



UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA DE INGENIEROS INDUSTRIALES DE ALBACETE

GRADUADO EN INGENIERÍA EN ELECTRÓNICA IND. Y AUTOMÁTICA

TRABAJO FIN DE GRADO

**DISEÑO DE UN DISPOSITIVO DE MONITORIZACIÓN DE
REDES ELÉCTRICAS EN EL ENTORNO DEL INTERNET
INDUSTRIAL DE LAS COSAS (IIOT)**

Nº DE TFG: 01EN1796/17XI

AUTOR: Rubén García Segovia

TUTOR: Miguel Martínez Iniesta

Junio, 2018

En estos agradecimientos y dedicatorias quiero nombrar a varias personas,

Los primeros serán mis compañeros de carrera,

Por todo el tiempo pasado juntos, sin vosotros no habría sido posible.

La segunda dedicatoria va para mi madre,

Por todas aquellas veces que me dijo que estudiase,

y sin embargo, nunca le hice caso,

¡Qué razón llevaba!

*Por último quiero agradecerle a ese chaval de 18 años que escogió formarse
como ingeniero, ¡Menudos años me has hecho pasar!*

ÍNDICE GENERAL

ÍNDICE GENERAL	1
ÍNDICE DE FIGURAS	5
INTRODUCCIÓN	9
1. Antecedentes	9
2. Objeto del trabajo	10
3. Justificación	11
DESCRIPCIÓN	13
Calidad de la energía	13
Diagrama de bloques del dispositivo	18
Medición de parámetros de red	18
Comunicación mediante el protocolo SPI	18
Cálculo de parámetros de red	19
Almacenamiento de la información	20
Servidor Web	20
Aplicación móvil	20
Pantalla táctil	20
FTP	20
Sistema de alimentación ininterrumpida	20
DESARROLLO	21
1. Monitorización red eléctrica trifásica	21
1.1 Voltaje e intensidad	21
1.2 Conversor analógico digital ADE7912	22
1.3 Medición voltaje	26
1.4 Medición intensidad	27
1.5 Protocolo de comunicación SPI (Serial Peripheral Interface)	28
1.6 Protocolo de comunicación SPI en el ADE7912	29
1.7 Registros de interés en el ADE7912	31
1.8 Sincronización de varios ADE7912	32

1.9 Medición de temperatura con el ADE7912	35
1.10 Registros de configuración	36
1.11 Conversor DC-DC	37
1.12 Conversión analógico digital.....	39
1.13 Filtro RC paso bajo anti aliasing.....	41
1.14 Verificación de redundancia cíclica CRC.....	43
1.15 Puesta en marcha del ADE7912	45
2. Actuadores.....	47
2.1 Relé de estado sólido.....	47
3. Sistema de adquisición, trata y representación de datos	48
3.1 Sistema Empotrado	48
3.2 Dualidad Esclavo-Maestro	49
3.3 TC4066BP.....	49
3.4 DM7404	50
3.5 Compatibilidad entre protocolos SPI.....	50
3.6 Servidor	52
3.7 Computador de placa reducida	52
3.8 Sistema Operativo	54
3.9 PHP7.0 + Apache2.0.....	54
3.10 Biblioteca WiringPi	55
3.11 Pantalla LCD de 7 pulgadas táctil.....	56
3.12 Python + Tkinter	57
3.13 Aplicación para móviles Android	58
4. Diseño Hardware	59
4.1 Sistema de alimentación ininterrumpida	59
4.2 Placa principal.....	69
4.3 Placa de adquisición de datos	80
5. Desarrollo software	85
6. Diagramas de flujo.....	88
6.1 Sistema	88
6.2 Servidor web.....	94

6.3 GUI táctil.....	97
6.4 Aplicación móvil para Android	98
7. Montaje	107
7.1 Monitorización de señales	107
7.2 Placa UNI-DS3.....	109
7.3 MCP3204 – ADC 12 bits.....	110
7.4 Pantalla táctil.....	111
7.5 Fotos del montaje final	111
CONCLUSIONES Y TRABAJOS FUTUROS	113
BIBLIOGRAFÍA	115
ANEXOS	117
1. Hojas de características.....	117
1.1 Resistencia Shunt Vishay.....	117
1.2 Relé de estado sólido G3PH-5150B DC5-24.....	118
1.3 TC4066BP	120
1.4 UNI-DS3	125
1.5 MCP3204	129
1.6 BSS138.....	135
1.7 DM7404.....	138
1.8 LM317.....	142
1.9 7805.....	145
1.10 7833.....	149
1.11 PIC 18F8520.....	152
2. Código utilizado.....	167
2.1 PIC 18F8520.....	167
2.2 spi.c	172
2.3 Lectura.c.....	176
2.4 ftp.c	183
2.5 memoria.c	184
2.6 index.php.....	185
2.7 Grafica.php.....	188

2.8 Grafica_GUI.php	191
2.9 movil.php	194
2.10 gui.py	195
2.11 ActivityMain.java	199
2.12 Content_main.xml	208
LICENCIA DE USO	211

ÍNDICE DE FIGURAS

Figura 1: Especificaciones de la tensión según AENOR en su norma 50160.	14
Figura 2: Especificaciones de la tensión según AENOR en su norma 50160.	14
Figura 3: Principales problemas de la red eléctrica.	15
Figura 4: Empresas entrevistadas en el estudio sobre los costes de la mala calidad de la energía.....	15
Figura 5: Coste de una interrupción de tensión dependiendo de su duración. ...	16
Figura 6: Coste de una interrupción de tensión dependiendo de su duración y del consumo eléctrico.....	16
Figura 7: Tiempo total de las interrupciones.	17
Figura 8: Coste anual que suponen las interrupciones en el consumo eléctrico. 17	17
Figura 9: Diagrama de bloques del sistema completo.....	18
Figura 10: Red trifásica a 4 hilos.	21
Figura 11: Esquema modelo del ADE7912/ADE7913.....	23
Figura 12: diagrama de bloques del ADE7912.	24
Figura 13: diagrama de bloques del ADE7913.	24
Figura 14: Rango de voltaje de entrada para el pin IP (intensidad).....	25
Figura 15: Rango de voltaje de entrada para el pin VP (Voltaje).....	25
Figura 16: Circuito trifásico a monitorizar. Voltaje.	26
Figura 17: señal de voltaje condicionada para su medida.....	26
Figura 18: Circuito trifásico a monitorizar. Intensidad.	27
Figura 19: Señal de intensidad acondicionada para su medida.....	27
Figura 20: Cronograma de la acción de lectura en el ADE7912 en el modo burst.	29
Figura 21: Cronograma de la acción de lectura en el ADE7912 en el modo normal.	29
Figura 22: Cronograma de la acción de escritura en el ADE7912.....	30
Figura 23: Sincronización de dos ADE7912.....	32
Figura 24: Selección de frecuencias en el registro CONFIG y el valor inicial del contador.....	33
Figura 25: Sincronización de la toma de valores en tres ADE7912.....	33
Figura 26: Registro CNT_SNAPSHOT con el valor del contador [12 bits].....	33
Figura 27: COUNTER1 y COUNTER0 forman el contador de 12 bits.....	34
Figura 28: Diagrama del conversor DC-DC de aislamiento.	37
Figura 29: señal PWM a partir de la señal de reloj.	37
Figura 30: señal PWM de 4 ADE7912 trabajando conjuntamente.	38
Figura 31: Diagrama de bloques del ADC.....	39

Figura 32: Nivel de ruido en la banda muestreada.	40
Figura 33: Efectos del Aliasing.	41
Figura 34: Filtro RC paso bajo.	41
Figura 35: Análisis en frecuencia del filtro RC paso bajo calculado.	42
Figura 36: Frecuencia de corte en detalle.	42
Figura 37: Generación del CRC de las lecturas del ADC.	43
Figura 38: Diagrama del generador del ADC_CRC.	43
Figura 39: Diagrama del encendido de todos los ADE7912.	46
Figura 40: Alimentación del circuito integrado ADE7912.	46
Figura 41: Relé de estado sólido G3PH-5150B DC5-24.	47
Figura 42: PIC18F8520.	48
Figura 43: TCP40666BP.	49
Figura 44: DM7404.	50
Figura 45: Conversor lógico bipolar.	50
Figura 46: Raspberry pi 3 modelo B.	52
Figura 47: GPIO de la Raspberry Pi 3 model B.	53
Figura 48: Resumen completo del hardware de la Raspberry Pi 3 model B.	53
Figura 49: LCD 7" táctil para la Raspberry Pi 3.	56
Figura 50: logotipo de la Python Software Foundation.	57
Figura 51: logotipo Android Studio.	58
Figura 52: Sistema de alimentación ininterrumpida.	61
Figura 53: Fuente de alimentación 230AC – 14.5DC.	63
Figura 54: Señal rectificada (Vn001) y señal del regulador (Vn002).	63
Figura 55: Circuito controlador de la batería.	64
Figura 56: Circuito regulador a 5V.	65
Figura 57: Circuito regulador a 3.3V.	65
Figura 58: PCB Sistema de alimentación ininterrumpida en 3D.	66
Figura 59: PCB Sistema de alimentación ininterrumpida en negativo.	66
Figura 60: PCB Sistema de alimentación ininterrumpida en negativo en 3D.	66
Figura 61: Serigrafía capa superior.	67
Figura 62: Pasta de soldadura capa superior.	67
Figura 63: cobre capa superior.	67
Figura 64: Cobre capa inferior.	67
Figura 65: Dibujo de los taladros.	67
Figura 66: Esquema placa principal.	69
Figura 67: Conexionado del ADE7912.	71
Figura 68: Conexionado entre ADE7912.	71
Figura 69: Multiplexado del SPI en el PIC.	72
Figura 70: nivelado de señales lógicas.	72
Figura 71: PIC 18F8520.	73

Figura 72: PCB placa principal en 3D.....	74
Figura 73: PCB placa principal en negativo.....	74
Figura 74: PCB placa principal en 3D.....	74
Figura 75: serigrafía capa superior.....	75
Figura 76: cobre capa superior.	76
Figura 77: pasta soldadura capa superior.....	77
Figura 78: cobre capa inferior.....	78
Figura 79: dibujo de los taladros.....	79
Figura 80: imagen representativa del bus de conexiones disponible.....	80
Figura 81: imagen representativa de las conexiones al cuadro eléctrico.....	80
Figura 82: Esquemático placa de adquisición de datos.	81
Figura 83:PCB adquisición de datos en 3D.....	82
Figura 84: Negativo de la PCB.	82
Figura 85: PCB adquisición de datos en 3D.....	82
Figura 86: cobre capa superior.	83
Figura 87: pasta de soldadura parte superior.....	83
Figura 88: cobre capa inferior.....	84
Figura 89: dibujo de los taladros.....	84
Figura 90: diagrama de bloques del sistema.	85
Figura 91: Diagrama de bloques servidor web y aplicación android.....	86
Figura 92: Diagrama de bloques interfaz gráfica	87
Figura 93: Diagrama de flujo PIC.....	88
Figura 94: Diagrama de flujo script SPI	89
Figura 95: Diagrama de flujo script lectura.....	90
Figura 96: Diagrama de flujo del script ftp.....	92
Figura 97: Diagrama de flujo del script memoria.	93
Figura 98: Página web.....	94
Figura 99: Diagrama de flujo pagina web.	95
Figura 100: Diagrama de flujo función grafica.php.....	96
Figura 101: Interfaz gráfica táctil.	97
Figura 102: Interfaz gráfica táctil.	97
Figura 103: Diagrama de flujo interfaz gráfica.....	98
Figura 104: Capturas de pantalla de la aplicación.	99
Figura 105: Capturas de pantalla de la aplicación.	99
Figura 106: Capturas de pantalla de la aplicación.	100
Figura 107: Diagrama de flujo script movil.php.....	101
Figura 108: Estructura del proyecto.....	102
Figura 109: Editor gráfico Android Studio.....	102
Figura 110: Estructura de la clase MainActiviy en la aplicación android.....	103
Figura 111: Diagrama de flujo OnCreate(Bundle).....	104

Figura 112: Diagrama de flujo de la clase ‘CargalmImagenes’.....	105
Figura 113: Diagrama de flujo de la función ‘descargalmImagen’.....	106
Figura 114: Montaje para la captación de datos.....	107
Figura 115: Generación de la señal de voltaje.	107
Figura 116: Generación de la señal de intensidad.	108
Figura 117: Lectura y superposición de las señales de voltaje e intensidad.....	108
Figura 118: Lectura y superposición de las señales de voltaje e intensidad.....	108
Figura 119: Placa UNI DS3.	109
Figura 120: Conexión del MSP3204.	110
Figura 121: Cronograma de uso del ADC MSP3204.	110
Figura 122: Pantalla táctil conectada a la raspberry pi.	111
Figura 123: UNI-DS3 con las conexiones pertinentes.	111
Figura 124: UNI-DS3 con las conexiones pertinentes.	112
Figura 125: Pantalla táctil con la interfaz gráfica en ella.....	112

INTRODUCCIÓN

1. Antecedentes

Es necesario para la obtención de título de Graduado en Ingeniería en Electrónica Industrial y Automática la realización de un Trabajo Fin de Grado.

Con esta finalidad, el alumno D. Rubén García Segovia, ha realizado este Trabajo Fin de Grado que tiene por título "Diseño de un dispositivo de monitorización de redes eléctricas en el entorno del internet industrial de las cosas (IIOT)".

Este trabajo ha sido tutorizado por el profesor D. Miguel Martínez Iniesta, perteneciente al Departamento de Ingeniería Eléctrica, Electrónica, Automática y Comunicaciones y que imparte docencia en la Escuela de Ingenieros Industriales de Albacete de la Universidad de Castilla -La Mancha.

2. Objeto del trabajo

El objetivo general del TFG será la realización, diseño y prototipado de un dispositivo capaz de medir y calcular diversos parámetros de la red eléctrica de una industria y almacenar dicha información en una base de datos.

La base de datos será accesible a través de internet, contando con una interfaz de usuario funcional y que permita la visualización y ordenación de toda la información recogida por el dispositivo central.

También será accesible a través de una pantalla táctil, contando con una interfaz gráfica funcional y que permita la visualización y actuación sobre todo el sistema de una forma sencilla e intuitiva.

Por último, se va a desarrollar una aplicación móvil para el entorno Android para facilitar un acceso más rápido a la información, aunque dicha aplicación sólo permitirá la visualización limitada (los datos más importantes) de la información.

Con esto se pretende crear una herramienta que facilite a una empresa el control energético, el análisis de la red en busca de fallos o problemas en el suministro eléctrico, y encontrar fallos de rendimiento y/o funcionamiento en sus propias máquinas.

El objetivo general se va a dividir en diferentes objetivos parciales que se irán detallando a lo largo del desarrollo del TFG:

- Medición de voltajes e intensidades en una red eléctrica industrial, aislando el circuito de control de la toma de fuerza. La comunicación mediante el protocolo serie SPI de los distintos sensores con la unidad principal.
- Cálculo de parámetros básicos en base a las lecturas de voltajes e intensidades realizados: Potencia activa, potencia aparente, potencia reactiva, factor de potencia, etc.
- Almacenamiento de toda la información en una base de datos localizada en un computador de placa reducida. Comunicación entre el computador de placa reducida y la unidad principal para la obtención de toda la información obtenida y calculada.
- Creación de una interfaz de usuario que permita acceder a toda la información almacenada en la base de datos a través de un servidor web.
- Creación de una interfaz gráfica que permite acceder a toda la información almacenada en la base de datos a través de una pantalla táctil.
- Creación de una aplicación móvil que permita visualizar de forma simple la información más importante.

3. Justificación

El creciente auge de dispositivos conectados a internet, o ‘Internet Of Things’ está empezando a ser aplicado a la industria para conseguir plantas industriales más eficientes y autónomas, con un control descentralizado y remoto. Debido a esto son muchas las empresas especializadas del sector de la electrónica las que están desarrollando dispositivos siguiendo esta corriente. Por ello, siguiendo esta tendencia, se ha escogido un trabajo fin de grado enfocado al Internet Industrial de las cosas (IIOT).

La temática a desarrollar en este TFG está relacionada con las siguientes asignaturas del plan de estudios de Grado en Ingeniería en Electrónica Industrial y Automática:

- Tecnología Electrónica: Ensamblado de PCB.
- Sensores y Actuadores: Acondicionamiento de sensores.
- Tecnología Analógica: Diseño de circuitos analógicos.
- Tecnología Eléctrica: Cálculo y análisis de la red eléctrica.
- Electrónica de Potencia: Cálculo y análisis de la red eléctrica.
- Electrónica Industrial: Calidad de la energía.
- Procesado Digital de Señales.
- Técnicas de Simulación Avanzadas: Simulación de circuitos eléctricos analógicos y digitales.
- Digital I: Diseño de circuitos digitales y conversores analógico-digitales.
- Digital II: Programación de microcontroladores.
- Sistemas Empotrados: Protocolos de comunicación y raspberry pi.

Además, a título personal, siempre me ha interesado el campo de la automática y el entorno industrial. Esto, junto con mi interés por las redes eléctricas, han dado como resultado la idea de juntar en un único proyecto todas estas disciplinas para con ello poder aumentar mis conocimientos en todas estas ramas de la ingeniería.

DESCRIPCIÓN

Primero se va a exponer y explicar qué es la calidad de la energía y el porqué de su importancia en la industria.

Calidad de la energía

La calidad de la energía es un factor muy importante a tener en cuenta en una industria, debido a que el deterioro, la avería o los problemas que son dados en las máquinas de la empresa están directamente relacionados con la calidad de la energía consumida. A continuación se exponen dos definiciones de calidad de energía según dos grandes organismos internacionales de ingeniería:

Según el IEEE (Institute of Electrical and Electronic Engineers), la calidad de la energía es la alimentación y puesta a tierra de los equipos electrónicos sensibles de un modo adecuado para su correcto funcionamiento.

Según el IEC (International Electrotechnical Commission) son las características de la electricidad en un punto dado de la red eléctrica, evaluadas con relación a un conjunto de parámetros técnicos de referencia.

El principal problema de la calidad de la energía es solventable a través de dos soluciones: construcción de máquinas menos sensibles y la instalación de dispositivos a medida que se encarguen de subsanar la mala calidad de la energía.

El concepto de calidad de energía no es nuevo, pero debido a la tendencia de la industria hacia usar semiconductores de potencia no lineales (como inversores, convertidores, tiristores, etc) y el aumento de la sensibilidad de los equipos finales han obligado a la industria a diseñar dispositivos específicos para mejorar este aspecto de la electricidad.

Prueba de que la preocupación por la calidad de la energía va en aumento es la creación de nuevas normas AENOR sobre las especificaciones de la tensión.

Característica	Especificaciones de la tensión según la norma EN 50160		Observaciones
	Baja tensión	Media tensión	
Frecuencia	50 Hz \pm 1% (10s / 95% / semana) 50 Hz + 4% (todo el tiempo / 100% / semana)		
Valor eficaz de la tensión	Valor nominal Un	Valor declarado Uc (1 \leq Uc \leq 35 kV)	
Variaciones de tensión	Un \pm 10% (10m / 95% / semana)	Uc \pm 10% (10m / 95% / semana)	
Variaciones rápidas de tensión	5% de Un 10% de Un esporádicamente Flicker: Pit \leq 1(10m / 95% / semana)	4% de Uc 6% de Uc esporádicamente Flicker: Pit \leq 1(10m / 95% / semana)	Pit: severidad del Flicker a largo término
Huecos de tensión	Unas decenas hasta 1000 por año, profundidad inferior a 60% de Un y duración inferior a 1s.	Unas decenas hasta 1000 por año, profundidad inferior a 60% de Uc y duración inferior a 1s.	Valores indicativos
Armónicos de tensión	U3 \leq 5% U9 \leq 1,5% U15 \leq 0,5% U21 \leq 0,5% U5 \leq 6% U7 \leq 5% U11 \leq 3,5% U13 \leq 3% U17 \leq 2% U19...U23 \leq 1,5% U2 \leq 2% U4 \leq 1% U6...U24 \leq 0,5% THD \leq 8% considerando hasta orden 40		Porcentajes de Uc o de Un
Desequilibrio de tensión	Componente de secuencia inversa inferior al 2% de la componente de la secuencia directa (10m / 95% / semana)		

Figura 1: Especificaciones de la tensión según AENOR en su norma 50160.

Característica	Especificaciones de la tensión según la norma EN 50160		Observaciones
	Baja tensión	Media tensión	
Interrupciones de corta duración	Orden de varios cientos por año, con duración inferior a 1s en el 70% de los casos.	Orden de varias decenas hasta cientos por año, con duración inferior a 1s en el 70% de los casos.	Valores indicativos
Interrupciones de larga duración	Duración superior a 3m De 10 a 50 por año, sin considerar interrupciones programadas		Valores indicativos
Sobretensiones repetitivas entre conductores y tierra	1,5 kV valor eficaz	1,7Uc en sistemas con neutro puesto tierra 2,0Uc en sistemas con neutro aislado o puesta a tierra resonante.	
Sobretensiones transitorias entre conductores y tierra	6 kV valor eficaz		
Interarmónicos	En estudio		
Transmisión de señales	3s / 99% / día	3s / 99% / dia. En estudio para la banda comprendida entre 9 y 95 kHz	

Figura 2: Especificaciones de la tensión según AENOR en su norma 50160.

La inmunidad de los equipos es clasificada en tres índices distintos:

Índice	Inmunidad	Ejemplos
I	Alta	motores, transformadores, cargas resistivas
II	Moderada	Relés, PLCs, robótica
III	Baja	Ordenadores, DSPs, equipo médico

Tabla 1: Inmunidad frente a la calidad de la energía.

Las principales perturbaciones de la electricidad según el EPRI (Electric Power Research Institute) en 1994 son:

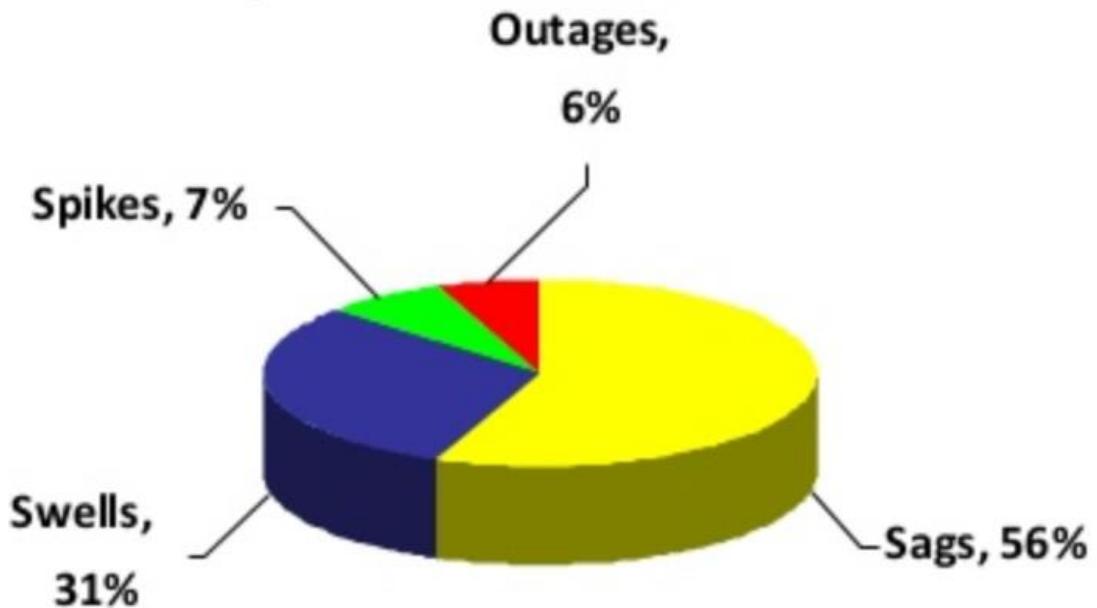


Figura 3: Principales problemas de la red eléctrica.

A pesar de ser un estudio de 1994, es muy esclarecedor conocer que el 56% de los fallos vienen dados por los huecos de tensión, el 6% por interrupciones de tensión, un 7% son transitorios y por último tenemos un 31% debido a sobretensiones.

El EPRI también ha realizado varios estudios donde estudian el impacto de la calidad de la energía en la industria. El estudio que se ha propuesto para estudio tiene por título 'The Cost of Power Disturbances to Industrial & Digital Economy Companies' y ha sido llevado a cabo por EPRI en 2001.

Para analizar y sacar conclusiones del estudio primero tenemos que saber cuales fueron las empresas entrevistadas:

	Digital Economy	Continuous Process Manufacturing (CPM)	Fabrication & Essential Services (F&ES)	Total
1 to 19 employees	179	166	159	504
20 to 249 employees	101	87	101	289
250 + employees	62	74	56	192
Total	342	327	316	985

Figura 4: Empresas entrevistadas en el estudio sobre los costes de la mala calidad de la energía.

Como se puede observar en la figura 4, la población del estudio cuenta con un total de 985 empresas, donde han sido agrupados en 3 categorías según tamaño de empresa y en tres tipos distintos de empresa. Es apreciable que la proporción de empresas grandes es menor por lo que la media en cuanto a los gastos ocasionados por la calidad de la energía estará viciada a la baja.

La figura 5 muestra el gasto que conlleva una interrupción de tensión en una industria. Como se ha podido ver anteriormente, este tipo de perturbaciones suelen suponer el 6% del total de perturbaciones de la red.

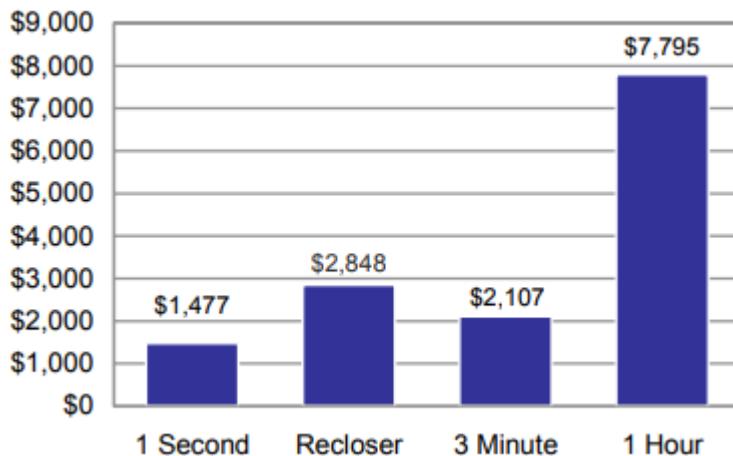


Figura 5: Coste de una interrupción de tensión dependiendo de su duración.

La siguiente figura 6 muestra el gasto por interrupción de voltaje teniendo en cuenta el consumo eléctrico medio de la empresa y la duración de la interrupción, ofreciéndonos datos más interesantes que la anterior gráfica.

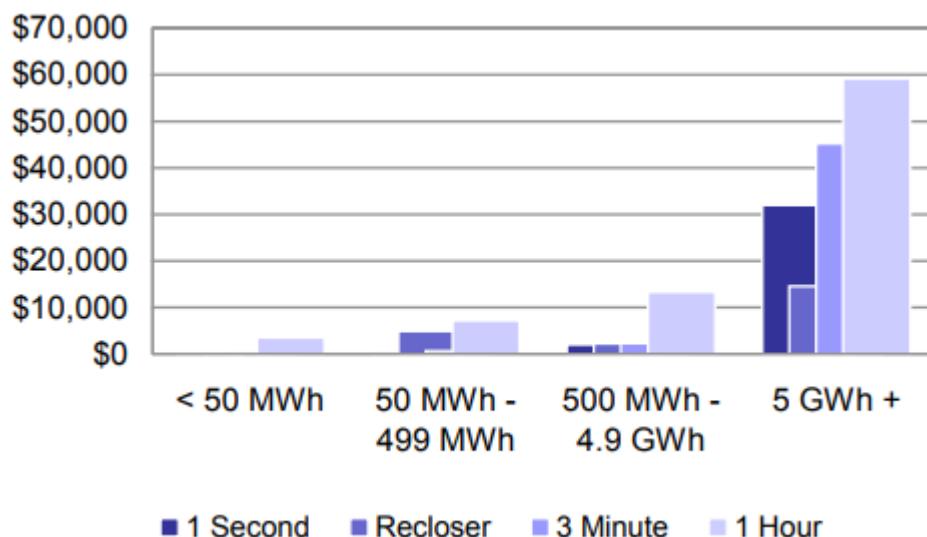


Figura 6: Coste de una interrupción de tensión dependiendo de su duración y del consumo eléctrico.

Una vez que se ha estudiado cuantas pérdidas se pueden generar dependiendo del tiempo que una interrupción se mantiene, es interesante conocer con qué frecuencia se dan estas interrupciones y cuanto suelen durar. Esto se puede apreciar en la Figura 7:

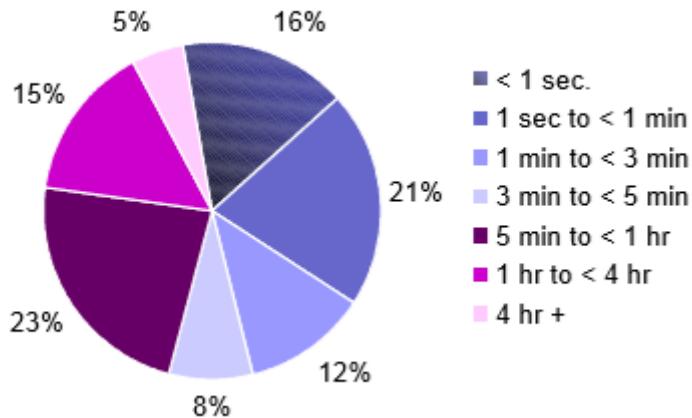


Figura 7: Tiempo total de las interrupciones.

Por último se tiene el coste anual que ha supuesto para la empresa entrevistada las interrupciones en su consumo eléctrico desglosado en 4 diferentes grupos de empresas dependiendo de su consumo total.

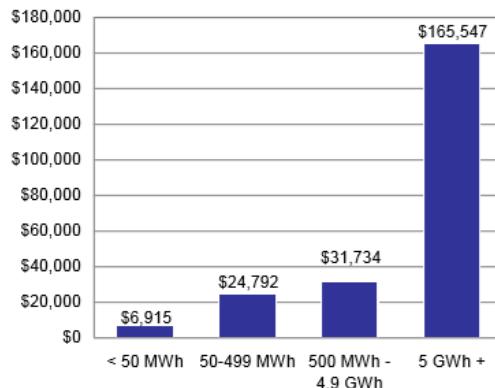


Figura 8: Coste anual que suponen las interrupciones en el consumo eléctrico.

Si se tiene en cuenta que este estudio sólo ha profundizado en las interrupciones del suministro eléctrico (que sólo suponen el 6% de los fallos en la red eléctrica) se pone en manifiesto la importancia de la calidad de la electricidad en la industria.

Diagrama de bloques del dispositivo

Antes de comenzar con la división estructural del proyecto en diferentes apartados o capítulos se ha realizado un diagrama de bloques para explicar de una forma más clara y sencilla el conjunto del dispositivo:

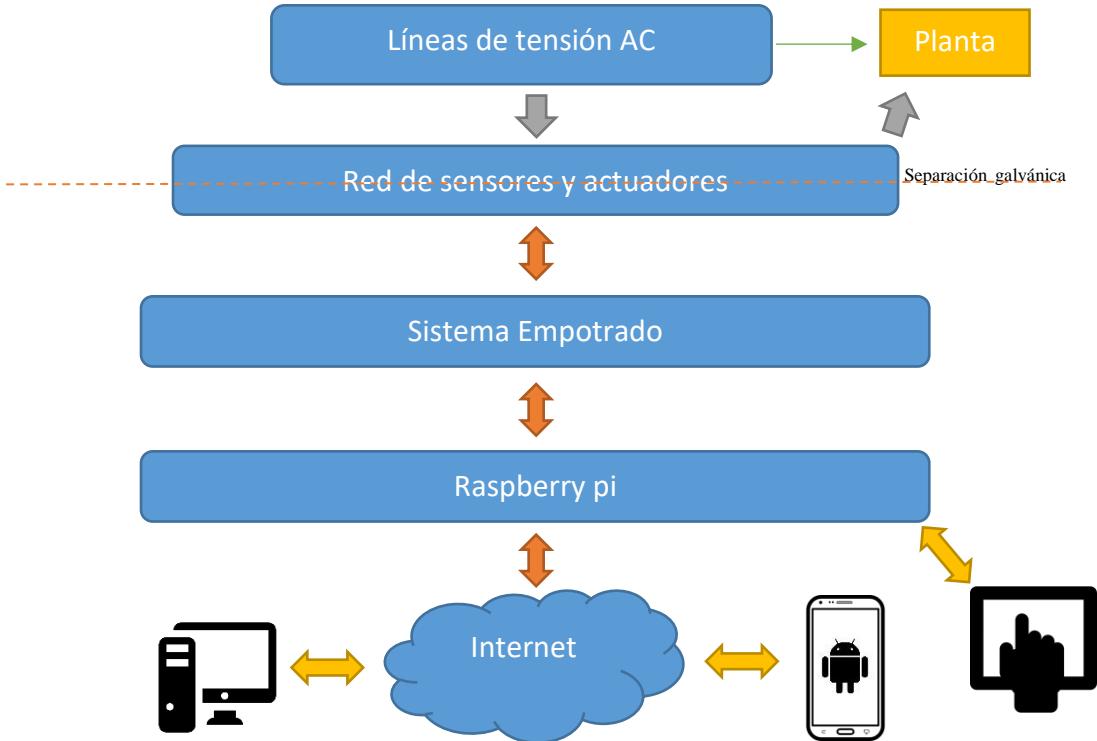


Figura 9: Diagrama de bloques del sistema completo.

El trabajo se ha estructurado en diferentes apartados o capítulos en los que se ha dividido el desarrollo del TFG para una mejor aclaración y entendimiento del sistema.

Medición de parámetros de red

La medición del voltaje de cada una de las tres líneas de fase con respecto a tierra y el voltaje de la línea neutra con respecto a la línea de tierra. También se va a medir las 4 intensidades circulantes por cada una de estas 4 líneas comentadas, así como la temperatura ambiente del cuadro eléctrico.

Todas estas mediciones se van a realizar en un circuito diferente al de control y se contará con un aislamiento galvánico para evitar fallos o problemas ocasionados por sobretensiones o fluctuaciones en la red.

Comunicación mediante el protocolo SPI

La transmisión de los parámetros de red y la unidad principal se harán a través del protocolo SPI, por lo que dicho protocolo será objeto de estudio.

Cálculo de parámetros de red

El post-procesado de los parámetros de red voltaje, intensidad y temperatura se realizará en el computador de placa reducida aprovechando su amplia capacidad de cálculo. Una vez que se tengan calculados dichos parámetros, se podrá realizar un cálculo más complejo de los siguientes datos:

- **Factor de potencia:** El factor de potencia es la relación existente entre la potencia activa del circuito y la potencia aparente. Es un parámetro de gran importancia a la hora de analizar si nuestra carga funciona correctamente. Además, las compañías comercializadoras de electricidad penalizan duramente el consumo eléctrico con un factor de potencia bajo.
- **Potencia aparente, activa y reactiva:** La potencia aparente es resultado de la suma vectorial de la potencia activa y reactiva y por tanto es el mejor indicador de potencia total consumida por una instalación eléctrica.

La potencia activa es aquella potencia eléctrica que se transforma en calor o trabajo, por lo que es el mejor indicador para designar el trabajo realizado por una máquina.

Por último, la potencia reactiva es aquella potencia consumida mediante la generación de campos eléctricos y magnéticos, necesarios para el correcto funcionamiento de muchas máquinas.

- **Desequilibrios entre las fases:** Mediante la comparación de intensidades y voltajes se puede averiguar si existen desequilibrios de carga entre las fases, lo que generará desestabilidad a la red eléctrica y generación de armónicos indeseados.
- **Huecos de tensión:** Disminución brusca de tensión a un valor de entre el 90% y el 1% de la tensión nominal por un período de tiempo comprendido entre 1ms y 1 minuto. Si la duración es mayor se considera interrupción.
- **Sobretensiones:** Aumento brusco de la tensión a un valor de entre el 10% y el 80% que duración usual de unos pocos ciclos.
- **Espectro frecuencial:** La realización de una transformada de Fourier para el análisis del espectro frecuencial de la señal permite encontrar armónicos no deseados en la red. Dichos armónicos pueden hacer que nuestras máquinas se sobrecalienten, disminuyan su vida útil, funcionen erróneamente y hasta la avería de las mismas en el peor de los casos.

- **THD:** La distorsión armónica total es una unidad de medida de componentes armónicos en nuestra red eléctrica.
- **Consumo eléctrico:** Aparte del cálculo de las potencias instantáneas, también es de interés el almacenar información sobre el consumo eléctrico por minutos, horas, días, meses y años.

Almacenamiento de la información

La información recolectada sobre todos los parámetros leídos y calculados se organizarán de forma correcta en una base de datos dentro del computador de placa reducida.

Servidor Web

Toda la base de datos tiene que ser accesible mediante un servidor web que contará con una interfaz de usuario simple que permita organizar y mostrar toda la información de forma clara y concisa. Este servidor web ha de ser accesible desde cualquier dispositivo con conexión a Internet.

Aplicación móvil

Se contará con una aplicación para la plataforma Android que facilitará la conexión con el servidor y base de datos a través de un teléfono móvil o Tablet. Desde esta aplicación será visible la información y los parámetros de red más importantes.

Pantalla táctil

Se contará con una pantalla táctil de siete pulgadas para la visualización de los parámetros más importantes del servidor. Desde la pantalla táctil se podrá actuar sobre los actuadores y el sistema en general.

FTP

Se ha implementado el protocolo FTP (File Transfer Protocol) para poder acceder a todos los datos guardados dentro del servidor. También se podrá actualizar el software del sistema a través de él.

Sistema de alimentación ininterrumpida

El dispositivo contará con un sistema de alimentación ininterrumpida de modo que si la alimentación de entrada se interrumpe, el sistema pueda funcionar con una batería y por tanto sea capaz de alertar del problema.

DESARROLLO

1. Monitorización red eléctrica trifásica

1.1 Voltaje e intensidad

El punto inicial de nuestro sistema será la monitorización de los valores más básicos de una red eléctrica trifásica: El voltaje y la intensidad. La obtención de los mismos deberá ser con la mayor precisión posible debido a alto rigor con el que se espera obtener (y a que su vez se necesita) el cálculo de parámetros tan importantes como la frecuencia de la onda, factor de potencia, potencia consumida, etc.

La red eléctrica que se va a monitorizar no depende de una única carga ni será de obligado cumplimiento que dicha carga tenga que ser trifásica, por lo que la monitorización de la red debe ser construida a prueba de fallos o desequilibrios en la misma. En la siguiente figura 10 se puede ver una red eléctrica de ejemplo.

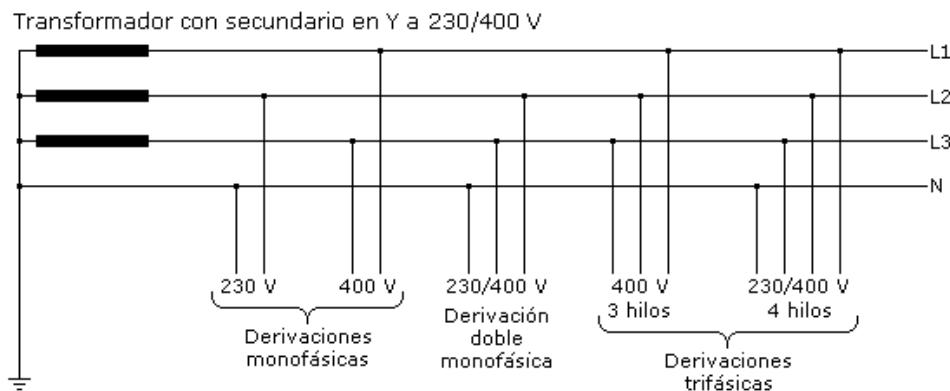


Figura 10: Red trifásica a 4 hilos.

Como el encargado de procesar los datos recogidos es un circuito digital de monitorización basado en un microcontrolador, se necesitará un conversor analógico digital de alta precisión. Una vez realizado un breve estudio de mercado, se ha seleccionado el conversor analógico digital ADE7912 y ADE7913 de ANALOG DEVICES que cumplen con todos los requisitos necesitados.

1.2 Conversor analógico digital ADE7912

Las características que se requieren del conversor analógico digital son las siguientes:

- Alta resolución para efectuar cálculos precisos, como la FFT (Fast Fourier Transform).
- Capacidad para medir distintos parámetros a la vez o capacidad de coexistir con otros conversores analógico digitales en paralelo.
- Interfaz de comunicación veloz y simple.
- Frecuencia de muestreo mayor que 1000Hz.
- Separación galvánica entre la entrada analógica y la salida digital.

El conversor encontrado es el ADE7912 y el ADE7913 que se ajustan perfectamente a las necesidades del sistema, ya que se trata de un conversor específico para la monitorización trifásica y el control de la calidad de energía. Según el fabricante, este conversor está especialmente indicado para:

- Equipos de medida polifásicos.
- Monitorización de la calidad de la energía.
- Inversores solares.
- PLCs industriales.
- Interfaz de sensores con aislamiento galvánico.

Debido a que el conversor ADE7912/ADE7913 está específicamente desarrollado para la monitorización de una red eléctrica, ofrece la capacidad de medir los valores de tensión, intensidad y temperatura ambiente al mismo tiempo, con una precisión de 24 bits y una tasa de muestreo de hasta 8000Hz.

Permite sincronización automática entre 4 ADE diferentes y el funcionando con el mismo cristal. Posee un protocolo de comunicación SPI con una separación galvánica entre la entrada analógica y la salida digital que facilita el uso de un microcontrolador para el sistema empotrado.

La única diferencia entre la versión 7912 y la versión 7913 radica en que la versión ADE7912 cuenta únicamente con 2 ADCs más la medición de temperatura y la versión 7913 cuenta con 3 ADCs. Como sólo se van a utilizar para la monitorización de voltaje, intensidad y temperatura, se usarán 4 ADE7912.

Tal y como se muestra en la figura 11, el propio fabricante propone un esquema de conexión:

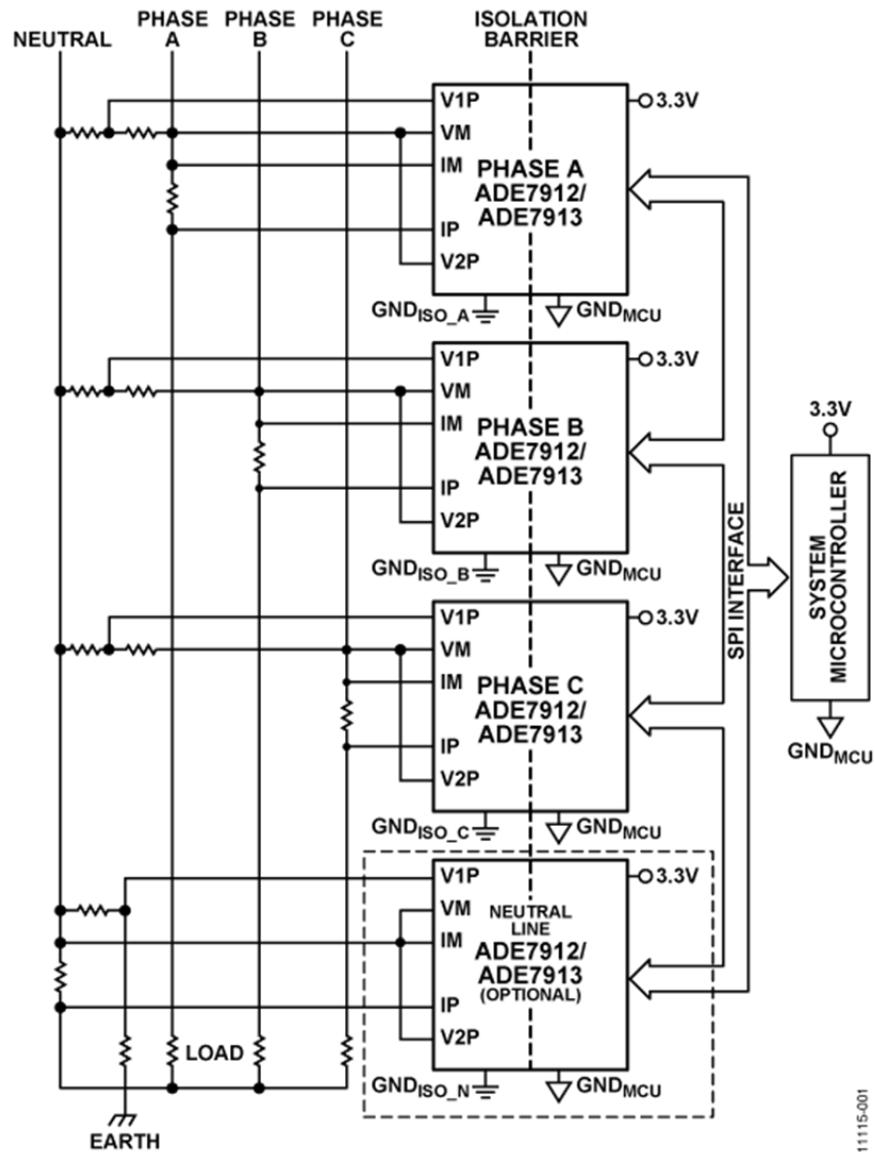


Figura 11: Esquema modelo del ADE7912/ADE7913.

Como se puede apreciar en la figura 11, este esquema modelo permite, mediante el uso de 4 conversores ADE, medir los voltajes entre cada una de las 3 fases con neutro y el voltaje de la línea neutro respecto a la línea de tierra (un parámetro importante a la hora de detectar desequilibrios) además de permitir la monitorización de las 4 corrientes circulantes y además, de necesitarse, la temperatura ambiente de los cables.

Para entender mejor el funcionamiento del conversor ADE791x y las diferencias existentes entre los dos modelos, en las siguientes figuras 12 y 13 veremos dos diagramas de bloques proporcionados por el fabricante:

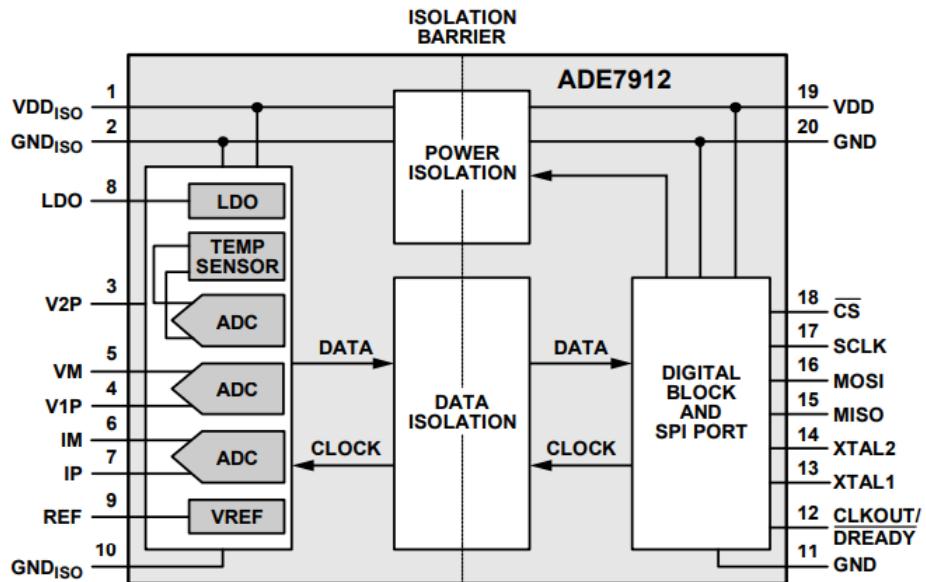


Figura 12: diagrama de bloques del ADE7912.

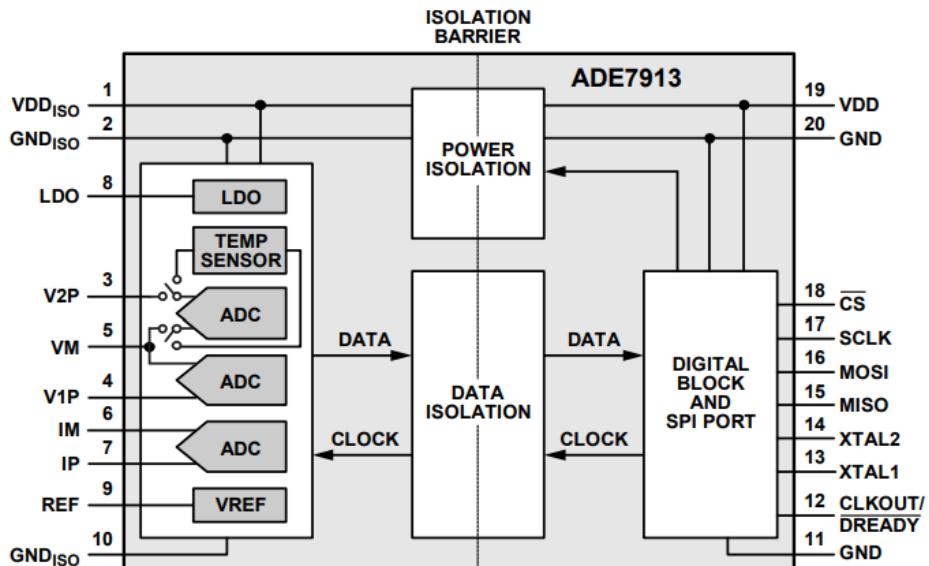


Figura 13: diagrama de bloques del ADE7913.

En ambos modelos se tienen 4 partes bien diferenciadas: el bloque digital de comunicación SPI y sincronización con el reloj (que son exactamente iguales en ambos modelos). La separación galvánica de alimentación tanto de la alimentación como de los datos y por último el bloque encargado de la conversión ADC. Ahí es donde radica la diferencia de ambos modelos: En el 7912 podemos ver que contamos dos ADC de libre uso junto con un tercer ADC destinado únicamente al sensor de temperatura interno, mientras que en el ADE 7913 este ADC se puede configurar para monitorizar una señal externa.

Para diseñar un correcto circuito de acondicionamiento de señal y conversión posterior, se necesita saber que rangos de voltaje se admiten como entrada en dichos circuitos integrados. Esta información está disponible en el manual del fabricante:

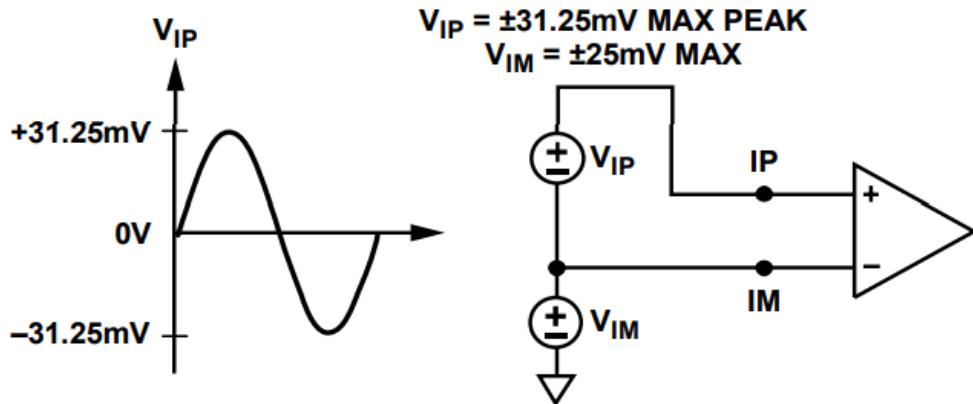


Figura 14: Rango de voltaje de entrada para el pin IP (intensidad).

Como se puede apreciar en la figura 14, el rango de entrada para el pin IM (el encargado de monitorizar la corriente circulante) no ha de ser mayor de una onda bidireccional de 31.25mV.

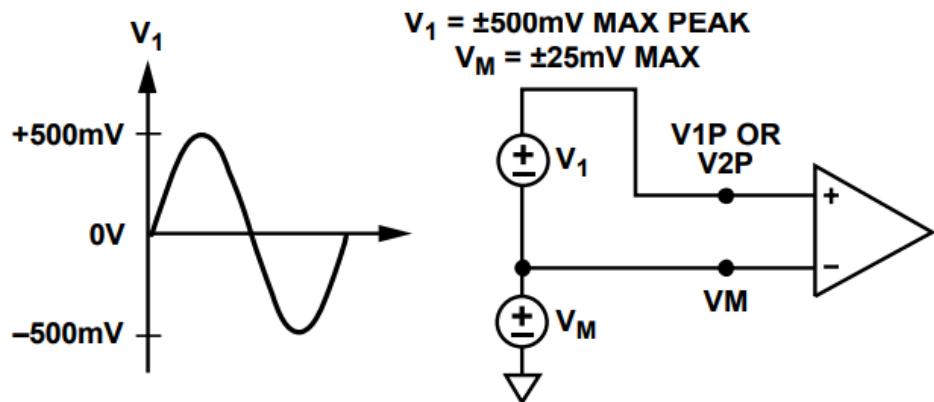


Figura 15: Rango de voltaje de entrada para el pin VP (Voltaje).

Finalmente, en la figura 15 se observa que para el pin de entrada del voltaje, el integrado admite hasta ± 500mV.

Estos voltajes han de ser acondicionados mediante el uso de resistencias de alta precisión y con un voltaje de película capaz de soportar el voltaje requerido entre sus terminales.

Para el cálculo de los valores de las resistencias, primero se ha de fijar el máximo voltaje que el dispositivo va a ser capaz de medir. El dispositivo está enfocado en el IIOT, por lo que fijaremos este voltaje en ± 1000V, así podrá ser capaz de medir hasta un incremento del 300% de la tensión esperada.

1.3 Medición voltaje

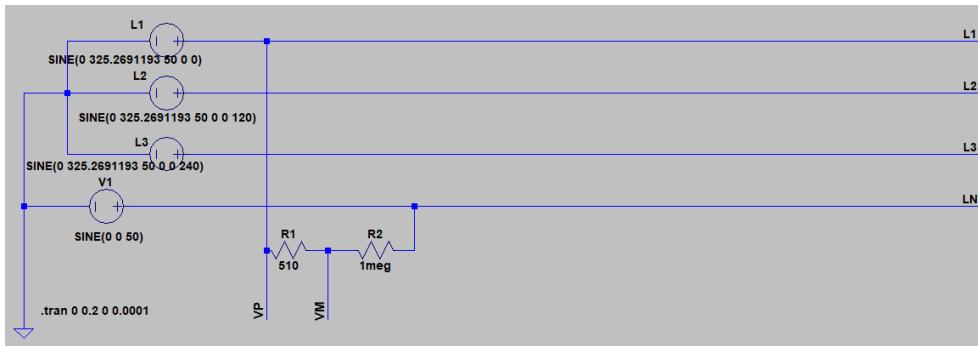


Figura 16: Circuito trifásico a monitorizar. Voltaje.

En la figura 16 se puede observar los valores escogidos de las resistencias para hacer el divisor de tensión. Para llegar a este cálculo hemos seguido los siguientes pasos:

Primero: Fijamos R_2 en $1M\Omega$. Se ha elegido un valor de resistencia muy alta para evitar la pérdida de energía en la medida.

$$I = \frac{V}{R}; i = \frac{1000}{10^6 + R} \quad (1)$$

$$V = I * R; 0.5V = \frac{1000}{10^6 + R} * R \quad (2)$$

Operando con las ecuaciones (1) y (2) se saca un valor de $R=500.25\Omega$. Como es un valor no normalizado, se ha tomado $R=510 \Omega$ y se realiza un estudio para averiguar hasta qué voltaje podrá ser capaz el sistema de medir:

$$0.5V = \frac{X}{10^6 + 510} * 510; X = 980.8921V \quad (3)$$

Teniendo en cuenta que no se han operado con valores eficaces, realizamos la última transformación:

$$V_{rms}: \frac{980.8921}{\sqrt{2}} = 693.5954V \quad (4)$$

Para comprobar todos estos cálculos, se ha realizado una simulación con una onda senoidal de amplitud 980.89V, obteniendo la siguiente respuesta:

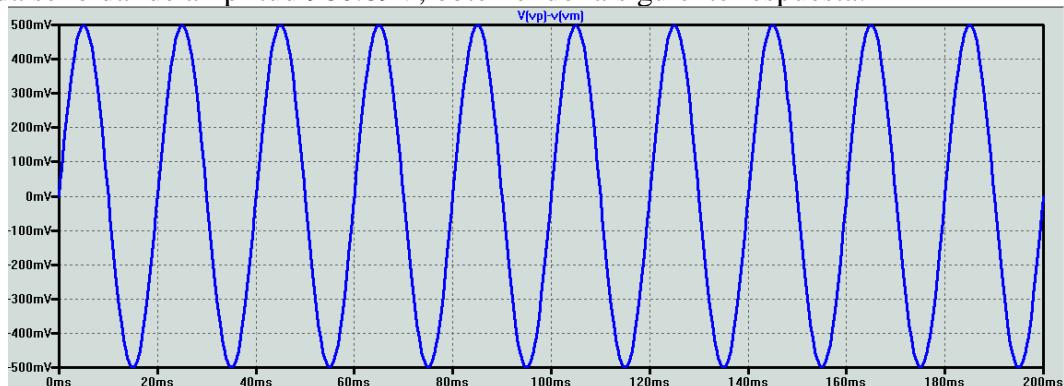


Figura 17: señal de voltaje condicionada para su medida.

1.4 Medición intensidad

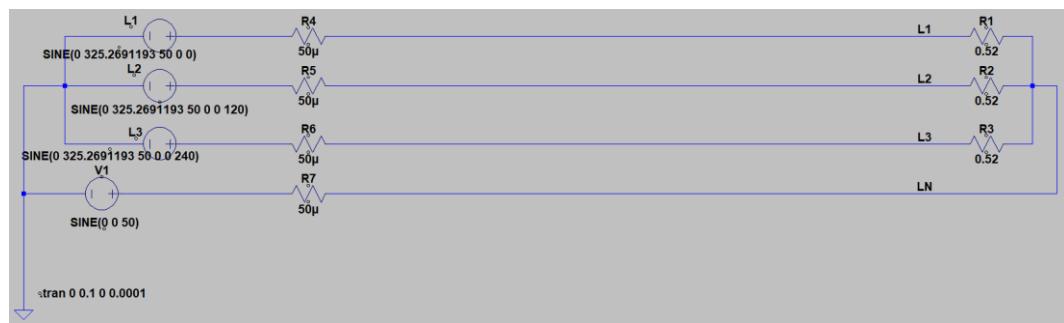


Figura 18: Circuito trifásico a monitorizar. Intensidad.

En la figura 18 se pueden observar los valores escogidos de la resistencia para realizar la medición de la intensidad circulante por la fase L1. De forma análoga, todas las demás líneas tendrán la misma resistencia para mantener el equilibrio de la red. Para llegar a este cálculo hemos seguido los siguientes pasos:

Primero se han de buscar los valores normalizados de las resistencias shunt. Para esto se ha realizado un seguimiento de mercado en los proveedores más reconocidos hasta encontrar la resistencia ‘WSBS8518L0500JK40’ de Vishay, cuyas características se encuentran en el anexo 1.

Una vez que ha sido seleccionada una resistencia, se va a proceder al cálculo de la máxima intensidad medible por el dispositivo:

$$V = I * R; \quad 31.25mV = i * 50\mu\Omega \quad (6)$$

$$i = 625A \quad (7)$$

Se ha de tener en cuenta que este valor es la amplitud máxima alcanzable, no la Irms. Para comprobar todos estos cálculos, se ha realizado una simulación con una resistencia de carga igual a 0.52Ω , lo que genera una intensidad pico de 625A, obteniendo como resultado en nuestras resistencias shunt:

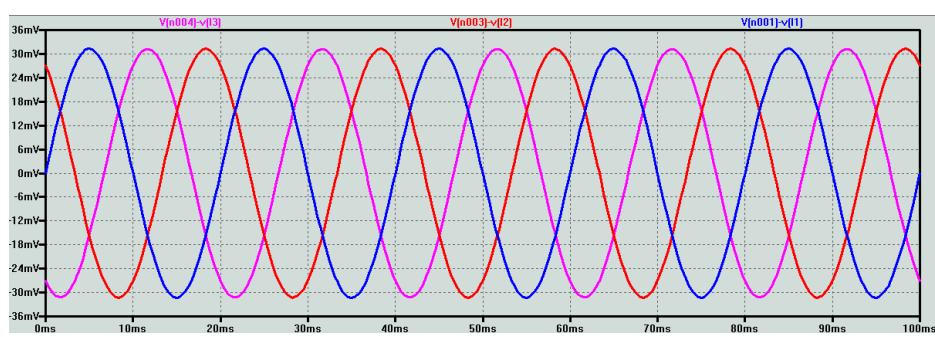


Figura 19: Señal de intensidad acondicionada para su medida.

1.5 Protocolo de comunicación SPI (Serial Peripheral Interface)

El protocolo SPI (Serial Peripheral Interface) es una interfaz de comunicación síncrona usado en comunicaciones de corta distancia, principalmente en los llamados sistemas empotrados. El protocolo fue elaborado por Motorola en los años ochenta y se ha convertido en uno de los mayores estándares de la industria electrónica.

El sistema tiene una comunicación full duplex usando una arquitectura maestro-esclavo, donde sólo puede haber un único maestro pero sí varios esclavos. Esto lo hace idóneo para su implementación en sistemas empotrados donde un único microprocesador PIC controla uno o varios circuitos integrados.

La principal ventaja de este sistema de comunicación es su versatilidad, dejando a decisión del diseñador cómo será su implementación. Esto, sin embargo, tiene los inconvenientes de que la implementación de un sistema a prueba de fallos tenga que ser vía software, junto con el tratamiento de los datos y el uso de los 4 pines necesarios para esta comunicación:

- **SCLK (Signal Clock):** Es el pin que transmite el pulso de sincronización. La gran ventaja del bus SPI es su gran versatilidad con este pin, ya que la frecuencia y los pulsos no tienen por qué ser regulares para una transmisión correcta de la información.
- **MOSI (Master Output Slave Input):** Pin usado para el envío de un dato por parte del maestro al esclavo.
- **MISO (Master Input Slave Output):** Pin usado para la recepción de un dato por parte del esclavo.
- **SS (Signal Select):** Cada esclavo se activará mediante un pin SS coordinado por el maestro. Esta es la mayor desventaja de un sistema de comunicación SPI, ya que requerirá de un uso excesivo de pines en el microcontrolador maestro.

Por lo tanto se ha decidido usar el protocolo de comunicación SPI para la transmisión de datos entre los sensores, el microcontrolador PIC y la raspberry Pi. A continuación se explicará en detalle el sistema de transferencia SPI que el conversor ADC utiliza.

1.6 Protocolo de comunicación SPI en el ADE7912

El conversor analógico-digital ADE7912 usa el protocolo SPI para su comunicación, por lo que se tendrá que estudiar cómo se hace uso de él y cuales son sus registros.

La frecuencia de comunicación del SPI tiene que estar comprendida entre 5.6MHz y 250Khz, y el pin MISO estará en alta impedancia cuando no se estén transmitiendo datos desde el conversor.

Este conversor tiene dos modos diferentes de lectura: modo normal y modo ‘Burst’ (ráfaga). Con el primer modo se podrán leer los registros de forma individual y selectiva, con el modo ráfaga se podrán leer los registros IWV, V1WV, V2WV, ADC_CRC, STATUS0 y CNT_SNAPSHOT de forma consecutiva. En la figura 20 se dispone del cronograma del modo ráfaga, mientras que en la figura 21 vemos el cronograma del modo normal.

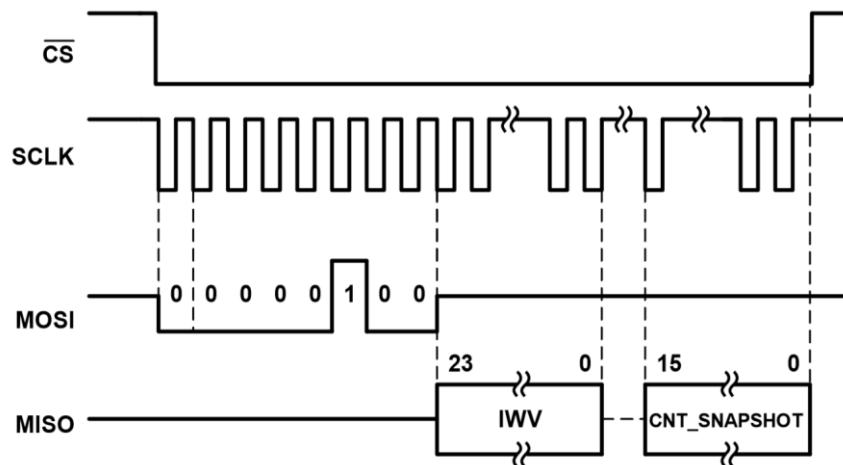


Figura 20: Cronograma de la acción de lectura en el ADE7912 en el modo burst.

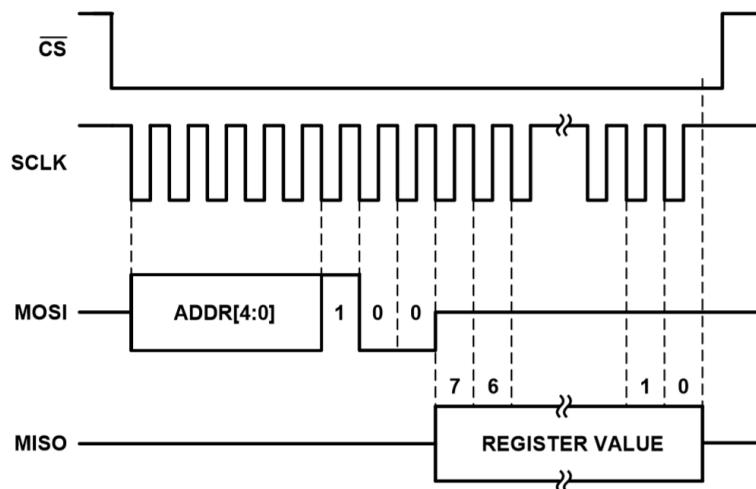


Figure 40. SPI Read Operation of an 8-Bit Register

Figura 21: Cronograma de la acción de lectura en el ADE7912 en el modo normal.

Para escribir un registro que tenga capacidad de ser escrito, se efectuará tal y como se puede apreciar en la figura 22:

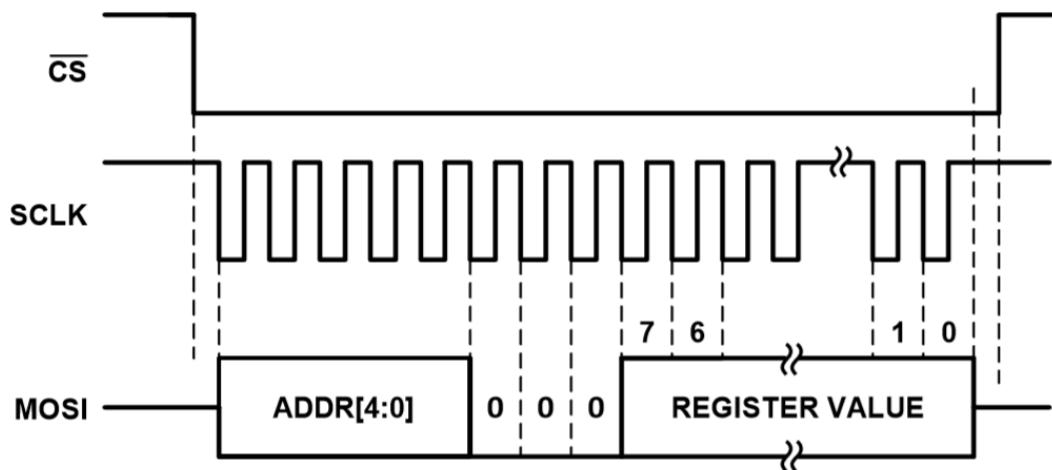


Figure 42. SPI Write Operation

Figura 22: Cronograma de la acción de escritura en el ADE7912.

1.7 Registros de interés en el ADE7912

A continuación se dispone de una tabla en la que están contenidos los registros de interés del ADE7912 que se tendrán que configurar debidamente para el correcto funcionamiento del conversor.

Dirección	Nombre	R/W	Longitud	Valor por defecto	Descripción
0x0	IWV	R	24 bits	0x000000	Valor instantáneo de I
0x1	V1WV	R	24 bits	0x000000	Valor instantáneo de V1
0x2	V2WV	R	24 bits	0x000000	Medida de la temperatura
0x5	ADC_CRC	R	16 bits	N/A	Valor CRC de IWV, V1WV y V2WV
0x7	CNT_SNAPSHOT	R	16 bits	0x00	Captura del contador usado para la sincronización de varios ADE7912
0x8	CONFIG	R/W	8bits	0x01	Registro de configuración
0x9	STATUS0	R	8 bits	0x01	Registro de status
0xA	Lock	W	8 bits	0x00	Registro de protección contra escrituras.
0xB	SYN_SNAP	W	8 bits	0x00	Registro de Sync
0xC	COUNTER0	R/W	8 bits	N/A	Contiene el contador de Sync
0xD	COUNTER1	R/W	8 bits	N/A	Contiene el contador de Sync
0xE	EMI_CTRL	R/W	8 bits	0xFF	Control del bloque PWM en el conversor DC-DC.
0xF	STATUS1	R	8 bits	0x00	Registro de status
0x18	TEMPOS	R	8 bits	0x00	Offset en la temperatura

Tabla 2: Registros de interés en el ADE7912.

1.8 Sincronización de varios ADE7912

Debido a que se van a utilizar más de un ADE7912 para la obtención de los datos de cada una de las líneas que se van a monitorizar, es importante que todos los conversores analógico-digitales estén sincronizados y tomen sus lecturas en el mismo instante de tiempo. De no ser así, el cálculo de los parámetros de red sería errónea. Para esta tarea el propio integrado del ADE7912 está preparado para tal fin y por tanto ofrece diversas características que ayudan a su sincronización.

Sólo uno de los cuatro ADE7912 estará sincronizado con un reloj, y éste será el encargado de pasar una señal de reloj válida a los otros 3 dispositivos que se van a utilizar. El primer ADE7912 usará el pin DREADY para tal fin, proporcionando en el pin XTAL1 las frecuencias a los otros 3 dispositivos. Por último, el pin DREADY de los dispositivos sincronizados por XTAL1 puede conectarse a una entrada del MCU ya que si el registro CONFIG tiene el bit0 puesto a 0, este pin entrará en estado bajo los primeros 64 ciclos de reloj de funcionamiento, avisando así al microcontrolador cuando debe empezar las lecturas por SPI. Todo esto se puede apreciar en la figura 23.

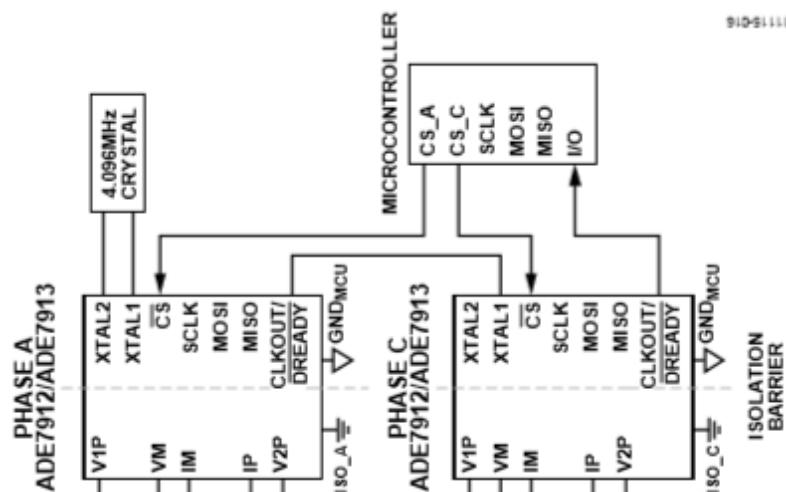


Figura 23: Sincronización de dos ADE7912..

La frecuencia de muestreo de todos los conversores deben estar asignados de igual forma, por lo que se van a configurar 8Khz en los cuatro ADE7912. Existe un contador que en función de la frecuencia del reloj y del periodo del ADC toma un valor u otro. Este contador sirve para la sincronización entre varios conversores analógico digitales. En la figura 24 se puede apreciar cual es el valor inicial de este contador dependiendo únicamente de la frecuencia de muestreo, y en el cronograma de la figura 25 se puede apreciar como se utiliza este contador para la sincronización de los periodos de ADC para finalmente conseguir una sincronización exacta en la toma de valores.

Bits[5:4] (ADC_FREQ) in CONFIG Register	ADC Output Frequency (kHz)	Counter C₀ Initial Value (CLKIN = 4.096 MHz)	Counter C₀ Initial Value as a Function of CLKIN
00	8	511	$\frac{CLKIN}{8000} - 1$
01	4	1023	$\frac{CLKIN}{4000} - 1$
10	2	2047	$\frac{CLKIN}{2000} - 1$
11	1	4095	$\frac{CLKIN}{1000} - 1$

Figura 24: Selección de frecuencias en el registro CONFIG y el valor inicial del contador.

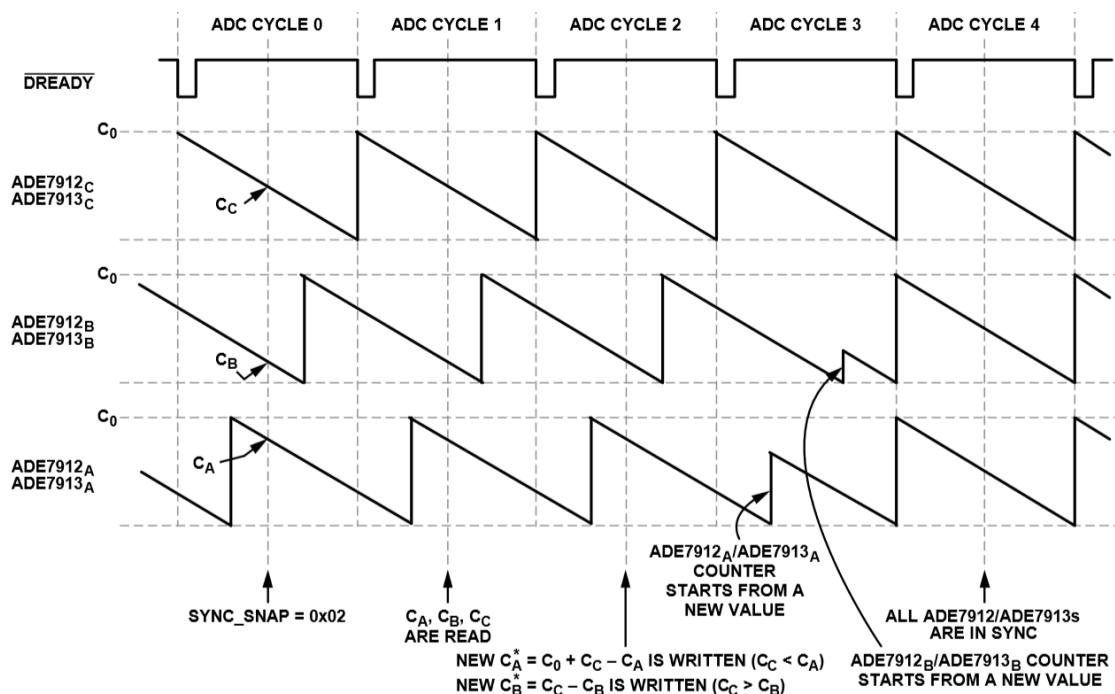


Figura 25: Sincronización de la toma de valores en tres ADE7912.

Cuando escribimos el byte SYNC_SNAP con un 0x02, el ADE7912 guarda el valor del contador. Un ciclo después el byte SYNC_SNAP vuelve a valer 0. El contador se guarda en el registro de 16 bits de CNT_SNAPSHOT. Idealmente la diferencia entre todos los CNT_SNAPSHOT debe ser 0, pero se considera aceptable un error de ± 1 . Dicho registro se puede observar en la figura 26.

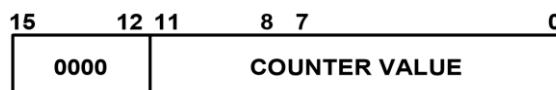


Figura 26: Registro CNT_SNAPSHOT con el valor del contador [12 bits].

Una vez que se ha detectado que los CNT_SNAPSHOT no están sincronizados, hay dos posibles formas para sincronizar de nuevo todos los ADE7912:

La primera opción es escribir un 0x01 en el byte SYNC_SNAP, esto forzará a todos los ADE7912 a empezar de nuevo, volviendo todos los contadores a sus estados iniciales, y, si todos los ADE7912 tienen la misma frecuencia de muestreo, la sincronización debería estar completa. Sin embargo, este comando suele generar un desfase de unos cuantos grados entre las fases, por lo que no se recomienda como método de sincronización, sí es utilizado en el momento de encendido de los conversores.

La segunda opción pasa por forzar el comienzo del contador (y por tanto periodo de muestro) a un determinado valor para sincronizar todos los contadores (Como se puede apreciar en la figura 25). Este inicio forzado hace uso de dos registros de 8bits, tal y como se puede apreciar en la figura 27:

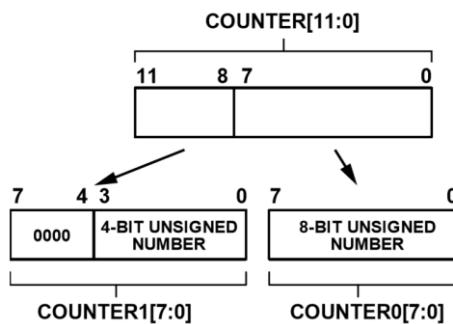


Figura 27: COUNTER1 y COUNTER0 forman el contador de 12 bits.

Además, de este modo no se tendrán que resetear todas las fases, sino que se irán sincronizando de forma correcta aquellas fases que estén afectadas por la desincronización. De este modo, siempre tendremos que sincronizar todos los ADE7912 con respecto a uno que se coja como referencia. En el caso de la figura 25, la referencia es Cc. Como norma general, la sincronización debe ser comprobada cada pocos segundos.

1.9 Medición de temperatura con el ADE7912

El conversor analógico-digital ADE7912 también posee un sensor de temperatura que se usará para medir la temperatura ambiente del dispositivo y cables. Como se está utilizando el ADE7912 y no el modelo ADE7913, los datos relativos a la temperatura los encontraremos en el registro V2WV, que en el modelo ADE7913 puede ser multiplexado para medir un voltaje más. El tiempo que requiere del sensor de temperatura para realizar la primera medición es de 5ms. La temperatura viene dada en grados centígrados y viene dada por la siguiente expresión:

$$\text{Temperatura} = \text{gain} \times V2WV + 8.72101 \times 10^{-5} * \text{TEMPOS} * 2^{11} - 306.47 \quad (8)$$

- Temperatura: Grados centígrados con un error de $\pm 5^{\circ}\text{C}$.
- Gain: Ganancia es igual a 8.72101×10^{-5} cuando el bit7 (BW) del registro de configuración es 0 y 8.21015×10^{-5} cuando es 1.
- TEMPOS: es el registro de 8 bits donde el offset del sensor de temperatura es almacenado. Este dato es estático y es almacenado durante el proceso de ensamblado del circuito integrado.

Por otro lado, el fabricante recomienda calibrar manualmente la ganancia del sensor de temperatura para obtener unas mejores mediciones. Dicho proceso se basa en medir la temperatura exacta mediante un termofusible a la vez que se está midiendo con el ADE7912. Se procederá a la lectura del registro V2WV y entonces la ganancia se expresará como en la ecuación 9.

$$\text{Gain} = \frac{\text{temperatura} + 306.47}{V2WV + k * \text{TEMPOS} * 2^{11}} \quad (9)$$

- Donde $k=1$ cuando el bit 7 (BW) del registro CONFIG es 0 y k es 1.062223 cuando el bit 7 (BW) del registro CONFIG es 1.

Sin embargo, para el uso que se planea dar al dispositivo, no se ha procedido a una mejor calibración ya que con una lectura aproximada de la temperatura es suficiente.

1.10 Registros de configuración

Se poseen un total de 5 registros de configuración distintos: CONFIG, EMI_CTRL, SYNC_SNAP, COUNTER0 y COUNTER1. También hay otro registro que interesa: Lock. A continuación se explican cada uno de estos registros:

CONFIG (8 bits)

bit	Nombre	Valor por defecto	Descripción
0	CLKOUT_EN	0	Habilita la funcionalidad CLKOUT en el pin #DREADY/CLKOUT dependiendo de su valor.
1	Reservado	0	No tiene función
2	PWRDWN_EN	0	'0' Activa y '1' desactiva el conversor DC-DC.
3	TEMP_EN	0	Sólo para ADE7913. Multiplexa la lectura de voltaje y el sensor de temperatura en V2WV.
5:4	ADC_FREQ	00	Configuran la frecuencia. (Figura 24).
6	SWRST	0	Se pone a uno para iniciar un software reset.
7	BW	0	Selecciona la banda del filtro digital paso bajo del ADC.

Tabla 3: Bits del registro CONFIG en detalle.

EMI_CTRL (8 bits)

Este registro se encarga de controlar el control del bloque de control PWM en el conversor DC-DC. Cada uno de sus pines tiene asignado un *slot* diferente (Pin0 – slot0, Pin1 – slot1, etc).

SYNC_SNAP (8 bits)

Bit	Nombre	Valor por defecto	Descripción
0	Sync	0	Cuando se pone a 1 se resetean todos los ADC y empiezan a la vez. (Ver apartado de sincronización).
1	Snap	0	Cuando se pone a 1 se guarda el valor del contador. (Ver apartado de sincronización)
7:2	Reservado	000000	No tienen funcionalidad.

Tabla 4: Bits del registro SYNC_SNAP en detalle.

Counter0 y Counter1

Usados para resetear el contador del ADC a un valor predeterminado para configurar la sincronización entre varios ADE7912. Ver apartado de sincronización y la figura 27

Lock

El registro Lock sirve únicamente para bloquear la escritura de todos los registros comentados anteriormente. Si se escribe un 0xCA en este registro, la escritura será bloqueada. Escribiendo un 0x9C se desbloquearán dichos registros. Este registro Lock sólo es de escritura, resultando 0x00 en cualquier caso de lectura.

1.11 Conversor DC-DC

El circuito integrado ADC7912 posee dos lados diferenciados y aislados entre sí: El lado del ADC y la red trifásica y el lado digital y de transmisión de datos. Estas dos partes tienen alimentación aisladas galvánicamente a través de un conversor DC-DC interno que es controlado por la entrada VDD. En la figura 28 se puede ver un esquema de dicho conversor.

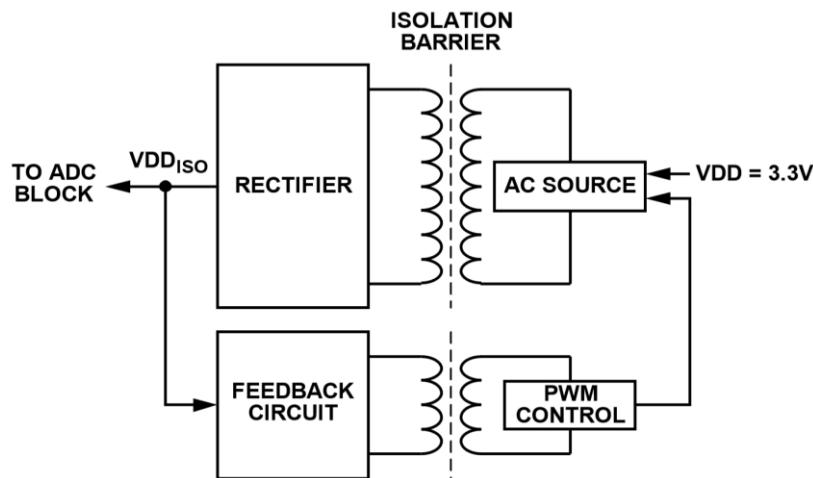


Figura 28: Diagrama del conversor DC-DC de aislamiento.

El circuito se alimenta a través de una VDD de 3.3V (Es importante mantener dicho valor entre 2.97V y 3.63V). Este voltaje es convertido a una tensión AC mediante un circuito oscilador. Mediante un transformador y un rectificador se obtiene una VDD' totalmente aislada de la alimentación primaria de 3.3V. El control PWM crea una salida constante de VDDiso en 2.8V modificando la entrada AC inicial. El bloque PWM trabaja a una frecuencia de CLKIN/4, y en la figura 29 se puede ver como es su funcionamiento:

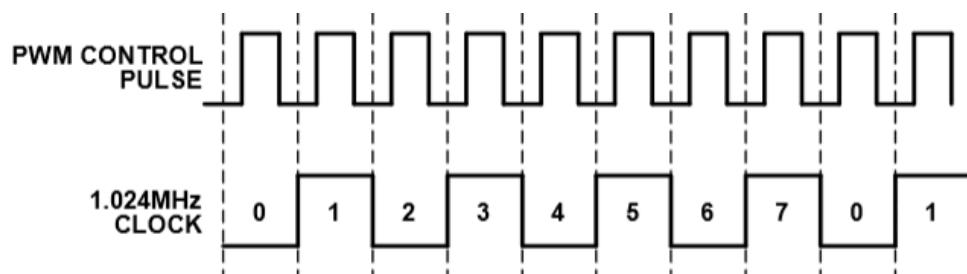


Figura 29: señal PWM a partir de la señal de reloj.

Cada vez que un pulso PWM es generado, la señal AC transmite una señal de alta frecuencia a través de la barrera de aislamiento para permitir una transferencia eficiente de energía entre los transformadores. Esta transferencia crea unas intensidades de alta frecuencia que pueden propagarse por el circuito de masa y los planos de alimentación, causando radiaciones EMI (Interferencias electromagnéticas). Por ello el fabricante especifica varios consejos a la hora de diseñar una placa PCB con este elemento. Además de las diversas técnicas de diseño

que existen para reducir las emisiones EMI, el ADE7912 contiene un registro EMI_CTRL, que permite su configuración a medida para reducir las radiaciones EMI.

El reloj que gobierna al PWM se divide en 8 diferentes *slots* (Se pueden ver enumerados en la figura 29). Cada bit del registro EMI_CTRL controla cada uno de esos *slots*. Cuando un bit de este registro está configurado a 1, en ese *slot* se generará un pulso. Para un correcto funcionamiento se recomienda que al menos 4 de estos 8 bits sean puestos a 1 mientras que los demás se mantienen a 0.

En el caso de la lectura polifásica de una red, habría que superponer dichos pulsos en parejas para reducir las emisiones EMI. En el caso de que tengamos cuatro ADE7912 a la vez, dos de ellos se podrían configurar con un 0xAA y los otros dos como 0x55. Este ejemplo se ve ilustrado en la figura 30.

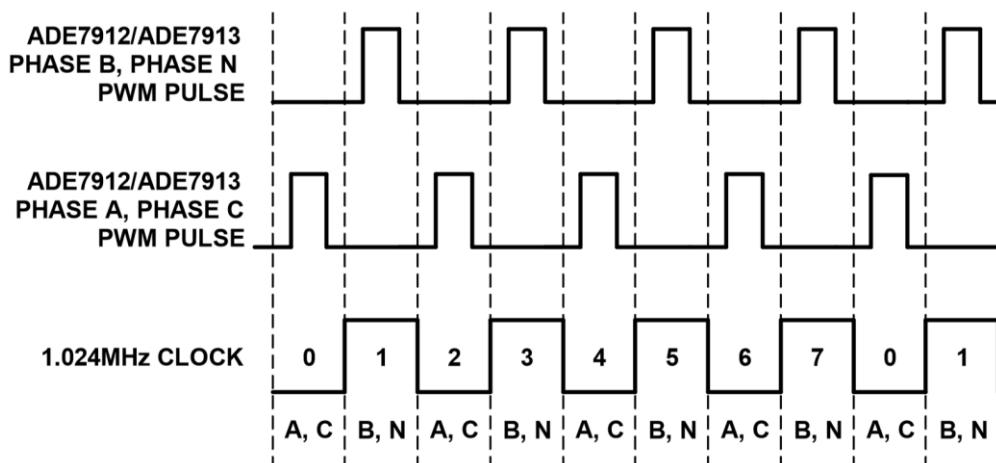


Figura 30: señal PWM de 4 ADE7912 trabajando conjuntamente.

1.12 Conversión analógico digital

La conversión analógico digital se realiza mediante un ADC sigma-delta de segundo orden. Sin embargo, para simplificar la explicación teórica, el diagrama de bloques de la figura 31 representa un ADC de primer orden, explicandose los principios fundamentales de cómo funciona un ADC sigma-delta.

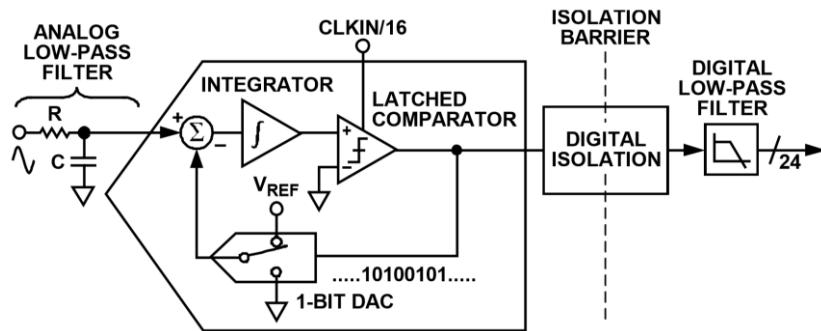


Figura 31: Diagrama de bloques del ADC.

La modulación Sigma-Delta convierte la entrada de la señal en una cadena continua de unos y ceros con una frecuencia determinada por la velocidad de muestreo.

En el ADE7912, la frecuencia de muestreo es igual a CLKIN/4. El bloque ‘1-bit DAC’ (un conversor digital analógico de 1 bit de resolución que devuelve masa o Vref) en el lazo de realimentación es controlado por la cadena de unos y ceros de la salida. La salida de este bloque se resta a la entrada de la señal analógica. Si la ganancia del lazo es demasiado alta, el valor medio de la salida DAC puede alcanzar el mismo valor que la señal de entrada. Para cualquier valor en la entrada en un determinado intervalo de tiempo, los datos del bloque del ADC son inútiles. Al coger muchas muestras y calcular su media, el resultado calculado termina siendo el correcto. Este promedio se completa en la segunda parte del ADC, con el filtro digital de paso bajo, después de haber atravesado el circuito de aislamiento hacia la otra parte del integrado ADE7912. Mediante el cálculo de la media de un gran número de bits recogidos por la modulación Sigma-Delta, el filtro paso bajo es capaz de producir un dato de 24 bits proporcional a la señal de entrada.

El conversor Sigma-Delta usa dos técnicas para alcanzar una gran resolución desde algo tan sencillo como es la técnica de conversion a 1-bit que se ha explicado anteriormente. La primera técnica es el sobremuestreo. Sobremuestrear la entrada significa que cogemos muestras a una frecuencia muy superior a la banda de interés (En el caso del ADE7912, puede ser de 40Hz hasta 2kHz o hasta 3.3Khz). Con esto se consigue repartir el ruido en una banda mayor. Al tener una banda mayor, el ruido generado en nuestra banda de interés es menor. Esto se puede apreciar en la figura 32.

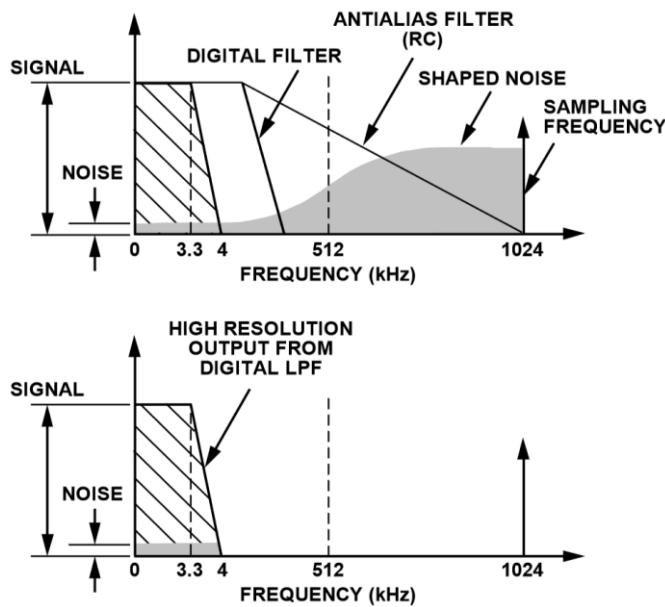


Figura 32: Nivel de ruido en la banda muestreada.

Sin embargo, esta técnica no es suficiente para quitar todo el ruido de la banda de interés, ya que para alcanzar los niveles que se requieren se necesitaría un sobremuestreo mayor del que se puede conseguir. Para mantener este sobremuestreo en un rango razonable, es posible un modelado del ruido. La técnica del modelado de ruido es una técnica para aumentar la relación señal a ruido aparente de la señal resultante. Esto se traduce en un menor ruido en la banda deseada y un mayor ruido en la banda no deseada. Este efecto se puede ver en la primera gráfica de la figura 32. Este proceso se realiza a través del integrador que se puede apreciar en la Figura 31.

La banda de interés es función de la frecuencia de salida del ADC (figura 24), la frecuencia de reloj de entrada al circuito integrado y el bit 7 (BW) del registro CONFIG. Si BW es 0 la banda del ADC será hasta 3.3Khz. Si es colocado a 1, la banda será hasta 2kHz. En la tabla 5 se puede ver con más detalle.

CLKIN (MHz)	ADC_FREQ	Frecuencia salida ADC	Banda del ADC con BW=0	Banda del ADC con BW=1
4.096	00	8000	3300	2000
	01	4000	1650	1000
	10	2000	825	500
	11	1000	412	250
4.21	00	8222	3391	2055
	01	4111	1695	1027
	10	2055	847	513
	11	1027	423	256
3.6	00	7031	2900	1757
	01	3515	1450	878
	10	1757	725	439
	11	878	362	219

Tabla 5: Tamaño de la banda de la señal dependiendo de varios factores.

1.13 Filtro RC paso bajo anti aliasing

A pesar de tener un filtro paso bajo digital, se va a necesitar de un filtro paso bajo analógico para evitar la producción del aliasing. El aliasing es el efecto que causa que ciertas frecuencias de unas señales analógicas se multipliquen a la hora de ser muestreadas. Exactamente cuando dichas frecuencias son más grandes que la mitad de la frecuencia de muestreo. (A esta frecuencia en concreto se le conoce como la frecuencia de Nyquist). Todos los ADC tienen este problema, sin importar que tecnología usen para la conversión. En la figura 33 se puede apreciar este efecto.

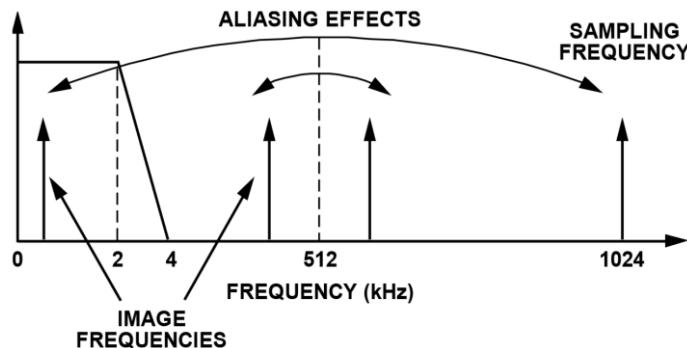


Figura 33: Efectos del Aliasing.

Estas frecuencias se verán duplicadas entorno $0.5f$, $1f$, $1.5f$, $2f$... (Siendo f la frecuencia de Nyquist del sistema). Por ello no se puede arreglar este problema mediante un filtro digital y necesitaríamos un filtro analógico en la entrada del sistema. Teóricamente será suficiente con un filtro paso bajo de 512KHz, pero el fabricante recomienda un filtro paso bajo RC (tal y como se ve en la figura 31) con una frecuencia de unos 5Khz de corte.

Se va a instalar un filtro RC con la siguiente configuración:

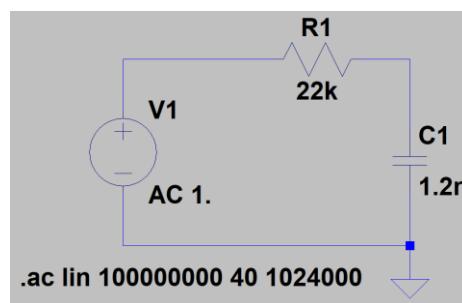


Figura 34: Filtro RC paso bajo.

El cálculo de estos valores se ha realizado mediante la siguiente ecuación:

$$f_c = \frac{1}{2\pi RC} \quad (10)$$

Se ha estudiado la respuesta en frecuencia del filtro paso bajo con los siguientes valores de los componentes:

- $R = 22K$
- $C = 1.2nF$

Obteniéndose una frecuencia de:

$$f_c = \frac{1}{2\pi 22 * 2^{11} * 1.2 * 10^{-9}} = 6.029KHz. \quad (11)$$

Para verificar estos resultados se ha optado por realizar una simulación en el programa LTspice tal y como se ve en la figura 34. Se ha hecho un uso de una simulación con análisis frecuencial entre las frecuencias de 40Hz y 1024KHz, obteniendo la siguiente línea de respuesta temporal:

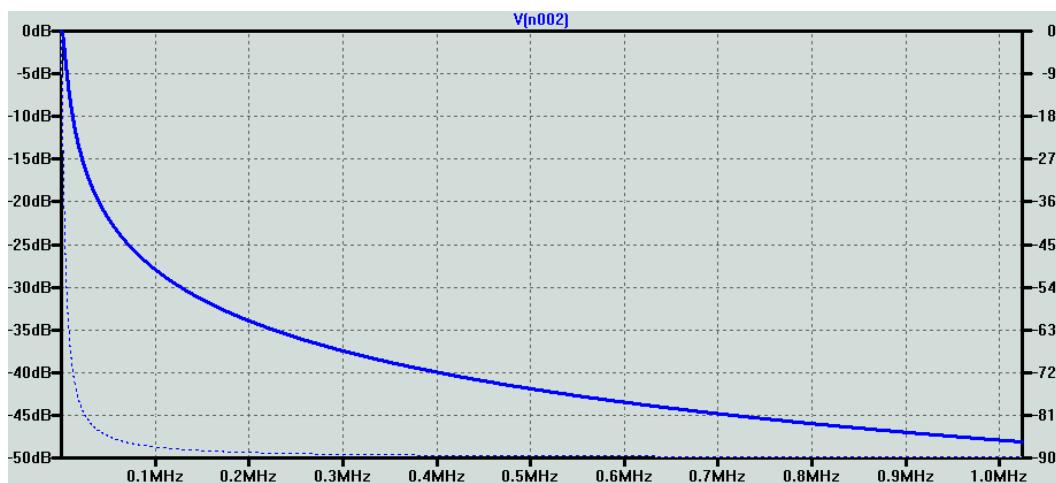


Figura 35: Análisis en frecuencia del filtro RC paso bajo calculado.

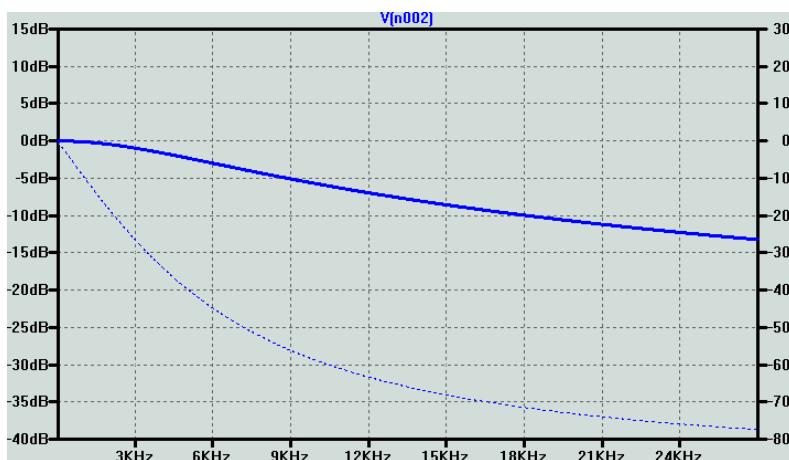


Figura 36: Frecuencia de corte en detalle.

1.14 Verificación de redundancia cíclica CRC

El conversor analógico-digital ADE7912 posee un sistema de CRC (Cyclic Redundancy Check). La verificación por redundancia cíclica es un sistema de detección de errores y viene implementado tanto en los registros de lecturas (V1WV, V2WV y IWV) como para los registros de configuración.

El CRC de interés es el de los registros de lecturas analógicas. Dicho CRC se guardará en el registro ADC_CRC. El algoritmo CRC está basado en el algoritmo CRC-16-CCITT. Los registros son introducidos en el registro LFSR tal y como se muestra gráficamente en la figura 37.

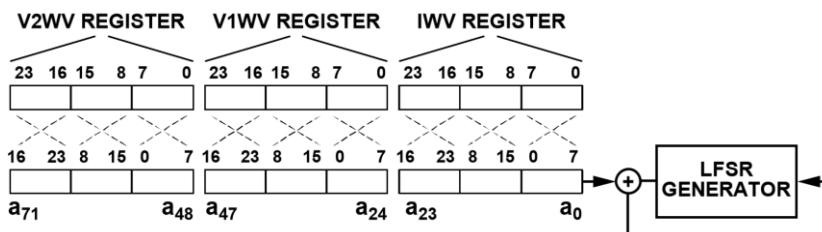


Figura 37: Generación del CRC de las lecturas del ADC.

Teniendo ya este registro a0-a71, empezaremos a calcular los distintos valores del registro ADC_CRC de la siguiente forma:

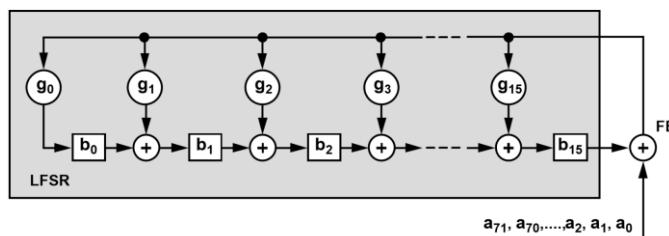


Figura 38: Diagrama del generador del ADC_CRC.

Teniendo que:

- $b_i(0) = 1$, donde $i = 0, 1, 2, \dots, 15$ son los bits iniciales que conforman el CRC, siendo b_{15} el MSB y el b_0 el LSB.
- g_i , donde $i = 0, 1, 2, \dots, 15$ son los coeficientes de generación del polinomio definido por el algoritmo CRC-16-CCITT:

$$G(x) = x^{16} + x^{12} + x^5 + 1 \quad (12)$$

$$g_0 = g_5 = g_{12} = 1 \quad (13)$$

Todos los demás coeficientes de g_i son 0.

$$FB(j) = a_{j-1} \text{ XOR } b_{15}(j-1) \quad (14)$$

$$b_0(j) = FB(j) \text{ AND } g_0 \quad (15)$$

$$b_i(j) = FB(j) \text{ AND } g_i \text{ XOR } b_{i-1}(j-1), i = 1, 2, 3, \dots, 15 \quad (16)$$

Las ecuaciones 14, 15 y 16 tendrán que ser repetidas para cada uno de los valores de j (j=1 hasta j=72). El valor que finalmente será escrito en el ADC_CRC serán los bits b_i(72) para i=0,1,2,...,15.

Dependiendo de la frecuencia escogida en el registro CONFIG mediante los bits 5:4 de ADC_FREQ, el valor del CRC almacenado en el registro serán ‘n’ o ‘n-1’, siendo n el ciclo actual de trabajo. Para un valor de 8Khz (El máximo) el CRC contendrá el valor generado para el mismo ciclo de salida en el que se encuentra. Para otro valor de frecuencia de salida del ADC, el CRC guardado será el generado en el ciclo anterior.

1.15 Puesta en marcha del ADE7912

Para un sistema con múltiples ADE7912 conectados a un microcontrolador y cuya frecuencia de reloj es suministrada gracias a un único cristal, habrá que seguir el siguiente procedimiento recomendado por el fabricante:

1. Suministrar Vdd a todos los dispositivos. Para garantizar un correcto encendido, la alimentación tiene que alcanzar los 2.97V en menos de 23ms desde el punto de los 2.6V. El primer dispositivo ADE7912A (fase A) estará conectado al cristal de 4.096 Mhz mientras que los demás dispositivos no están conectados aún a ninguna frecuencia de reloj.
2. El conversor DC-DC alimenta la parte protegida del circuito integrado. Es entonces cuando el modulador empieza a funcionar. Este proceso toma aproximadamente 100ms siempre y cuando el condensador recomendado para VDDiso haya sido colocado (figura 40). Después de este tiempo, el lado aislado galvánicamente del ADE7912A es totalmente funcional.
3. Para determinar cuando el ADE7912A está listo para aceptar comandos, se puede leer el registro STATUS0 y comprobar si su bit 0 está puesto en 0. Esto suele ocurrir a los 20ms del encendido.
4. Inicializar el registro CONFIG con el bit 0 puesto a 1. Esto habilitará la señal de reloj en el pin CLKOUT para los demás dispositivos.
5. El proceso de encendido de los otros 3 dispositivos ADE7912 se ejecutan tal y como se ha explicado en los pasos 2 y 3.
6. Inicializar correctamente los pines necesarios del registro CONFIG de todos los dispositivos, y conectar el pin DREADY del ADE7912C a una entrada del microcontrolador. Para esto el bit de CLKOUT del ADE7912C debe ser puesto a 0.
7. Inicializar el registro EMI_CTRL con los valores deseados.
8. Escribir en SYNC_SNAP un 0x01 de forma paralela en todos los dispositivos para empezar la lectura del ADC de todos los dispositivos en el mismo instante.
9. Bloquear la escritura de los registros de configuración.
10. Por último, cada unos pocos minutos comprobar la sincronización de las lecturas, tal y como se explicó en el apartado de sincronización de los ADE7912.

Todos estos pasos se pueden ver de forma clara en un diagrama V-T en la figura 39.

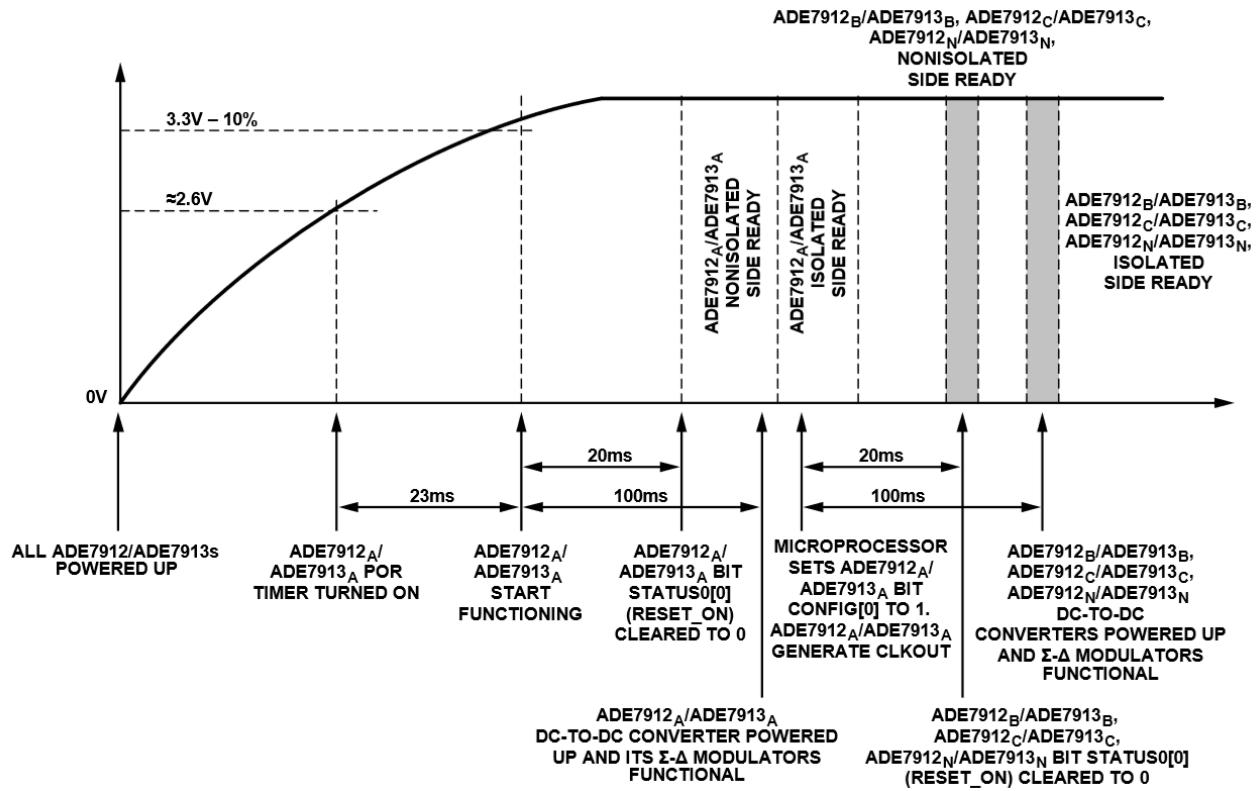


Figura 39: Diagrama del encendido de todos los ADE7912.

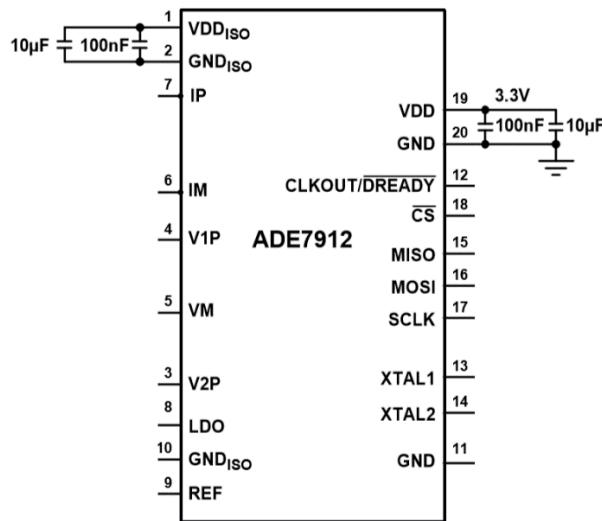


Figura 40: Alimentación del circuito integrado ADE7912.

2. Actuadores

2.1 Relé de estado sólido

Un relé de estado sólido es un dispositivo electrónico capaz de interrumpir el paso de electricidad de una red de corriente alterna o continua. Este dispositivo electrónico es controlado por una señal de control aislada del sistema central y que en el caso de este proyecto, será una salida digital de 5V del sistema empotrado.

El relé de estado sólido no tiene partes mecánicas móviles (a diferencia de un relé normal), lo que conlleva un menor desgaste con el paso del tiempo. El relé de estado sólido (O SSR en inglés) utiliza semiconductores de potencia como tiristores y transistores, esto le permite poder cortar corrientes de muy alto amperaje y con una velocidad muy alta en comparación con los relés de partes mecánicas móviles.

El SSR *G3PH-5150B DC5-24* con función de paso por zero que se ha elegido es capaz de conmutar tensiones de hasta 480V y una intensidad de hasta 150A, podremos colocar tantos SSR como se requiera para poder controlar todos los dispositivos que se quieran controlar en el sistema. La única limitación viene dada por el número de salidas digitales del sistema empotrado, pero realizando un sistema de multiplexión, se podrán controlar tantos dispositivos como se necesiten.

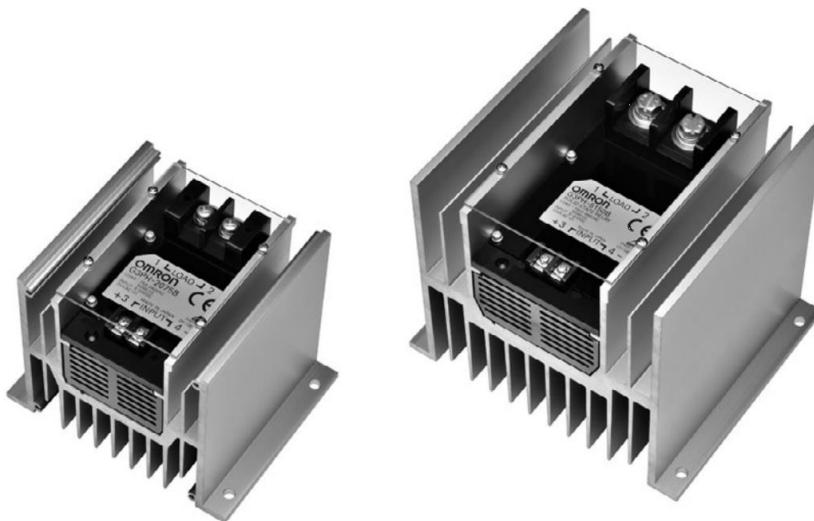


Figura 41: Relé de estado sólido G3PH-5150B DC5-24.

3. Sistema de adquisición, trata y representación de datos

3.1 Sistema Empotrado

Después de un estudio exhaustivo de los microcontroladores disponibles, se ha decidido utilizar el PIC18F8520, correspondiente a las MCU de gama media de 8 bits de la familia PIC del fabricante Microchip.

La decisión del uso de esta MCU, viene dada por las características que lo hacen ideal para el uso que se va a dar en este trabajo: Una estructura optimizada para ser utilizada con la programación en C, una velocidad de reloj de hasta 40MHz, diversos contadores de hasta 16bits, un módulo MSSP que permite una comunicación SPI como esclavo y como maestro, multiplicador por hardware, multitud de pines de salida digital para controlar tantos SSR como se necesiten, etc.



Figura 42: PIC18F8520.

3.2 Dualidad Esclavo-Maestro

Para el funcionamiento correcto del sistema, el PIC18F8520 deberá ser alternado entre modos SPI: maestro y esclavo. El microcontrolador funcionará en modo maestro cuando esté recolectando datos del ADC. El mismo pasará a funcionar como esclavo cuando tenga que transmitir estos mismos datos a la raspberry pi. Para una correcta sincronización entre la raspberry pi y el PIC se va a hacer uso de un puerto digital del PIC y un puerto digital del GPIO de la raspberry para generar un medio de comunicación básico.

No sólo será necesario una sincronización entre ambos dispositivos, sino que habrá que aislar los circuitos del ADC y de la raspberry para que no existan interferencias entre ellos. Para ello se ha seleccionado el CI TC4066BP.

3.3 TC4066BP

El CI del TC4066BP contiene cuatro circuitos que actúan como interruptores independientes. Cuando el bit de control está en alto, la entrada y salida correspondiente se encontrarán en conexión, asegurando una resistencia entre patillas menor a $500\ \Omega$. Cuando la patilla de control se encuentre en nivel lógico bajo, la resistencia entre las patillas de entrada/salida será mayor de $1M\Omega$ ohm. En la figura 43 se puede apreciar un modelo simplificado del funcionamiento del mismo:

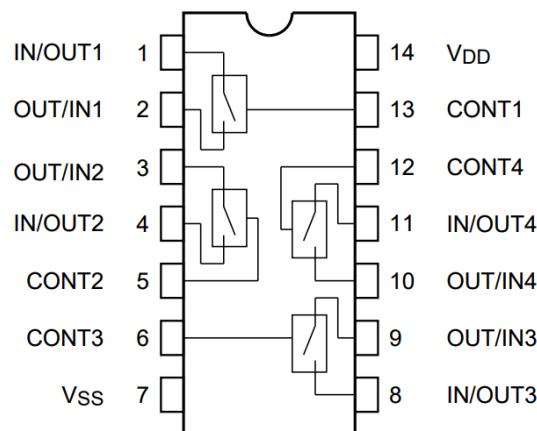


Figura 43: TCP4066BP

Se va a hacer uso de 3 de los *switches* del sistema, ya que se van a separar las siguientes conexiones: *MISO*, *MOSI* y *CLK*. Estas conexiones vienen debidamente explicadas y representadas en esquema de montaje. En el diseño del sistema se van a usar dos TC4066BP, para separar las señales del SPI del ADE7912 y de la raspberry pi.

3.4 DM7404

El CI DM7404 funciona como un operacional NOT cuya finalidad es la negación del pin de control. Esta negación será utilizada para intercambiar la activación entre los dos TC4066BP y poder intercambiar entre los dispositivos SPI.

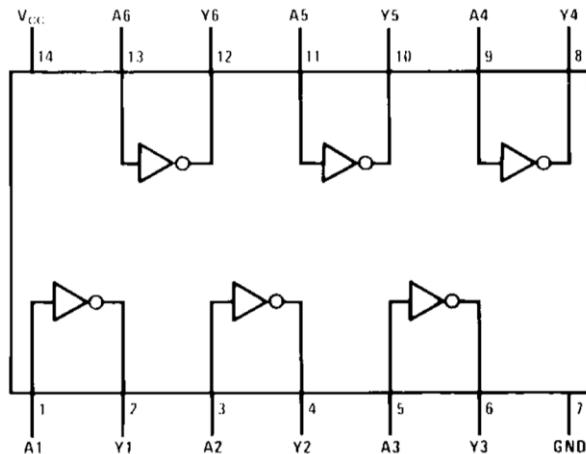


Figura 44: DM7404

3.5 Compatibilidad entre protocolos SPI

El protocolo SPI utilizado por la Raspberry Pi 3 y el ADC ADE7912 utilizan un nivel lógico superior de reloj de 3.3V, mientras que la lógica del reloj usada por el microcontrolador es de 5V. Por ello, se ha escogido el diseño de un nivelador bipolar de señal para poder unir protocolos y habilitar la comunicación entre dispositivos. Este diseño es muy conocido y utilizado, por lo que se puede comprar ya montado o elegir montarlo de forma manual:

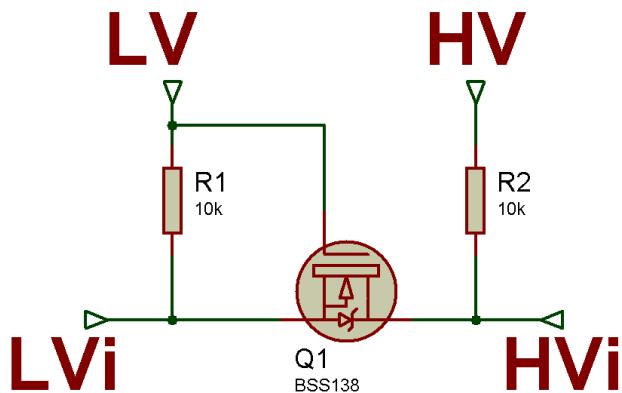


Figura 45: Conversor lógico bipolar

En el circuito se tienen cuatro posibles escenarios:

- Transmitiendo el lado de 3.3V un ‘1’ digital, tenemos en LV 3.3V y en LVinput 3.3V, por lo que la caída de tensión en la puerta del mosfet es 0V y el circuito no conduce. Al no conducir el mosfet, en la entrada del lado de 5V tendremos 5V debido a R2.
- Transmitiendo el lado de 3.3V un ‘0’ digital, tenemos en LV 3.3V y en LVinput 0V, por lo que el mosfet entra en conducción. Estando en conducción, el voltaje del drenador del mosfet viene dado por la tensión del sumidero, en este caso, 0V.
- Transmitiendo el lado de 5V un ‘1’ digital, tenemos en HV 5V y en HVinput 5V. Sin embargo, en el lado de baja tensión el pull up activa 3.3V en la entrada correspondiente. El mosfet no conduce en este caso.
- Transmitiendo el lado de 5V un ‘0’ digital, tenemos en HV 5V y en HVinput 0V. Esto genera que el diodo del mosfet se polarice de forma directa, marcando un voltaje de 0.7V en el lado de baja tensión.

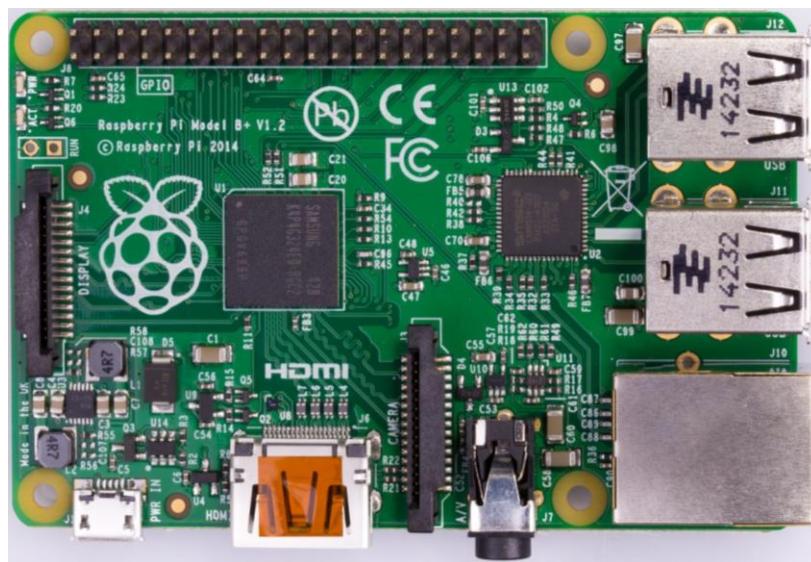
3.6 Servidor

Una vez que se tiene todo el trabajo de captación de datos, es necesario seleccionar una plataforma que albergue el papel de servidor, al igual que sea capaz de procesar los datos, puesto que los cálculos más pesados se van a realizar en el servidor. Un requisito de esta plataforma debe ser la incorporación de un módulo wifi para poder tener un sistema que no necesite una conexión por cable a internet. Para conformar un bloque único de un dispositivo IIOT, también es necesario que la placa sea de tamaño reducido para aumentar su movilidad.

Finalmente, como no se requiere de un servidor que aloje datos muy pesados ni que sea capaz de recibir miles de peticiones en un breve periodo de tiempo, se ha optado por un ordenador de placa reducida (*Single Board Computer*), específicamente la Raspberry Pi 3 modelo B, cuyas especificaciones se detallan a continuación.

3.7 Computador de placa reducida

Es un computador de placa reducida de bajo coste diseñado en Reino Unido por la fundación Raspberry Pi con el principal propósito de incentivar la enseñanza de ciencias de la informática.



En cuanto al apartado de las comunicaciones, la raspberry pi 3 B tiene integrado un módulo de red inalámbrico Wireless LAN, un módulo de Bluetooth de bajo consumo 4.1 y cuatro conectores USB 2.0.

Tratando las diferentes interfaces hardware que tiene este modelo de Raspberry Pi, se cuenta con un puerto GPIO con 40 pines entre los que se encuentran una UART, un bus I2C, un bus SPI con dos *Chip Selects*, salidas de 3V3, 5V, tierra y diversos pines que pueden ser configurados como entradas o salidas. También cuenta con un puerto CSI-2 para la implementación de una cámara y un puerto específico *DSI Display* para una pantalla LCD táctil

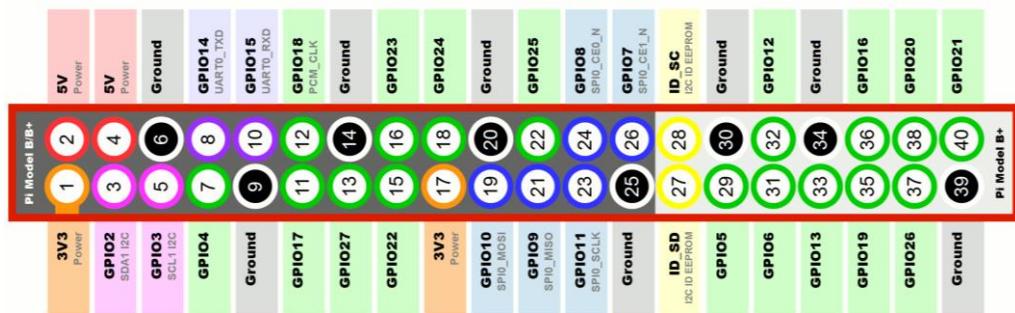


Figura 47: GPIO de la Raspberry Pi 3 model B.

En cuanto a la alimentación, la placa se alimenta a través de un puerto micro USB, con un consumo de corriente de hasta 2.5A a una tensión de alimentación de 5V.

Finalmente se puede observar en la siguiente figura un resumen completo del hardware de la Raspberry Pi 3 B:

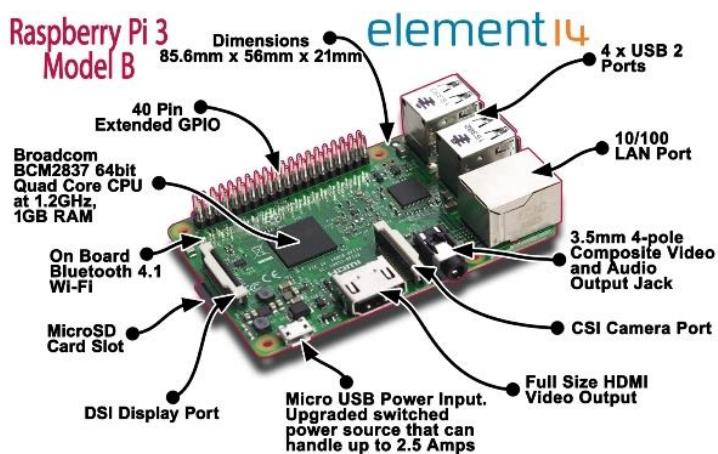


Figura 48: Resumen completo del hardware de la Raspberry Pi 3 model B.

3.8 Sistema Operativo

Como ya se ha comentado en el anterior apartado, el software que se va a utilizar en la raspberry pi será la distribución Raspbian. Raspbian es la distribución más utilizada y probada en todo el ecosistema que rodea a esta SBC, por esto, y por su filosofía de código libre, se ha escogido respecto a otras como pueden ser Windows 10 o Pidora, entre otras.

Para la instalación de este sistema operativo, es necesario la descarga del mismo desde la página web oficial del proyecto Raspberry Pi (<http://raspberrypi.org>) y la grabación la imagen de disco en la tarjeta micro SD siguiendo las instrucciones que el fabricante proporciona.

En el primer encendido del sistema, el sistema operativo se instalará y una vez transcurrido el tiempo de instalación, se podrá acceder al SO completo a través del puerto HDMI y mediante el uso de teclado y ratón, como si de un ordenador cualquiera se tratase. Sin embargo, para futuras conexiones, se podrán activar los servicios de SSH (Secure SHell) o intérprete de órdenes seguro y el servicio de VNC (Virtual Network Computing), los cuales permitirán acceder mediante una conexión a internet a la raspberry de forma remota.

3.9 PHP7.0 + Apache2.0

Para la construcción del sitio web accesible mediante internet se va a utilizar el entorno de apache2 más PHP7.0, descartándose el uso de bases de datos SQL debido a su lentitud de procesado y acceso.

Apache2 es un servidor web HTTP de código abierto para plataformas UNIX (como Raspbian). Entre sus ventajas están su modularidad (se pueden instalar módulos que aumenten las prestaciones de nuestro servidor fácilmente), es de código abierto y es muy popular, pudiéndose recibir soporte de manera sencilla y rápida.

<https://www.apache.org/>

PHP (Hypertext Preprocessor) es un lenguaje de código abierto muy popular y especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML. Su potencia de procesado permite elaborar sitios webs complejos y dinámicos, siendo un lenguaje de programación ideal para el desarrollo del proyecto.

<http://php.net/>

3.10 Biblioteca WiringPi

Como la programación del sistema de adquisición de datos va a ser programado en C, se va a utilizar una biblioteca que permita el acceso del SPI de la raspberry pi y el uso de los pines de entrada/salida digitales que la raspberry pi posee en el GPIO. Por esto, la elección ha sido WiringPi, biblioteca la cual viene pre-instalada en las nuevas versiones de Raspbian.

Su está protegido bajo la licencia LGPLv3 y fue diseñada por Gordon Henderson. La biblioteca cuenta con las siguientes funciones que serán usadas en el proyecto:

```
#include <wiringPISPI.h>
#include <wiringPi.h>
```

Primeramente se tendrá que incluir la librería WiringPiSPI y WiringPi al ejecutable.

```
int wiringPISPISETUP(int channel, int speed);
```

La ejecución de esta función iniciará el protocolo para la comunicación del SPI. En esta función se seleccionan dos parámetros: El canal a elegir (entre el 0 y el 1) y la velocidad del reloj para la transmisión (entre 500Khz hasta 32MHz).

```
int wiringPISPIDataRW(int channel, unsigned char *data, int len);
```

Esta función manda un array de datos (unsigned char) de longitud ‘len’ por el canal seleccionado (0 o 1). Esto es importante ya que el GPIO de la raspberry pi 3 dispone de dos Chip Select distintos, uno para cada canal.

```
void pinMode (int pin, int mode);
void digitalWrite(int pin, int value);
int digitalRead(int pin);
```

Esto permite seleccionar el modo en el cual el pin seleccionado (variable pin): entrada o salida. Una vez que ha sido seleccionado en alguno de los dos modos, podremos seleccionar una salida (0 -nivel bajo- o 1 -nivel alto-) o si se trata de una entrada, podremos leer su estado (0 o 1).

La biblioteca también permite acceder a funciones de temporización, I2C, comunicación serie, etc que no se han explicado debido a que no se van a utilizar.

3.11 Pantalla LCD de 7 pulgadas táctil

Para aumentar la funcionalidad y versatilidad del sistema se ha decidido incorporar una pantalla LCD de 7 pulgadas táctil para poder representar el estado del sistema y como sistema de interacción alternativo a la aplicación móvil y al servidor web.

La pantalla LCD tiene una resolución de 800x480 píxeles con una matriz táctil dc tecnología capacitiva. Esta LCD se alimenta directamente desde el GPIO de la raspberry pi 3, usando sólo dos pines de los disponibles para alimentación y el puerto especial para el LCD DSI.

Para aumentar la funcionalidad de este accesorio, se ha implementado una GUI táctil mediante la programación python y el uso de la libreria Tkinter.



Figura 49: LCD 7'' táctil para la Raspberry Pi 3.

3.12 Python + Tkinter

Python es un lenguaje de programación interpretado de alto nivel cuya programación se basa en una estructura limpia y legible. Este lenguaje fue creado en 1991 por Guido Van Rossum. El uso de este lenguaje de programación obliga al programador a usar un código limpio y legible.

Python es multiplataforma y multiparadigma, pudiéndolo usar como lenguaje orientado a objetos, como lenguaje imperativo, funcional y programación por procedimientos. Debido a su potencia, facil uso e iniciación y gran versatilidad es un lenguaje muy extendido para soluciones de proyectos profesionales.

La librería Tkinter (tkinter a partir de python 3.0) es el estándar de la herramienta Tk GUI para Python. Tanto Tk como TKinter están disponibles para la mayoría de plataformas Unix, al igual que para la plataforma Windows.

Dentro de la librería tkinter se pueden encontrar diversos módulos, los cuales serán de gran utilidad para la creación de una interfaz gráfica que contenga cajas de texto desplazable, imágenes, botones, etcétera.

<https://www.python.org/>

<https://wiki.python.org/moin/TkInter>



Figura 50: logotipo de la Python Software Foundation.

3.13 Aplicación para móviles Android

Java es un lenguaje de programación de propósito general multiplataforma. En el proyecto se ha utilizado Java como lenguaje base para la programación de la aplicación móvil.

El sistema operativo que se ha elegido ha sido Android. Android es un sistema operativo basado en el núcleo Linux, originalmente creado por Android Inc, empresa que fue comprada más tarde por Google. Se ha elegido este sistema debido a su sencillez en uso y programación. El ecosistema de Android brinda al desarrollador una librería inmensa de proyectos y funciones ya programadas y de libre uso, a parte de multitud de tutoriales y apoyo por parte de Google.

El software utilizado para su programación es el llamado Android Studio, que ofrece una alta gama de librerías y códigos de ejemplo que han permitido avanzar de una forma más rápida en su desarrollo.

El mayor problema de Android es a su vez su principal virtud: En el sistema Android coexisten tablets y móviles de todo tipo y tamaño, por lo que la programación de una aplicación válida para todos estos dispositivos es una tarea complicada. Por ello se ha seleccionado como modelo estándar una pantalla de 5,5 pulgadas. Más específicamente, el móvil modelo donde ha sido testeada la aplicación ha sido un Xiaomi Mi A1.



Figura 51: logotipo Android Studio.

4. Diseño Hardware

El diseño hardware se ha repartido en tres secciones diferenciadas: El sistema de alimentación ininterrumpida, cuya finalidad es la generación de tensión de alimentación a 3.3V, 5V y carga de la batería. La placa central, que alojará todos los circuitos integrados necesarios y que contará con las salidas necesarias para el GPIO de la raspberry pi. Por último, la placa de acondicionamiento de señales, donde se alojarán las resistencias shunt y los bornes de conexión para la red eléctrica.

4.1 Sistema de alimentación ininterrumpida

El sistema necesita una fuente de alimentación primaria con la capacidad de proveer alimentación a pesar de pequeños cortes y huecos de tensión en la alimentación primaria. Por ello, se ha diseñado un sistema de alimentación específico usando una batería de 12V la cual será capaz de mantener una tensión estable en todos los dispositivos de nuestro sistema.

El sistema de alimentación se divide en 3 circuitos electrónicos bien diferenciados: alimentación de 12V a 5V, alimentación de 12V a 3.3V y circuito de carga para la batería de 12V.

El circuito completo lo podemos ver en la siguiente página en A3:

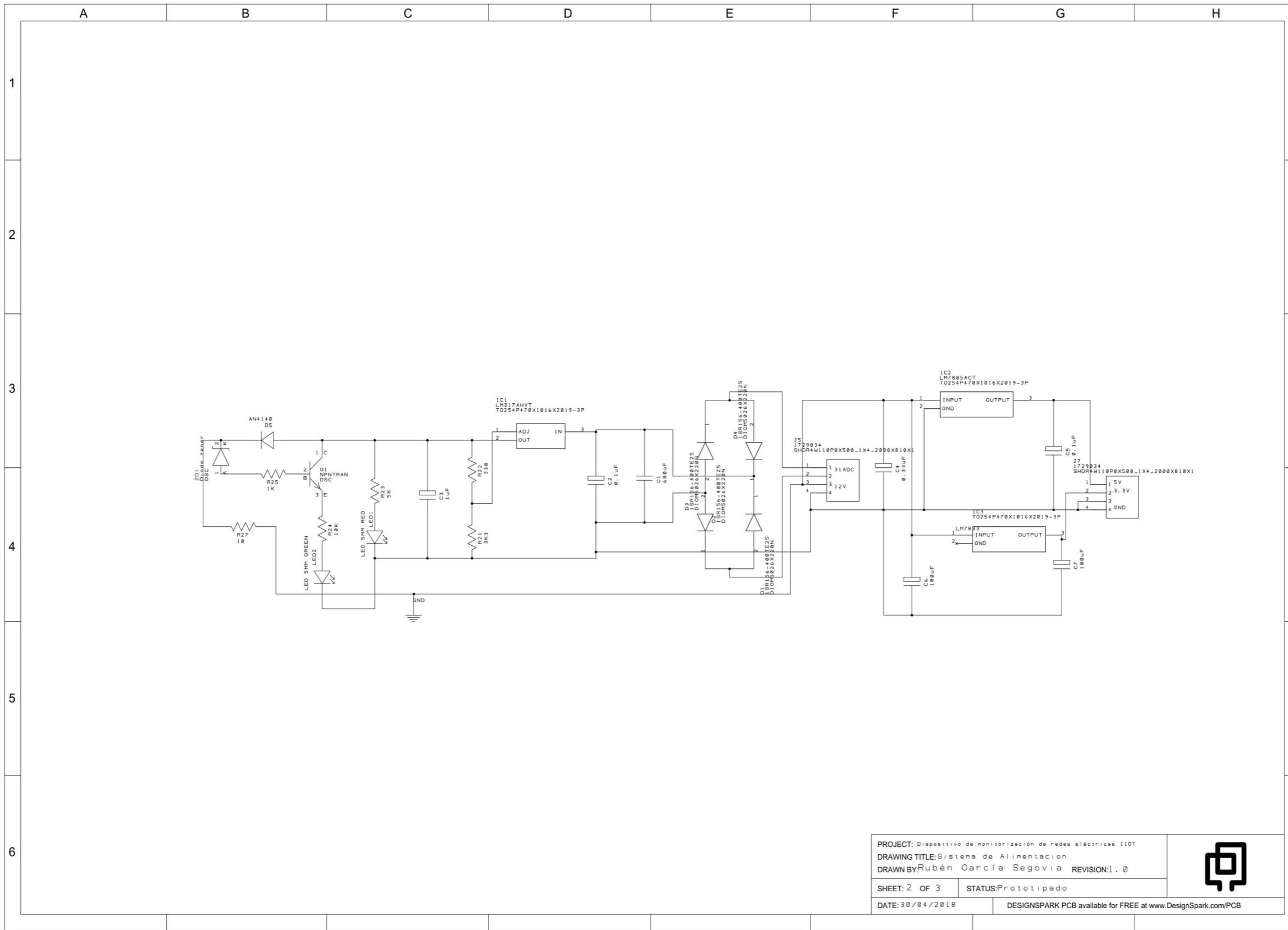


Figura 52: Sistema de alimentación ininterrumpida

La primera parte del circuito de carga para la batería se puede ver en la figura 53.

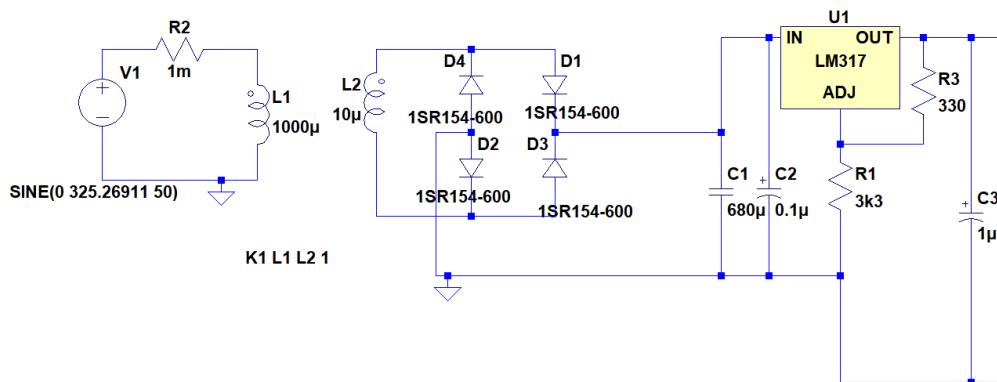


Figura 53: Fuente de alimentación 230AC – 14.5DC

Primero se ha atenuado la onda senoidal primaria a través de un transformador 230AC-32AC, para rectificarla con un puente de diodos rectificadores, obteniendo como resultado una tensión continua regulada de 31V aproximadamente. Después se ha fijado la tensión (y se han eliminado los rizados) con un condensador electrolítico de 680uF. El resultado a la entrada del LM317 se puede apreciar en la figura 54.

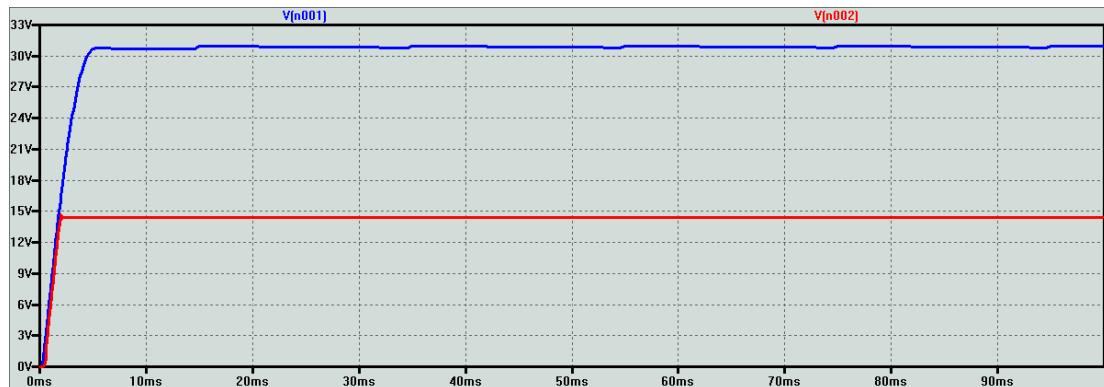


Figura 54: Señal rectificada (Vn001) y señal del regulador (Vn002)

Una vez que tenemos 31V en el pin de entrada del LM317, pasamos al cálculo de las resistencias tal y como viene descrito en el datasheet del propio circuito integrado (incluido en el anexo de hojas de fabricante). Estas resistencias tendrán que ser calculadas debido a que el LM317 es un circuito integrado de voltaje de salida variable. Dependiendo de la configuración y valores de las mismas, se obtendrá el resultado deseado: 14.5V.

$$V_{out} = 1.25V \left(1 + \frac{R_2}{R_1} \right) + I_{adj} * R_2 \quad (17)$$

Para una I_{adj} común menor a 100 microamperios, se podrá tomar este término como nulo. Fijando una R_1 como 330Ω (El fabricante recomienda elegir un valor menor a $1K\Omega$)

$$14.5 = 1.25 \left(1 + \frac{R_2}{330} \right); R_2 = 3498\Omega; R_2 \simeq 3K3\Omega \quad (18)$$

Teniendo la tensión de carga de la batería obtenida, se ha diseñado un circuito analógico para regular el estado de carga de la batería:

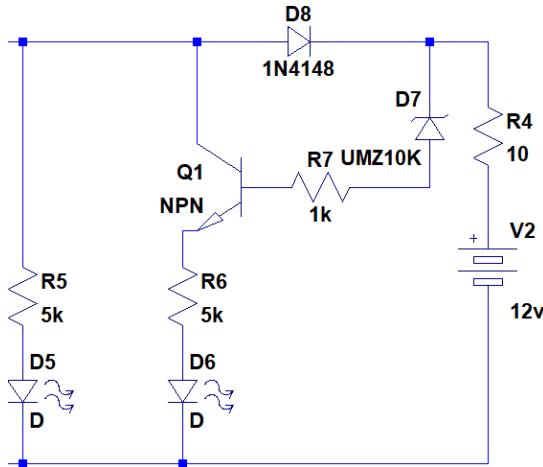


Figura 55: Circuito controlador de la batería.

Se pueden plantear tres casos distintos: batería cargada, batería descargada y corte en la alimentación principal (con la batería cargada o no).

Con un corte en la alimentación principal, el diodo D8 evitirá que la batería se descargue a través de la fuente de alimentación diseñada, cediendo ésta toda su potencia a la alimentación de nuestro sistema.

Al tener la batería cargada y el diodo zener D7 con una tensión de ruptura de 11V. Cuando sea superado el umbral de ruptura (Cuando la tensión de la batería sea mayor a 13.5V), se activará el transistor NPN Q1 y se encenderá el led D6, que será el indicador de batería cargada. Esto desviarán todo el flujo de corriente por el diodo, dejando a la batería sin cargar hasta que no baje de esta tensión.

Cuando la tensión en la batería sea menor que el umbral de ruptura del zener D7, todo el flujo de corriente cargará la batería.

Como se puede observar, el led D5 siempre estará activo, ya que es el piloto que avisará visualmente si hay o no alimentación principal. La resistencia R4 de 10Ω se encarga de limitar la corriente de carga de la batería.

Para regular la tensión de salida de 12V y ajustarla a los requerimientos de nuestros CI y raspberry pi, se han utilizado los reguladores de tensión comerciales LM7805 y LM7833, los cuales permiten hasta una intensidad de salida de 1.5A, suficiente para nuestro sistema.

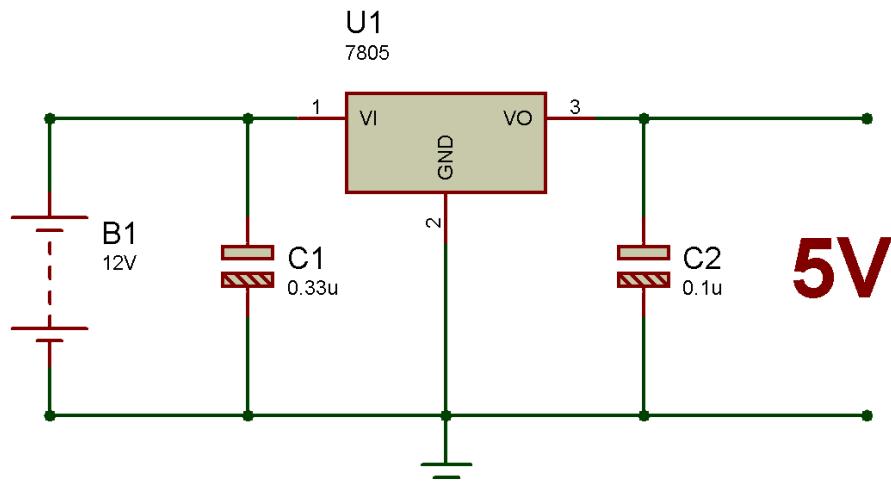


Figura 56: Circuito regulador a 5V.

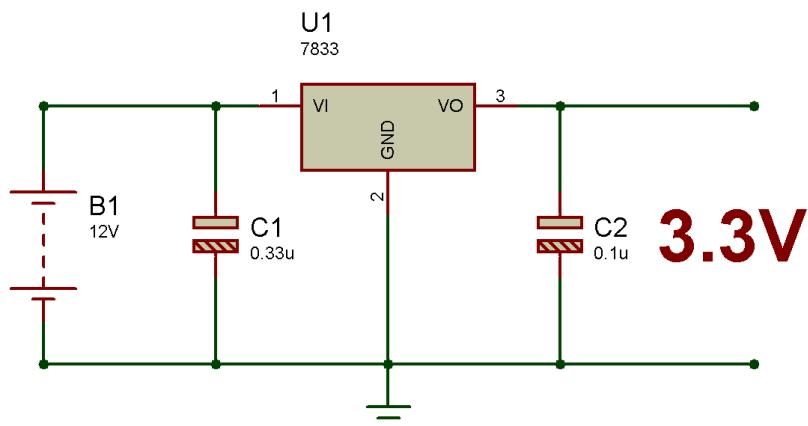


Figura 57: Circuito regulador a 3.3V.

No se han realizado cálculos para estos dos circuitos, ya que se ha utilizado el circuito recomendado por el fabricante. Para más información, el datasheet se encuentra en el anexo de hojas de fabricantes.

Se ha utilizado un conector de 4 patillas en las que se encuentran disponibles los 5V, 3.3V y dos GND. Se ha utilizado otro conector de 4 patillas para conectar los bornes de la batería de 12V y para conectar el segundo bobinado de un transformador 230-31ADC.

Los archivos gerber necesarios para la fabricación de la pcb se encuentran en el CD adjuntado con el TFG y a continuación convertidos en imagen:

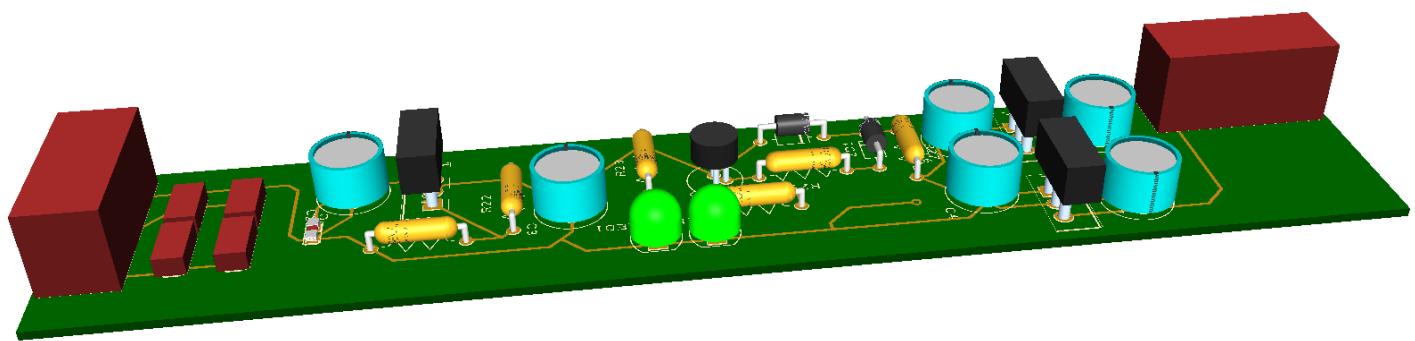


Figura 58: PCB Sistema de alimentación ininterrumpida en 3D

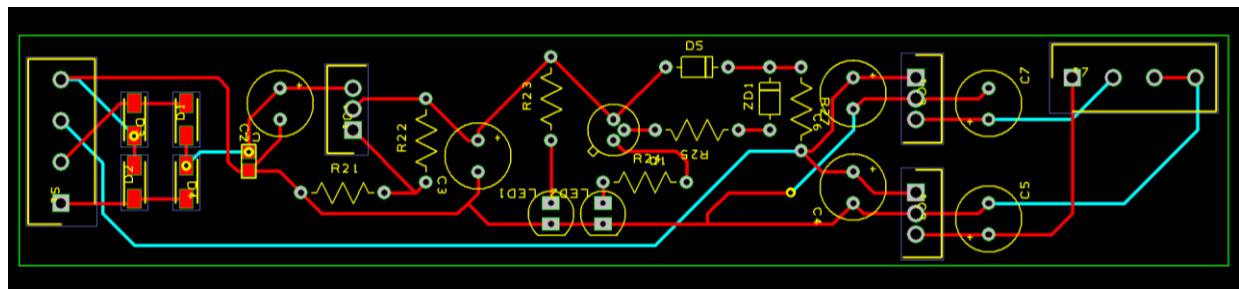


Figura 59: PCB Sistema de alimentación ininterrumpida en negativo.

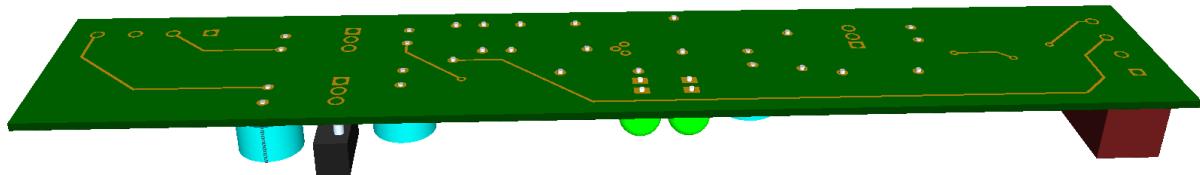


Figura 60: PCB Sistema de alimentación ininterrumpida en negativo en 3D.

Gerbers más significativos exportados como imagen:

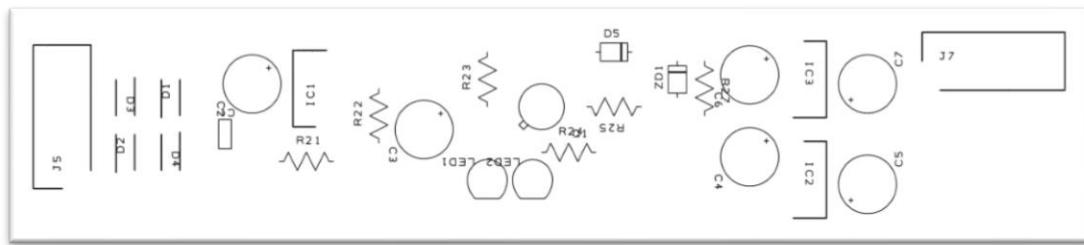


Figura 61: Serigrafía capa superior.

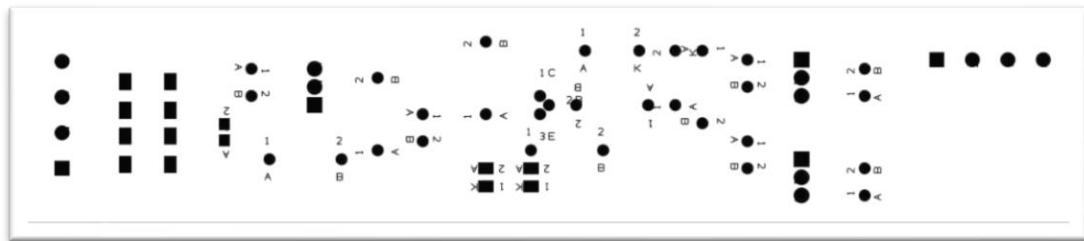


Figura 62: Pasta de soldadura capa superior.

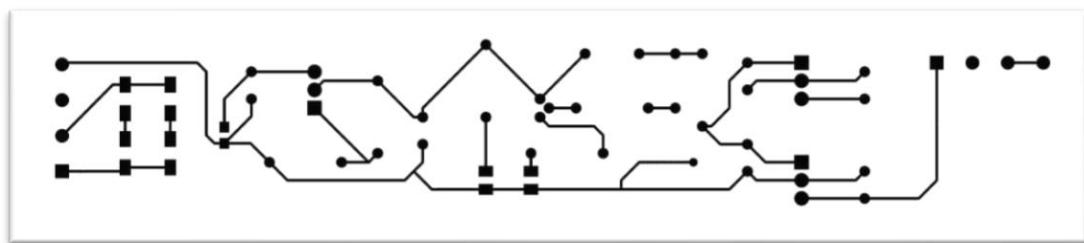


Figura 63: cobre capa superior.

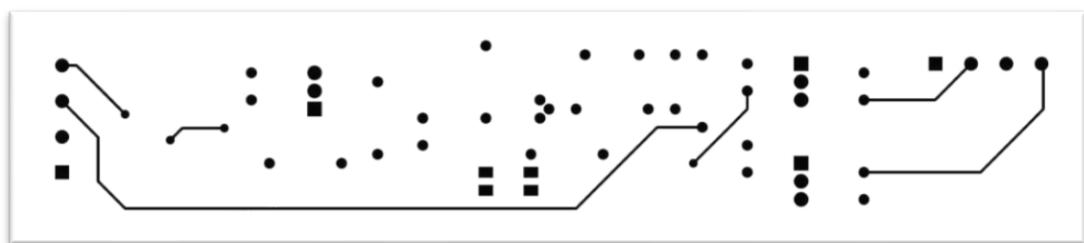


Figura 64: Cobre capa inferior.

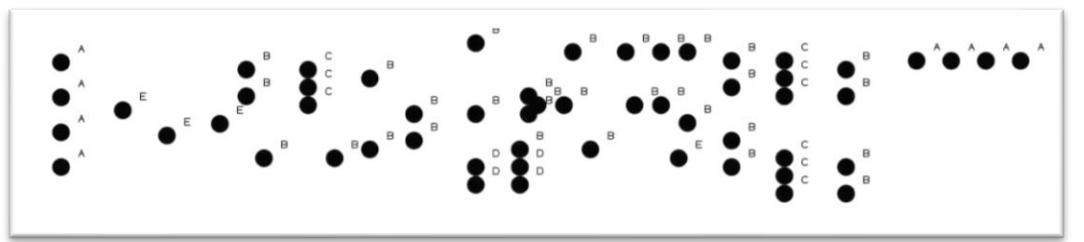


Figura 65: Dibujo de los taladros.

4.2 Placa principal

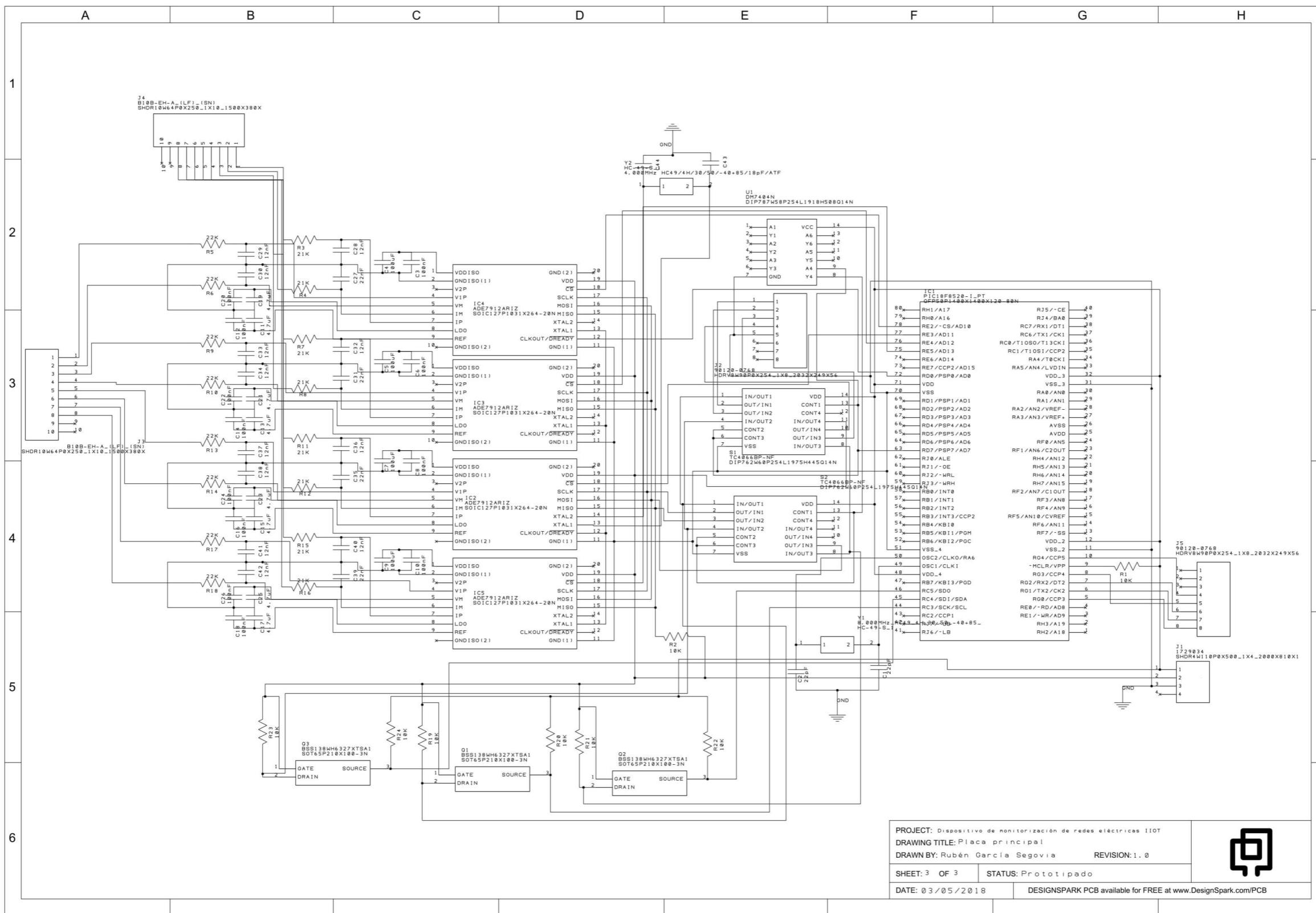


Figura 66: Esquema placa principal

En la placa principal se alojará el sistema empotrado: el PIC 18F8520 junto con los tres conversores analógico digitales, junto con las conexiones necesarias para conectar esta placa con la placa de alimentación, de adquisición de datos y con la raspberry pi.

El primer aspecto a inspeccionar es la configuración de los ADE7912, los cuales incluyen alimentación a 3.3V, los filtros paso bajo diseñados anteriormente y la debida configuración de pines y condensadores de desacople:

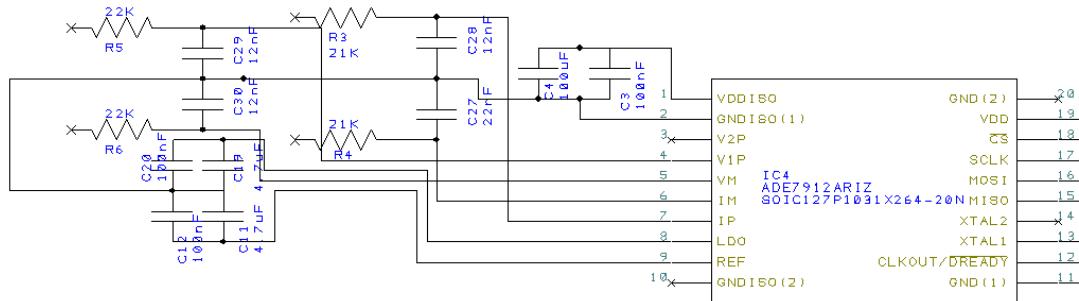


Figura 67: Conexionado del ADE7912

A la salida de las resistencias R5, R3, R6 y R4 se obtienen las conexiones IP, IM, VP y VM provenientes de la placa de adquisición de datos.

Para el conexionado de alimentación y entre los distintos ADC, se conectan todos los MOSI, MISO y CLK entre sí. A la línea de MISO se le pone un pull-up debido a que es un pin con drenador abierto. Todas estas líneas se conectan al chip TC4066BP, y anteriormente al regulador de nivel lógico ya explicado. También se cuenta con un cristal de cuarzo de 4.096Mhz en uno de los 4 ADE7912.

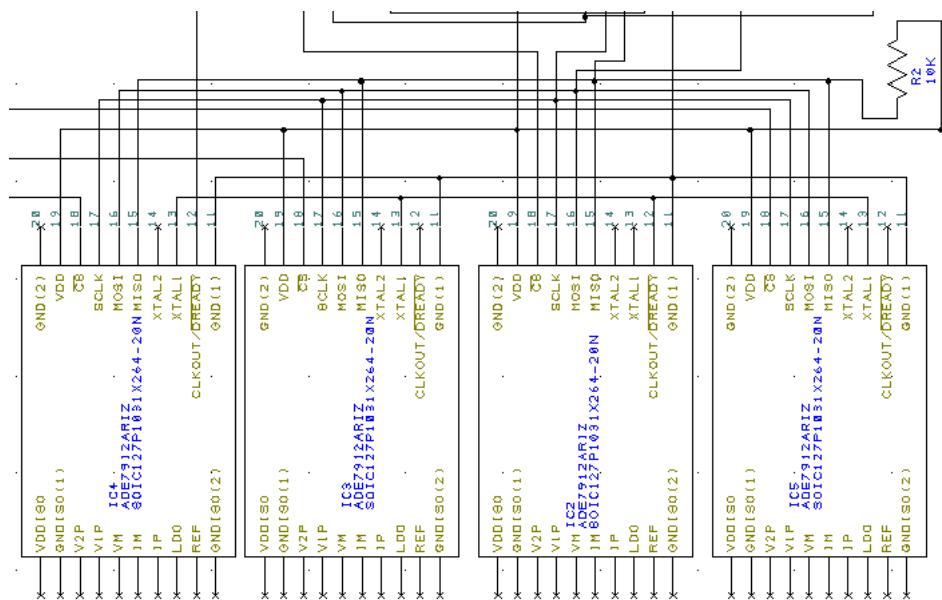


Figura 68: Conexionado entre ADE7912

El pin SCK, MISO, MOSI y PINENABLE están conectados a través de los dos dispositivos TC4066BP, los cuales son controlados por el pin de control RD7 y su negación realizada mediante el chip DM7404. Este multiplexado habilita la conexión SPI en los pines de los diferentes ADE7912, o los habilita en los pines exteriores accesibles para la Raspberry Pi.

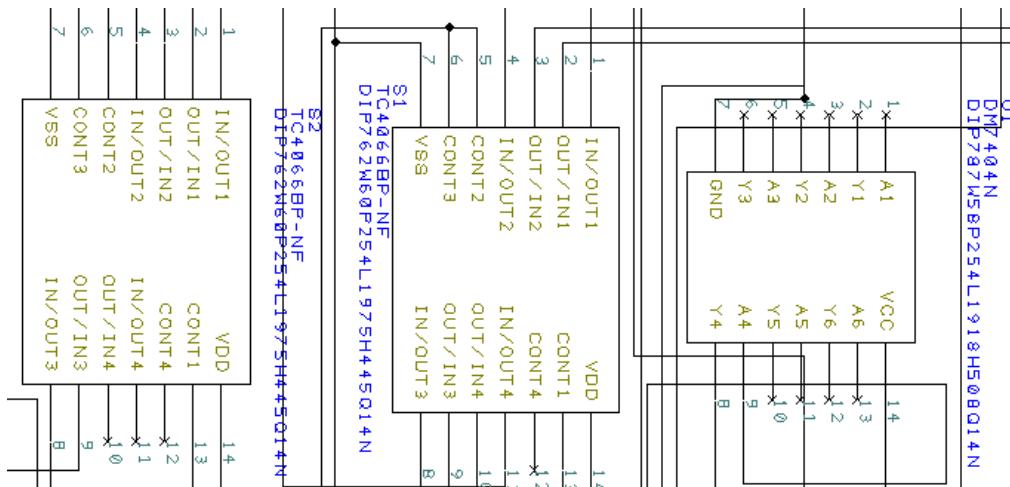


Figura 69: Multiplexado del SPI en el PIC.

Antes del multiplexado, con la ayuda del nivelador bidireccional 5V-3.3V acondicionamos las señales del SPI para proteger a los circuitos integrados de la tensión del microcontrolador.

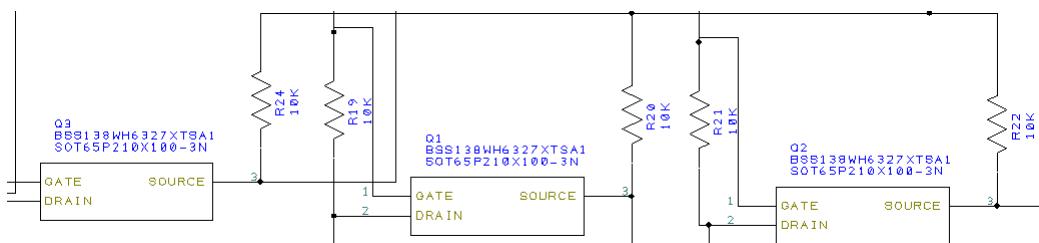


Figura 70: nivelado de señales lógicas.

Se nivelan las señales de CLK, MISO y MOSI. La señal de control no es necesaria ser nivelada ya que con el uso del TC4066BP, se ha cogido la señal de 3.3V de alimentación y la usamos como RD7 en el pin de conexión de la raspberry pi.

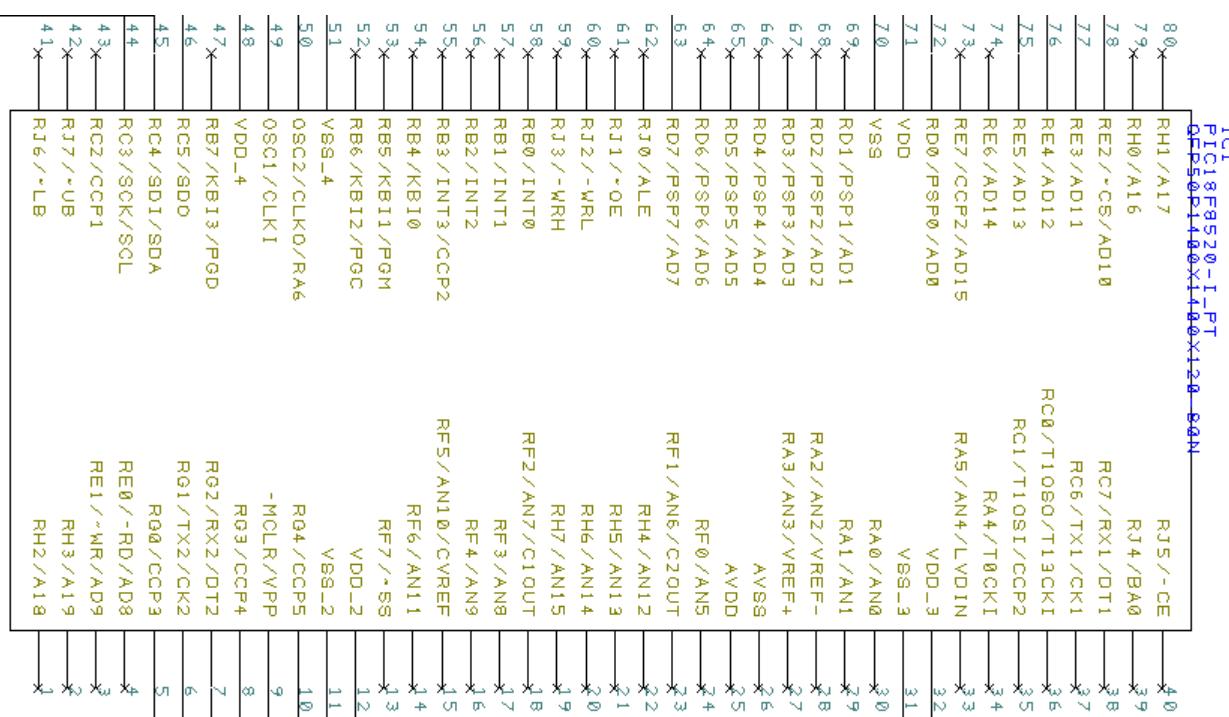


Figura 71: PIC 18F8520.

A continuación se detalla el uso de cada uno de los pines utilizados en la implementación del PIC18F8520:

- Vdd: 5V.
- Vss: GND.
- RG0-RG4: Accesibles para el exterior (Actuadores).
- MCLR/VPP: Pull up de 10k.
- RC3: señal de reloj SPI.
- RC4: señal MISO del SPI.
- RC5: señal MOSI del SPI.
- OSC2-OSC1: circuito de reloj del sistema (8MHz y dos condensadores cerámicos de 22pF).
- RD7: pin RASPIENABLE, realiza el intercambio entre las señales SPI y el estado esclavo-maestro del PIC.
- RD0: pin de entrada del ADE7912, avisa cuando se ha recogido un dato.
- RE2-5: Chip selects de los 4 respectivos ADE7912.

Además de todas las conexiones explicadas anteriormente, la placa principal posee pines accesibles en el exterior para conectar con la raspberry pi, con la fuente de alimentación, con la placa de adquisición de datos, y con cuatro posibles actuadores distintos.

Los archivos gerber para la fabricación de la placa PCB se encuentran en el CD del trabajo. A continuación las imágenes:

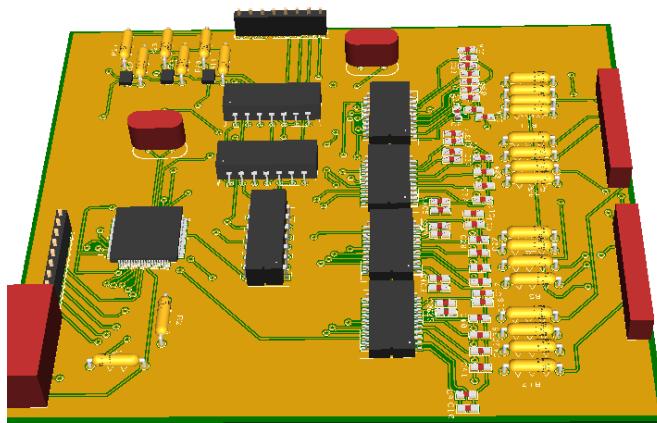


Figura 72: PCB placa principal en 3D.

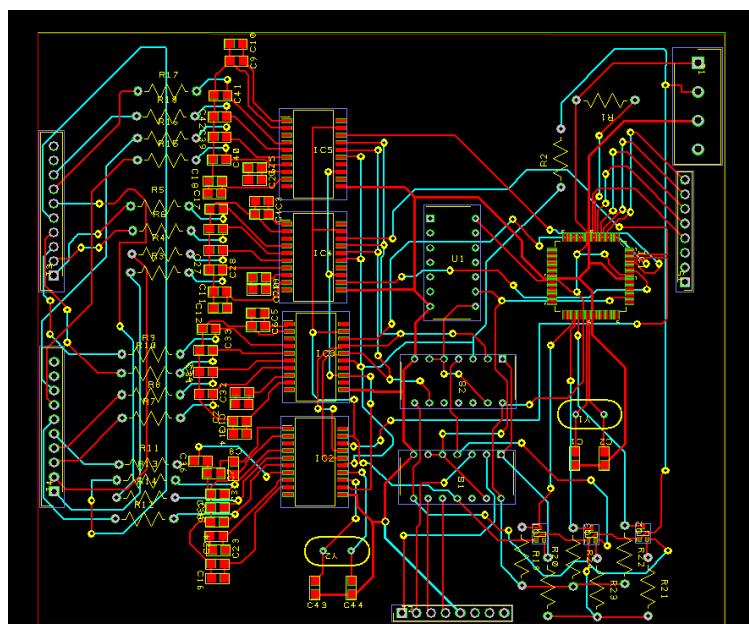


Figura 73: PCB placa principal en negativo.

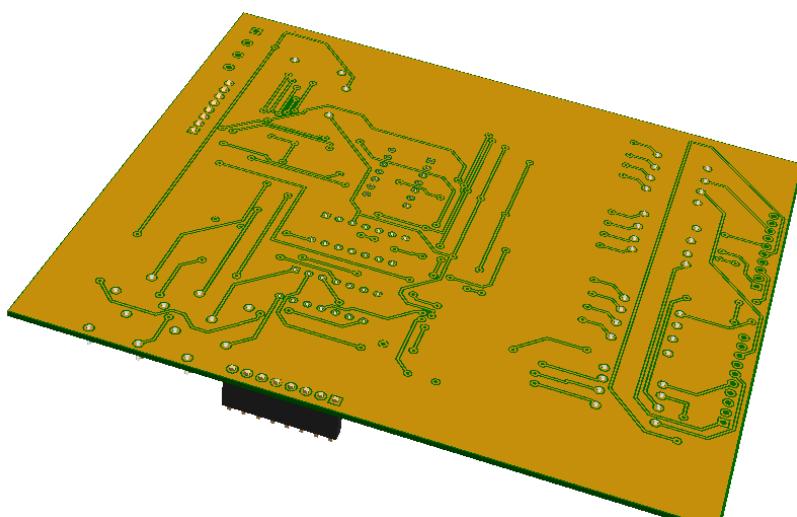


Figura 74: PCB placa principal en 3D.

Gerbers más significativos exportados como imagen:

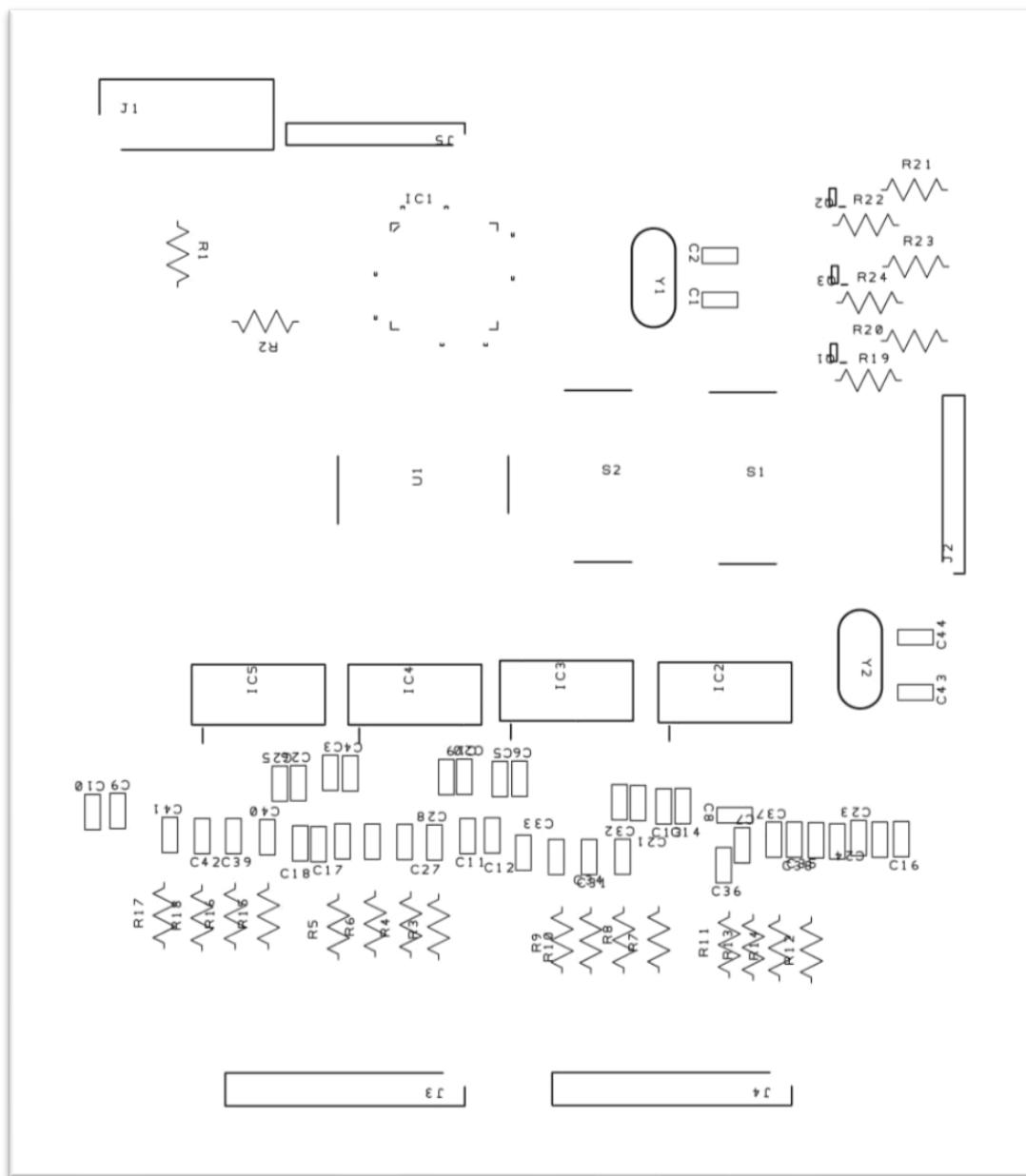


Figura 75: serigrafía capa superior.

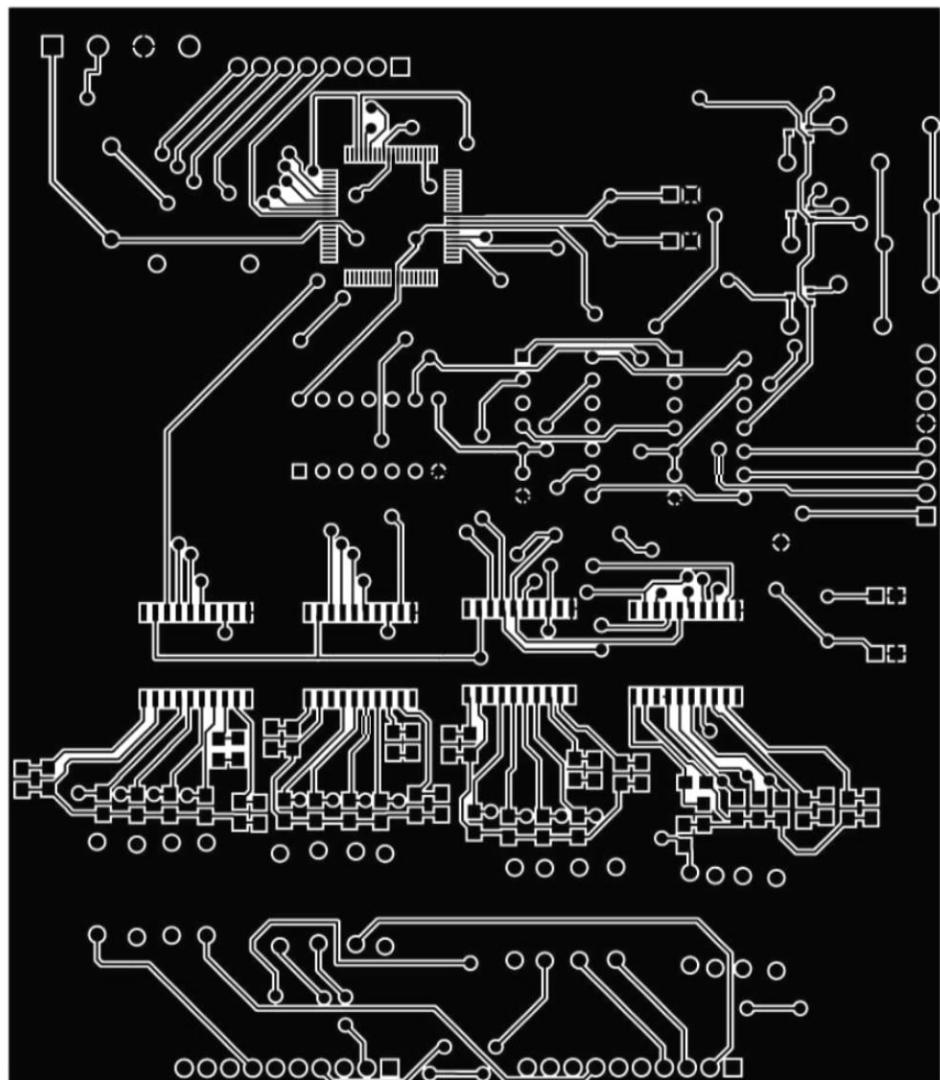


Figura 76: cobre capa superior.

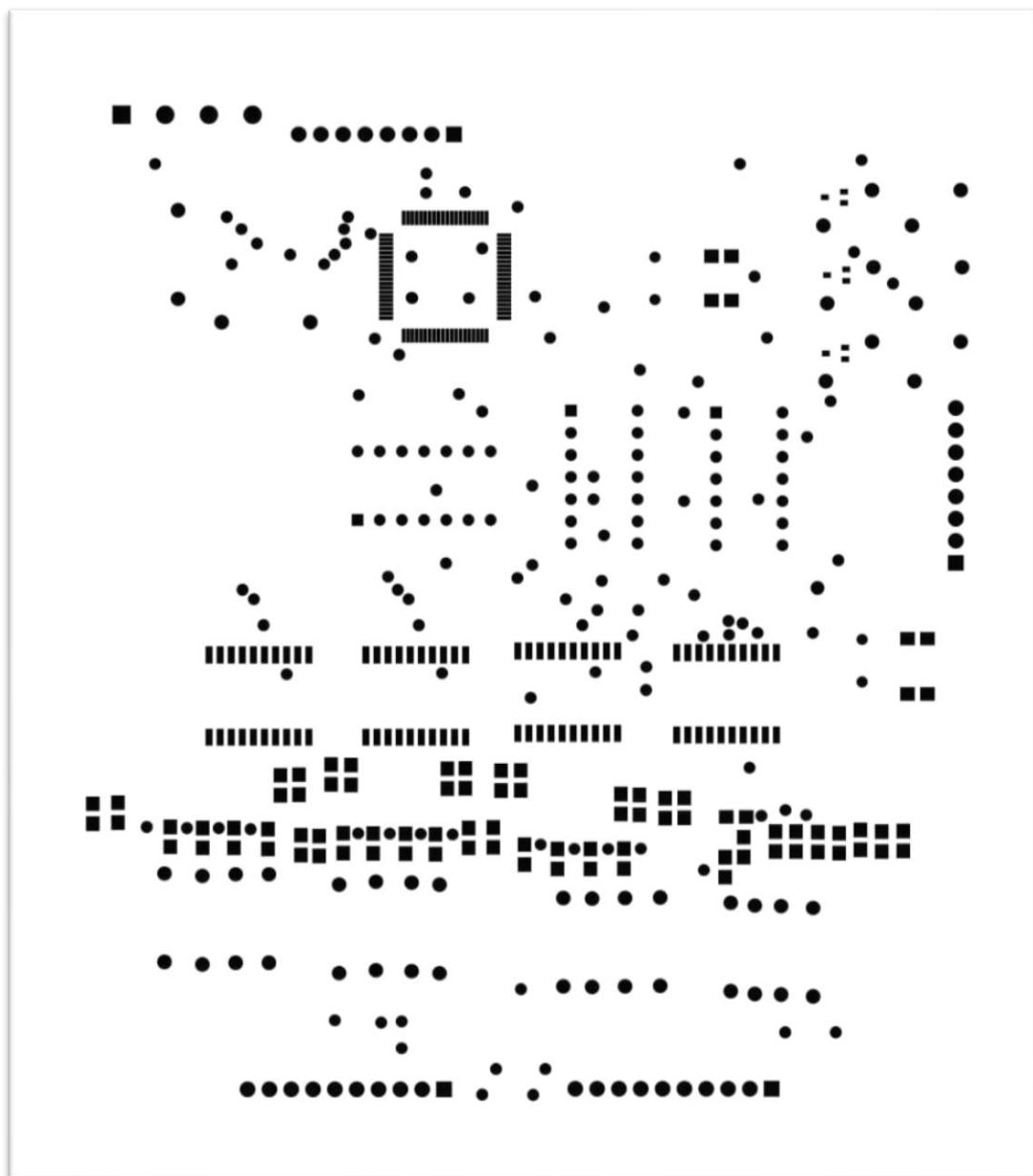


Figura 77: pasta soldadura capa superior.

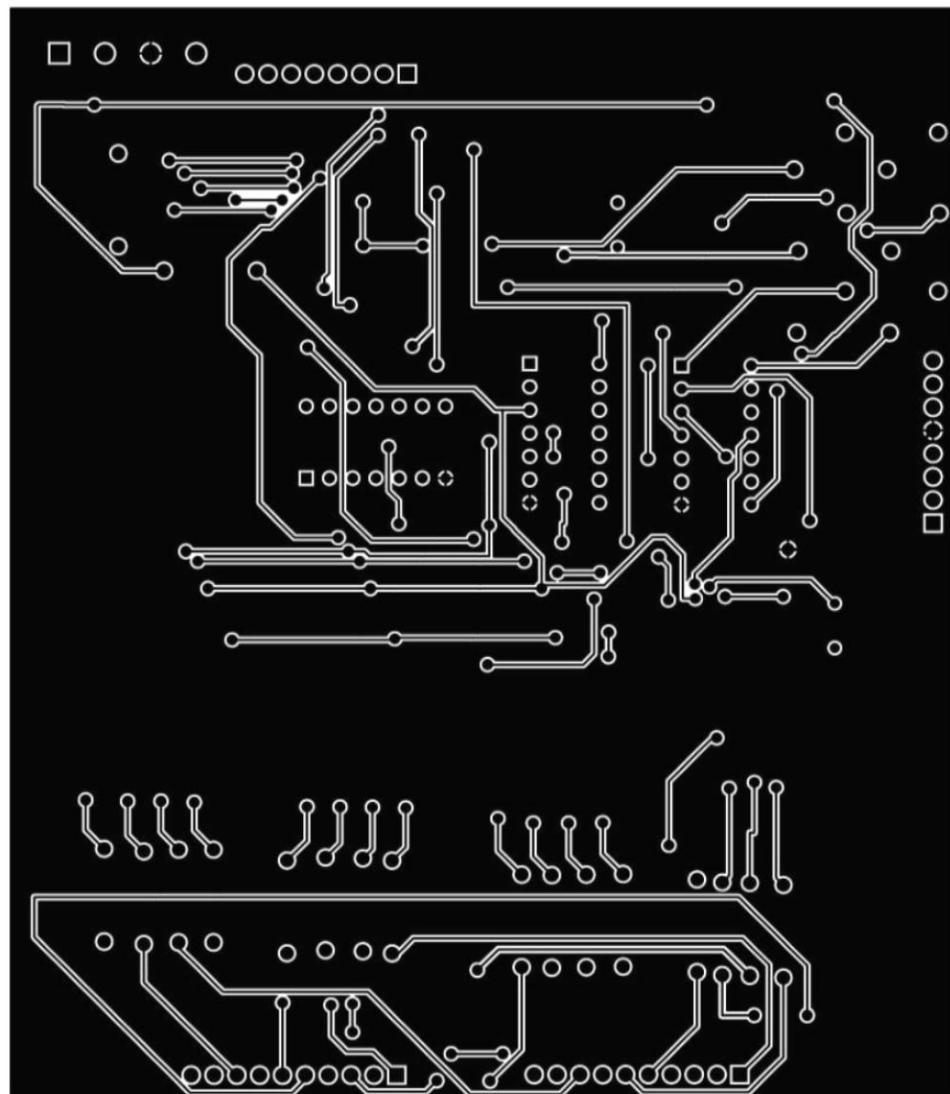


Figura 78: cobre capa inferior.

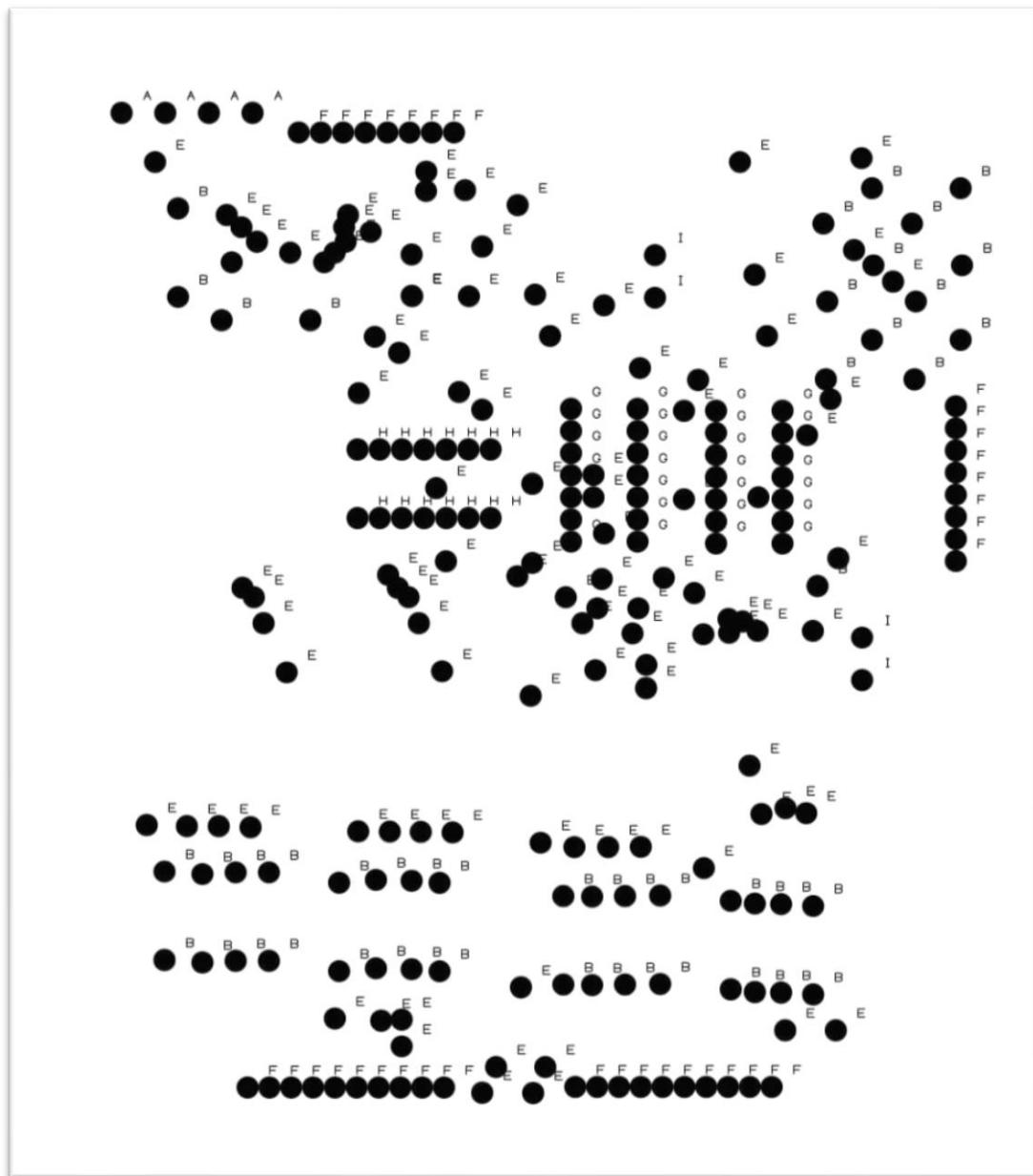


Figura 79: dibujo de los taladros.

4.3 Placa de adquisición de datos

Esta placa contiene los circuitos necesarios para la adquisición de datos de la red eléctrica. Cuenta con conexiones para el cuadro eléctrico y transforma el voltaje e intensidad circulante por la instalación en una señal de voltaje de baja señal adecuada para ser leída por el circuito integrado ADE7912. El circuito utilizado es el mismo explicado anteriormente en el apartado de medición de voltaje e intensidad. Cuenta con dos conexiónados de 10 pines para ser conectado a la placa principal:

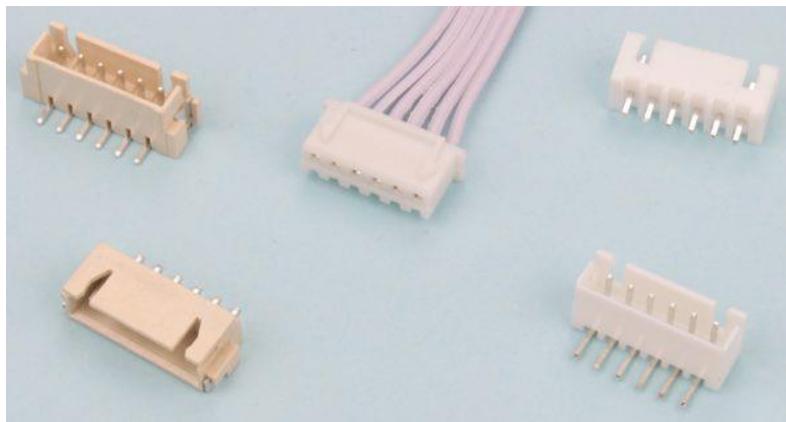


Figura 80: imagen representativa del bus de conexiones disponible.

Las conexiones para la red eléctrica usan conexiones de la marca Phoenix y son como la siguiente representación:



Figura 81: imagen representativa de las conexiones al cuadro eléctrico.

Se utilizan conectores similares en la placa de la fuente de alimentación, y en la placa principal para la conexión de los actuadores, y de la alimentación. En esta placa hay que conectar todas las fases incluido neutro y la tierra, pero sin cortar a ésta última. El esquemático completo de esta placa y los archivos gerbers están a continuación:

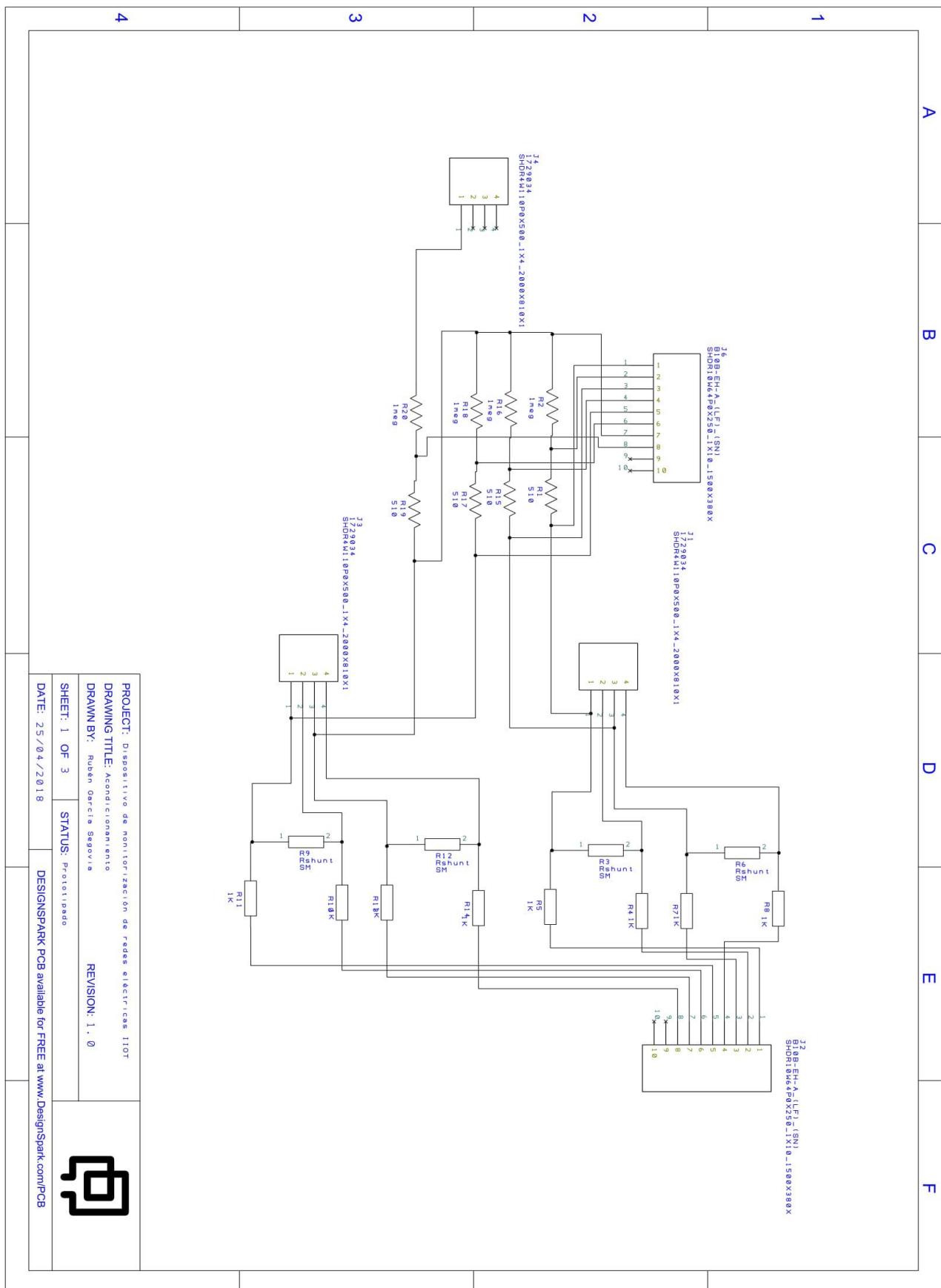


Figura 82: Esquemático placa de adquisición de datos.

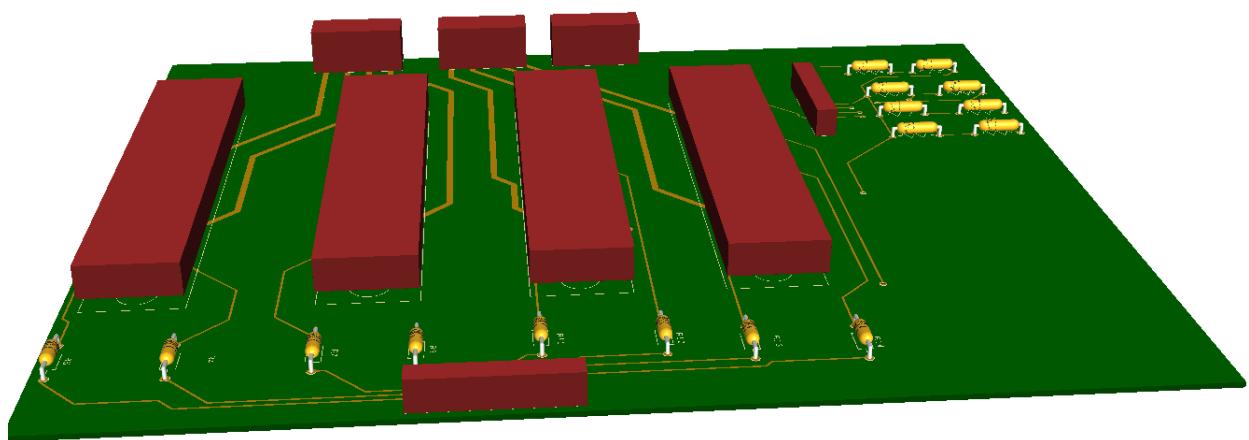


Figura 83: PCB adquisición de datos en 3D.

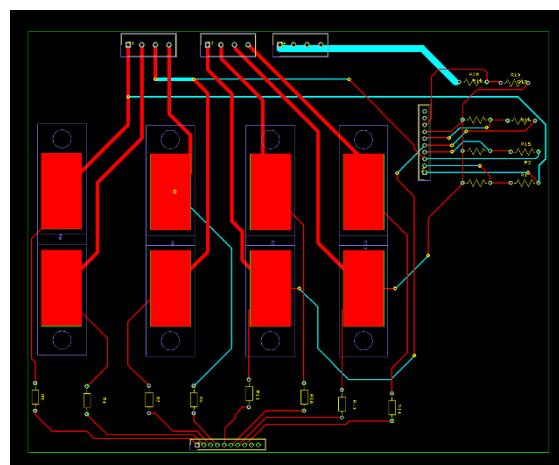


Figura 84: Negativo de la PCB.

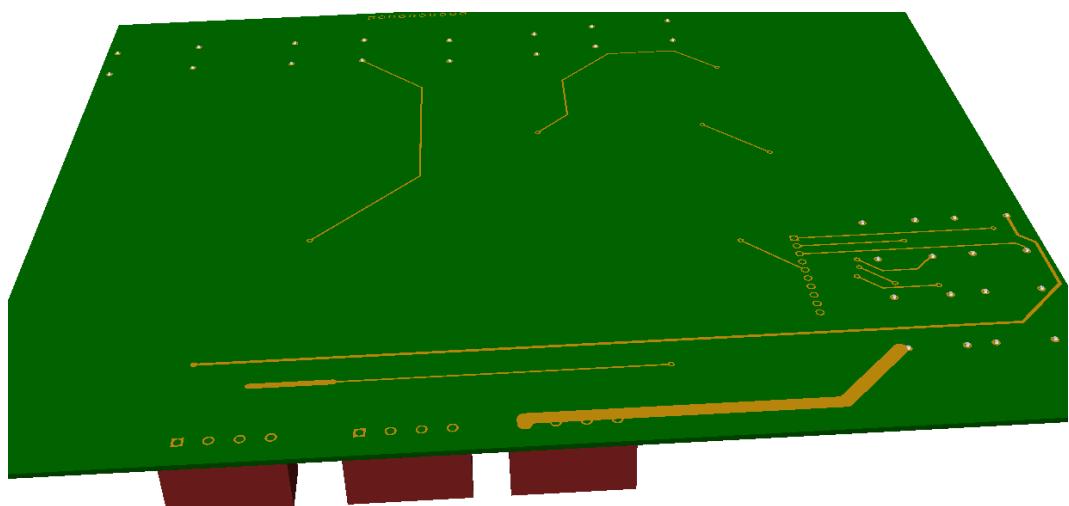


Figura 85: PCB adquisición de datos en 3D.

gerbers más significativos exportados como imagen:

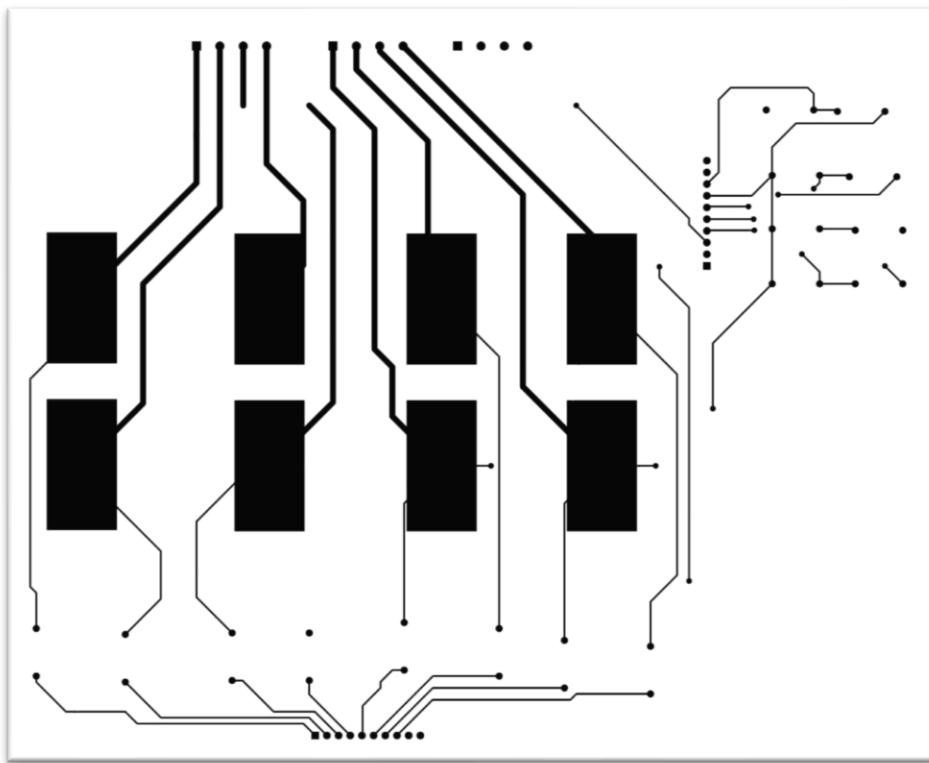


Figura 86: cobre capa superior.

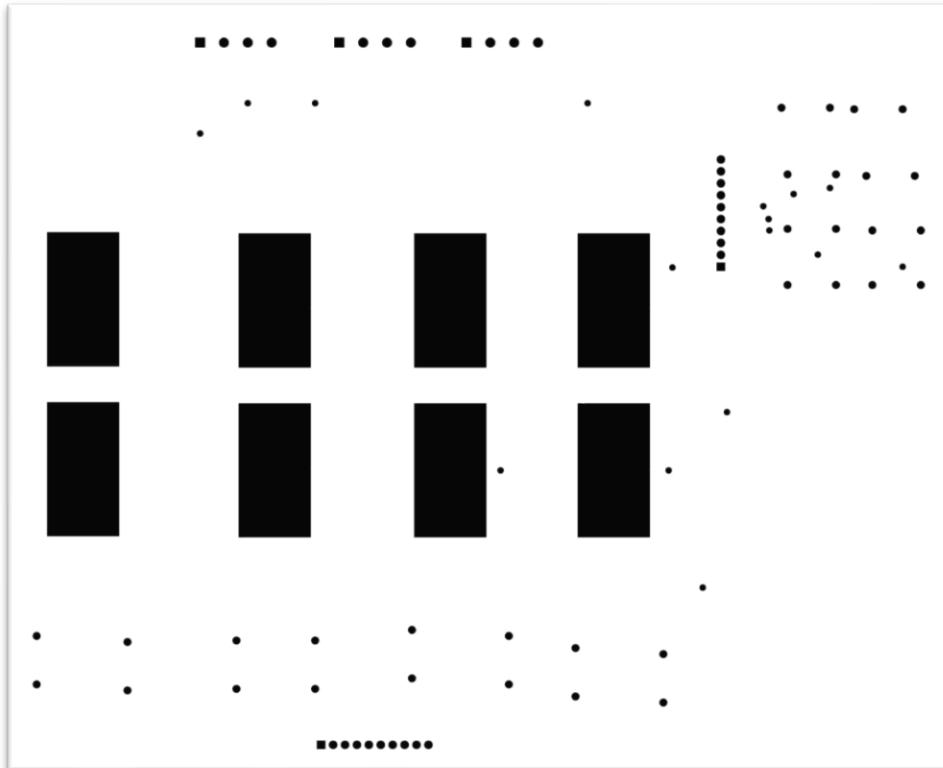


Figura 87: pasta de soldadura parte superior.

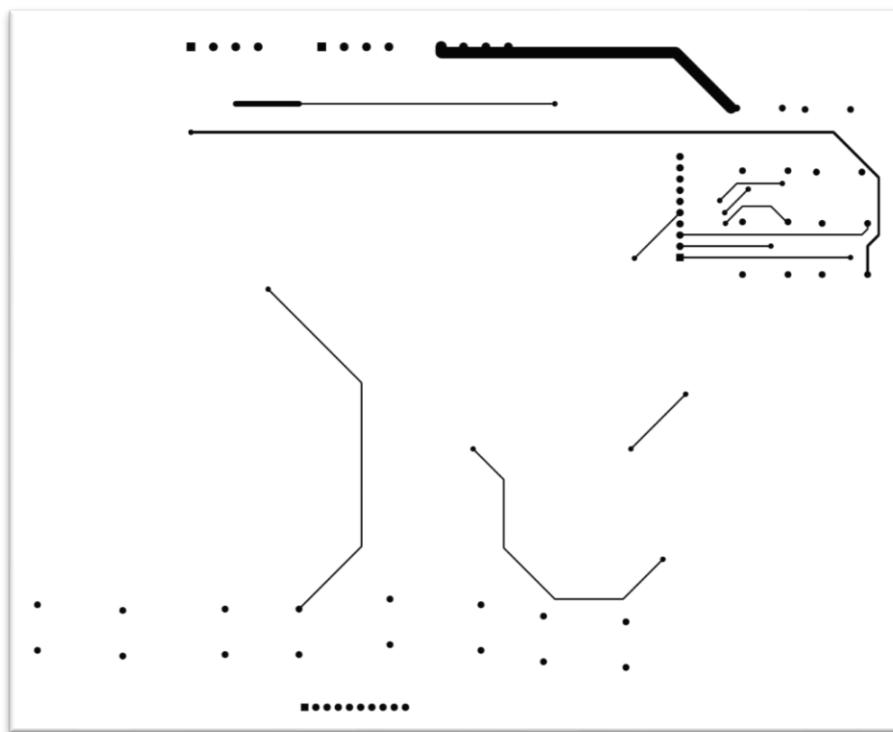


Figura 88: cobre capa inferior.

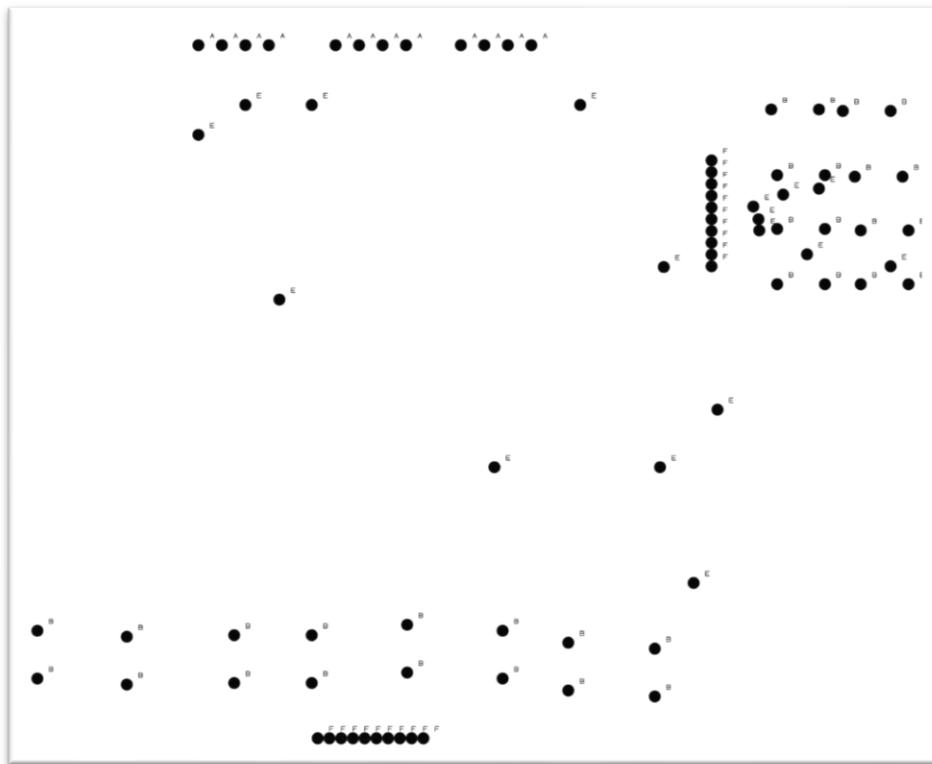


Figura 89: dibujo de los taladros..

5. Desarrollo software

El desarrollo software se va a separar en cuatro capas o niveles de programación: Sistema, web, aplicación android y GUI táctil. En la siguiente imagen se puede ver el diagrama de bloques del sistema:

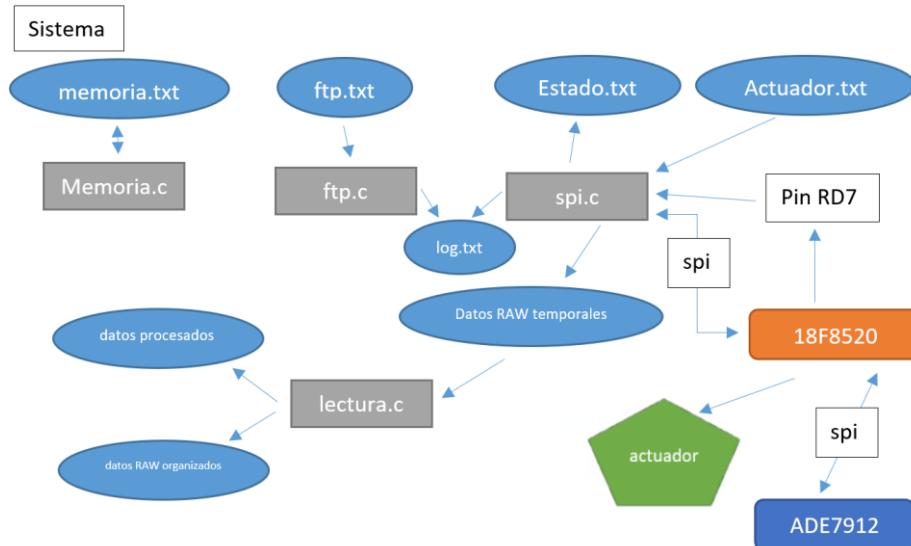


Figura 90: diagrama de bloques del sistema.

Nota: ‘*datos raw*’ son los archivos sin procesar del script `spi.c`, se almacenan los valores de la red eléctrica y tiempo relativo.

En el diagrama se pueden apreciar los archivos (en azul claro) que intervienen en el funcionamiento del sistema:

- **Memoria.txt:** Almacena el dato de la memoria ocupada por los datos *raw*. El script `memoria.c` lo crea y a su vez lo comprueba periódicamente.
- **Ftp.txt:** Archivo que almacena el estado del FTP (Activado – 1 o desactivado – 0). El script `ftp.c` comprueba periódicamente este fichero y se encarga de desactivar o activar el servidor ftp.
- **Estado.txt:** Archivo que almacena el estado de la red eléctrica (0 – Estable, 1- Corte).
- **Actuador.txt:** Archivo que almacena el estado del actuador. Si el contenido de este fichero es un 1, el script `spi.c` enviará de forma inmediata al PIC 18F8520 la orden de activar el actuador.
- **Datos raw / Datos procesados:** En primera instancia se tienen los datos *raw* por unidad de tiempo. Estos archivos *raw* desordenados los genera el script `spi.c`. Una vez por minuto, el script `lectura.c` lee estos datos, los procesa, y guarda dos copias de forma ordenada en la base de datos: Los datos procesados y los datos *raw*. Cuando esto es realizado, el script `spi.c` está capturando los nuevos datos *raw* del minuto siguiente.

- **Log.txt:** En el log se registran todos los eventos importantes del sistema así como la fecha y hora en el que se han producido.

A continuación se puede apreciar el diagrama de bloques de la capa del servidor web y la capa de la aplicación móvil:

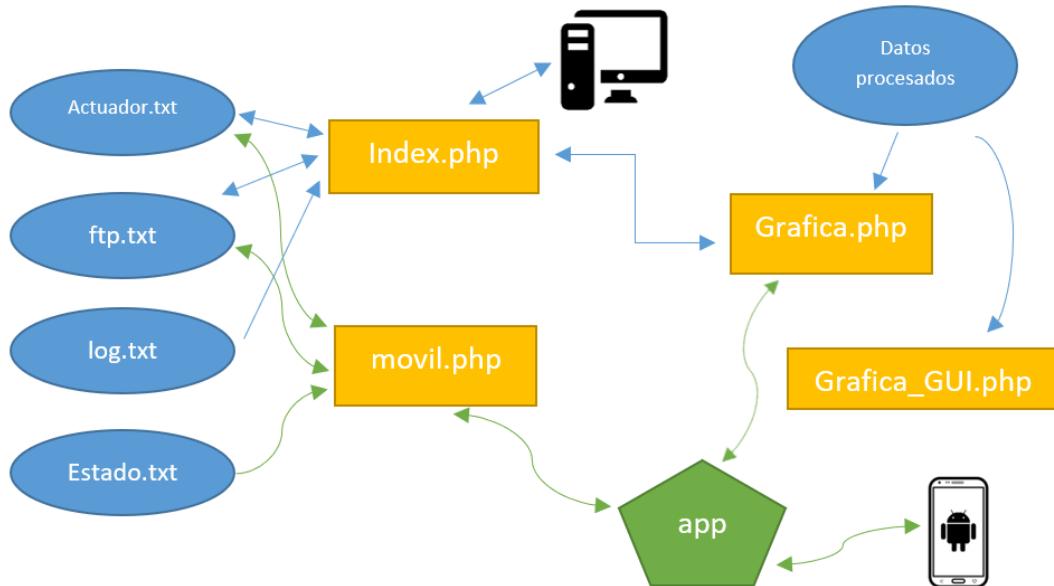


Figura 91: Diagrama de bloques servidor web y aplicación android

Capa servidor web:

- **Index.php:** Página web que presenta la gráfica seleccionada, permite actuar y leer los estados del actuador y del servidor ftp, y además muestra el log.txt para poder ver los eventos del sistema.
- **Grafica.php:** Al pasarle por url los parámetros de la gráfica requerida, devuelve una imagen de los datos correspondientes al día de la petición. Dichos datos los coge de la base de datos.
- **Grafica_GUI.php:** De igual forma que grafica.php, devuelve los parámetros de la gráfica requerida, sin embargo los devuelve en una imagen con una resolución especial para la interfaz gráfica requerida en la pantalla táctil. Se explicará con más detalle más adelante.

Capa aplicación móvil:

- **Movil.php:** Permite leer y actuar sobre el actuador y el servidor ftp (mediante sus respectivos ficheros .txt) y permite leer el estado actual de la red (estado.txt).
- **App:** Aplicación programada en Java que se conecta a movil.php y Grafica.php para obtener todos los datos necesarios para mostrar en la aplicación móvil. También permite activar o desactivar el actuador y servidor FTP.

Por último, se tiene el diagrama de la capa de la interfaz gráfica de la pantalla táctil:

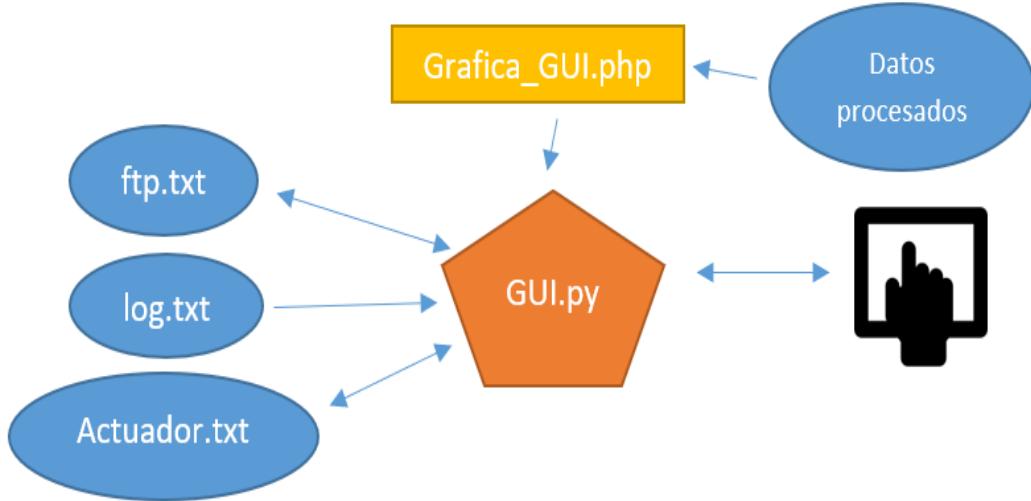


Figura 92: Diagrama de bloques interfaz gráfica

Como se puede ver en la imagen anterior, la interfaz gráfica y aplicación de la pantalla táctil está controlada por un archivo python central, el cual coge el archivo log.txt para mostrar todos los eventos del sistema, usa a su vez un script en el servidor web para descargarse las imágenes de las gráficas que se van a mostrar, y controla el estado del servidor FTP y del actuador a través de los ficheros de texto correspondientes.

Debido a la baja resolución de pantalla que posee la pantalla LCD táctil que se dispone, se ha tenido que modificar el script grafica.php y crear uno secundario (grafica_gui.php) para la creación de las gráficas en un formato y resolución acorde con la pantalla táctil.

6. Diagramas de flujo

6.1 Sistema

PIC18F8520

Se va empezar con el diagrama de flujo del código grabado en el PIC 18f8520:

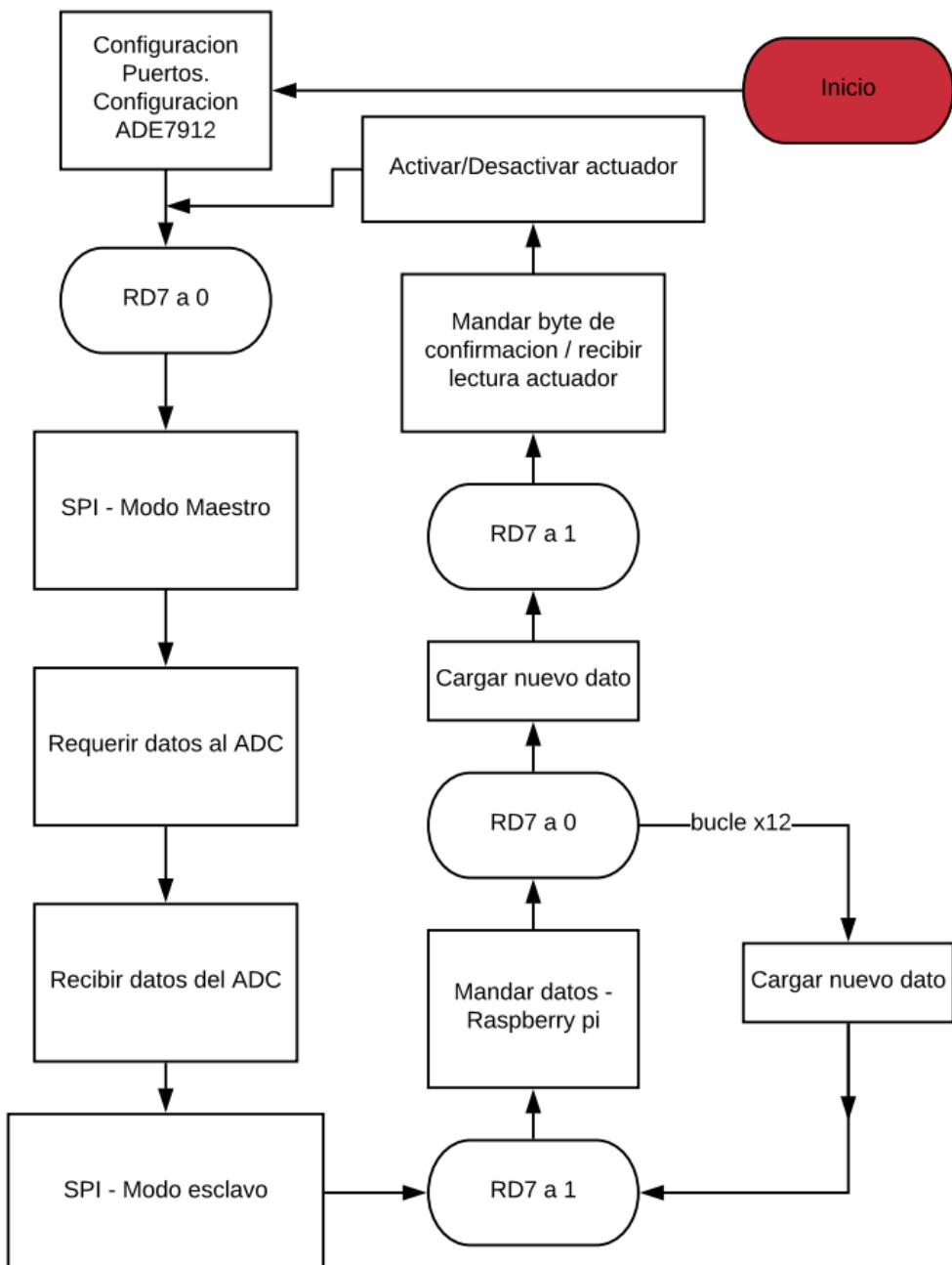


Figura 93: Diagrama de flujo PIC

Como se puede apreciar, se envian 12 bytes más uno de confirmación. Estos 12 bytes son los datos de 24 bits de resolución que ofrece el ADE7912. Estos datos son la intensidad, el voltaje circulante y la temperatura. Como se puede ver, el pin RD7 controla la comunicación entre el ADC y la raspberry pi.

SPI.c

A continuación se tiene el diagrama de flujo del script spi.c:

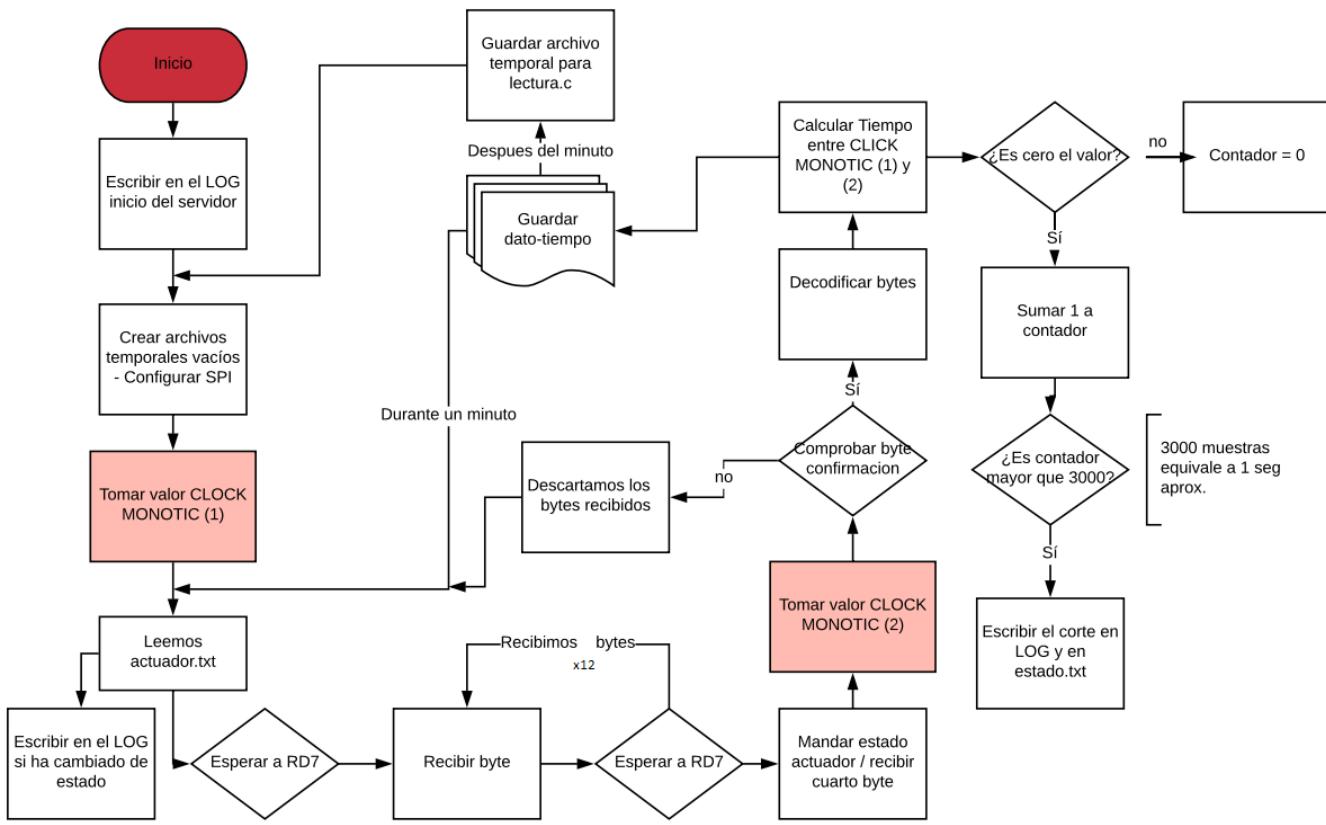


Figura 94: Diagrama de flujo script SPI

Como se puede apreciar en el diagrama de flujo, una vez arrancado el script no se detendrá hasta que se pare de forma manual. El script lee todos los datos del PIC durante un minuto y los almacena diferentes archivos de texto que posteriormente lectura.c coge para su tratamiento. A la vez que realiza la operación de obtención de datos, también detecta en tiempo real los cortes en la alimentación y manda activar/desactivar al actuador depende de como lo ordenemos en el archivo de texto.

Se utiliza la función CLOCK MONOTIC de raspberry pi ya que es una función especialmente diseñada para la temporización, ya que es un contador que no depende de la fecha actual recolectada de un servidor, sino que siempre va en aumento, por ello es totalmente fiable y siempre que restemos el valor (1) al valor (2) obtendremos un valor positivo en microsegundos.

También se puede observar como los datos (Tanto intensidad, voltaje como temperatura) son descartados si el byte de confirmación recibido es erróneo. De esta forma se pueden evitar los valores incorrectos debido a la desincronización entre dispositivos.

lectura.c

En la siguiente figura se muestra el diagrama de flujo del script encargado de la trata de datos:

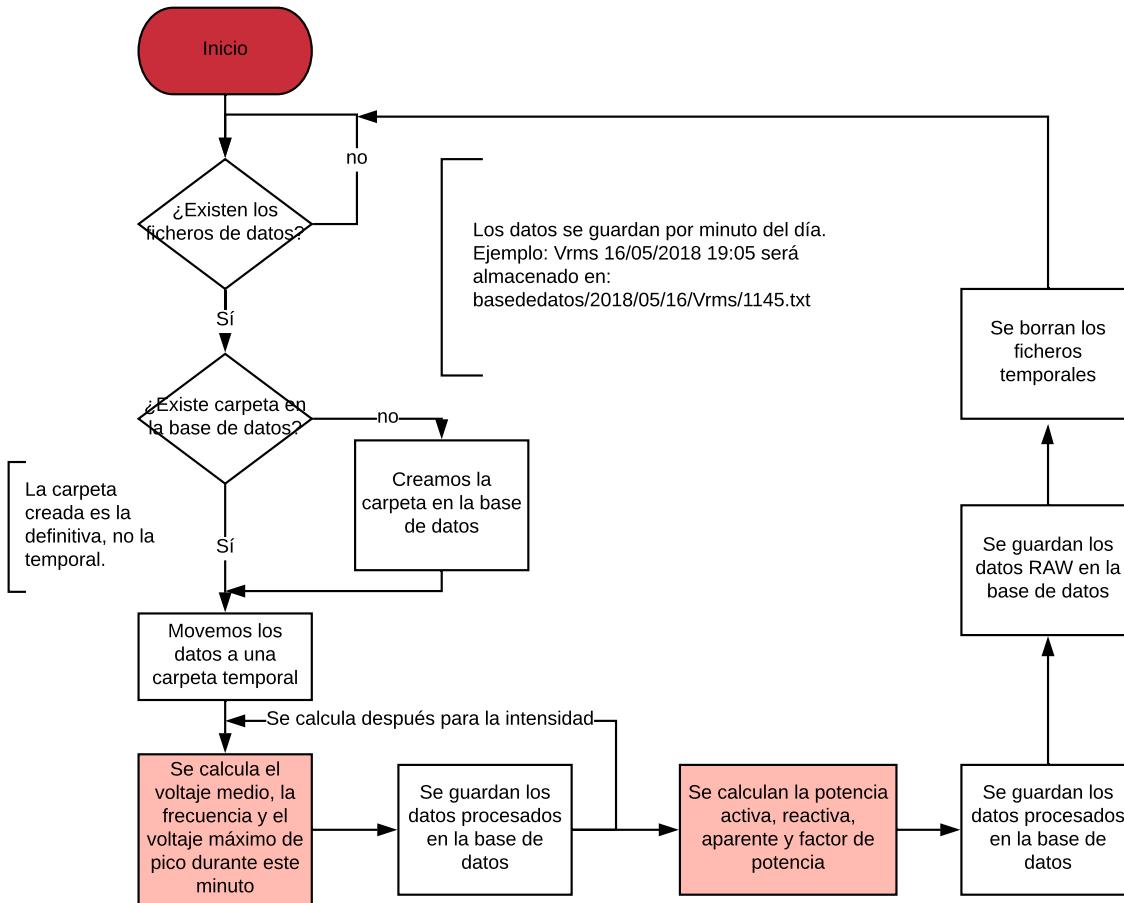


Figura 95: Diagrama de flujo script lectura

Primero se entra en un bucle infinito hasta que spi.c le cede los datos raw. Entonces crea la carpeta oportuna en la base de datos (según la fecha en la que se encuentre el sistema) y mueve los ficheros creados por el script spi.c hacia una carpeta temporal. Se calculan los valores medios de Vrms e Irms, Ifrec, Vfrec y Ipico. Una vez que se tienen estos 6 datos calculados, se procede a calcular la potencia activa, reactiva y aparente del sistema, junto con el factor de potencia.

Para calcular Vrms (o Irms) se ha procedido a usar la siguiente ecuación:

$$V_{rms} = \sqrt{\frac{1}{t} \int_0^t V^2(t) dt} \quad (19)$$

Para calcular las potencias y el factor de potencia, se han usado las siguientes ecuaciones:

$$Pot_{aparente} = I_{rms} * V_{rms} \quad (20)$$

$$Pot_{activa} = \frac{1}{t} \int_0^t V * i \, dt \quad (21)$$

$$Pot_{reactiva} = \sqrt{Pot_{aparente}^2 - Pot_{activa}^2} \quad (22)$$

$$fdp = \frac{Pot_{activa}}{Pot_{aparente}} \quad (23)$$

Para el caso de la potencia aparente, como Irms y Vrms ya están tomadas en el intervalo de un minuto, la potencia aparente también será la total durante ese minuto. Para el caso de la potencia activa tendremos que realizar el sumatorio (integral) de los valores durante la duración de todo el minuto. Por último, la potencia reactiva se resuelve por el teorema de pitágoras en el triángulo de potencias.

Para el cálculo de la frecuencia, simplemente se cuentan las veces que el valor de frecuencia se aproxima al valor pico, por lo que en un intervalo determinado de tiempo se cuentan ‘x’ picos en la onda, lo que permitirá sacar la frecuencia media de la onda durante ese intervalo.

Los datos sin tratar se guardan en ‘/home/pi/Desktop/basededatos/datos/fecha’

Los datos tratados se guardan en ‘home/pi/Desktop/basededatos/fecha’

Haciendo clara esta distinción se podrá acceder de forma más cómoda a los datos necesarios a través del servidor FTP para su procesado mediante software específico (LTspice, por ejemplo).

También se ha diseñado una función que calcula el desfase entre dos fases de intensidad o tensión. Esta función está pensada para el caso real de estar monitorizando las 3 fases a la vez. Permite comprobar si las fases están en sus ángulos correctos.

ftp.c

El script que activa o desactiva el servidor FTP es muy sencillo:

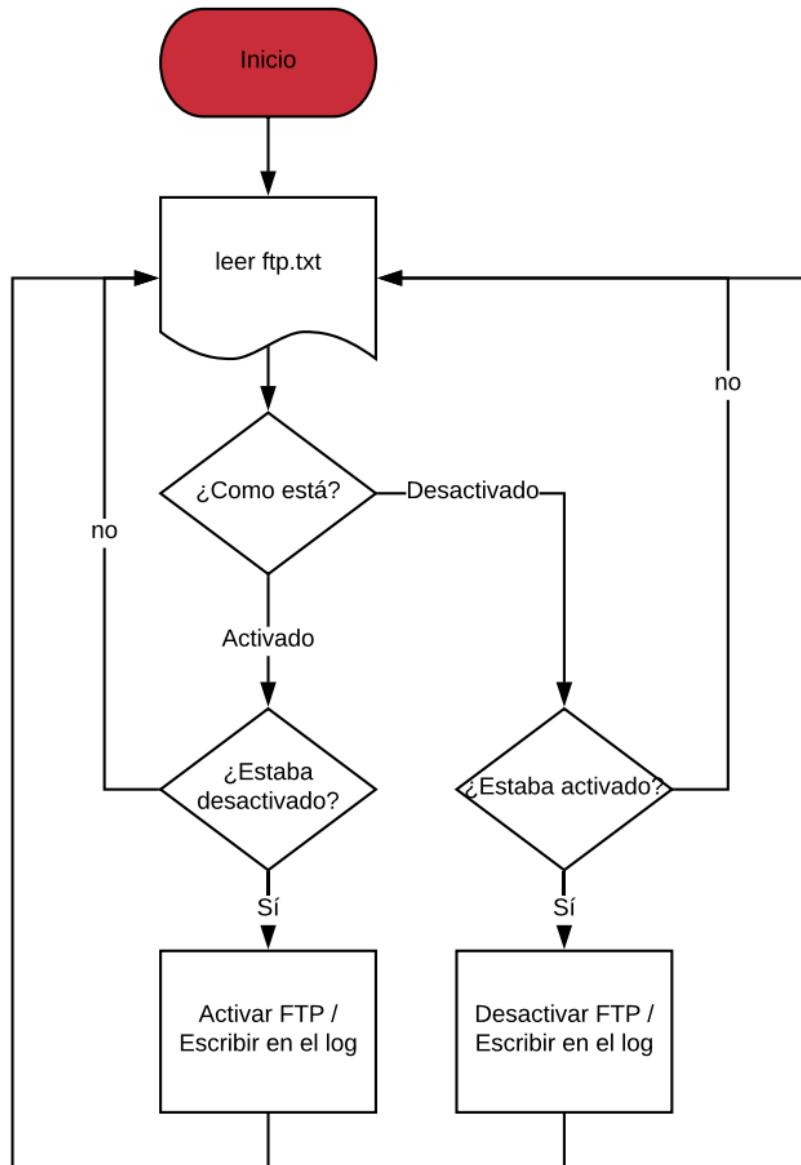


Figura 96: Diagrama de flujo del script ftp

El script está constantemente leyendo el archivo ftp.txt y comprueba que no se haya realizado ningún cambio en su valor. Esta comprobación se realiza una vez por segundo, por lo que es de esperar tal delay cuando se activa o desactiva el FTP.

Memoria.c

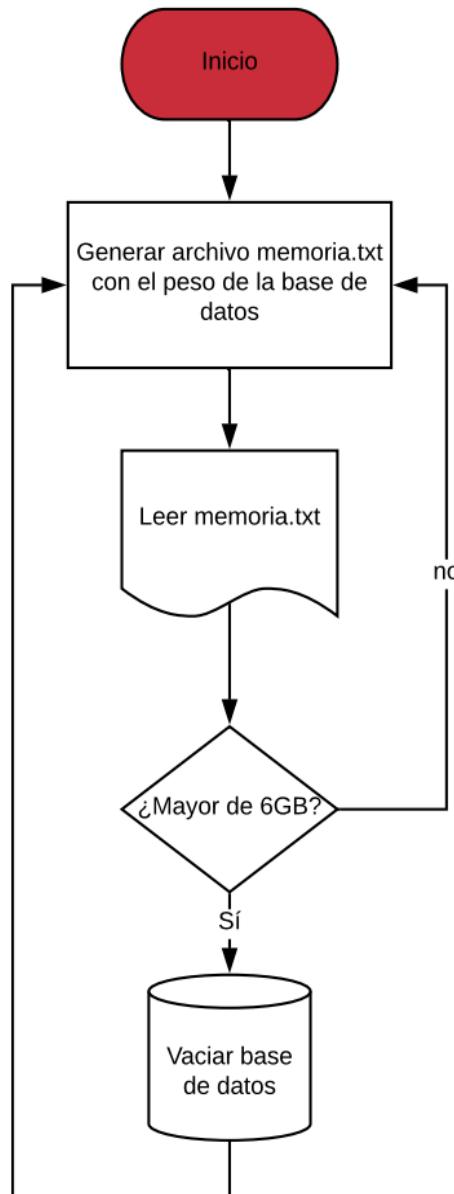


Figura 97: Diagrama de flujo del script memoria.

El script está constantemente creando el fichero memoria.txt con el peso ocupado por la base de datos. Realizamos esto debido a que la instrucción que recupera el tamaño de los archivos es una función del sistema ‘du -bsh’ y no una función de C. Cuando la base de datos supera los 6GB de datos (aproximadamente cada tres horas y media). Este límite se puede aumentar hasta los 15GB debido al espacio disponible en la tarjeta de la raspberry pi.

Los datos borrados son los archivos *raw* de la base de datos, por lo que las representaciones, datos procesados y gráficos no serán borrados del sistema.

6.2 Servidor web

[Index.php](#)

En la siguiente figura se puede ver el resultado final del servidor web:

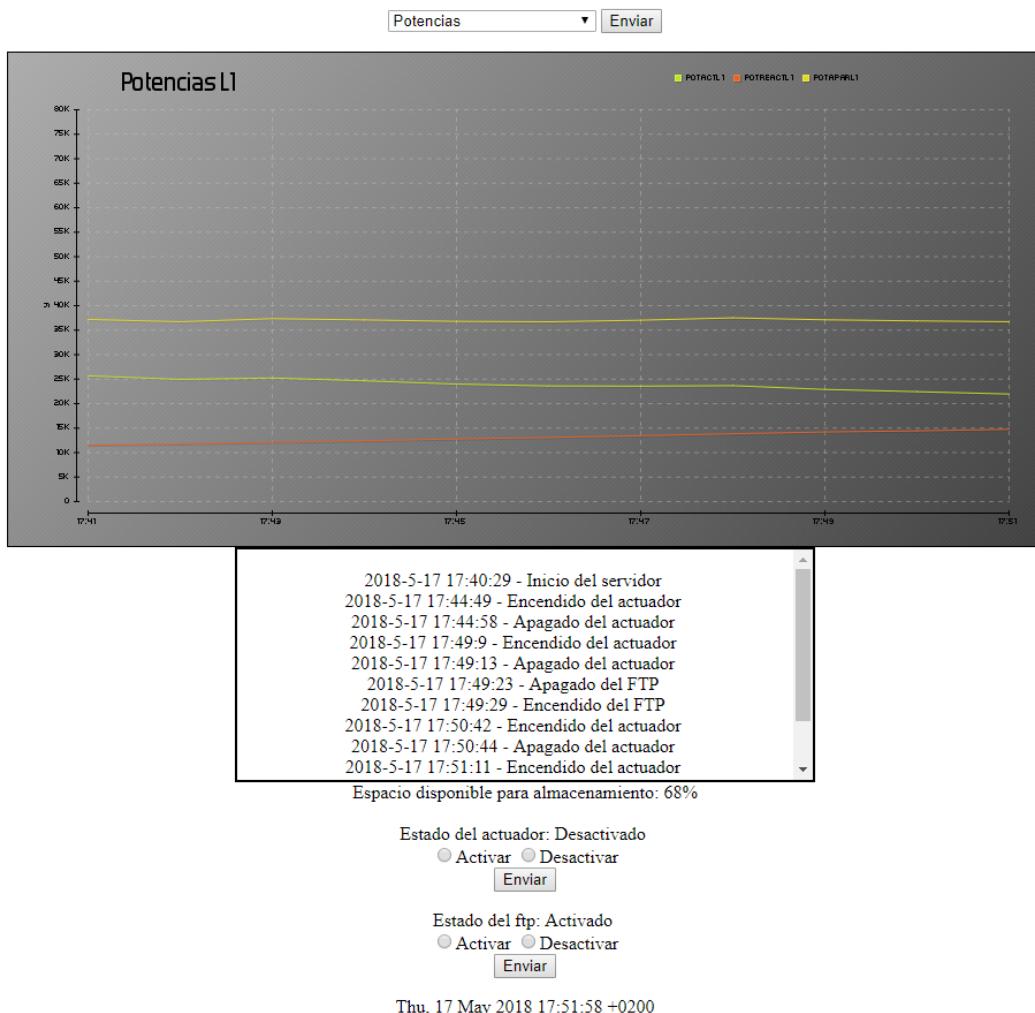


Figura 98: Página web.

Como se puede apreciar, se muestra la gráfica de las potencias en nuestro sistema (Aparente en amarillo, Activa en verde y reactiva en rojo). Con el primer botón de “enviar” junto con la lista desplegable podremos seleccionar que gráfica mostrar entre: Vrms/Irms, Potencias, frecuencias y factor de potencia.

También se puede ver como el log está presentado dentro de una caja de texto con un scroll vertical donde podremos ver todos los eventos importantes del sistema. Junto con esta caja de texto contamos con el estado del actuador y del ftp, ambos accionables desde la página web.

A continuación se pone el diagrama de flujo:

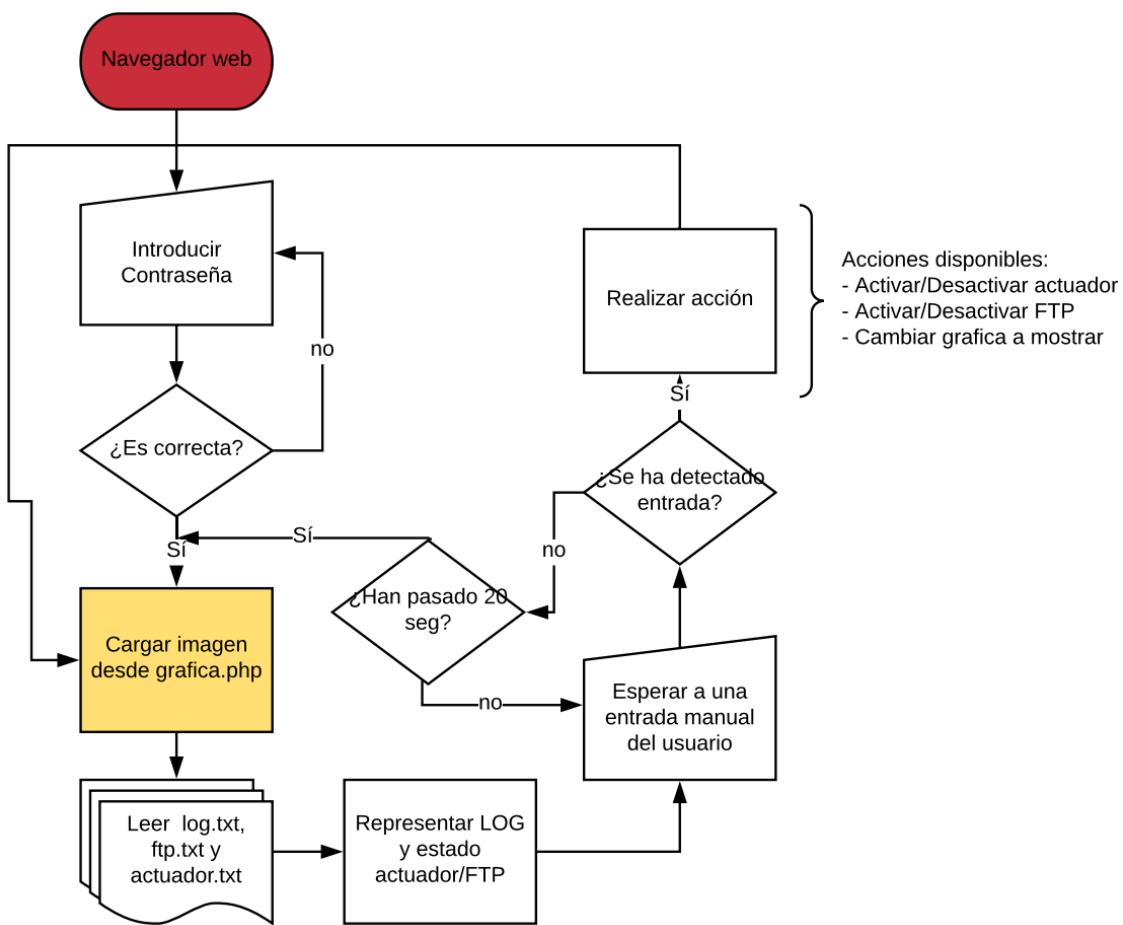


Figura 99: Diagrama de flujo pagina web.

Lo primero que se pide es acceder con una contraseña. En este caso se ha seleccionado ‘rubenelportero’ para tal efecto. Una vez que hemos introducido la contraseña, el sistema pedirá la gráfica al archivo grafica.php (la función que genera la representación que se le pida). Después lee el archivo log, ftp y actuador y representa toda la información recogida en el navegador web desde el cual se está accediendo. Esperamos a una entrada manual del usuario (activar/desactivar actuador-ftp o cambiar de gráfica a representar) y si no se ha realizado ninguna entrada manual en 20 segundos, la página se recargará automáticamente.

Grafica.php Grafica_GUI.php

A pesar de ser dos archivos diferentes, el diagrama de flujo es el mismo ya que son el mismo fichero pero cambiando la configuración realizada en la librería pChart para sacar una imagen con unas características en particular u otras.

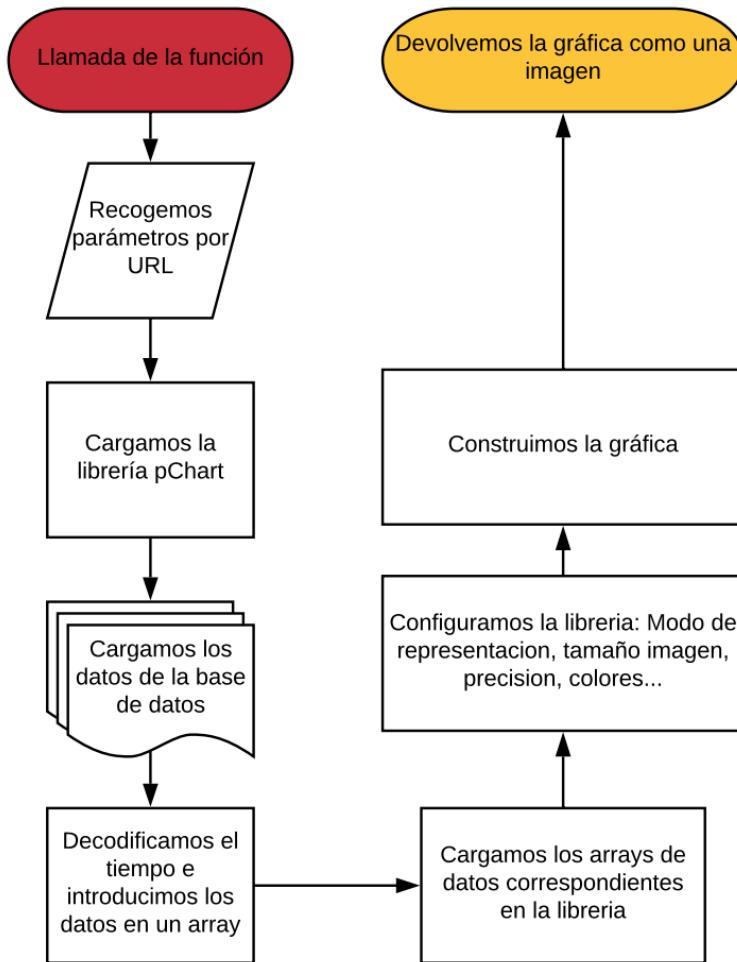


Figura 100: Diagrama de flujo función `grafica.php`.

A la librería simplemente se le pasa un parámetro por URL: la gráfica a representar. Dependiendo del valor recibido (numero entero) representará y configurará la librería para mostrar un tipo de gráfica u otro. También recolectará los datos necesarios de la base de datos dependiendo de la gráfica exigida. Por último devuelve la salida en formato imagen, perfecta para poder ser descargada por la aplicación móvil o la GUI en python o para ser impresa por pantalla en la página web

6.3 GUI táctil

gui.py

En las siguientes capturas se puede apreciar la interfaz gráfica desarrollada para la pantalla táctil:

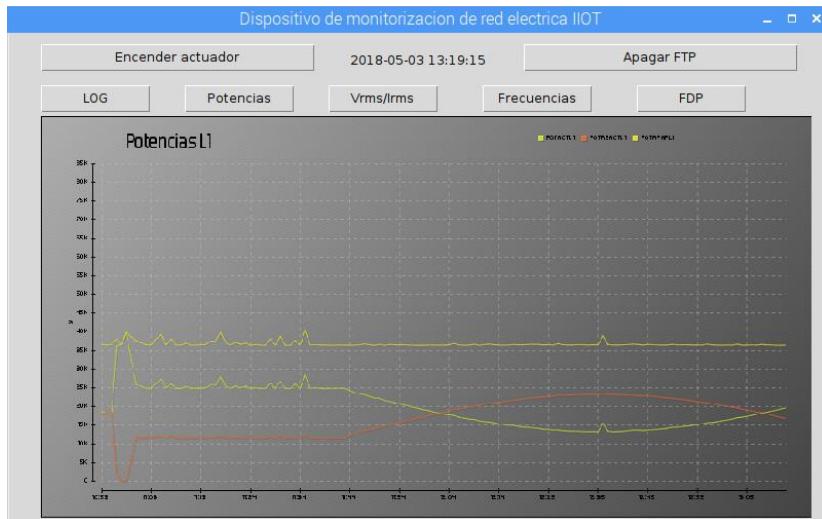


Figura 101: Interfaz gráfica táctil.

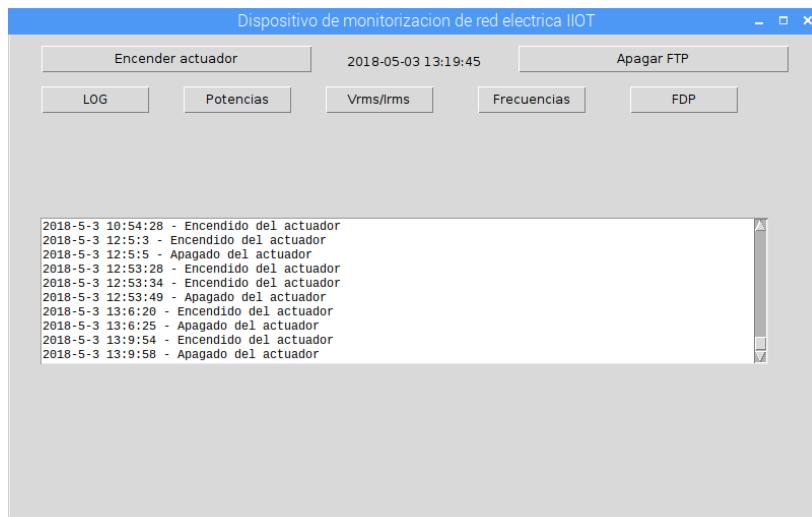


Figura 102: Interfaz gráfica táctil.

Como se puede apreciar, disponemos de un total de 5 botones para seleccionar entre los datos que se quieren mostrar y dos botones más para activar o desactivar el actuador y el ftp (A la vez que se muestran sus estados actuales).

El diagrama de flujo se muestra en la siguiente imagen:

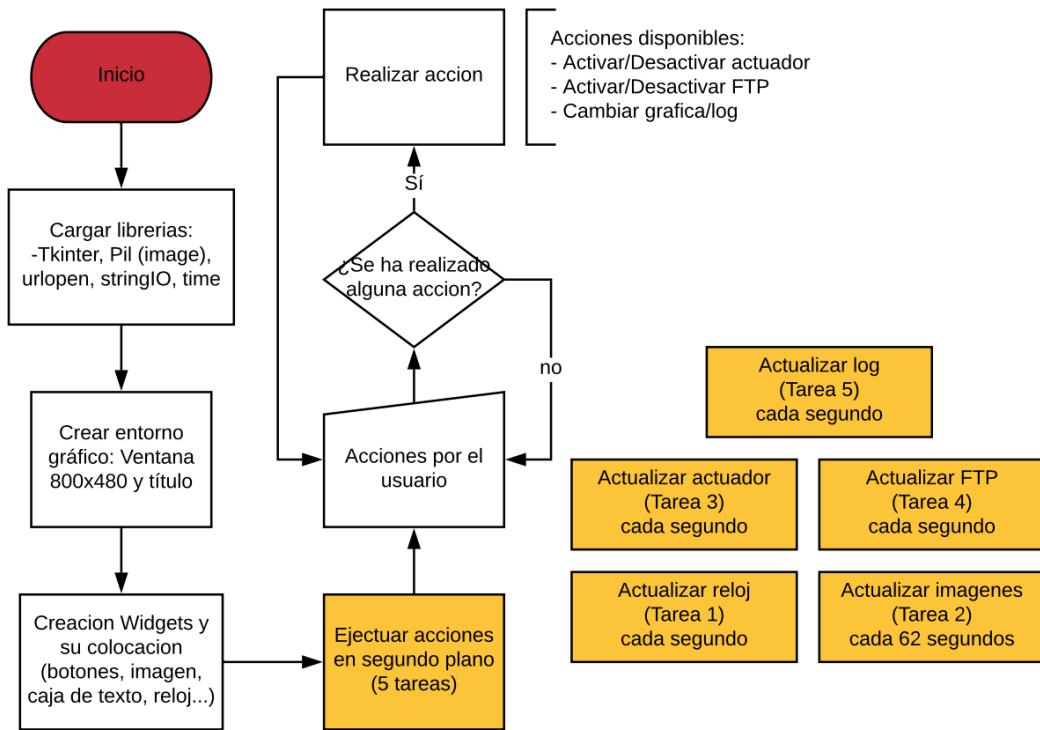


Figura 103: Diagrama de flujo interfaz gráfica.

Primeramente se cargan las librerías necesarias para el tratamiento de los datos leídos. Seguidamente se colocan los widgets en su respectivo lugar y se inician todas las acciones que se ejecutan en segundo plano. Una vez todo representado en pantalla, simplemente se espera a que el usuario pulse alguno de los botones que se muestran por pantalla para realizar la acción requerida.

Las cinco tareas en segundo plano son la siguientes:

1. Actualizar el reloj (A cada segundo).
2. Actualizar imágenes: Se descargarán todas las imágenes del servidor (Grafica_GUI.php) y se guardan en la carpeta de la interfaz gráfica, así la navegación entre gráficas es más fluida cuando vamos interactuando con la interfaz gráfica.
3. Actualizar actuador: A cada segundo leemos el archivo actuador.txt para ver si se ha producido algún cambio en el actuador desde otro periférico: La aplicación en android o el servidor web.
4. Actualizar FTP: A cada segundo leemos el archivo ftp.txt para comprobar si se ha producido algún cambio desde otro periférico: La aplicación para android o desde el servidor web.
5. Actualizar LOG: A cada segundo borramos todo el texto disponible en la caja de texto y re-escribimos el contenido de LOG.txt. Esto lo realizamos para estar totalmente actualizados de los eventos que ocurren en el sistema.

6.4 Aplicación móvil para Android

La aplicación móvil esta compuesta por un proyecto en android studio disponible en el CD del trabajo. Al contar el proyecto completo con muchos archivos, sólo se va a realizar un diagrama de flujo de la clase principal y de la programación en Java. También se va a explicar la parte del servidor web.

A continuación se exponen unas imágenes de la aplicación móvil:

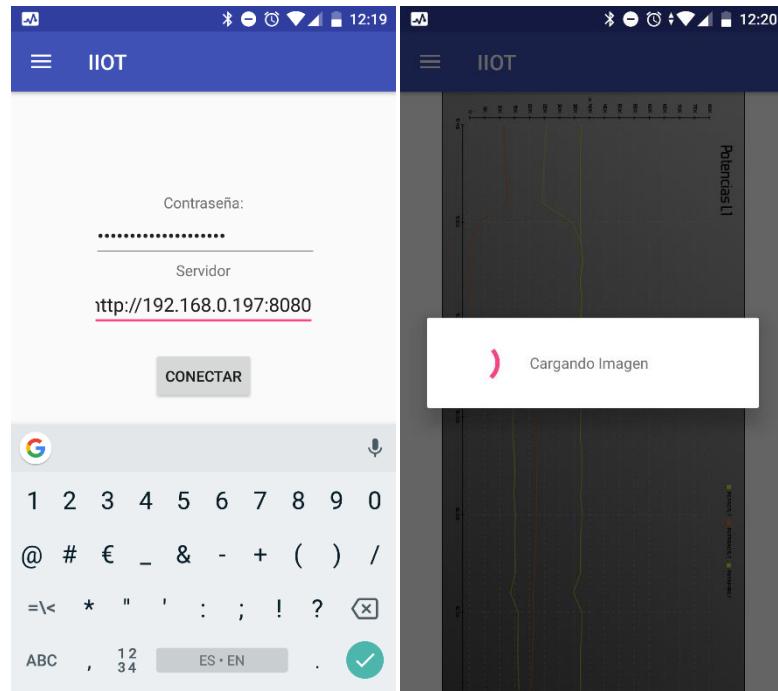


Figura 104: Capturas de pantalla de la aplicación.

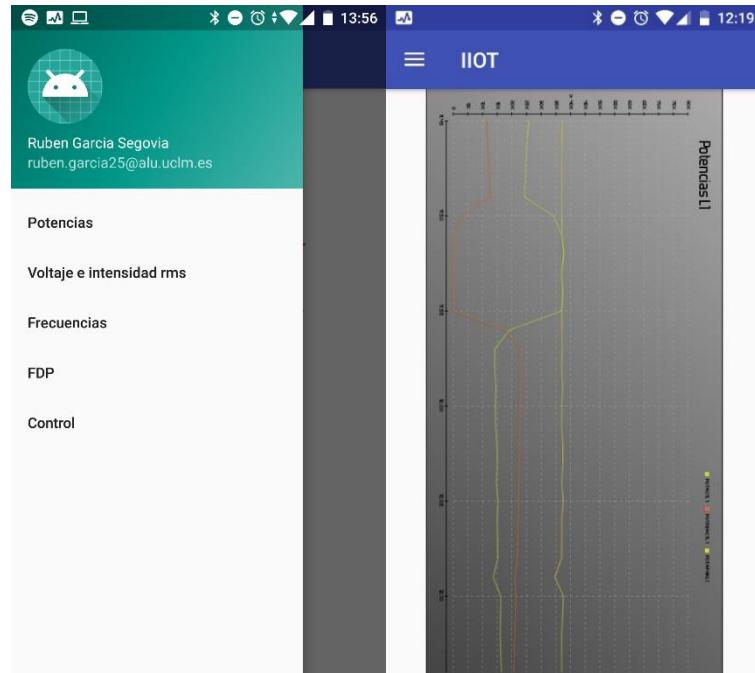


Figura 105: Capturas de pantalla de la aplicación.

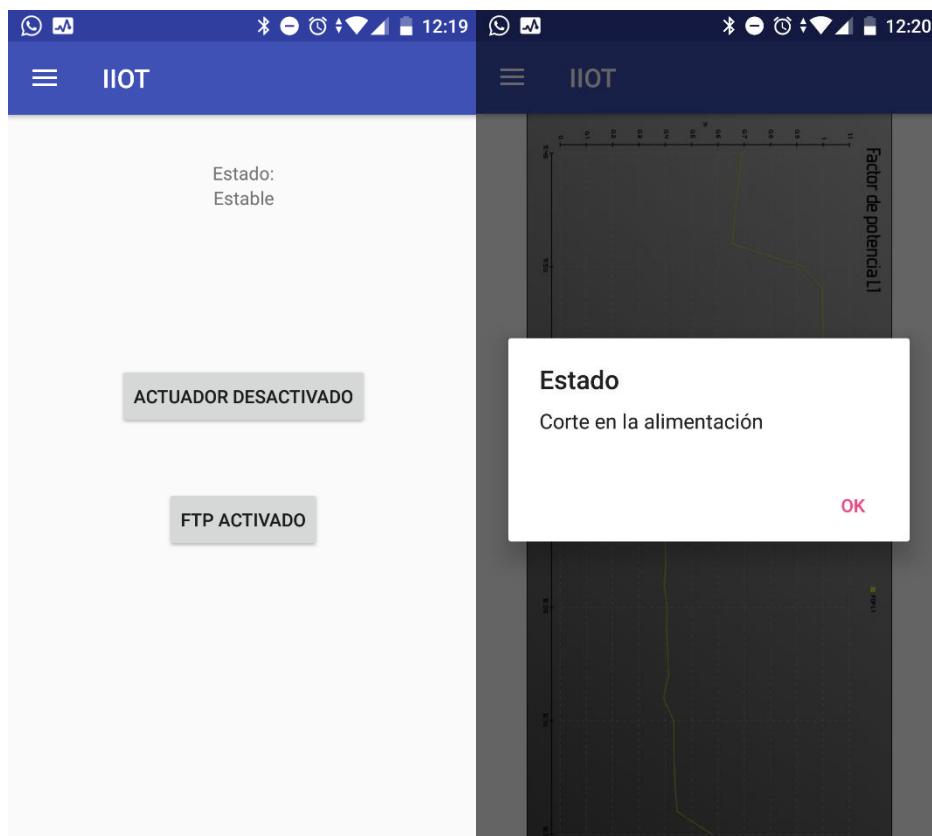


Figura 106: Capturas de pantalla de la aplicación.

Como se puede ver en las imágenes, se ha utilizado un estilo conocido como material design (diseñada por google) que ofrece una interfaz clara y sencilla de utilizar, aparte de ajustarse de manera correcta a todo tipo de pantallas y teléfonos. Sin embargo, a la hora de representar las gráficas se expone la imagen descargada del servidor con una resolución y zoom específico para móviles con pantalla táctil de 5,5 pulgadas.

movil.php

A través de este sencillo script en PHP la aplicación móvil se conecta con los servidores y es capaz de actuar sobre el actuador, el servidor FTP y puede saber el estado actual de la red.

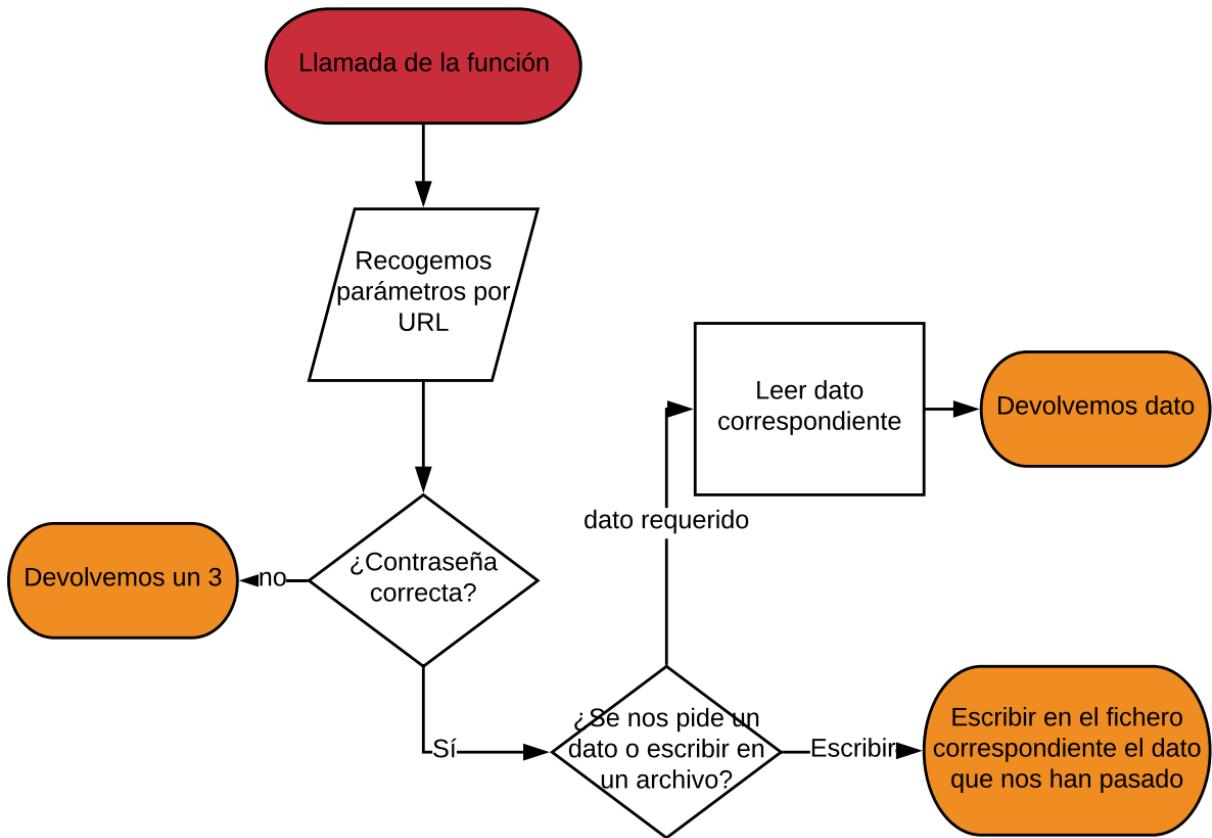


Figura 107: Diagrama de flujo script movil.php.

Los parámetros que se pasan por URL son tres: La contraseña, la variable (actuador, ftp o estado de red) y el dato (0 o 1 se escriben en la respectiva variable, un 2 si queremos leer el estado actual de la variable seleccionada).

Proyecto Android Studio

Como ya se ha comentado anteriormente, Android Studio es una suite de programación para el ecosistema de android desarrollada por Google, su uso permite programar en diversos lenguajes de programación para tal plataforma. El elegido ha sido Java. La aplicación se ha realizado a partir de un ejemplo de la librería del IDE, la cual ya deja programado la pantalla en blanco y el menú deslizable izquierdo cuando el proyecto es creado.

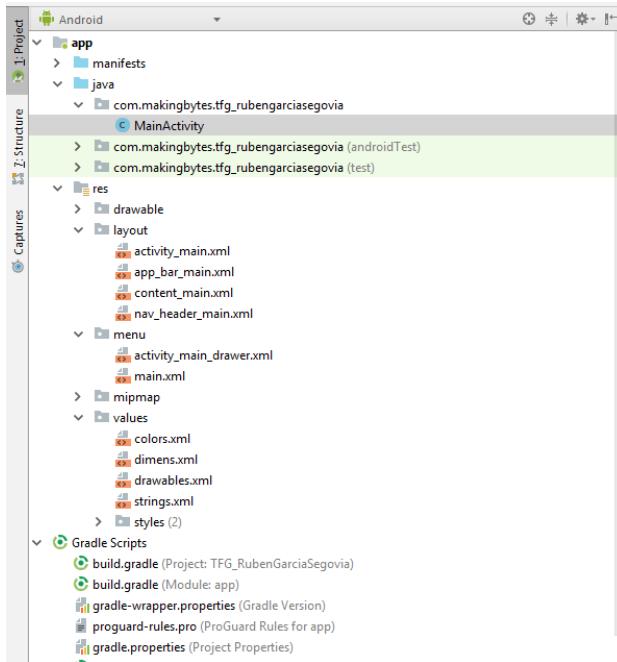


Figura 108: Estructura del proyecto.

El programa también cuenta con un editor gráfico, el cual ayudará con la programación de la parte xml y simular cómo sería la vista de nuestra aplicación en diversos tamaños de pantalla

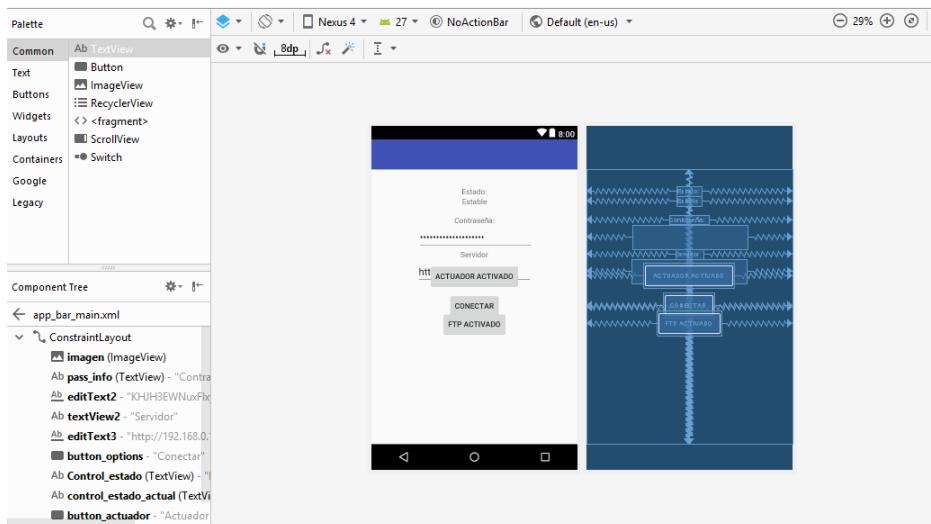


Figura 109: Editor gráfico Android Studio.

Por último, presenta todas las clases, funciones y variables creadas en nuestra aplicación de una forma sencilla y clara, ideal para poder programar de forma eficiente:

```

    < C  MainActivity
    > C  Cargalmagenes
    > C  cargar_url
    > C  lectura_estado
    > C  lectura_boton
    > C  lectura_ftp
        m T  onCreate(Bundle): void t AppCompatActivity
        m A  actualizar_control(): void
        m A  actualizar_fotos(): void
        m A  ocultarcontrol(): void
        m T  onBackPressed(): void t FragmentActivity
        m T  onCreateOptionsMenu(Menu): boolean t Activity
        m T  onKeyDown(int, KeyEvent): boolean t AppCompatActivity
        m A  pirarse(): void
        m A  ocultar(): void
        m A  seleccionarfoto(String): void
        m A  conectar_url(String): void
        m A  actualizar_estado(): void
        m T  onNavigationItemSelected(MenuItem): boolean
        f  direccion: String = "Hola"
        f  contra: String = ""
        f  imgImagen: ImageView
        f  actu: boolean = false
        f  ftp: boolean = false
        f  introducido: boolean = false
        f  respuesta_url: String = "Hello World"
        f  corte: boolean = false
        f  seccion: int = 1

```

Figura 110: Estructura de la clase MainActiviy en la aplicación android.

Puesto que la programación en Android no es secuencial, sino que va por eventos e interrupciones, se van a explicar las funciones más importantes y su funcionamiento, sin diseñar un diagrama de flujo que englobe a todo el código a la vez.

La función `onCreate(Bundle)` se ejecuta cuando la aplicación se abre. Oculta los botones de control (actuador y ftp) y oculta el widget imagen (donde posteriormente se cargarán las imágenes de las gráficas). Por lo que únicamente es mostrado por pantalla el log-in (como se puede apreciar en la figura 114). Aquí la aplicación espera a tres eventos distintos: La pulsación del botón de FTP, del botón actuador y del botón ‘Conectar’.

Como los botones FTP y actuador están ocultos, no pueden ser pulsados en este momento, por lo que el usuario únicamente puede introducir una contraseña y el servidor.

La variable booleana (true o false) llamada ‘introducido’ detecta si el primer paso de login al servidor se ha cumplido o no. Esta variable existe para no poder cambiar de sección usando el menú desplazable, ya que éste estará bloqueado.

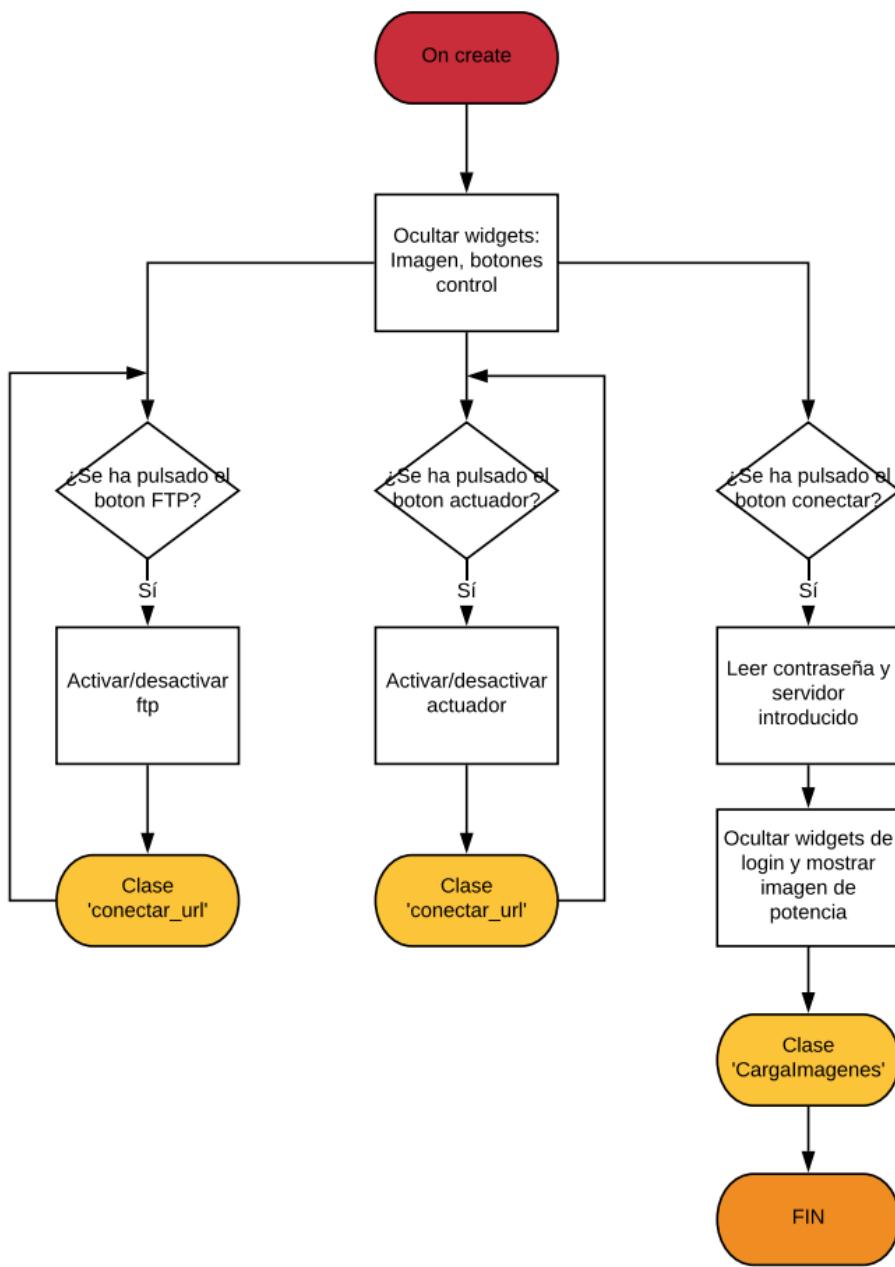


Figura 111: Diagrama de flujo OnCreate(Bundle).

La función OnCreateOptionsMenu se ejecuta al momento en el que el menú desplazable es creado. Detecta qué sección hemos pulsado, asignándole su número a la variable global 'sección'. Dependiendo de nuestra selección, cargará la clase 'CargaImagenes' con la imagen correspondiente, o mostrará los widgets de control (dos TextView y dos botones).

La función OnKeyDown preguntará si queremos salir de la aplicación cuando pulsamos el botón de atrás dentro de la aplicación.

La función actualizar_control() y actualizar_fotos() son los encargados de actualizar periódicamente la gráfica seleccionada, el estado del actuador, el estado del ftp y el estado de la red eléctrica.

La clase ‘CargaImagenes’ se encarga de descargar la imagen del servidor y cargarla en el widget ‘imageview’ correspondiente, todo en segundo plano.

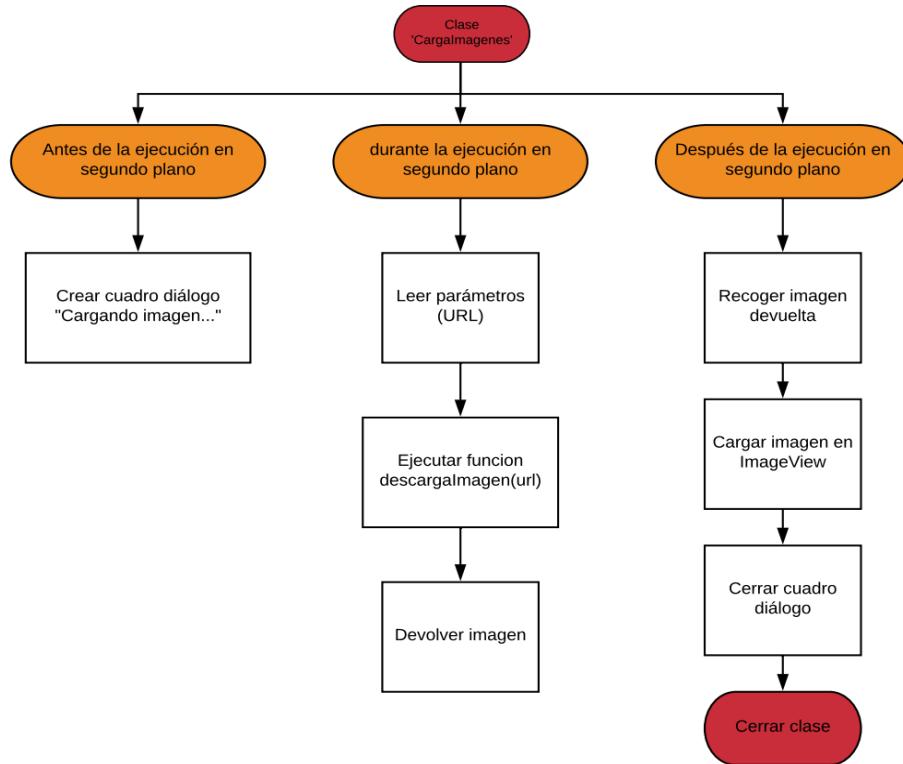


Figura 112: Diagrama de flujo de la clase ‘CargaImagenes’.

La clase ‘CargaImagenes’ se divide en tres acciones: La primera antes de que la ejecución en segundo plano empiece. Aquí es donde se crea un cuadro de diálogo advirtiendo al usuario que se va a cargar una imagen y que la aplicación se va a parar.

Durante la ejecución en segundo plano se leen los parámetros mandados, se ejecuta la función ‘descargaImagen’ (explicada más adelante) y se devuelve el resultado obtenido.

Por último, después de la ejecución en segundo plano, colocamos la imagen devuelta en el widget imageview correspondiente y cerramos el cuadro diálogo de cargando imagen. Seguido de esto la clase se cerrará automáticamente.

La clase ‘conectar_url’ realiza el mismo proceso que ‘CargaImagenes’ pero sin devolver ninguna salida después de la ejecución en segundo plano. Simplemente se conecta a la dirección, suficiente para que el servidor coja los parámetros por url.

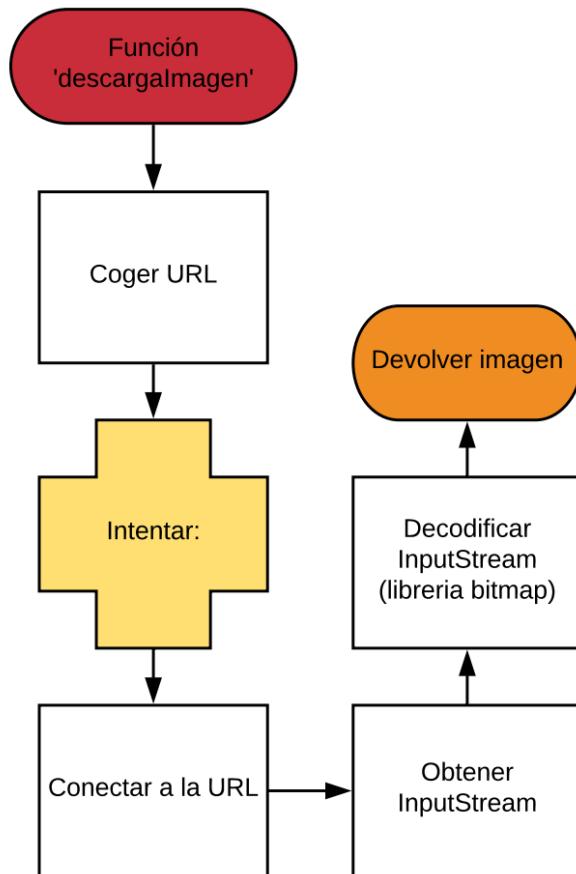


Figura 113: Diagrama de flujo de la función 'descargaImagen'.

La función 'descargaImagen' recoge la url que le han mandado e intenta conectarse a ella. En caso de no obtener un error en la conexión, obtiene el InputStream de la conexión, y lo decodifica usando la librería bitmap, devolviendo la imagen a la que se ha accedido en memoria RAM. La misma no será guardada en la memoria interna del teléfono. En caso de obtener algún error debido a que el servidor no se encuentra disponible, o el teléfono no tiene conexión a internet, la función se cierra y no devuelve nada.

Las clases 'lectura_estado', 'lectura_actuador' y 'lectura_ftp' realizan el mismo proceso que 'CargaImagenes', pero decodificando el inputStream en un String en vez de en una imagen, y colocando la salida correspondiente en su respectivo TextView. En el caso de lectura_estado, también activa una alarma pop-up en el caso de que la red haya fallado.

7. Montaje

Debido a la limitación de componentes electrónicos disponibles en la Universidad, se ha realizado el siguiente montaje en el laboratorio:

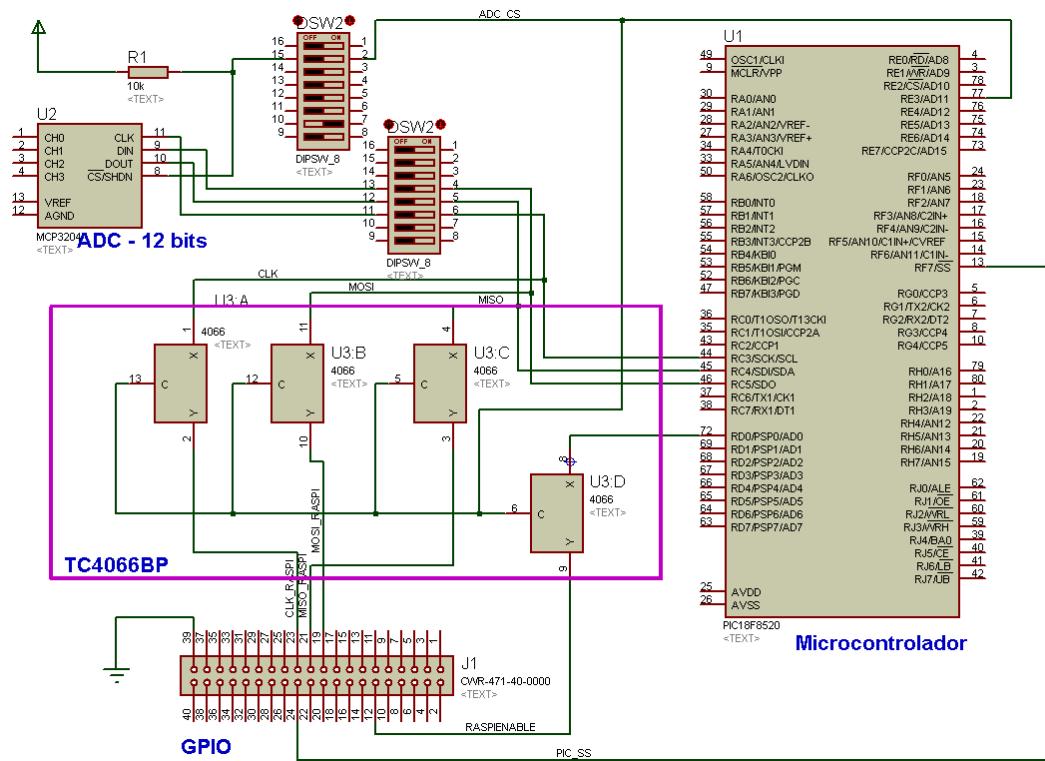


Figura 114: Montaje para la captación de datos.

Como se puede apreciar, el circuito está simplificado respecto al circuito hardware diseñado anteriormente, sin embargo, es suficiente para realizar las pruebas en el laboratorio y comprobar la viabilidad del proyecto.

7.1 Monitorización de señales

El montaje se ha dispuesto para medir dos señales analógicas a través del canal CH0 y CH1 del ADC MCP3204. Estas señales serán generadas por dos generadores de onda AFG 3021B a las cuales se les va a aplicar un factor x100 por software:



Figura 115: Generación de la señal de voltaje.

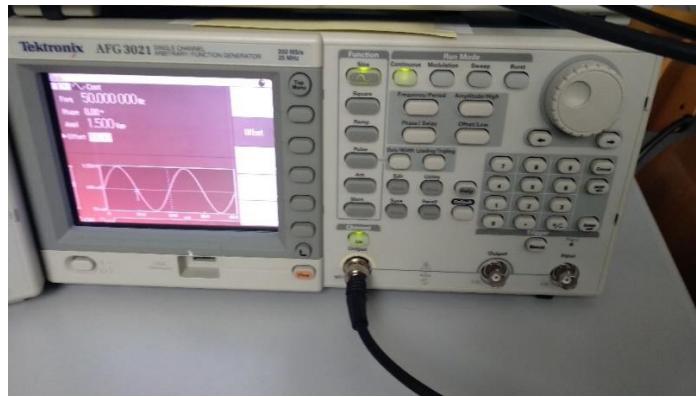


Figura 116: Generación de la señal de intensidad.

A su vez, estas dos señales de control son monitorizadas a través de un osciloscopio de dos bandas:

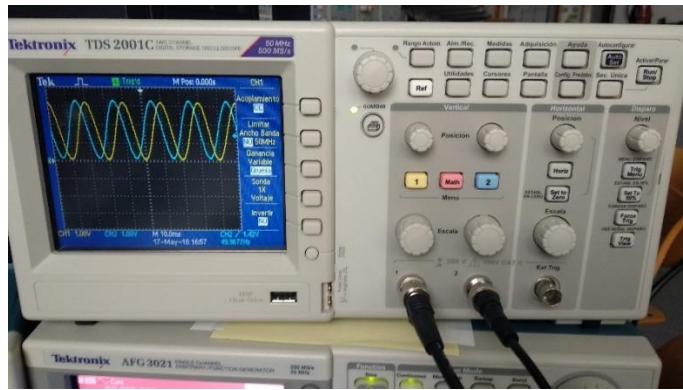


Figura 117: Lectura y superposición de las señales de voltaje e intensidad.

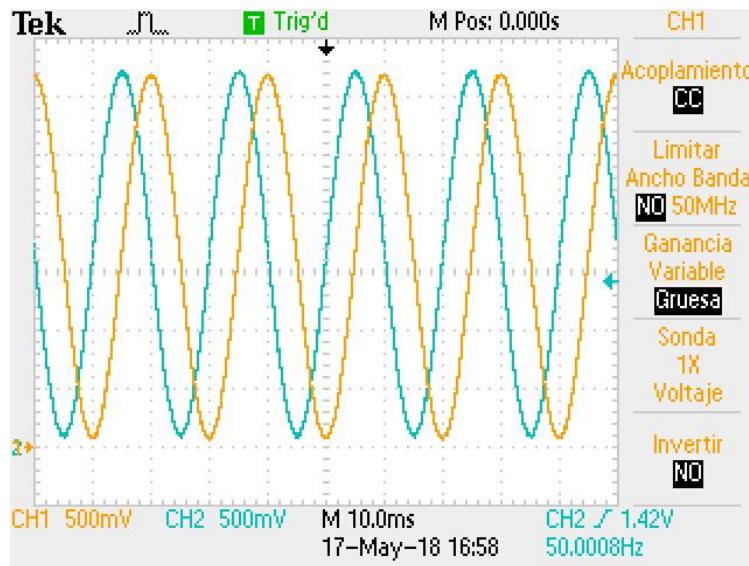


Figura 118: Lectura y superposición de las señales de voltaje e intensidad.

Esta configuración permite comprobar que los resultados que devuelve el sistema son acordes con las ondas introducidas.

7.2 Placa UNI-DS3

Para la implementación del sistema en la universidad, la experimentación y las pruebas previas al ensamblado final del producto, se ha utilizado la placa UNI-DS3.

Ésta es una placa entrenadora altamente configurable, la cual dispone de una alta gama de microcontroladores a utilizar, estando entre ellos el microcontrolador que vamos a utilizar: PIC18F8520.

También dispone de multitud de periféricos que ayudan al prototipado de nuestro sistema: Para una implementación básica en el laboratorio se va a utilizar el ADC de 12 bits accesible mediante SPI que viene incluido en la UNI-DS3: MCP3204.

Finalmente se ha elegido esta placa debido a su facilidad de uso y el software disponible para el uso y programación propia del microcontrolador, centrándonos en la tarea del desarrollo software del sistema y la comunicación del sistema con la raspberry pi.



Figura 119: Placa UNI DS3.

7.3 MCP3204 – ADC 12 bits

Para la prueba y el testeo del sistema, se ha utilizado el MCP3204 incorporado en la placa UNI-DS3, para ello se tendrá que conocer el circuito interno de la UNI-DS3, mostrado en la figura 129:

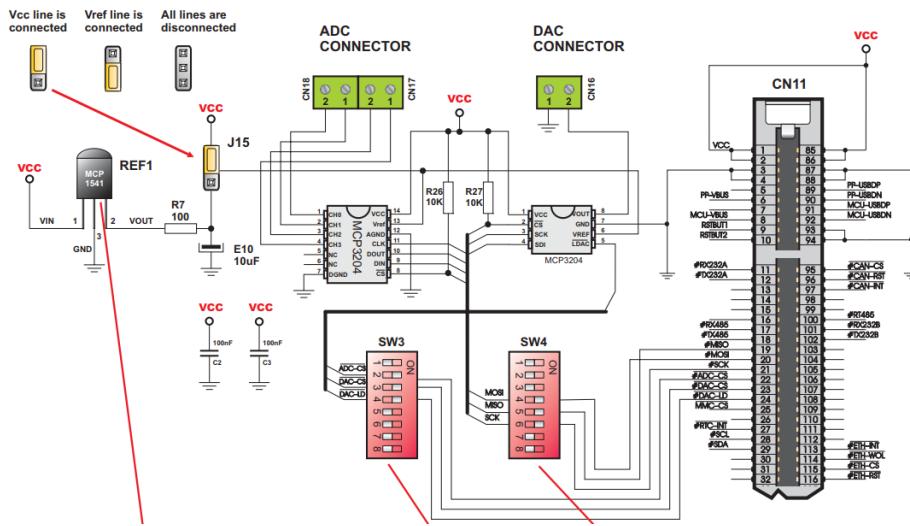


Figura 120: Conexión del MSP3204.

El conexionado es sencillo debido a que ya viene implementado en la placa entrenadora, sin embargo, se tendrá que conocer la metodología de trabajo del circuito integrado. Para ello se puede revisar el propio datasheet, donde se muestra el cronograma de la figura 130:

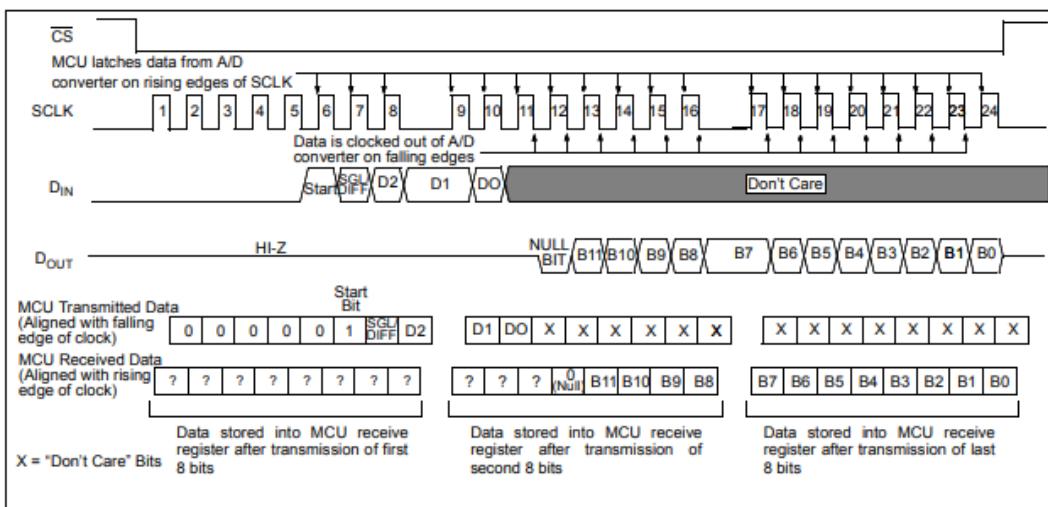


Figura 121: Cronograma de uso del ADC MSP3204.

No se ha tratado en profundidad el uso del MSP3204 debido a que el objeto de este trabajo es el uso del ADE7912, el cual ya se ha explicado con anterioridad. Sin embargo, Este dispositivo está incluido en el ANEXO 1 de hojas técnicas.

7.4 Pantalla táctil

La pantalla táctil de 7 pulgadas posee una resolución de 800x480 píxeles y posee hasta 10 puntos táctiles simultáneos. Se alimenta a 5V y se conecta a la raspberry pi a través del puerto DSI especial para este uso. La alimentación puede ser cogida directamente del GPIO de la raspberry y la pantalla se ha puesto en una base de metacrilato diseñada especialmente para tal fin en la universidad:



Figura 122: Pantalla táctil conectada a la raspberry pi.

7.5 Fotos del montaje final

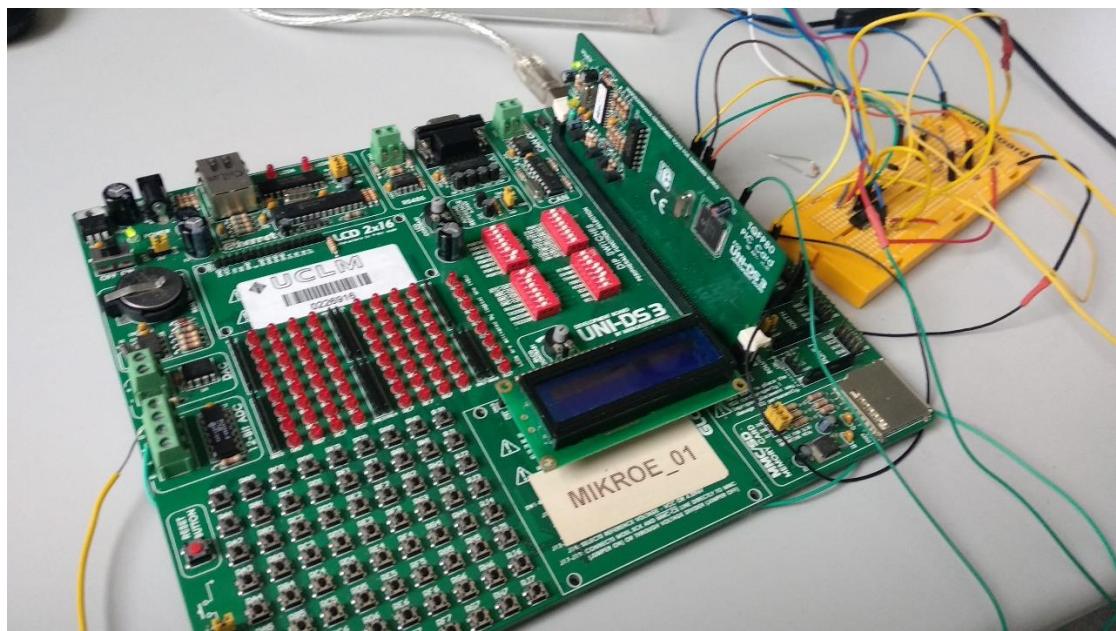


Figura 123: UNI-DS3 con las conexiones pertinentes.

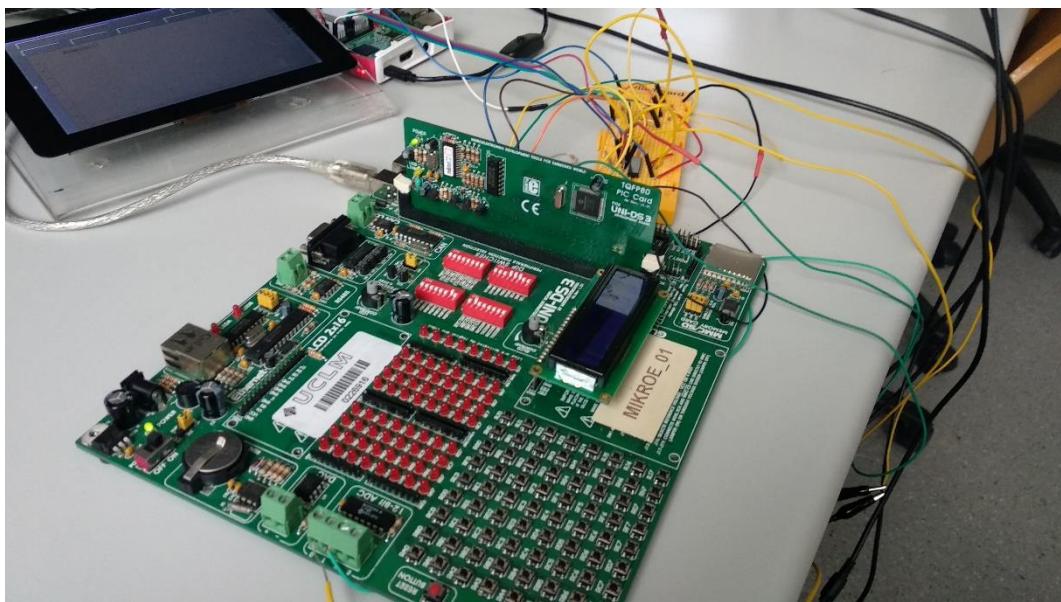


Figura 124: UNI-DS3 con las conexiones pertinentes.



Figura 125: Pantalla táctil con la interfaz gráfica en ella.

CONCLUSIONES Y TRABAJOS FUTUROS

Al término de este Trabajo Fin de Grado, se recogen una serie de conclusiones descritas a continuación:

En primer lugar, se comprueba cómo un alumno que ha superado con éxito todas las asignaturas que este grado propone es capaz de identificar una necesidad, o una posible mejora en un determinado proceso (en este caso, en una instalación eléctrica), y aplicar los conocimientos no sólo aprendidos a lo largo de su etapa estudiantil, sino aprendiendo e investigando nuevos campos de la ingeniería y del conocimiento para resolver el problema de la mejor forma posible. En este caso específico, se han utilizado conocimientos de tres campos distintos de la ingeniería: informática, electrónica y electricidad para el diseño integral de una solución IIOT (Industrial Internet Of Things). A pesar de los recursos limitados con los que se cuentan en la Escuela de Ingenieros Industriales de Albacete, ha sido posible diseñar el sistema de forma teórica y de una forma práctica, quedando así demostrado la viabilidad del proyecto llevado a cabo.

Se ha podido comprobar también cómo las nuevas tecnologías tienden a convertirse en sistemas autónomos en un solo encapsulado, siendo prácticamente una pérdida de tiempo diseñar por uno mismo una solución específica. Esto se ha podido ver claramente con el encapsulado ADE7912, donde podemos ver que la industria ha diseñado un circuito integrado que recoge una señal analógica, la convierte a señal digital junto con una barrera de separación galvánica y habilitando al ingeniero su uso en una alta gama de microcontroladores disponibles. Esto, además, muestra el interés de los fabricantes por impulsar el IIOT/IOT y la monitorización eléctrica, cada vez más presente en hogares y fábricas. Sobre todo con la llegada de las Smart Cities, la revolución de los coches eléctricos y las energías renovables, un tema que daría para otro Trabajo Fin de Grado completo.

Por último, se ha demostrado el potencial del software libre y tener una comunidad detrás de ella, habiendo usado para este proyecto numerosos software con licencias de este estilo: Python, PHP, Android, Raspbian, Apache... A parte de haber encontrado ayuda y tutoriales de aprendizaje de forma desinteresada en sitios web como StackOverFlow y GitHub. Por este mismo motivo, y para fomentar la investigación, aprendizaje y mejora de este proyecto, se ha licenciado este TFG bajo una licencia Creative Commons.

En cuanto a trabajos futuros se recogen varias líneas de mejora:

- Fabricación de las placas PCB diseñadas. Debido a no tener la tecnología de fabricación de PCB disponible, no se han podido fabricar las placas de circuito impreso, por ello se ha dejado el desarrollo teórico de las mismas para ser aplicado en un futuro.
- Implementación de red GSM para el envío de los datos a través de internet de una forma totalmente autónoma e inalámbrica, permitiendo así colocar el sistema en fábricas o ubicaciones de difícil acceso o sin conexión a internet.
- Seguridad: A pesar de haber instalado una contraseña de acceso a ciertas áreas de la página web, un punto a mejorar es la seguridad del servidor, ya que de preparar el servidor para ser accesible mediante internet (con la ip externa de nuestra instalación) puede ser fácilmente controlada por algún Cracker.
- Dotar al software de mayor adaptación y configurabilidad: En este trabajo no es posible, pero un sistema de alertas o accionadores automáticos podría mejorar mucho el sistema. Por ejemplo: Cuando la tensión de V11 baje de 180V, desactivar el actuador1 de forma automática, y activar el actuador2.
- Análisis de armónicos: Aunque se poseen los datos sin procesar necesarios para analizar el espectro frecuencial de nuestra instalación usando un software específico, se podría haber implementado una FFT en algún lenguaje de programación.
- Realización de cajón: Modelar un cajón o caja donde incluir toda la electrónica del sistema y poder colocarla en la pared junto con el cuadro eléctrico monitorizado.
- Detección de errores y fallos: El sistema actual sólo posee un sistema de detección de corte en la red eléctrica. Se pueden programar la detección de múltiples errores y problemas: Desequilibrios, bajadas de tensión, huecos de tensión, sobretensiones, armónicos, etc.

BIBLIOGRAFÍA

1. PIC 18F8520 Datasheet: Microchip Technology.
2. ADE7912 Datasheet: Analog Devices.
3. Relé de estado sólido G3PH Datasheet: Omron.
4. TC4066BP Datasheet: Toshiba
5. Resistencia Shunt Datasheet: Vishay.
6. DM7404 Datasheet: Fairchild.
7. BSS138 Datasheet: Infineon.
8. Uni-DS3 manual de usuario: MikroElectronika.
9. MSP3204 Datasheet: Microchip Technology.
10. David Lineweber y Shawn Mcnulty. (2001).The Cost of Power Disturbances to Industrial & Digital Economy Companies, *Primen*.
11. Jose Salas. 2013. Comunicación entre dos pícs maestro-esclavo mediante SPI. Link: <http://todolectrodo.blogspot.com.es/2013/03/comunicacion-entre-2-pics-maestro.html>
12. WiringPi Library. Link: <http://wiringpi.com/>
13. Raspberry Pi Foundation. Link: <https://www.raspberrypi.org/>
14. Python Software Foundation. Link: <https://www.python.org/>
15. Tkinter library. Link: <https://wiki.python.org/moin/TkInter>
16. Joan Ribas Lequerica. 2013. Desarrollo de aplicaciones para android, *Anaya*
17. David Sastre. 2012. Descargar imagen desde URL en android. Link: <https://sekthdroid.wordpress.com/2012/11/29/descargar-imagen-desde-url-en-android/>
18. Pchart library. Link: <http://www.pchart.net/>
19. Rubén García Segovia. MakingBytes. Link: <https://github.com/rubenelportero/MakingBytes>
20. AENOR. Características de la tensión suministrada por las redes generales de distribución. UNE-EN 50160, 2015.

ANEXOS

1. Hojas de características

1.1 Resistencia Shunt Vishay

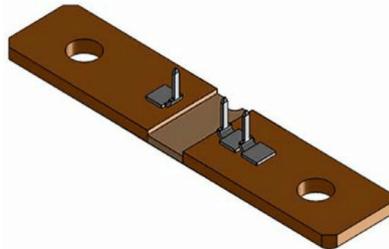


www.vishay.com

WSBS8518...40

Vishay Dale

Power Metal Strip® Shunt Resistor With Three Sense Pins, Very Low Value (50 $\mu\Omega$, 100 $\mu\Omega$, and 125 $\mu\Omega$)



DESIGN TOOLS (click logo to get started)



FEATURES

- High power to resistor size ratio
- Sense pins allow for consistent contact location
- Proprietary processing technique produces extremely low resistance values
- Welded terminal to element construction
- Solid metal manganese-copper alloy resistive element with low TCR (< 20 ppm/ $^{\circ}\text{C}$)
- Very low inductance (< 5 nH)
- Low thermal EMF (< 1 $\mu\text{V}/^{\circ}\text{C}$ available)
- Material categorization: for definitions of compliance please see www.vishay.com/doc?99912



STANDARD ELECTRICAL SPECIFICATIONS

GLOBAL MODEL	SIZE	POWER RATING $P_{70^{\circ}\text{C}}$ W	TOLERANCE $\pm \%$	RESISTANCE VALUE RANGE Ω	RESISTANCE VALUES CURRENTLY AVAILABLE ⁽¹⁾ Ω	WEIGHT (typical) g
WSBS8518...40	8518	36	5, 10	50 μ to 1000 μ	50 μ , 100 μ , 125 μ	50 μ = 38.6, 100 μ / 125 μ = 37.1

Note

⁽¹⁾ Other values may be available, contact factory

TECHNICAL SPECIFICATIONS

PARAMETER	UNIT	RESISTOR CHARACTERISTICS
Temperature coefficient	ppm/ $^{\circ}\text{C}$	± 200 for 50 $\mu\Omega$
		± 175 for 100 $\mu\Omega$ / 125 $\mu\Omega$
Temperature coefficient (element material)	ppm/ $^{\circ}\text{C}$	± 20
Thermal EMF	$\mu\text{V}/^{\circ}\text{C}$	< 1 for 50 $\mu\Omega$ and < 3 for 100 $\mu\Omega$, 125 $\mu\Omega$
Inductance	nH	< 5
Operating temperature range	$^{\circ}\text{C}$	-65 to +170
Maximum current rating	A	$(P/R)^{1/2}$

GLOBAL PART NUMBER INFORMATION

GLOBAL PART NUMBERING: WSBS8518L1000JT40 (WSBS8518...40, 0.000100 Ω , $\pm 5 \%$, tray pack)

W	S	B	S	8	5	1	8	L	1	0	0	0	J	T	4	0
GLOBAL MODEL				RESISTANCE VALUE				TOLERANCE CODE				PACKAGING CODE				SPECIAL
WSBS8518				$L = \text{m}\Omega$ L0500 = 0.000050 Ω L1000 = 0.000100 Ω L1250 = 0.000125 Ω				$J = \pm 5 \%$ $K = \pm 10 \%$				K = bulk pack T = tray pack				40 = three sense pins attached

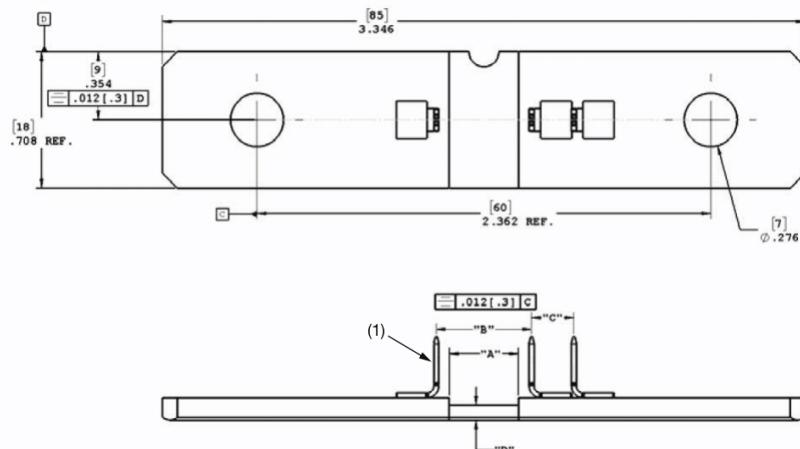


www.vishay.com

WSBS8518...40

Vishay Dale

DIMENSIONS in inches (millimeters)



**RESISTANCE
VALUE ($\mu\Omega$)**

**ELEMENT
MATERIAL**

**A
REFERENCE**

**B
 $\pm 0.005 [\pm 0.13]$**

**C
 $\pm 0.005 [\pm 0.13]$**

**D
 $\pm 0.002 [\pm 0.05]$**

50 Mn-Cu 0.145 [3.68] 0.135 [3.43] 0.220 [5.59] 0.079 [2.00]

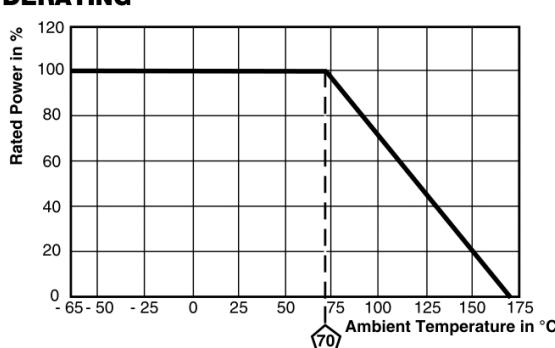
100 Mn-Cu 0.370 [9.40] 0.495 [12.57] 0.220 [5.59] 0.079 [2.00]

125 Mn-Cu 0.480 [12.19] 0.585 [14.86] 0.220 [5.59] 0.079 [2.00]

Note

(1) Minimum pull strength of 200 N

DERATING



TOLERANCES ON DECIMALS
.xxx ± 0.005 [.x ± 0.1]

UNLESS OTHERWISE LISTED

PERFORMANCE

TEST	CONDITIONS OF TEST	TEST LIMITS
Thermal shock	-55 °C to +150 °C, 1000 cycles, 15 min at each extreme	$\pm 0.5 \%$ ΔR
Short time overload	5x rated power for 5 s	$\pm 0.5 \%$ ΔR
Low temperature storage	-65 °C for 24 h	$\pm 0.5 \%$ ΔR
High temperature exposure	1000 h at +170 °C	$\pm 1.0 \%$ ΔR
Bias humidity	+85 °C, 85 % RH, 10 % bias, 1000 h	$\pm 0.5 \%$ ΔR
Mechanical shock	100 g's for 6 ms, 5 pulses	$\pm 0.5 \%$ ΔR
Vibration	Frequency varied 10 Hz to 2000 Hz in 1 min, 3 directions, 12 h	$\pm 0.5 \%$ ΔR
Load life	1000 h at +70 °C, 1.5 h "ON", 0.5 h "OFF"	$\pm 1.0 \%$ ΔR
Moisture resistance	MIL-STD-202, method 106, 0 % power, 7b not required	$\pm 0.5 \%$ ΔR

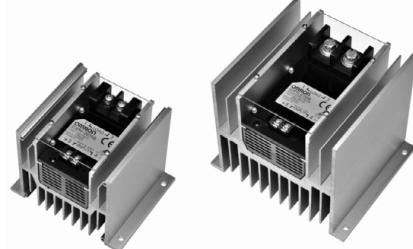
1.2 Relé de estado sólido G3PH-5150B DC5-24

New Product News

OMRON

High-power Solid State Relays G3PH

High-power, Load-control SSRs with High Current of 75 or 150 A and High Voltage of 240 or 480 VAC



Refer to Safety Precautions for All Solid State Relays.

Ordering Information

Solid State Relays

Insulation method	Operation indicator	Zero cross function	Applicable output load*	Rated input voltage	Model
Photocoupler	Yes (yellow)	Yes	75 A, 100 to 240 VAC	5 to 24 VDC	G3PH-2075B DC5-24
			100 to 240 VAC	100 to 240 VAC	G3PH-2075B AC100-240
		No	150 A, 100 to 240 VAC	5 to 24 VDC	G3PH-2150B DC5-24
			100 to 240 VAC	100 to 240 VAC	G3PH-2150B AC100-240
		Yes	75 A, 100 to 240 VAC	5 to 24 VDC	G3PH-2075BL DC5-24
			150 A, 100 to 240 VAC	5 to 24 VDC	G3PH-2150BL DC5-24
		Yes	75 A, 180 to 480 VAC	5 to 24 VDC	G3PH-5075B DC5-24
			100 to 240 VAC	100 to 240 VAC	G3PH-5075B AC100-240
		No	150 A, 180 to 480 VAC	5 to 24 VDC	G3PH-5150B DC5-24
			100 to 240 VAC	100 to 240 VAC	G3PH-5150B AC100-240
		No	75 A, 180 to 480 VAC	5 to 24 VDC	G3PH-5075BL DC5-24
			150 A, 180 to 480 VAC	5 to 24 VDC	G3PH-5150BL DC5-24

Note: The Thyristor Module is built in.

*The applicable output load depends on the ambient temperature. For details, refer to *Load Current vs. Ambient Temperature* in *Engineering Data* on page 2.

Options (Order Separately)

Thyristor Module

Name	Applicable output load*	Applicable models	Model
Thyristor Module	75 A, 75 to 264 VAC	G3PH-2075B(L)	G32A-P2075
	150 A, 75 to 264 VAC	G3PH-2150B(L)	G32A-P2150
	75 A, 150 to 520 VAC	G3PH-5075B(L)	G32A-P5075
	150 A, 150 to 528 VAC	G3PH-5150B(L)	G32A-P5150

*The applicable output load depends on the ambient temperature. For details, refer to *Load Current vs. Ambient Temperature* in *Engineering Data* on page 2.

Specifications

Ratings

Input

Rated voltage	Operating voltage	Impedance (input current)	Voltage level	
			Must operate voltage	Must release voltage
5 to 24 VDC	4 to 30 VDC	(5 mA max.)*	4 VDC max.	1.0 VDC max.
100 to 240 VAC	75 to 264 VAC	41 kΩ ±20%	75 VAC max.	20 VAC max.

*A constant-current circuit is used for the input current to the G3PH.

Output

Model	Item	Applicable load		
		Rated load voltage	Load voltage range	Load current*
G3PH-2075B(L)	100 to 240 VAC	75 to 264 VAC	1 to 75 A (at 40°C)	800 A (60 Hz, 1 cycle)
G3PH-2150B(L)	100 to 240 VAC	75 to 264 VAC	1 to 150 A (at 40°C)	1,800 A (60 Hz, 1 cycle)
G3PH-5075B(L)	180 to 480 VAC	150 to 528 VAC	1 to 75 A (at 40°C)	800 A (60 Hz, 1 cycle)
G3PH-5150B(L)	180 to 480 VAC	150 to 528 VAC	1 to 150 A (at 40°C)	1,800 A (60 Hz, 1 cycle)

*The load current depends on the ambient temperature. For details, refer to *Load Current vs. Ambient Temperature* in *Engineering Data* on page 2.

OMRON

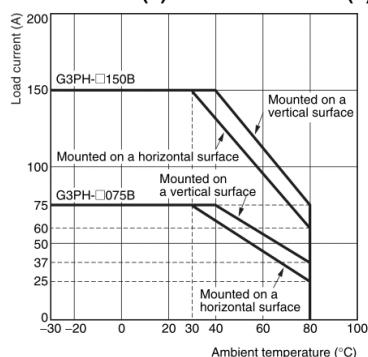
G3PH

Characteristics

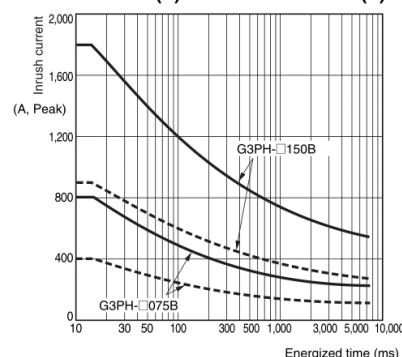
Item	Model	G3PH-2075B	G3PH-2150B	G3PH-5075B	G3PH-5150B	G3PH-2075BL	G3PH-2150BL	G3PH-5075BL	G3PH-5150BL
Operate time		1/2 of load power source cycle + 1 ms max. for DC input 3/2 of load power source cycle + 1 ms max. for AC input				1 ms max.			
Release time		1/2 of load power source cycle + 1 ms max. for DC input 3/2 of load power source cycle + 1 ms max. for AC input				1/2 of load power source cycle + 1 ms max.			
Output ON voltage drop		1.6 V (RMS) max.							
Leakage current		30 mA max. (at 200 VAC)	60 mA max. (at 400 VAC)		30 mA max. (at 200 VAC)		60 mA max. (at 400 VAC)		
Insulation resistance		100 MΩ min. (at 500 VDC)							
Dielectric strength		2,500 VAC, 50/60 Hz for 1 min							
Vibration resistance		10 to 55 to 10 Hz, 0.375-mm single amplitude (0.75-mm double amplitude)							
Shock resistance		500 m/s ²							
Ambient storage temperature		-30 to 100°C (with no icing or condensation)							
Ambient operating temperature		-30 to 80°C (with no icing or condensation)							
Ambient operating humidity		45% to 85%							
Weight		Approx. 1.8 kg	Approx. 3.0 kg	Approx. 1.8 kg	Approx. 3.0 kg	Approx. 1.8 kg	Approx. 3.0 kg	Approx. 1.8 kg	Approx. 3.0 kg

Engineering Data

Load Current vs. Ambient Temperature
G3PH-□075B (L) and G3PH-□150B (L)

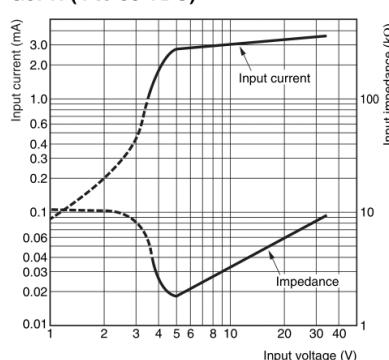


Inrush Current Resistance: Non-repetitive
G3PH-□075B (L) and G3PH-□150B (L)

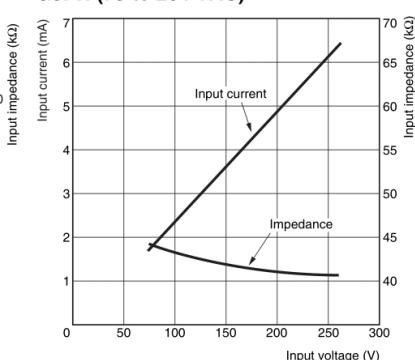


Keep the inrush current to below the inrush current resistance value (i.e., below the broken line) if it occurs repetitively.

Input Voltage vs. Input Impedance
G3PH (4 to 30 VDC)



G3PH (75 to 264 VAC)



1.3 TC4066BP

TOSHIBA

TC4066BP/BF/BFT

TOSHIBA CMOS Digital Integrated Circuit Silicon Monolithic

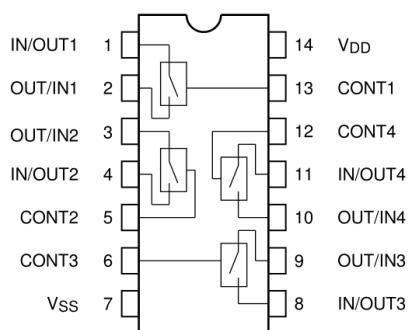
TC4066BP, TC4066BF, TC4066BFT

Quad Bilateral Switch

TC4066B contains four independent circuits of bidirectional switches. When control input CONT is set to "H" level, the impedance between input and output of the switch becomes low and when it is set to "L" level, the impedance becomes high. This can be applied for switching of analog signals and digital signals.

- ON-resistance, R_{on}
 - 250 Ω (typ.) : $V_{DD} - V_{SS} = 5$ V
 - 110 Ω (typ.) : $V_{DD} - V_{SS} = 10$ V
 - 70 Ω (typ.) : $V_{DD} - V_{SS} = 15$ V
- OFF-resistance, R_{off}
 - R_{off} (typ.) > $10^9 \Omega$

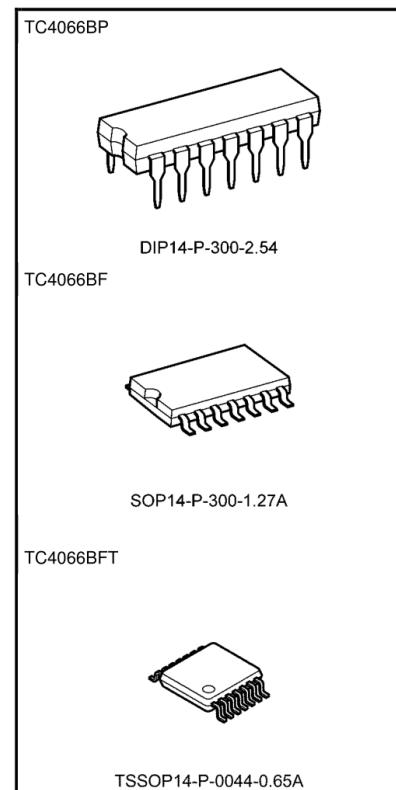
Pin Assignment (top view)



Truth Table

Control	Impedance between IN/OUT-OUT/IN (Note 1)
H	0.5 to $5 \times 10^2 \Omega$
L	> $10^9 \Omega$

Note 1: See static electrical characteristics

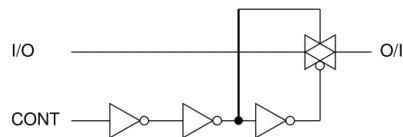


Weight
 DIP14-P-300-2.54 : 0.96 g (typ.)
 SOP14-P-300-1.27A : 0.18 g (typ.)
 TSSOP14-P-0044-0.65A : 0.06 g (typ.)

Start of commercial production
1978-09

TOSHIBA**TC4066BP/BF/BFT****Logic Diagram**

1/4 TC4066B

**Absolute Maximum Ratings**

Characteristics	Symbol	Rating	Unit
DC supply voltage	V _{DD}	V _{SS} - 0.5 to V _{SS} + 20	V
Control input voltage	V _{CIN}	V _{SS} - 0.5 to V _{DD} + 0.5	V
Switch I/O voltage	V _I /V _O	V _{SS} - 0.5 to V _{DD} + 0.5	V
Power dissipation	P _D	300 (DIP)/180 (SOP/TSSOP)	mW
Potential difference across I/O during ON	V _I - V _O	±0.5	V
Control input current	I _{CIN}	±10	mA
Operating temperature range	T _{opr}	-40 to 85	°C
Storage temperature range	T _{stg}	-65 to 150	°C

Note: Exceeding any of the absolute maximum ratings, even briefly, lead to deterioration in IC performance or even destruction.

Using continuously under heavy loads (e.g. the application of high temperature/current/voltage and the significant change in temperature, etc.) may cause this product to decrease in the reliability significantly even if the operating conditions (i.e. operating temperature/current/voltage, etc.) are within the absolute maximum ratings and the operating ranges.

Please design the appropriate reliability upon reviewing the Toshiba Semiconductor Reliability Handbook ("Handling Precautions"/"Derating Concept and Methods") and individual reliability data (i.e. reliability test report and estimated failure rate, etc.).

Operating Ranges (V_{SS} = 0 V)

Characteristics	Symbol	Test Condition	Min	Typ.	Max	Unit
DC supply voltage	V _{DD}	—	3	—	18	V
Input/Output voltage	V _{IN} /V _{OUT}	—	0	—	V _{DD}	V

Note: The operating ranges must be maintained to ensure the normal operation of the device.
Unused control inputs must be tied to either V_{DD} or V_{SS}.

TOSHIBA**TC4066BP/BF/BFT****Electrical Characteristics (V_{SS} = 0 V, unless specified otherwise)**

Characteristics	Symbol	Test Condition	V _{DD} (V)	-40°C		25°C			85°C		Unit
				Min	Max	Min	Typ.	Max	Min	Max	
Control input high voltage	V _{IH}	I _S = 10 µA	5	3.5	—	3.5	2.75	—	3.5	—	V
			10	7.0	—	7.0	5.50	—	7.0	—	
			15	11.0	—	11.0	8.25	—	11.0	—	
Control input low voltage	V _{IL}	I _S = 10 µA	5	—	1.5	—	2.25	1.5	—	1.5	V
			10	—	3.0	—	4.50	3.0	—	3.0	
			15	—	4.0	—	6.75	4.0	—	4.0	
On-state resistance	R _{ON}	0 ≤ V _S ≤ V _{DD} R _L = 10 kΩ	5	—	800	—	290	950	—	1200	Ω
			10	—	210	—	120	250	—	300	
			15	—	140	—	85	160	—	200	
ΔOn-state resistance (between any 2 switches)	R _{ONΔ}	—	5	—	—	—	10	—	—	—	Ω
			10	—	—	—	6	—	—	—	
			15	—	—	—	4	—	—	—	
Input/output leakage current	I _{OFF}	V _{IN} = 18 V, V _{OUT} = 0 V V _{IN} = 0 V, V _{OUT} = 18 V	18	—	±100	—	±0.1	±100	—	±1000	nA
			18	—	±100	—	±0.1	±100	—	±1000	
Quiescent supply current	I _{DD}	V _{IN} = V _{SS} , V _{DD} (Note 1)	5	—	0.25	—	0.001	0.25	—	7.5	µA
			10	—	0.50	—	0.001	0.50	—	15.0	
			15	—	1.00	—	0.002	1.00	—	30.0	
Control Input current "H" level	I _{IH}	V _{IH} = 18 V	18	—	0.1	—	10 ⁻⁵	0.1	—	1.0	µA
	I _{IL}	V _{IL} = 0 V	18	—	-0.1	—	-10 ⁻⁵	-0.1	—	-1.0	

Note 1: All valid input combinations.

TOSHIBA**TC4066BP/BF/BFT****Switching Characteristics (Ta = 25°C)**

Characteristics	Symbol	Test Condition	V _{SS} (V)	V _{DD} (V)	Min	Typ.	Max	Unit
			0	5				
Phase difference between input to output	φ _{I-O}	C _L = 50 pF	0	5	—	15	40	ns
			0	10	—	8	20	
			0	15	—	5	15	
Propagation delay time (control-OUT)	t _{pZL} t _{pZH}	R _L = 1 kΩ C _L = 50 pF	0	5	—	55	120	ns
			0	10	—	25	40	
			0	15	—	20	30	
Propagation delay time (control -OUT)	t _{pLZ} t _{pHZ}	R _L = 1 kΩ C _L = 50 pF	0	5	—	45	80	ns
			0	10	—	30	70	
			0	15	—	25	60	
Max control input repetition rate	f _{max} (C)	R _L = 1 kΩ C _L = 50 pF	0	5	—	10	—	MHz
			0	10	—	12	—	
			0	15	—	12	—	
-3dB cutoff frequency	f _{max} (I-O)	R _L = 1 kΩ C _L = 15 pF (Note 1)	-5	5	—	30	—	MHz
Total harmonic distortion	—	R _L = 10 kΩ f = 1 kHz (Note 2)	-5	5	—	0.03	—	%
-50dB feed through frequency	—	R _L = 1 kΩ (Note 3)	-5	5	—	600	—	kHz
-50dB crosstalk frequency	—	R _L = 1 kΩ (Note 4)	-5	5	—	1	—	MHz
Crosstalk (control-OUT)	—	R _{IN} = 1 kΩ R _{OUT} = 10 kΩ C _L = 15 pF	0	5	—	200	—	mV
Input capacitance	C _{IN}	Control input	—	5	—	7.5	—	
		Switch I/O	—	10	—	—	—	
Feed through capacitance	C _{IN-OUT}	—	—	—	0.5	—	—	pF

Note 1: Sine wave of ± 2.5 V_{p-p} shall be used for V_{IS} and the frequency of $20 \log 10 \frac{V_{OS}}{V_{IS}}$ = -3 dB shall be f_{max}.

Note 2: V_{IS} shall be sine wave of ± 2.5 V_{p-p}

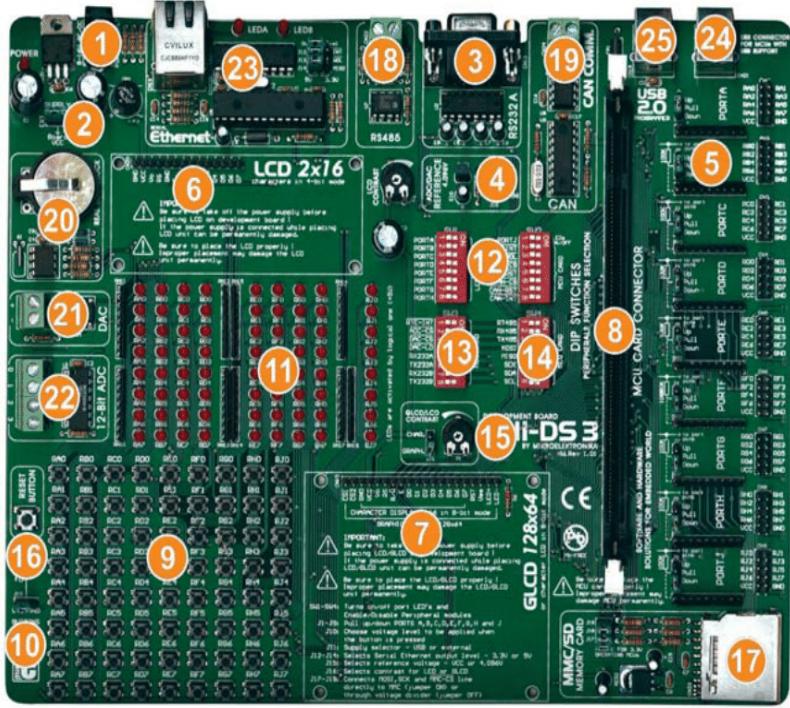
Note 3: Sine wave of ± 2.5 V_{p-p} shall be used for V_{IS} and the frequency of $20 \log 10 \frac{V_{OS}}{V_{IS}}$ = -50 dB shall be feed-through.

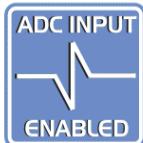
Note 4: Sine wave of ± 2.5 V_{p-p} shall be used for V_{IS} and the frequency of $20 \log 10 \frac{V_{OS}}{V_{IS}}$ = -50 dB shall be crosstalk.

1.4 UNI-DS3

UNI-DS3 KEY FEATURES

1. External power supply from 8 to 16 V AC/DC;
2. Choose between external and USB power supply. You don't need an external supply if you choose powering from PC's USB port ;
3. RS232 communication port ;
4. 4.096V voltage reference is used for working with A/D converter ;
5. If you set jumper to the upper position the pins of appropriate port are set to logical one (pull-up). If you set jumper to the lower position, the pins are set to logical zero (pull-down). It is very important to select pull-up for the port if you expect logical zero on it's inputs and vice versa ;
6. You can connect LCD if you need it for your application in 4-bit mode ;
7. You can connect Graphic LCD if you need it for your application or LCD in 8-bit mode ;
8. MCU Card socket ;
9. 72 buttons enable you to control every pin on your microcontroller ;
10. You can choose how to affect a pin by pressing button, high state or low state ;
11. See all the signals - each pin has an LED ;
12. All switches on SW1 and switch 1 on SW2 are used to turn LEDs on all MCU ports ON or OFF. Switches 2, 3, 4 and 5 on SW2 are used to enable Serial Ethernet and switches 6, 7 and 8 are used to enable CAN communication ;
13. Switch 1 on SW3 enables Real Time Clock Interrupt. Switches 2, 3 and 4 on SW2 are used to enable A/D and D/A modules. Switches 5, 6, 7, and 8 on SW3 are used to enable RS232 communication ;
14. Switches 1, 2 and 3 on SW2 are used to enable RS485 communication, switches 4, 5 and 6 to enable SPI communication lines and switches 7 and 8 to enable Real Time Clock ;
15. Set LCD contrast according to your display characteristics ;
16. Reset circuit - if the reset button is pressed a hardware reset will happen (MCU will start executing from the beginning) ;
17. MMC/SD slot for multimedia cards with storage space up to 2GB;
18. RS485 communication port ;
19. CAN communication port ;
20. Real Time Clock ;
21. D/A converter output ;
22. A/D converter input ;
23. Serial Ethernet on board ;
24. USB connector for MCUs with USB support ;
25. USB connector for USB 2.0 programmer ;





A/D CONVERTER INPUT

Analog-to-digital converter is a semiconductor device used to convert an analog signal into a digital code. In the real world, most of the signals sensed and processed by humans are analog signals. Analog-to-digital conversion is a primary means by which analog signals are converted into digital data which can be processed by computers for various purposes. Analog signal is a signal that may assume any value within a continuous range. Device used to convert an analog signal into an analog voltage or current is known as a transducer. The analog-to-digital converter is used to translate further this analog voltage or current into digital codes that consist of 1's and 0's.

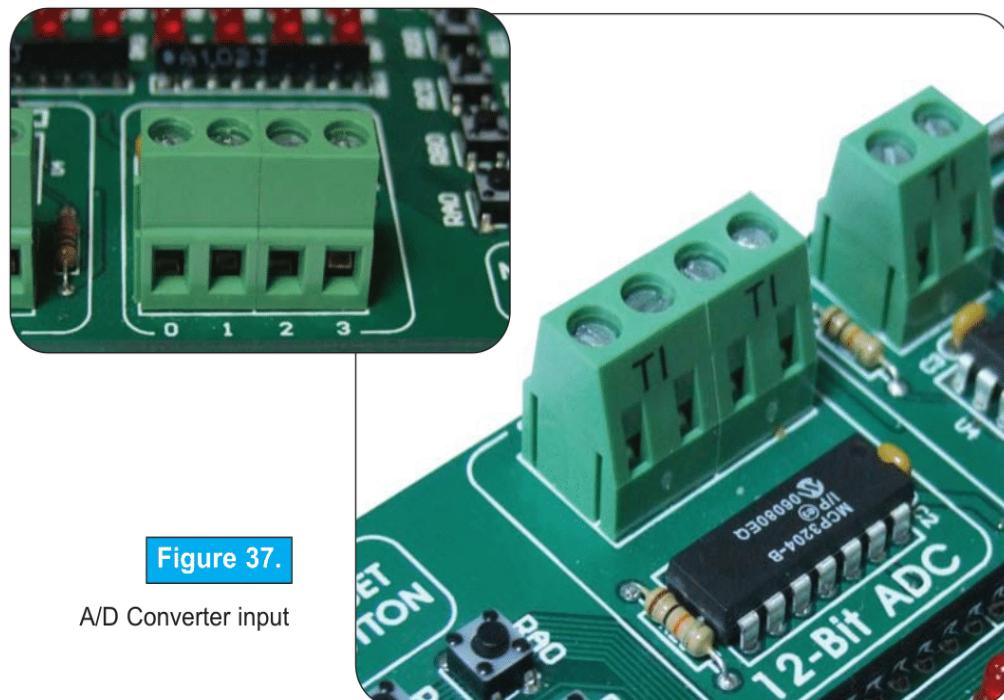


Figure 37.

A/D Converter input

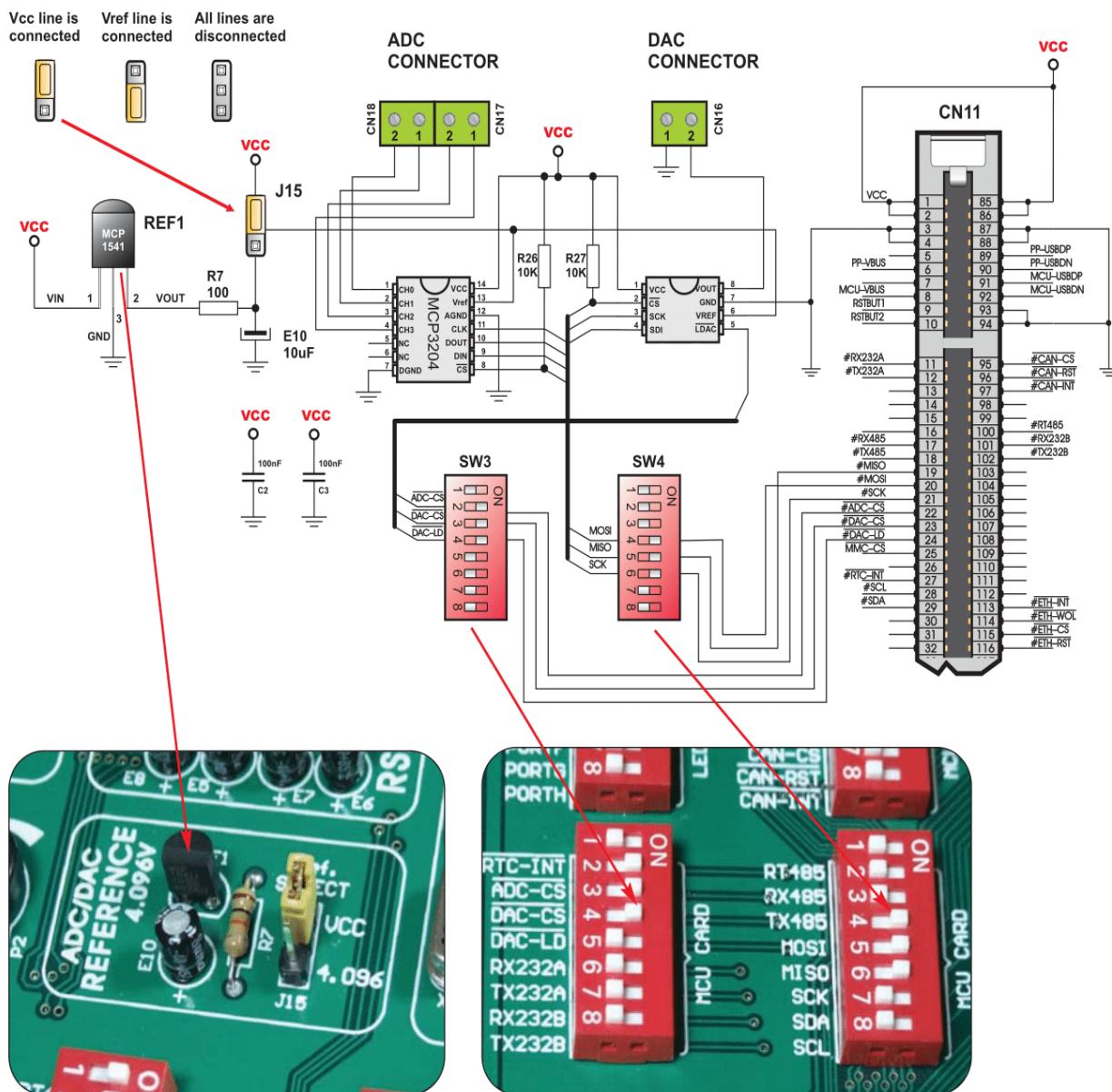
UNI-DS2 development board has 12-bit, 4 input A/D converter MCP3204 with serial interface. As shown in Figure 39, switches on **SW3** and **SW4** are used to enable communication between microcontroller and AD converter device. Switch 2 on **SW3** is used for AD Chip Select (AD-CS), and switches 4, 5 and 6 on **SW3** have to be turned on in order to enable SPI communication between ADC device and microcontroller.

UNI-DS3 User's Manual

MIKROELEKTRONIKA
DEVELOPMENT TOOLS

A/D-D/A CONVERTERS SCHEMATIC

Figure 39. Converters schematically



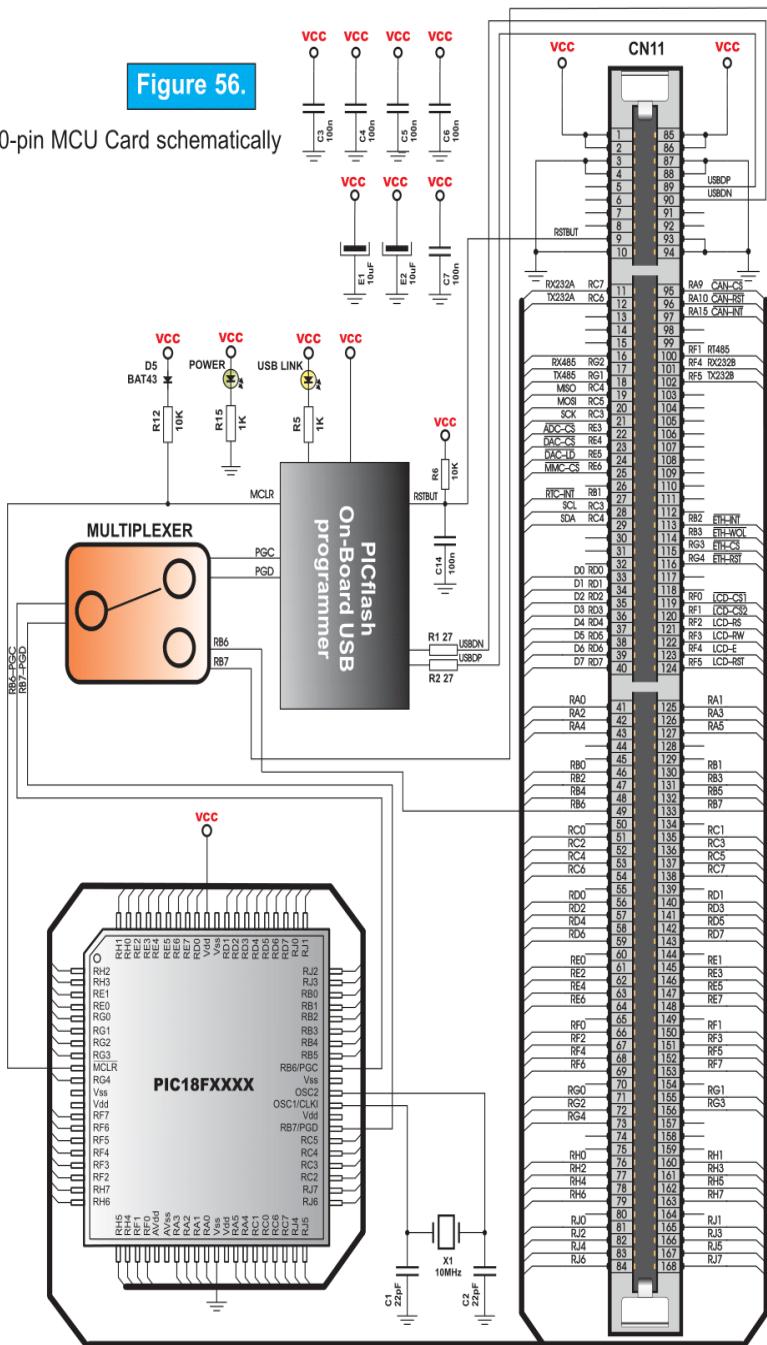
UNI-DS3 User's Manual

MIKROELEKTRONIKA
DEVELOPMENT TOOLS

PIC 80-PIN MCU CARD

Figure 56.

PIC 80-pin MCU Card schematically



page

56

MIKROELEKTRONIKA SOFTWARE AND HARDWARE SOLUTIONS FOR THE EMBEDDED WORLD

UNI-DS3

with Serial Ethernet

1.5 MCP3204



MCP3204/3208

2.7V 4-Channel/8-Channel 12-Bit A/D Converters with SPI™ Serial Interface

Features

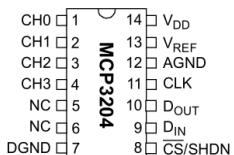
- 12-bit resolution
- ± 1 LSB max DNL
- ± 1 LSB max INL (MCP3204/3208-B)
- ± 2 LSB max INL (MCP3204/3208-C)
- 4 (MCP3204) or 8 (MCP3208) input channels
- Analog inputs programmable as single-ended or pseudo-differential pairs
- On-chip sample and hold
- SPI serial interface (modes 0,0 and 1,1)
- Single supply operation: 2.7V - 5.5V
- 100 ksps max. sampling rate at $V_{DD} = 5V$
- 50 ksps max. sampling rate at $V_{DD} = 2.7V$
- Low power CMOS technology:
 - 500 nA typical standby current, 2 μ A max.
 - 400 μ A max. active current at 5V
- Industrial temp range: -40°C to +85°C
- Available in PDIP, SOIC and TSSOP packages

Applications

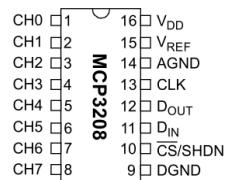
- Sensor Interface
- Process Control
- Data Acquisition
- Battery Operated Systems

Package Types

PDIP, SOIC, TSSOP



PDIP, SOIC

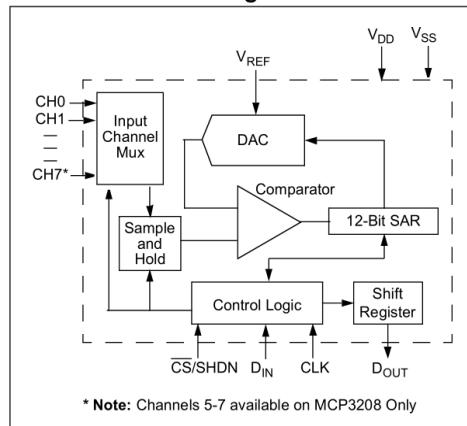


Description

The Microchip Technology Inc. MCP3204/3208 devices are successive approximation 12-bit Analog-to-Digital (A/D) Converters with on-board sample and hold circuitry. The MCP3204 is programmable to provide two pseudo-differential input pairs or four single-ended inputs. The MCP3208 is programmable to provide four pseudo-differential input pairs or eight single-ended inputs. Differential Nonlinearity (DNL) is specified at ± 1 LSB, while Integral Nonlinearity (INL) is offered in ± 1 LSB (MCP3204/3208-B) and ± 2 LSB (MCP3204/3208-C) versions.

Communication with the devices is accomplished using a simple serial interface compatible with the SPI protocol. The devices are capable of conversion rates of up to 100 ksps. The MCP3204/3208 devices operate over a broad voltage range (2.7V - 5.5V). Low current design permits operation with typical standby and active currents of only 500 nA and 320 μ A, respectively. The MCP3204 is offered in 14-pin PDIP, 150 mil SOIC and TSSOP packages. The MCP3208 is offered in 16-pin PDIP and SOIC packages.

Functional Block Diagram



* Note: Channels 5-7 available on MCP3208 Only

MCP3204/3208

1.0 ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings*

V_{DD} 7.0V

All inputs and outputs w.r.t. V_{SS} -0.6V to V_{DD} +0.6V

Storage temperature -65°C to +150°C

Ambient temp. with power applied -65°C to +125°C

Soldering temperature of leads (10 seconds) +300°C

ESD protection on all pins > 4 kV

***Notice:** Stresses above those listed under "Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

PIN FUNCTION TABLE

Name	Function
V_{DD}	+2.7V to 5.5V Power Supply
DGND	Digital Ground
AGND	Analog Ground
CH0-CH7	Analog Inputs
CLK	Serial Clock
D_{IN}	Serial Data In
D_{OUT}	Serial Data Out
CS/SHDN	Chip Select/Shutdown Input
V_{REF}	Reference Voltage Input

ELECTRICAL SPECIFICATIONS

Electrical Characteristics: Unless otherwise noted, all parameters apply at $V_{DD} = 5V$, $V_{SS} = 0V$, $V_{REF} = 5V$, $T_{AMB} = -40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$, $f_{SAMPLE} = 100$ ksps and $f_{CLK} = 20 \times f_{SAMPLE}$

Parameters	Sym	Min	Typ	Max	Units	Conditions
Conversion Rate						
Conversion Time	t_{CONV}	—	—	12	clock cycles	
Analog Input Sample Time	t_{SAMPLE}		1.5		clock cycles	
Throughput Rate	f_{SAMPLE}	—	—	100 50	ksps ksps	$V_{DD} = V_{REF} = 5V$ $V_{DD} = V_{REF} = 2.7V$
DC Accuracy						
Resolution			12		bits	
Integral Nonlinearity	INL	— —	±0.75 ±1.0	±1 ±2	LSB	MCP3204/3208-B MCP3204/3208-C
Differential Nonlinearity	DNL	—	±0.5	±1	LSB	No missing codes over-temperature
Offset Error		—	±1.25	±3	LSB	
Gain Error		—	±1.25	±5	LSB	
Dynamic Performance						
Total Harmonic Distortion		—	-82	—	dB	$V_{IN} = 0.1V$ to $4.9V$ @1 kHz
Signal to Noise and Distortion (SINAD)		—	72	—	dB	$V_{IN} = 0.1V$ to $4.9V$ @1 kHz
Spurious Free Dynamic Range		—	86	—	dB	$V_{IN} = 0.1V$ to $4.9V$ @1 kHz
Reference Input						
Voltage Range		0.25	—	V_{DD}	V	Note 2
Current Drain		—	100 0.001	150 3.0	μA μA	$\overline{CS} = V_{DD} = 5V$

Note 1: This parameter is established by characterization and not 100% tested.

2: See graphs that relate linearity performance to V_{REF} levels.

3: Because the sample cap will eventually lose charge, effective clock rates below 10 kHz can affect linearity performance, particularly at elevated temperatures. See Section 6.2, "Maintaining Minimum Clock Speed", for more information.

MCP3204/3208

ELECTRICAL SPECIFICATIONS (CONTINUED)

Electrical Characteristics: Unless otherwise noted, all parameters apply at $V_{DD} = 5V$, $V_{SS} = 0V$, $V_{REF} = 5V$, $T_{AMB} = -40^{\circ}C$ to $+85^{\circ}C$, $f_{SAMPLE} = 100$ kspS and $f_{CLK} = 20 \times f_{SAMPLE}$						
Parameters	Sym	Min	Typ	Max	Units	Conditions
Analog Inputs						
Input Voltage Range for CH0-CH7 in Single-Ended Mode		V_{SS}	—	V_{REF}	V	
Input Voltage Range for IN+ in pseudo-differential Mode		IN-	—	$V_{REF} + IN -$		
Input Voltage Range for IN- in pseudo-differential Mode		$V_{SS} - 100$	—	$V_{SS} + 100$	mV	
Leakage Current		—	0.001	± 1	μA	
Switch Resistance		—	1000	—	Ω	See Figure 4-1
Sample Capacitor		—	20	—	pF	See Figure 4-1
Digital Input/Output						
Data Coding Format		Straight Binary				
High Level Input Voltage	V_{IH}	0.7 V_{DD}	—	—	V	
Low Level Input Voltage	V_{IL}	—	—	0.3 V_{DD}	V	
High Level Output Voltage	V_{OH}	4.1	—	—	V	$I_{OH} = -1$ mA, $V_{DD} = 4.5$ V
Low Level Output Voltage	V_{OL}	—	—	0.4	V	$I_{OL} = 1$ mA, $V_{DD} = 4.5$ V
Input Leakage Current	I_{LI}	-10	—	10	μA	$V_{IN} = V_{SS}$ or V_{DD}
Output Leakage Current	I_{LO}	-10	—	10	μA	$V_{OUT} = V_{SS}$ or V_{DD}
Pin Capacitance (All Inputs/Outputs)	$C_{IN,COUT}$	—	—	10	pF	$V_{DD} = 5.0$ V (Note 1) $T_{AMB} = 25^{\circ}C$, $f = 1$ MHz
Timing Parameters						
Clock Frequency	f_{CLK}	—	—	2.0 1.0	MHz MHz	$V_{DD} = 5$ V (Note 3) $V_{DD} = 2.7$ V (Note 3)
Clock High Time	t_{HI}	250	—	—	ns	
Clock Low Time	t_{LO}	250	—	—	ns	
CS Fall To First Rising CLK Edge	t_{SUCS}	100	—	—	ns	
Data Input Setup Time	t_{SU}	—	—	50	ns	
Data Input Hold Time	t_{HD}	—	—	50	ns	
CLK Fall To Output Data Valid	t_{DO}	—	—	200	ns	See Figures 1-2 and 1-3
CLK Fall To Output Enable	t_{EN}	—	—	200	ns	See Figures 1-2 and 1-3
CS Rise To Output Disable	t_{DIS}	—	—	100	ns	See Figures 1-2 and 1-3
CS Disable Time	t_{CSH}	500	—	—	ns	
D_{OUT} Rise Time	t_R	—	—	100	ns	See Figures 1-2 and 1-3 (Note 1)
D_{OUT} Fall Time	t_F	—	—	100	ns	See Figures 1-2 and 1-3 (Note 1)
Power Requirements						
Operating Voltage	V_{DD}	2.7	—	5.5	V	
Operating Current	I_{DD}	—	320 225	400 —	μA	$V_{DD} = V_{REF} = 5$ V, D_{OUT} unloaded $V_{DD} = V_{REF} = 2.7$ V, D_{OUT} unloaded
Standby Current	I_{DDS}	—	0.5	2.0	μA	$\bar{CS} = V_{DD} = 5.0$ V

Note 1: This parameter is established by characterization and not 100% tested.

2: See graphs that relate linearity performance to V_{REF} levels.

3: Because the sample cap will eventually lose charge, effective clock rates below 10 kHz can affect linearity performance, particularly at elevated temperatures. See Section 6.2, "Maintaining Minimum Clock Speed", for more information.

MCP3204/3208

ELECTRICAL SPECIFICATIONS (CONTINUED)

Electrical Characteristics: Unless otherwise noted, all parameters apply at $V_{DD} = 5V$, $V_{SS} = 0V$, $V_{REF} = 5V$, $T_{AMB} = -40^{\circ}C$ to $+85^{\circ}C$, $f_{SAMPLE} = 100$ kspS and $f_{CLK} = 20*f_{SAMPLE}$

Parameters	Sym	Min	Typ	Max	Units	Conditions
Temperature Ranges						
Specified Temperature Range	T_A	-40	—	+85	°C	
Operating Temperature Range	T_A	-40	—	+85	°C	
Storage Temperature Range	T_A	-65	—	+150	°C	
Thermal Package Resistance						
Thermal Resistance, 14L-PDIP	θ_{JA}	—	70	—	°C/W	
Thermal Resistance, 14L-SOIC	θ_{JA}	—	108	—	°C/W	
Thermal Resistance, 14L-TSSOP	θ_{JA}	—	100	—	°C/W	
Thermal Resistance, 16L-PDIP	θ_{JA}	—	70	—	°C/W	
Thermal Resistance, 16L-SOIC	θ_{JA}	—	90	—	°C/W	

Note 1: This parameter is established by characterization and not 100% tested.

2: See graphs that relate linearity performance to V_{REF} levels.

3: Because the sample cap will eventually lose charge, effective clock rates below 10 kHz can affect linearity performance, particularly at elevated temperatures. See Section 6.2, "Maintaining Minimum Clock Speed", for more information.

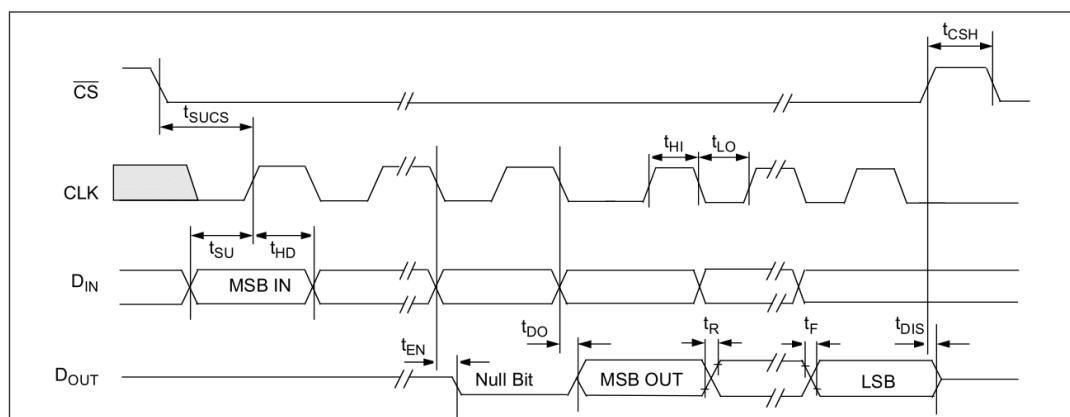


FIGURE 1-1: Serial Interface Timing.

MCP3204/3208

5.0 SERIAL COMMUNICATIONS

Communication with the MCP3204/3208 devices is accomplished using a standard SPI-compatible serial interface. Initiating communication with either device is done by bringing the CS line low (see Figure 5-1). If the device was powered up with the CS pin low, it must be brought high and back low to initiate communication. The first clock received with CS low and D_{IN} high will constitute a start bit. The SGL/DIFF bit follows the start bit and will determine if the conversion will be done using single-ended or differential input mode. The next three bits (D0, D1 and D2) are used to select the input channel configuration. Table 5-1 and Table 5-2 show the configuration bits for the MCP3204 and MCP3208, respectively. The device will begin to sample the analog input on the fourth rising edge of the clock after the start bit has been received. The sample period will end on the falling edge of the fifth clock following the start bit.

Once the D0 bit is input, one more clock is required to complete the sample and hold period (D_{IN} is a “don’t care” for this clock). On the falling edge of the next clock, the device will output a low null bit. The next 12 clocks will output the result of the conversion with MSB first, as shown in Figure 5-1. Data is always output from the device on the falling edge of the clock. If all 12 data bits have been transmitted and the device continues to receive clocks while the CS is held low, the device will output the conversion result LSB first, as shown in Figure 5-2. If more clocks are provided to the device while CS is still low (after the LSB first data has been transmitted), the device will clock out zeros indefinitely. If necessary, it is possible to bring CS low and clock in leading zeros on the D_{IN} line before the start bit. This is often done when dealing with microcontroller-based SPI ports that must send 8 bits at a time. Refer to Section 6.1 for more details on using the MCP3204/3208 devices with hardware SPI ports.

TABLE 5-1: CONFIGURATION BITS FOR THE MCP3204

Control Bit Selections				Input Configuration	Channel Selection
Single/ Diff	D2*	D1	D0		
1	X	0	0	single-ended	CH0
1	X	0	1	single-ended	CH1
1	X	1	0	single-ended	CH2
1	X	1	1	single-ended	CH3
0	X	0	0	differential	CH0 = IN+ CH1 = IN-
0	X	0	1	differential	CH0 = IN- CH1 = IN+
0	X	1	0	differential	CH2 = IN+ CH3 = IN-
0	X	1	1	differential	CH2 = IN- CH3 = IN+

* D2 is a “don’t care” for MCP3204

TABLE 5-2: CONFIGURATION BITS FOR THE MCP3208

Control Bit Selections				Input Configuration	Channel Selection
Single/ Diff	D2	D1	D0		
1	0	0	0	single-ended	CH0
1	0	0	1	single-ended	CH1
1	0	1	0	single-ended	CH2
1	0	1	1	single-ended	CH3
1	1	0	0	single-ended	CH4
1	1	0	1	single-ended	CH5
1	1	1	0	single-ended	CH6
1	1	1	1	single-ended	CH7
0	0	0	0	differential	CH0 = IN+ CH1 = IN-
0	0	0	1	differential	CH0 = IN- CH1 = IN+
0	0	1	0	differential	CH2 = IN+ CH3 = IN-
0	0	1	1	differential	CH2 = IN- CH3 = IN+
0	1	0	0	differential	CH4 = IN+ CH5 = IN-
0	1	0	1	differential	CH4 = IN- CH5 = IN+
0	1	1	0	differential	CH6 = IN+ CH7 = IN-
0	1	1	1	differential	CH6 = IN- CH7 = IN+

MCP3204/3208

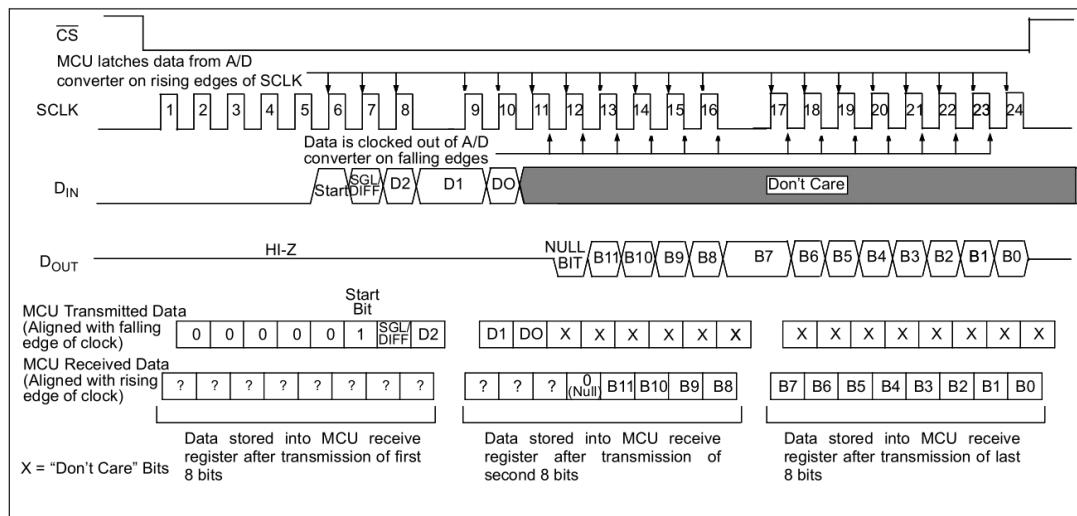


FIGURE 6-1: SPI Communication using 8-bit segments (Mode 0,0: SCLK idles low).

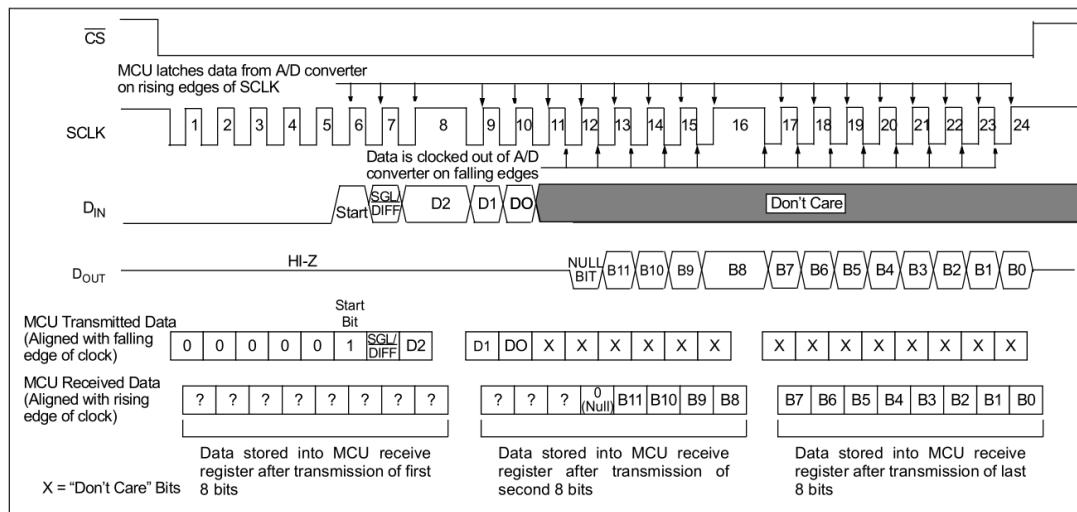


FIGURE 6-2: SPI Communication using 8-bit segments (Mode 1,1; SCLK idles high).

1.6 BSS138



BSS138W

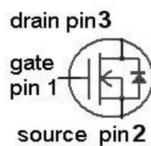
SIPMOS® Small-Signal-Transistor

Features

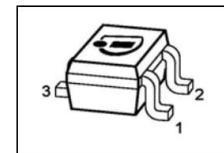
- N-channel
- Enhancement mode
- Logic level
- dv/dt rated
- Pb-free lead-plating; RoHS compliant
- Qualified according to AEC Q101
- Halogen-free according to IEC61249-2-21

Product Summary

V_{DS}	60	V
$R_{DS(on),max}$	3.5	Ω
I_D	0.28	A



PG-SOT-323



Type	Package	Tape and Reel	Marking
BSS138W	PG-SOT-323	H6327: 3000 /	SWs
BSS138W	PG-SOT-323	H6433: 10000	SWs

Maximum ratings, at $T_j=25$ °C, unless otherwise specified

Parameter	Symbol	Conditions	Value	Unit
Continuous drain current	I_D	$T_A=25$ °C	0.28	A
		$T_A=70$ °C	0.22	
Pulsed drain current	$I_{D,pulse}$	$T_A=25$ °C	1.12	
Reverse diode dv/dt	dv/dt	$I_D=0.28$ A, $V_{DS}=48$ V, $di/dt=200$ A/ μ s, $T_{j,max}=150$ °C	6	kV/ μ s
Gate source voltage	V_{GS}		± 20	V
ESD class (JESD22-A114-HBM)			0 (<250V)	
Power dissipation	P_{tot}	$T_A=25$ °C	0.50	W
Operating and storage temperature	T_j, T_{stg}		-55 ... 150	°C
IEC climatic category; DIN IEC 68-1			55/150/56	



BSS138W

Parameter	Symbol	Conditions	Values			Unit
			min.	typ.	max.	

Thermal characteristics

Thermal resistance, junction - minimal footprint	R_{thJA}		-	-	250	K/W
--	------------	--	---	---	-----	-----

Electrical characteristics, at $T_j=25$ °C, unless otherwise specified**Static characteristics**

Drain-source breakdown voltage	$V_{(BR)DSS}$	$V_{GS}=0$ V, $I_D=250$ μ A	60	-	-	V
Gate threshold voltage	$V_{GS(th)}$	$V_{GS}=V_{DS}$, $I_D=26$ μ A	0.6	1.0	1.4	
Drain-source leakage current	$I_D(\text{off})$	$V_{DS}=60$ V, $V_{GS}=0$ V, $T_j=25$ °C	-	-	0.1	μ A
		$V_{DS}=60$ V, $V_{GS}=0$ V, $T_j=150$ °C	-	-	5	
Gate-source leakage current	I_{GSS}	$V_{GS}=20$ V, $V_{DS}=0$ V	-	1	10	nA
Drain-source on-state resistance	$R_{DS(on)}$	$V_{GS}=4.5$ V, $I_D=0.03$ A	-	3	4.0	Ω
		$V_{GS}=4.5$ V, $I_D=0.16$ A	-	3.2	6	
		$V_{GS}=10$ V, $I_D=0.2$ A	-	2.1	3.5	
Transconductance	g_{fs}	$ V_{DS} >2 I_D R_{DS(\text{on})\text{max}}$, $I_D=0.22$ A	0.12	0.23	-	s



BSS138W

Parameter	Symbol	Conditions	Values			Unit
			min.	typ.	max.	

Dynamic characteristics

Input capacitance	C_{iss}	$V_{GS}=0 \text{ V}, V_{DS}=25 \text{ V}, f=1 \text{ MHz}$	-	32	43	pF
Output capacitance	C_{oss}		-	7.2	10	
Reverse transfer capacitance	C_{rss}		-	2.8	4.2	
Turn-on delay time	$t_{d(on)}$	$V_{DD}=30 \text{ V}, V_{GS}=10 \text{ V}, I_D=0.2 \text{ A}, R_G=6 \Omega$	-	2.2	3.3	ns
Rise time	t_r		-	3.0	4.5	
Turn-off delay time	$t_{d(off)}$		-	6.7	10	
Fall time	t_f		-	8.2	12	

Gate Charge Characteristics

Gate to source charge	Q_{gs}	$V_{DD}=48 \text{ V}, I_D=0.2 \text{ A}, V_{GS}=0 \text{ to } 10 \text{ V}$	-	0.10	0.13	nC
Gate to drain charge	Q_{gd}		-	0.3	0.4	
Gate charge total	Q_g		-	1.0	1.5	
Gate plateau voltage	$V_{plateau}$		-	3.2	-	

Reverse Diode

Diode continuous forward current	I_s	$T_A=25 \text{ }^\circ\text{C}$	-	-	0.28	A
Diode pulse current	$I_{s,pulse}$		-	-	1.12	
Diode forward voltage	V_{SD}	$V_{GS}=0 \text{ V}, I_F=0.28 \text{ A}, T_j=25 \text{ }^\circ\text{C}$	-	0.85	1.2	V
Reverse recovery time	t_{rr}	$V_R=30 \text{ V}, I_F=0.28 \text{ A}, di_F/dt=100 \text{ A}/\mu\text{s}$	-	8.3	12.4	ns
Reverse recovery charge	Q_{rr}		-	3.3	5	nC

1.7 DM7404

DM7404 Hex Inverting Gates



August 1986
Revised February 2000

DM7404

Hex Inverting Gates

General Description

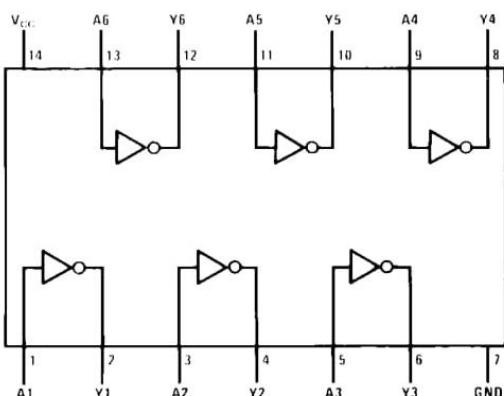
This device contains six independent gates each of which performs the logic INVERT function.

Ordering Code:

Order Number	Package Number	Package Description
DM7404M	M14A	14-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-012, 0.150 Narrow
DM7404N	N14A	14-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300 Wide

Devices also available in Tape and Reel. Specify by appending the suffix letter "X" to the ordering code.

Connection Diagram



Function Table

$$Y = \bar{A}$$

Inputs	Output
A	Y
L	H
H	L

H = HIGH Logic Level
L = LOW Logic Level

Absolute Maximum Ratings (Note 1)

Supply Voltage	7V
Input Voltage	5.5V
Operating Free Air Temperature Range	0°C to +70°C
Storage Temperature Range	-65°C to +150°C

Note 1: The "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. The device should not be operated at these limits. The parametric values defined in the Electrical Characteristics tables are not guaranteed at the absolute maximum ratings. The "Recommended Operating Conditions" table will define the conditions for actual device operation.

Recommended Operating Conditions

Symbol	Parameter	Min	Nom	Max	Units
V _{CC}	Supply Voltage	4.75	5	5.25	V
V _{IH}	HIGH Level Input Voltage	2			V
V _{IL}	LOW Level Input Voltage			0.8	V
I _{OH}	HIGH Level Output Current			-0.4	mA
I _{OL}	LOW Level Output Current			16	mA
T _A	Free Air Operating Temperature	0		70	°C

Electrical Characteristics

over recommended operating free air temperature range (unless otherwise noted)

Symbol	Parameter	Conditions	Min	Typ (Note 2)	Max	Units
V _I	Input Clamp Voltage	V _{CC} = Min, I _I = -12 mA			-1.5	V
V _{OH}	HIGH Level Output Voltage	V _{CC} = Min, I _{OH} = Max V _{IL} = Max	2.4	3.4		V
V _{OL}	LOW Level Output Voltage	V _{CC} = Min, I _{OL} = Max V _{IH} = Min		0.2	0.4	V
I _I	Input Current @ Max Input Voltage	V _{CC} = Max, V _I = 5.5V			1	mA
I _{IH}	HIGH Level Input Current	V _{CC} = Max, V _I = 2.4V			40	µA
I _{IL}	LOW Level Input Current	V _{CC} = Max, V _I = 0.4V			-1.6	mA
I _{OS}	Short Circuit Output Current	V _{CC} = Max (Note 3)	-18		-55	mA
I _{CCH}	Supply Current with Outputs HIGH	V _{CC} = Max		6	12	mA
I _{CCL}	Supply Current with Outputs LOW	V _{CC} = Max		18	33	mA

Note 2: All typicals are at V_{CC} = 5V, T_A = 25°C.

Note 3: Not more than one output should be shorted at a time.

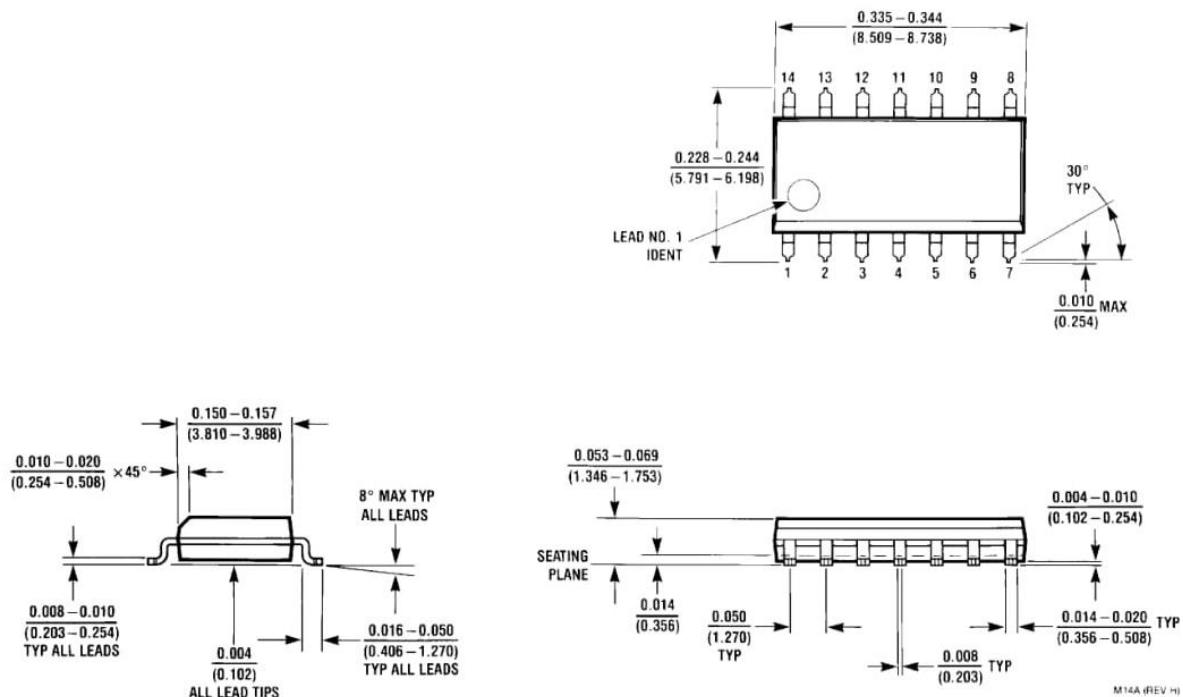
Switching Characteristics

at V_{CC} = 5V and T_A = 25°C

Symbol	Parameter	Conditions	Min	Max	Units
t _{PLH}	Propagation Delay Time LOW-to-HIGH Level Output	C _L = 15 pF R _L = 400Ω		22	ns
t _{PHL}	Propagation Delay Time HIGH-to-LOW Level Output			15	ns

DM7404

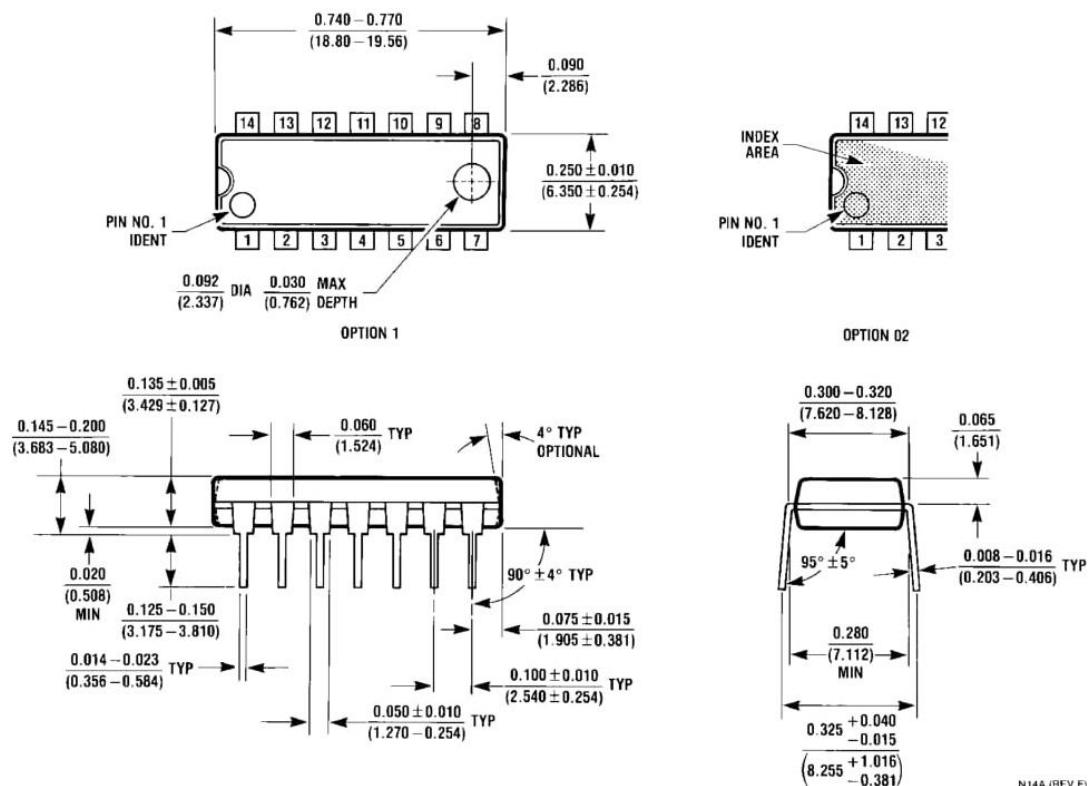
Physical Dimensions inches (millimeters) unless otherwise noted



14-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-012, 0.150 Narrow
Package Number M14A

DM7404 Hex Inverting Gates

Physical Dimensions inches (millimeters) unless otherwise noted (Continued)



**14-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300 Wide
Package Number N14A**

Fairchild does not assume any responsibility for use of any circuitry described, no circuit patent licenses are implied and Fairchild reserves the right at any time without notice to change said circuitry and specifications.

LIFE SUPPORT POLICY

FAIRCHILD'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF FAIRCHILD SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and (c) whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
 2. A critical component in any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

www.fairchildsemi.com

1.8 LM317

Product
FolderSample &
BuyTechnical
DocumentsTools &
SoftwareSupport &
Community

LM317

SLVS044X – SEPTEMBER 1997 – REVISED SEPTEMBER 2016

LM317 3-Terminal Adjustable Regulator

1 Features

- Output Voltage Range Adjustable From 1.25 V to 37 V
- Output Current Greater Than 1.5 A
- Internal Short-Circuit Current Limiting
- Thermal Overload Protection
- Output Safe-Area Compensation

2 Applications

- ATCA Solutions
- DLP: 3D Biometrics, Hyperspectral Imaging, Optical Networking, and Spectroscopy
- DVR and DVS
- Desktop PC
- Digital Signage and Still Camera
- ECG Electrocardiogram
- EV HEV Charger: Level 1, 2, and 3
- Electronic Shelf Label
- Energy Harvesting
- Ethernet Switch
- Femto Base Station
- Fingerprint and Iris Biometrics
- HVAC: Heating, Ventilating, and Air Conditioning
- High-Speed Data Acquisition and Generation
- Hydraulic Valve
- IP Phone: Wired and Wireless
- Intelligent Occupancy Sensing
- Motor Control: Brushed DC, Brushless DC, Low-Voltage, Permanent Magnet, and Stepper Motor
- Point-to-Point Microwave Backhaul
- Power Bank Solutions
- Power Line Communication Modem
- Power Over Ethernet (PoE)
- Power Quality Meter
- Power Substation Control
- Private Branch Exchange (PBX)
- Programmable Logic Controller
- RFID Reader
- Refrigerator
- Signal or Waveform Generator
- Software Defined Radio (SDR)
- Washing Machine: High-End and Low-End
- X-ray: Baggage Scanner, Medical, and Dental

3 Description

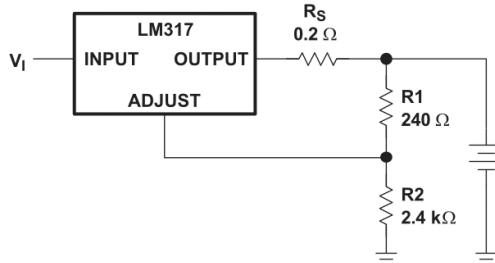
The LM317 device is an adjustable three-terminal positive-voltage regulator capable of supplying more than 1.5 A over an output-voltage range of 1.25 V to 37 V. It requires only two external resistors to set the output voltage. The device features a typical line regulation of 0.01% and typical load regulation of 0.1%. It includes current limiting, thermal overload protection, and safe operating area protection. Overload protection remains functional even if the ADJUST terminal is disconnected.

Device Information⁽¹⁾

PART NUMBER	PACKAGE	BODY SIZE (NOM)
LM317DCY	SOT-223 (4)	6.50 mm × 3.50 mm
LM317KCS	TO-220 (3)	10.16 mm × 9.15 mm
LM317KCT	TO-220 (3)	10.16 mm × 8.59 mm
LM317KTT	TO-263 (3)	10.16 mm × 9.01 mm

(1) For all available packages, see the orderable addendum at the end of the data sheet.

Battery-Charger Circuit



Copyright © 2016, Texas Instruments Incorporated



An IMPORTANT NOTICE at the end of this data sheet addresses availability, warranty, changes, use in safety-critical applications, intellectual property matters and other important disclaimers. PRODUCTION DATA.



LM317

SLVS044X—SEPTEMBER 1997—REVISED SEPTEMBER 2016

www.ti.com

8 Application and Implementation

NOTE

Information in the following applications sections is not part of the TI component specification, and TI does not warrant its accuracy or completeness. TI's customers are responsible for determining suitability of components for their purposes. Customers should validate and test their design implementation to confirm system functionality.

8.1 Application Information

The flexibility of the LM317 allows it to be configured to take on many different functions in DC power applications.

8.2 Typical Application

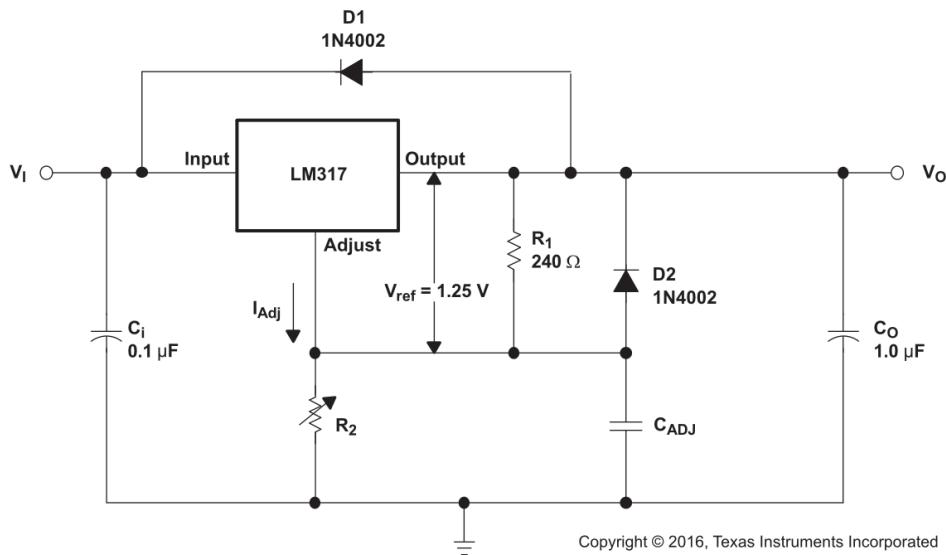


Figure 9. Adjustable Voltage Regulator

8.2.1 Design Requirements

- R1 and R2 are required to set the output voltage.
- C_{ADJ} is recommended to improve ripple rejection. It prevents amplification of the ripple as the output voltage is adjusted higher.
- C_i is recommended, particularly if the regulator is not in close proximity to the power-supply filter capacitors. A 0.1-μF or 1-μF ceramic or tantalum capacitor provides sufficient bypassing for most applications, especially when adjustment and output capacitors are used.
- C_O improves transient response, but is not needed for stability.
- Protection diode D2 is recommended if C_{ADJ} is used. The diode provides a low-impedance discharge path to prevent the capacitor from discharging into the output of the regulator.
- Protection diode D1 is recommended if C_O is used. The diode provides a low-impedance discharge path to prevent the capacitor from discharging into the output of the regulator.

8.2.2 Detailed Design Procedure

V_O is calculated as shown in [Equation 1](#). I_{ADJ} is typically 50 μA and negligible in most applications.

$$V_O = V_{REF} \left(1 + R2 / R1 \right) + (I_{ADJ} \times R2) \quad (1)$$



LM317

www.ti.com

SLVS044X –SEPTEMBER 1997–REVISED SEPTEMBER 2016

System Examples (continued)

8.3.5 1.25-V to 20-V Regulator Circuit With Minimum Program Current

Because the value of V_{REF} is constant, the value of R1 determines the amount of current that flows through R1 and R2. The size of R2 determines the IR drop from ADJUSTMENT to GND. Higher values of R2 translate to higher V_{OUT} .

$$V_{OUT} = V_{REF} \left(1 + \frac{R_2 + R_3}{R_1} \right) - 10V \quad (2)$$

$$(R_1 + R_2)_{min} = V_{out\text{reg(min)}} \quad (3)$$

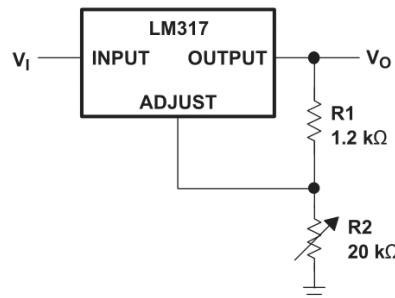


Figure 16. 1.25-V to 20-V Regulator Circuit With Minimum Program Current

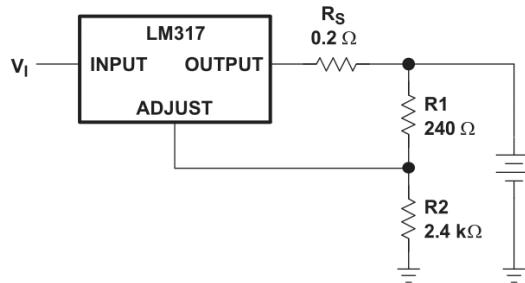
8.3.6 Battery-Charger Circuit

The series resistor limits the current output of the LM317, minimizing damage to the battery cell.

$$V_{OUT} = 1.25V \times \left(\frac{R_2}{R_1 + 1} \right) \quad (4)$$

$$I_{OUT(\text{short})} = \frac{1.25V}{R_S} \quad (5)$$

$$\text{Output impedance} = R_S \times \left(\frac{R_2}{R_1 + 1} \right) \quad (6)$$



Copyright © 2016, Texas Instruments Incorporated

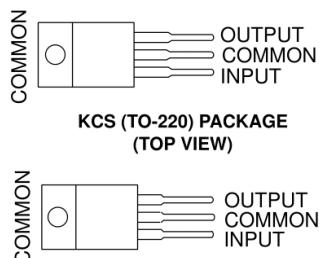
Figure 17. Battery-Charger Circuit

1.9 7805

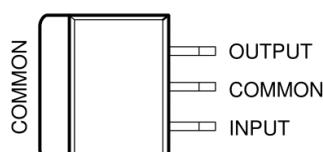
μ A7800 SERIES POSITIVE-VOLTAGE REGULATORS

SLVS056J – MAY 1976 – REVISED MAY 2003

- 3-Terminal Regulators
- Output Current up to 1.5 A
- Internal Thermal-Overload Protection

KC (TO-220) PACKAGE
(TOP VIEW)

- High Power-Dissipation Capability
- Internal Short-Circuit Current Limiting
- Output Transistor Safe-Area Compensation

KTE PACKAGE
(TOP VIEW)**description/ordering information**

This series of fixed-voltage integrated-circuit voltage regulators is designed for a wide range of applications. These applications include on-card regulation for elimination of noise and distribution problems associated with single-point regulation. Each of these regulators can deliver up to 1.5 A of output current. The internal current-limiting and thermal-shutdown features of these regulators essentially make them immune to overload. In addition to use as fixed-voltage regulators, these devices can be used with external components to obtain adjustable output voltages and currents, and also can be used as the power-pass element in precision regulators.

ORDERING INFORMATION

T _J	V _{O(NOM)} (V)	PACKAGE [†]	ORDERABLE PART NUMBER	TOP-SIDE MARKING
0°C to 125°C	5	POWER-FLEX (KTE)	Reel of 2000	μ A7805CKTER
		TO-220 (KC)	Tube of 50	μ A7805CKC
		TO-220, short shoulder (KCS)	Tube of 20	μ A7805CKCS
	8	POWER-FLEX (KTE)	Reel of 2000	μ A7808CKTER
		TO-220 (KC)	Tube of 50	μ A7808CKC
		TO-220, short shoulder (KCS)	Tube of 20	μ A7808CKCS
	10	POWER-FLEX (KTE)	Reel of 2000	μ A7810CKTER
		TO-220 (KC)	Tube of 50	μ A7810CKC
	12	POWER-FLEX (KTE)	Reel of 2000	μ A7812CKTER
		TO-220 (KC)	Tube of 50	μ A7812CKC
		TO-220, short shoulder (KCS)	Tube of 20	μ A7812CKCS
	15	POWER-FLEX (KTE)	Reel of 2000	μ A7815CKTER
		TO-220 (KC)	Tube of 50	μ A7815CKC
		TO-220, short shoulder (KCS)	Tube of 20	μ A7815CKCS
	24	POWER-FLEX (KTE)	Reel of 2000	μ A7824CKTER
		TO-220 (KC)	Tube of 50	μ A7824CKC

[†] Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at www.ti.com/sc/package.



Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

PRODUCTION DATA information is current as of publication date.
Products conform to specifications per the terms of Texas Instruments standard warranty. Production processing does not necessarily include testing of all parameters.

Copyright © 2003, Texas Instruments Incorporated



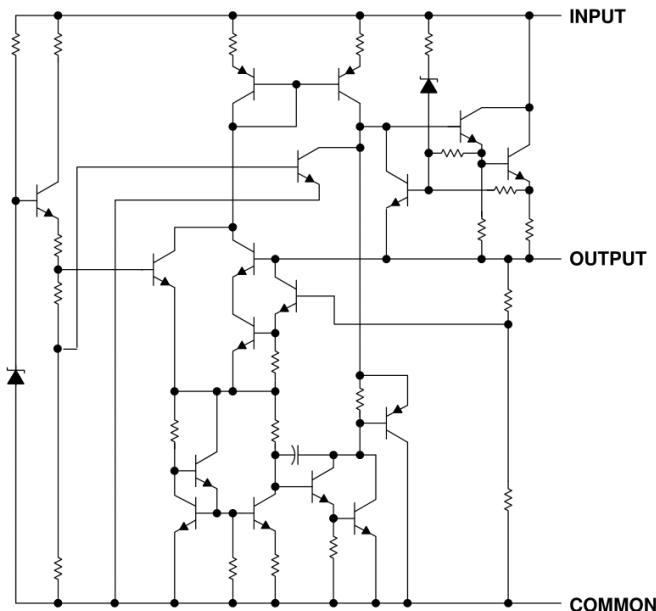
POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

1

μA7800 SERIES POSITIVE-VOLTAGE REGULATORS

SLVS056J – MAY 1976 – REVISED MAY 2003

schematic



absolute maximum ratings over virtual junction temperature range (unless otherwise noted)†

Input voltage, V_I : μ A7824C	40 V
All others	35 V
Operating virtual junction temperature, T_J	150°C
Lead temperature 1,6 mm (1/16 inch) from case for 10 seconds	260°C
Storage temperature range, T_{stg}	-65°C to 150°C

[†] Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

package thermal data (see Note 1)

PACKAGE	BOARD	θ_{JC}	θ_{JA}
POWER-FLEX (KTE)	High K, JESD 51-5	3°C/W	23°C/W
TO-220 (KC/KCS)	High K, JESD 51-5	3°C/W	19°C/W

NOTE 1: Maximum power dissipation is a function of $T_J(\max)$, θ_{JA} , and T_A . The maximum allowable power dissipation at any allowable ambient temperature is $P_D = (T_J(\max) - T_A)/\theta_{JA}$. Operating at the absolute maximum T_J of 150°C can affect reliability.

**μ A7800 SERIES
POSITIVE-VOLTAGE REGULATORS**

SLVS056J – MAY 1976 – REVISED MAY 2003

recommended operating conditions

			MIN	MAX	UNIT
V_I Input voltage	μ A7805C	7	25		
	μ A7808C	10.5	25		
	μ A7810C	12.5	28		
	μ A7812C	14.5	30		
	μ A7815C	17.5	30		
	μ A7824C	27	38		
I_O Output current			1.5	A	
T_J Operating virtual junction temperature	μ A7800C series	0	125	$^{\circ}$ C	

electrical characteristics at specified virtual junction temperature, $V_I = 10$ V, $I_O = 500$ mA (unless otherwise noted)

PARAMETER	TEST CONDITIONS	T_J [†]	μ A7805C			UNIT
			MIN	TYP	MAX	
Output voltage	$I_O = 5$ mA to 1 A, $V_I = 7$ V to 20 V, $P_D \leq 15$ W	25°C	4.8	5	5.2	V
		0°C to 125°C	4.75		5.25	
Input voltage regulation	$V_I = 7$ V to 25 V	25°C		3	100	mV
	$V_I = 8$ V to 12 V			1	50	
Ripple rejection	$V_I = 8$ V to 18 V, $f = 120$ Hz	0°C to 125°C	62	78		dB
Output voltage regulation	$I_O = 5$ mA to 1.5 A	25°C		15	100	mV
	$I_O = 250$ mA to 750 mA			5	50	
Output resistance	$f = 1$ kHz	0°C to 125°C		0.017		Ω
Temperature coefficient of output voltage	$I_O = 5$ mA	0°C to 125°C		-1.1		mV/ $^{\circ}$ C
Output noise voltage	$f = 10$ Hz to 100 kHz	25°C		40		μ V
Dropout voltage	$I_O = 1$ A	25°C		2		V
Bias current		25°C		4.2	8	mA
Bias current change	$V_I = 7$ V to 25 V	0°C to 125°C		1.3		mA
	$I_O = 5$ mA to 1 A			0.5		
Short-circuit output current		25°C		750		mA
Peak output current		25°C		2.2		A

[†] Pulse-testing techniques maintain the junction temperature as close to the ambient temperature as possible. Thermal effects must be taken into account separately. All characteristics are measured with a 0.33- μ F capacitor across the input and a 0.1- μ F capacitor across the output.



POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

μA7800 SERIES POSITIVE-VOLTAGE REGULATORS

SLVS056J – MAY 1976 – REVISED MAY 2003

APPLICATION INFORMATION

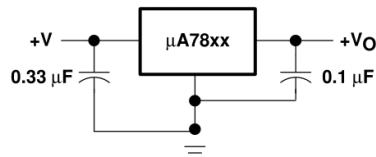


Figure 1. Fixed-Output Regulator

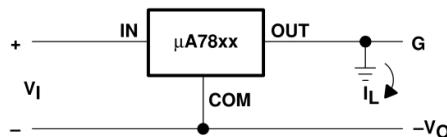
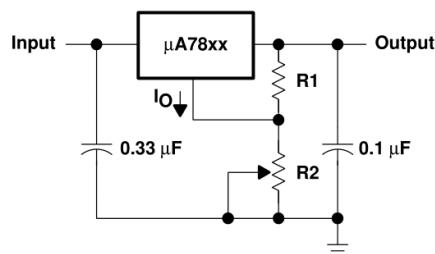


Figure 2. Positive Regulator in Negative Configuration (V_I Must Float)



NOTE A: The following formula is used when V_{XX} is the nominal output voltage (output to common) of the fixed regulator:

$$V_O = V_{XX} + \left(\frac{V_{XX}}{R_1} + I_Q \right) R_2$$

Figure 3. Adjustable-Output Regulator

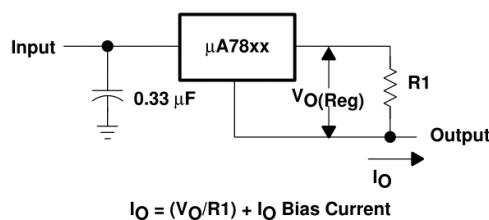


Figure 4. Current Regulator

1.10 7833

**1A Positive
Voltage Regulator**
LM7833/LM7847

1A Positive Voltage Regulator**General Description**

- The TCI LM78XX family is monolithic fixed voltage regulator integrated circuit. They are suitable for applications that required supply current up to 1A.
- The LM78M is available in D-PACK (TO-252) and TO-220 packages.



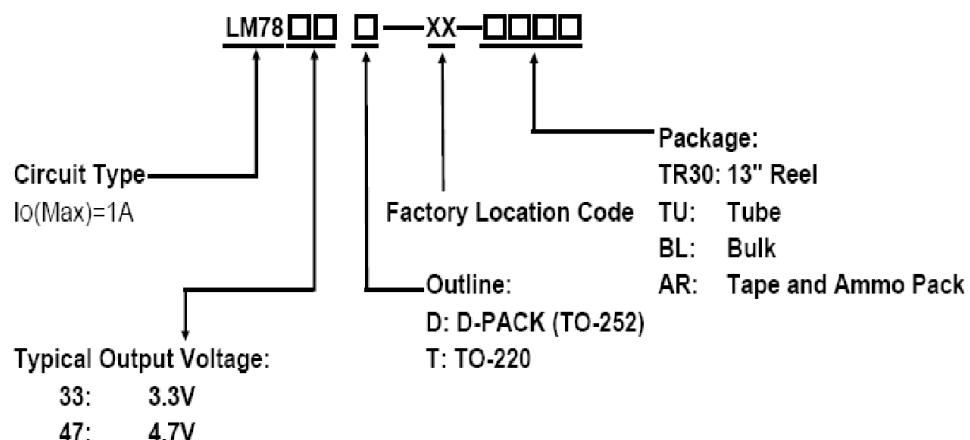
D-PACK (TO-252) TO-220

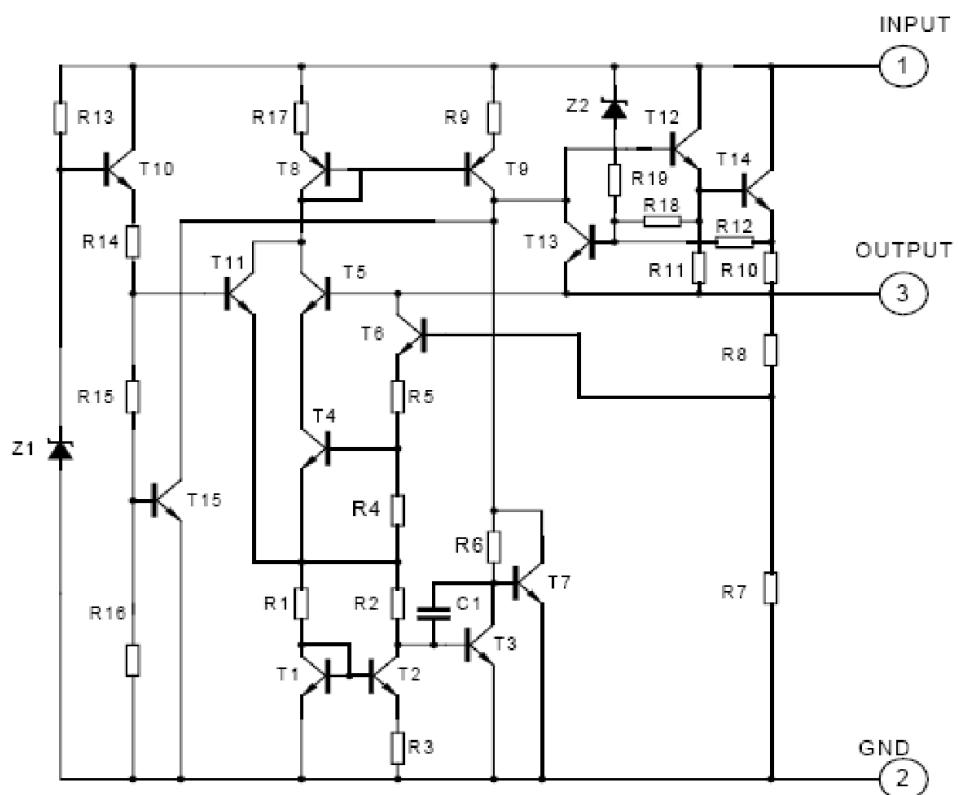
**Features**

- Output Current up to 1A
- Fixed output voltage of 3.3V and 4.7V available
- Thermal overload shutdown protection
- Short circuit current limiting
- Output transistor SOA protection
- RoHS Compliance

Applications

- High Efficiency Linear Regulator
- Post Regulation for Switching Supply
- Microprocessor Power Supply
- Mother Board

Ordering Information

1A Positive Voltage Regulator**LM7833/LM7847****Pin Configuration**Outline: D
D-PACK
(TO-252)Outline: T
TO-220**Block Diagram**

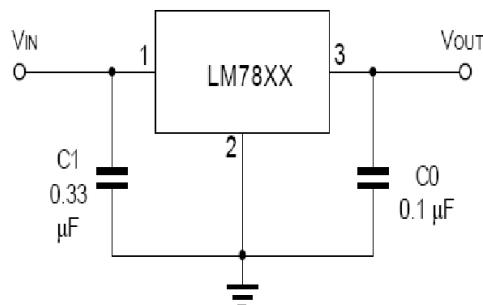
1A Positive Voltage Regulator

LM7833/LM7847

For LM7847 ($V_{IN}=9.7V$, $I_{OUT}=0.5A$, $C_1=0.33\mu F$, $C_0=0.1\mu F$)

Symbol	Description	LM7847			Unit	Test Conditions
		Min.	Typ.	Max.		
V_{OUT}	Output Voltage	4.512	4.70	4.888	V	$I_{OUT}=5mA-1.0A$
		4.465	-	4.935	V	$7.2V \leq V_{IN} \leq 19.7V$, $I_{OUT}=5mA-1.0A$
ΔV_{OUT}	Load Regulation	-	-	47	mV	$I_{OUT}=5mA-1.0A$
		-	-	24	mV	$I_{OUT}=0.25A-0.75A$
ΔV_{OUT}	Line Regulation	-	-	47	mV	$7.2V \leq V_{IN} \leq 19.7V$
		-	-	47	mV	$7.2V \leq V_{IN} \leq 19.7V$, $I_{OUT}=1.0A$
I_Q	Quiescent Current	-	-	8.0	mA	$I_{OUT} \leq 1.0A$
ΔI_Q	Quiescent Current Change	-	-	1.0	mA	$7.2V \leq V_{IN} \leq 19.7V$
		-	-	0.5	mA	$I_{OUT}=5mA-1.0A$
e_N	Output Noise Voltage	-	40	-	μV	$10Hz \leq f \leq 100KHz$
$\Delta V_o/\Delta T$	Temperature coefficient of V_{OUT}	-	-0.6	-	$mV/^\circ C$	$I_{OUT}=5mA$
RR	Ripple Rejection	62	80	-	dB	$7.7V \leq V_{IN} \leq 17.7V$, $f=120Hz$
I_{PEAK}	Peak Output Current	-	1.8	-	A	-
I_{sc}	Short-Circuit Current	-	250	-	mA	$V_{IN}=35V$
V_D	Dropout Voltage	-	2.0	-	V	-

Typical Application



Note: Bypass capacitors are recommended for optimum stability and transient response and should be located as close as possible to the regulators.

1.11 PIC 18F8520



PIC18F6520/8520/6620/ 8620/6720/8720

64/80-Pin High-Performance, 256 Kbit to 1 Mbit Enhanced Flash Microcontrollers with A/D

High-Performance RISC CPU:

- C compiler optimized architecture/instruction set:
 - Source code compatible with the PIC16 and PIC17 instruction sets
- Linear program memory addressing to 128 Kbytes
- Linear data memory addressing to 3840 bytes
- 1 Kbyte of data EEPROM
- Up to 10 MIPS operation:
 - DC – 40 MHz osc./clock input
 - 4 MHz – 10 MHz osc./clock input with PLL active
- 16-bit wide instructions, 8-bit wide data path
- Priority levels for interrupts
- 31-level, software accessible hardware stack
- 8 x 8 Single Cycle Hardware Multiplier

External Memory Interface

(PIC18F8X20 Devices Only):

- Address capability of up to 2 Mbytes
- 16-bit interface

Peripheral Features:

- High current sink/source 25 mA/25 mA
- Four external interrupt pins
- Timer0 module: 8-bit/16-bit timer/counter
- Timer1 module: 16-bit timer/counter
- Timer2 module: 8-bit timer/counter
- Timer3 module: 16-bit timer/counter
- Timer4 module: 8-bit timer/counter
- Secondary oscillator clock option – Timer1/Timer3
- Five Capture/Compare/PWM (CCP) modules:
 - Capture is 16-bit, max. resolution 6.25 ns (Tcy/16)
 - Compare is 16-bit, max. resolution 100 ns (Tcy)
 - PWM output: PWM resolution is 1 to 10-bit
- Master Synchronous Serial Port (MSSP) module with two modes of operation:
 - 3-wire SPI™ (supports all 4 SPI modes)
 - I²C™ Master and Slave mode
- Two Addressable USART modules:
 - Supports RS-485 and RS-232
- Parallel Slave Port (PSP) module

Analog Features:

- 10-bit, up to 16-channel Analog-to-Digital Converter (A/D):
 - Conversion available during Sleep
- Programmable 16-level Low-Voltage Detection (LVD) module:
 - Supports interrupt on Low-Voltage Detection
- Programmable Brown-out Reset (PBOR)
- Dual analog comparators:
 - Programmable input/output configuration

Special Microcontroller Features:

- 100,000 erase/write cycle Enhanced Flash program memory typical
- 1,000,000 erase/write cycle Data EEPROM memory typical
- 1 second programming time
- Flash/Data EEPROM Retention: > 40 years
- Self-reprogrammable under software control
- Power-on Reset (POR), Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own On-Chip RC Oscillator for reliable operation
- Programmable code protection
- Power saving Sleep mode
- Selectable oscillator options including:
 - 4X Phase Lock Loop (of primary oscillator)
 - Secondary Oscillator (32 kHz) clock input
- In-Circuit Serial Programming™ (ICSP™) via two pins
- MPLAB® In-Circuit Debug (ICD) via two pins

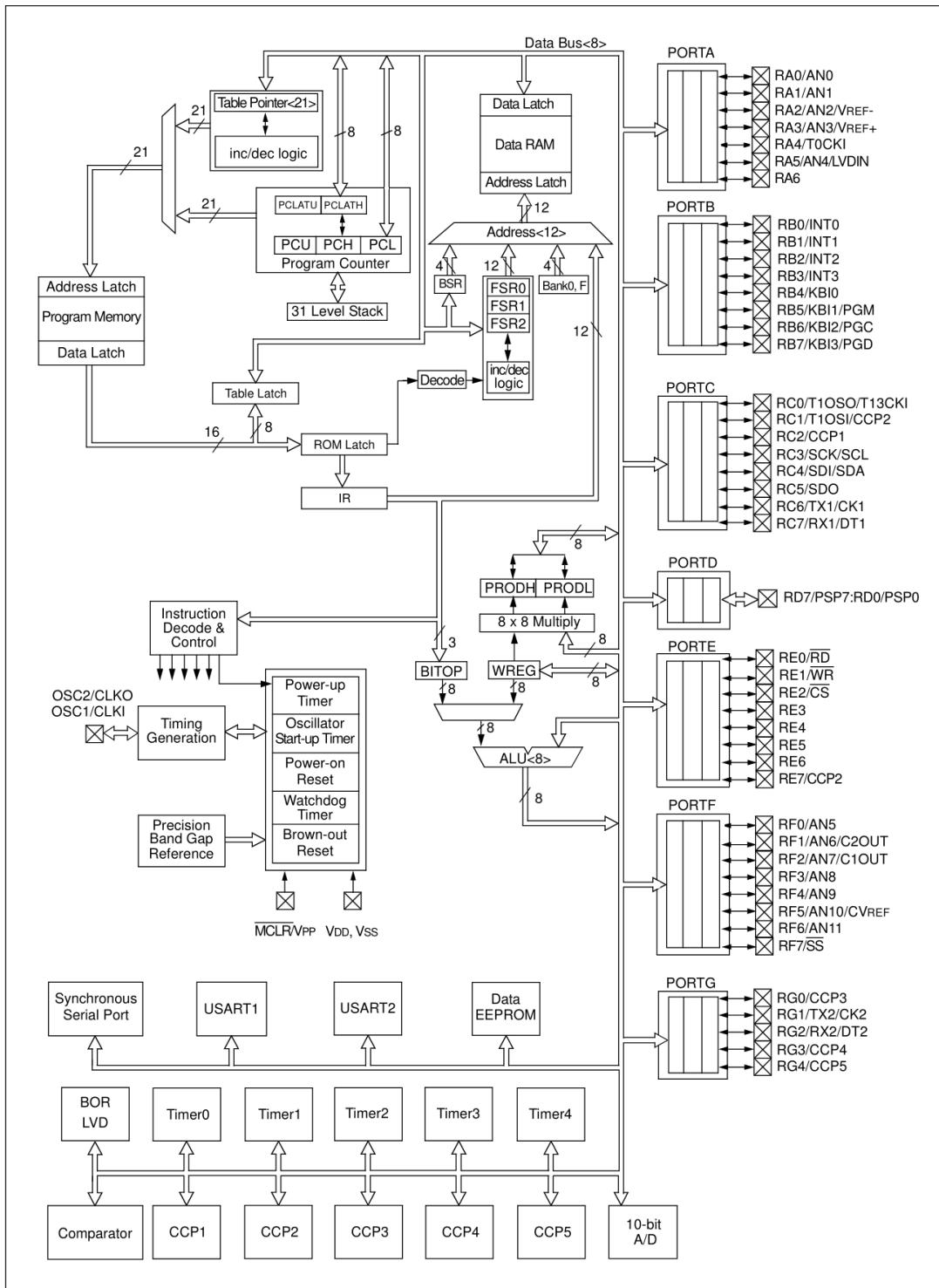
CMOS Technology:

- Low-power, high-speed Flash technology
- Fully static design
- Wide operating voltage range (2.0V to 5.5V)
- Industrial and Extended temperature ranges

Device	Program Memory		Data Memory		I/O	10-bit A/D (ch)	CCP (PWM)	MSSP		USART	Timers 8-bit/16-bit	Ext Bus	Max Fosc (MHz)
	Bytes	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)				SPI	Master I ² C				
PIC18F6520	32K	16384	2048	1024	52	12	5	Y	Y	2	2/3	N	40
PIC18F6620	64K	32768	3840	1024	52	12	5	Y	Y	2	2/3	N	25
PIC18F6720	128K	65536	3840	1024	52	12	5	Y	Y	2	2/3	N	25
PIC18F8520	32K	16384	2048	1024	68	16	5	Y	Y	2	2/3	Y	40
PIC18F8620	64K	32768	3840	1024	68	16	5	Y	Y	2	2/3	Y	25
PIC18F8720	128K	65536	3840	1024	68	16	5	Y	Y	2	2/3	Y	25

PIC18F6520/8520/6620/8620/6720/8720

FIGURE 1-1: PIC18F6X20 BLOCK DIAGRAM



PIC18F6520/8520/6620/8620/6720/8720

TABLE 4-3: REGISTER FILE SUMMARY

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Details on page:
TOSU	—	—	—	—	Top-of-Stack Upper Byte (TOS<20:16>)				--- 0 0000	32, 42
TOSH	Top-of-Stack High Byte (TOS<15:8>)								0000 0000	32, 42
Tosl	Top-of-Stack Low Byte (TOS<7:0>)								0000 0000	32, 42
STKPTR	STKFUL	STKUNF	—	—	Return Stack Pointer				00-0 0000	32, 43
PCLATU	—	—	bit 21	—	Holding Register for PC<20:16>				--10 0000	32, 44
PCLATH	Holding Register for PC<15:8>								0000 0000	32, 44
PCL	PC Low Byte (PC<7:0>)								0000 0000	32, 44
TBLPTRU	—	—	bit 21 ⁽²⁾	—	Program Memory Table Pointer Upper Byte (TBLPTR<20:16>)				--00 0000	32, 64
TBLPTRH	Program Memory Table Pointer High Byte (TBLPTR<15:8>)								0000 0000	32, 64
TBLPTRL	Program Memory Table Pointer Low Byte (TBLPTR<7:0>)								0000 0000	32, 64
TABLAT	Program Memory Table Latch								0000 0000	32, 64
PRODH	Product Register High Byte								xxxx xxxx	32, 85
PRODL	Product Register Low Byte								xxxx xxxx	32, 85
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 0000	32, 89
INTCON2	RBU	INTEDG0	INTEDG1	INTEDG2	INTEDG3	TMR0IP	INT3IP	RBIP	1111 1111	32, 90
INTCON3	INT2IP	INT1IP	INT3IE	INT2IE	INT1IE	INT3IF	INT2IF	INT1IF	1100 0000	32, 91
INDF0	Uses contents of FSR0 to address data memory – value of FSR0 not changed (not a physical register)								n/a	57
POSTINC0	Uses contents of FSR0 to address data memory – value of FSR0 post-incremented (not a physical register)								n/a	57
POSTDEC0	Uses contents of FSR0 to address data memory – value of FSR0 post-decremented (not a physical register)								n/a	57
PREINC0	Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register)								n/a	57
PLUSW0	Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) – value of FSR0 offset by value in WREG								n/a	57
FSR0H	—	—	—	—	Indirect Data Memory Address Pointer 0 High Byte				---- 0000	32, 57
FSR0L	Indirect Data Memory Address Pointer 0 Low Byte								xxxx xxxx	32, 57
WREG	Working Register								xxxx xxxx	32
INDF1	Uses contents of FSR1 to address data memory – value of FSR1 not changed (not a physical register)								n/a	57
POSTINC1	Uses contents of FSR1 to address data memory – value of FSR1 post-incremented (not a physical register)								n/a	57
POSTDEC1	Uses contents of FSR1 to address data memory – value of FSR1 post-decremented (not a physical register)								n/a	57
PREINC1	Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register)								n/a	57
PLUSW1	Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) – value of FSR1 offset by value in WREG								n/a	57
FSR1H	—	—	—	—	Indirect Data Memory Address Pointer 1 High Byte				---- 0000	33, 57
FSR1L	Indirect Data Memory Address Pointer 1 Low Byte								xxxx xxxx	33, 57
BSR	—	—	—	—	Bank Select Register				---- 0000	33, 56
INDF2	Uses contents of FSR2 to address data memory – value of FSR2 not changed (not a physical register)								n/a	57
POSTINC2	Uses contents of FSR2 to address data memory – value of FSR2 post-incremented (not a physical register)								n/a	57
POSTDEC2	Uses contents of FSR2 to address data memory – value of FSR2 post-decremented (not a physical register)								n/a	57

Legend: x = unknown, u = unchanged, - = unimplemented, q = value depends on condition

Note 1: RA6 and associated bits are configured as port pins in RCIO and ECIO Oscillator modes only and read '0' in all other oscillator modes.

2: Bit 21 of the TBLPTRU allows access to the device configuration bits.

3: These registers are unused on PIC18F6X20 devices; always maintain these clear.

PIC18F6520/8520/6620/8620/6720/8720

TABLE 4-3: REGISTER FILE SUMMARY (CONTINUED)

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Details on page:
PREINC2	Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register)								n/a	57
PLUSW2	Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) – value of FSR2 offset by value in WREG								n/a	57
FSR2H	—	—	—	—	Indirect Data Memory Address Pointer 2 High Byte					---- 0000 33, 57
FSR2L	Indirect Data Memory Address Pointer 2 Low Byte								xxxx xxxx	33, 57
STATUS	—	—	—	N	OV	Z	DC	C	---x xxxx	33, 59
TMR0H	Timer0 Register High Byte								0000 0000	33, 133
TMR0L	Timer0 Register Low Byte								xxxx xxxx	33, 133
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	1111 1111	33, 131
OSCCON	—	—	—	—	—	—	—	SCS	---- ---0	25, 33
LVDCON	—	—	IRVST	LVDEN	LVDL3	LVDL2	LVDL1	LVDL0	--00 0101	33, 235
WDTCON	—	—	—	—	—	—	—	SWDTE	---- ---0	33, 250
RCON	IPEN	—	—	RI	TO	PD	POR	BOR	0--1 1lqq	33, 60, 101
TMR1H	Timer1 Register High Byte								xxxx xxxx	33, 135
TMR1L	Timer1 Register Low Byte								xxxx xxxx	33, 135
T1CON	RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	0-00 0000	33, 135
TMR2	Timer2 Register								0000 0000	33, 141
PR2	Timer2 Period Register								1111 1111	33, 142
T2CON	—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	33, 141
SSPBUF	SSP Receive Buffer/Transmit Register								xxxx xxxx	33, 157
SSPADD	SSP Address Register in I ² C Slave mode, SSP Baud Rate Reload Register in I ² C Master mode.								0000 0000	33, 166
SSPSTAT	SMP	CKE	D/A	P	S	R/W	UA	BF	0000 0000	33, 158
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	33, 168
SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000	33, 169
ADRESH	A/D Result Register High Byte								xxxx xxxx	34, 215
ADRESL	A/D Result Register Low Byte								xxxx xxxx	34, 215
ADCON0	—	—	CHS3	CHS2	CHS1	CHS0	GO/DONE	ADON	--00 0000	34, 213
ADCON1	—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	--00 0000	34, 214
ADCON2	ADFM	—	—	—	—	ADCS2	ADCS1	ADCS0	0--- -000	34, 215
CCP1RH	Capture/Compare/PWM Register 1 High Byte								xxxx xxxx	34, 151, 152
CCP1RL	Capture/Compare/PWM Register 1 Low Byte								xxxx xxxx	34, 151, 152
CCP1CON	—	—	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	34, 149
CCP2RH	Capture/Compare/PWM Register 2 High Byte								xxxx xxxx	34, 151, 152
CCP2RL	Capture/Compare/PWM Register 2 Low Byte								xxxx xxxx	34, 151, 152
CCP2CON	—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	34, 149

Legend: x = unknown, u = unchanged, - = unimplemented, q = value depends on condition

Note 1: RA6 and associated bits are configured as port pins in RCIO and ECIO Oscillator modes only and read '0' in all other oscillator modes.

2: Bit 21 of the TBLPTRU allows access to the device configuration bits.

3: These registers are unused on PIC18F6X20 devices; always maintain these clear.

PIC18F6520/8520/6620/8620/6720/8720

TABLE 4-3: REGISTER FILE SUMMARY (CONTINUED)

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Details on page:					
CCPR3H	Capture/Compare/PWM Register 3 High Byte								xxxxx xxxx	34, 151, 152					
CCPR3L	Capture/Compare/PWM Register 3 Low Byte								xxxxx xxxx	34, 151, 152					
CCP3CON	—	—	DC3B1	DC3B0	CCP3M3	CCP3M2	CCP3M1	CCP3M0	--00 0000	34, 149					
CVRCON	CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0	0000 0000	34, 229					
CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	0000 0000	34, 223					
TMR3H	Timer3 Register High Byte								xxxxx xxxx	34, 143					
TMR3L	Timer3 Register Low Byte								xxxxx xxxx	34, 143					
T3CON	RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	<u>T3SYNC</u>	TMR3CS	TMR3ON	0000 0000	34, 143					
PSPCON	IBF	OBF	IBOV	PSPMODE	—	—	—	—	0000 ----	34, 129					
SPBRG1	USART1 Baud Rate Generator								0000 0000	34, 205					
RCREG1	USART1 Receive Register								0000 0000	34, 206					
TXREG1	USART1 Transmit Register								0000 0000	34, 204					
TXSTA1	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	34, 198					
RCSTA1	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	34, 199					
EEADRH	—	—	—	—	—	—	EE Addr Register High	-----	----000	34, 79					
EEADR	Data EEPROM Address Register								0000 0000	34, 79					
EEDATA	Data EEPROM Data Register								0000 0000	34, 79					
EECON2	Data EEPROM Control Register 2 (not a physical register)								-----	34, 79					
EECON1	EEPGD	CFG8	—	FREE	WRERR	WREN	WR	RD	xx-0 x000	34, 80					
IPR3	—	—	RC2IP	TX2IP	TMR4IP	CCP5IP	CCP4IP	CCP3IP	--11 1111	35, 100					
PIR3	—	—	RC2IF	TX2IF	TMR4IF	CCP5IF	CCP4IF	CCP3IF	--00 0000	35, 94					
PIE3	—	—	RC2IE	TX2IE	TMR4IE	CCP5IE	CCP4IE	CCP3IE	--00 0000	35, 97					
IPR2	—	CMIP	—	EEIP	BCLIP	LVDIP	TMR3IP	CCP2IP	-1-1 1111	35, 99					
PIR2	—	CMIF	—	EEIF	BCLIF	LVDIF	TMR3IF	CCP2IF	-0-0 0000	35, 93					
PIE2	—	CMIE	—	EEIE	BCLIE	LVDIE	TMR3IE	CCP2IE	-0-0 0000	35, 96					
IPR1	PSPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0111 1111	35, 98					
PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	35, 92					
PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	35, 95					
MEMCON ⁽³⁾	EBDIS	—	WAIT1	WAIT0	—	—	WM1	WM0	0-00 --00	35, 71					
TRISJ ⁽³⁾	Data Direction Control Register for PORTJ								1111 1111	35, 125					
TRISH ⁽³⁾	Data Direction Control Register for PORTH								1111 1111	35, 122					
TRISG	—	—	—	Data Direction Control Register for PORTG					---1 1111	35, 120					
TRISF	Data Direction Control Register for PORTF								1111 1111	35, 117					
TRISE	Data Direction Control Register for PORTE								1111 1111	35, 114					
TRISD	Data Direction Control Register for PORTD								1111 1111	35, 111					
TRISC	Data Direction Control Register for PORTC								1111 1111	35, 109					
TRISB	Data Direction Control Register for PORTB								1111 1111	35, 106					
TRISA	—	TRISA6 ⁽¹⁾	Data Direction Control Register for PORTA					-----	-111 1111	35, 103					

Legend: x = unknown, u = unchanged, - = unimplemented, q = value depends on condition

Note 1: RA6 and associated bits are configured as port pins in RCIO and ECIO Oscillator modes only and read '0' in all other oscillator modes.

2: Bit 21 of the TBLPTRU allows access to the device configuration bits.

3: These registers are unused on PIC18F6X20 devices; always maintain these clear.

PIC18F6520/8520/6620/8620/6720/8720

TABLE 4-3: REGISTER FILE SUMMARY (CONTINUED)

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Details on page:
LATJ ⁽³⁾									xxxx xxxx	35, 125
LATH ⁽³⁾									xxxx xxxx	35, 122
LATG	—	—	—		Read PORTG Data Latch, Write PORTG Data Latch				---x xxxx	35, 120
LATF					Read PORTF Data Latch, Write PORTF Data Latch				xxxx xxxx	35, 117
LATE					Read PORTE Data Latch, Write PORTE Data Latch				xxxx xxxx	35, 114
LATD					Read PORTD Data Latch, Write PORTD Data Latch				xxxx xxxx	35, 111
LATC					Read PORTC Data Latch, Write PORTC Data Latch				xxxx xxxx	35, 109
LATB					Read PORTB Data Latch, Write PORTB Data Latch				xxxx xxxx	35, 106
LATA	—	LATA6 ⁽¹⁾			Read PORTA Data Latch, Write PORTA Data Latch ⁽¹⁾				-xxx xxxx	35, 103
PORTJ ⁽³⁾					Read PORTJ pins, Write PORTJ Data Latch				xxxx xxxx	36, 125
PORTH ⁽³⁾					Read PORTH pins, Write PORTH Data Latch				xxxx xxxx	36, 122
PORTG	—	—	—		Read PORTG pins, Write PORTG Data Latch				---x xxxx	36, 120
PORTF					Read PORTF pins, Write PORTF Data Latch				xxxx xxxx	36, 117
PORTE					Read PORTE pins, Write PORTE Data Latch				xxxx xxxx	36, 114
PORTD					Read PORTD pins, Write PORTD Data Latch				xxxx xxxx	36, 111
PORTC					Read PORTC pins, Write PORTC Data Latch				xxxx xxxx	36, 109
PORTB					Read PORTB pins, Write PORTB Data Latch				xxxx xxxx	36, 106
PORTA	—	RA6 ⁽¹⁾			Read PORTA pins, Write PORTA Data Latch ⁽¹⁾				-x0x 0000	36, 103
TMR4									0000 0000	36, 148
PR4									1111 1111	36, 148
T4CON	—	T4OUTPS3	T4OUTPS2	T4OUTPS1	T4OUTPS0	TMR4ON	T4CKPS1	T4CKPS0	-000 0000	36, 147
CCPR4H									xxxx xxxx	36, 151, 152
CCPR4L									xxxx xxxx	36, 151, 152
CCP4CON	—	—	DC4B1	DC4B0	CCP4M3	CCP4M2	CCP4M1	CCP4M0	0000 0000	36, 149
CCPR5H									xxxx xxxx	36, 151, 152
CCPR5L									xxxx xxxx	36, 151, 152
CCP5CON	—	—	DC5B1	DC5B0	CCP5M3	CCP5M2	CCP5M1	CCP5M0	0000 0000	36, 149
SPBRG2									0000 0000	36, 205
RCREG2									0000 0000	36, 206
TXREG2									0000 0000	36, 204
TXSTA2	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	36, 198
RCSTA2	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	36, 199

Legend: x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

Note 1: RA6 and associated bits are configured as port pins in RCIO and ECIO Oscillator modes only and read '0' in all other oscillator modes.

2: Bit 21 of the TBLPTRU allows access to the device configuration bits.

3: These registers are unused on PIC18F6X20 devices; always maintain these clear.

PIC18F6520/8520/6620/8620/6720/8720

10.0 I/O PORTS

Depending on the device selected, there are either seven or nine I/O ports available on PIC18FXX20 devices. Some of their pins are multiplexed with one or more alternate functions from the other peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin.

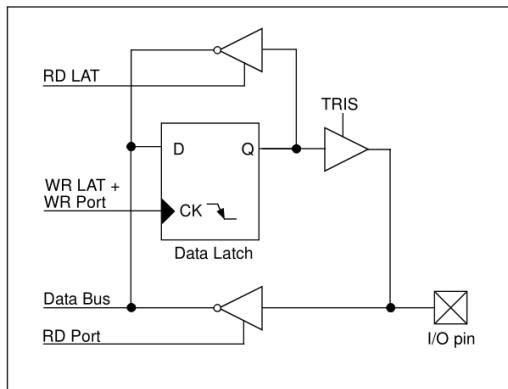
Each port has three registers for its operation. These registers are:

- TRIS register (data direction register)
- PORT register (reads the levels on the pins of the device)
- LAT register (output latch)

The Data Latch (LAT register) is useful for read-modify-write operations on the value that the I/O pins are driving.

A simplified version of a generic I/O port and its operation is shown in Figure 10-1.

FIGURE 10-1: SIMPLIFIED BLOCK DIAGRAM OF PORT/LAT/TRIS OPERATION



10.1 PORTA, TRISA and LATA Registers

PORTA is a 7-bit wide, bidirectional port. The corresponding data direction register is TRISA. Setting a TRISA bit (= 1) will make the corresponding PORTA pin an input (i.e., put the corresponding output driver in a high-impedance mode). Clearing a TRISA bit (= 0) will make the corresponding PORTA pin an output (i.e., put the contents of the output latch on the selected pin).

Reading the PORTA register reads the status of the pins, whereas writing to it will write to the port latch.

The Data Latch register (LATA) is also memory mapped. Read-modify-write operations on the LATA register, read and write the latched output value for PORTA.

The RA4 pin is multiplexed with the Timer0 module clock input to become the RA4/T0CKI pin. The RA4/T0CKI pin is a Schmitt Trigger input and an open-drain output. All other RA port pins have TTL input levels and full CMOS output drivers.

The RA6 pin is only enabled as a general I/O pin in ECIO and RCIO Oscillator modes.

The other PORTA pins are multiplexed with analog inputs and the analog VREF+ and VREF- inputs. The operation of each pin is selected by clearing/setting the control bits in the ADCON1 register (A/D Control Register 1).

Note: On a Power-on Reset, RA5 and RA3:RA0 are configured as analog inputs and read as '0'. RA6 and RA4 are configured as digital inputs.

The TRISA register controls the direction of the RA pins, even when they are being used as analog inputs. The user must ensure the bits in the TRISA register are maintained set when using them as analog inputs.

EXAMPLE 10-1: INITIALIZING PORTA

```

CLRF    PORTA ; Initialize PORTA by
               ; clearing output
               ; data latches
CLRF    LATA ; Alternate method
               ; to clear output
               ; data latches
MOVLW  0x0F ; Configure A/D
MOVWF  ADCON1 ; for digital inputs
MOVLW  0xCF ; Value used to
               ; initialize data
               ; direction
MOVWF  TRISA ; Set RA<3:0> as inputs
               ; RA<5:4> as outputs

```

PIC18F6520/8520/6620/8620/6720/8720

17.0 MASTER SYNCHRONOUS SERIAL PORT (MSSP) MODULE

17.1 Master SSP (MSSP) Module Overview

The Master Synchronous Serial Port (MSSP) module is a serial interface, useful for communicating with other peripheral or microcontroller devices. These peripheral devices may be serial EEPROMs, shift registers, display drivers, A/D converters, etc. The MSSP module can operate in one of two modes:

- Serial Peripheral Interface (SPI)
- Inter-Integrated Circuit (I^2C)
 - Full Master mode
 - Slave mode (with general address call)

The I^2C interface supports the following modes in hardware:

- Master mode
- Multi-Master mode
- Slave mode

17.2 Control Registers

The MSSP module has three associated registers. These include a status register (SSPSTAT) and two control registers (SSPCON1 and SSPCON2). The use of these registers and their individual configuration bits differ significantly, depending on whether the MSSP module is operated in SPI or I^2C mode.

Additional details are provided under the individual sections.

17.3 SPI Mode

The SPI mode allows 8 bits of data to be synchronously transmitted and received simultaneously. All four modes of SPI are supported. To accomplish communication, typically three pins are used:

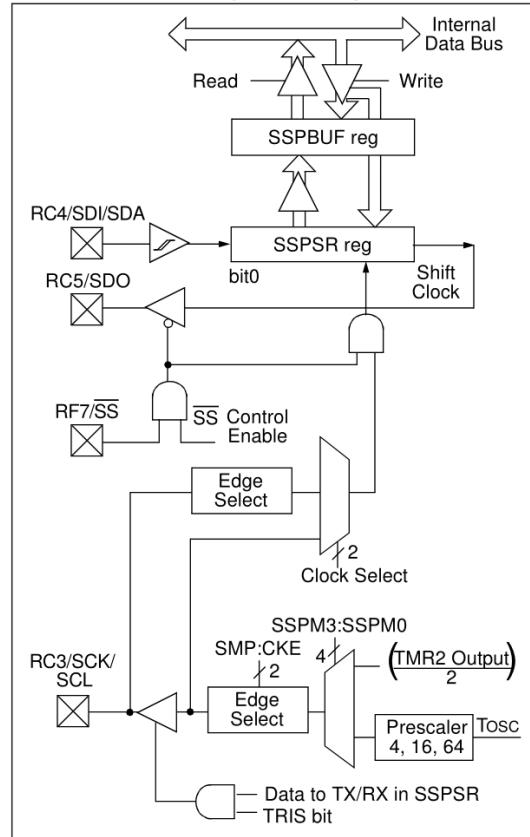
- Serial Data Out (SDO) – RC5/SDO
- Serial Data In (SDI) – RC4/SDI/SDA
- Serial Clock (SCK) – RC3/SCK/SCL

Additionally, a fourth pin may be used when in a Slave mode of operation:

- Slave Select (SS) – RF7/SS

Figure 17-1 shows the block diagram of the MSSP module when operating in SPI mode.

FIGURE 17-1: MSSP BLOCK DIAGRAM (SPI MODE)



PIC18F6520/8520/6620/8620/6720/8720

17.3.1 REGISTERS

The MSSP module has four registers for SPI mode operation. These are:

- MSSP Control Register 1 (SSPCON1)
- MSSP Status Register (SSPSTAT)
- Serial Receive/Transmit Buffer (SSPBUF)
- MSSP Shift Register (SSPSR) – Not directly accessible

SSPCON1 and SSPSTAT are the control and status registers in SPI mode operation. The SSPCON1 register is readable and writable. The lower 6 bits of the SSPSTAT are read-only. The upper two bits of the SSPSTAT are read/write.

SSPSR is the shift register used for shifting data in or out. SSPBUF is the buffer register to which data bytes are written to or read from.

In receive operations, SSPSR and SSPBUF together create a double-buffered receiver. When SSPSR receives a complete byte, it is transferred to SSPBUF and the SSPIF interrupt is set.

During transmission, the SSPBUF is not double-buffered. A write to SSPBUF will write to both SSPBUF and SSPSR.

REGISTER 17-1: SSPSTAT: MSSP STATUS REGISTER (SPI MODE)

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE	D/A	P	S	R/W	UA	BF
bit 7							
bit 0							

- bit 7 **SMP:** Sample bit
SPI Master mode:
 1 = Input data sampled at end of data output time
 0 = Input data sampled at middle of data output time
SPI Slave mode:
 SMP must be cleared when SPI is used in Slave mode.
- bit 6 **CKE:** SPI Clock Select bit
 1 = Transmit occurs on transition from active to Idle clock state
 0 = Transmit occurs on transition from Idle to active clock state
Note: Polarity of clock state is set by the CKP bit (SSPCON1<4>).
- bit 5 **D/A:** Data/Address bit
 Used in I²C mode only.
- bit 4 **P:** Stop bit
 Used in I²C mode only. This bit is cleared when the MSSP module is disabled, SSPEN is cleared.
- bit 3 **S:** Start bit
 Used in I²C mode only.
- bit 2 **R/W:** Read/Write bit information
 Used in I²C mode only.
- bit 1 **UA:** Update Address bit
 Used in I²C mode only.
- bit 0 **BF:** Buffer Full Status bit (Receive mode only)
 1 = Receive complete, SSPBUF is full
 0 = Receive not complete, SSPBUF is empty

Legend:

R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared x = Bit is unknown

PIC18F6520/8520/6620/8620/6720/8720

REGISTER 17-2: SSPCON1: MSSP CONTROL REGISTER1 (SPI MODE)

| R/W-0 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| WCOL | SSPOV | SSPEN | CKP | SSPM3 | SSPM2 | SSPM1 | SSPM0 |
| bit 7 | | | | bit 0 | | | |

- bit 7 **WCOL:** Write Collision Detect bit (Transmit mode only)
 1 = The SSPBUF register is written while it is still transmitting the previous word
 (must be cleared in software)
 0 = No collision
- bit 6 **SSPOV:** Receive Overflow Indicator bit
SPI Slave mode:
 1 = A new byte is received while the SSPBUF register is still holding the previous data. In case of overflow, the data in SSPSR is lost. Overflow can only occur in Slave mode. The user must read the SSPBUF, even if only transmitting data, to avoid setting overflow
 (must be cleared in software).
 0 = No overflow
Note: In Master mode, the overflow bit is not set, since each new reception (and transmission) is initiated by writing to the SSPBUF register.
- bit 5 **SSPEN:** Synchronous Serial Port Enable bit
 1 = Enables serial port and configures SCK, SDO, SDI and \overline{SS} as serial port pins
 0 = Disables serial port and configures these pins as I/O port pins
Note: When enabled, these pins must be properly configured as input or output.
- bit 4 **CKP:** Clock Polarity Select bit
 1 = Idle state for clock is a high level
 0 = Idle state for clock is a low level
- bit 3-0 **SSPM3:SSPM0:** Synchronous Serial Port Mode Select bits
 0101 = SPI Slave mode, clock = SCK pin, \overline{SS} pin control disabled, \overline{SS} can be used as I/O pin
 0100 = SPI Slave mode, clock = SCK pin, \overline{SS} pin control enabled
 0011 = SPI Master mode, clock = TMR2 output/2
 0010 = SPI Master mode, clock = Fosc/64
 0001 = SPI Master mode, clock = Fosc/16
 0000 = SPI Master mode, clock = Fosc/4
Note: Bit combinations not specifically listed here are either reserved, or implemented in I²C mode only.

Legend:

R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared x = Bit is unknown

PIC18F6520/8520/6620/8620/6720/8720

17.3.2 OPERATION

When initializing the SPI, several options need to be specified. This is done by programming the appropriate control bits (SSPCON1<5:0> and SSPSTAT<7:6>). These control bits allow the following to be specified:

- Master mode (SCK is the clock output)
- Slave mode (SCK is the clock input)
- Clock Polarity (Idle state of SCK)
- Data input sample phase (middle or end of data output time)
- Clock edge (output data on rising/falling edge of SCK)
- Clock Rate (Master mode only)
- Slave Select mode (Slave mode only)

The MSSP consists of a Transmit/Receive Shift Register (SSPSR) and a Buffer register (SSPBUF). The SSPSR shifts the data in and out of the device, MSb first. The SSPBUF holds the data that was written to the SSPSR until the received data is ready. Once the 8 bits of data have been received, that byte is moved to the SSPBUF register. Then, the Buffer Full detect bit, BF (SSPSTAT<0>) and the interrupt flag bit, SSPIF, are set. This double-buffering of the received data (SSPBUF) allows the next byte to start reception before

reading the data that was just received. Any write to the SSPBUF register during transmission/reception of data will be ignored and the Write Collision detect bit, WCOL (SSPCON1<7>), will be set. User software must clear the WCOL bit so that it can be determined if the following write(s) to the SSPBUF register completed successfully.

When the application software is expecting to receive valid data, the SSPBUF should be read before the next byte of data to transfer is written to the SSPBUF. Buffer Full bit, BF (SSPSTAT<0>), indicates when SSPBUF has been loaded with the received data (transmission is complete). When the SSPBUF is read, the BF bit is cleared. This data may be irrelevant if the SPI is only a transmitter. Generally, the MSSP interrupt is used to determine when the transmission/reception has completed. The SSPBUF must be read and/or written. If the interrupt method is not going to be used, then software polling can be done to ensure that a write collision does not occur. Example 17-1 shows the loading of the SSPBUF (SSPSR) for data transmission.

The SSPSR is not directly readable or writable and can only be accessed by addressing the SSPBUF register. Additionally, the MSSP Status register (SSPSTAT) indicates the various status conditions.

EQUATION 17-1: LOADING THE SSPBUF (SSPSR) REGISTER

```

LOOP BTFSS SSPSTAT, BF      ;Has data been received (transmit complete)?
    BRA  LOOP            ;No
    MOVF SSPBUF, W        ;WREG reg = contents of SSPBUF
    MOVWF RXDATA          ;Save in user RAM, if data is meaningful
    MOVF TXDATA, W        ;W reg = contents of TXDATA
    MOVWF SSPBUF          ;New data to transmit

```

PIC18F6520/8520/6620/8620/6720/8720

17.3.3 ENABLING SPI I/O

To enable the serial port, SSP Enable bit, SSPEN (SSPCON1<5>), must be set. To reset or reconfigure SPI mode, clear the SSPEN bit, reinitialize the SSPCON registers and then set the SSPEN bit. This configures the SDI, SDO, SCK and SS pins as serial port pins. For the pins to behave as the serial port function, some must have their data direction bits (in the TRIS register) appropriately programmed as follows:

- SDI is automatically controlled by the SPI module
- SDO must have TRISC<5> bit cleared
- SCK (Master mode) must have TRISC<3> bit cleared
- SCK (Slave mode) must have TRISC<3> bit set
- SS must have TRISF<7> bit set

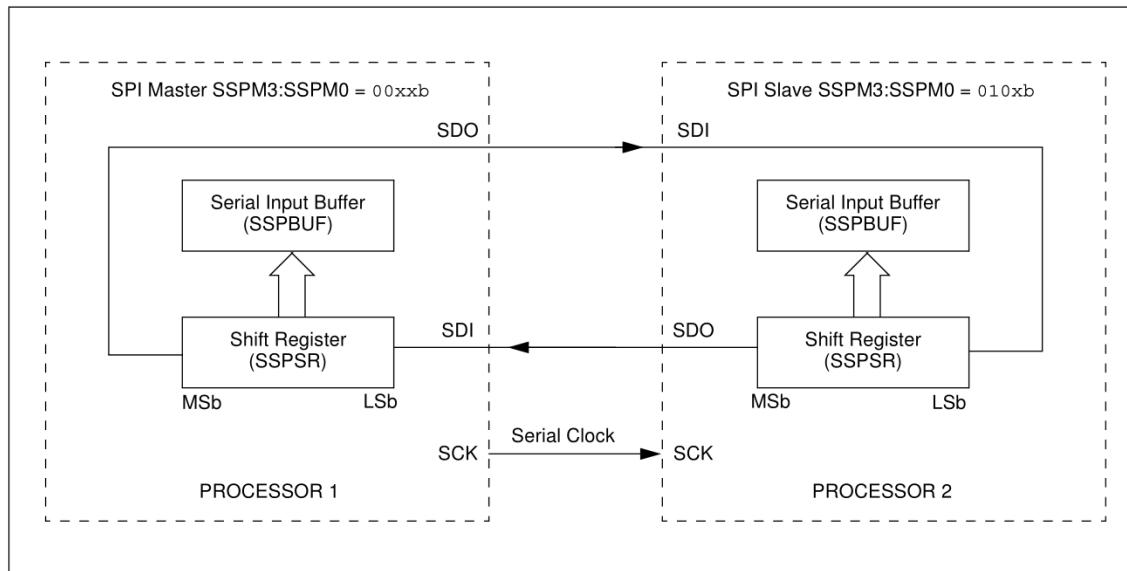
Any serial port function that is not desired may be overridden by programming the corresponding data direction (TRIS) register to the opposite value.

17.3.4 TYPICAL CONNECTION

Figure 17-2 shows a typical connection between two microcontrollers. The master controller (Processor 1) initiates the data transfer by sending the SCK signal. Data is shifted out of both shift registers on their programmed clock edge and latched on the opposite edge of the clock. Both processors should be programmed to the same Clock Polarity (CKP), then both controllers would send and receive data at the same time. Whether the data is meaningful (or dummy data) depends on the application software. This leads to three scenarios for data transmission:

- Master sends data – Slave sends dummy data
- Master sends data – Slave sends data
- Master sends dummy data – Slave sends data

FIGURE 17-2: SPI MASTER/SLAVE CONNECTION



PIC18F6520/8520/6620/8620/6720/8720

17.3.5 MASTER MODE

The master can initiate the data transfer at any time because it controls the SCK. The master determines when the slave (Processor 2, Figure 17-2) is to broadcast data by the software protocol.

In Master mode, the data is transmitted/received as soon as the SSPBUF register is written to. If the SPI is only going to receive, the SDO output could be disabled (programmed as an input). The SSPSR register will continue to shift in the signal present on the SDI pin at the programmed clock rate. As each byte is received, it will be loaded into the SSPBUF register as if a normal received byte (interrupts and status bits appropriately set). This could be useful in receiver applications as a "Line Activity Monitor" mode.

The clock polarity is selected by appropriately programming the CKP bit (SSPCON1<4>). This then, would give waveforms for SPI communication, as shown in

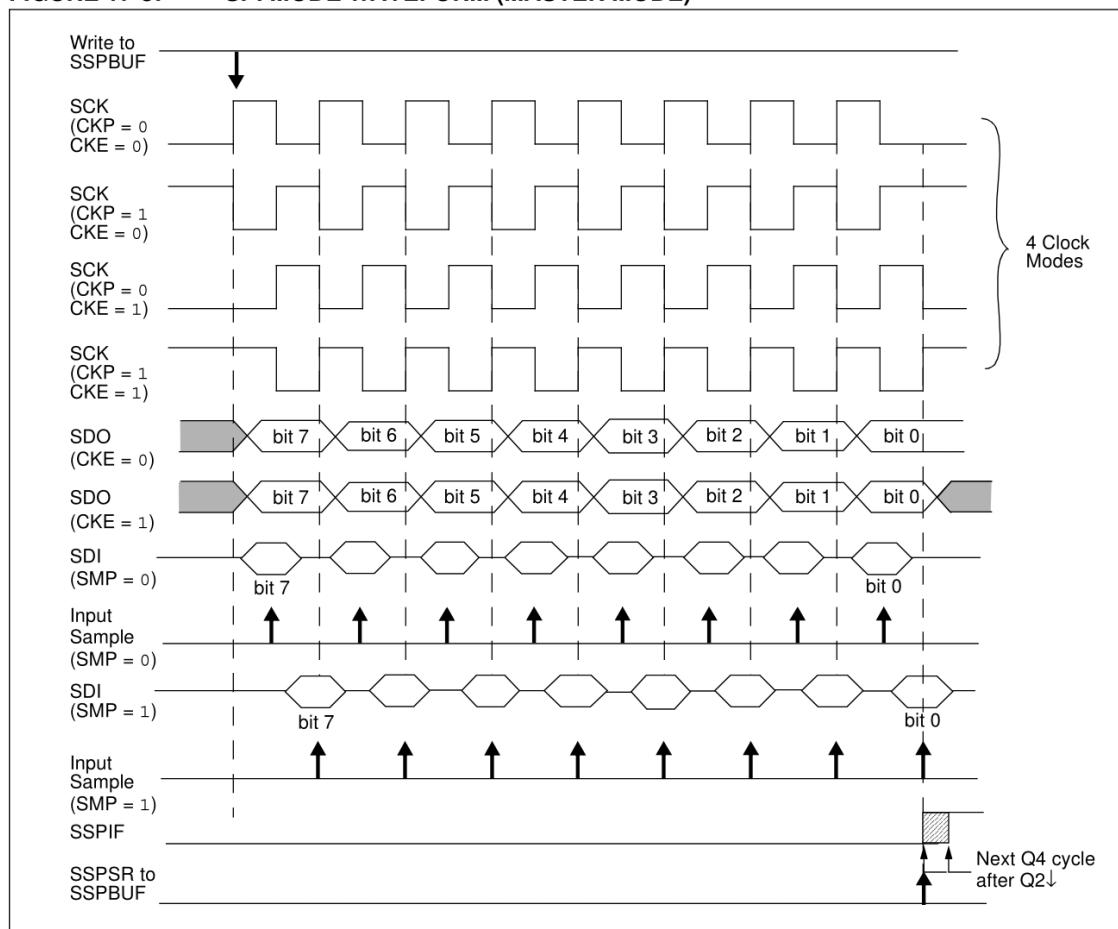
Figure 17-3, Figure 17-5 and Figure 17-6, where the MSB is transmitted first. In Master mode, the SPI clock rate (bit rate) is user-programmable to be one of the following:

- Fosc/4 (or Tcy)
- Fosc/16 (or 4 • Tcy)
- Fosc/64 (or 16 • Tcy)
- Timer2 output/2

This allows a maximum data rate (at 40 MHz) of 10.00 Mbps.

Figure 17-3 shows the waveforms for Master mode. When the CKE bit is set, the SDO data is valid before there is a clock edge on SCK. The change of the input sample is shown based on the state of the SMP bit. The time when the SSPBUF is loaded with the received data is shown.

FIGURE 17-3: SPI MODE WAVEFORM (MASTER MODE)



PIC18F6520/8520/6620/8620/6720/8720

17.3.6 SLAVE MODE

In Slave mode, the data is transmitted and received as the external clock pulses appear on SCK. When the last bit is latched, the SSPIF interrupt flag is set.

While in Slave mode, the external clock is supplied by the external clock source on the SCK pin. This external clock must meet the minimum high and low times as specified in the electrical specifications.

While in Sleep mode, the slave can transmit/receive data. When a byte is received, the device will wake-up from Sleep.

17.3.7 SLAVE SELECT SYNCHRONIZATION

The \overline{SS} pin allows a Synchronous Slave mode. The SPI must be in Slave mode with \overline{SS} pin control enabled ($SSPCON1<3:0> = 04h$). The pin must not be driven low for the \overline{SS} pin to function as an input. The Data Latch must be high. When the \overline{SS} pin is low, transmission and reception are enabled and the SDO pin is driven. When the \overline{SS} pin goes high, the SDO pin is no

longer driven, even if in the middle of a transmitted byte and becomes a floating output. External pull-up/pull-down resistors may be desirable, depending on the application.

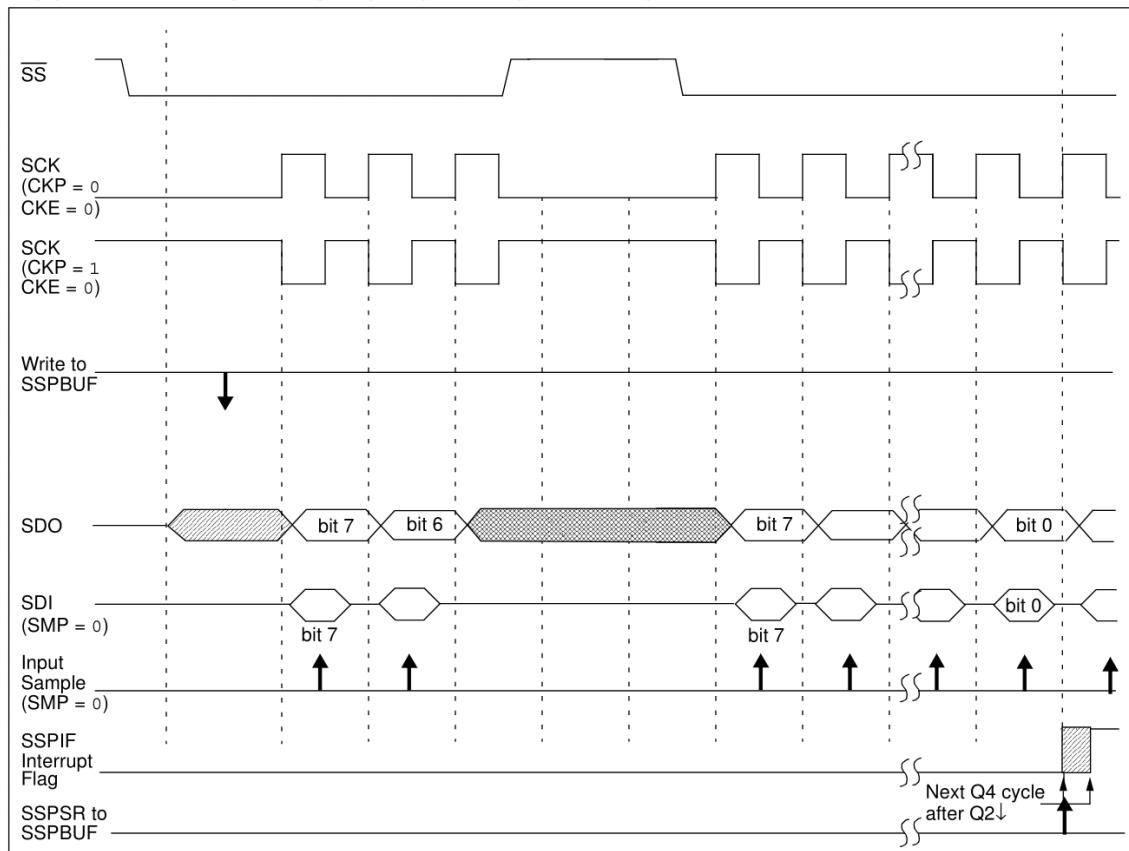
Note 1: When the SPI is in Slave mode with \overline{SS} pin control enabled ($SSPCON<3:0> = 0100$), the SPI module will reset if the SS pin is set to VDD.

2: If the SPI is used in Slave mode with CKE set, then the \overline{SS} pin control must be enabled.

When the SPI module resets, the bit counter is forced to '0'. This can be done by either forcing the \overline{SS} pin to a high level or clearing the SSPEN bit.

To emulate two-wire communication, the SDO pin can be connected to the SDI pin. When the SPI needs to operate as a receiver, the SDO pin can be configured as an input. This disables transmissions from the SDO. The SDI can always be left as an input (SDI function), since it cannot create a bus conflict.

FIGURE 17-4: SLAVE SYNCHRONIZATION WAVEFORM



PIC18F6520/8520/6620/8620/6720/8720

FIGURE 17-5: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 0)

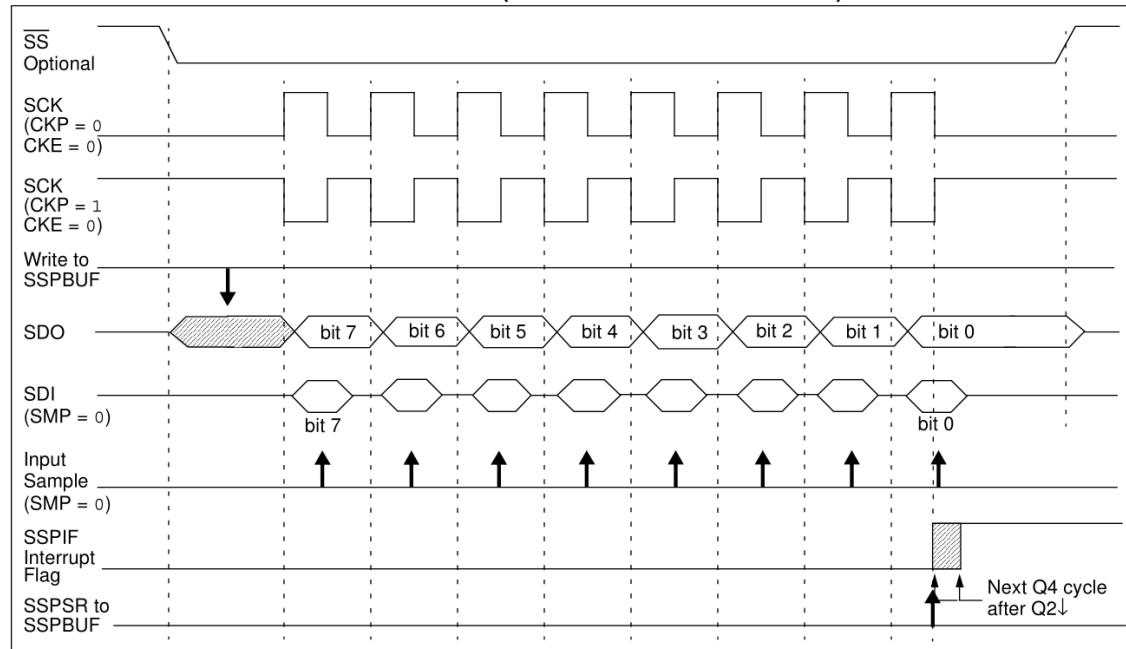
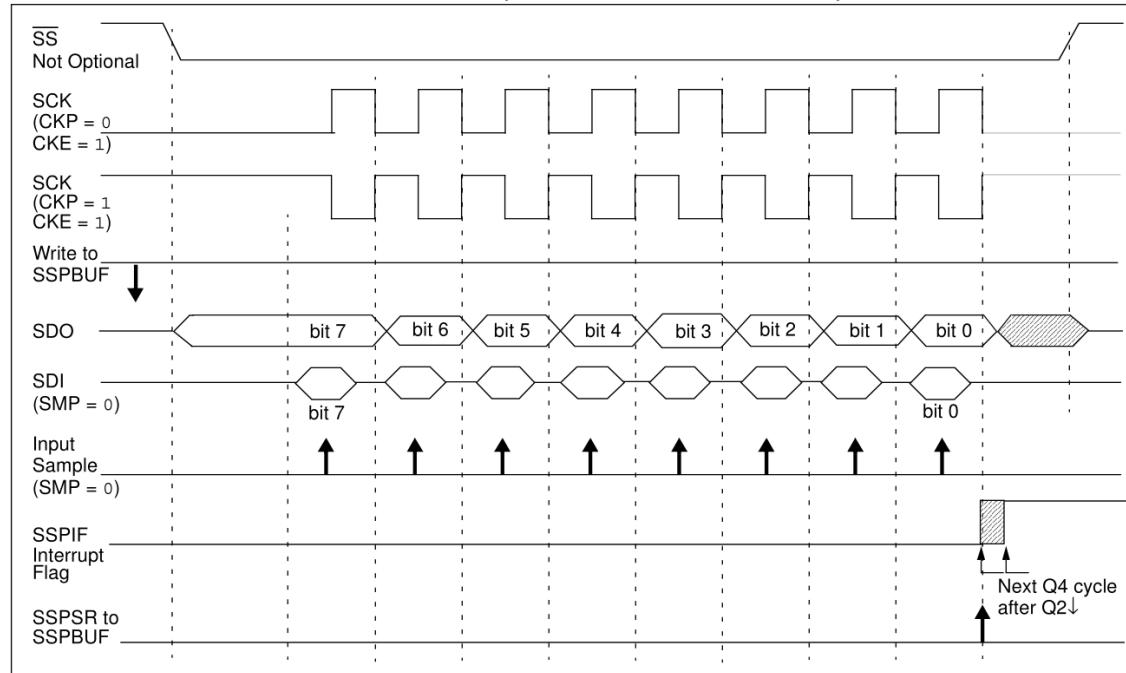


FIGURE 17-6: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 1)



2. Código utilizado

2.1 PIC 18F8520

```

// Ruben Garcia Segovia
// ADC UNIDS-3 conectado por SPI junto con la Raspberry Pi
// Simulando al conversor ADE7912.

// Librerias
#include "spi.h"
#include <delays.h>
#include <p18f8520.h>

//Configuracion del pic
#pragma config OSC= HS, WDT = OFF, LVP = OFF

//Los cuatro Bytes del ADC que se van a leer
char response1_1 = 0x00;
char response1_2 = 0x00; //Voltaje
char response1_3 = 0x00;
char response1_4 = 0x00;

char response2_1 = 0x00;
char response2_2 = 0x00; //Intensidad
char response2_3 = 0x00;
char response2_4 = 0x00;

char response3_1 = 0x00;
char response3_2 = 0x00; //Temperatura
char response3_3 = 0x00;
char response3_4 = 0x00;

//El dato recibido de la raspberry
char Raspi_spi = 0x00;
int contador = 0; //Contador = 0 para LN, 1 para L1, 2-L2, 3-L3
float sincro = 3840000; //Cada 2 minutos contabilizados...

//Son cuatro char, ya que guardamos las dos respuestas del conversor analógico digital
//Ya que tenemos dos datos a recuperar de 24 bits.

void main(void) {
    TRISDbits.TRISD0 = 1; //Aviso del ADE7912 Para saber cuando debemos pedir datos
    TRISDbits.TRISD7 = 0; //Para gobernar la raspberry pi
    TRISGbits.TRISG0 = 0; //Para gobernar el actuador
    TRISB=0xFF;
    TRISE=0X00;//gobernar todos los ADE7912
    LATE=0xFF; //Para gobernar el ADC y la raspberry;
    LATGbits.LATG7=0;

    //INICIO CONFIGURACION //

    LATDbits.LATD7=0; //Iniciamos RD7 a 0 para avisar a la raspberry pi.
    OpenSPI(SPI_FOSC_16,MODE_11,SMPEND); //Iniciamos SPI como maestro
    while(SSPBUF==1) { //Comprueba que el ADC0 esté listo y lo configura
        LATDbits.LATE2=0;
        WriteSPI(0x4C);
        ReadSPI();
        SSPBUF = SSPBUF&0x01;
        LATDbits.LATE2=1;
    }
    LATDbits.LATE2=0;
    WriteSPI(0x40); //Registro de configuracion
    WriteSPI(0x81); //8Khz, BW seleccionado, modo multiples ADC...
    WriteSPI(0xE0); //Registro EMI
    WriteSPI(0xAA);
    LATDbits.LATE2=1;

    while(SSPBUF==1) { //Comprueba que el ADC1 esté listo y lo configura
        LATDbits.LATE3=0;
        WriteSPI(0x4C);
        ReadSPI();
        SSPBUF = SSPBUF&0x01;
        LATDbits.LATE3=1;
    }
    LATDbits.LATE3=0;
}

```

```

WriteSPI(0x40); //Registro de configuracion
WriteSPI(0x81); //8Khz, BW seleccionado, modo multiples ADC...
WriteSPI(0xE0); //Registro EMI
WriteSPI(0xAA);
LATEbits.LATE3=1;

while(SSPBUF==1) { //Comprueba que el ADC2 esté listo y lo configura
    LATEbits.LATE4=0;
    WriteSPI(0x4C);
    ReadSPI();
    SSPBUF = SSPBUF&0x01;
    LATEbits.LATE4=1;
}

LATEbits.LATE4=0;
WriteSPI(0x40); //Registro de configuracion
WriteSPI(0x81); //8Khz, BW seleccionado, modo multiples ADC...
WriteSPI(0xE0); //Registro EMI
WriteSPI(0x55);
LATEbits.LATE4=1;

while(SSPBUF==1) { //Comprueba que el ADC3 esté listo y lo configura
    LATEbits.LATE5=0;
    WriteSPI(0x4C);
    ReadSPI();
    SSPBUF = SSPBUF&0x01;
    LATEbits.LATE5=1;
}

LATEbits.LATE5=0;
WriteSPI(0x40); //Registro de configuracion
WriteSPI(0x81); //8Khz, BW seleccionado, modo multiples ADC...
WriteSPI(0xE0); //Registro EMI
WriteSPI(0x55);
LATEbits.LATE5=1;

//FIN CONFIGURACION //

while(1){ //De forma indefinida...
if(sincro>= 3840000) {
    LATEbits.LATE2=0; //Llamada simultanea a todos los ADC
    LATEbits.LATE3=0;
    LATEbits.LATE4=0;
    LATEbits.LATE5=0;
    WriteSPI(0xB0); //Sync_snap
    WriteSPI(0x01); //Reset y sincronizacion
    LATEbits.LATE2=1; //Llamada simultanea a todos los ADC
    LATEbits.LATE3=1;
    LATEbits.LATE4=1;
    LATEbits.LATE5=1;
    sincro = 0;
}
while(PORTDbits.PORTD0==1||contador>0); // Hasta que no nos avise el ADC de que está
listo.
switch(contador){
    case 0:
        LATEbits.LATE2=0;
        break;
    case 1:
        LATEbits.LATE3=0;
        break;
    case 2:
        LATEbits.LATE4=0;
        break;
    default:
        LATEbits.LATE5=0;
        break;
}
WriteSPI(0x04); //IWF
ReadSPI();
responsel_1=SSPBUF;
ReadSPI();
responsel_2=SSPBUF;

```

```

ReadSPI();
responsel_3=SSPBUF;
ReadSPI();
responsel_4=SSPBUF;
WriteSPI(0x0C); //V1WV
ReadSPI();
response2_1=SSPBUF;
ReadSPI();
response2_2=SSPBUF;
ReadSPI();
response2_3=SSPBUF;
ReadSPI();
response2_4=SSPBUF;
WriteSPI(0x14); //V2WV (Temperatura)
ReadSPI();
response3_1=SSPBUF;
ReadSPI();
response3_2=SSPBUF;
ReadSPI();
response3_3=SSPBUF;
ReadSPI();
response3_4=SSPBUF;
switch(contador){
    case 0:
        LATEdots.LATE2=1;
        contador++;
        break;
    case 1:
        LATEdots.LATE3=1;
        contador++;
        break;
    case 2:
        LATEdots.LATE4=1;
        contador++;
        break;
    default:
        LATEdots.LATE5=1;
        contador = 0;
        break;
}
 ****
//          VOLTAJE      //
Nop(); //Le dejamos un poco para que no sea inmediato
Nop(); //Iniciamos SPI en modo ESCLAVO.
SSPBUF = responsel_1; //Cargamos el primer byte
Nop();
while(!SSPSTATbits.BF){ //Esperamos a que lo recojan
    LATD = 0xFF; //Activamos el PIN RASPIENABLE
}
LATD = 0X00;
Raspi_spi = SSPBUF; //Recuperamos el dato recibido

SSPBUF = response1_2; //Cargamos el segundo byte
Nop();
while(!SSPSTATbits.BF){ //Esperamos a que lo recojan
    LATD=0xFF; //Activamos el PIN RASPIENABLE
}
LATD=0X00;
Raspi_spi = SSPBUF; //Recuperamos el dato recibido

SSPBUF = response1_3; //Cargamos el tercer byte
Nop();
while(!SSPSTATbits.BF){ //Esperamos a que lo recojan
    LATD=0xFF; //Activamos el PIN RASPIENABLE
}
LATD=0X00;
Raspi_spi = SSPBUF; //Recuperamos el dato recibido

```

```

SSPBUF = response1_4; //Cargamos el cuarto byte
Nop();
while(!SSPSTATbits.BF){ //Esperamos a que lo recojan
    LATD=0xFF; //Activamos el PIN RASPIENABLE
}
LATD=0X00;
Raspi_spi = SSPBUF; //Recuperamos el dato recibido

//          INTENSIDAD           //
SSPBUF = response2_1; //Cargamos el primer byte
Nop();
while(!SSPSTATbits.BF){ //Esperamos a que lo recojan
    LATD = 0xFF; //Activamos el PIN RASPIENABLE
}
LATD = 0X00;
Raspi_spi = SSPBUF; //Recuperamos el dato recibido

SSPBUF = response2_2; //Cargamos el segundo byte
Nop();
while(!SSPSTATbits.BF){ //Esperamos a que lo recojan
    LATD=0xFF; //Activamos el PIN RASPIENABLE
}
LATD=0X00;
Raspi_spi = SSPBUF; //Recuperamos el dato recibido

SSPBUF = response2_3; //Cargamos el tercer byte
Nop();
while(!SSPSTATbits.BF){ //Esperamos a que lo recojan
    LATD=0xFF; //Activamos el PIN RASPIENABLE
}
LATD=0X00;
Raspi_spi = SSPBUF; //Recuperamos el dato recibido

SSPBUF = response2_4; //Cargamos el cuarto byte
Nop();
while(!SSPSTATbits.BF){ //Esperamos a que lo recojan
    LATD=0xFF; //Activamos el PIN RASPIENABLE
}
LATD=0X00;
Raspi_spi = SSPBUF; //Recuperamos el dato recibido

//          TEMPERATURA           //
SSPBUF = response3_1; //Cargamos el primer byte
Nop();
while(!SSPSTATbits.BF){ //Esperamos a que lo recojan
    LATD = 0xFF; //Activamos el PIN RASPIENABLE
}
LATD = 0X00;
Raspi_spi = SSPBUF; //Recuperamos el dato recibido

SSPBUF = response3_2; //Cargamos el segundo byte
Nop();
while(!SSPSTATbits.BF){ //Esperamos a que lo recojan
    LATD=0xFF; //Activamos el PIN RASPIENABLE
}
LATD=0X00;
Raspi_spi = SSPBUF; //Recuperamos el dato recibido

SSPBUF = response3_3; //Cargamos el tercer byte
Nop();
while(!SSPSTATbits.BF){ //Esperamos a que lo recojan
    LATD=0xFF; //Activamos el PIN RASPIENABLE
}
LATD=0X00;
Raspi_spi = SSPBUF; //Recuperamos el dato recibido

SSPBUF = response3_4; //Cargamos el cuarto byte
Nop();
while(!SSPSTATbits.BF){ //Esperamos a que lo recojan
    LATD=0xFF; //Activamos el PIN RASPIENABLE
}

```

```
LATD=0X00;
Raspi_spi = SSPBUF; //Recuperamos el dato recibido

// CONFIRMACION //
SSPBUF = 0xAA; //Cargamos el byte de confirmacion
Nop();
while(!SSPSTATbits.BF){ //Esperamos a que lo recojan
    LATD=0xFF; //Activamos el PIN RASPIENABLE
}
LATD=0X00;
Raspi_spi = SSPBUF; //Recuperamos el dato recibido

if (Raspi_spi == 0x10){ //Activamos el actuador si asi nos lo han pedido
    LATGbits.LATG0=1;
}
if (Raspi_spi == 0x20){ //Desactivamos el actuador si asi nos lo han pedido
    LATGbits.LATG0=0;
}
//Terminamos por ahora
CloseSPI();
}
```

2.2 spi.c

```
//Ruben Garcia Segovia - 15/05/2018
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <stdint.h>
#include <wiringPiSPI.h>
#include <wiringPi.h>
#include <time.h>

int main (void) {
    FILE *log;
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    log=fopen("../basedatos/log.txt","a"); //Guardamos el inicio del servidor
    fprintf(log,"n%d-%d-%d %d:%d:%d - Inicio del servidor",tm.tm_year + 1900, tm.tm_mon + 1, tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
    fclose(log);
    while(1){ //Bucle infinito
        FILE *fp; //Nos preparamos para abrir todos los archivos necesarios
        FILE *fp2;
        FILE *fp3;
        FILE *fp4;
        FILE *fp5;
        FILE *fp6;
        FILE *fp7;
        FILE *fp8;
        FILE *actuador;
        fp=fopen("voltajeltemporal.txt","w"); //Los archivos RAW los guardamos siempre en
        archivos temporales
        fp2=fopen("voltaje2temporal.txt","w");
        fp3=fopen("voltaje3temporal.txt","w");
        fp4=fopen("voltajentemporal.txt","w");
        fp5=fopen("intensidad1temporal.txt","w");
        fp6=fopen("intensidad2temporal.txt","w");
        fp7=fopen("intensidad3temporal.txt","w");
        fp8=fopen("temperaturatemporal.txt","w");
        fprintf(fp, "Ruben Garcia Segovia - VoltajeL1");
        fprintf(fp2, "Ruben Garcia Segovia - VoltajeL2");
        fprintf(fp3, "Ruben Garcia Segovia - VoltajeL3");
        fprintf(fp4, "Ruben Garcia Segovia - VoltajeLN");
        fprintf(fp5, "Ruben Garcia Segovia - IntensidadL1");
        fprintf(fp6, "Ruben Garcia Segovia - IntensidadL2");
        fprintf(fp7, "Ruben Garcia Segovia - IntensidadL3");
        fprintf(fp8, "Ruben Garcia Segovia - Temperatura");
        printf("Pruebas de SPI_Ruben_Garcia_Segovia\n");

        double BILLION= 1000000000;
        int actuador_memory = 0;
        struct timespec time_origen, time_actual;
        double accum=0;
        float huecoV=0;
        float huecoI=0;
        int huecoVestado=0;
        int huecoIestado=0;
        int num_adc = 0;
        char caractuador[10];
        wiringPiSetup ();
        pinMode (1, INPUT);

        if(wiringPiSPISetup(0, 1000000) < 0) //Iniciamos el SPI 500Khz, SPIO como CS
        {
            fprintf(stderr, "Error al leer el SPI: %s",strerror (errno));
            exit(1);
        }
        unsigned char ByteSPI[15]; //Variable para transmitir y recibir datos por spi
        long voltaje,intensidad;
        float voltajefinal,intensidadfinal;
        double notacion[4],notacion2[4];
        delay(1000); //1 segundo para dar tiempo a que se establezca el chip SPI
```

```

//Cargamos los datos
clock_gettime(CLOCK_MONOTONIC, &time_origen); //El reloj monotonic da siempre un valor
positivo
//e incremental en el tiempo, se utiliza para saber el tiempo que ha tardado en ejecutar
'x' lineas de codigo.
//Lo ejecutamos aqui para tomar el primer valor
while(accum<=50){ //accum son los segundos transcurridos desde el anterior clock_gettime.
Este bucle se realiza una vez por segundo
actuador=fopen("actuador.txt","r");
fgets(caractuador, 10, actuador); //Leemos el estado del actuador
fclose(actuador);
while(digitalRead(1)==LOW); //Esperamos a que el pic nos de la orden de comunicar
//Voltaje //
wiringPiSPIDataRW (0, ByteSPI, 1); //Primer Byte
ByteSPI[1] = ByteSPI[0];
while(digitalRead(1)==LOW);
wiringPiSPIDataRW (0, ByteSPI, 1); //Segundo Byte
ByteSPI[2] = ByteSPI[0];
while(digitalRead(1)==LOW);
wiringPiSPIDataRW (0, ByteSPI, 1); //Tercer Byte
ByteSPI[3] = ByteSPI[0];
while(digitalRead(1)==LOW);
wiringPiSPIDataRW (0, ByteSPI, 1); //Cuarto Byte
ByteSPI[4] = ByteSPI[0];
//Intensidad //
wiringPiSPIDataRW (0, ByteSPI, 1); //Primer Byte
ByteSPI[5] = ByteSPI[0];
while(digitalRead(1)==LOW);
wiringPiSPIDataRW (0, ByteSPI, 1); //Segundo Byte
ByteSPI[6] = ByteSPI[0];
while(digitalRead(1)==LOW);
wiringPiSPIDataRW (0, ByteSPI, 1); //Tercer Byte
ByteSPI[7] = ByteSPI[0];
while(digitalRead(1)==LOW);
wiringPiSPIDataRW (0, ByteSPI, 1); //Cuarto Byte
ByteSPI[8] = ByteSPI[0];
//Temperatura //
wiringPiSPIDataRW (0, ByteSPI, 1); //Primer Byte
ByteSPI[9] = ByteSPI[0];
while(digitalRead(1)==LOW);
wiringPiSPIDataRW (0, ByteSPI, 1); //Segundo Byte
ByteSPI[10] = ByteSPI[0];
while(digitalRead(1)==LOW);
wiringPiSPIDataRW (0, ByteSPI, 1); //Tercer Byte
ByteSPI[11] = ByteSPI[0];
while(digitalRead(1)==LOW);
wiringPiSPIDataRW (0, ByteSPI, 1); //Cuarto Byte
ByteSPI[12] = ByteSPI[0];
if (caractuador[0] == '1'){ //Si el actuador esta activado...
ByteSPI[0] = 0x10; //Se manda un 0x10 al pic (activar actuador)
if(actuador_memory==0){ //Si no estaba activado de antes...
FILE *log;
time_t t = time(NULL);
struct tm tm = *localtime(&t);
log=fopen("../basededatos/log.txt","a"); //Se apunta en el log
fprintf(log,"%d-%d-%d %d:%d:%d - Encendido del actuador",tm.tm_year + 1900,
tm.tm_mon + 1, tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
fclose(log);
}
actuador_memory = 1;
} else if(caractuador[0] == '0' && actuador_memory == 1){ //Si no esta activado y antes
si lo estaba
ByteSPI[0]= 0x20; //Se manda 0x20 al pic (Desactivar actuador)
FILE *log;
time_t t = time(NULL);
struct tm tm = *localtime(&t);
log=fopen("../basededatos/log.txt","a");//Se apunta en el log
fprintf(log,"%d-%d-%d %d:%d:%d - Apagado del actuador",tm.tm_year + 1900,
tm.tm_mon + 1, tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
fclose(log);
actuador_memory = 0;
}

```

```

} else {
    ByteSPI[0]= 0x20; //Se manda el 0x20
}
while(digitalRead(1)==LOW); //Nos esperamos a que el pic nos de permite de comunicacion
wiringPiSPIDataRW (0, ByteSPI, 1); //Byte de confirmacion y envio de actuador

//Calculo del voltaje
voltaje = ByteSPI[1] * 1024;
voltaje += ByteSPI[2] * 512;
voltaje += ByteSPI[3] * 256;
voltaje = voltaje + ByteSPI[4];
voltajefinal = (voltaje*4.096)/4096;
notacion[num_adc] = (voltajefinal * 100);

//Calculo de la intensidad
intensidad = ByteSPI[5] * 1024;
intensidad += ByteSPI[6] * 512;
intensidad += ByteSPI[7] * 256;
intensidad = intensidad + ByteSPI[8];
voltajefinal = (intensidad*4.096)/4096;
notacion2[num_adc] = (intensidadfinal * 100);

//Calculo de la temperatura
temperatura = ByteSPI[5] * 1024;
temperatura += ByteSPI[6] * 512;
temperatura += ByteSPI[7] * 256;
temperatura = temperatura + ByteSPI[8]; //Sustituir valores por los del ADC
temperaturafinal[num_adc] = ganancia[num_adc]*temperatura +
8.72101*TEMPOS[num_adc]*(2^11)*10^(-5) - 306.47

//Detectamos posibles cortes
if(huecoV>3000 && huecoVestado==0){
    huecoVestado=1;
}
if(huecoI>3000 && huecoIestado==0){
    huecoIestado=1;
} //Tres mil muestras es aproximadamente 1 segundo
if(huecoVestado==1 && huecoV >3000){
    FILE *log;
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    log=fopen("../basededatos/log.txt","a"); //Escribimos en el log el corte
    fprintf(log,"%d-%d-%d %d:%d:%d - Corte en el Voltaje",tm.tm_year + 1900, tm.tm_mon
+ 1, tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
    fclose(log);
    log=fopen("../monitorizacion/estado.txt", "w");
    fprintf(log,"1");
    fclose(log);
    huecoVestado = 2;
}
if(huecoVestado==2 && huecoV ==0){
    FILE *log;
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    log=fopen("../basededatos/log.txt","a"); //Escribimos en el log la reanudacion
    fprintf(log,"%d-%d-%d %d:%d:%d - Reanudado el Voltaje",tm.tm_year + 1900,
tm.tm_mon + 1, tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
    fclose(log);
    log=fopen("../monitorizacion/estado.txt", "w");
    fprintf(log,"0");
    fclose(log);
    huecoVestado = 0;
}
if(huecoIestado==1 && huecoI >3000){
    FILE *log;
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    log=fopen("../basededatos/log.txt","a");
    fprintf(log,"%d-%d-%d %d:%d:%d - Corte en el Intensidad",tm.tm_year + 1900,
tm.tm_mon + 1, tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
    fclose(log);
}

```

```

log=fopen("../monitorizacion/estado.txt","w");
fprintf(log,"1");
fclose(log);
huecoIestado = 2;
}
if(huecoIestado==2 && huecoI ==0){
    FILE *log;
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    log=fopen("../basededatos/log.txt","a");
    fprintf(log,"\n%d-%d-%d %d:%d:%d - Reanudada la Intensidad",tm.tm_year + 1900,
    tm.tm_mon + 1, tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
    fclose(log);
    log=fopen("../monitorizacion/estado.txt","w");
    fprintf(log,"0");
    fclose(log);
    huecoIestado = 0;
}

clock_gettime(CLOCK_MONOTONIC, &time_actual); //Tomamos el segundo valor del reloj
monotonic
//Con la siguiente ecuacion sacamos el tiempo transcurrido en notacion cientifica.
accum = (time_actual.tv_sec - time_origen.tv_sec) + (time_actual.tv_nsec -
time_origen.tv_nsec) / BILLION ;

if(ByteSPI[0]==0xAA && num_adc==3){ //Se comprueba byte de confirmacion
    printf("%f\n",accum);
    fprintf(fp,"%le", accum,notacion[1]);
    fprintf(fp2,"%le", accum,notacion[2]);
    fprintf(fp3,"%le", accum,notacion[3]);
    fprintf(fp4,"%le", 0, accum,notacion[0]);
    fprintf(fp5,"%le", accum,notacion2[1]);
    fprintf(fp6,"%le", accum,notacion2[2]);
    fprintf(fp7,"%le", accum,notacion2[3]);
    fprintf(fp8,"%le", accum,temperaturafinal);
    num_adc = 0;
    for(int i = 0; i<4;i++){
        if(notacion[i]<50){
            huecoV++;
        } else {
            huecoV=0;
        }
        if(notacion2[i]<50){
            huecoI++;
        } else {
            huecoI=0;
        }
    }
}
//Cerramos ficheros
fclose(fp);
fclose(fp2);
fclose(fp3);
fclose(fp4);
fclose(fp5);
fclose(fp6);
fclose(fp7);
fclose(fp8);
rename("voltaje1temporal.txt","VL1.txt");
rename("voltaje2temporal.txt","VL2.txt");
rename("voltaje3temporal.txt","VL3.txt");
rename("voltajentemporal.txt","VLN.txt");
rename("intensidad1temporal.txt","IL1.txt");
rename("intensidad2temporal.txt","IL2.txt");
rename("temperaturatemporal.txt","temperatura.txt");
rename("intensidad3temporal.txt","IL3.txt");//En cuanto se renombra este fichero, el
script lectura.c se ejecuta
}
return (0) ;
}

```

2.3 Lectura.c

```

//Ruben Garcia Segovia 15/05/2018
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <time.h>
#include <sys/stat.h>
#include <unistd.h>

//Funcion que hace la lectura del voltaje
void LecturaVoltaje(int linea, float *outVrms, float *outFrecuencia, float *outpico, int
*detector_V, char* nombrelinea){
    char caracteres[40];
    char *caracteres2;
    char *search = "    ";
    float i = 0;
    double media = 0;
    double analizador=0;
    double analizador_0=0;
    float pico = 0;
    float rms=0;
    int estado = 0;
    int estado2=1;
    int cuenta = 0;
    float rms2=0;
    float Vmin= 0;
    int estado3 = 0;
    int estado4=1;
    int estadof = 0;
    int cuenta2 = 0;
    float frecuencia = 0;
    int festado = 0;
    float ftiempo = 0;
    float ftiempo2 = 0;
    float ftiempo3 = 0;
    float fcuenta = 0;
    float tiempo_actual;
    float tiempo_recogida;
    int detector = 0;
    float detector_inicio;
    float detector_final;
    float detector_tiempo;
    float detector_valor;
    float tiempo_corte;
    FILE *V;

    switch(linea) { //Seleccionamos que linea vamos a procesar
    case 1 :
        V = fopen("./Voltaje/VL1.txt","r");
        break;

    case 2 :
        V = fopen("./Voltaje/VL2.txt","r");
        break;

    default :
        V = fopen("./Voltaje/VL3.txt","r");
    }
    printf("\nAnalizando la traza...\n\n");
    fgets(caracteres,40,V); //Cogemos la primera linea y sus 40 caracteres (que se ignoran)
    while(feof(V) == 0) { //Mientras que el archivo no este leido completamente...
        i++;
        fgets(caracteres,40,V); //Estraemos la linea correspondiente
        caracteres2 = strtok(caracteres, search);
        caracteres2 = strtok(NULL, search); //Hacemos una separacion entre el tiempo y el
        voltaje
        analizador = atof(caracteres2); //Analizador contiene el valor en V.
        // ***** VRMS **** //
        rms += (analizador)*(analizador);
        cuenta++;
        // ***** VRMS **** //
    }
}

```

```

// **** Frecuencia **** //
if(atof(caracteres) > 1 && pico>200) { //atof(caracteres)' es el tiempo
    if (analizador > (pico - 10) && festado == 0) {
        ftiempo = atof(caracteres);
        festado = 1;
    }
    if (analizador > (pico - 10) && festado == 2){
        ftiempo2 = atof(caracteres);
        frecuencia += 1/(ftiempo2-ftiempo);
        fcuenta++;
        festado = 3;
    }
    if (analizador < (pico/2) && festado == 1){
        festado = 2;
    }
    if (analizador < (pico/2) && festado == 3){
        festado = 0;
    }
}
// **** Frecuencia **** //
// **** Vmax **** //
if (((analizador) > pico) && analizador_0 > (analizador - 5) && analizador_0 != analizador){
    pico = analizador;
}
if (analizador < Vmin) {
    Vmin = analizador;
}
// **** Vmax **** //
// **** //
analizador_0 = analizador;
}
rms = sqrt(rms/cuenta); //Calculo final de Vrms
frecuencia = frecuencia / fcuenta; //Calculo final de la frecuencia
fclose(V);

*outVrms = rms; //Sacamos el valor
*outFrecuencia = 0.001 + frecuencia; //A prueba de errores. La frecuencia no puede ser 0.
*outpico = pico; //Sacamos el valor.

}
//Funcion que hace la lectura de la intensidad. Es el mismo procesado que el voltaje, pero en la linea de intensidad.
void LecturaIntensidad(int linea, float *outIrms, float *outFrecuencia, float *outpico,int *detector_I, char* nombrelinea){
    char caracteres[40];
    char *caracteres2;
    char *search = "    ";
    double media = 0;
    double analizador=0;
    double analizador_0=0;
    double analizador_0_0=0;
    double pico = 0;
    float rms=0;
    int estado = 0;
    int estado2=1;
    int cuenta = 0;
    float rms2=0;
    float cuentapico=0;
    int estado3 = 0;
    int estado4=1;
    int cuenta2 = 0;
    int frecuencia_estado= 0;
    float frecuencia = 0;
    int festado = 0;
    float Imin = 0;
    float ftiempo = 0;
    float ftiempo2 = 0;
    float ftiempo3 = 0;
    float fcuenta = 0;
    float detector = 0;
}

```

```

float detector_inicio;
float detector_final;
float detector_tiempo;
float detector_valor;
FILE *I;

switch(linea) {
case 1 :
    I = fopen("./Intensidad/IL1.txt","r");
    break;

case 2 :
    I = fopen("./Intensidad/IL2.txt","r");
    break;

default :
    I = fopen("./Intensidad/IL3.txt","r");
}
printf("\nAnalizando la traza...\n\n");
fgets(caracteres,40,I);
while(feof(I) == 0) {
    fgets(caracteres,40,I);
    caracteres2 = strtok(caracteres, search);
    caracteres2 = strtok(NULL, search);
    analizador = atof(caracteres2);
// ***** IRMS *****
    rms += (analizador)*(analizador);
    cuenta++;
// ***** Frecuencia *****
// ***** Imax *****
    if(atof(caracteres) > 1 && pico > 200) {
        if (analizador > (pico - 10) && festado == 0) {
            ftiempo = atof(caracteres);
            festado = 1;
        }
        if (analizador > (pico - 10) && festado == 2){
            ftiempo2 = atof(caracteres);
            frecuencia += 1/(ftiempo2-ftiempo);
            fcuenta++;
            festado = 3;
        }
        if (analizador < (pico/2) && festado == 1){
            festado = 2;
        }
        if (analizador < (pico/2) && festado == 3){
            festado = 0;
        }
    }
// ***** Frecuencia *****
// ***** Imax *****
    if (((analizador) > pico) && analizador_0 > (analizador - 5) && analizador_0 != analizador){
        pico = analizador;
    }
    if (analizador < Imin) {
        Imin = analizador;
    }
// ***** Imax *****
//*****
    analizador_0 = analizador;
}
rms = sqrt(rms/cuenta);
frecuencia = frecuencia/fcuenta;
fclose(I);
*outIrms = rms;
*outFrecuencia = 0.001 + frecuencia;
*outpico = pico;
}

//Funcion que calcula las potencias de las fases dadas
void Potencias(int linea,float *activa,float *reactiva,float *aparente,float *factordepotencia, float Irms, float Vrms){

```

```

FILE *V;
FILE *I;
char caracteresV[40];
char caracteresI[40];
char *caracteresV2;
char *caracteresI2;
char *search = "    ";
float potenciaactiva=0;
float potenciareactiva=0;
float potenciaaparente=0;
float cosphi = 0;
float cuenta=0;
float Voltaje, Intensidad;
switch(linea) { //Dependiendo de la linea que escojamos...
case 1 :
    V = fopen("./Voltaje/VL1.txt","r");
    I = fopen("./Intensidad/IL1.txt","r");
    break;

case 2 :
    V = fopen("./Voltaje/VL2.txt","r");
    I = fopen("./Intensidad/IL2.txt","r");
    break;

default :
    V = fopen("./Voltaje/VL3.txt","r");
    I = fopen("./Intensidad/IL3.txt","r");
}
printf("\nAnalizando la traza...\n\n");
fgets(caracteresV,40,V); //Siempre se descartan la primera linea
fgets(caracteresI,40,I);
while(feof(V) == 0 && feof(I) == 0) { //Mientras que el archivo no se termine...
    cuenta++;
    fgets(caracteresV,40,V);
    fgets(caracteresI,40,I);
    caracteresV2 = strtok(caracteresV, search);
    caracteresV2 = strtok(NULL, search);
    caracteresI2 = strtok(caracteresI, search);
    caracteresI2 = strtok(NULL, search);
    Voltaje = atof(caracteresV2); //Valor numerico del voltaje
    Intensidad = atof(caracteresI2); //Valor numerico de la intensidad
    potenciaactiva += Voltaje*Intensidad;
}
potenciaactiva = potenciaactiva / cuenta; //La potencia activa es la multiplicacion de
valores instantaneos entre numero de valores cogidos
potenciaaparente = Irms*Vrms; //La potencia aparente es la multiplicacion de sus valores
medios
potenciareactiva = potenciaaparente*potenciaaparente - potenciaactiva*potenciaactiva;
//La reactiva es aparente menos activa debido a el triangulo dado
cosphi = potenciaactiva/potenciaaparente; //El coseno es la potencia activa entre la
aparente.
*reactiva = sqrt(potenciareactiva); //Sacamos valores por la funcion
*activa = potenciaactiva;
*aparente = potenciaaparente;
*factordepotencia = cosphi;
}
//Funcion que calcula el desfase entre dos fases dadas
float desfaseV(int linea1, int linea2, float frecuencia, float picol,float pico2){
    FILE *V1;
    FILE *V2;
    char caracteresV1[40];
    char caracteresV2[40];
    char *caracteresV12;
    char *caracteresV22;
    char *search = "    ";
    float Voltaje1, Voltaje2,Voltaje1_0=-1,Voltaje2_0=-1;
    float ftiempol,ftiempo2;
    int fase1 = 1;
    float pivot=0;
    float resultado;
    switch(linea1) { //Elegimos lineas

```

```

case 1 :
    V1 = fopen("./Voltaje/VL1.txt","r");
    break;

case 2 :
    V1 = fopen("./Voltaje/VL2.txt","r");
    break;

default :
    V1 = fopen("./Voltaje/VL3.txt","r");
}
switch(linea2) { //Elegimos lineas
case 1 :
    V2 = fopen("./Voltaje/VL1.txt","r");
    break;

case 2 :
    V2 = fopen("./Voltaje/VL2.txt","r");
    break;

default :
    V2 = fopen("./Voltaje/VL3.txt","r");
}
printf("\nAnalizando la traza...\n\n");
fgets(caracteresV1,40,V1); //Ignoramos la primera linea
fgets(caracteresV2,40,V2);
while(feof(V1) == 0 && feof(V2) == 0) { //Mientras el archivo no se termine...
    fgets(caracteresV1,40,V1);
    fgets(caracteresV2,40,V2);
    caracteresV12 = strtok(caracteresV1, search);
    caracteresV12 = strtok(NULL, search);
    caracteresV22 = strtok(caracteresV2, search);
    caracteresV22 = strtok(NULL, search);
    Voltaje1 = atof(caracteresV12); //Voltaje uno
    Voltaje2 = atof(caracteresV22); //Voltaje dos
    if(Voltaje1_0 >= (pico1/2) && Voltaje1 < (pico1/2) ){
        ftiempol = atof(caracteresV1);
        fase1 = 0;
    }
    //Basicamente sacamos el tiempo entre que una fase cruza el valor pico/2 y lo hace
    //la otra.
    if(Voltaje2_0>= (pico2/2) && Voltaje2 < (pico2/2) && fase1 == 0){
        ftiempo2 = atof(caracteresV2);
        fase1 = 3;
    }

    if(fase1==3){
        ftiempol = ftiempol - ftiempo2;
        pivote = 1/frecuencia;
        resultado = 360 * ftiempol;
        resultado = resultado / pivote;
        if(resultado<0) resultado = 360 + resultado; //No sacamos valores negativos
        return resultado; //Devolvemos el resultado final
    }
    Voltaje1_0 = Voltaje1;
    Voltaje2_0 = Voltaje2;
}
}

//Funcion que guarda el dato procesado para su representacion grafica
void archivo(int anyo, int mes, int dia, int minuto, float dato, char* nombre){
    char string[40];
    FILE *TXT;
    sprintf(string,40,"../basededatos/%d/%d/%d/%s.txt",anyo,mes,dia,nombre);
    TXT=fopen(string,"a");
    fprintf(TXT,"%d %f\n",minuto, dato);
    fclose(TXT);
}

//Funcion que guarda todos los datos RAW del minuto
void guardararchivo(int anyo, int mes, int dia, int minuto, char* nombre,char* tipo){
    char string[40], stringsavedata[70];
    mkdir("../basededatos/datos",0777);
}

```

```

snprintf(string,40,"../basededatos/datos/%d",anyo);
mkdir(string,0777);
snprintf(string,40,"../basededatos/datos/%d/%d",anyo,mes);
mkdir(string,0777);
snprintf(string,40,"../basededatos/datos/%d/%d/%d",anyo,mes,dia);
mkdir(string,0777);
snprintf(string,40,"../basededatos/datos/%d/%d/%d/%s",anyo,mes,dia,nombre);
mkdir(string,0777);
snprintf(stringsavedata,70,"mv -f ./%s/%s.txt
../basededatos/datos/%d/%d/%d/%s/%d.txt",tipo,nombre,anyo,mes,dia,nombre,minuto);
printf("%s",stringsavedata);
system(stringsavedata);
}

int main()
{
    FILE *TXT;
    float rms, frecuencia, pico=0,
    pico2=0,L1potenciactiva,L1potenciareactiva,L1potenciaparente,L1cosphi,L1Vrms,L1Irms,L1L2de
    sfase;
    char string[40];
    int minutodeldia, dia, mes,anyo,detectorV=0, detectorI = 0;
    while(1) { //Se ejecuta de forma infinita hasta que lo paremos manualmente
        if (fopen("../monitorizacion/IL3.txt","r") != NULL) { //El proceso empieza cuando el
        script 'spi' termina su parte
            //Primero calculamos la fecha en la que estamos
            time_t t = time(NULL);
            struct tm tm = *localtime(&t);
            dia = tm.tm_mday;
            mes = tm.tm_mon + 1;
            anyo = tm.tm_year + 1900;
            minutodeldia = tm.tm_min + (tm.tm_hour*60);
            snprintf(string,30,"../basededatos/%d",anyo);
            mkdir(string,0777);
            snprintf(string,30,"../basededatos/%d/%d",anyo,mes);
            mkdir(string,0777);
            snprintf(string,30,"../basededatos/%d/%d/%d",anyo,mes,dia);
            mkdir(string,0777);
            //Creamos los directorios donde vamos a guardar la informacion
            //Movemos los archivos RAW a la carpeta correspondiente para empezar a tratarlos.
            system("mv -f ../monitorizacion/IL3.txt ./Intensidad/IL3.txt");
            system("mv -f ../monitorizacion/IL2.txt ./Intensidad/IL2.txt");
            system("mv -f ../monitorizacion/IL1.txt ./Intensidad/IL1.txt");

            system("mv -f ../monitorizacion/VL3.txt ./Voltaje/VL3.txt");
            system("mv -f ../monitorizacion/VL2.txt ./Voltaje/VL2.txt");
            system("mv -f ../monitorizacion/VL1.txt ./Voltaje/VL1.txt");
            system("mv -f ../monitorizacion/VLN.txt ./Voltaje/VLN.txt");

            //Se trata el voltaje
            LecturaVoltaje(1,&L1Vrms,&frecuencia,&pico,&detectorV, "VL1");

            //Y se guardan los archivos de Vrms, frecuencia y su valor pico.
            archivo(anyo,mes,dia,minutodeldia,L1Vrms,"VRMS1");
            archivo(anyo,mes,dia,minutodeldia,frecuencia,"FrecV1");
            archivo(anyo,mes,dia,minutodeldia,pico,"PicoV1");

            //Lo mismo para la intensidad
            LecturaIntensidad(1,&L1Irms,&frecuencia,&pico2,&detectorI, "IL1");

            archivo(anyo,mes,dia,minutodeldia,L1Irms,"IRMS1");
            archivo(anyo,mes,dia,minutodeldia,frecuencia,"FrecIL1");
            archivo(anyo,mes,dia,minutodeldia,pico2,"PicoIL1");

            //Se calculan las potencias

            Potencias(1,&L1potenciactiva,&L1potenciareactiva,&L1potenciaparente,&L1cosphi,L1Ir
            ms,L1Vrms);

            //Se guardan el triangulo de potencias y el factor de potencia.
            archivo(anyo,mes,dia,minutodeldia,L1potenciactiva,"PotActL1");
}

```

```
archivo(ancho,mes,dia,minutodeldia,L1potenciareactiva,"PotReactL1");
archivo(ancho,mes,dia,minutodeldia,L1potenciaparente,"PotaparL1");
archivo(ancho,mes,dia,minutodeldia,L1cosphi,"FdpL1");

//Por ultimo, guardamos todos los datos RAW para su analisis mediante software
especifico.
guardararchivo(ancho,mes,dia,minutodeldia,"VL1","Voltaje");
guardararchivo(ancho,mes,dia,minutodeldia,"VL2","Voltaje");
guardararchivo(ancho,mes,dia,minutodeldia,"VL3","Voltaje");
guardararchivo(ancho,mes,dia,minutodeldia,"VLN","Voltaje");

guardararchivo(ancho,mes,dia,minutodeldia,"IL1","Intensidad");
guardararchivo(ancho,mes,dia,minutodeldia,"IL2","Intensidad");
guardararchivo(ancho,mes,dia,minutodeldia,"IL3","Intensidad");

} else {
    sleep(1); //Dormimos 1seg para no dejar un bucle infinito.
}
}

return 0;
}
```

2.4 ftp.c

```

//Ruben Garcia Segovia - 15/05/2018
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

FILE *actuador;

int main (void) {
    char ftp_past; //Variable para saber si se ha cambiado el estado del ftp
    while(1){
        char ftp_actuador[10]; //Almacenamos el documento de ftp.
        actuador=fopen("/home/pi/Desktop/monitorizacion/ftp.txt","r");
        fgets(ftp_actuador, 10, actuador);
        fclose(actuador);
        if(ftp_actuador[0]=='1'){ //Si es 1 esta activado
            if(ftp_past=='0'){ //Si ha cambiado escribimos, sino, no.
                FILE *log;
                time_t t = time(NULL);
                struct tm tm = *localtime(&t);
                log=fopen("../basededatos/log.txt","a"); //Escribimos en el log
                fprintf(log,"%n%d-%d-%d %d:%d:%d - Encendido del FTP",tm.tm_year + 1900,
                tm.tm_mon + 1, tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
                fclose(log);
                system("sudo /etc/init.d/proftpd start"); //Inicia el servidor FTP
            }
        } else {
            if(ftp_past=='1') //Si ha cambiado de estado escribimos, sino, no.
            {
                FILE *log;
                time_t t = time(NULL);
                struct tm tm = *localtime(&t);
                log=fopen("../basededatos/log.txt","a"); //Escribimos en el log
                fprintf(log,"%n%d-%d-%d %d:%d:%d - Apagado del FTP",tm.tm_year + 1900,
                tm.tm_mon + 1, tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
                fclose(log);
                system("sudo /etc/init.d/proftpd stop"); //Para el servidor FTP
            }
        }
        sleep(1); //Dormimos un segundo, para no crear un bucle while(1) infinito
        ftp_past = ftp_actuador[0];
    }
}

```

2.5 memoria.c

```
//Ruben Garcia Segovia - 15/05/2018.
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

FILE *memoria;

int main (void) {
    while(1){
        char memo_disponible[2];
        system("du -bsh /home/pi/Desktop/basededatos/datos > memoria.txt"); //Comprobamos el
        espacio ocupado de los datos
        memoria=fopen("./memoria.txt","r");
        fgets(memo_disponible, 2, memoria);
        if(memo_disponible[0]=="6" && memo_disponible[1]==",") { //Si supera los 6 GB...
            system("rm -r /home/pi/Desktop/basededatos/datos"); //Borramos todo
        }
        fclose(memoria);
        sleep(300); //Dormimos 5 minutos
    }
}
```

2.6 index.php

```

<?php
$pass = $_GET['pass']; //Recogemos variables por la URL
$grafica_mostrar = $_GET['type'];
?>

<DIV align="center">
<?php //Centramos todo a la derecha
//Si no nos hemos logueado aún, nos pide la contraseña
if(!($_POST['contra']=="rubenelportero" || isset($_POST['grafica'])) ||
isset($_POST['actuador']) || isset($_POST['ftp']) || $pass == "KHJH3EWNuxFIxy6ifbgF")){
?>
<form method="POST">
Contraseña <input type="password" name="contra"> <br>
<input type="submit" name="submit" value="Submit">
</form>
<?php
}
//Una vez que introducimos la contraseña correcta, cargamos toda la pagina web
if($_POST['contra']=="rubenelportero" || isset($_POST['grafica'])) ||
isset($_POST['actuador']) || $_POST['ftp'] || $pass == "KHJH3EWNuxFIxy6ifbgF"){
//Si hemos recibido la contraseña por url, la aceptamos y cargamos la grafica que toque
if($pass == "KHJH3EWNuxFIxy6ifbgF"){
    $_POST['grafica'] = $grafica_mostrar;
}
//Recarga de la pagina web cada 20 segundos con la contraseña y la grafica correspondiente.
header("refresh:20;url=http://192.168.0.197/index.php?pass=KHJH3EWNuxFIxy6ifbgF&type=".$_POST[
'grafica']."'");
?>
<form action="/index.php" method="POST">
<select name="grafica">
<option value="0" <?php if($_POST['grafica']==0):?> selected="selected" <?php endif?>
>Potencias</option>
<option value="1" <?php if($_POST['grafica']==1):?> selected="selected" <?php
endif?>>Voltaje e Intensidad rms</option>
<option value="3" <?php if($_POST['grafica']==3):?> selected="selected" <?php
endif?>>Frecuencias</option>
<option value="4" <?php if($_POST['grafica']==4):?> selected="selected" <?php
endif?>>Factor de potencia</option>
</select>
<input type="submit">
</form>

<?php
//Las graficas se crean en otro archivo dentro del servidor
switch($_POST['grafica']){
    case 0:
        echo "<img src='./grafica.php?type0=4'>";
        break;
    case 1:
        echo "<img src='./grafica.php?type0=0'>";
        echo "<br>";
        break;
    case 3:
        echo "<img src='./grafica.php?type0=1'>";
        break;
    case 4:
        echo "<img src='./grafica.php?type0=3'>";
        break;
}
?>

<div id="log" style="border : solid 2px #000000;
background : #ffffff;
color : #000000;
width : 500px;
height: 200px;
overflow : auto; ">

<?php

```

```

//Escribimos el log en la caja con scroll correspondiente
$myfile = fopen("/home/pi/Desktop/basededatos/log.txt", "r");
while(!feof($myfile))
{
    echo fgets($myfile);
    echo "<br>";
}
fclose($myfile);
//Leemos el estado del actuador y del ftp
$file = fopen("/home/pi/Desktop/monitorizacion/actuador.txt", "r");
$actu_txt = fgets($file);
fclose($file);
$file = fopen("/home/pi/Desktop/monitorizacion/ftp.txt", "r");
$ftp_txt = fgets($file);
fclose($file);
if ($actu_txt=="1"){
    $actu_txt = "Activado";
} else {
    $actu_txt = "Desactivado";
}
if ($ftp_txt=="1"){
    $ftp_txt = "Activado";
} else {
    $ftp_txt = "Desactivado";
}
?>

</div>

<?php
//Si pulsamos en el boton del actuador o del ftp, realizamos la accion correspondiente
//editar el .txt)
$file2 = fopen("/home/pi/Desktop/monitorizacion/actuador.txt", "r+");
if($_POST['actuador']=="Activar"){
    $actu_txt = "Activado";
    fwrite($file2,"1");
} else if($_POST['actuador']=="Desactivar"){
    $actu_txt = "Desactivado";
    fwrite($file2,"0");
}
fclose($file2);

$file3 = fopen("/home/pi/Desktop/monitorizacion/ftp.txt", "r+");
if($_POST['ftp']=="Activar"){
    $ftp_txt = "Activado";
    fwrite($file3,"1");
} else if($_POST['ftp']=="Desactivar"){
    $ftp_txt = "Desactivado";
    fwrite($file3,"0");
}
fclose($file3);

//Sacamos el espacio libre en el disco
$espaciolibre = disk_free_space("/");
$espaciotal = disk_total_space("/");
//Y lo pasamos a porcentaje
$espacioporcentaje = ($espaciolibre/$espaciotal)*100;
echo "Espacio disponible para almacenamiento: ";
//Y se muestra sin decimales
echo intval($espacioporcentaje);
echo "%";

// Creamos un formulario para activar o desactivar el actuador y el FTP
?>
<form method="POST">
<br>Estado del actuador: <?php echo $actu_txt?> <br>
<input type="hidden" name="actuador" value="holo!">
<input type="radio" name="actuador" value="Activar">Activar

```

```
<input type="radio" name="actuador" value="Desactivar">Desactivar<br>
<input type="submit" name="submit" value="Enviar">
</form>

<form method="POST">
Estado del ftp: <?php echo $ftp_txt?> <br>
<input type="hidden" name="ftp" value="hola!">
<input type="radio" name="ftp" value="Activar">Activar
<input type="radio" name="ftp" value="Desactivar">Desactivar<br>
<input type="submit" name="submit" value="Enviar">
</form>
<?php echo date(DATE_RFC2822);
//Representamos la hora
?
?>
</DIV>
```

2.7 Grafica.php

```

<?php
//Ejemplo de uso en la libreria Pchart Modificado por
//Ruben Garcia Segovia
/* Set the default timezone */
date_default_timezone_set('Etc/GMT');

/* pChart library inclusions */
include("./library/pchart2/class/pData.class.php");
include("./library/pchart2/class/pDraw.class.php");
include("./library/pchart2/class/pImage.class.php");

/* Create and populate the pData object */
$MyData = new pData();
$BaseTs = mktime(0,0,0,4,18,2018);
$fecha = getdate();

$tipo = $_GET['type0']; //Cogemos la grafica a representar por la url

//Con esta matriz representamos las distintas graficas dependiendo del valor obtenido por url.
$type[0][0] = "VRMS1";
$type[0][1] = "IRMS1";
$type[0][2] = "Valores rms";
$type[1][0] = "FrecV1";
$type[1][1] = "FrecI1";
$type[1][2] = "Frecuencias de onda";
$type[2][0] = "PicoV1";
$type[2][1] = "PicoI1";
$type[2][2] = "Valores maximos";
$type[3][0] = "FdpL1";
$type[3][1] = "";
$type[3][2] = "Factor de potencia L1";
$type[4][0] = "PotActL1";
$type[4][1] = "PotReactL1";
$type[4][2] = "Potencias L1";

//Leemos el archivo con los datos a representar
$myfile =
=fopen("/home/pi/Desktop/basededatos/".$fecha[year]."/".$fecha[mon]."/".$fecha[mday]."/".$type
[$tipo][0].".txt", "r");
$datos = array();
$tiempos = array();
$i = 0;

while(!feof($myfile)) //Hasta llegar al final del archivo
{
    $leer = fgets($myfile);
    list($tiempo, $dato) = explode(" ",$leer); //Separamos la linea por el espacio
    $datos[$i] = $dato; //Obtenemos el dato
    $tiempos[$i] = $tiempo; //Obtenemos el tiempo
    $i = $i + 1;
}

fclose($myfile);

if($tipo != 3) { //Depende de la representacion a hacer, tenemos que recoger mas de una
grafica
$b = 0;
$myfile2 =
=fopen("/home/pi/Desktop/basededatos/".$fecha[year]."/".$fecha[mon]."/".$fecha[mday]."/".$type[
$tipo][1].".txt", "r");
$datos2 = array();
while(!feof($myfile2))
{
    $leer = fgets($myfile2);
    list($tiempo, $dato) = explode(" ",$leer);
    $datos2[$b] = $dato;
    $b = $b + 1;
}

```

```

fclose ($myfile2);
}

if($tipo == 4) { //Depende de la representacion a hacer, tenemos que recoger mas de una
grafica
$c = 0;
$myfile3 =
fopen("/home/pi/Desktop/basededatos/".$fecha[year]."/".$fecha[mon]."/".$fecha[mday]."/PotaparL
1.txt", "r");
$datos3 = array();
while(!feof($myfile3))
{
    $leer = fgets($myfile3);
    list($tiempo, $dato) = explode(" ",$leer);
    $datos3[$c] = $dato;
    $c = $c + 1;
}

fclose ($myfile3);
}

$i = $i - 1;
$z = 10;
$y=0;
if($i < 30){ //Dependiendo del numero de muestras, representamos mas o menos valores de
tiempo
    $f=1; //La representacion en la misma, pero con menos muestras se ve el tiempo en mas
detalle
} else if ($i < 120){
    $f = 4;
} else if($i < 240){
    $f = 9;
} else {
    $f = 59;
}
for($j=0; $j<$i; $j++)
{
    $MyData->addPoints($datos[$j], ".$type[$tipo][0]."); //Introducimos un valor a la grafica
    if($tipo!=3){
        $MyData->addPoints($datos2[$j], ".$type[$tipo][1]."); //Introducimos un valor a la
grafica
    }
    if($tipo==4){
        $MyData->addPoints($datos3[$j], "PotaparL1"); //Introducimos un valor a la grafica
    }
    if($z>$f){
        $MyData->addPoints($BaseTs + $tiempos[$j]*60,"Tiempo"); //Introducimos un valor al
tiempo
        $y = $j;
        $z=1;
    } else {
        $z = $z + 1;
        $MyData->addPoints($BaseTs + $tiempos[$y]*60,"Tiempo"); //Introducimos un valor al
tiempo
    }
}

//Ejes de la representacion
$MyData->setAxisName(0,"Y");
$MyData->setAxisDisplay(0,AXIS_FORMAT_METRIC); //Formato metrico de representacion
$MyData->setAbscissa("Tiempo");
$MyData->setXAxisDisplay(AXIS_FORMAT_TIME,"H:i"); //El tiempo en HH:MM

/* Create the pChart object */
$myPicture = new pImage(900,430,$MyData); //La resolucion de la imagen final

/* Draw a background */
$Settings = array("R"=>90, "G"=>90, "B"=>90, "Dash"=>1, "DashR"=>120, "DashG"=>120,

```

```

"DashB"=>120);
$myPicture->drawFilledRectangle(0,0,900,430,$Settings);

/* Overlay with a gradient */
$Settings = array("StartR"=>200, "StartG"=>200, "StartB"=>200, "EndR"=>50, "EndG"=>50,
"EndB"=>50, "Alpha"=>50);
$myPicture->drawGradientArea(0,0,900,530,DIRECTION_VERTICAL,$Settings);
$myPicture->drawGradientArea(0,0,900,530,DIRECTION_HORIZONTAL,$Settings);

/* Add a border to the picture */
$myPicture->drawRectangle(0,0,899,429,array("R"=>0,"G"=>0,"B"=>0));

/* Write the chart title */
$myPicture->setFontProperties(array("FontName"=>"./library/pchart2/fonts/Forgotte.ttf","FontSize"=>11));
$myPicture->drawText(150,35,"".$type[$tipo][2].",",array("FontSize"=>20,"Align"=>TEXT_ALIGN_BOTTOMMIDDLE));

/* Set the default font */
$myPicture->setFontProperties(array("FontName"=>"./library/pchart2/fonts/pf_arma_five.ttf","FontSize"=>6));

/* Define the chart area */
$myPicture->setGraphArea(60,40,880,400);

/* Draw the scale */
setScaleSettings =
array("XMargin"=>10,"YMargin"=>10,"Floating"=>TRUE,"GridR"=>200,"GridG"=>200,"GridB"=>200,"RemoveSkippedAxis"=>TRUE,"DrawSubTicks"=>FALSE,"Mode"=>SCALE_MODE_ADDALL_START0,"LabelingMethod"=>LABELING_DIFFERENT);

$myPicture->drawScale($scaleSettings);

/* Draw the line chart */
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));
$myPicture->drawLineChart();

/* Write the chart legend */
$myPicture->drawLegend(580,20,array("Style"=>LEGEND_NOBORDER,"Mode"=>LEGEND_HORIZONTAL));

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.drawSplineChart.network.png");

?>

```

2.8 Grafica_GUI.php

```

<?php
//Ejemplo de uso en la libreria Pchart Modificado por
//Ruben Garcia Segovia
//Version GUI (resolucion distinta)
/* Set the default timezone */
date_default_timezone_set('Etc/GMT');

/* pChart library inclusions */
include("./library/pchart2/class/pData.class.php");
include("./library/pchart2/class/pDraw.class.php");
include("./library/pchart2/class/pImage.class.php");

/* Create and populate the pData object */
$MyData = new pData();
$BaseTs = mktime(0,0,0,4,18,2018);
$fecha = getdate();

$tipo = $_GET['type0']; //Cogemos la grafica a representar por la url

//Con esta matriz representamos las distintas graficas dependiendo del valor obtenido por url.
$type[0][0] = "VRMS1";
$type[0][1] = "IRMS1";
$type[0][2] = "Valores rms";
$type[1][0] = "FrecV1";
$type[1][1] = "FrecI1";
$type[1][2] = "Frecuencias de onda";
$type[2][0] = "PicoV1";
$type[2][1] = "PicoI1";
$type[2][2] = "Valores maximos";
$type[3][0] = "FdpL1";
$type[3][1] = "";
$type[3][2] = "Factor de potencia L1";
$type[4][0] = "PotActL1";
$type[4][1] = "PotReactL1";
$type[4][2] = "Potencias L1";

//Leemos el archivo con los datos a representar
$myfile =
=fopen("/home/pi/Desktop/basededatos/".$fecha[year]."/".$fecha[mon]."/".$fecha[mday]."/".$type[$tipo][0].".txt", "r");
$datos = array();
$tiempos = array();
$si = 0;

while(!feof($myfile)) //Hasta llegar al final del archivo
{
    $leer = fgets($myfile);
    list($tiempo, $dato) = explode(" ",$leer); //Separamos la linea por el espacio
    $datos[$si] = $dato; //Obtenemos el dato
    $tiempos[$si] = $tiempo; //Obtenemos el tiempo
    $si = $si + 1;
}

fclose($myfile);

if($tipo != 3) { //Depende de la representacion a hacer, tenemos que recoger mas de una
grafica
$b = 0;
$myfile2 =
=fopen("/home/pi/Desktop/basededatos/".$fecha[year]."/".$fecha[mon]."/".$fecha[mday]."/".$type[$tipo][1].".txt", "r");
$datos2 = array();
while(!feof($myfile2))
{
    $leer = fgets($myfile2);
    list($tiempo, $dato) = explode(" ",$leer);
    $datos2[$b] = $dato;
    $b = $b + 1;
}

```

```

fclose($myfile2);
}

if($tipo == 4) { //Depende de la representacion a hacer, tenemos que recoger mas de una
grafica
$c = 0;
$myfile3 =
fopen("/home/pi/Desktop/basededatos/".$fecha[year]."/".$fecha[mon]."/".$fecha[mday]."/PotaparL
1.txt", "r");
$datos3 = array();
while(!feof($myfile3))
{
    $leer = fgets($myfile3);
    list($tiempo, $dato) = explode(" ",$leer);
    $datos3[$c] = $dato;
    $c = $c + 1;
}

fclose($myfile3);
}

$tiempo = $tiempo - 1;
$z = 10;
$y=0;
if($tiempo < 30){ //Dependiendo del numero de muestras, representamos mas o menos valores de tiempo
    $f=1; //La representacion en la misma, pero con menos muestras se ve el tiempo en mas
    detalle
} else if ($tiempo < 120){
    $f = 4;
} else if($tiempo < 240){
    $f = 9;
} else {
    $f = 59;
}
for($j=0; $j<$tiempo; $j++)
{
    $MyData->addPoints($datos[$j],".$type[$tipo][0]."); //Introducimos un valor a la grafica
    if($tipo!=3){
        $MyData->addPoints($datos2[$j],".$type[$tipo][1]."); //Introducimos un valor a la
        grafica
    }
    if($tipo==4){
        $MyData->addPoints($datos3[$j],"PotaparL1"); //Introducimos un valor a la grafica
    }
    if($z>$f){
        $MyData->addPoints($BaseTs + $tiempos[$j]*60,"Tiempo"); //Introducimos un valor al
        tiempo
        $y = $j;
        $z=1;
    } else {
        $z = $z + 1;
        $MyData->addPoints($BaseTs + $tiempos[$y]*60,"Tiempo"); //Introducimos un valor al
        tiempo
    }
}

$MyData->setAxisName(0,"Y");
$MyData->setAxisDisplay(0,AXIS_FORMAT_METRIC); //Formato metrico de representacion
$MyData->setAbscissa("Tiempo");
$MyData->setXAxisDisplay(AXIS_FORMAT_TIME,"H:i"); //El tiempo en HH:MM

/* Create the pChart object */
$myPicture = new pImage(750,390,$MyData); //La resolucion de la imagen optima para la GUI

/* Draw a background */
$Settings = array("R"=>90, "G"=>90, "B"=>90, "Dash"=>1, "DashR"=>120, "DashG"=>120,
"DashB"=>120);

```

```

$myPicture->drawFilledRectangle(0,0,750,390,$Settings);

/* Overlay with a gradient */
$Settings = array("StartR"=>200, "StartG"=>200, "StartB"=>200, "EndR"=>50, "EndG"=>50,
"EndB"=>50, "Alpha"=>50);
$myPicture->drawGradientArea(0,0,750,390,DIRECTION_VERTICAL,$Settings);
$myPicture->drawGradientArea(0,0,750,390,DIRECTION_HORIZONTAL,$Settings);

/* Add a border to the picture */
$myPicture->drawRectangle(0,0,749,389,array("R"=>0,"G"=>0,"B"=>0));

/* Write the chart title */
$myPicture->setFontProperties(array("FontName"=>"./library/pchart2/fonts/Forgotte.ttf","FontSize"=>11));
$myPicture->drawText(150,35,"".$type[$tipo][2].",",array("FontSize"=>20,"Align"=>TEXT_ALIGN_BOTTOMMIDDLE));

/* Set the default font */
$myPicture->setFontProperties(array("FontName"=>"./library/pchart2/fonts/pf_arma_five.ttf","FontSize"=>6));

/* Define the chart area */
$myPicture->setGraphArea(30,40,740,375);

/* Draw the scale */
$scaleSettings =
array("XMargin"=>10,"YMargin"=>10,"Floating"=>TRUE,"GridR"=>200,"GridG"=>200,"GridB"=>200,"RemoveSkippedAxis"=>TRUE,"DrawSubTicks"=>FALSE,"Mode"=>SCALE_MODE_ADDALL_START0,"LabelingMethod"=>LABELING_DIFFERENT);

$myPicture->drawScale($scaleSettings);

/* Draw the line chart */
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));
$myPicture->drawLineChart();

/* Write the chart legend */
$myPicture->drawLegend(580,20,array("Style"=>LEGEND_NOBORDER,"Mode"=>LEGEND_HORIZONTAL));

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.drawSplineChart.network.png");

?>

```

2.9 móvil.php

```
<?php
$pass = $_GET['pass'];
$dato = $_GET['type']; //Recogemos las variables por URL
$variable = $_GET['trigger'];
if($pass == "KHJH3EWNuxFIxy6ifbgF"){ //Comprobamos la contraseña
    switch ($variable){ //Seleccionamos entre los tres modos: Actuador, FTP o Estado de
        alimentacion
        case 0: //0 o 1 se escribe en el .txt correspondiente, un 2 devuelve el estado del archivo
            if($dato < 2) {
                $file = fopen("/home/pi/Desktop/monitorizacion/actuador.txt", "r+");
                fwrite($file,$dato);
                fclose($file);
            } else {
                $file = fopen("/home/pi/Desktop/monitorizacion/actuador.txt", "r+");
                echo fgets($file);
                fclose($file);
            }
            break;
        case 1:
            if($dato < 2) {
                $file = fopen("/home/pi/Desktop/monitorizacion/ftp.txt", "r+");
                fwrite($file,$dato);
                fclose($file);
            } else {
                $file = fopen("/home/pi/Desktop/monitorizacion/ftp.txt", "r+");
                echo fgets($file);
                fclose($file);
            }
            break;
        case 2:
            $file = fopen("/home/pi/Desktop/monitorizacion/estado.txt", "r+");
            echo fgets($file);
            fclose($file);
            break;
    }
} else {
    echo 3; //El 3 representa una contraseña mal introducida
}
?>
```

2.10 gui.py

```

# Ruben Garcia Segovia - 15/05/2018
import Tkinter as tk # for Python3 use import tkinter as tk
from PIL import Image
from urllib import urlopen
from StringIO import StringIO
import time

def tiempoactual(): #Imprimimos la hora en la GUI
    reloj.configure(text = time.strftime("%Y-%m-%d %H:%M:%S"))
    reloj.after(1000,tiempoactual) #Y lo hacemos cada segundo

def actualizar_imagenes(): #Actualizamos todas las imagenes
    global boton_pulsado #Boton que indica que grafica estamos representando
    conseguir_imagenes()
    conseguir_imagenes2()
    conseguir_imagenes3()
    conseguir_imagenes4()
    if(boton_pulsado == 1):
        toggle_text4()
    if(boton_pulsado == 2):
        toggle_text5()
    if(boton_pulsado == 3):
        toggle_text6()
    if(boton_pulsado == 4):
        toggle_text7()
    root.after(62000,actualizar_imagenes) #Y lo hacemos cada 62 seg

def conseguir_imagenes(): #Grafica de potencia
    URL = 'http://192.168.0.197/grafica_GUI.php?type0=4' #Nos conectamos al servidor
    data = urlopen(URL).read() #Leemos la respuesta del servidor
    file_pot = StringIO(data) #Guardamos como archivo de entrada la respuesta
    img = Image.open(file_pot) #Lo abrimos como imagen
    img.save('./potencias.png') #Guardamos la imagen
    file_pot = None #vaciamos RAM.
    data = None

def conseguir_imagenes2():
    URL2 = 'http://192.168.0.197/grafica_GUI.php?type0=1'
    data2 = urlopen(URL2).read()
    file_frec = StringIO(data2)
    img2 = Image.open(file_frec)
    img2.save('./frecuencias.png')
    file_frec = None
    data2 = None

def conseguir_imagenes3():
    URL3 = 'http://192.168.0.197/grafica_GUI.php?type0=3'
    data3 = urlopen(URL3).read()
    file_fdp = StringIO(data3)
    img3 = Image.open(file_fdp)
    img3.save('./fdp.png')
    file_fdp = None
    data3 = None

def conseguir_imagenes4():
    URL4 = 'http://192.168.0.197/grafica_GUI.php?type0=0'
    data4 = urlopen(URL4).read()
    file_medias = StringIO(data4)
    img4 = Image.open(file_medias)
    img4.save('./medias.png')
    file_medias = None
    data4 = None

def actuador(funcion): #Cambiamos el estado del actuador
    f = open ('../monitorizacion/actuador.txt','w')
    f.write(funcion)
    f.close()

```

```

def actuador_lec(): #Leemos el estado del actuador
    f = open ('../monitorizacion/actuador.txt','r')
    return f.read()
    f.close()

def actuador_boton(): #Comprueba el estado del actuador a cada segundo
    if actuador_lec() == "1":
        button.configure(text= "Apagar actuador")
    else:
        button.configure(text= "Encender actuador")
    button.after(1000,actuador_boton)

def ftp(funcion): #Cambiamos el estado del ftp
    f = open ('../monitorizacion/ftp.txt','w')
    f.write(funcion)
    f.close()

def ftp_boton(): #Comprueba el estado del FTP a cada segundo
    if ftp_lec() == "1":
        button2.configure(text= "Apagar FTP")
    else:
        button2.configure(text= "Encender FTP")
    button2.after(1000,ftp_boton)

def ftp_lec(): #Leemos el estado del FTP
    f = open ('../monitorizacion/ftp.txt','r')
    return f.read()
    f.close()

def log_lec(): #Cargamos en la caja de texto el log entero.
    editArea.delete("1.0",tk.END) #Borramos el texto de la caja de texto
    f = open('../basededatos/log.txt', 'r') #leemos el archivo .txt
    editArea.insert(tk.INSERT,f.read()) #Insertamos el texto (Incluidos saltos de linea)
    editArea.see(tk.END) #Hacemos auto-scroll
    f.close() #Cerramos el archivo.
    root.after(1000,log_lec) #Ejecutamos esto cada segundo.

def toggle_text1(): #Detecta la pulsacion del boton_actuador
    if button["text"] == "Encender actuador":
        actuador("1")
        button["text"] = "Apagar actuador"
    else:
        actuador("0")
        button["text"] = "Encender actuador"

def toggle_text2(): #Detecta la pulsacion del boton_FTP
    if button2["text"] == "Encender FTP":
        ftp("1")
        button2["text"] = "Apagar FTP"
    else:
        ftp("0")
        button2["text"] = "Encender FTP"

def toggle_text3(): #Detecta la pulsacion del boton_LOG
    global boton_pulsado
    label.config(image=' ')
    log_lec()
    boton_pulsado = 0

def toggle_text4(): #Detecta la pulsacion del boton_potencias
    global boton_pulsado
    image.configure(file=".//potencias.png")
    label.config(image=image)
    boton_pulsado = 1

def toggle_text5(): #Detecta la pulsacion del boton_Vmedias
    global boton_pulsado
    image.configure(file=".//medias.png")

```

```

label.config(image=image)
boton_pulsado = 2

def toggle_text6(): #Detecta la pulsacion del boton_frecuencias
    global boton_pulsado
    image.configure(file="./frecuencias.png")
    label.config(image=image)
    boton_pulsado = 3

def toggle_text7(): #Detecta la pulsacion del boton_FDP
    global boton_pulsado
    image.configure(file="./fdp.png")
    label.config(image=image)
    boton_pulsado = 4

root = tk.Tk() #Iniciamos el entorno
root.geometry('800x480') #Las mismas dimensiones que la pantalla tactil
root.title("Dispositivo de monitorizacion de red electrica IIOT") #Titulo de la ventana

reloj = tk.Label(root, text="Cargando...") #Label de la hora
reloj.place(x=330,y=18) #Con sus coordenadas

#Todos los Widget.place hacen alusion a las cordenadas XY absolutas de la ventana 800x480

button = tk.Button( text="Encender actuador", width=30, command=toggle_text1) #Creamos el boton del actuador
button.place(x=30, y = 10)

button2 = tk.Button( text="Apagar FTP", width=30, command=toggle_text2) #Creamos el boton del FTP
button2.place(x= 500, y=10)

actuador_boton() #Comprobamos el estado del actuador
ftp_boton() #Comprobamos el estado del ftp

button3 = tk.Button( text="LOG", width=10, command=toggle_text3) #Creamos el boton del LOG
button3.place(x= 30, y=50)

button4 = tk.Button( text="Potencias", width=10, command=toggle_text4) #Creamos el boton de potencias
button4.place(x= 170, y=50)

button5 = tk.Button( text="Vrms/Irms", width=10, command=toggle_text5) #Creamos el boton Vrms/Irms
button5.place(x= 310, y=50)

button6 = tk.Button( text="Frecuencias", width=10, command=toggle_text6) #Creamos el boton frecuencias
button6.place(x= 460, y=50)

button7 = tk.Button( text="FDP", width=10, command=toggle_text7) #Creamos el boton FTP
button7.place(x= 610, y=50)

#Creacion de la caja de texto para el log
frame1 = tk.Frame(root,width=400, height=480,bg = '#ffffff',borderwidth=1, relief="sunken")
#Frame donde se aloja el texto
scrollbar = tk.Scrollbar(frame1) #Creamos el scrollbar
#El texto ira sobre el frame:
editArea = tk.Text(frame1, width=100, height=10,
wrap="word",yscrollcommand=scrollbar.set,borderwidth=0, highlightthickness=0)
scrollbar.config(command=editArea.yview) #Scrollbar solo en vertical
scrollbar.pack(side="right", fill="y")
editArea.pack(side="left", fill="both", expand=True)
frame1.place(x=30,y=180)

image = tk.PhotoImage(file="./potencias.png") #Cargamos el widget de imagen
label = tk.Label(image=image)
label.pack()
label.place(x=30,y=80)

```

```
toggle_text4() #Autoactivamos el primer boton  
tiempoactual() #Encendemos el reloj  
actualizar_imagenes() #Actualizamos las imagenes  
  
root.mainloop()
```

2.11 ActivityMain.java

```

package com.makingbytes.tfg_rubengarciasegovia;
//Ruben Garcia Segovia
//Programacion de la aplicacion Android 15/05/2018

import android.app.AlertDialog;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.Handler;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.TextView;
import android.support.design.widget.NavigationView;
import android.support.v4.view.GravityCompat;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;
import android.widget.Button;
import android.view.KeyEvent;
import android.app.AlertDialog;
import android.content.DialogInterface;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URI;
import java.net.URL;
import java.net.URLConnection;
import java.util.Timer;
import java.util.TimerTask;

public class MainActivity extends AppCompatActivity //Clase principal de la aplicacion
    implements NavigationView.OnNavigationItemSelectedListener {
    //Variables globales
    public static String direccion = "Hola"; //Direccion introducida por el susuario
    public String contra = ""; //Contraseña introducida por el usuario
    private ImageView imgImagen;
    public boolean actu = false; //El estado del actuador
    public boolean ftp = false; //El estado del FTP
    public boolean introducido = false; //Variable que comprueba si el user ha introducido
    datos o no
    public boolean corte = false; //Estado de la red
    public int seccion = 1; //Seccion elegida (potencia, control, fdp, Voltaje medio...)

    @Override
    protected void onCreate(Bundle savedInstanceState) { //Al abrir la aplicacion
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
            this, drawer, toolbar, R.string.navigation_drawer_open,
            R.string.navigation_drawer_close);
        drawer.addDrawerListener(toggle);
        toggle.syncState();
        imgImagen = (ImageView) findViewById(R.id.imagen);
        imgImagen.setVisibility(View.GONE); //Ocultamos la grafica
        ocultarcontrol(); // Ocultamos los botones del apartado de control
    }
}

```

```

//Al ocultar los botones del apartado de control y la grafica, solo queda visible
//el Login donde se pide el servidor y la contraseña
final Button button = findViewById(R.id.button_options);

button.setOnClickListener(new View.OnClickListener() { //Si el boton de conectar se
pulsa...
    public void onClick(View v) {
        ocultar(); //Ocultamos los botones de conexion
        EditText servidor = (EditText) findViewById(R.id.editText3);
        EditText contra_introducido = (EditText) findViewById(R.id.editText2);
        CargaImagenes nuevaTarea = new CargaImagenes();
        direccion = servidor.getText().toString(); //Cogemos el servidor
        contra = contra_introducido.getText().toString(); //Cogemos la contraseña
        nuevaTarea.execute(direccion + "/grafica.php?type=4"); //Seleccionamos la
        primera grafica
        imgImagen.setRotation(90); //Rotamos el widget 90 grados
        imgImagen.setLayoutParams().height += 850; //Le hacemos zoom en vertical y
        horizontal
        imgImagen.setLayoutParams().width += 2000;
        imgImagen.setVisibility(View.VISIBLE); //Hacemos la imagen visible
        introducido = true; //Se confirma que se ha introducido los datos de conexion
        actualizar_control(); //Se ejecutan las funciones que regulan la
        auto-actualizacion
        actualizar_fotos(); //del apartado de control y las graficas representadas
    }
});

final Button button_actuador = findViewById(R.id.button_actuador);

button_actuador.setOnClickListener(new View.OnClickListener() { //Si pulsamos el
boton del actuador
    public void onClick(View v) {
        if(actu == false){ //Si esta desactivado se activa el actuador
            conectar_url(direccion + "/movil.php?pass="+contra+"&type=1&trigger=0");
            button_actuador.setText("Actuador Activado");
            actu = true;
        }else{ //Si esta activado se desactiva el actuador
            conectar_url(direccion + "/movil.php?pass="+contra+"&type=0&trigger=0");
            button_actuador.setText("Actuador Desactivado");
            actu = false;
        }
    }
});
final Button button_ftp = findViewById(R.id.button_ftp);

button_ftp.setOnClickListener(new View.OnClickListener() { //Si pulsamos el boton de
ftp
    public void onClick(View v) {
        if(ftp == false){ //Activamos o desactivamos el ftp en la raspberry pi
            conectar_url(direccion + "/movil.php?pass="+contra+"&type=1&trigger=1");
            button_ftp.setText("FTP Activado");
            ftp = true;
        }else {
            conectar_url(direccion + "/movil.php?pass="+contra+"&type=0&trigger=1");
            button_ftp.setText("FTP Desactivado");
            ftp = false;
        }
    }
});
NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);
navigationView.setNavigationItemSelectedListener(this);

}

private void actualizar_control(){
    final Handler handler = new Handler();
    Timer timer = new Timer();
    TimerTask task = new TimerTask() {
        @Override
        public void run() {
            handler.post(new Runnable() {

```

```

        public void run() {
            actualizar_estado(); //Es ejecutado cada 5000 seg.
        }
    });
}
timer.schedule(task, 1,5000); //cada 5000 mseg se ejecuta 'public void run'
}
private void actualizar_fotos(){
    final Handler handler = new Handler();
    Timer timer = new Timer();
    TimerTask task = new TimerTask() {
        @Override
        public void run() { //Se ejecuta cada 60 segundos
            handler.post(new Runnable() {
                public void run() {
                    switch(seccion){ //Dependiendo de la sección en la que estemos, se
                        actualiza
                        case 1: //una foto u otra.
                            seleccionarfoto(direccion + "/grafica.php?type0=4");
                            break;
                        case 2:
                            seleccionarfoto(direccion + "/grafica.php?type0=0");
                            break;
                        case 3:
                            seleccionarfoto(direccion + "/grafica.php?type0=1");
                            break;
                        case 4:
                            seleccionarfoto(direccion + "/grafica.php?type0=3");
                            break;
                        case 5:
                            break;
                    }
                }
            });
        }
    };
    timer.schedule(task, 1,60000);
}
private void ocultarcontrol() {
    TextView estado1 = (TextView) findViewById(R.id.control_estado_actual);
    TextView estado2 = (TextView) findViewById(R.id.Control_estado);
    Button botonactuador = (Button) findViewById(R.id.button_actuador);
    Button botonftp = (Button) findViewById(R.id.button_ftp);
    estado1.setVisibility(View.GONE);
    estado2.setVisibility(View.GONE); //Ocultamos los widgets del apartado de control
    botonftp.setVisibility(View.GONE);
    botonactuador.setVisibility(View.GONE);
}

@Override
public void onBackPressed() {
    DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
    if (drawer.isDrawerOpen(GravityCompat.START)) {
        drawer.closeDrawer(GravityCompat.START);
    } else {
        super.onBackPressed();
    }
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
}

@Override

```

```

public boolean onKeyDown(int keyCode, KeyEvent event) {
    if (keyCode == KeyEvent.KEYCODE_BACK && event.getRepeatCount() == 0) {
        // Con esto detectamos el botón de atrás
        AlertDialog.Builder alert = new AlertDialog.Builder(this);
        alert.setTitle("Salir");
        alert.setMessage("¿Quieres salir?"); //Preguntamos si realmente se quiere salir
        alert.setNegativeButton("No", null);
        alert.setPositiveButton("Si", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialogo1, int id) {
                pirarse(); //y nos vamos
            }
        });
        alert.show();
        return true;
    }
    return super.onKeyDown(keyCode, event);
}

private void pirarse() {
    Toast.makeText(getApplicationContext(), "...Cerrando aplicación...", Toast.LENGTH_SHORT).show();
    this.finish(); //Mostramos un mensaje y nos vamos de la app
}

private void ocultar() { //Funcion que oculta los widget del log in
    final Button button = findViewById(R.id.button_options);
    TextView textouno = (TextView) findViewById(R.id.pass_info);
    TextView textodos = (TextView) findViewById(R.id.textView2);
    EditText cajauno = (EditText) findViewById(R.id.editText3);
    EditText cajados = (EditText) findViewById(R.id.editText2);
    button.setVisibility(View.GONE);
    textouno.setVisibility(View.GONE);
    textodos.setVisibility(View.GONE);
    cajauno.setVisibility(View.GONE);
    cajados.setVisibility(View.GONE);
}

private void seleccionarfoto(String direccion) { //Funcion que ejecuta el thread de
carga de imagen
    CargaImagenes nuevaTarea = new CargaImagenes(); //Se hace en segundo plano para no
bloquear la app
    nuevaTarea.execute(direccion);
    imgImagen.setVisibility(View.VISIBLE);
}

private void conectar_url(String direccion) { //funcion que ejecuta el thread de carga a
la url
    cargar_url nuevaTarea = new cargar_url(); //Se hace en segundo plano para no
bloquear la app
    nuevaTarea.execute(direccion);
    try {
        Thread.sleep(100);
    } catch (InterruptedException ex) {
    }
}

private void actualizar_estado(){ //Actualizamos los estados de los 3 objetos que
tenemos en el apartado de control
    lectura_estado nuevaTarea = new lectura_estado();
    nuevaTarea.execute(direccion + "/movil.php?pass="+contra+"&type=2&trigger=2");
    lectura_boton nuevaTarea2 = new lectura_boton();
    nuevaTarea2.execute(direccion + "/movil.php?pass="+contra+"&type=2&trigger=0");
    lectura_ftp nuevaTarea3 = new lectura_ftp();
    nuevaTarea3.execute(direccion + "/movil.php?pass="+contra+"&type=2&trigger=1");
}

@SuppressWarnings("StatementWithEmptyBody")
@Override
public boolean onNavigationItemSelected(MenuItem item) {
    // Handle navigation view item clicks here.
    int id = item.getItemId();
}

```

```

//Dependiendo de la seccion que pulsemos en el menu:
if (id == R.id.potencias && introducido == true) {
    ocultarcontrol();
    seleccionarfoto(direccion + "/grafica.php?type0=4");
    seccion= 1;
} else if (id == R.id.medias && introducido == true) {
    ocultarcontrol();
    seleccionarfoto(direccion + "/grafica.php?type0=0");
    seccion= 2;
} else if (id == R.id.frecuencias && introducido == true) {
    ocultarcontrol();
    seleccionarfoto(direccion + "/grafica.php?type0=1");
    seccion= 3;
} else if (id == R.id.fdp && introducido == true) {
    ocultarcontrol();
    seleccionarfoto(direccion + "/grafica.php?type0=3");
    seccion= 4;
} else if (id == R.id.control && introducido == true) { //Si es el apartado de control...
    seccion= 5;
    imgImagen.setVisibility(View.GONE); //Ocultamos las graficas
    TextView estado1 = (TextView) findViewById(R.id.control_estado_actual);
    TextView estado2 = (TextView) findViewById(R.id.Control_estado);
    Button botonactuador = (Button) findViewById(R.id.button_actuador);
    Button botonftp = (Button) findViewById(R.id.button_ftp);
    estado1.setVisibility(View.VISIBLE);
    estado2.setVisibility(View.VISIBLE); //Mostramos los widget
    botonftp.setVisibility(View.VISIBLE);
    botonactuador.setVisibility(View.VISIBLE);
    actualizar_estado(); //Actualizamos valores
}

DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
drawer.closeDrawer(GravityCompat.START);
return true;
}

/****************Lectura imagenes *****/
private class CargaImagenes extends AsyncTask<String, Void, Bitmap> {

    ProgressDialog pDialog;

    @Override
    protected void onPreExecute() { //Se ejecuta antes del thread.
        // TODO Auto-generated method stub
        super.onPreExecute();

        pDialog = new ProgressDialog(MainActivity.this);
        pDialog.setMessage("Cargando Imagen"); //Ventana de espera
        pDialog.setCancelable(true);
        pDialog.setProgressStyle(ProgressDialog.STYLE_SPINNER);
        pDialog.show();
    }

    @Override
    protected Bitmap doInBackground(String... params) { //Se ejecuta EN el thread
        // TODO Auto-generated method stub
        Log.i("doInBackground", "Entra en doInBackground");
        String url = params[0];
        Bitmap imagen = descargarImagen(url); //ejecutamos la funcion que recupera la imagen del server
        return imagen;
    }

    @Override
    protected void onPostExecute(Bitmap result) { //Despues de la ejecucion del thread
        // TODO Auto-generated method stub
        super.onPostExecute(result);
        imgImagen.setImageBitmap(result); //Cargamos la foto en el widget correspondiente
        pDialog.dismiss(); //Cerramos la ventana de cargando
    }
}

```

```

}

private Bitmap descargarImagen(String imageHttpAddress) { //Funcion que recoge la
    imagen
    URL imageUrl = null;
    Bitmap imagen = null;
    try { //Usamos un try por que no sabemos que si el servidor estara funcionando o
        imageUrl = new URL(imageHttpAddress); //Cogemos la url
        HttpURLConnection conn = (HttpURLConnection) imageUrl.openConnection();
        //Creamos la conexion
        conn.connect(); //Nos conectamos
        imagen = BitmapFactory.decodeStream(conn.getInputStream()); //Decodificamos
        la imagen del inputStream
    } catch (IOException ex) {
        ex.printStackTrace();
    }

    return imagen; //Devolvemos la imagen.
}

}

*****      pasar datos por URL *****/
private class cargar_url extends AsyncTask<String, Void, String> {

    ProgressDialog pDialog;

    @Override
    protected void onPreExecute() {
        // TODO Auto-generated method stub
        super.onPreExecute();

        pDialog = new ProgressDialog(MainActivity.this);
        pDialog.setMessage("Cargando..."); //Cuadro de carga
        pDialog.setCancelable(true);
        pDialog.setProgressStyle(ProgressDialog.STYLE_SPINNER);
        pDialog.show();
    }

    @Override
    protected String doInBackground(String... params) {
        // TODO Auto-generated method stub
        Log.i("doInBackground", "Entra en doInBackground");
        String url = params[0];
        String resultado = conectar(url); //Se ejecuta la funcion de conectar
        return resultado;
    }

    @Override
    protected void onPostExecute(String result) {
        // TODO Auto-generated method stub
        super.onPostExecute(result); //Al terminar cerramos el cuadro de carga
        pDialog.dismiss(); //Aqui no hacemos nada con el string, puesto que
        //Este thread se usa para mandar datos al servidor, y no para
        recogerlos

    private String conectar(String imageHttpAddress) {
        URL direUrl = null;
        String resulta = null;
        try { //Nos conectamos de igual forma que con la imagen
            direUrl = new URL(imageHttpAddress);
            HttpURLConnection conn = (HttpURLConnection) direUrl.openConnection();
            conn.connect();
            InputStream stream = conn.getInputStream(); //Cogemos la respuesta
            BufferedReader reader = new BufferedReader(new InputStreamReader(stream)); //La
            metemos en el buffer
            StringBuffer buffer = new StringBuffer();
            String line = "";
            line = reader.readLine(); //Leemos la primera linea devuelta al buffer
            buffer.append(line);
        }
    }
}

```

```

        return buffer.toString(); //Decodificamos a string
    } catch (IOException ex) {
        ex.printStackTrace();
    }

    return resulta;
}

// Nota: las clases cargar_url, lectura_estado, lectura_boton y lectura_ftp son iguales,
// pero cambian
// sus acciones en el momento de finalizar el thread.

/***************** Lectura del estado *****/
private class lectura_estado extends AsyncTask<String, Void, String> {

    @Override
    protected void onPreExecute() {
        // TODO Auto-generated method stub
        super.onPreExecute();
    }

    @Override
    protected String doInBackground(String... params) {
        // TODO Auto-generated method stub
        Log.i("doInBackground", "Entra en doInBackground");
        String url = params[0];
        String resultado = conectar(url);
        return resultado;
    }

    @Override
    protected void onPostExecute(String result) {
        // TODO Auto-generated method stub
        super.onPostExecute(result);
        TextView estado = (TextView) findViewById(R.id.control_estado_actual);
        if (result.equals("1")){ //Si se ha detectado el corte, se ejecuta la accion
        correspondiente
            estado.setText("Corte");
            if(corte==false) {
                AlertDialog.Builder alert = new AlertDialog.Builder(MainActivity.this);
                alert.setTitle("Estado");
                alert.setMessage("Corte en la alimentación");
                alert.setNegativeButton("Ok", null);
                alert.show();
                corte = true;
            }
        } else {
            corte = false;
            estado.setText("Estable");
        }
    }
}

private String conectar(String imageHttpAddress) {
    URL direUrl = null;
    String resulta = null;
    try {
        direUrl = new URL(imageHttpAddress);
        HttpURLConnection conn = (HttpURLConnection) direUrl.openConnection();
        conn.connect();
        InputStream stream = conn.getInputStream();
        BufferedReader reader = new BufferedReader(new InputStreamReader(stream));
        StringBuffer buffer = new StringBuffer();
        String line = "";
        line = reader.readLine();
        buffer.append(line);
        return buffer.toString();
    } catch (IOException ex) {

```

```

        ex.printStackTrace();
    }

    return resulta;
}

} ****Lectura del actuador ****
private class lectura_boton extends AsyncTask<String, Void, String> {

    @Override
    protected void onPreExecute() {
        // TODO Auto-generated method stub
        super.onPreExecute();
    }

    @Override
    protected String doInBackground(String... params) {
        // TODO Auto-generated method stub
        Log.i("doInBackground", "Entra en doInBackground");
        String url = params[0];
        String resultado = conectar(url);
        return resultado;
    }

    @Override
    protected void onPostExecute(String result) {
        // TODO Auto-generated method stub
        super.onPostExecute(result);
        final Button button_actuador = findViewById(R.id.button_actuador);
        if (result.equals("1")){
            actu = true;
            button_actuador.setText("Actuador Activado");
        } else {
            actu = false;
            button_actuador.setText("Actuador Desactivado");
        }
    }
}

private String conectar(String imageHttpAddress) {
    URL direUrl = null;
    String resulta = null;
    try {
        direUrl = new URL(imageHttpAddress);
        HttpURLConnection conn = (HttpURLConnection) direUrl.openConnection();
        conn.connect();
        InputStream stream = conn.getInputStream();
        BufferedReader reader = new BufferedReader(new InputStreamReader(stream));
        StringBuffer buffer = new StringBuffer();
        String line = "";
        line = reader.readLine();
        buffer.append(line);
        return buffer.toString();
    } catch (IOException ex) {
        ex.printStackTrace();
    }
    return resulta;
}

} ****Lectura del ftp ****
private class lectura_ftp extends AsyncTask<String, Void, String> {

    @Override
    protected void onPreExecute() {
        // TODO Auto-generated method stub
        super.onPreExecute();
    }
}

```

```
}

@Override
protected String doInBackground(String... params) {
    // TODO Auto-generated method stub
    Log.i("doInBackground", "Entra en doInBackground");
    String url = params[0];
    String resultado = conectar(url);
    return resultado;
}

@Override
protected void onPostExecute(String result) {
    // TODO Auto-generated method stub
    super.onPostExecute(result);
    final Button button_actuador = findViewById(R.id.button_ftp);
    if (result.equals("1")){
        ftp = true;
        button_actuador.setText("FTP Activado");
    } else {
        ftp = false;
        button_actuador.setText("FTP Desactivado");
    }
}

private String conectar(String imageHttpAddress) {
    URL direUrl = null;
    String resulta = null;
    try {
        direUrl = new URL(imageHttpAddress);
        HttpURLConnection conn = (HttpURLConnection) direUrl.openConnection();
        conn.connect();
        InputStream stream = conn.getInputStream();
        BufferedReader reader = new BufferedReader(new InputStreamReader(stream));
        StringBuffer buffer = new StringBuffer();
        String line = "";
        line = reader.readLine();
        buffer.append(line);
        return buffer.toString();
    } catch (IOException ex) {
        ex.printStackTrace();
    }
    return resulta;
}
}
```

2.12 Content_main.xml

```

<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context=".MainActivity"
    tools:showIn="@layout/app_bar_main">
    <!-- Widget de imagen para las diferentes graficas-->
    <ImageView
        android:id="@+id/imagen"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <!-- Widget de texto para el login-->
    <TextView
        android:id="@+id/pass_info"
        android:layout_width="wrap_content"
        android:layout_height="18dp"
        android:text="Contraseña:"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintHorizontal_bias="0.501"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.174" />
    <!-- Widget caja de texto para el login, de aqui se lee la contraseña-->
    <!-- Tambien posee la contraseña por defecto-->
    <EditText
        android:id="@+id/editText2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="textPassword"
        android:text="KHJH3EWNxuFIxy6ifbgF"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.513"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/pass_info"
        app:layout_constraintVertical_bias="0.0" />
    <!-- Widget de texto para el login-->
    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Servidor"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/editText2"
        app:layout_constraintVertical_bias="0.0" />
    <!-- Widget caja de texto para la recogida de la url del servidor-->
    <!-- Por defecto es 192.168.0.197-->
    <EditText
        android:id="@+id/editText3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10"

```

```

    android:inputType="textPersonName"
    android:text="http://192.168.0.197"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.5"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView2"
    app:layout_constraintVertical_bias="0.0" />

<!-- Widget boton que confirma que los datos se han introducido-->
<Button
    android:id="@+id/button_options"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Conectar"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/editText3"
    app:layout_constraintVertical_bias="0.068" />
<!-- Widget de texto para la sección control-->
<TextView
    android:id="@+id/Control_estado"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Estado:"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.5"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.065" />
<!-- Widget de texto con el estado actual de la red-->
<TextView
    android:id="@+id/control_estado_actual"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Estable"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.501"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/Control_estado"
    app:layout_constraintVertical_bias="0.0" />

<!-- Widget boton del actuador-->
<Button
    android:id="@+id/button_actuador"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Actuador Activado"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/control_estado_actual"
    app:layout_constraintVertical_bias="0.27" />

<!-- Widget boton del ftp-->
<Button
    android:id="@+id/button_ftp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="FTP Activado"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/button_actuador"
    app:layout_constraintVertical_bias="0.17" />

</android.support.constraint.ConstraintLayout>

```


LICENCIA DE USO

Rubén García Segovia, autor de este trabajo fin de grado, pongo todo el documento, excepto anexos, bajo la licencia Creative Commons internacional 4.0 de reconocimiento y compartir igual.



Bajo esta licencia, usted está autorizado a leer, redistribuir, cambiar, adaptar y hacer uso comercial de esa obra siempre y cuando se cumplan los siguientes términos y condiciones:

1. Dar autoría al autor original de la obra (Rubén García Segovia – <https://github.com/rubenelportero/>).
2. Poner su trabajo bajo esta misma licencia u otra similar.

Para leer la licencia completa u obtener más información, visitar el siguiente enlace o ver la licencia completa adjuntada en el CD (En inglés):

<https://creativecommons.org/licenses/by-sa/4.0/legalcode>

El anexo 1 queda exento de esta licencia debido a que se trata de datasheets y hojas de fabricante, material con licencias propietarias que dependen únicamente de los fabricantes.

El anexo 2, junto con todo el código de programación en los distintos lenguajes de programación usados y adjuntados en este trabajo están bajo la licencia GPLv3 de GNU, siendo código de software libre, permitiendo la modificación, redistribución, mejora e incluso uso comercial de todo el código incluido en el trabajo de fin de grado siempre y cuando se mantenga la licencia. Para más detalles, revisar la licencia completa en la siguiente url o ver la licencia completa adjuntada en el CD (En inglés):

<https://www.gnu.org/licenses/gpl-3.0.html>

Todo el proyecto está disponible en el repositorio GitHub correspondiente:

<https://github.com/rubenelportero/Design-of-a-electrical-networking-and-control-system>