UNIVERSIDAD TECNOLOGICA NACIONAL
Facultad Regional Bahía Blanca

Proyecto Final:

TRAZADOR DE CURVAS DE SEMICONDUCTORES

INTEGRANTES:

Acosta, Demis Legajo: 16140
Leonhardt, Federico Legajo: 15810

DOCENTES:

Ing. Friedrich, Guillermo
Ing. Laiuppa, Adrián

Resumen

Las tecnologías de la información y la electrónica brindan grandes capacidades didácticas. De esta forma, luego de una extensa etapa de búsqueda y en conformidad con la cátedra Proyecto Final se desarrolló en varias etapas un trazador de curvas de semiconductores. El mismo posee la capacidad de graficar las curvas características de salida de dispositivos BJT, MOSFET, FET, DIODO. El equipo final fue diseñado en torno a dos placas de desarrollo. Por un lado, se empleó una STM Discovery Cortex M4 conectada a una placa amplificadora de potencia, para implementar las rutinas de excitación de los dispositivos. En una segunda etapa de desarrollo, se integró al sistema preexistente una RASPBERRY PI 3 y se desarrolló una aplicación de visualización y control, que proporciona dos interfaces de usuario: ventanas y un servidor web.

Tabla de contenido

Figuras

1. Introducción

Los dispositivos semiconductores son los elementos primordiales en los circuitos electrónicos. Debido a su gran importancia, en la carrera de Ingeniería Electrónica, más precisamente en la cátedra Dispositivos Electrónicos, el funcionamiento y composición de los componentes abordan el tema principal. Es por esto que para comprenderlos en detalle, los laboratorios de dicha materia apuntan directamente a profundizar los conocimientos prácticos.

Para poder ensayar estos dispositivos se deben realizar una serie de pasos los cuales van acompañados de elementos adicionales tales como fuentes de alimentación, divisores resistivos, multímetros, osciloscopios, tablas de Excel, entre otros. Las pruebas se realizan paso por paso y "manualmente", de no ser así el alumno no podría comprender lo que realmente está pasando, o dicho de otro modo, cómo se comporta el dispositivo frente a los ensayos a los cuales se ve sometido. Considerando lo complejo y trabajoso que resulta levantar punto a punto una curva característica de cualquier semiconductor, surgió la idea de diseñar un equipo que pueda realizar los ensayos de manera autónoma, brindando curvas que le permitan al alumno compararlas con las trazadas manualmente, así como también con las hojas de datos de los diferentes dispositivos. Esto, sin lugar a duda, fue el principal propósito que impulsó el desarrollo del Trazador de Curvas de Semiconductores.

En el ámbito comercial, las marcas más reconocidas en lo referido a mediciones electrónicas, ponen a disposición de sus clientes trazadores de curvas de distintas características; a pesar de que esto sea una oferta tentadora para las Universidades, el costo de los mismos supera los miles de dólares.

Lo mencionado en el párrafo previo era un problema a la hora contrastar el funcionamiento de nuestro Trazador, nos preguntábamos en cada momento "¿estarán bien las curvas?", éste problema pudo ser resuelto cuando los docentes de la cátedra Dispositivos Electrónicos recordaron que en su lugar de trabajo existía un dispositivo que realizaba lo que necesitábamos. Luego de varios días de organización y con los respectivos permisos, la predisposición de dichos profesores fue tal que pudimos acceder a un lugar reservado y utilizar un trazador comercial. Esto nos permitió corroborar y calibrar nuestro Trazador de Curvas de Semiconductores. Luego de contrastar nuestro prototipo con aquel equipo, pudimos concluir que los resultados eran muy similares, tomando en consideración los niveles de potencia de trabajo y los límites de exactitud que manejan estos dispositivos de ensayos.

2. Conceptos Teóricos

2.1. ¿Que son los semiconductores?

Un dispositivo semiconductor es un componente electrónico que emplea las propiedades electrónica de los materiales semiconductores, principalmente del silicio, el germanio y el arseniuro de galio, así como de los semiconductores orgánicos. Los dispositivos semiconductores han reemplazado a los dispositivos termoiónicos (tubos de vacío) en la mayoría de las aplicaciones.
Los materiales semiconductores son tan útiles debido a que su comportamiento puede ser fácilmente manipulado por la adición de impurezas, conocidas como dopaje . La conducción de corriente en un semiconductor se produce a través de electrones y agujeros móviles o "libres", conocidos conjuntamente como portadores de carga. El dopaje de un semiconductor como el silicio con una pequeña cantidad de átomos de impurezas, tales como el fósforo o boro, aumenta en gran medida el número de electrones o agujeros libres dentro del semiconductor. Cuando un semiconductor dopado contiene huecos en exceso que se llama "tipo P" y cuando contiene un exceso de electrones libres se conoce como de "tipo N", donde P (positivo para agujeros) o N (negativo para electrones) es el signo de los portadores de carga móviles mayoritarios. La unión que se forma entre los semiconductores de tipo-N y tipo-P se denominan junturas P-N.

2.2. Tipos de semiconductores

El dispositivo semiconductor más simple que puede construirse es un diodo, el cual está formado por un semiconductor tipo N y un tipo P unidos el uno contra el otro. Los electrones donados por el semiconductor tipo N se difundirán por los límites y caerán en los agujeros del semiconductor tipo P, llenándolos. De un modo similar, los agujeros donados por el semiconductor tipo P pueden difundirse de la otra forma y absorber los electrones donados. El resultado neto es que todas las partículas y agujeros donados se cancelan, dejando una hilera de iones positivos en el lado N y una hilera de iones negativos en el lado P.

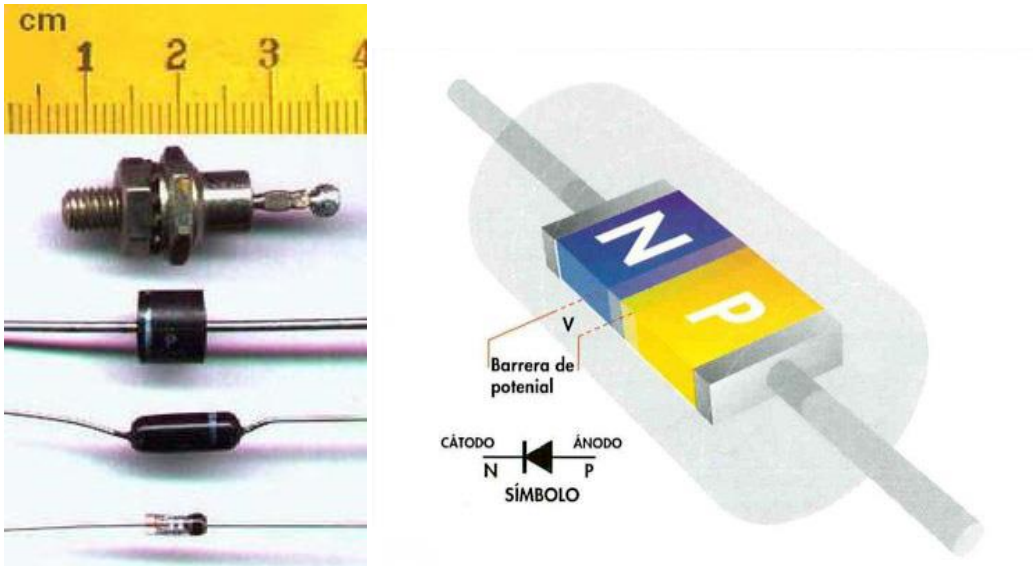

Figura 1: Semiconductor tipo Diodo

El dispositivo semiconductor más difundido es el transistor. Un transistor típico se conforma simplemente de tres capas de semiconductor: o bien dos semiconductores tipo P formando un sandwich con un tipo N en medio, o la inversa. Las cargas (o electrones o agujeros) fluyen de un trozo de "pan" del "sandwich" al otro trozo. Durante un breve periodo se hallan en el pequeño semiconductor de tipo distinto (el "relleno"). En consecuencia, cuando aplicamos pequeños cambios de voltaje a la parte central del dispositivo, podemos producir cambios más grandes en la corriente que fluye entre las dos piezas de los extremos, de un modo muy parecido a como pequeños cambios en una válvula pueden producir cambios más grandes en el flujo de agua en una gran tubería.

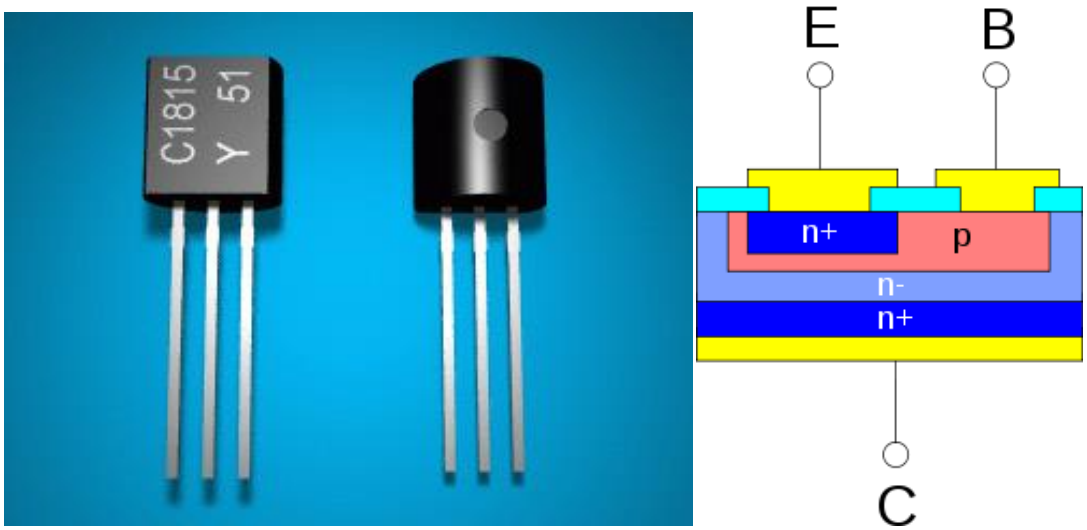

Figura 2: Semiconductor tipo Transistor

El funcionamiento básico del transistor consiste en aplicar una tensión entre emisor y colector, y una corriente suficiente en la base para que circule corriente entre colector y emisor. La relación entre la corriente de la base y la que circula por colector- emisor se denomina Ganancia del transistor.

Los transistores tienen usos muy diferenciados y de gran importancia tecnológica. Por una parte, son el dispositivo esencial de los circuitos osciladores, que son capaces de generar señales alternas a partir del consumo de energía eléctrica procedente de la corriente continua. Por otra, es posible usarlos como elementos que responden a una entrada binaria (tensión por encima o por debajo de un cierto valor) con una salida también binaria, en forma de tensión o de intensidad. Esta última propiedad hace al transistor el elemento fundamental de los circuitos electrónicos, en particular de los propios de la tecnología digital.

Las múltiples aplicaciones del transistor han provocado su desarrollo y evolución hacia otros dispositivos más específicos como tiristores, diacs y triacs.

Para describir de manera completa el funcionamiento y las características de los dispositivos semiconductores se utilizan diversas curvas características que relacionan las tensiones y corrientes entre sus terminales.

2.3. ¿Qué es un trazador de curvas?

Los semiconductores presentan un comportamiento complejo. Por un lado son dispositivos no lineales, esto es, su relación tensión-corriente no se puede describir mediante una ecuación diferencial lineal y además las funciones que describen dicha relación son ecuaciones transcendentes y por ello de difícil solución. Para complicar aún más el panorama, las funciones que rigen el comportamiento no son únicas sino que varían en función de la magnitud de las variables de corriente y tensión. Es decir, que los semiconductores tienen diversos modos de comportamiento y cada uno de estos modos presenta dificultades superiores a las normales en su resolución.

Un trazador de curvas de semiconductores es un equipo electrónico de prueba utilizado para analizar las características de dispositivos tales como diodos y transistores. Se basa principalmente en la capacidad de inyectar tensión y corriente al elemento sometido al ensayo. Es de suma importancia tener en cuenta que el método por el cual se realiza lo anteriormente mencionado se debe llevar a cabo siguiendo una serie de pasos bien definidos para lograr que las curvas se asemejen cualitativamente a las dispuestas por los fabricantes de los diferentes dispositivos. A pesar de que en el ámbito comercial existan trazadores de primera marca, el valor de los mismos excede el presupuesto que posee una universidad para comprar elementos con tal fin.

2.4. ¿Cómo funcionan los trazadores comerciales?

En pocas palabras se trata de aplicar un barrido (que varía automáticamente de forma continua con el tiempo) de tensión a dos terminales del dispositivo bajo prueba, y se toma la medida de la cantidad de corriente que el dispositivo permite que fluya en cada nivel voltaje.

El trazador de curvas puede visualizar todos los parámetros de interés tales como la tensión directa del diodo, la corriente de fuga inversa, la tensión de ruptura inversa, y así sucesivamente.

En este sentido, el aporte destacado del equipo aquí presentado se encuentra en la posibilidad de obtener la ganancia dinámica de los distintos dispositivos, seleccionar la cantidad de curvas requeridas, niveles de tensión y corrientes de ensayo máximos, entre otras.

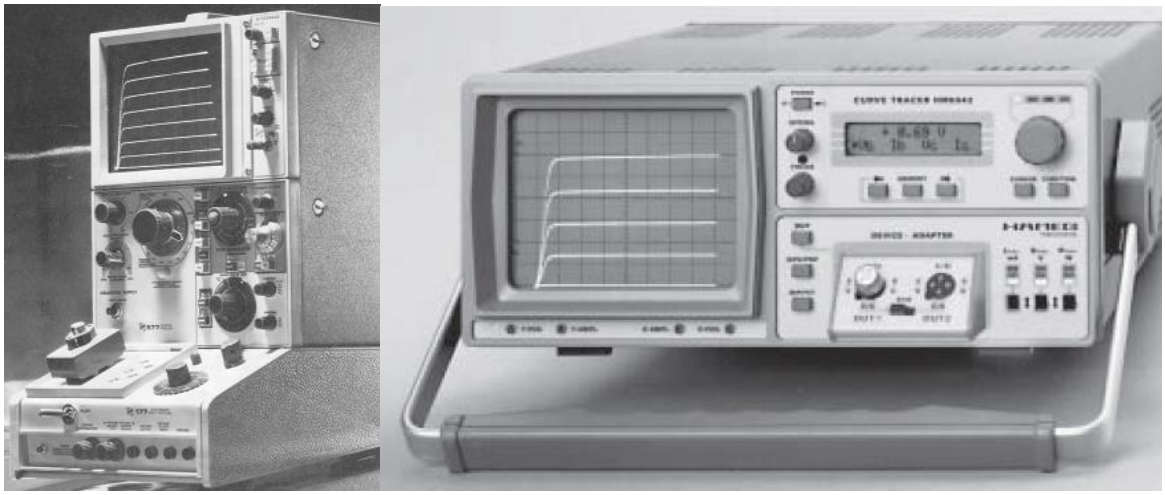

Figura 3: Trazadores comerciales

Las curvas de un transistor son un poco más complejas que las curvas de un diodo. La cantidad de corriente que pasa entre un colector BJT y un emisor depende del nivel actual presente en la base. Al proporcionar un voltaje escalonado en la base y realizar un barrido de voltaje completo para cada paso de voltaje de base en el colector, podemos capturar una familia de varias curvas IV (corriente-tensión) características. Este mismo principio se aplica a encontrar las curvas IV de un MOSFET .

El trazador de curvas puede generar y visualizar una familia de curvas de corriente de colector, IC, frente a voltaje de colector a emisor, VCE, para varios valores de corriente de base, IB.

Básicamente, para generar esta gráfica se debe contar con:

1. Un generador de voltaje de barrido para controlar el voltaje del colector.

2. Una fuente de corriente base que se puede controlar para proporcionar un número de incrementos iguales de las corrientes de base con cada barrido del generador de voltaje.

3. Una fuente de sincronización para cambiar la corriente base al inicio de cada barrido de tensión.

Forma de onda del generador de voltaje de barrido Vs: Cada barrido se produce con un período de tiempo T. Este es el voltaje de suministro del colector que se aplica repetitivamente al transistor.

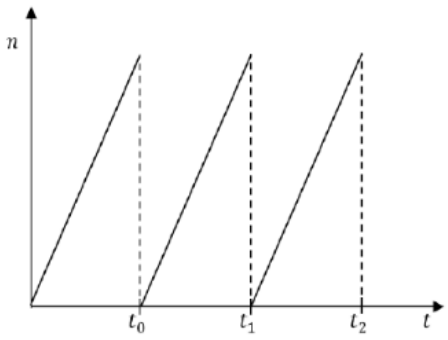

Figura 4: Forma de onda de generador de voltaje de barrido

Salida de la fuente de corriente base: Para cada barrido de tensión consecutiva, la corriente de base, IB, se incrementa en escalones iguales, con los pasos sincronizados al comienzo de cada barrido de tensión del colector. A medida que finaliza el último período de incremento, el generador de corriente base repite la secuencia de pasos, proporcionando una visualización estable y continua.

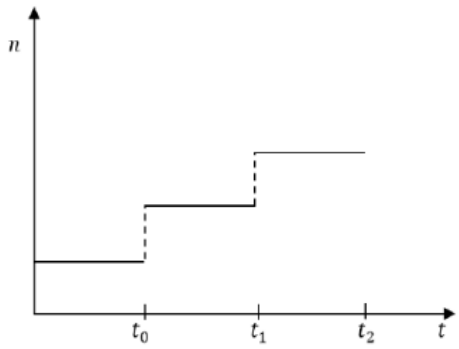

Figura 5: Forma de onda de corriente de base

A continuación se muestra un diagrama simplificado del funcionamiento de un trazador de curvas convencional:

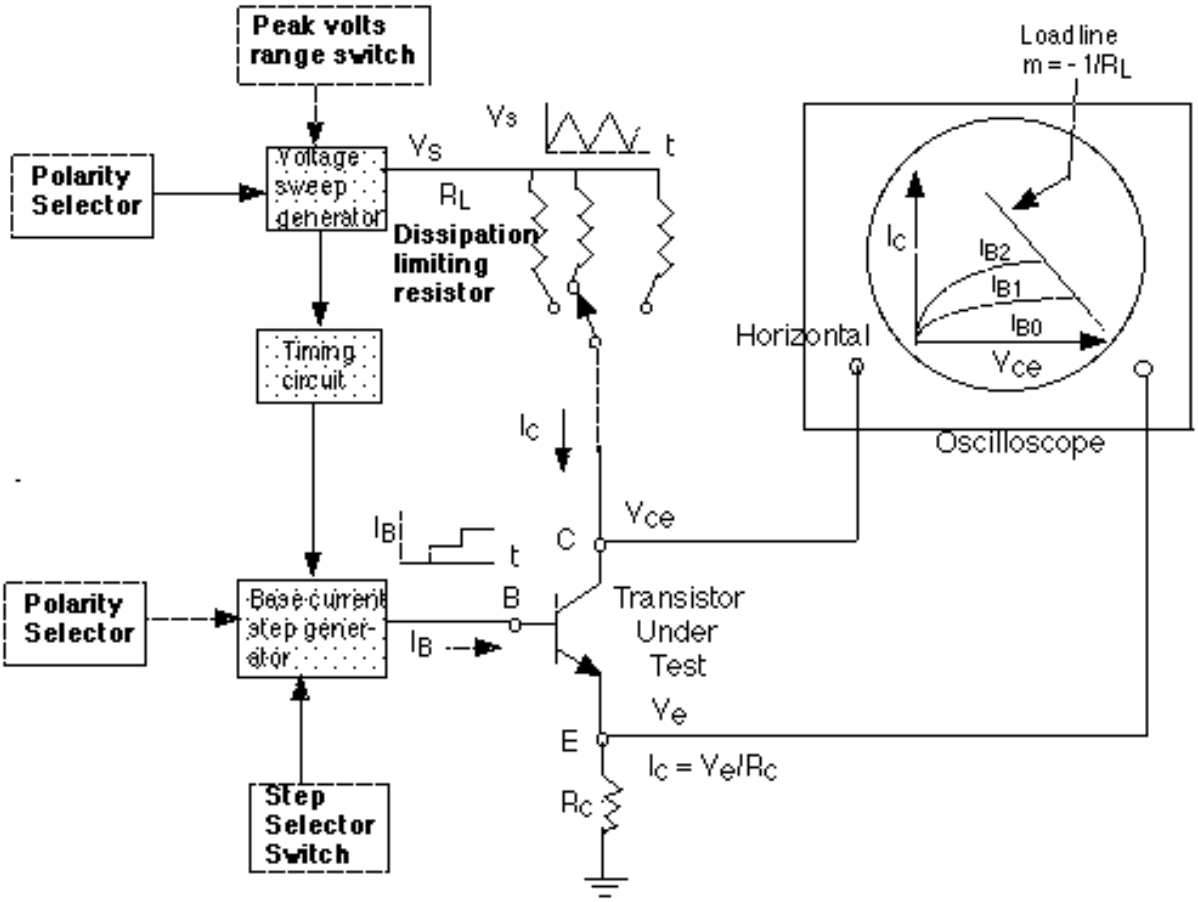

Figura 6: Diagrama de funcionamiento trazador de curvas

El voltaje de colector a emisor, Vce, proporciona el barrido horizontal, mientras que el voltaje en la resistencia de detección de corriente, Rc, que es proporcional a la corriente del colector, proporciona el barrido vertical, lo que da como resultado una familia de curvas de Ic frente a Vce para una serie de cambios de incremento iguales en la corriente de base.

2.5. Un poco de historia

Antes de la introducción de los semiconductores, había trazadores de curvas de tubos de vacío (p. Ej., Tektronix 570). Los primeros trazadores de curvas de semiconductores usaban circuitos de tubos de vacío, ya que los dispositivos semiconductores disponibles hasta entonces, no podían hacer todo lo necesario en un trazador de curvas.

Figura 7: Trazador Tektronix 570

Hoy en día, los trazadores de curva son completamente de estado sólido y están sustancialmente automatizados para facilitar la carga de trabajo del operador, capturar datos automáticamente y garantizar la seguridad del trazador de curvas y del dispositivo bajo prueba.

Los diseños modernos de instrumentos trazadores de curvas tienden a ser modulares, lo que permite que los especificadores del sistema los configuren para que coincidan con las aplicaciones para las que serán utilizados. Por ejemplo, los nuevos sistemas trazadores de curvas basados en el mainframe se pueden configurar especificando el número y el nivel de potencia de las Unidades de medida de fuente (SMU) que se conectarán en las ranuras del panel posterior del chasis. Este diseño modular también brinda la flexibilidad de incorporar otros tipos de instrumentación para manejar una gama más amplia de aplicaciones. Estos sistemas basados en mainframe suelen incluir una PC autónoma para simplificar la configuración de la prueba, el análisis de datos, la representación gráfica y la impresión, y almacenamiento de resultados a bordo. Los usuarios de este tipo de sistemas incluyen investigadores de semiconductores, ingenieros de modelado de dispositivos, ingenieros de confiabilidad, ingenieros de clasificación de matrices e ingenieros de desarrollo de procesos.

Además de los sistemas basados en mainframe, están disponibles otras soluciones trazadoras de curvas que permiten a los integradores de sistemas combinar una o más Unidades de medida de fuente (SMU) discretas con un software de trazador de curva de ejecución de controlador de PC separado. Las SMU discretas ofrecen una gama más amplia de niveles de corriente, voltaje y potencia que los sistemas basados en mainframe y permiten que el sistema se reconfigure a medida que cambian las necesidades de prueba. Se han desarrollado nuevas interfaces de usuario basadas en el asistente para facilitar a los estudiantes o usuarios de la industria menos experimentados encontrar y ejecutar las pruebas que necesitan, como la prueba de seguimiento de la curva FET.

3. Descripción del Proyecto

El proyecto desarrollado consta de varios módulos, cada una de los cuales posee una tarea específica. A lo largo de este documento se irá profundizando cada etapa para comprender su rol individual y su relación en conjunto.
Todas las funciones del Trazador son comandadas en conjunto por dos programas. El primero y principal se encuentra en una placa de desarrollo Discovery STM32f4, y el segundo en un kit de desarrollo RASPBERRY PI 3. Ambos programas se comunican entre sí mediante mensajes, los cuales se combinan y logran que las tareas necesarias para todo el control sobre el hardware se realice de manera efectiva.

El Trazador consta de una placa madre sobre la cual se encuentra montado el kit programable STM32F4, una placa de potencia que posee la fuente de alimentación y además permite adaptar tensiones y corrientes, un transformador de tensión, una fuente switching, un kit RASPBERRY PI 3, un display LCD 5'' y todas las interconexiones necesarias para la implementación.
El dispositivo desarrollado se divide principalmente en cuatro etapas:

1. Etapa de Control de lógica operacional (STM32F4).
2. Etapa de Potencia.
3. Etapa de Comunicación.
4. Etapa de Control de interfaz de usuario (RASPBERRY PI 3).

Como se dijo anteriormente, cada una de ellas tiene funciones bien definidas, y necesariamente deben trabajar en conjunto para efectuar las operaciones, cálculos, adaptaciones eléctricas y visualización de parámetros y resultados obtenidos.

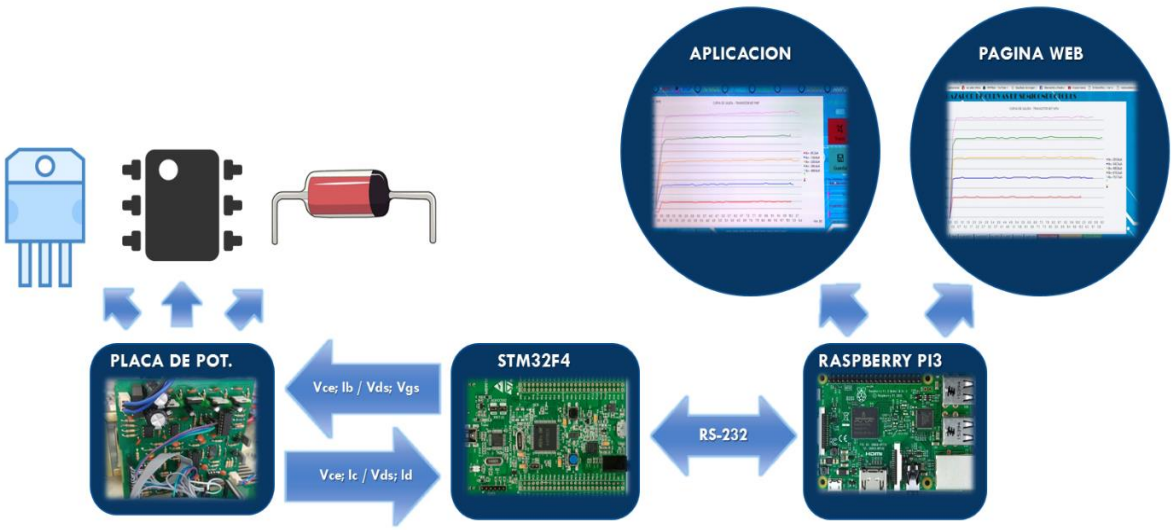

Figura 8: Esquema de composición del Trazador

3.1. Etapa de Control de lógica operacional (STM32F4)

El kit STM32F4 dispone de numerosas entradas y salidas digitales, entradas y salidas analógicas, conversores A/D y D/A, timer's; además posee funciones adicionales que por el momento no son de interés para el dispositivo desarrollado.
Para el trazador de curvas se han empleado:

- 6 salidas digitales en modo pull-up: manejo de las llaves analógicas de la placa de potencia.
- 1 salida digital: indicador luminoso en modo intermitente cuando el equipo se encuentra "calculando".
- 2 entradas ADC: lectura de parámetros de tensión y/o corriente del semiconductor ensayado.
- 2 salidas DAC: suministro de tensión y/o corriente para semiconductor ensayado.
- RS232- USART: comunicación entre placas de desarrollo.

El bucle principal del sistema de control comienza a operar por medio de mensajes a través del puerto serie del dispositivo. Los parámetros seleccionados por parte del usuario, mediante la interfaz gráfica, son traducidos en la placa de control para efectuar las distintas operaciones. Dentro de las posibilidades de elección por parte del usuario, se establecen las salidas necesarias para configurar los niveles de tensión y corriente con que se someterá al semiconductor ensayado.
El dispositivo realiza una serie de verificaciones previas para asegurar que la conexión de los terminales del semiconductor se haya efectuado correctamente y que el mismo no se encuentre previamente dañado.
Luego de la configuración inicial para el semiconductor en cuestión, se procede a realizar los pasos necesarios para el levantamiento de las curvas de salida. Esto se logra mediante la variación de los DAC de salida para excitar el semiconductor y la lectura continua de los ADC que monitorean los valores alcanzados por el mismo.
Una vez obtenidos los valores de tensión y corriente que conformarán las curvas de salidas para el número de trazos elegido previamente, o bien se haya alcanzado el valor máximo de corriente seleccionada, se enviará un paquete de datos desde la placa STM32F4 hacia la RASPBERRY PI 3, encargada de traducir y representar dichos valores en una gráfica de dos ejes.

3.1.1. El Microprocesador

El sistema embebido utilizado en el proyecto es un microprocesador de 32bits ARM CortexM4, montado sobre el Kit STM32F4 Discovery. La elección de este microprocesador se debió a que se trata de un kit que se ha utilizado a lo largo de la carrera por lo que la experiencia sobre éste permitió facilitar el proceso de desarrollo y obtener el máximo provecho de las herramientas que brinda.

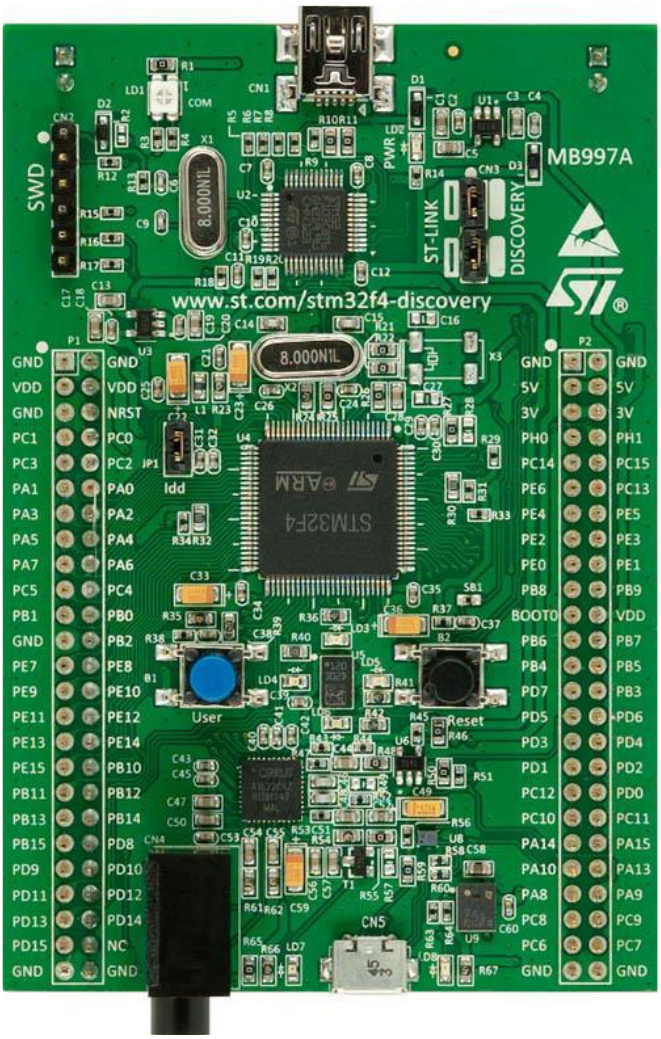

Figura 9: Kit STM32F4 Discovery

El microprocesador STM32F407VGT6 presenta las siguientes características:

- Frecuencia máxima de operación 168MHz / 210 DMIPS.
- Memoria Flash de 1 Mbyte.
- Memoria RAM de 192 Kbytes.
- Unidad de Punto Flotante (FPU) y DSP MAC (Procesamiento digital de señales).
- 1 Módulo USB OTG HS/FS.
- 1 Módulo Ethernet.
- 17 Timer’s.
- 3 ADC’s (Conversor Analógico-Digital).
- Diseñado para un alto rendimiento y muy elevada transferencia de datos.
- Eficiencia energética y ultra-bajo consumo de energía (RTC menor a 1µA en modo VBAT y 3,6V hasta 1,7V VDD).

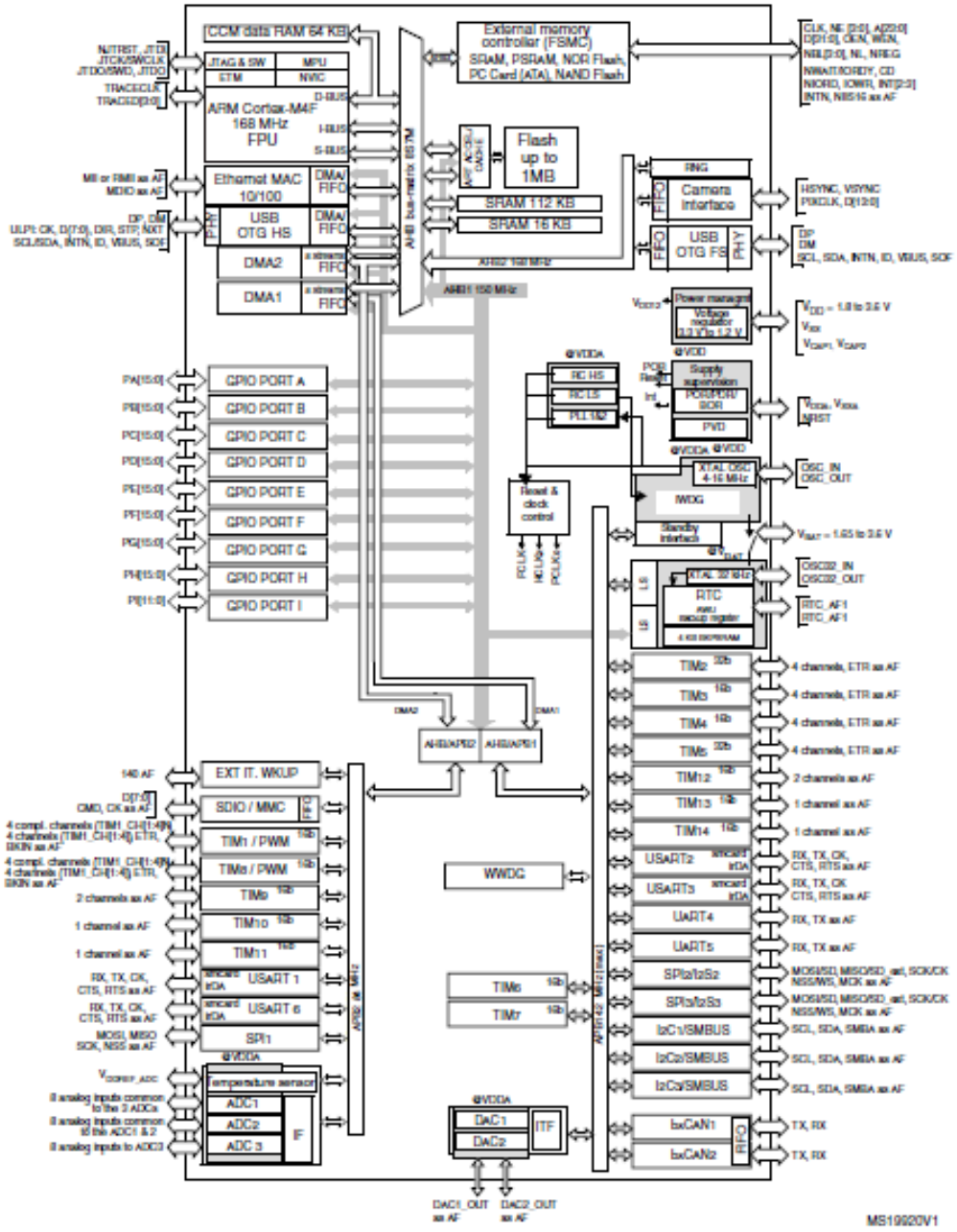

Figura 10: Diagrama interno STM32F4

3.1.2. Desarrollo de programa de lógica operacional

Como se mencionó anteriormente, el equipo desarrollado opera bajo dos programas independientes corriendo en placas separadas. Al encender el dispositivo comienzan a ejecutarse ambos programas. Una vez que la interfaz de usuario se hace visible en el display, el trazador se encuentra listo para recibir la configuración respecto al tipo de dispositivo a ensayar. Los parámetros seleccionados por el usuario son interpretados por ambos programas, los cuales se encuentran en constante comunicación a través del puerto serie.
Cada uno de los programas en ejecución fue desarrollado, necesariamente, en distintos entornos. Por un lado, la placa STM32F4 fue programada en lenguaje C utilizando el software Coocox CoIDE.
Sin profundizar demasiado en torno a los detalles de su programación es importante destacar que es en este lugar en donde se reciben, a través del puerto serie, los parámetros seleccionados por el usuario. A partir de ellos se configuran las entradas y salidas para ensayo y captura de datos. Se archivan los resultados en arreglos de datos y por último se prepara un paquete que se envían nuevamente por el puerto serie hacia la RASPBERRY PI 3 para ser procesado.

TIPO	PINES	VARIABLE	DESCRIPCION
SALIDAS DIGITALES	Puerto E Pin 14, Puerto E Pin 15	IB_100uA	Puerto E Pin 14 = 1, Puerto E Pin 15 = 1
		IB_1mA	Puerto E Pin 14 = 0, Puerto E Pin 15 = 1
		IB_10mA	Puerto E Pin 14 = 1, Puerto E Pin 15 = 0
	Puerto E Pin 7, Puerto E Pin 8	IB_pos	Puerto E Pin 7 = 0, Puerto E Pin 8 = 1
		IB_neg	Puerto E Pin 7 = 1, Puerto E Pin 8 = 0
	Puerto E Pin 11, Puerto E Pin 12	UCE_pos	Puerto E Pin 11 = 0, Puerto E Pin 12 =1
		UCE_neg	Puerto E Pin 11 = 1, Puerto E Pin 12 = 0
	Puerto D Pin 15	Led CALCULANDO	-
SALIDAS ANALOGICAS	Puerto A Pin 4	corriente_de_base	0….3,3V
	Puerto A Pin 5	tension_colector_emisor	0….3,3V
ENTRADAS ANALOGICAS	Puerto C Pin 4	Ic_ist	0….3,3V
	Puerto C Pin 5	UCE_ist	0….3,3V
COMUNICACIÓN SERIAL	Puerto A Pin 2	TX, RX	-
	Puerto A Pin 3		

Figura 11: Tabla de pines usados de STM32F4

3.1.3. Placa madre de control de lógica

3.1.3.1. Esquemático de Placa electrónica

Teniendo en cuenta futuras ampliaciones o modificaciones del sistema sin necesidad de hacer grandes cambios de hardware, se diseñó esta placa de forma tal que queden accesibles gran parte de los puertos del kit STM32F4. Se encuentran disponibles para su uso los 50 pines que integran la bornera izquierda del kit STM32F4.
Asimismo, se agregaron salidas a transistor para poder ser utilizadas en aplicaciones que demanden más corriente de la que el kit STM32F4 puede entregar, en el caso del Trazador estos pines proporcionan la corriente necesaria para indicador de equipo “calculando”.
Por último, se añadió una fuente de alimentación lineal de 5V para alimentación del kit. La misma se encuentra filtrada a fin de reducir los posibles ruidos que puedan interferir a la placa de control.

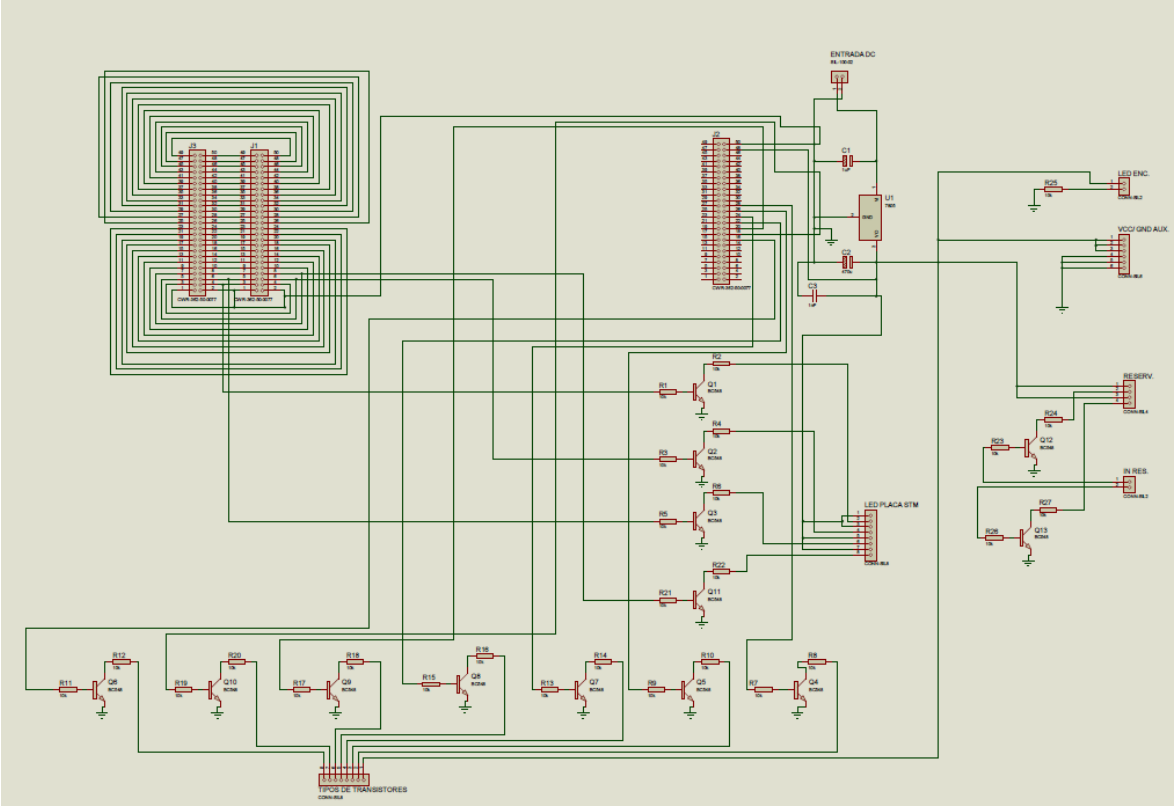

Figura 12: Circuito esquemático de placa electrónica de control

3.1.3.2. Diseño de Placa electrónica

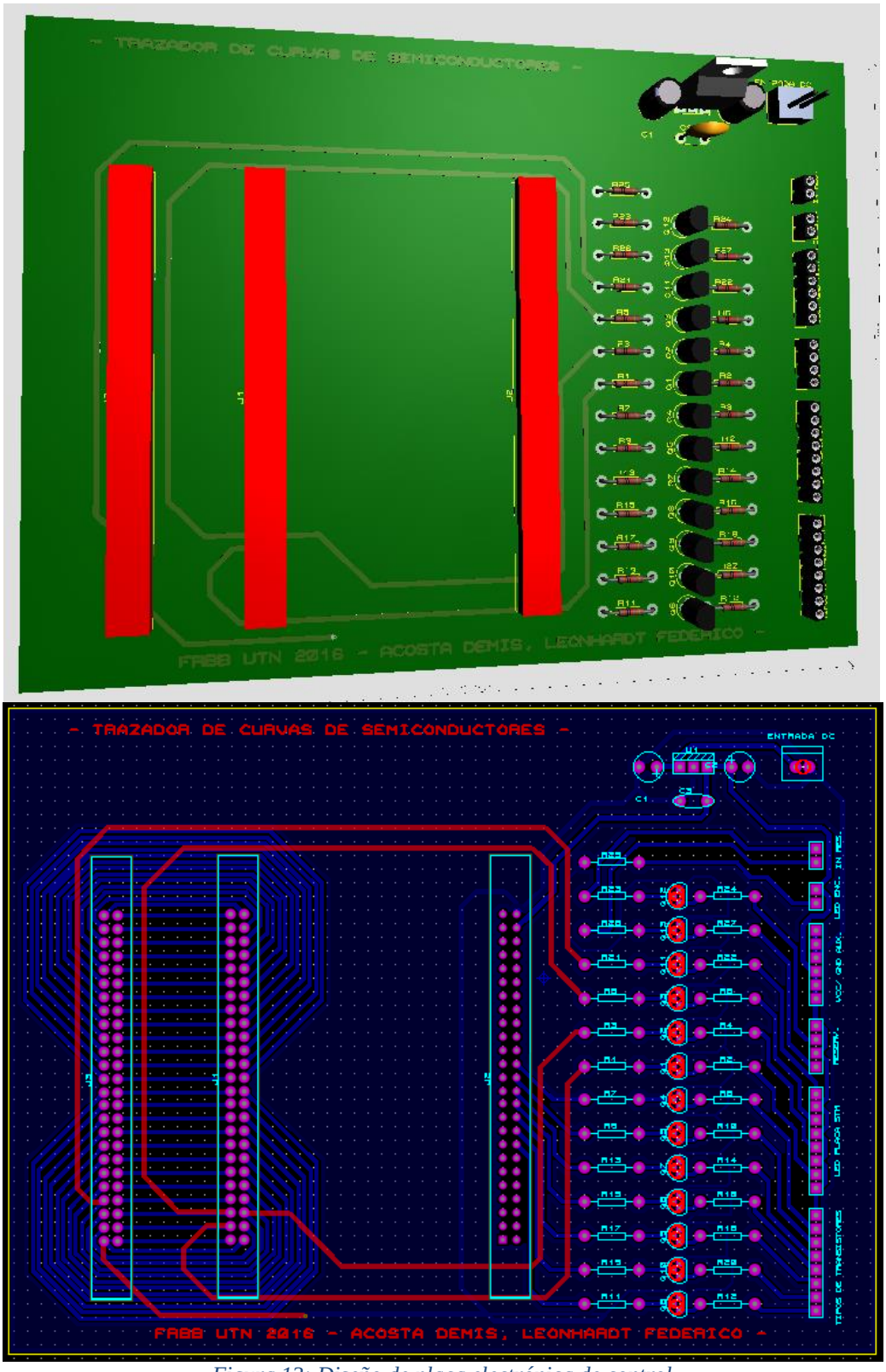

Figura 13: Diseño de placa electrónica de control

3.1.3.3 Fabricación y montaje de Placa

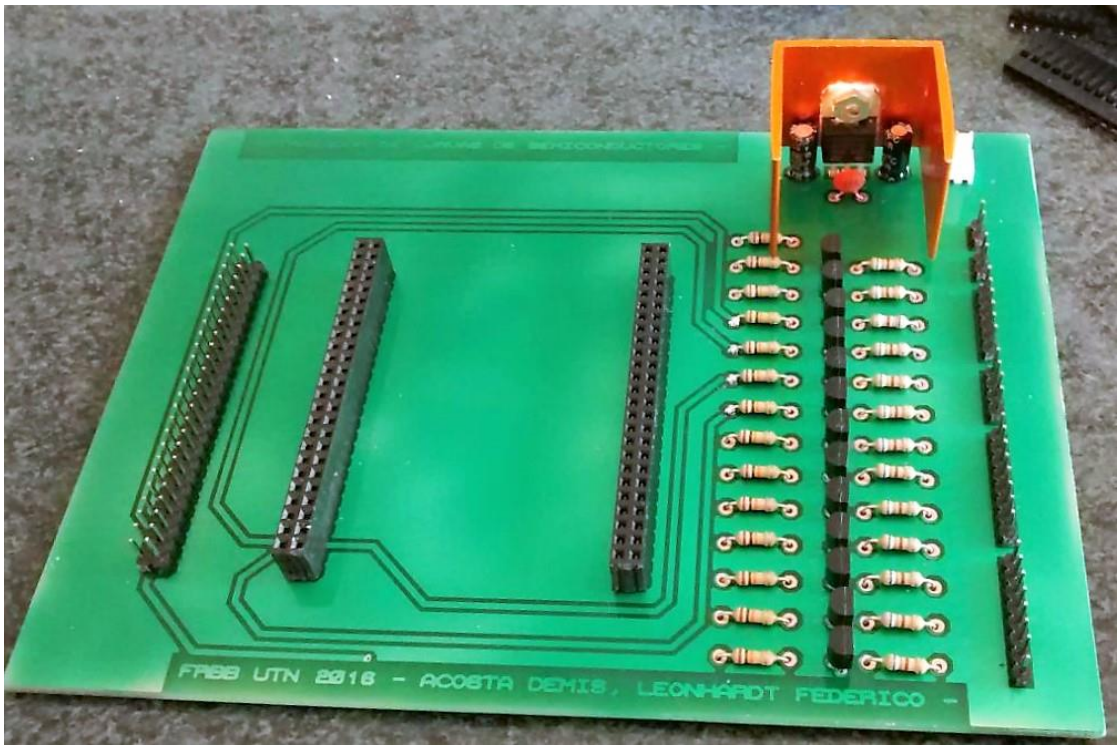

Figura 14: Montaje de componentes placa electrónica de control

Figura 15: Montaje de Kit STM32F4 en placa electrónica de control

3.2. Etapa de potencia

Esta etapa se encuentra integrada en una placa de 100mm x 100mm de doble capa. Se procedió a realizar el diseño de esta manera debido a necesidad de contar con un equipo lo más compacto posible.
La placa funciona con una tensión de entrada alterna del orden de los 18V. Este nivel de voltaje es provisto por un transformador con fuente partida de 220V/15V y con una potencia disponible de 150W. Para alimentar los diversos dispositivos de la placa se diseñaron fuentes lineales de distintos niveles.
La fuente de alimentación necesaria para el dispositivo de ensayo consta de cuatro reguladores lineales que brindan +12V,-12V,+15V,-15V. Dichos niveles de tensión son requeridos para los amplificadores operaciones, llaves analógicas y etapa amplificadora de potencia de salida.
Para la realización del Trazador fue necesario utilizar conversores analógico-digital (ADC) y digital-analógico (DAC). La placa STM32F4 cuenta con convertidores integrados, pero es necesario destacar que los mismos proporcionan un nivel máximo de tensión 3.3V. Por tal motivo, fue necesario utilizar amplificadores que eleven este nivel hasta 10V. Otro punto interesante a mencionar es que dicho proyecto posee la capacidad de ensayar dispositivos que requieren tensiones y corrientes negativas por lo que fue indispensable la integración de amplificadores operacionales en configuración de inversor. Debido a que los conversores analógico-digital utilizados soportan una tensión máxima de 3.3V positiva, la placa de potencia posee amplificadores que rectifican, amplifican y recortan las tensiones y corrientes muestreadas con el fin de lograr la compatibilidad entre la etapa de control y la etapa de potencia, de modo que no se produzcan niveles que puedan dañar alguno de los componentes.
La placa de potencia posee un total de dieciocho conexiones, entre estas se encuentran:

- Entrada de alimentación ±15V.
- Entradas digitales para la elección de polaridad de las señales de ensayo.
- Entradas digitales para elección del rango de corriente base o tensión de compuerta-fuente.
- Entradas analógicas que permiten la inyección de las tensiones necesarias para el barrido de las curvas.
- Salidas analógicas que brindan valores de 0 a 3.3V, los cuales son registrados por la etapa de control.
- Salidas para las puntas de prueba del equipo.

3.2.1. Adaptaciones eléctricas

El trazador tiene la capacidad de entregar en los ensayos una corriente máxima de drenador o colector de 500mA, además una tensión compuerta-fuente de 10V y en caso de BJT's esta tensión es necesaria para generar la corriente de base máxima. Estos valores de corrientes y voltajes no pueden ser generados por la etapa de control, por tal motivo fue necesario implementar una placa de potencia que adapte los valores máximos que soporta la STM32F4 a los requerimientos necesarios para el trazado de las curvas. A la hora de recolectar los datos, fue necesaria una circuitería que limite las máximas tensiones dispuestas en las entradas de la etapa de control con el fin de no dañarlas.

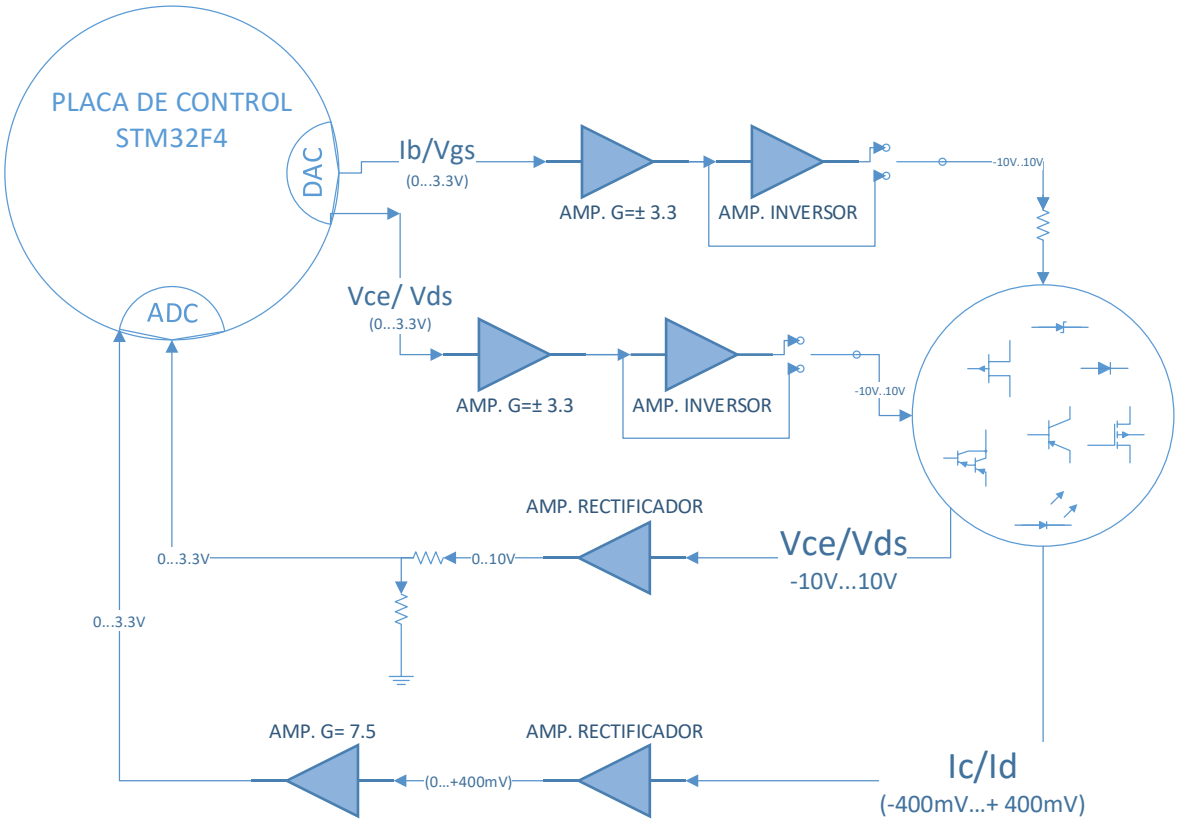

Figura 16: Esquema eléctrico de Placa de Potencia y adaptaciones

3.2.2. Diseño y fabricación de placa de potencia

3.2.2.1. Diseño de placa electrónica

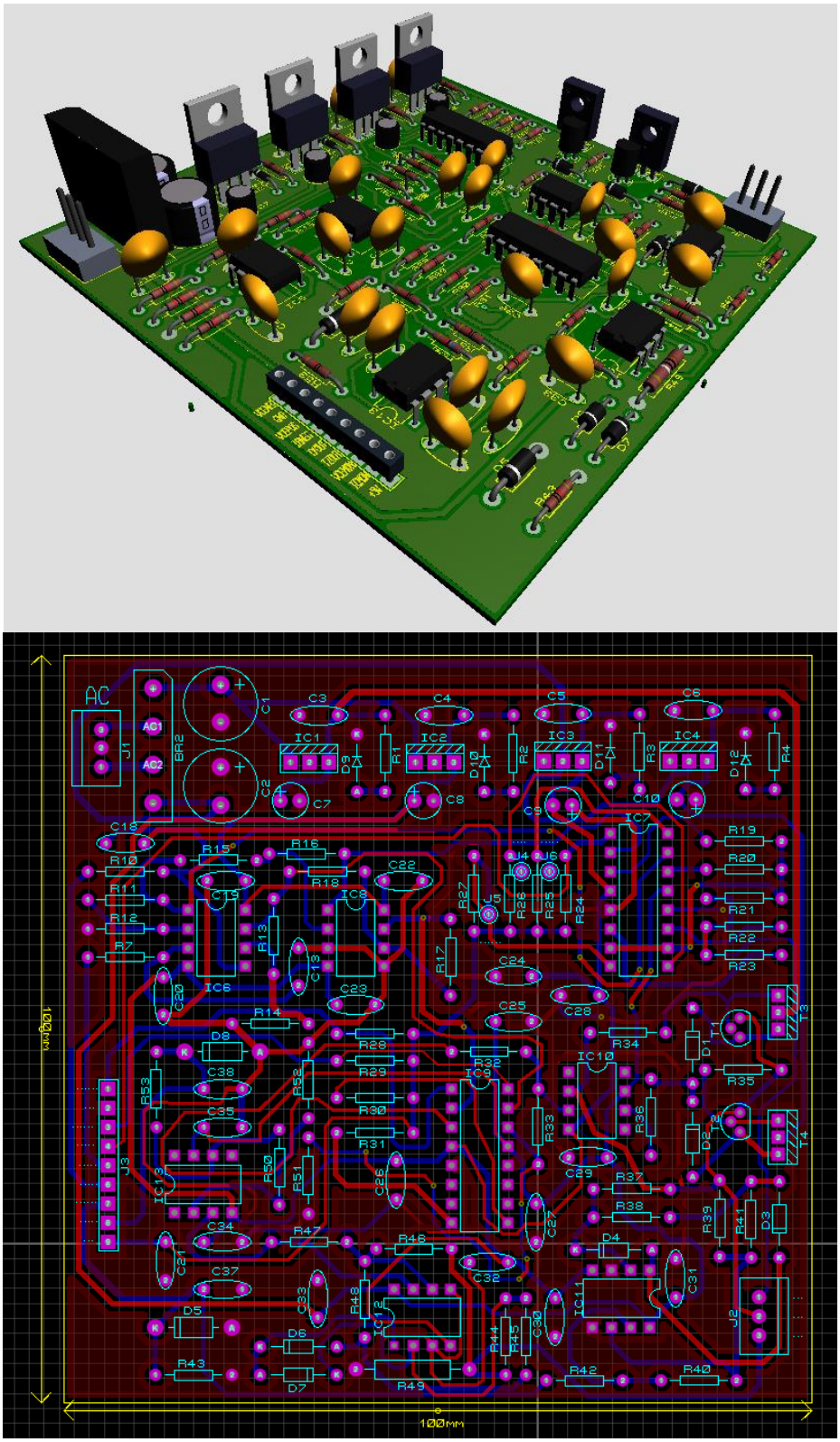

Figura 17: Diseño de placa electrónica de Potencia

3.2.2.2. Esquemático general del circuito de potencia

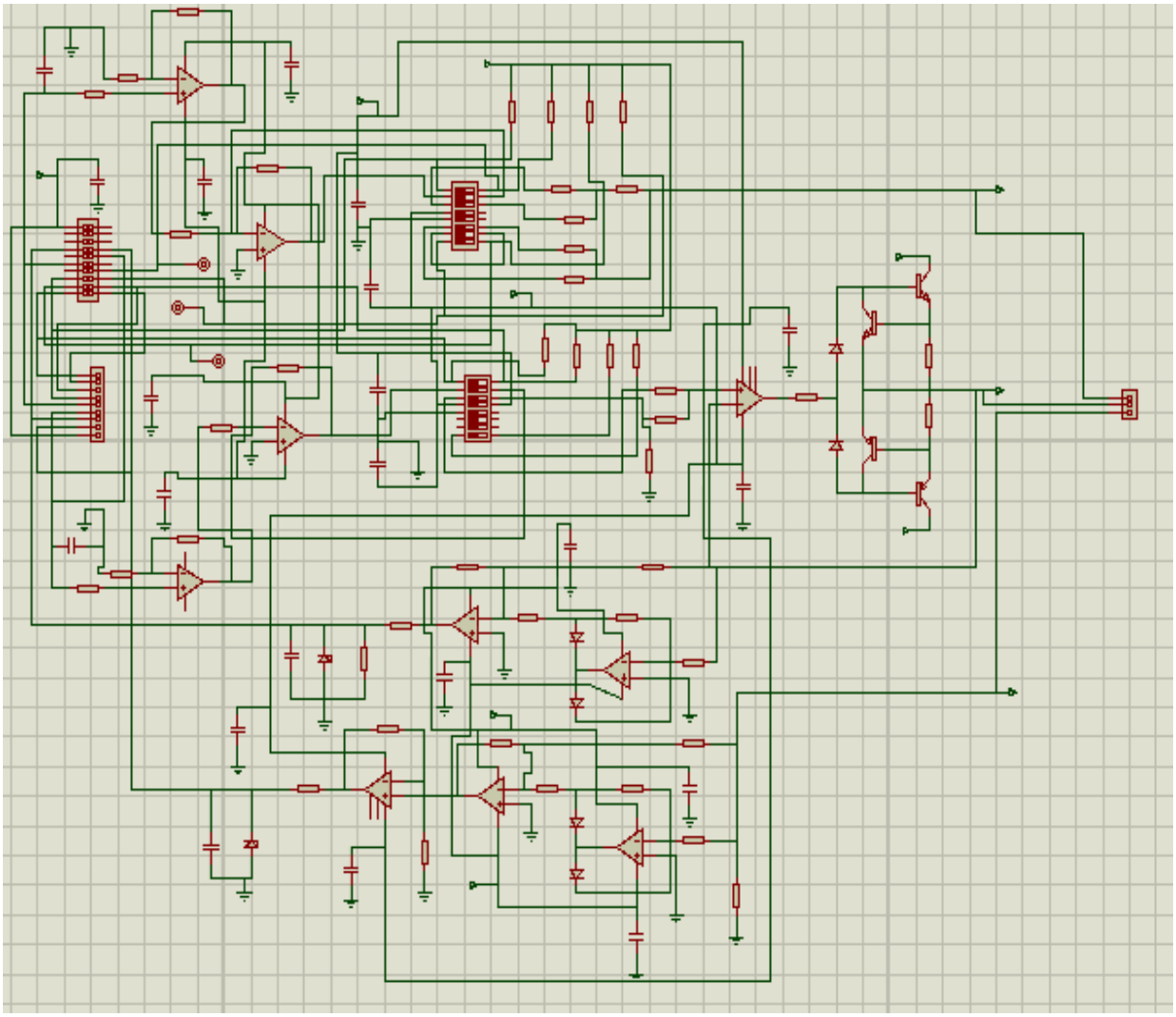

Figura 18: Circuito de placa electrónica de Potencia

Debido a las dimensiones y a la complejidad de cada parte del circuito de potencia, a continuación se detallarán las principales características que componen cada etapa.

3.2.2.3. Rectificación y fuentes de alimentación

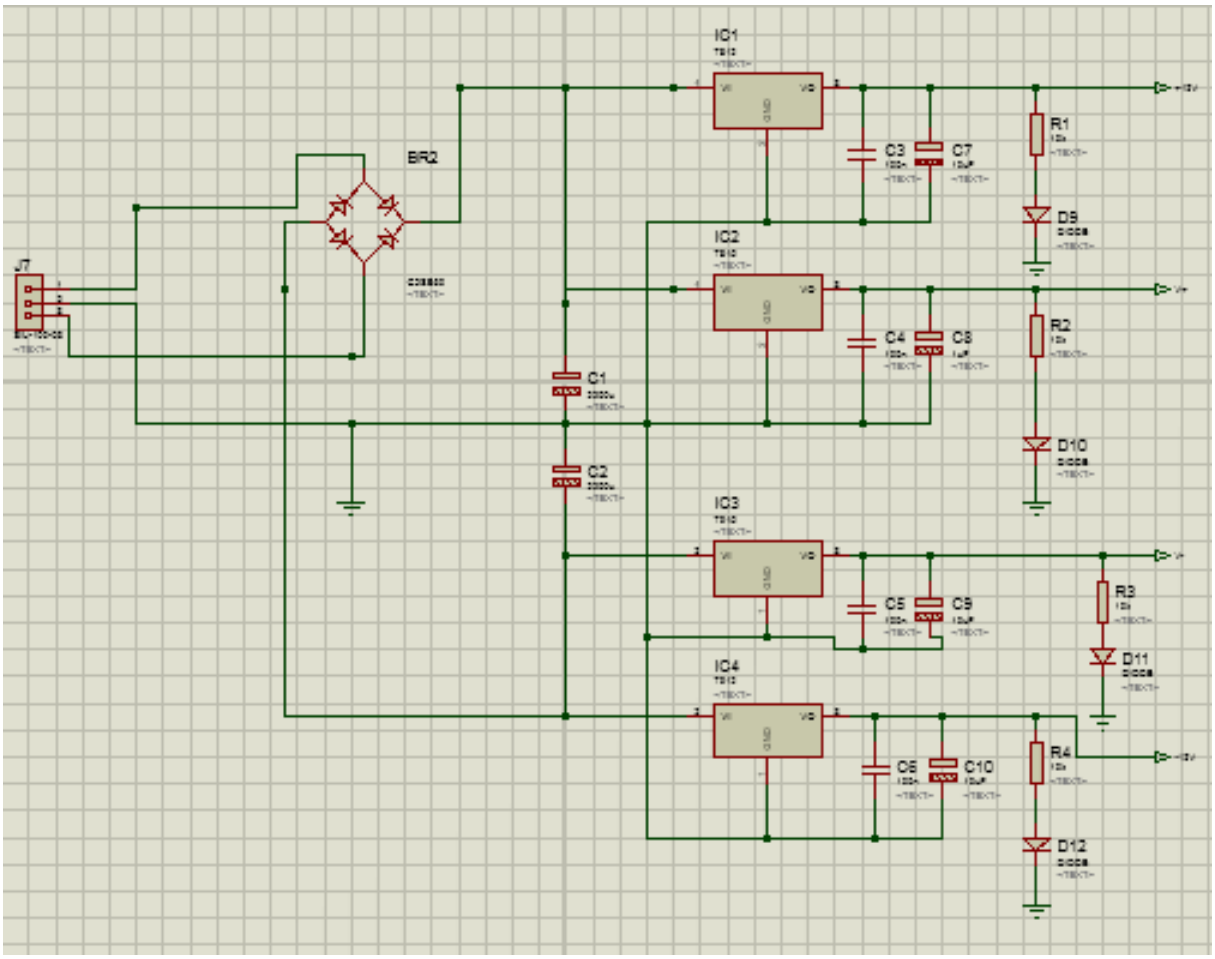

Figura 19: Etapa de fuentes lineales de placa de Potencia

Esta sección de la placa de potencia brinda las tensiones de alimentación necesarias para el funcionamiento del hardware:

- Rectificación de las tensiones de alimentación provenientes de un transformador de 15V+15V.
- +12V, -12V necesarios para la etapa amplificadora de salida.
- +15V,-15V necesarios para los amplificadores operacionales.

3.2.2.4. Entradas, salidas, amplificación, inversión de tensiones y selección de polaridad

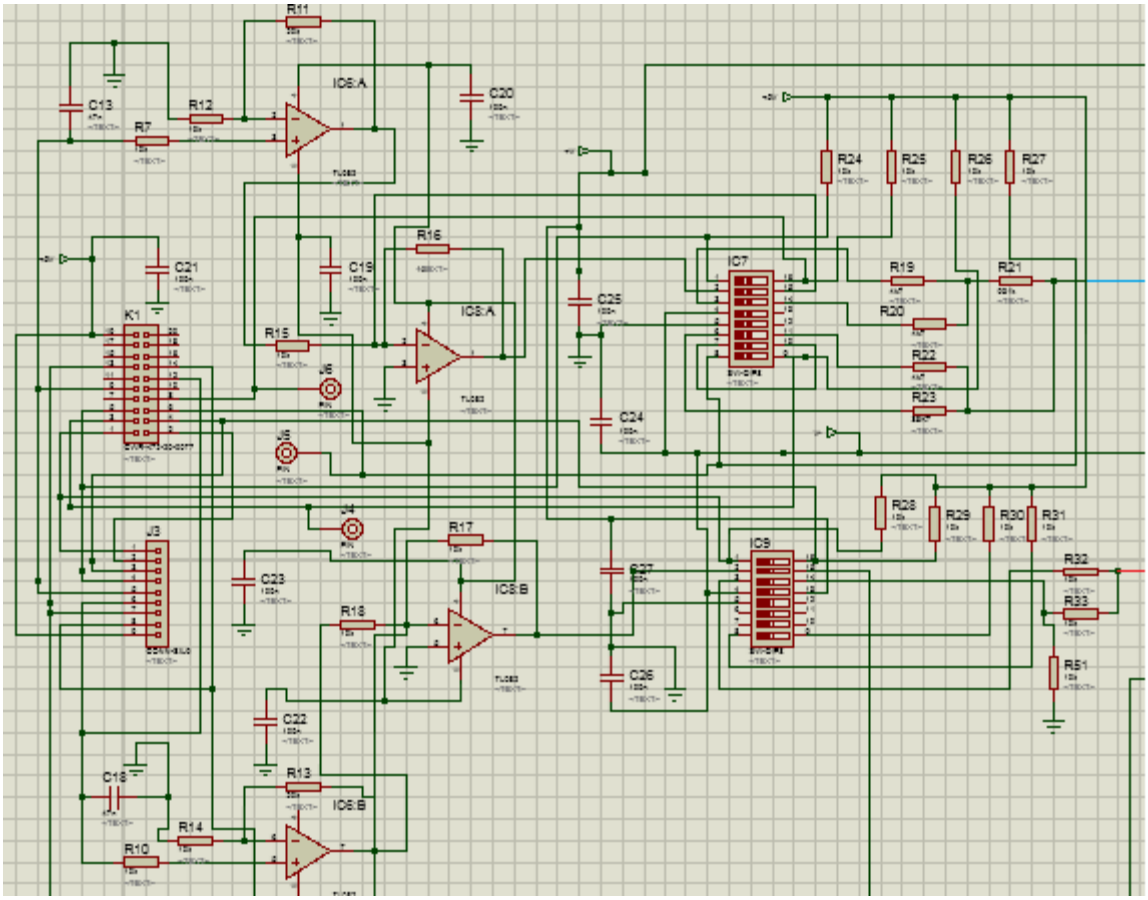

Figura 20: Etapa de entradas, salidas, amplificadores e inversores de placa de Potencia

Se puede observar principalmente:

- J3: Entradas y salidas analógicas y digitales de la placa de potencia.
- IC6/IC8: Amplificadores operaciones elevadores e inversores de tensión.
- IC7/IC9: Llaves analógicas DG221DJ comandados por la etapa de control.
- R24..R31: Resistencias de pull-up.
- En trazo de color celeste corriente de base tensión compuerta-fuente generada.
- En trazo de color rojo tensión inyectada al amplificador en configuración seguidor, luego esta tensión es inyectada a la etapa amplificadora de potencia de salida. (fuente de corriente).

3.2.2.5. Etapa amplificadora de potencia de salida (fuente de corriente)

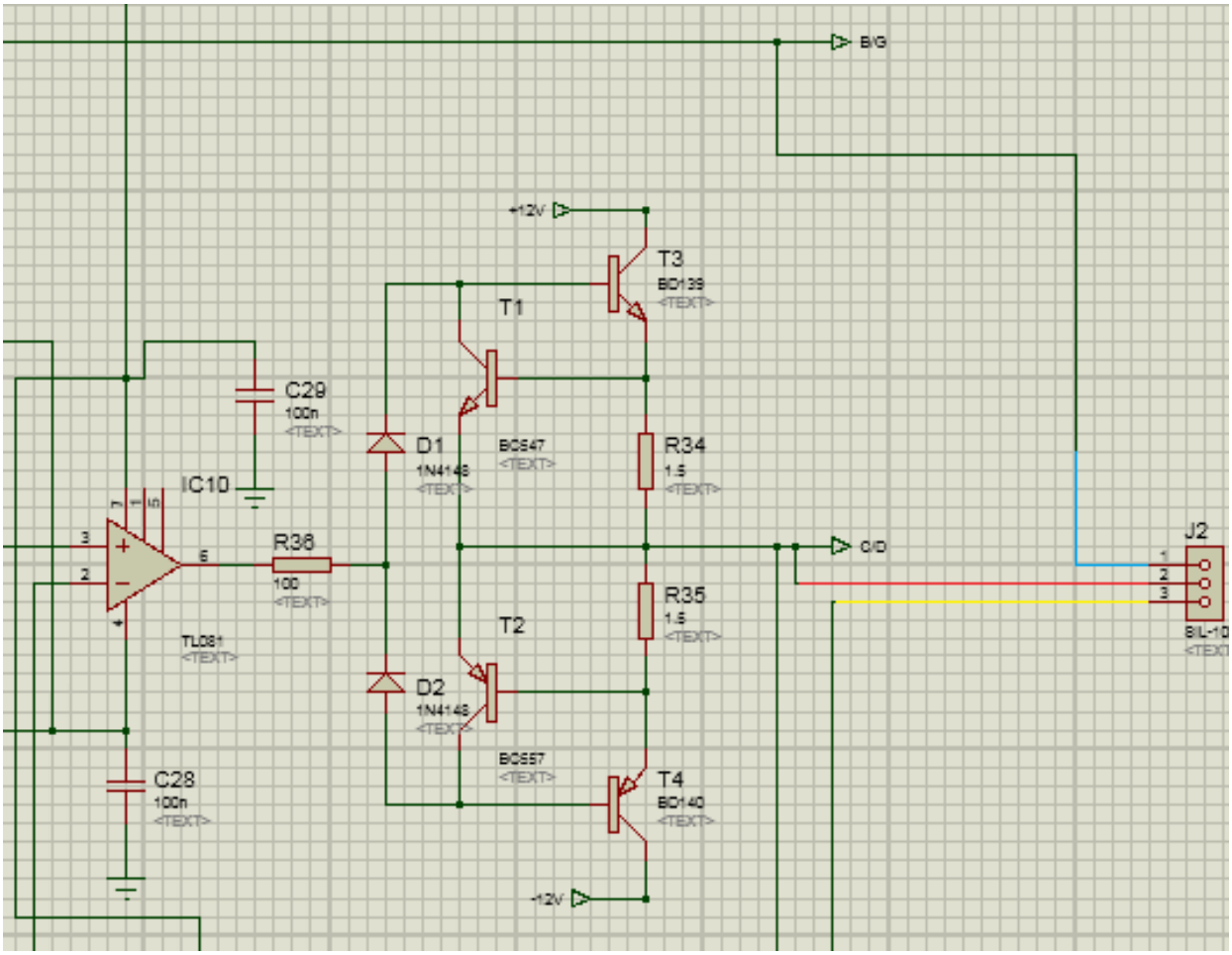

Figura 21: Etapa amplificadora de potencia de salida de placa de Potencia

Se puede observar principalmente:

- En trazo de color celeste corriente de Base o tensión de Compuerta-Fuente.
- En trazo de color rojo corriente de Colector o Drenador.
- En trazo de color amarillo corriente de Emisor o Fuente.

3.2.2.6. Etapa rectificadora y adaptadora de los niveles de tensión que serán inyectados en la placa de control STM32F4

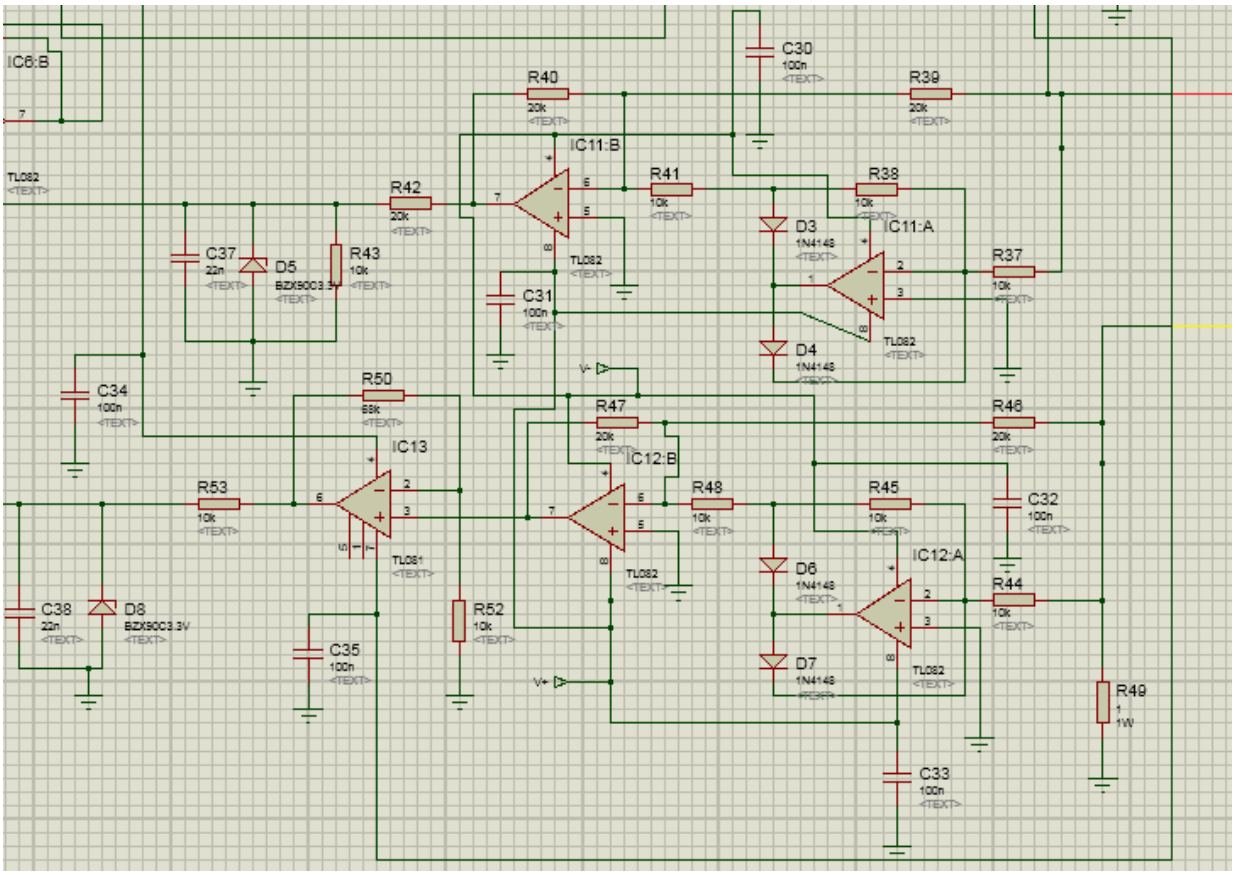

Figura 22: Etapa rectificadora y adaptadora de placa de Potencia

Se puede observar principalmente:

- En trazo de color rojo corriente de Colector o Drenador.
- En trazo de amarillo corriente de Emisor o Fuente.
- Amplificadores en configuración rectificador, amplificador elevador de tensión.
- D5/D8: Diodos limitadores de tensión máxima inyectada en la STM32F4.

3.3. Etapa de comunicación

Esta etapa logra el vínculo entre ambos kit de desarrollo, es de vital importancia que la comunicación se dé de manera fluida, de no ser así se perdería sincronismo imposibilitando el funcionamiento. La comunicación se realiza por medio del puerto RS-232 que posee los kit programable, con la siguiente configuración:

- Velocidad: 9600 baudios.
- Longitud de palabra: 8 bits.
- 1 bit de parada.
- Sin Paridad.
- Control de flujo de hardware desactivado.

Cabe destacar que no ha sido necesario la incorporación de conversores de adaptación.

Figura 23: Etapa de comunicación

Tanto los parámetros que se reciben en la placa STM32F4 como los resultados que se envían hacia la placa RASPBERRY PI3, tienen un "protocolo" que permite ser identificado por ambos extremos:

COMANDO	PARAMETRO	SIGNIFICADO	RESPUESTA DE STM32F4
B	0-255	Ancho de pulso para Ib o Vgs, rango 0..10V	-
C	0-255	Ancho de pulso para Vce o Vds, rango 0..10V	-
D	0=10µA 1=100 µA 2=1mA	Rango de corriente de base	-
E	0= pos 1= neg	Polaridad de Ib o Vgs	-
F	0= pos 1= neg	Polaridad de Vce o Vds	-
G	-	Consulta de valor actual de Vce o Vds	0…1024 corresponde al parámetro/100 en V.
H	-	Consulta de valor actual de Ic o Id	0…1024 corresponde al parámetro/2 en mA.
I	0-1024	Corriente de colector o drenador máxima = parámetro/2 en mA	
J	1 =NPN 2= PNP 3=N-MOSFET 4= P-MOSFET 5= N-JFET 6= P-JFET 7= DIODO	Configuración de dispositivo a ensayar	
K	0….10	Número de curvas	
L	-	Inicio de generación de curvas	

Figura 24: Tabla de parámetros de mensajes

3.4. Etapa de Control de interfaz de usuario (RASPBERRY PI 3)

Considerando que el proyecto aquí presentado es consecuencia de una fase previa en la cual se debía contar con una PC para su utilización, se optó por que esta nueva etapa sea capaz de lograr un equipo totalmente autónomo, es decir, que no se necesite de ningún dispositivo extra para su funcionamiento. Es por esto que se añadió al equipo Trazador un kit de desarrollo RASPBERRY PI 3. Esta poderosa placa fue utilizada para implementar, entre otras cosas, una interfaz de usuario para el mando local desde un pantalla LCD y un servidor web para acceso remoto.
Entre sus principales características se destacan:

- 4× ARM Cortex-A53, 1.2GHz.
- 1 Gb de memoria RAM.
- 40 pines de propósito general.
- 4 puertos USB.
- Ethernet.
- WIFI integrado.
- Bluetooth 4.1 baja energía BLE.

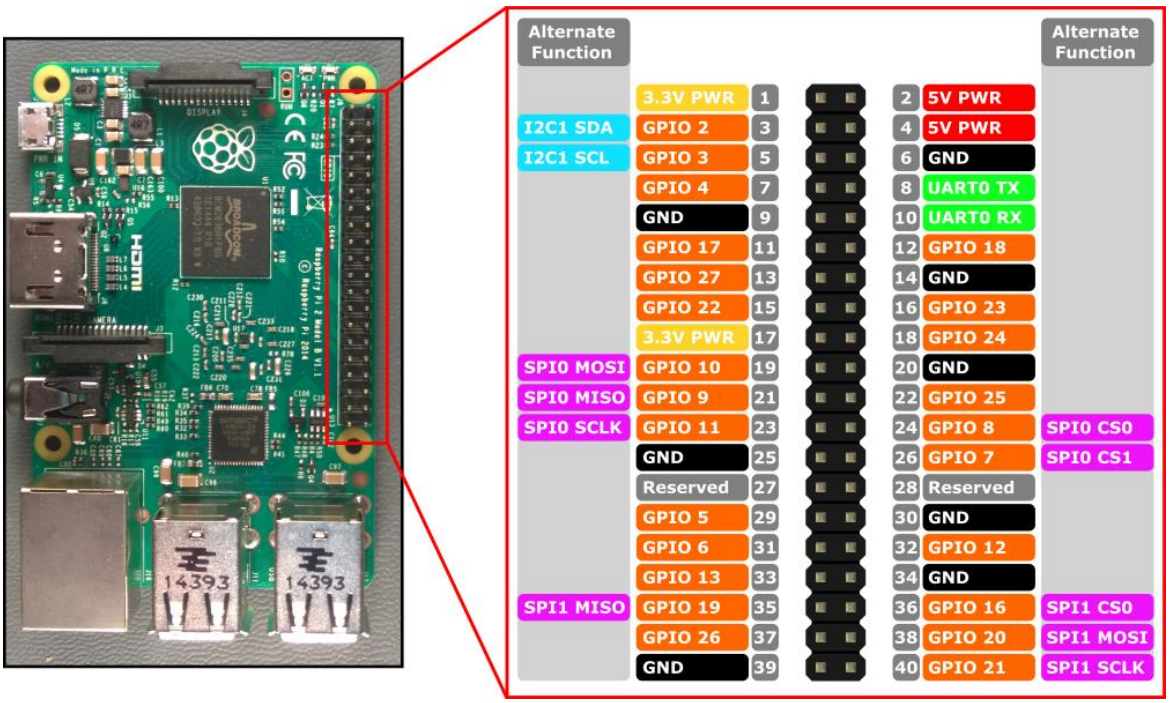

Figura 25: Componentes de RASPBERRY PI 3

Las interfaces de hardware para RASPBERRY PI 3 están expuestas a través del grupo de 40 pines de la placa. Este kit también tiene la capacidad de conexión con un monitor vía un puerto HDMI. Podemos conectarle nuestro monitor, televisor o incluso pantallas táctiles.

- HDMI.
- Jack de 3.5mm para salida de audio.
- Puertos USB.
- Puertos para cámara.
- 1 bus I2C.
- 2 bus SPI.
- **1** UART en serie (RPi3 solo incluye mini UART).
- 24 Pines entra/salida.
- Wifi.
- Bluetooth.
- **2** Pines de alimentación de 5V.
- **2** Pines de alimentación de 3.3V.
- **8** Pines de tierra.

3.4.1. Sistemas operativos compatibles

La RASPBERRY PI no es compatible con los sistemas operativos que todos usamos en nuestros ordenadores, esto es debido a que poseen arquitectura ARM y no x86-64 como los ordenadores normales que todos utilizamos. No obstante, los desarrolladores han conseguido crear sistemas operativos orientados específicamente a la RASPBERRY PI, basándose en distribuciones de escritorio y servidores tan populares como Debian, Arch Linux, o Windows.

3.4.1.2. Elección de sistema operativo

Figura 26: Sistema operativo de RASPBERRY PI 3

Como se mencionó anteriormente el proyecto Trazador es el sucesor de un sistema no autónomo en el cual se debía contar con la presencia de una PC para el trazado de las curvas, en aquella etapa la interfaz gráfica montada en la PC fue desarrollada en Visual Basic bajo el Sistema Operativo Windows. Dicha herramienta, a pesar de no ser de las más actuales, poseía todo lo necesario para que la primer versión del Trazador funcione correctamente. El conocimiento y los resultados obtenidos con este software, dieron como resultado una tendencia a favor de seguir en un entorno similar. Es por esto precisamente que se optó por elegir para esta etapa final, que nuestra RASPBERRY PI funcione bajo la plataforma Windows. Se presentaron muchos inconvenientes, principalmente por la falta de experiencia en esta nueva placa tanto en el manejo del hardware como de su software.

Para el desarrollo de nuestra aplicación y Servidor Web decidimos utilizar el sistema operativo propuesto por Windows. Se denomina Windows 10 IoT Core y es una versión muy limitada de Windows 10 que no cuenta con muchas de las opciones de su hermano mayor porque, simplemente, no las necesita. Para empezar, se trata de una edición para arquitecturas de 32 bits y concretamente para la ARMv7.
En Windows 10 IoT Core nos encontramos con una plataforma hecha por desarrolladores y para desarrolladores.
Esta plataforma invita a los desarrolladores a crear e inventar todo tipo de soluciones adaptadas a la Internet de las Cosas y no centrarlas en el segmento de los PCs y portátiles tradicionales. A pesar de que a medida que fuimos investigando sobre este nuevo entorno nos dimos cuenta que era totalmente diferente a lo que conocíamos, con mucho esfuerzo y luego de leer muchos foros y manuales, logramos al fin que el Trazador funcione de manera autónoma bajo esta nueva plataforma.

3.4.1.3. Limitaciones de Windows 10 IoT

Windows 10 IoT permite construir aplicaciones modernas apuntando a los dispositivos de Windows 10 tales como PC, tablet, teléfono, etc.
La Plataforma universal de Windows (conocida generalmente como *UWP*) es una plataforma común de aplicaciones presentes en todos los dispositivos que cuentan con Windows 10 .

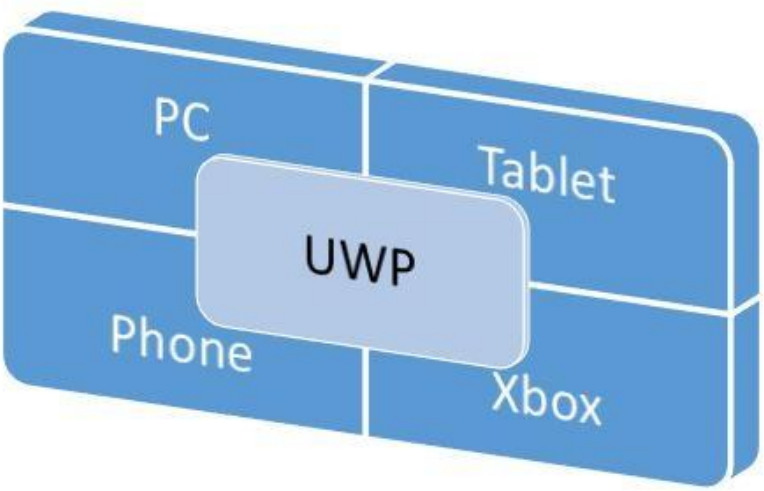

Figura 27: Plataforma universal de Windows en RASPBERRY PI 3

Ahora bien, insistiendo nuevamente en que nuestro equipo debía ser accesible desde cualquier dispositivo sin importar el Sistema Operativo que se posee, se nos presentaba una gran limitación ya que no todos los usuarios dispondrían del sistema operativo de Windows 10. Por este motivo fue necesario incorporar un servidor web adicional a nuestra aplicación de forma tal que sea posible acceder al equipo en tiempo real a partir de una página web convencional.
Este programa forma la base para controlar la RASPBERRY PI. Al ejecutar un servidor web simple en este kit, podemos enviar datos al servidor web desde cualquier dispositivo a

través de HTML. Dicho servidor escuchará la entrada en un puerto en particular. El mismo se ejecutará como un servicio y proporcionará una página web que contiene todos los controles. Dicha página se mantiene aislada de la programación principal.

3.4.2. Desarrollo de interfaz gráfica

Como se mencionó anteriormente, el kit RASPBERRY PI 3, posee la capacidad de brindar al usuario una aplicación gráfica y una página web, motivo por el cual fue necesario utilizar varios lenguajes de programación. En el caso de la aplicación se programó en lenguaje C#. A la hora de realizar la página web, el código adoptado fue HTML5 y JAVA scripts.
Buscando el punto de equilibrio entre la operatividad o la complejidad para el usuario, el entorno gráfico se diseñó con una serie de botones, una barra lineal de ajuste, indicadores y un área de graficación.
Por medio de la interfaz gráfica de puede seleccionar el tipo de semiconductor a ensayar, la máxima tensión y corriente a la cual será sometido el mismo y la cantidad de trazos deseados.
Como información adicional de gran interés para el usuario, la interfaz presenta dos indicadores numéricos los cuales brindan la ganancia del componente y el valor de la máxima corriente seleccionada. El área mayor de la ventana se encuentra ocupada por la zona de graficación, allí se muestran las curvas, y valores de corrientes y tensiones alcanzadas durante el ensayo.
Cabe destacar que en caso de realizar una conexión errónea o someter al ensayo un componente defectuoso el software detiene el trazado y comunica por medio de un mensaje cual fue el motivo de la interrupción.

Figura 28: Interfaz gráfica en RASPBERRY PI 3

Otro punto importante a considerar es que la aplicación fue desarrollada utilizando distintas librerías que proporcionan herramientas muy interesantes como son el ajuste automático de escala para las distintas plataformas. Esto permite que el equipo pueda ser conectado a cualquier pantalla a través de su puerto HDMI instalado en la tapa trasera del mismo. Es decir, además de su propia pantalla y acceso web, el Trazador tiene la posibilidad de conectarse a un monitor o proyector externo.

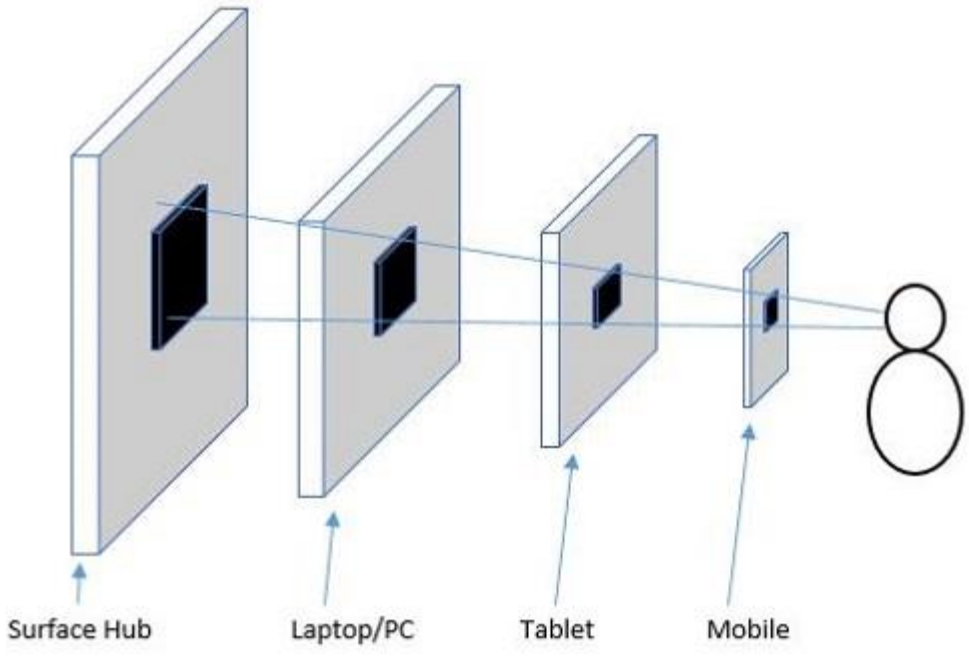

Figura 29: Adaptación de plataformas en RASPBERRY PI 3

3.4.3. Desarrollo de servidor Web.

El servidor implementado escucha un puerto único, los comandos recibidos de la página web, es decir, las decisiones tomadas por un cliente web, son pasadas al trazador.
La simplicidad del programa se debe a que la página web está escrita con puro HTML5, CSS3 y Jquery. Para el diseño de la misma se utilizó el software Dreamweaver.
Por otro lado, el servidor web y la aplicación local encargada de la graficación de las curvas, fueron programados en Visual Studio 2015, utilizando lenguaje Visual C#.
La página web está compuesta por dos archivos HTML y JAVA scripts básicos denominados cuerpo y encabezado respectivamente. Para la hoja de estilo, es decir, para lograr la estética deseada en la página, también hay un archivo CSS.
El proyecto completo que incluye la aplicación local y el servidor web mencionado han sido montados en el kit de desarrollo RASPBERRY PI 3.

3.4.3.1. Protocolo TCP/IP

3.4.3.1.2. Modelo de capas

El modelo TCP/IP estructura el problema de la comunicación en cinco capas relativamente independientes entre sí:

- Capa física.
- Capa de acceso a la red.
- Capa de internet.
- Capa extremo-a-extremo o de transporte.
- Capa de aplicación.

3.4.3.1.3. Capa física

Define la interfaz física entre el dispositivo de transmisión de datos (por ejemplo, la estación de trabajo o computadora) y el medio de transmisión o red. Esta capa se encarga de la especificación de las características del medio de transmisión, la naturaleza de las señales, la velocidad de datos y cuestiones afines.
Para la realización del servidor web se utilizó un kit didáctico RASPBERRY PI3. La capa física de dicho dispositivo se encuentra basada en el chip LAN9514-JZX, el mismo es un concentrador USB 2.0 y un controlador Ethernet 10/100, además de poseer cuatro conexiones USB y conexión Ethernet, la RASPBERRY brinda la posibilidad de conectare a una red WI-FI (802.11 b / g / n LAN inalámbrica).
Entre las diferentes características del chip LAN9514-JZX podemos encontrar:

- Protección contra ESD incorporada de ± 8kV / 15kV en descargas de aire y USB en PHYs USB y Ethernet.
- 24MHz Reloj proporcionado para conectar concentradores USB adicionales.
- MAC Ethernet 10/100 integrado con soporte dúplex completo.
- PHY Ethernet 10/100 integrado con HP Auto-MDIX.
- Implementa modos de operación de potencia reducida.
- Resistencias pull-up / pull-down de terminación USB integradas.
- Protección interna de cortocircuito de las clavijas de señal diferencial USB.
- Soporte completo y semi-dúplex con control de flujo.
- Soporta el funcionamiento con bus y autoalimentado.

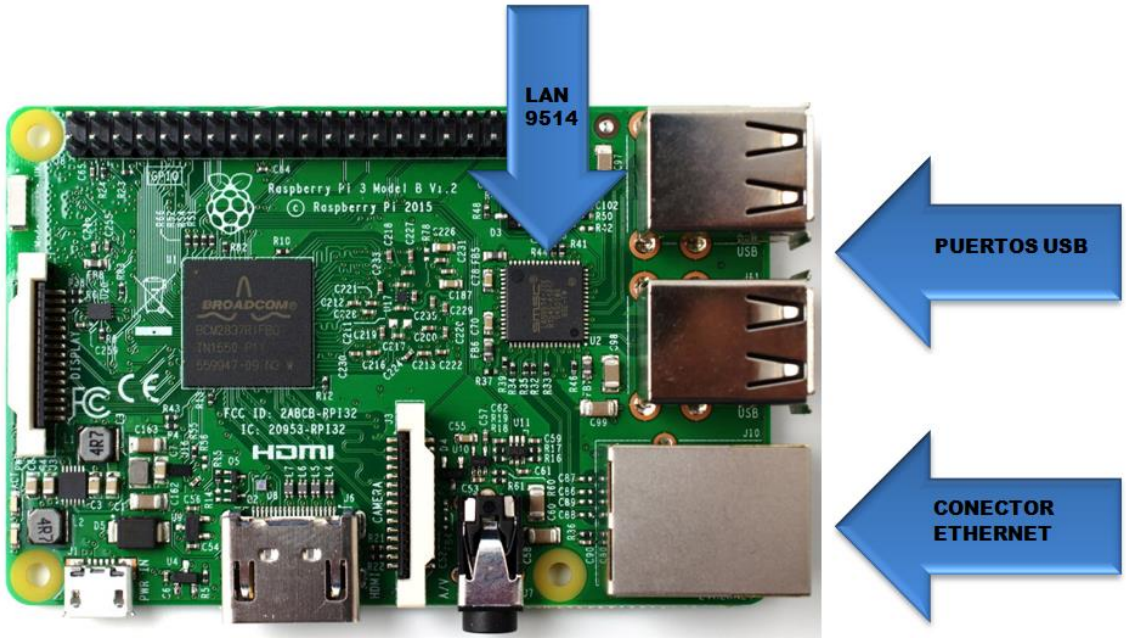

Figura 30: Periféricos RASPBERRY PI 3

3.4.3.1.4. Capa de acceso a la red

Es responsable del intercambio de datos entre el sistema final (servidor, estación de trabajo, etc.) y la red a la cual está conectado. El emisor debe proporcionar a la red la dirección del destino, de tal manera que ésta pueda encaminar los datos hasta el destino apropiado. El emisor puede requerir ciertos servicios que pueden ser proporcionados por el nivel de red, por ejemplo, solicitar una determinada prioridad. El software en particular que se use en esta capa dependerá del tipo de red que se disponga. Así, se han desarrollado, entre otros, diversos estándares para la conmutación de circuitos, la conmutación de paquetes (por ejemplo, retransmisión de tramas) y para las redes de área local (por ejemplo, Ethernet). Por tanto, tiene sentido separaren una capa diferente todas aquellas funciones que tengan que ver con el acceso a la red.
Para sistema finales conectados a la misma red, la capa de acceso a la red está relacionada con el acceso y encaminamiento de los datos.

3.4.3.1.4. Capa de internet

Posee los procedimientos necesarios para que dispositivos conectados en diferentes redes se puedan comunicar, es decir, logra que los datos puedan atravesar las distintas redes interconectadas.

3.4.3.1.5. Capa de transporte

Independientemente de la naturaleza de las aplicaciones que estén intercambiando datos, es usual requerir que los datos se intercambien de forma fiable. Esto es, sería deseable asegurar que todos los datos llegan a la aplicación destino y en el mismo orden en el que fueron enviados. Los mecanismos que proporcionan esta fiabilidad son esencialmente independientes de la naturaleza intrínseca de las aplicaciones. Por tanto, tiene sentido agrupar todos estos mecanismos en una capa común compartida por todas las aplicaciones; ésta se denomina capa extremo-a-extremo, o capa de transporte. El protocolo para el

control de la transmisión, TCP (*Transmission Control Protocol*), es el más utilizado para proporcionar esta funcionalidad.

3.4.3.1.6. Capa de aplicación

Esta capa contiene toda la lógica necesaria para posibilitar las distintas aplicaciones de usuario. Para cada tipo particular de aplicación, como por ejemplo, la transferencia de archivos, se necesitará un módulo bien diferenciado.
El trazador posee un servidor web simple, el mismo se encuentra corriendo en la RASPBERRY PI3 y es capaz de almacenar una página simple en HTML, la cual se pone a disposición de los clientes. El servidor escucha la entrada del puerto 8090. Este servidor se ejecuta como un servicio y proporciona una página web. Desde la página web es posible modificar los parámetros del trazado.

3.4.3.2. Arquitectura

Este proyecto, más precisamente la comunicación que posee el mismo, sigue una estructura del tipo cliente – servidor. Considerando la capacidad de que el cliente sea cualquier host (PC, celular, tablet, Mac…) fue necesario escribir un programa que pueda correr en cualquier software utilizado para navegar en la web, es decir, por ejemplo, Chrome, Navigator, Opera, Explorer…es decir, que la comunicación entre el software del servidor y el software del cliente tenía que ser lo más versátil posible. La comunicación entre el cliente y el servidor se da a través de una red, brindando la posibilidad de que el usuario manipule el trazador sin necesidad de instalar ningún software adicional. Lo anteriormente mencionado brinda la capacidad de que cualquier persona, sin importar que dispositivo tenga, pueda conectarse al trazador y manejarlo desde su explorador web.

3.4.3.2.1. Arquitectura cliente-servidor

Al encender el dispositivo, una vez que la aplicación se encuentra visible, el servidor web estará activo y publicará la última curva trazada. La conexión se puede realizar con un cable Ethernet, solo basta configurar el host cliente en modo dinámico y acceder al explorador web colocando en el mismo el IP del trazador. Cabe destacar que el IP del trazador es fijo.
Otra de las posibilidades que brinda el trazador es la de permitir un manejo del mismo inalámbricamente al poseer este conexión WI-FI, con solo conectarlo a una red existente y configurando el router de manera correcta, se puede utilizar el trazador desde un host, un punto importante a considerar es que el host cliente debe estar conectado a la misma red. Esto permite agilizar notablemente la utilización del dispositivo en un ámbito educativo donde puede existir más de un alumno que necesite utilizar el dispositivo, por ejemplo.

3.4.3.3. Servidor Web implementado en Raspberry

Debido a que un servidor web sólo necesita ejecutarse en segundo plano, sin ninguna pantalla, creamos una aplicación de fondo simple. En VS2015, primero creamos una solución en blanco y la llamamos "Trazador".

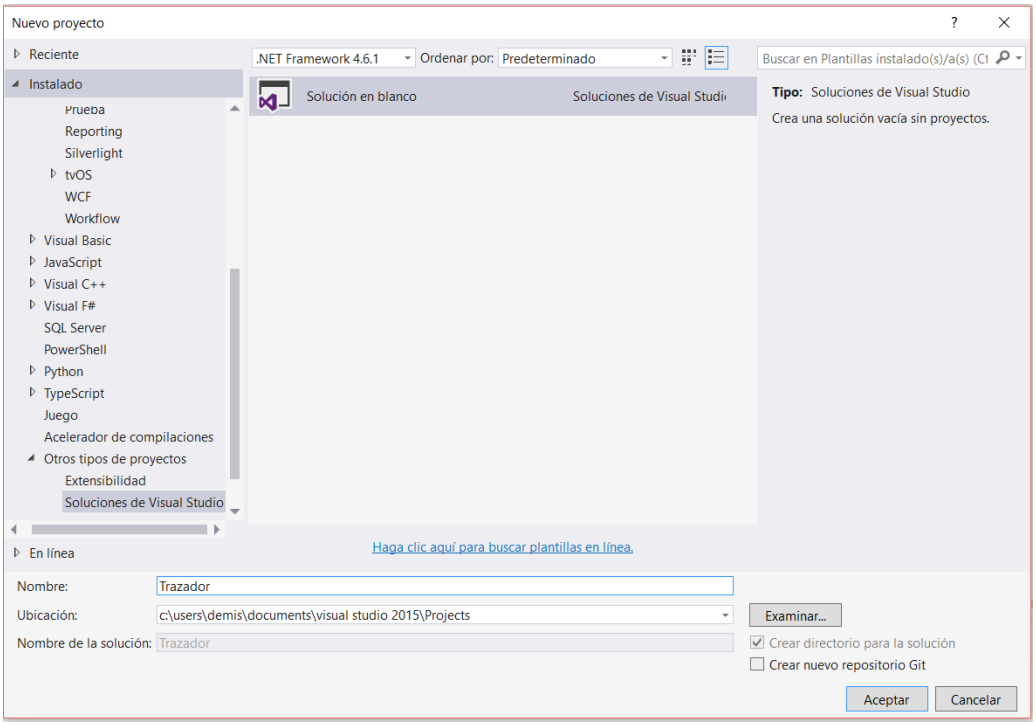

Figura 31: Creación de solución en Visual Studio 2015

A continuación, añadimos al proyecto el servidor, con el nombre WebService.

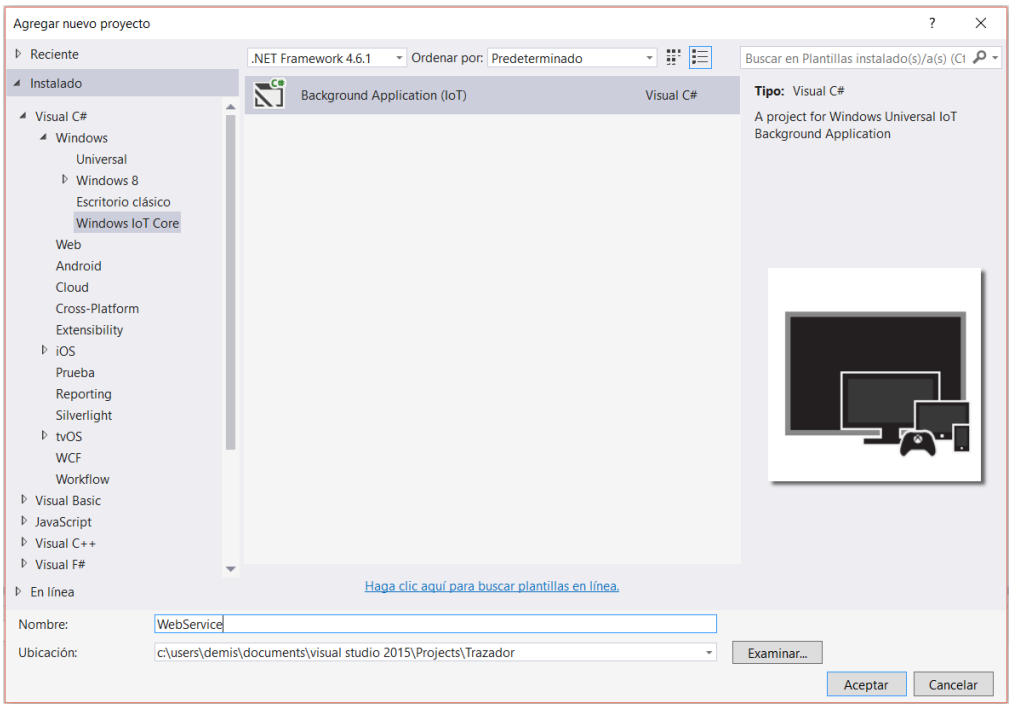

Figura 32: Creación de servidor en Visual Studio 2015

Se agregó al proyecto Trazador una clase C# como WebService.

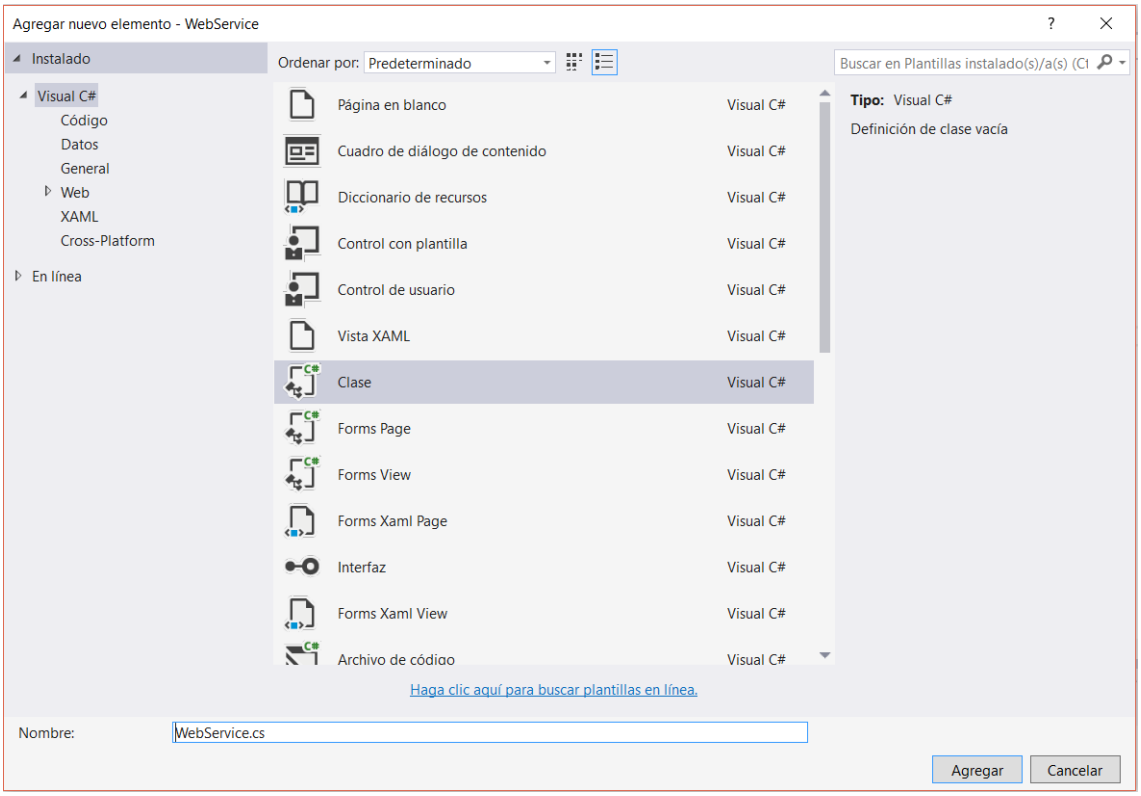

Figura 33: Creación de clase C# - servidor en Visual Studio 2015

Para que funcione de acuerdo a nuestros requerimientos fue necesario realizar algunas modificaciones al proyecto base que proporciona VS2015:

Se debió modificar la capacidad del paquete para que se ejecute como servidor y como servicio. Para esto se eliminaron algunas líneas del código original:

```
Category="windows.backgroundTasks"EntryPoint="WebService.StartupTask">
<BackgroundTasks>
<iot:Task Type="startup" />
</BackgroundTasks>
```

Figura 34: Eliminación de líneas - servidor en Visual Studio 2015

Se reemplazaron con el código siguiente. Esto permite que funcione como servicio.

```
<uap:ExtensionCategory="windows.appService"EntryPoint="WebService.StartupTask">
<uap:AppServiceName="com.brahas.WebService" />
</uap:Extension>
Now to extend the capability to run as a service add the following line to <Capabilities>
<CapabilityName="internetClientServer" />
With that addition, <Capabilities> will look like this.
<Capabilities>
<CapabilityName="internetClient" />
<CapabilityName="internetClientServer" />
</Capabilities>
Extension>
```

Figura 35: Reemplazo de líneas - servidor en Visual Studio 2015

En el código se agregaron las siguientes dos variables privadas.

```
privateBackgroundTaskDeferral deferral;
privateAppServiceConnectionasc;
```

Figura 36: Agregado de líneas - servidor en Visual Studio 2015

A continuación, agregamos el código siguiente al método Run:

```
deferral = taskInstance.GetDeferral();
varws = new WebService();
var td = taskInstance.TriggerDetails as AppServiceTriggerDetails;
asc = td.AppServiceConnection;
ws.SetConnection (asc);
awaitThreadPool.RunAsync(wi =>
{
ws.Start();
});
```

Figura 37: Agregado de líneas en método Run - servidor en Visual Studio 2015

La variable AppServiceConnection almacena la conexión establecida desde la aplicación Trazador. Esta conexión se utiliza para enviar comandos recibidos de la página web a la aplicación Trazador.

Por otro lado, creamos un StreamSocketListener y lo vinculamos a un puerto único. Debido a que la interfaz IoT utiliza 8080, decidimos utilizar el puerto 8090. Se puede utilizar cualquier puerto que no entre en conflicto con los servicios existentes. El servidor web estará escuchando este puerto. Agreguemos un evento ConnectionReceived, el servidor web escucha este puerto y ante una solicitud, proporcionará una página web al navegador.

```
privateasyncvoidListener_ConnectionReceived(StreamSocketListener sender,
StreamSocketListenerConnectionReceivedEventArgsargs)
{
if (!bMsg)
{
bMsg = true;
StringBuilder request = newStringBuilder();
try
{
using (IInputStream input = args.Socket.InputStream)
{
byte[] data = newbyte[BUFFERSIZE];
IBuffer buffer = data.AsBuffer();
uintdataRead = BUFFERSIZE;
while (dataRead == BUFFERSIZE)
{
awaitinput.ReadAsync(buffer, BUFFERSIZE, InputStreamOptions.Partial);
request.Append(Encoding.UTF8.GetString(data, 0, data.Length));
dataRead = buffer.Length;
}
```

```
}
stringinputString = GetQuery(request);
if (serviceConnection != null)
awaitserviceConnection.SendMessageAsync(newValueSet { newKeyValuePair<string,
object>("Query", inputString) });
OutputWebPage(args, inputString);
}
catch (Exception Ex)
{
Debug.WriteLine("Exception occurred: {0}", Ex);
bMsg = false;
}
bMsg = false;
}
```

Figura 38: Evento de ConnectionReceived - servidor en Visual Studio 2015

Usamos una variable como flag (flagbMsg). Esto evita que se envíen mensajes múltiples al Navegador los cuales podrían ocasionar errores. El método GetQuery quita la entrada del explorador para métodos POST o GET y devuelve la consulta. Las solicitudes se descomponen utilizando espacios.

```
...
.
varrequestLines = request.ToString().Split(' ');
if (requestLines[0] == "POST")
{
returnrequestLines[requestLines.Length - 1];
}
data = requestLines.Length>1 ? requestLines[1] : "Unregistered".ToString();
.
...
```

Figura 39: Método POST - servidor en Visual Studio 2015

El mensaje así recuperado se envía a serviceConnection: ésta es la aplicación que inicia el servicio.

```
if (serviceConnection != null)
awaitserviceConnection.SendMessageAsync(newValueSet { newKeyValuePair<string,
object>("Query", inputString) });
```

Figura 40: Aplicación ServiceConnection - servidor en Visual Studio 2015

En el método "Start" definimos las siguientes variables

```
headerFile = File.ReadAllText("web\\header.html");
cssFile = File.ReadAllText("web\\theme.css");
bodyFile = File.ReadAllText("web\\body.html");
```

Figura 41: Método Start - servidor en Visual Studio 2015

Por último, OutputWebPage concatena estos archivos, cssFile, bodyFile (denotando encabezado, hoja de estilo y cuerpo de HTML) y devolverá la salida apropiada.

```
...
.
byte[] bodyArray = Encoding.UTF8.GetBytes(
$"<!DOCTYPE
html>\n<html>\n<head>{headerFile}<style>{cssFile}</style>\n</head>\n<body>{bodyFi
le}<p>Feedback Received: <spanid=\"retVal\">{data}</span></p>\n</body>\n</html>");
varbodyStream = new MemoryStream(bodyArray);
var header = "HTTP/1.1 200 OK\r\n" +
$"Content-Length: {bodyStream.Length}\r\n" +
"Connection: close\r\n\r\n";
.
...
```

Figura 42: Composición de página Web - servidor en Visual Studio 2015

El intervalo de etiqueta con id = "retVal" le permite proporcionar una retroalimentación a la página web. Los datos en sí se pueden enviar desde la aplicación. Si comprobamos la secuencia de comandos Jquery en el archivo de encabezado, veremos el código siguiente:

```
$(".btnjs").click(function () {
var ran = Math.random();
var name = this.name;
$("#log").text(name + ": Clicked");
$.get("", { [name]: "Clicked_Action " + ran}, function (result) {
$("#retVal").text(result);
});
});
```

Figura 43: Retroalimentación para página Web - servidor en Visual Studio 2015

El servidor programado de este modo permite la conexión de varios clientes en simultáneo. Por ejemplo, es posible que la configuración de los parámetros del ensayo se realice entre distintas plataformas (web y aplicación local) o entre varios usuarios.

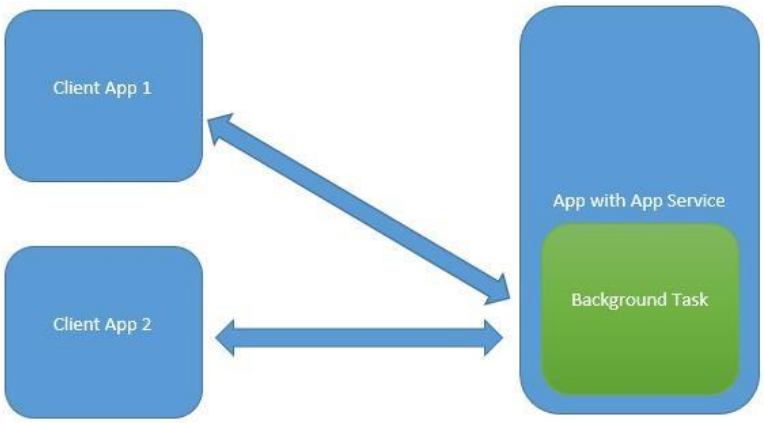

Figura 44: Conexión de clientes - servidor en Visual Studio 2015

3.4.4. Desarrollo de página web

Para realizar el diseño de la página web se optó por el software Dreamweaver 2017 debido a que el mismo es fácil de utilizar y además posee todas las herramientas necesarias para realizar un entorno web amigable y de buen aspecto.
El sitio web está programado en HTML5, además se utilizó CSS y jQuery para lograr un mejor diseño y una más amplia funcionalidad.

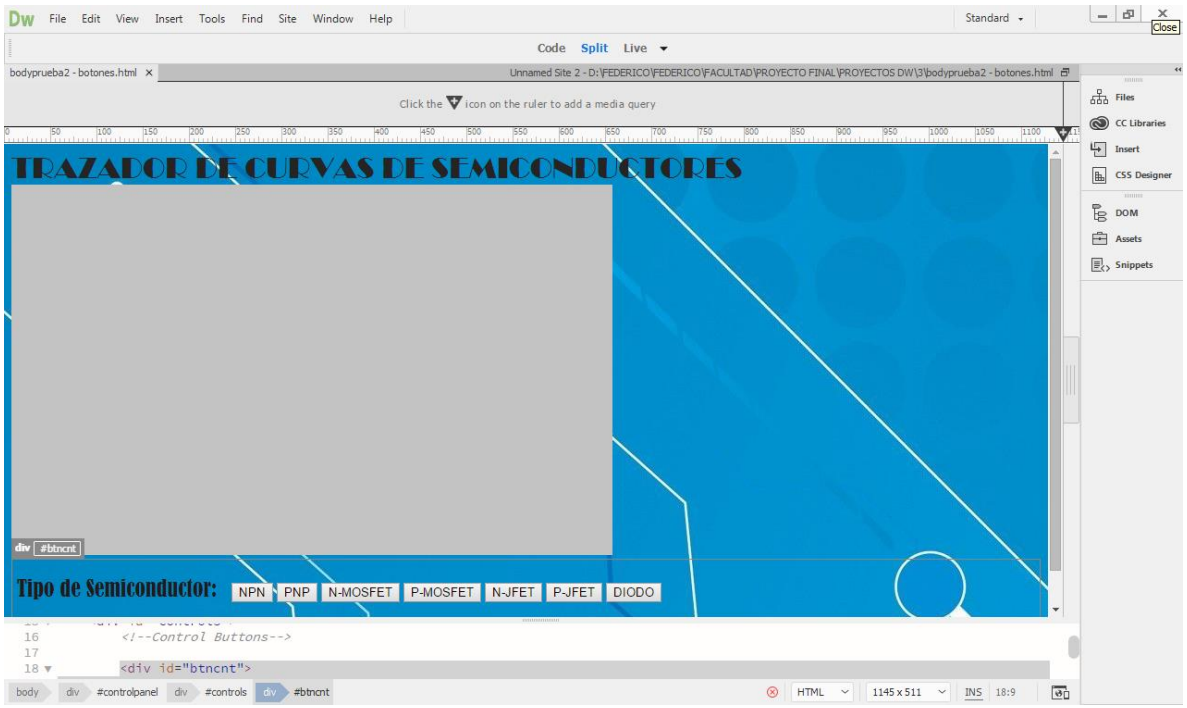

Figura 45: Diseño de página Web

El servidor web utilizado en este proyecto permite implementar el código JAVA script y el HTML en archivos separados denominados Header y Body respectivamente. A la hora de publicar la gráfica dinámica, es decir, que se actualice luego de cada trazado, nos encontramos con una dificultad, nuestro servidor web no poseía la capacidad de mostrar una imagen dinámicamente una vez que la página sea publicada, en otras palabras, solo publicaba imágenes codificadas en base64, las cuales deberían formar parte del propio código HTML, por tal motivo, cada gráfica debería estar codificada y formar parte del archivo Body. Para afrontar el inconveniente mencionado fue necesario lograr que la app graficadora. que se encuentra corriendo en la RASPBERRY PI, ponga a disposición del servidor web una imagen codificada, es decir, luego de trazar la curva y guardarla en un directorio, la aplicación debía reutilizar esta imagen, codificarla y dejarla a disposición del servidor. Para que esto sea posible se utilizó el siguiente código, cabe destacar que la programación de la app se realizó en C#:

```
StorageFolder storageFolder = KnownFolders.SavedPictures;
                              var file = await storageFolder.CreateFileAsync("Curva.jpg",
CreationCollisionOption.OpenIfExists);

                              byte[] fileBytes = null;
                              using (var stream = await file.OpenReadAsync())
                              {
                                  fileBytes = new byte[stream.Size];
                                  using (var reader = new DataReader(stream))
```

```
                {
                    await reader.LoadAsync((uint)stream.Size);
                    reader.ReadBytes(fileBytes);
                }
            }

            byte[] imageArray = fileBytes;
            string base64ImageRepresentation =
Convert.ToBase64String(imageArray);

            imagen = "" + base64ImageRepresentation;
```

Figura 46: Creación de imagen para gráfica dinámica

Una vez que se tenía la imagen codificada era necesario rearmar el código HTML completo, la página web debía ser publicada desde la primera línea de código hasta la última. Para lograr armar el código HTML se optó por subdividir el archivo Body en tres partes. La primera denominada Body1 es un archivo compuesto por puro HTML que posee las líneas de código anterior a la publicación de la imagen:

```
<body background=
"..... <!----- imagen de
fondo codificada en base64 ----->


<font color="#262525" size="+3" face="Broadway">TRAZADOR DE CURVAS DE
SEMICONDUCTORES</font><br>
    <img src="
```

Figura 47: Creación de primera parte del cuerpo de página Web

Como se mencionó en las líneas anteriores, el servidor web debe concatenar todo el código HTML, rearmar la página y publicarla, es decir que además de poseer la imagen codificada, fue necesario implementar un archivo denominado Body2 el cual posee todo el código HTML posterior a la publicación de la imagen:

```
" />


    <div id="controlpanel">
        <!--Start of Control Panel-->
        <div id="controls">
            <!--Control Buttons-->

            <div id="btncnt">

                <button class="semnpn" name="ButtonNpn">NPN</button>
                <button class="sempnp" name="ButtonPnp">PNP</button>
                <button class="semn-mosfet" name="ButtonN-mosfet">N-MOSFET</button>
                <button class="semp-mosfet" name="ButtonP-mosfet">P-MOSFET</button>
                <button class="semn-jfet" name="ButtonN-jfet">N-JFET</button>
                <button class="semp-jfet" name="ButtonP-jfet">P-JFET</button>
                <button class="semdiodo" name="ButtonDiodo">DIODO</button>
                <!--<button class="semamp-op" name="ButtonAmp-op">AMP-OP</button>-->
                <button type="button" class="btn-danger">TRAZAR CURVA</button>
                <button type="button" class="btn-warning">GUARDAR CURVA</button>
            <!--    <button type="button" class="btn-success">Success Button</button>-->
            </div>
            <div id="btnrange">


                <div id="btnsc">
                    <br>

                    <table width="200" border="1">
                        <tbody>
                            <tr>
                                <td>
                                    <input class="sldrclass" id="Slider1" type="range"
min="100" max="500" step="10" />
```

```
                                    <font color="#221B1B" size="+2" face="Bernard MT
Condensed">    Tension  </font>
                                </td>
                                <td>
                                    <input class="sldrclass" id="Slider2" type="range"
min="100" max="500" step="10" />
                                    <font color="#221B1B" size="+2" face="Bernard MT
Condensed">    Corriente    </font>
                                </td>
<td><img src="…
<!----- imagen LA codificada en base64 ----->
</tr>

                        </tbody>
                    </table>

                </div>


            </div>


        </div>

    </div>
```

Figura 48: Creación de segunda parte del cuerpo de página Web

Luego de obtener la imagen codificada, el servidor web debía unificar todo el código HTML proveniente del archivo body1, body2 y la imagen codificada para luego poder publicar la página completa con la nueva curva, para ello fue necesario implementar las siguientes líneas de código:

```
BODY = string.Concat(string.Concat(bodyFile1, imagen), bodyFile2);

                            byte[] bodyArray = Encoding.UTF8.GetBytes(
                                $"<!DOCTYPE
html>\n<html>\n<head>{headerFile}<style>{cssFile}</style>\n</head>\n<body>{BODY}<p>Feedback
Received: <span id=\"retVal\">{data}</span></p>\n</body>\n</html>");
```

Figura 49: Composición de página Web

La página web utilizada pone a disposición del usuario una serie de botones que permiten modificar opciones del trazado y a su vez brinda la posibilidad de trazar la curva propiamente dicha desde la página con un botón denominado “Trazar curva”. Para que la aplicación pueda interpretar los botones disponibles en la página web, se utilizó código JavaScript, el mismo se encuentra en un archivo denominado Header, el cual posee todas las funciones necesarias relacionadas con cada botón pulsado. El servidor web también debe incluir este archivo en la página web, es decir, todo el código JAVA script se encuentra disponible al momento de publicar la página, permitiendo que la aplicación se comunique con la página e interprete el mensaje correspondiente al pulsado de cada botón:

```
<meta charset="utf-8" />
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<title>Trazador de curvas de Semiconductores</title>
<script>

    $(document).ready(function () {

        $(".btnjs").click(function () {
```

```
            var ran = Math.random();
            var name = this.name;
            $("#log").text(name + ": Clicked");
            $.get("", { [name]: "Clicked_Action " + ran}, function (result) {
                $("#retVal").text(result);
            });
        });

           $(".btn-danger").click(function () {
            var name = this.name;
            var ran = Math.random();
            $("#log").text(name + ": Clicked");
            $.get("", { [name]: "Clicked_Action " + ran }, function (result) {
                $("#retVal").text(result);
            });
        });

        $(".btn-warning").click(function () {
            var name = this.name;
            var ran = Math.random();
            $("#log").text(name + ": Clicked");
            $.get("", { [name]: "Clicked_Action " + ran }, function (result) {
                $("#retVal").text(result);
            });
        });

        $(".btn-success").click(function () {
            var name = this.name;
            var ran = Math.random();
            var ckb = $("input[name=" + name + "]").is(':Checked') ? "Checked_Action " +
ran : "UnChecked_Action " + ran;
            $("#log").text(name + ckb);

            //alternatively ->  ckb = $(".btnin::checked").length
            $.get("", { [name]: ckb }, function (res) {
                $("#retVal").text(res);
            });
        });
        //*************
        $(".semnpn").click(function () {
            var name = this.name;
            var ran = Math.random();
            var ckb = $("input[name=" + name + "]").is(':Checked') ? "Checked_Action " +
ran : "UnChecked_Action " + ran;
            $("#log").text(name + ckb);

            //alternatively ->  ckb = $(".btnin::checked").length
            $.get("", { [name]: ckb }, function (res) {
                $("#retVal").text(res);
            });
        });

        $(".sempnp").click(function () {
            var name = this.name;
            var ran = Math.random();
            var ckb = $("input[name=" + name + "]").is(':Checked') ? "Checked_Action " +
ran : "UnChecked_Action " + ran;
            $("#log").text(name + ckb);

            //alternatively ->  ckb = $(".btnin::checked").length
            $.get("", { [name]: ckb }, function (res) {
                $("#retVal").text(res);
            });
        });

        $(".semn-mosfet").click(function () {
            var name = this.name;
            var ran = Math.random();
            var ckb = $("input[name=" + name + "]").is(':Checked') ? "Checked Action " +
ran : "UnChecked Action " + ran;
            $("#log").text(name + ckb);

            //alternatively ->  ckb = $(".btnin::checked").length
            $.get("", { [name]: ckb }, function (res) {
                $("#retVal").text(res);
            });
```

```
        });
            $(".semp-mosfet").click(function () {
            var name = this.name;
            var ran = Math.random();
            var ckb = $("input[name=" + name + "]").is(':Checked') ? "Checked_Action " +
ran : "UnChecked_Action " + ran;
            $("#log").text(name + ckb);

            //alternatively ->  ckb = $(".btnin::checked").length
            $.get("", { [name]: ckb }, function (res) {
                $("#retVal").text(res);
            });
        });
            $(".semn-jfet").click(function () {
            var name = this.name;
            var ran = Math.random();
            var ckb = $("input[name=" + name + "]").is(':Checked') ? "Checked_Action " +
ran : "UnChecked_Action " + ran;
            $("#log").text(name + ckb);

            //alternatively ->  ckb = $(".btnin::checked").length
            $.get("", { [name]: ckb }, function (res) {
                $("#retVal").text(res);
            });
        });
        $(".semp-jfet").click(function () {
            var name = this.name;
            var ran = Math.random();
            var ckb = $("input[name=" + name + "]").is(':Checked') ? "Checked_Action " +
ran : "UnChecked_Action " + ran;
            $("#log").text(name + ckb);

            //alternatively ->  ckb = $(".btnin::checked").length
            $.get("", { [name]: ckb }, function (res) {
                $("#retVal").text(res);
            });
        });
        $(".semdiodo").click(function () {
            var name = this.name;
            var ran = Math.random();
            var ckb = $("input[name=" + name + "]").is(':Checked') ? "Checked Action " +
ran : "UnChecked_Action " + ran;
            $("#log").text(name + ckb);

            //alternatively ->  ckb = $(".btnin::checked").length
            $.get("", { [name]: ckb }, function (res) {
                $("#retVal").text(res);
            });
        });
        $(".semamp-op").click(function () {
            var name = this.name;
            var ran = Math.random();
            var ckb = $("input[name=" + name + "]").is(':Checked') ? "Checked_Action " +
ran : "UnChecked_Action " + ran;
            $("#log").text(name + ckb);

            //alternatively ->  ckb = $(".btnin::checked").length
            $.get("", { [name]: ckb }, function (res) {
                $("#retVal").text(res);
            });
        });
            //*************
        $(".sldrclass").click(function () {
            var sl = this.id;
            var val = $("#" + sl).val();
            $("#" + sl).next().text(val);

            $.get("", { [sl]: val }, function () {
                $("#log").text(sl + " Value: " + val);
            });
        })
    });
</script>
```

Figura 50: Creación de cabecera de página Web

La siguiente imagen corresponde a una captura del software VS 2015, en esta se observan todos los archivos necesarios para que la página, con todos sus elementos, sea publicada por el servidor:

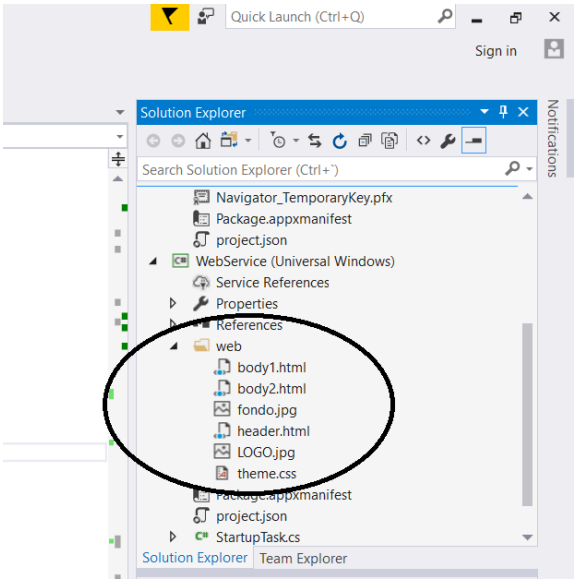

Figura 51: Componentes de página Web

3.4.4.1. Acceso y manejo de Página Web

De forma similar al panel de operación local, la página web exhibe las opciones configurables y representa los resultado gráficos y analíticos.

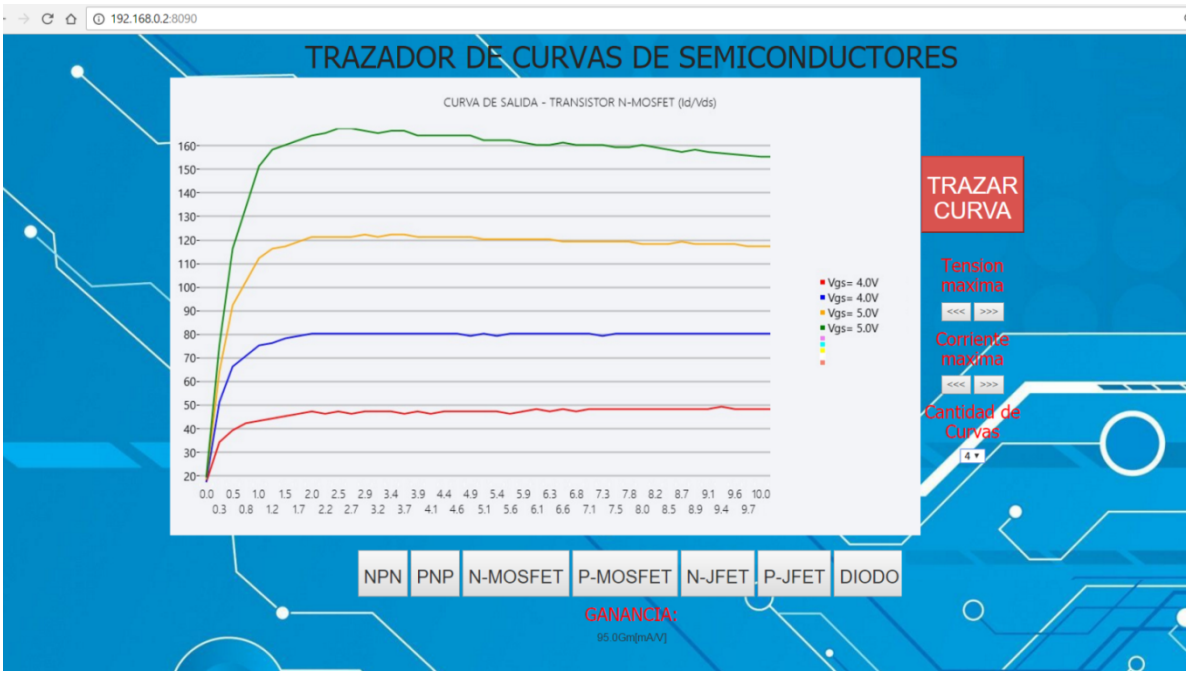

Figura 52: Acceso a página Web

3.4.5. Comunicación entre página web y aplicación

3.4.5.1. Socket

Básicamente, un socket permite la comunicación entre un proceso cliente y un proceso servidor y puede ser orientado a conexión o no. Un socket cliente en una computadora utiliza una dirección para llamar a un socket servidor en otro dispositivo. Una vez enlazados los sockets apropiados, los dispositivos involucrados en la comunicación pueden intercambiar datos. Normalmente las computadoras con socket servidores mantienen un puerto TCP o UDP abierto, preparado para llamadas de entrada eventuales. El cliente generalmente determina la identificación del socket del servidor deseado hallándolo en la base de datos.
Los sockets pueden crearse desde un programa (en un lenguaje como C o Java), permitiendo al programador proporcionar fácilmente funciones y aplicaciones de red. El mecanismo de programación de socket incluye una semántica suficiente como para permitir que se comuniquen dos procesos independientes en máquinas diferentes.

3.4.5.2. WebSocket

Como se mencionó anteriormente el envío de mensajes entre la página web y la aplicación se da por a través de un socket. El protocolo WebSockets proporciona una comunicación bidireccional rápida y segura entre un cliente y un servidor a través de la web. En la plataforma Windows universal (UWP), los programadores pueden utilizar las clases MessageWebSocket y StreamWebSocket para conectar con los servidores que soportan el protocolo WebSocket.

Las características más importantes son:

- En el marco del Protocolo de WebSocket, los datos se transfieren inmediatamente a través de una conexión full-duplex.
- Se permite que los mensajes sean enviados y recibidos desde ambos puntos finales en tiempo real.
- Los WebSockets son ideales para su uso aplicaciones en donde la comunicación se debe dar en tiempo real.

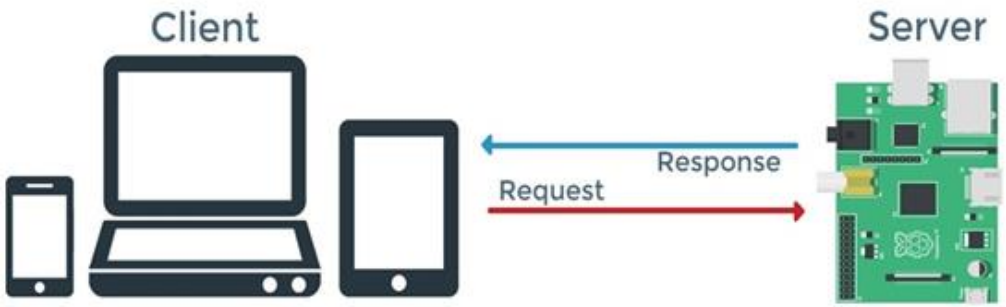

Figura 53: Comunicación por Websocket

4. Ensamble final del equipo

4.1. Diseño y montaje de gabinete

Dadas las dimensiones de las placas de potencia, placa de control, transformador y demás componentes requeridos por el dispositivo, se optó por utilizar un gabinete plástico de dimensiones 250mm de ancho, 2000mm de profundidad y 150mm de altura.
Para obtener un aspecto más agradable se pintaron sus partes al tono negro mate.

Figura 54: Diseño de gabinete de Trazador

4.2. Montaje de placas

Dentro del gabinete se dispusieron las placas electrónicas sobre la base del mismo. Asimismo, hacia un lateral se fijó el transformador.

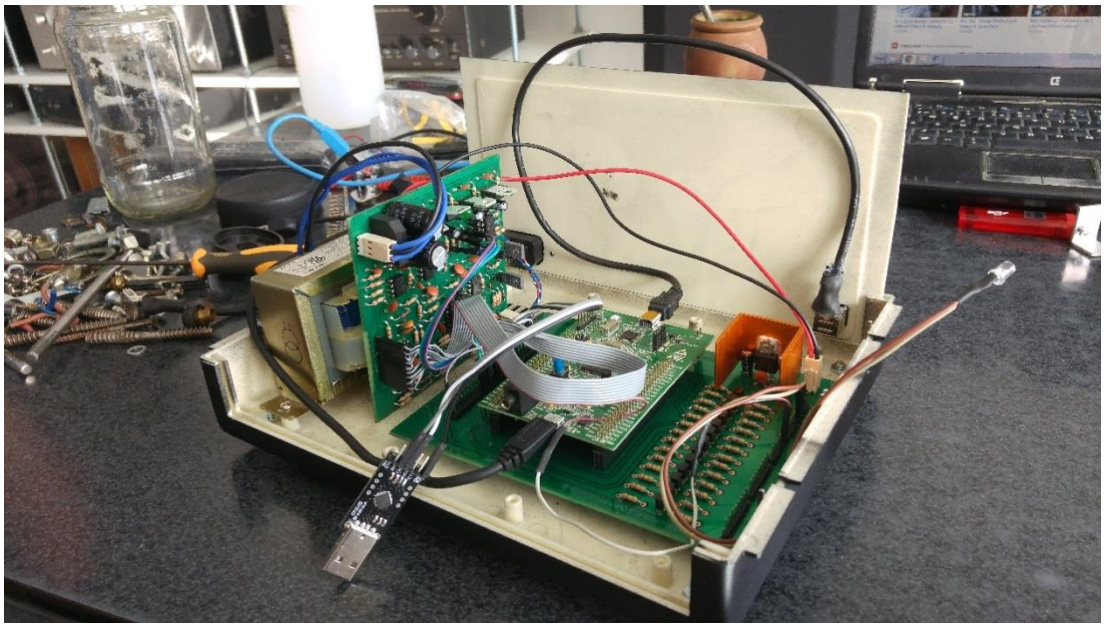

Figura 55: Montaje de placas en gabinete de Trazador

De manera que se pueda aprovechar el espacio disponible en el gabinete se colocó la placa de potencia de forma vertical, con sus partes de mayor disipación de temperatura hacia la zona de ventilación. Se fabricó un soporte de fijación para dicha placa:

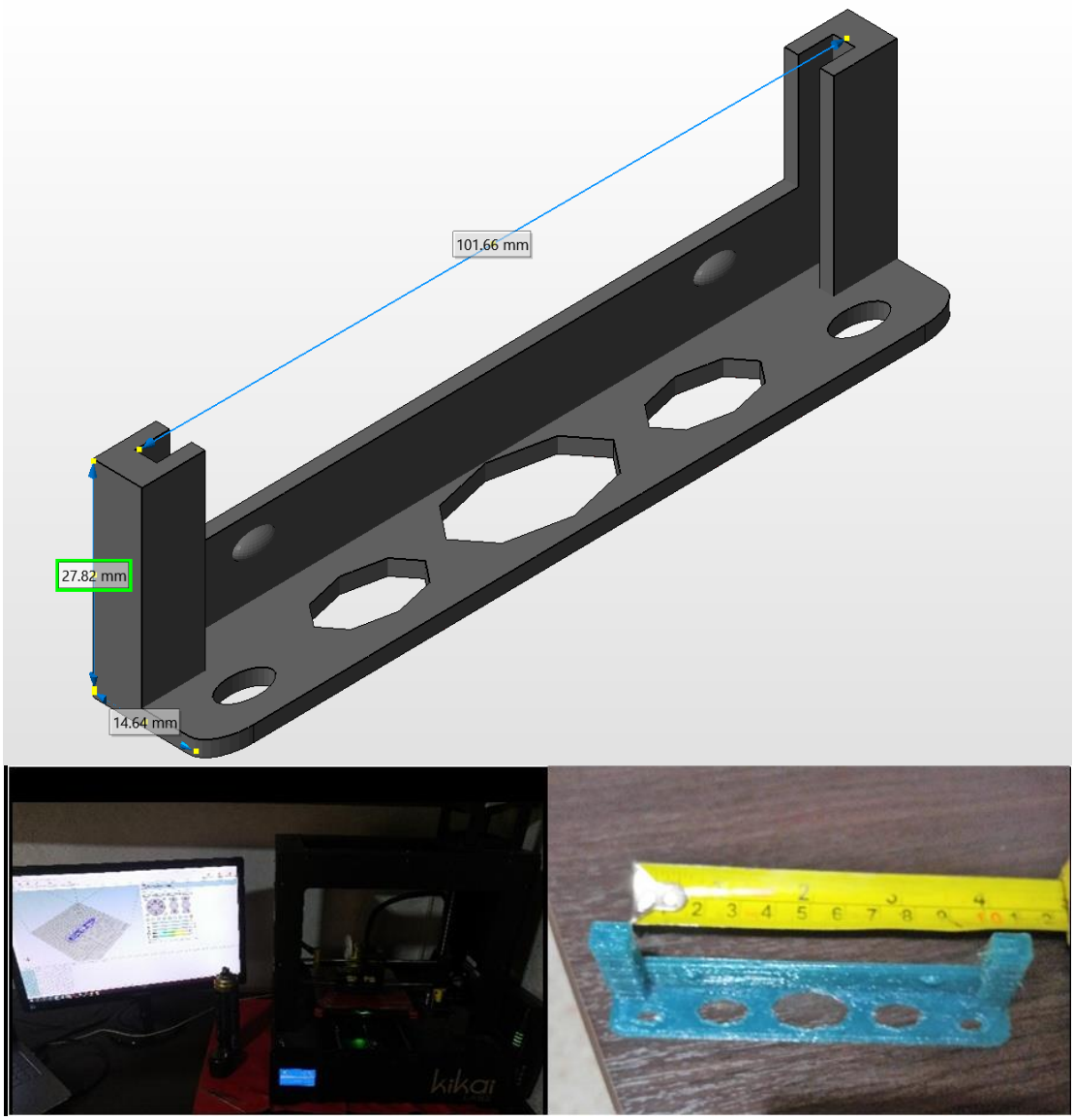

Figura 56: Diseño y fabricación de soporte de placas en gabinete de Trazador

La conexión entre la placa de potencia y la de control se realizó mediante un cable plano:

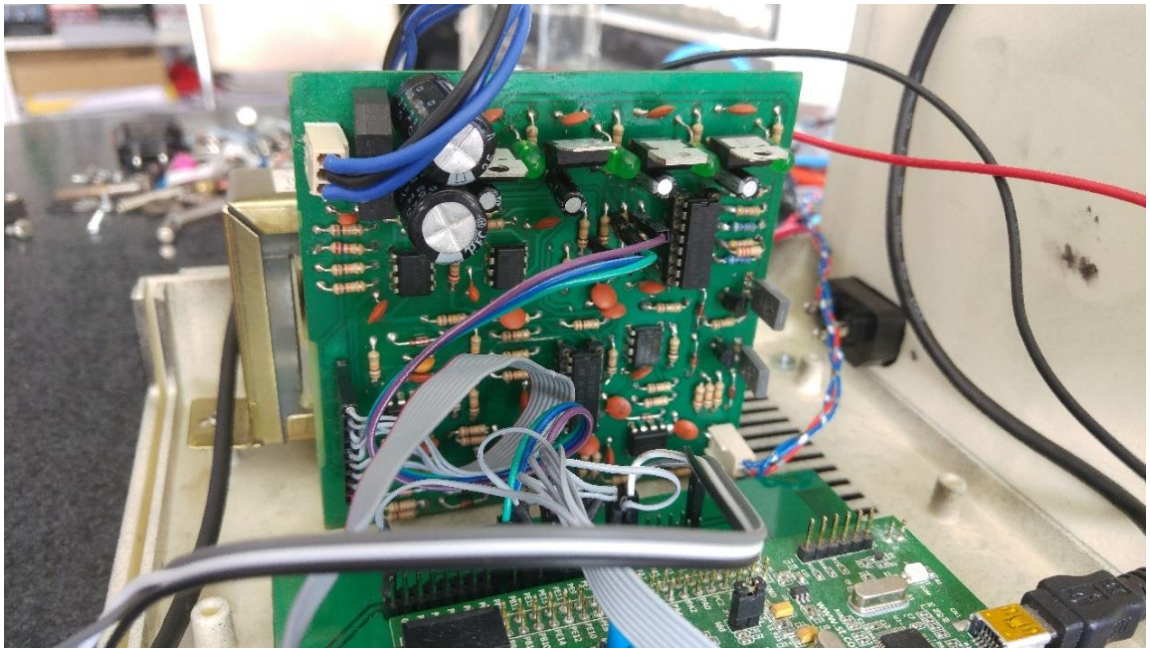

Figura 57: Instalación de placas en gabinete de Trazador

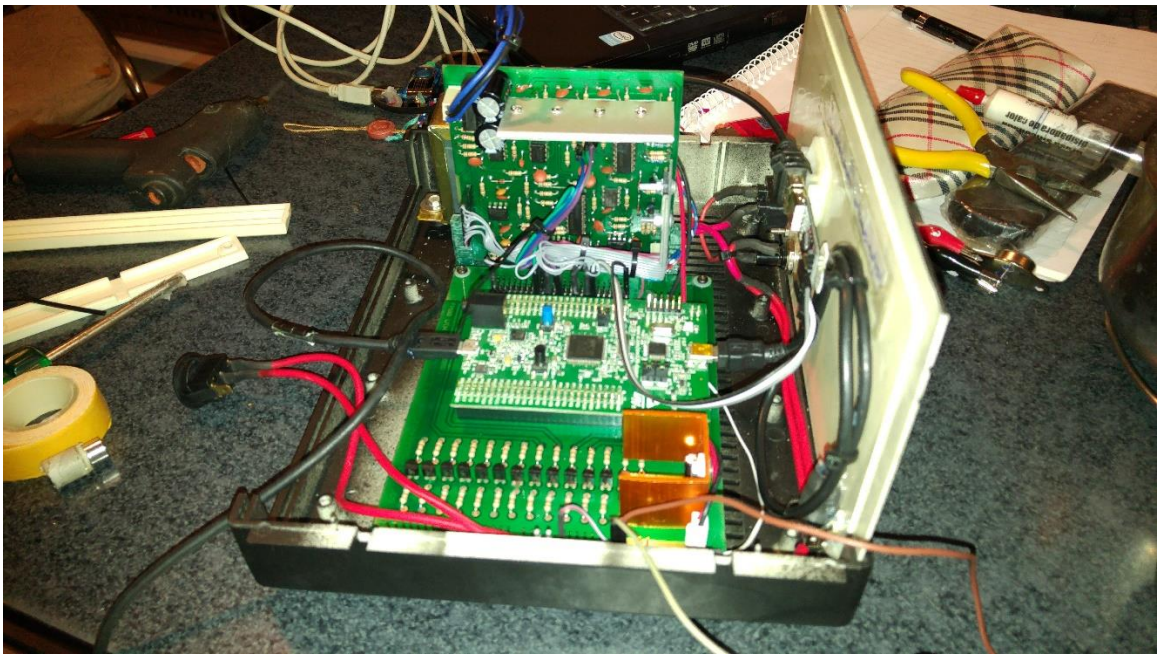

Figura 58: Conexionado de placas en gabinete de Trazador

4.3. Montaje de Kit Raspberry PI

Se diseñó un soporte de fijación para montar la placa sobre un lateral del gabinete:

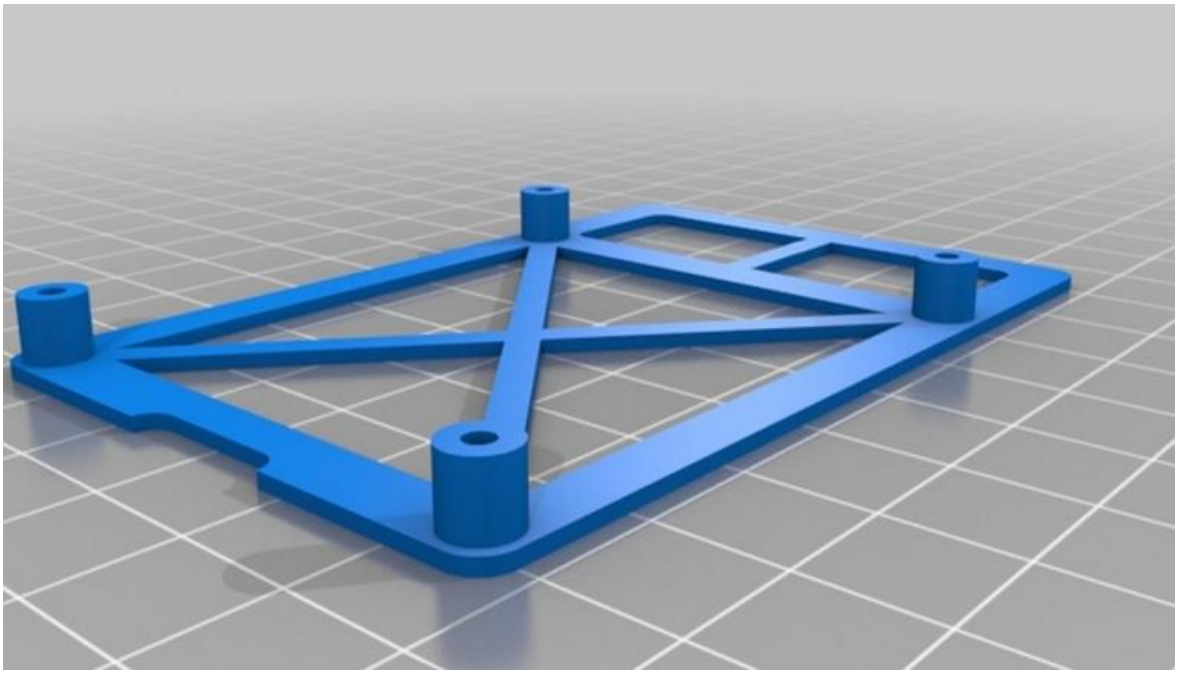

Figura 59: Diseño de soporte para RASPBERRY PI 3

Figura 60: Montaje de placa RASPBERRY PI 3 en gabinete de Trazador

4.4. Calado para conexiones accesibles

Figura 61: Calado para periféricos de Trazador

Además de los periféricos de la RASPBERRY PI se agregó una salida HDMI para conexión de pantalla de visualización externa.

4.5. Diseño y montaje de pies de apoyo

Para mejorar el aspecto y la visualización de la pantalla del dispositivo se incorporaron dos bases ajustables en altura, las cuales permiten obtener una inclinación máxima aproximada de 45º.

Figura 62: Soporte de apoyo para gabinete de Trazador

5. Ensayos y Mediciones

Luego de numerosos ensayos del dispositivo Trazador con diversos componentes y realizados todos los ajustes de ganancias de las adaptaciones de potencia, chequeados los niveles de temperaturas internas del gabinete y efectuada la verificación de las comunicaciones, se dio por finalizada la etapa de puesta en marcha del dispositivo desarrollado.
Para contrastar los resultados se coordinó junto con el docente e Ingeniero de la facultad FRBB, Leandro Ortiz un visita a la base naval Puerto Belgrano ubicada en la ciudad de Punta Alta, allí accedimos a un trazador comercial modelo Tektronix 577 disponible en el taller de electrónica. Dicho dispositivo fue utilizado para calibrar los valores de las curvas obtenidas con el Trazador de Curvas de Semiconductores desarrollado.
Se llegó a la conclusión de que los resultados obtenidos se aproximaban correctamente a los valores arrojados por el trazador Tektronix.
Cabe destacar que para este momento el equipo no contaba con la pantalla de aplicación propia por lo que se utilizó el software de PC desarrollado previamente en Visual Basic.
El equipo Tektronix posee rangos de trabajo más extensos que nuestro Trazador, por lo que necesariamente se definió una franja común para efectuar los ensayos comparativos.
Se obtuvieron gráficas representativas de funcionamiento de diodos rectificadores, diodos zener y algunos transistores:

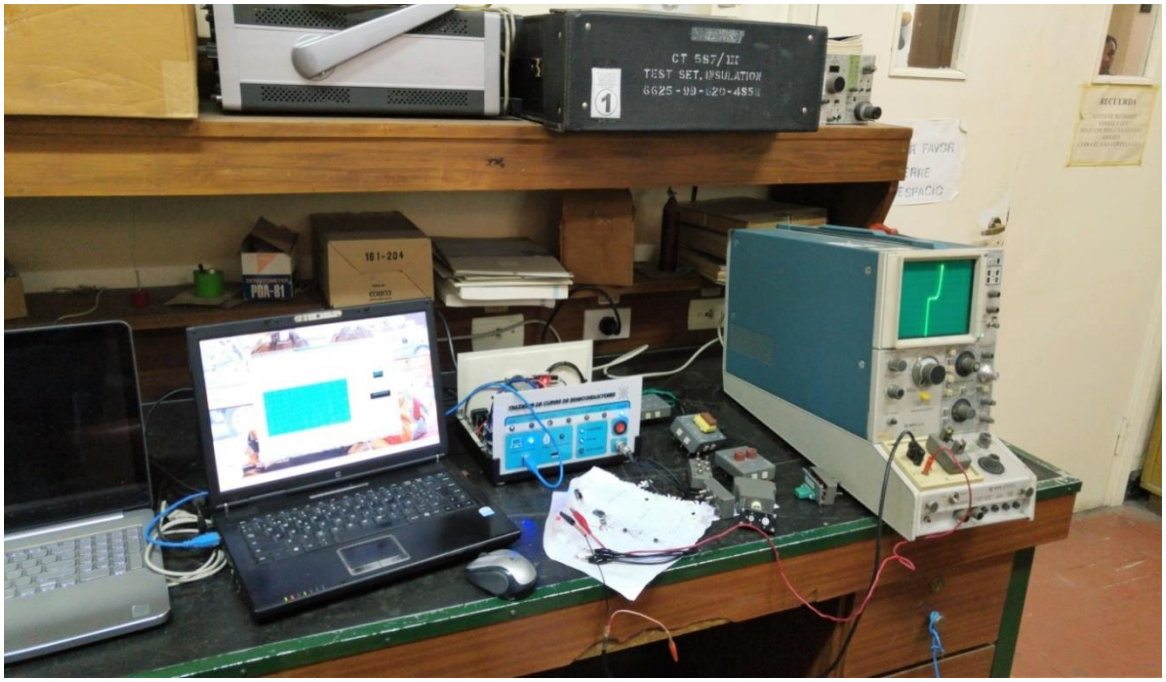

Figura 63: Ensayo de diodo con equipo Trazador

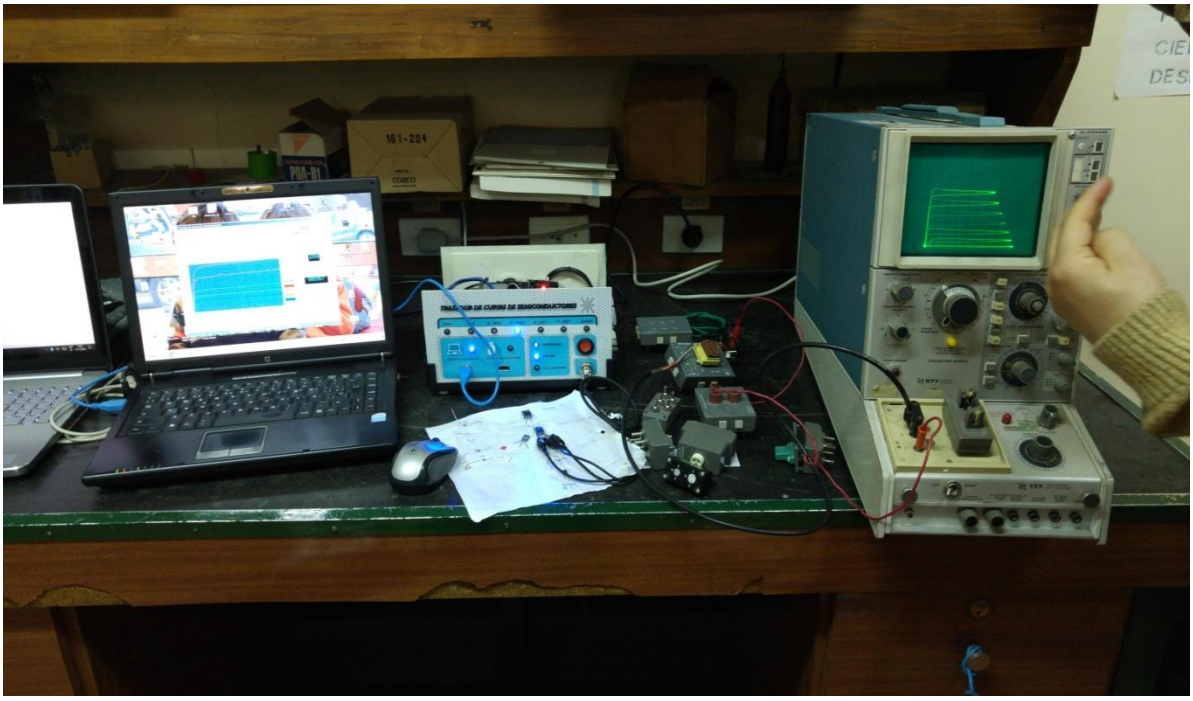

Figura 64: Ensayo de transistor con equipo Trazador

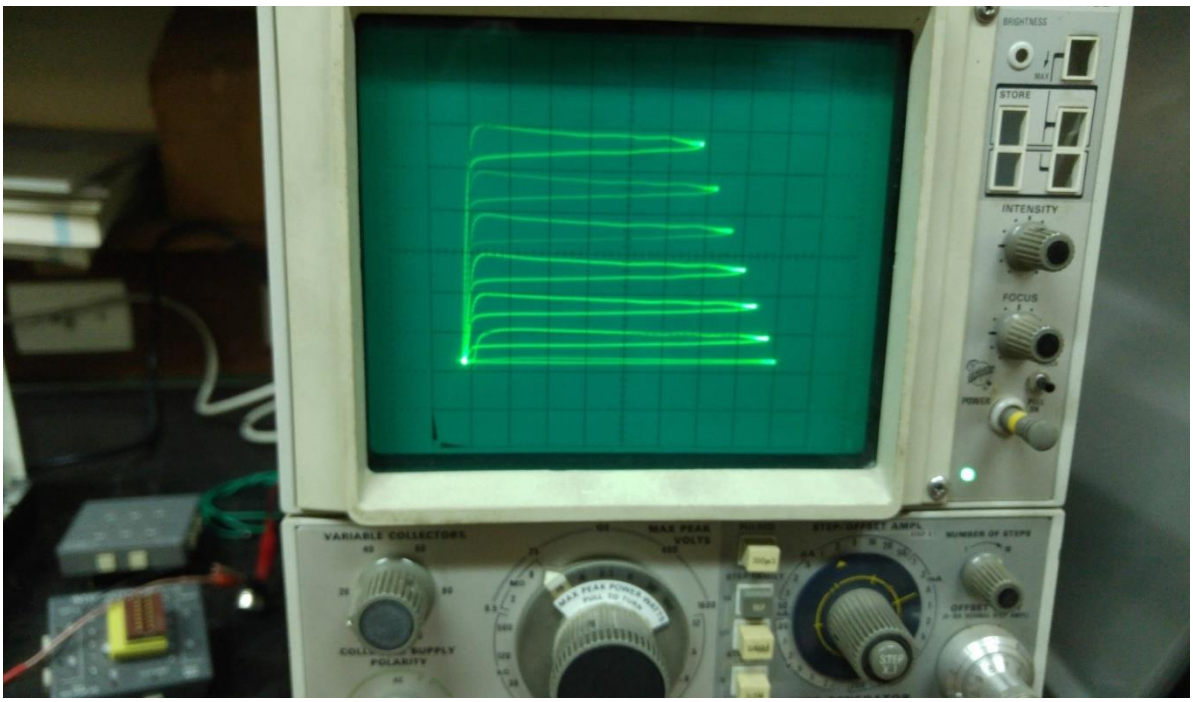

Figura 65: Curvas de transistor con equipo trazador Tektronix 577

6. Funcionamiento del equipo

El equipo debe ser conectado a la red domiciliaria 220Vac. Desde el botón frontal se enciende y muestra una pantalla de presentación:

Figura 66: Inicialización de aplicación local de Trazador

Una vez encendido el dispositivo y cargada la interfaz gráfica de la aplicación, se debe seleccionar el tipo de semiconductor a ensayar, la corriente y tensión máxima a la cual se someterá el mismo y, luego de verificar que el componente soporta dicho ensayo, se procede a elegir la cantidad de trazos, en caso de ser un diodo el trazo será único. Seleccionados estos parámetros inicia el proceso de trazado de la gráfica.

Figura 67: Funcionamiento de Trazador

6.1. Acceso desde página Web

El kit RASPBERRRY PI 3 posee una placa WIFI integrada que le permite conectarse a una red inalámbricamente o bien mediante un cable de red convencional, a través de su puerto Ethernet.
Para logar que los usuarios puedan acceder al equipo remotamente, ambos deben conectarse a una misma red. Es decir, se debe contar con un router que genere una red que permita el flujo de la información entre ambos extremos.
Una vez creada la red, los equipos y dispositivos necesarios deben conectarse a ella. Como se mencionó previamente la página web es accesible mediante un puerto particular: 8090.
Ahora bien, solo es necesario que desde cualquier buscador se coloque la dirección de IP del router seguido de ":" y luego el numero de puerto para que se haga visible la página Web del Trazador.

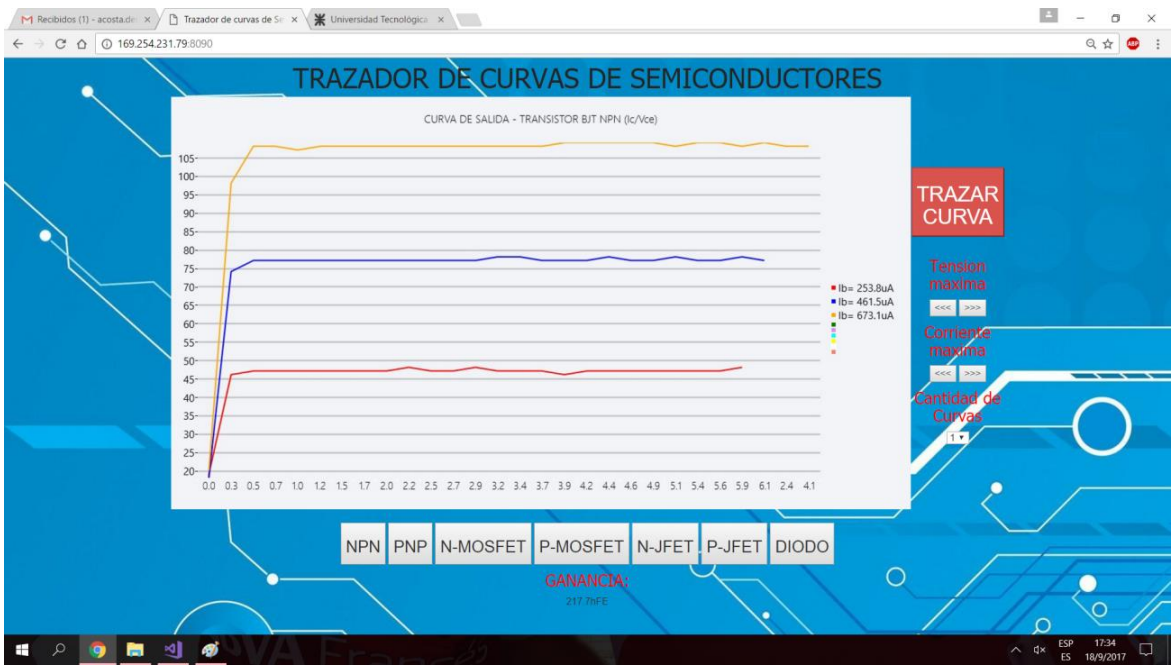

Figura 68: Acceso a Trazador desde página Web

Vale aclarar, la carga inicial de la página mostrará el último resultado obtenido de los ensayos.
Por otro lado, la modificación de parámetros desde la página también será visible por la aplicación local, es decir, un cambio de tipo de semiconductor en la web por ejemplo, modificará el parámetro correspondiente en la pantalla propia del equipo. Por tal motivo es recomendable que el Trazador sea comandado necesariamente por un usuario a la vez, de este modo los cambios realizados sobre cualquier plataforma repercuten directamente sobre la aplicación principal.

7. Medición de los dispositivos semiconductores usando el equipo

En esta sección se explica cómo realizar medidas básicas de los dispositivos más comunes.

Cuando se realice un ensayo, los parámetros elegidos deben ser acordes con el dispositivo a medir, de otro modo, el equipo presentará un mensaje indicando el error de conexión o dispositivo.

7.1. Transistores bipolares (NPN y PNP)

Conectamos el transistor en las tres puntas de prueba (E-S, B-G, C-D) con el orden correcto de acuerdo a la hoja de datos del dispositivo, en el menú seleccionamos NPN/PNP, indicamos la cantidad de curvas y valores máximos de ensayo y presionamos Trazar, en la pantalla se verá la gráfica correspondiente.

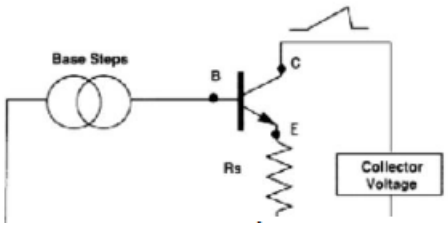

Figura 69: Conexión y ensayo de transistor NPN

Obtenemos la curva característica I-V de la corriente Ic en función de la tensión entre colector y emisor (Vce), para distintos valores de la tensión entre base y emisor (Vbe).

7.1.1. Medida de la tensión de saturación (Vce sat)

Ejemplo con transistor PNP MJ2955

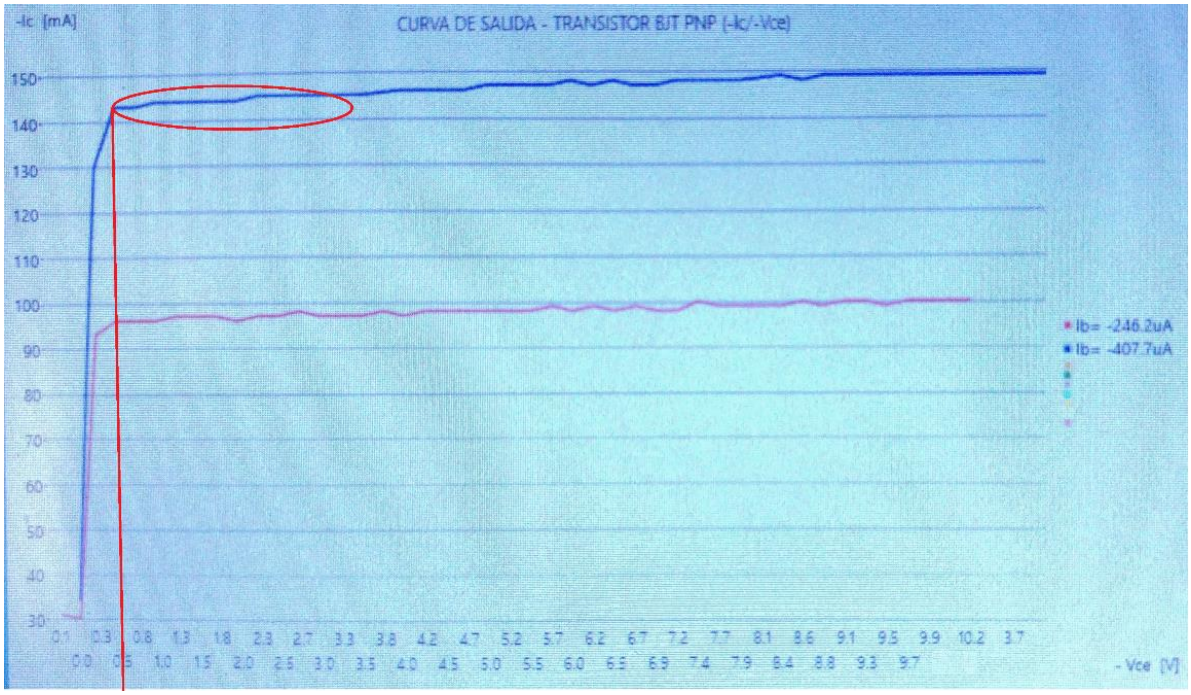

Figura 70: Curva de ensayo de PNP – Región de saturación

La línea vertical roja muestra de manera aproximada el valor del voltaje Vce a partir del cual el transistor empieza a operar en saturación. La zona rodeada muestra dicho efecto de saturación en la corriente que circula por el transistor.

7.1.2. Identificación de regiones de trabajo

Ejemplo con transistor NPN BD139

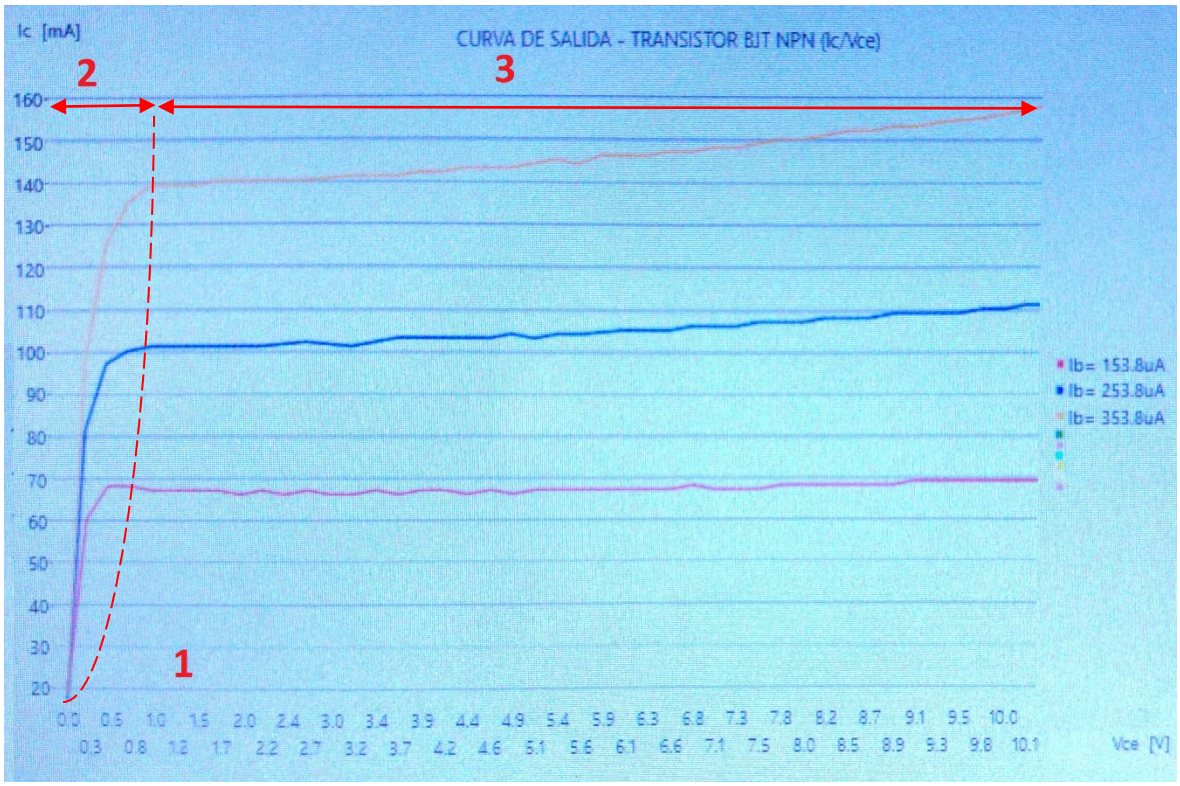

Figura 71: Curva de ensayo de NPN – Regiones de trabajo

Conocido el punto de operación de nuestro circuito de polarización, podemos determinar mediante la recta de carga la región de operación del transistor: corte (1), lineal (2) y saturación (3).

7.2. Diodos

En este caso, solo se utilizan dos puntas de prueba, C-D y E-S. Conectamos el Diodo en las puntas de prueba con el orden correcto, en el menú seleccionamos DIODO y presionamos Trazar, en la pantalla se verá el resultado.

7.2.1. Medida de la tensión de ruptura de un diodo

La tensión de ruptura de un diodo es un parámetro que define el máximo voltaje inverso que se puede aplicar sin que se cause un incremento exponencial de la corriente en el diodo. Esta característica se puede utilizar para hacer reguladores de tensión, como es el caso del Diodo Zener, un diodo muy dopado que explota la característica de la tensión de ruptura: cuando la tensión en polarización inversa alcanza el valor de la tensión de ruptura, el mismo campo eléctrico de la unión P-N es capaz de arrancar electrones de la banda de valencia permitiendo la conducción casi sin variación de la tensión.

El Trazador de Curvas de Semiconductores posee la capacidad de realizar ensayos a un amplio rango de diodos del tipo Zener. Cabe destacar que si a estos dispositivos los polarizamos en directa se apreciara una curva similar a la de un diodo rectificador como por ejemplo 1N4148.

7.2.2. Tensión en directa

Ejemplo con diodo 1N4148.

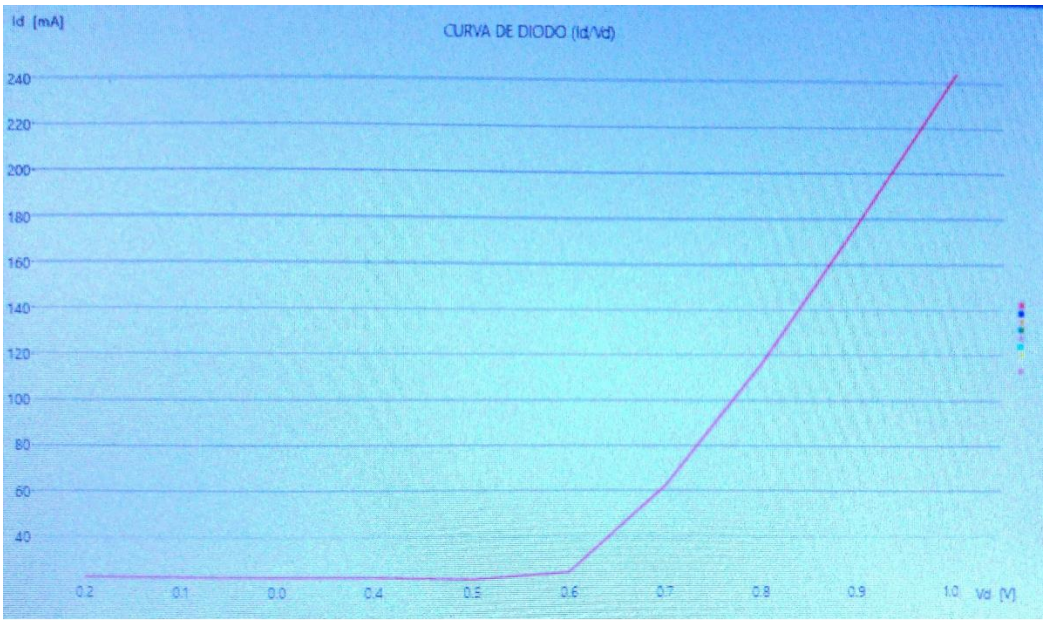

Figura 72: Curva de ensayo de DIODO

Se muestra en la imagen la clásica curva I/V de un diodo.

7.2.3. Tensión Inversa

Ejemplo con diodo zener 5,3V.

Figura 73: Curva de ensayo de Zener

7.3. Transistores FET (JFET, NMOS y PMOS)

Del mismo modo que en los BJT utilizamos las tres puntas de prueba (E-S, B-G, C-D) con el orden correcto de acuerdo a la hoja de datos del dispositivo. En el menú seleccionamos el tipo de FET elegido, indicamos la cantidad de curvas y valores máximos de ensayo y presionamos Trazar, en la pantalla se verá la gráfica correspondiente.

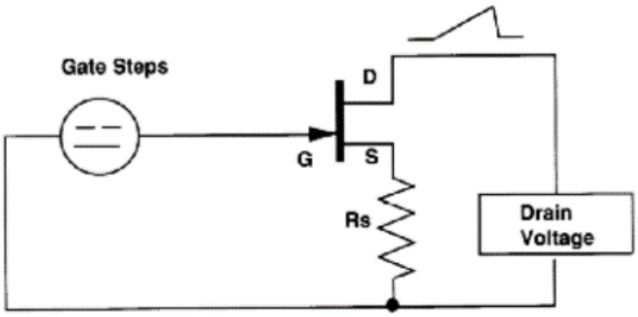

Figura 74: Conexión y ensayo de transistor FET

Ejemplo con transistor NMOSFET 2N7000

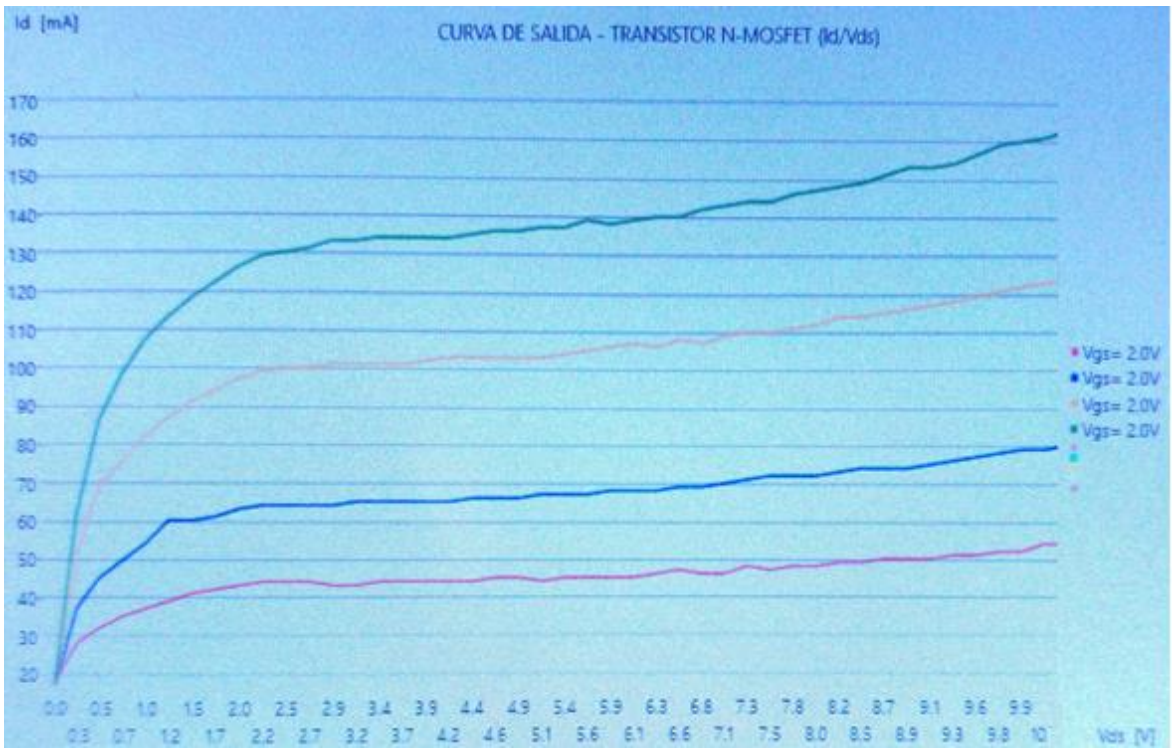

Figura 75: Curva de ensayo de NMOSFET

Podemos ver en la imagen la clásica curva IDS frente a VDS característica de un transistor MOSFET.
El procedimiento es análogo al caso de un transistor bipolar.

7.3.1. Determinación de región de trabajo

Ejemplo con transistor NMOSFET 2N7000

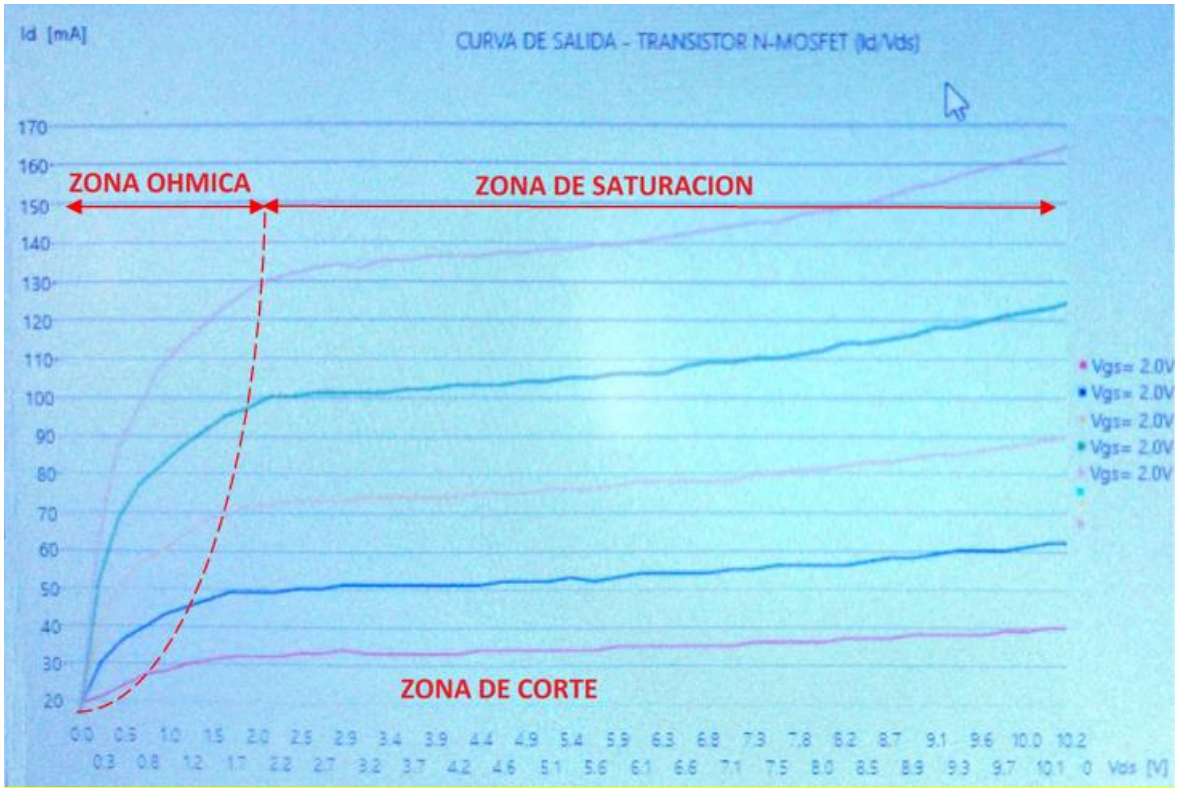

Figura 76: Curva de ensayo de NMOSFET – Regiones de trabajo

De forma similar a los BJT, conocido el punto de operación de nuestro circuito de polarización, podemos determinar mediante la recta de carga la región de operación del MOSFET.

8. Inversión realizada

Durante el desarrollo del dispositivo se efectuaron varios cambios en cuanto al diseño circuital y funcional del mismo, cabe destacar que aquí solo se hace mención a los materiales empleados para el prototipo final.

Ítem	Descripción	Cantidad	Precio unitario	Precio total
1	Kit de programación STM32F4	1	$650,00	$650,00
2	Placa PCB en Epoxi Doble + SolderMask 120x140mm	1	$440,00	$440,00
3	Placa PCB en Epoxi Doble + SolderMask 100x100mm	1	$195,00	$195,00
4	Envío de componentes desde GM electrónica (CABA)	1	$257,00	$257,00
5	Conversor USB-UART TTL CP2102 ARDUINO	1	$159,12	$159,12
6	Transformador 220VAC/15+15V 1A	1	$249,00	$249,00
7	Circuito integrado DG221CJ switch analógico	3	$135,44	$406,32
8	Resistencias varias 1/4w	1	$118,00	$118,00
9	Capacitores cerámicos y electrolíticos varios	1	$43,00	$43,00
10	Transistor BC548	15	$1,66	$24,90
11	Led 5mm alto brillo	15	$4,20	$63,00
12	Regulador de voltaje 5V 7805	4	$11,00	$44,00
13	Tira de pines hembra para poste P/CI (2x40)	3	$35,33	$105,99
14	Terminal de pines Macho P/CI 2 vías recto	1	$2,35	$2,35
15	Terminal de pines Hembra 2 vías	1	$3,60	$3,60
16	Tira de pines simple 1x40	1	$9,61	$9,61
17	Terminal 3A 250V contacto en 1 cara p/serie HU	40	$0,83	$33,30
18	Tira hembra para pines (1x40) sin terminales	1	$13,43	$13,43
19	Cable téster	1	$5,50	$5,50
20	Termo-contraíble 64mm	2	$3,96	$7,92
21	Rollo 5mts 1x0,25	1	$12,00	$12,00
22	Cocodrilo chico negro	3	$9,00	$27,00
23	Cable micrófono 2x0	1	$3,50	$3,50

24	Portaled 5mm	12	$3,20	$38,40
25	Resistencia 3w 1 ohm	1	$2,41	$2,41
26	Regulador de voltaje 12V 7812	1	$8,00	$8,00
27	Regulador de voltaje 15V 7815	1	$10,30	$10,30
28	Regulador de voltaje -12V 7912	1	$10,10	$10,10
29	Regulador de voltaje -15V 7915	1	$9,59	$9,59
30	Circuito integrado amplificador operacional TL082	4	$11,30	$45,20
31	Circuito integrado amplificador operacional TL081	2	$13,04	$26,08
32	Zócalo 2x4	6	$1,95	$11,70
33	Zócalo 2x8	2	$2,68	$5,36
34	Diodo 1N4148	8	$4,40	$35,20
35	Capacitor electrolíticos 2000x35V	2	$21,50	$43,00
36	Diodo Zener 3,3V	3	$5,00	$15,00
37	Transistor BD139	2	$13,00	$26,00
38	Transistor BD140	2	$13,00	$26,00
39	Led 5mm Difuso verde	4	$4,84	$19,36
40	Conector Hembra 3 vías	2	$3,15	$6,30
41	Conector Macho 3 vías recto	2	$3,87	$7,74
42	Porta-fusible panel 20x5mm con muesca	1	$11,28	$11,28
43	Conector tipo USB tipo B impresora p/CI	2	$25,06	$50,12
44	USB04H conector USB tipo A Hembra	1	$18,45	$18,45
45	PL-PC/CH macho tipo PC para chasis	1	$22,20	$22,20
46	Llave de encendido 3A 205V ON-ON redonda con luz	1	$25,29	$25,29
47	Aislante para disipadores de transistores	6	$2,60	$15,60
48	Aislante de mica para TO220	6	$3,88	$23,28
49	SVP macho para chasis 3 patas	1	$42,59	$42,59
50	SVP hembra para cable 3 patas	1	$42,59	$42,59
51	Bastidor plástico 200x240mmx160mm	1	$350,00	$350,00
52	Bulonería para fijación y patas	1	$63,00	$63,00
53	Pintura en aerosol negro mate	2	$164,00	$328,00

54	Ploteo de panel de control	1	$160,00	$160,00
55	Materiales consumibles (estaño. Cable, pines, precintos, etc.)	1	$300,00	$300,00
56	Raspberry Pi 3	1	$1600	$1600
57	Pantalla 5´´ hdmi Lcd	1	$1200	$1200
58	Fuente 5v para pantalla hdmi	1	$230	$230
59	Adaptador hdmi macho- hembra	2	$120	$240
60	Adatador tipo T hdmi	1	$45	$45
61	Extensión hdmi	2	$60	$120
62	Tecla dos puntos para bastidor	1	$33	$33
63	Ventilador cooler 12vdc	1	$52	$52
			TOTAL:	**$8191,98**

Figura 77: Tabla de inversión realizada

9. Conclusiones

Luego de muchas horas de trabajo en equipo, numerosos ensayos, calibraciones y modificaciones, a comienzos del mes de Julio del año 2017 el Trazador de Curvas de Semiconductores estaba abordando la etapa de finalización. Tras varias reuniones con los profesores de la cátedra de Proyecto Final, recibimos la propuesta de competir en el concurso SASE 2017. En primera instancia considerábamos improbable nuestra participación debido a que el equipo todavía no estaba culminado para ser presentado en dicho simposio. Pero como no podía ser de otra manera, nos llenaba de orgullo la idea de formar parte de la representación de la facultad en nuestra disciplina, motivo por el cual los esfuerzos debieron ser multiplicados aún más y para fines del mes de Julio se pudo concluir el proyecto casi en su totalidad, dejando pendientes algunos detalles menores. El SASE 2017 se realizó del 9 al 11 de Agosto, el Trazador fue presentado y obtuvo el segundo puesto del concurso.

Durante todo el camino recorrido, desde que nos propusimos diseñar un Trazador de Curvas hasta la finalización del proyecto en su totalidad, se transitó por muchas etapas, todas poseían su grado de dificultad, algunas de mayor complejidad, otras de menor, pero había algo que teníamos bien claro, era que el Trazador debía ser terminado en tiempo y forma.

Se presentaron diversos problemas durante el proceso, pero todos estos fueron superados satisfactoriamente, a esta altura de la carrera el conocimiento para poder resolver problemas y la constancia del grupo de trabajo fueron los principales motores que impulsaron la finalización de este proyecto.

La Electrónica pone a disposición de los que saben de ella muchísimas herramientas, y practicante todo lo que uno se propone se puede llevar a cabo, siempre y cuando la constancia y las ganas de superar el objetivo sean firmes y estén bien establecidas las metas. La metería Proyecto Final brinda apoyo constante a los alumnos que la cursan, pero solo en ellos está el poder de decidir si quieren culminar sus proyectos. Muchas veces se presentan piedras en el camino, como todos sabemos en Argentina a veces se hace muy difícil conseguir componentes electrónicos, es por eso que, como se mencionó anteriormente, fueron necesarias muchas modificaciones en el transcurso de todo el proyecto. Particularmente estas situaciones nos permitieron darnos cuenta de que realmente estábamos capacitados para esquivar obstáculos y seguir adelante. El trabajo en equipo es fundamental para lograr el objetivo, si el grupo funciona como tal y se poseen objetivos claros, a medida que los días pasan las metas se aproximan y se superan.

10. Referencias Bibliográficas

Los materiales bibliográficos consultados para realizar este proyecto son:

[1] Teoría de Circuitos y Dispositivos Electrónicos, Décima edición. Robert Boylestad, Louis Nashelsky.

[2] H. K. Gummel, H. C. Poon, "A Compact Bipolar Transistor Model", IEEE International Solid-State Circuits Conference, pp. 78-79, 1970.

[3] [1] Neamen D. A. Microelectronics, Circuit Analysis and Design, 3rd ed. New York: Mc Graw Hill International Edition, (2007).

[4] Electrónica de semiconductores, Volumen 12, Alan M. Portis,Hugh D. Young

[5] Hoja de datos: DG441, DG442, Switch analógico cuádruple.

[6] Hoja de datos: STM32F4DISCOVERY.

[7] https://sandervandevelde.wordpress.com/2016/04/08/building-a-windows-10-iot-core-background-webserver/amp/ /

[8] https://developer.microsoft.com/en-us/windows/iot/samples/serialuart

[9] https://www.hackster.io/fvdbosch/uart-for-serial-console-or-hat-on-raspberry-pi-3-5be0c2

[10] https://desarrolloweb.com/articulos/generar-graficos-dinamicamente-asp-net.html

[11] http://stackoverflow.com/questions/40773836/calling-external-chartjs-chart-from-a-javascript-file-into-html-in-flask

[12] https://klingdigital.wordpress.com/2015/03/10/download-image-bytes-to-file-display-in-xaml-winrt-universalapps/

[13] http://stackoverflow.com/questions/21325661/convert-image-path-to-base64-string

[14] http://stackoverflow.com/questions/31046618/get-base64-string-from-image-src

[15] https://www.google.com.ar/amp/s/devio.wordpress.com/2011/01/13/embedding-images-in-html-using-c/amp/

[16] http://www.w3ii.com/es/windows10_development/windows10_development_quick_guide.html

[17] https://ortizol.blogspot.com.ar/2015/11/cadenas-de-caracteres-y-manipulacion-de-texto-en-csharp-parte-2-5-string.html

[18] https://alakaxaml.wordpress.com/tag/winrt-xaml/

[19] https://www.codeproject.com/Articles/422523/Convert-XAML-Vector-Graphic-to-PNG

[20] http://dotnet-snippets.com/snippet/apps-save-images/3748

[21] https://emacs.stackexchange.com/questions/27060/embed-image-as-base64-on-html-export-from-orgmode

[22] http://stackoverflow.com/questions/41401047/display-local-images-in-uwp-webview-control

[23] https://github.com/Microsoft/Win2D-Samples/blob/master/ExampleGallery/Shared/Infrastructure/MainPage.xaml.cs

[24] http://help.infragistics.com/Help/Doc/WPF/2014.2/CLR4.0/html/Creating_a_Graphical_Image_from_a_Visual_in_WPF.html#_Ref319052987

11. Agradecimientos

Cuando comenzamos la etapa de búsqueda del proyecto integrador de la carrera, sólo teníamos en claro que queríamos desarrollar algo que permita mejorar el equipamiento disponible en la cátedra de Electrónica. Pero esto no debe mal interpretarse como un juicio de este grupo de trabajo hacia las herramientas con que cuenta el estudiante, todo lo contrario, más bien hace referencia a un sentimiento de compromiso que hemos tenido como una consecuencia de los gratos momentos vividos en la Universidad.
Haber llegado a la meta de ésta carrera nos trae consigo un inmenso orgullo personal que solo se puede comparar con el que nos provoca poder desarrollar algo para "nuestra" Universidad. Es por esto que no tenemos más que palabras de agradecimiento hacia la institución y sobre todo hacia las personas que integran el departamento de Ingeniería Electrónica. Desde becarios, pasando por alumnos, consejeros, docentes y hasta no docentes que a lo largo de los años nos nutrieron de conocimiento y sobre todo de capacidad resolutiva, preparándonos de la mejor manera para insertarnos al mundo laboral.