la caida es igual en ambos sentidos
    }
  }
}

//////////////////////////////////////////////////////////////////////////////
///////                  DAC set   saca un valor por el DAC
//////////////////////////////////////////////////////////////////////////////

void DACset(byte iDAC, int valorDAC)
{

  Wire.beginTransmission(direcDAC); // transmito la direccion
  Wire.write(trama1DAC | ((dacSelect[iDAC]) << 1));
  Wire.write(trama2DAC | ((valorDAC >> 8) & 0b1111));
  Wire.write((valorDAC & 0b11111111));
  Wire.endTransmission(); // stop transmitting
}

//////////////////////////////////////////////////////////////////////////////
///////                  POT set   activa el potenciometro digital
//////////////////////////////////////////////////////////////////////////////
void POTset(byte iPOT, byte datPOT)
{
```

```
  Wire.beginTransmission(direcPOT); // transmito la direccion
  Wire.write((channelPOT[iPOT] << 5) & 0b01100000);
  Wire.write(datPOT);
  Wire.endTransmission(); // stop transmitting
}

//////////////////////////////////////////////////////////////////////////////
///////                    mideterminal
//////////////////////////////////////////////////////////////////////////////

float mideTerminal(byte canal)
{

  if (canal & 1)
    digitalWrite(5, HIGH);
  else
    digitalWrite(5, LOW);
  if (canal & 2)
    digitalWrite(4, HIGH);
  else
    digitalWrite(4, LOW);
  if (canal & 4)
    digitalWrite(12, HIGH);
  else
    digitalWrite(12, LOW);

  delayMicroseconds(5); //3us de slew rate en el peor caso + 1us de Testablecimiento 5us por seguridad
  average = 0;
  //    for (int i = 0; i < 194; i++) {
  //    average = average + ((float)analogRead(analogInPin) / 102.4);
  //  }

  for (int i = 0; i < 19; i++)
  {
    average = average + ((float)analogRead(analogInPin) / 102.4);
  }

  return (average / 19.0); //esta incluido el factor 10 del divisor de tension
  // habria que filtrat 194 muestras para que se realizara en un ciclo de reloj y de esta manera evitar el
ruido de la alimentacion
  // voy a promediar con menos ya que son demasiados valores
  //promedio con 19
}

//////////////////////////////////////////////////////////////////////////////
///////                    measureI mide la intensidad
//////////////////////////////////////////////////////////////////////////////

float measureI(byte canal1, byte canal2, float POTresist)
{
  float v1 = 0;
```

```
  float v2 = 0;
  v1 = mideTerminal(canal1);
  v2 = mideTerminal(canal2);

  return ((v1 - v2) * 1000 / POTresist);
}

////////////////////////////////////////////////////////////////////////////////
///////                    POT resist
////////////////////////////////////////////////////////////////////////////////

float POTresist(int D)
{

  return (((256 - float(D)) / 256) * 200000 + 120);
}

////////////////////////////////////////////////////////////////////////////////
///////                    POTvalor
////////////////////////////////////////////////////////////////////////////////
//funcion para que me calcule el valor que le tengo que dar a POT para conseguir la R que deseo
float POTvalor(int POTres)
{

  return 256 - ((120 + POTres) * 256 / 200000);
}
```