Práctica 5:
Modulación de potencia de una lámpara incandecente usando Arduino y la Raspberry Pi

Fundamentos de Sistemas Embebidos

Autor: José Mauricio Matamoros de Maria y Campos

1. Objetivo

El alumno aprenderá a modular la potencia de una carga resistiva de alta potencia opto-acoplada a un microcontrolador por medio de un detector de cruce por cero y un TRIAC.

2. Introducción

La presente práctica resume los pasos a seguir para modular la cantidad de corriente que pasa por un foco incandescente con un microcontrolador. En particular, se interesa en el uso de un circuito de detección de cruce por cero para conmutar un triac acoplado a un Arduino UNO/Mega. Los datos registrados serán posteriormente enviados vía I^2C a una Raspberry Pi para controlar la intencidad del foco.

2.1. El puente rectificador

Un puente rectificador o rectificador de onda completa es un arreglo de 4 diodos que permiten el paso de la corriente en un sólo sentido, invirtiendo así la parte negativa de una señal de AC respecto a su voltaje de referencia. Los puentes rectificadores son un componente fundamental en los transformadores de corriente de AC a DC.

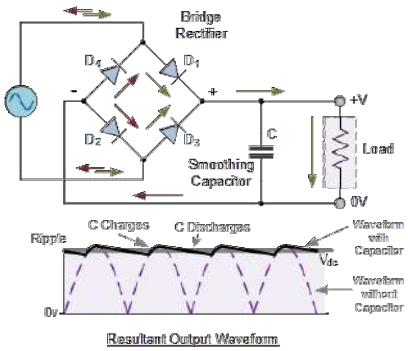


Figura 1: Rectificador de onda completa[1]

El circuito funciona de la siguiente manera. En la primera parte del ciclo, cuando el voltaje comienza a aumentar, la corriente fluye de la parte norte del puente (arriba) a través de D_1 hacia la carga, y luego de regreso por D_2 hacia el neutro (véase Figura 1). En esta primera etapa, D_3 y D_4 actuan como barreras evitando que la corriente fluya directamente de la fase al neutro (corto circuito). Tras pasar por la carga, el voltaje en en el ánodo de D_4 ha caído y es menor que en el cátodo, por lo que no habrá flujo de corriente en esta dirección, pero aún es positivo respecto al neutro ($V_N = 0$), por lo que la corriente tendrpa que pasar por D_1. De forma análoga, el voltaje en el cátodo de D_3 es menor que en el ánodo, por lo que tampoco habrá flujo en esta dirección.

En la segunda parte del ciclo, el voltaje de fase disminuye respecto al neutro, por lo que la corriente fluye de la parte sur del puente (abajo) a través de D_3 hacia la carga, y luego de regreso por D_4 hacia la fase (véase Figura 1). En esta segunda etapa, D_1 y D_2 actuan como barreras evitando que la corriente fluya directamente del neutro a la fase (corto circuito). Tras pasar por la carga, el voltaje en en el ánodo de D_2 ha caído y es menor que en el cátodo, por lo que no habrá flujo de corriente en esta dirección, pero aún es positivo respecto a la fase ($V_N = 0$), por lo que la corriente tendrpa que pasar por D_4. De forma análoga, el voltaje en el cátodo de D_1 es menor que en el ánodo, por lo que tampoco habrá flujo en esta dirección.

[1] Fuente de imagen: https://lasopaeden528.weebly.com/bridge-rectifier-calculator.html

2.2. Optoacopladores

Un optoacoplador o optoaislador es un circuito integrado que permite aislar mecánicamente dos circuitos, por lo que se usa comunmente para separar la lógica de control de los circuitos de potencia. El principio básico de un optoacoplador, como su nombre lo indica, es utilizar transductores ópticos para la transmisión de señales eléctricas. Así, en un lado del optoacoplador se tendrá siempre un diodo LED y en el otro extremo un fotoreceptor, que puede ser un foto SRC, un foto dárlington, un foto TRIAC o un fototransistor, siendo este último el más común.

En un optoacoplador, el diodo LED emite una cantidad de luz directamente proporcional a la corriente que circula por éste y, al incidir ésta en el fotoreceptor, el estímulo luminoso activa el paso de corriente a través de éste. Por ejemplo, en el caso de un fototransistor, al incidir la luz en la juntura de la base ésta se ioniza, generando un puente de iones que permite el flujo entre los extremos del transistor. Además, la mayoría de los optoacopladores tienen un pin conectado directamente a la base que sirve para ajustar la sensibilidad de la misma mediante la inyección de un voltaje pequeño.

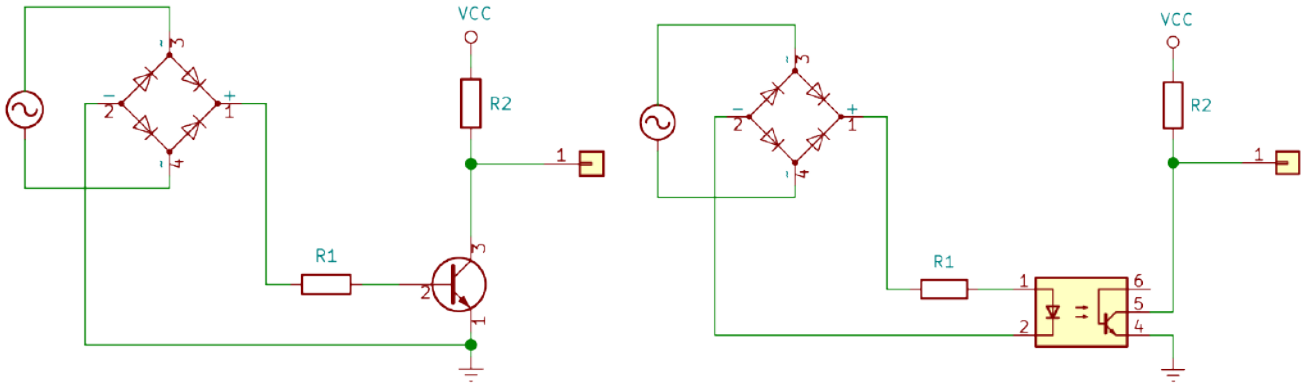
Los fototransistores foto-dárlingtons se utilizan principalmente en circuitos DC, mientras que los foto SCR y los foto TRIACs permiten controlar los circuitos de AC. Existen muchos otros tipos de combinaciones de fuente-sensor tales como LED-fotodiodo, LED-LÁSER, pares de lámpara-fotorresistencia, optoacopladores reflectantes y ranurados.

Por ejemplo, el integrado 4N25 puede usarse para monitorear el voltaje de la línea de tensión doméstica con un integrado. Según su hoja de especificaciones [1], la entrada del 4N25 acepta hasta 60mA, permite un flujo por el colector de hasta 50mA, y aisla hasta $5000V_{\text{RMS}}$. Si se toma como entrada un voltaje de línea rectificado de $127V_{\text{RMS}}$ y se limita la corriente a la corriente de prueba de 50mA indicada en la hoja de especificaciones [1], el 4N25 tendrá que acoplarse con una resistencia de al menos 3K9Ω aproximada mediante la fórmula:

$$R = \frac{V}{I} = \frac{127V_{\text{RMS}} \times \sqrt{2}}{0.050A} \approx \frac{179.61V}{0.050A} \approx 3692.1\Omega$$

2.3. Detector de cruce por cero

Un circuito detector de cruce por cero es un circuito electrónico diseñado para detectar cuando una señal senoidal pasa por cero. Estos circuitos se usan comunmente en electrónica de potencia tanto para detectar la frecuencia de la línea y hacer cálculo de fases. Además, al reducirse la diferencia de potencial a cero entre la fase y el neutro, la corriente instantánea también es cero, haciendo de éste el momento ideal para cortar la alimentación sin dañar las cargas.



(a) Detector de cruce por cero simple

(b) Detector de cruce por cero optoacoplado

Figura 2: Detectores de cruce por cero

La forma más sencilla de alambrar un circuito detector de cruce por cero es mediante un puente rectificador, dos resistencias y un transistor tipo NPN en modo interruptor (véase Figura 2a). El puente rectificador se encarga de invertir la parte negativa de la señal de AC, evitando así corrientes inversas que el transistor es incapaz de manejar. Cuando hay voltaje en la línea, éste habilita la base cerrando el circuito del transistor y conectando el pin de sensado a tierra (la resistencia de base evita el corto circuito y ajusta el umbral de sensibilidad). Tan pronto

como la diferencia de potencial entre la línea y el neutro cae a cero (o suficientemente bajo como la resistencia de base permita) el circuito se abre y en el pin de sensado se registra Vcc.

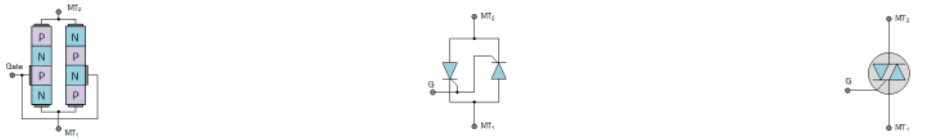
Para calcular R_1 es necesario tomar en cuenta las características eléctricas del transistor y los voltajes de pico de la línea. Se sabe que $V_{\text{pico}} = V_{\text{RMS}}\sqrt{2}$, por lo que usando la ley de ohm se tiene:

$$R_1 = \frac{V_{\text{RMS}}\sqrt{2}}{i_{\text{transistor}}} \approx \frac{1.4142 V_{\text{RMS}}}{i_{\text{transistor}}} \tag{1}$$

Sin embargo, el voltaje de la línea rara vez viene rectificado y un transitorio de corriente derivado de una descarga inductiva (arranque de refrigerador o microondas) puede incluso duplicar el voltaje de la línea, quemando no sólo el transistor sino el microprocesador. Es por esto que es muy aconsejable utilizar un optoacoplador en lugar de un simple transistor, tal como muestra la figura Figura 2b. Los principios de operación son los mismos.

2.4. TRIACs

Un TRIAC o triodo interruptor para corriente alterna (*Triode AC Switch*) es un integrado de estado sólido compuesto por dos tristores conectados en paralelo inverso (véase Figura 3b) que permite conmutar la corriente que pasa por un circuito de AC a alta frecuencia de manera similar a como operan los transistores bipolares y FETs en DC. Es decir, un TRIAC es un interruptor de estado sólido que puede operar a gran velocidad que, a diferencia de los relés, no existe la posibilidad de que un arco eléctrico funda los metales y el dispositivo se quede en encendido permanente, sino que al quemarse un TRIAC siempre abre el circuito. Por otro lado, basta una corriente muy pequeña entre el gate (G) y cualquiera de las terminales (MT_1 y MT_2) para encender al TRIAC, lo que lo convierte en el aliado ideal para controlar dispositivos de alta potencia con un microcontrolador.



(a) Construcción física (b) Equivalente en tristores (c) Símbolo

Figura 3: El Triac[2]

Como siempre, al utilizar un TRIAC es deseable aislar la parte de corriente directa del circuito de la parte de corriente alterna, es decir, el TRIAC deberá estar aislado pero acoplado al circuito DC. Esto normalmente se realiza mediante el uso de optoacopladores tipo MOC, tal como se ilustra en la Figura 4.

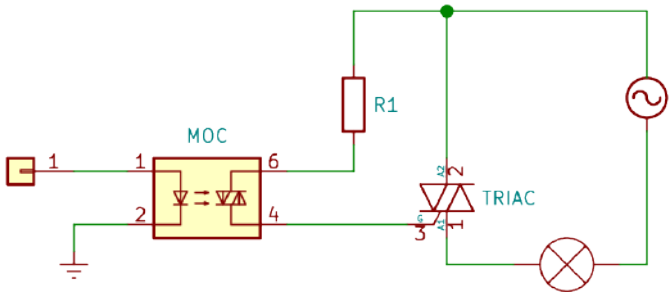


Figura 4: Triac optoacoplado

2.5. Modulación de potencia de carga resistiva en AC

A diferencia de un circuito de DC, la modulación de la potencia de una carga en un circuito de AC de una fase incolucra cuatro parámetros: i) el voltaje en la carga, ii) la corriente que circula por la carga, iii) la impedancia de la carga y iv) el ángulo de fase. O, matemáticamente hablando:

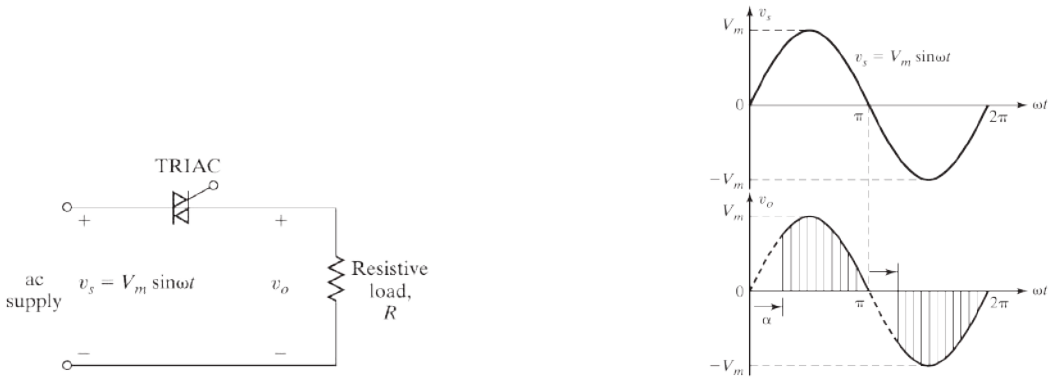
[2]Fuente de imagen: https://www.electronics-tutorials.ws/power/triac.html

$$p(t) = vi \tag{2}$$
$$= V_m \sin(\omega t + \theta_v)\, I_m \sin(\omega t + \theta_i) \tag{3}$$

donde:

- ω la velocidad angular tal que $\omega = 2\pi f$
- θ el ángulo de fase
- θ el ángulo de fase
- V_m el voltaje máximo
- I_m la corriente máxima

De estos cuatro parámetros, en un circuito puramente resistivo la impedancia R es constante, la corriente $i(\omega t + \theta_i)$ es proporcional al voltaje a la entrada de la carga y a la resistencia de la misma, y el voltaje $v(\omega t + \theta_v)$ oscila en el rango $[-V_m, V_m]$, con V_m el voltaje de pico. No obstante, se puede modificar voltaje promedio en la carga al variar el ángulo de fase y así controlar la potencia. Esta técnica es de hecho el principio fundamental de los circuitos convertidores de AC-AC a base de TRIAC como el que se muestra en la Figura 5.



(a) Circuito

(b) Señal voltaica

Figura 5: Convertidor AC-AC de una fase.[3]

Nótese que los ángulos de defasamiento de voltaje θ_v y corriente θ_i han desaparecido. Esto se debe a que en un circuito de AC de una sola fase $\theta_v = 0$ por ser la única fase y por ende la referencia. Además, en un circuito puramente resistivo las señales de voltaje y corriente están acopladas, por lo que $\theta_i = \theta_v$ tal como se muestra en la Figura 6.

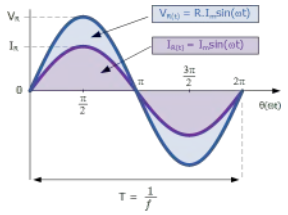


Figura 6: Voltaje y corriente en un circuito resistivo monofase.[4]

En la mayoría de los casos lo que interesa no es el cálculo de la potencia neta, sino controlar el factor de potencia, el decir, el porcentaje de la potencia máxima que la carga está entregando. Al estar sincronizadas la corriente y el voltaje por ser un circuito resistivo y no existir más que una fase, el cálculo de la potencia se simplifica y se convierte en el cociente del voltaje promedio aplicado a la carga respecto al voltaje RMS de la línea. Es decir

[3]Fuente de la imagen: Rashid [2, pp. 33].

$$P_f = \frac{V_o}{V_{RMS}} \tag{4}$$

y dado que V_{RMS} es fijo, lo que interesa calcular es el voltaje aplicado a la carga V_o que se calcula como [2, 579–583]:[5]

$$\begin{aligned} V_o^2 =& \frac{2}{2\pi}\int_{\alpha}^{\pi} 2V_{RMS}^2 \sin^2(\omega t)\, d(\omega t) && (5)\\ =& \frac{4V_{RMS}^2}{4\pi}\int_{\alpha}^{\pi} \left(1-\cos(2\omega t)\right) d(\omega t) && (6)\\ =& \frac{V_{RMS}^2}{\pi}\left(\pi - \alpha + \frac{\sin(2\alpha)}{2}\right) && (7) \end{aligned}$$

Por lo tanto

$$\begin{aligned} V_o =& \left[\frac{V_{RMS}^2}{\pi}\left(\pi - \alpha + \frac{\sin(2\alpha)}{2}\right)\right]^{\frac{1}{2}} && (8)\\ =& V_{RMS}\sqrt{\frac{1}{\pi}\left(\pi - \alpha + \frac{\sin(2\alpha)}{2}\right)} && (9) \end{aligned}$$

Ahora, supóngase que se desea modular la potencia de un foco incandescente de $60W$ conectado a una línea estándar de $V_{RMS} = 120V$, 60Hz. Como $P = \frac{V^2}{R}$ se puede calcular tanto la resistencia del foco como la corriente a fin de elegir el TRIAC adecuado:

$$\begin{aligned} R =& \frac{V^2}{P}\\ =& \frac{(120V)^2}{60W}\\ =& \frac{14400V^2}{60W}\\ =& 240\Omega \end{aligned}$$

y como $I = \frac{V}{R}$

$$\begin{aligned} I_{RMS} &= \frac{120V}{240\Omega} = 0.5A\\ I_m &= \sqrt{2}\times I_{RMS} = \sqrt{2}\times 0.5A\\ &= 0.7A \end{aligned}$$

[5] El valor medio o efectivo de cualquier función $f(\omega t)$ con periodo de $2\pi rad$ está determinado por $F = \sqrt{\frac{1}{2\pi}\int_0^{2\pi} f^2(\omega t)\, d\omega t}$

Ahora bien, si se usa un ángulo de disparo en el TRIAC de $\alpha = \frac{\pi}{2}$ la potencia de el foco con base en las Ecuaciones (4) y (9) será:

$$\begin{aligned}
P_f &= \frac{V_o}{V_{RMS}} \\
&= \frac{\cancel{V_{RMS}} \left[\frac{1}{\pi}\left(\pi - \alpha + \frac{\sin(2\alpha)}{2}\right)\right]^{\frac{1}{2}}}{\cancel{V_{RMS}}} \\
&= \left[\frac{1}{\pi}\left(\pi - \frac{\pi}{2} + \frac{\sin\left(\frac{2\pi}{2}\right)}{2}\right)\right]^{\frac{1}{2}} \\
&= \left[\frac{1}{\pi}\left(\frac{\pi}{2} + \frac{\sin(\pi)}{2}^{0}\right)\right]^{\frac{1}{2}} \\
&= \left(\frac{1}{\cancel{\pi}} \cdot \frac{\cancel{\pi}}{2}\right)^{\frac{1}{2}} \\
&= \sqrt{\frac{1}{2}} \\
&= 0.707 \\
&= 70.7\,\%
\end{aligned}$$

De este desarrollo se concluye, además, que el factor de potencia no depende de los valores de voltaje, sino sólo del ángulo de disparo del TRIAC α.

Como $\alpha = \omega t$, para conocer el tiempo de disparo τ basta con sustituir $t = \tau$ y despejar. Así:

$$\begin{aligned}
\tau =& \frac{\alpha}{\omega} \\
=& \frac{\alpha}{2\pi f} \\
=& \left.\frac{\frac{\pi}{2}}{2\pi f}\right|_{f=60\text{Hz}} \\
=& \frac{1}{4 \times 60\text{Hz}} = \frac{1}{240\text{Hz}} \\
=& 0.004166\bar{6}\text{s} \\
\approx& 4.2\text{ms}
\end{aligned}$$

Un problema importante a considerar es que la ecuación $P_f = \left[\frac{1}{\pi}\left(\pi - \alpha + \frac{\sin(2\alpha)}{2}\right)\right]^{\frac{1}{2}}$ no es biyectiva (tiene una componente periódica senoidal) y por lo tanto no es posible calcular su inversa de forma analítica. En otras palabras, no es posible obtener una expresión algebraica para calcular $\alpha\,(P_f)$, y por tampoco es posible inferir $\tau\,(P_f)$.

Existen varias soluciones alterna a este problema. Por ejemplo, pueden utilizarse varios métodos de aproximación numérica para cada uno de los segmentos de la curva y utilizar el más adecuado para en cada uno de los rangos de interés. Otros métodos que requieren un número mucho menor de cálculos hacen uso de una tabla de valores discretos (por ejemplo incrementos de 1 % en el factor de potencia) e interpolan linealmente entre estos, dejando el error como una perturbación a corregir por el controlador (véase Tabla 1).

Tabla 1: Relación entre factor de potencia y tiempo de disparo de TRIAC
Tabla para interpolación con incrementos de 5 % y alimentación de AC a 60Hz

Factor de potencia [%]	Tiempo de disparo [ms]	Factor de potencia [%]	Tiempo de disparo [ms]	Factor de potencia [%]	Tiempo de disparo [ms]
100	0.000	65	4.464	30	6.232
95	2.060	60	4.734	25	6.487
90	2.696	55	4.993	20	6.750
85	3.157	50	5.245	15	7.030
80	3.538	45	5.493	10	7.334
75	3.874	40	5.738	5	7.688
70	4.179	35	5.984	0	8.203

2.6. Bus I²C

I²C es un protocolo serial inventado por Phillips y diseñado para conectar dispositivos de baja velocidad mediante interfaces de dos hilos (Figura 7). El protocolo permite un número virtualmente ilimitado de dispositivos interconectados donde más de uno puede ser un dispositivo maestro. El bus I2C es popular debido a su facilidad de uso y fácil configuración. Sólo es necesario definir la velocidad máxima del bus, que está conformado por dos cables con resistencias pull-up [3].

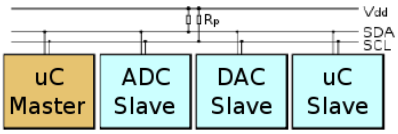


Figura 7: Bus I²C

I²C utiliza solamente dos cables: SCL (reloj) y SDA (datos). La transferencia de datos es serial y transmite paquetes de 8 bits con velocidades de hasta 5MHz. Además, es requisito que cada dispositivo esclavo tenga una dirección de 7 bits que (el bit más significativo se utiliza para indicar si el paquete es una lectura o una escritura) debe ser única en el bus. Los dispositivos maestros no necesitan dirección ya que estos generan la señal de reloj y coordinan a los dispositivos esclavos [3].

3. Material

Se asume que el alumno cuenta con un una Raspberry Pi con sistema operativo Raspbian e interprete de Python instalado. Se aconseja encarecidamente el uso de *git* como programa de control de versiones.

- 1 Arduino UNO o Arduino Mega
- 1 TRIAC BT138 o BT139
- 4 diodos 1N4007 o puente rectificador equivalente
- 1 optoacoplador MOC 3021
- 1 optoacoplador 4N25
- 1 foco incandescente (NO AHORRADOR NI LED)
- 1 resistencia de 15kΩ
- 1 resistencia de 10kΩ
- 2 resistencia de 4k7Ω
- 1 resistencia de 1kΩ
- 2 resistencia de 470Ω
- 2 resistencia de 330Ω
- 1 protoboard o circuito impreso equivalente
- 1 fuente de alimentación regulada a 5V y al menos 2 amperios de salida
- Cables y conectores varios

4. Instrucciones

1. Alambre el circuito mostrado en las Figuras 8 y 9.
2. Realice los programas de la Subsección 4.3
3. Analice los programas de la subsección 4.3 y realice los experimentos propuestos en la sección 5.

4.1. Paso 1: Alambrado

El proceso de alambrado de esta práctica considera dos circuitos. El primer circuito (véase Figura 8) opera con corriente alterna e integra un detector de cruce por cero y un convertidor AC-AC con base en un TRIAC. Ambos subcircuitos cuentan con optoacopladores que servirán como interfaz para una conexión segura al circuito de DC.

El segundo circuito (véase Figura 9) está encargado de detectar el cruce por cero y enviar la señal de activación al TRIAC en el momento oportuno de acuerdo con la potencia requerida por el usuario (el brillo del foco) mediante una interfaz gráfica.

Para este fin, se alambra la señal del subcircuito detector de cruce por cero a un pin te interrupción del Arduino que iniciará un *timer* en hardware y enviará la señal de activación al TRIAC una vez que transcurra el tiempo de activación para obtener así la potencia deseada. Asimismo, el Arduino recibirá via I^2C de la Raspberry Pi la potencia solicitada por el usuario mediante una interfaz web.

Cabe mencionar que, al ser un circuito completamente digital, al GPIO de la Raspberry Pi podría configurársele un pin en modo interrupción para recibir la señal del detector de cruce por cero y otro para el envío de la señal de activación del TRIAC. Sin embargo, el paquete `RPi.GPIO` para el control de la GPIO con Python no soporta el uso de interrupciones de timer en hardware, por lo que no es posible garantizar que la señal de activación del TRIAC será enviada sin retrasos. Es por este motivo que se utiliza un Arduino como auxiliar.

4.1.1. Circuito de potencia en AC

Alambre primero el circuito de corriente alterna de la Figura 8 tras verificar los valores de las resistencias de los optoacopladores. Considere que si la resistencia de gatillo es muy grande, el optoacoplador no recibirá suficente corriente y no encenderá lo suficiente como para disparar el fotosensor. Por otro lado, si la resistencia es demasiado pequeña el optoacoplador se quemará irremediablemente.

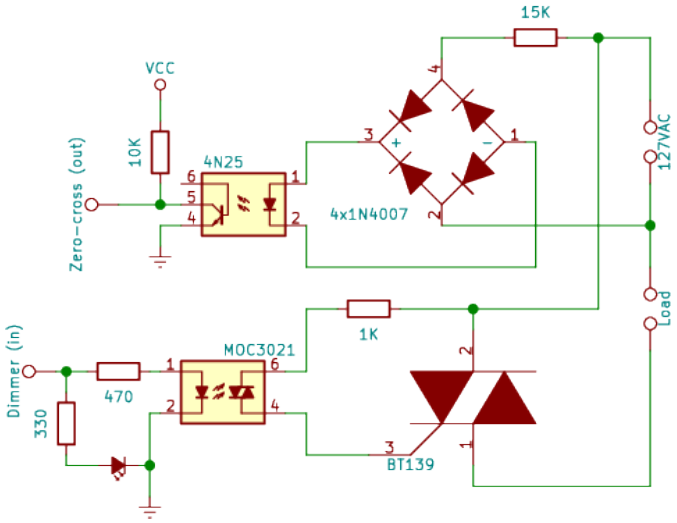


Figura 8: Circuito de potencia en AC

Tras alambrar el circuito, es una buena idea probar el detector de cruce por cero con un osciloscopio, o al menos con un led, que deberá encender tenuemente. De igual manera, conviene probar el encendido del foco inyectando 5V al MOC que acopla al TRIAC.

Tabla 2: Conexiones I^{2}C entre Raspberry Pi y un Arduino

Pin Raspberry		Conexión			Pin Arduino UNO	Pin Arduino Mega
3	(GPIO2)	Raspberry Pi SDA	→	Arduino SDA	A4	SDA (PIN 20)
5	(GPIO3)	Raspberry Pi SCL	→	Arduino SCL	A5	SCL (PIN 21)
6	(GND)	Raspberry Pi GND	→	Arduino GND	GND	GND

Importante

Asegúrese de verificar con un multímetro que el circuito de AC está debidamente aislado y que no se tienen valores mayores a 5V en el segmento de DC. De otro modo podría quemar su Arduino y su Raspberry Pi.

Continúe el alambrado del circuito.

4.1.2. Circuito de control en DC

Alambre el circuito de corriente directa de la Figura 9 tras verificar la tensión de las señales optoacopladas conectando el bus I^{2}C entre la Raspberry Pi y el Arduino como ilustran la Tabla 2 y la Figura 9. Hay tutoriales que sugieren utilizar un convertidor de niveles de voltaje cuando se conecta una Raspberry Pi a un arduino mediante I^{2}C, especialmente cuando la Raspberry Pi opera a 3.3V. Esto **NO** es necesario si la Raspberry Pi está configurada como dispositivo maestro o *master* y el Arduino como dispositivo esclavo o *slave*.

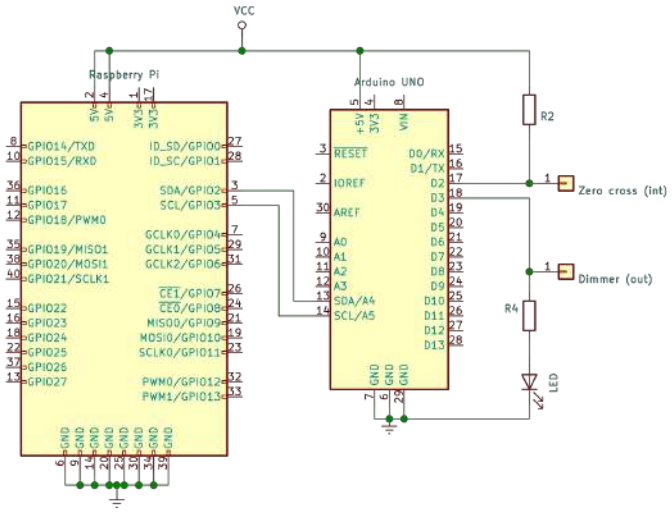


Figura 9: Circuito de control en DC

Esto es posible debido a que el Arduino no tiene resistencias de acoplamiento a positivo o *pull-up* integradas, mientras que los pines I^{2}C de la Raspberry Pi están conectados internamente a la línea de 3.3V mediante resistencias de 1.8kΩ. Por este motivo, tendrán que quitarse las resistencias de *pull-up* a cualquier otro dispositivo esclavo que se conecte al bus I^{2}C de la Raspberry Pi.[6]

A continuación pruebe el alambrado del circuito de DC con el programa de prueba del Apéndice A. El programa es muy simple, pues sólo configura interrupciones y cambia el estado de los pines cuando estas se producen.

Código ejemplo 1: `arduino-code-i2c.cpp:14` — Dirección asignada al dispositivo esclavo

```
1 #define I2C_SLAVE_ADDR 0x0A
```

[6]Para más información sobre el papel de las resistencias de acoplamiento a positivo o *pull-up* en un bus I^{2}C se puede consultar http://dsscircuits.com/articles/effects-of-varying-i2c-pull-up-resistors

Al terminar el alambrado debería tener completo el citcuito de la Figura 10

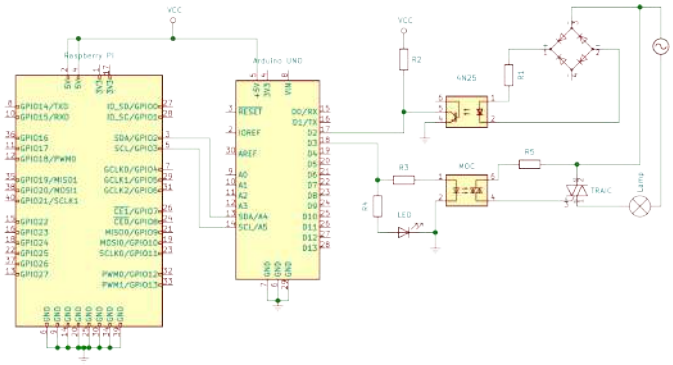


Figura 10: Diagrama de circuito alambrado completo

4.2. Paso 2: Configuración de comunicaciones I²C

Primero ha de configurarse la Raspberry Pi para funcionar como dispositivo maestro o *master* en el bus I²C. Para esto, inicie la utilidad de configuración de la Raspberry Pi con el comando

```
# raspi-config
```

y seleccione la opción 5: Opciones de Interfaz (*Interfacing Options*) y active la opción `P5` para habilitar el I²C.

A continuación, verifique que el puerto I²C no se encuentre en la lista negra. Edite el archivo `/etc/modprobe.d/raspi-blacklist.conf` y revise que la línea `blacklist spi-bcm2708` esté comentada con #.

Código ejemplo 2: /etc/modprobe.d/raspi-blacklist.conf

```
# blacklist spi and i2c by default (many users don't need them)
# blacklist i2c-bcm2708
```

Como paso siguiente, se habilita la carga del driver I²C. Esto se logra agregando la línea `i2c-dev` al final del archivo `/etc/modules` si esta no se encuentra ya allí.

Por último, se instalan los paquetes que permiten la comunicación mediante el bus I²C y se habilita al usuario predeterminado *pi* (o cualquier otro que se esté usando) para acceder al recurso.

```
# apt-get install i2c-tools python-smbus
# adduser pi i2c
```

Reinicie la Raspberry Pi y pruebe la configuración ejecutando `i2cdetect -y 1` para buscar dispositivos conectados al bus I²C. Debería ver una salida como la siguiente:

```
\$ i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

4.3. Paso 3: Control en lazo abierto de la potencia de una carga resistiva

Antes de proceder, verifique conexiones con un multímetro en busca de corto circuitos. En particular verifique que los circuitos de AC y DC funcionan de manera independiente y que existe una impedancia infinita entre pines optoacoplados.

Con la Raspberry Pi configurada, basta con generar los dos programas para transferir la potencia de salida deseada de la Raspberry Pi al Arduino quien se encargará cortar el flujo de corriente en el instante correcto para obtener la potencia deseada.

Primero, es necesario configurar al Arduino como dispositivo esclavo e inicializar el bus I^2C, tal como se muestra en los Códigos de Ejemplo 3 y 4. Las comunicaciones via I^2C son asíncronas, por lo que se requerirá almacenar la potencia deseada en una variable global que será leída por la función que atenderá las peticiones de datos del dispositivo maestro (la Raspberry Pi).

Código ejemplo 3: `arduino-code-i2c.cpp:14` — Dirección asignada al dispositivo esclavo

```
#define I2C_SLAVE_ADDR 0x0A
```

Código ejemplo 4: `arduino-code-i2c.cpp:28-33` — Configuración del bus I^2C y funciones de control

```
  // Configure I2C to run in slave mode with the defined address
  Wire.begin(I2C_SLAVE_ADDR);
  // Configure the handler for received I2C data
  Wire.onReceive(i2c_received_handler);
  // Configure the handler for request of data via I2C
  Wire.onRequest(i2c_request_handler);
```

Tanto el envío como la recepción de datos se realizan byte por byte, por lo que es necesario convertir la potencia (*float*) en un arreglo de bytes que pueda ser transmitido. Esto se hace en la funciones `i2c_request_handler` e `i2c_received_handler` tal como se ilustra en los Códigos de Ejemplo 5 y 6.

Código ejemplo 5: `arduino-code-i2c.cpp:46-48` — Escritura de flotantes en el bus I^2C

```
void i2c_request_handler(){
  Wire.write((byte*) &power, sizeof(float));
}
```

Código ejemplo 6: `arduino-code-i2c.cpp:54-62` — Lectura de flotantes del bus I^2C

```
void i2c_received_handler(int count){
  float f;
  byte *fp;
  if(count != 4) return;
  fp = &f;
  for(byte i = 0; i < count; ++i)
    fp[i] = (byte)Wire.read();
  power = f;
}
```

Del lado de la Raspberry Pi, primero ha inicializarse el bus I^2C y posteriormente se realizarán las lecturas en un poleo o bucle infinito, cada una de las cuales se irá almacenando en un archivo bitácora. La inicialización del bus requiere de una simple línea (véase Código de Ejemplo 7).

Código ejemplo 7: `raspberry-code-i2c.py:26` — Configuración del bus I^2C

```
  try:
```

La conversión de un arreglo de bytes a punto flotante en Python no es inmediata. Para esta opreación se utilizará la librería `struct` que empaquetará y desempaquetará flotantes (*float*) en listas de 4 bytes que pueden ser enviados o recibidos del arduino via I^2C tal como se muestra en los Códigos de Ejemplo 8 y 9

Código ejemplo 8: `raspberry-code-i2c.py:37-44` — Escritura de flotantes en el bus I^2C

```
def writePower(pwr):
  try:
    data = struct.pack('<f', pwr) # Packs number as float
    # Creates a message object to write 4 bytes from SLAVE_ADDR
    msg = smbus2.i2c_msg.write(SLAVE_ADDR, data)
    i2c.i2c_rdwr(msg)  # Performs write
  except:
    pass
```

Código ejemplo 9: `raspberry-code-i2c.py:25-35` — Lectura de flotantes del bus I^2C

```
def readPower():
  try:
    # Creates a message object to read 4 bytes from SLAVE_ADDR
    msg = smbus2.i2c_msg.read(SLAVE_ADDR, 4)
    i2c.i2c_rdwr(msg)  # Performs write
    data = list(msg)   # Converts stream to list
    pwr = struct.unpack('<f', ''.join([chr(c) for c in data]))
    # print('Received temp: {} = {}'.format(data, pwr))
    return pwr
  except:
    return None
```

El resto del programa es trivial, pues consiste sólo en solicitar al usuario un valor de potencia y enviarlo al Arduino.

Por conveniencia, los códigos completos de los programas de ejemplo se encuentran en los Apéndices A a C.

5. Experimentos

1. [6pt] Alambre el circuito completo y combine el código de los Apéndices A a C para poder controlar la intensidad del brillo del foco incandecente con la Raspberry Pi usando los valores tecleados en la consola (porentaje de 0–100 % de la potencia total).

2. [4pt] Modifique el código anterior para que la consola presente la potencia real modulada por el Arduino (equivalente en potencia del tiempo de encendido en milisegundos).

3. [+3pt] Modifique el código del punto 2 para que el arduino pueda modular la potencia del foco incandecente entre 0 % y 100 % con una resolución máxima de 1 %. Imprima la potencia reportada por el arduino con un dígito decimal.

4. [+2pt] Con base en lo aprendido, modifique el código delpunto 3 para que la Raspberry Pi sirva una página web donde se pueda modificar con un control gráfico la potencia de encendido del foco.

6. Referencias

Referencias

[1] *4N25 Optocoupler, Phototransistor Output, with Base Connection.* Vishay Semiconductors, 8 2010. Revised: January, 2010.

[2] Muhammad H Rashid. Power electronic, devices, circuits, and applications. *Handbook, Second Edition, Burlington*, 2006.

[3] I2C Info: A Two-wire Serial Protocol. I2c info – i2c bus, interface and protocol, 2020. https://i2c.info/, Last accessed on 2020-03-01.

[4] *MOC3021 Random-Phase Optoisolator TRIAC Driver Output.* Fairchild Semiconductors, 8 2010. Revised: January, 2010.

[5] The Arduino Project. Introduction to the arduino board, 2020. https://www.arduino.cc/en/reference/board, Last accessed on 2020-03-01.

A. Programa Ejemplo: `arduino-test-dc.cpp`

src/arduino–test–dc.cpp

```
1  // Digital 2 is Pin 2 in UNO
2  #define ZXPIN 2
3  // Digital 3 is Pin 3 in UNO
4  #define TRIAC 3
5
6  // Globals
7  volatile bool flag = false;
8  int step = 0;
9  int pfactor = 0;
10
11 // Prototypes
12 float read_temp(void);
13 float read_avg_temp(int count);
14
15 /**
16 * Setup the Arduino
17 */
18 void setup(void){
19   // Setup interupt pin (input)
20   pinMode(ZXPIN, INPUT);
21   attachInterrupt(digitalPinToInterrupt(ZXPIN), zxhandle, RISING);
22   // Setup output (triac) pin
23   pinMode(TRIAC, OUTPUT);
24   // Blink led on interrupt
25   pinMode(13, OUTPUT);
26
27   // Setup the serial port to operate at 9600bps
28   Serial.begin(9600);
29 }
30
31 void loop(){
32   if(!flag){
33     // Slack until next interrupt
34     delay(1);
35     return
36   }
37   // Reset flag & blink led
38   flag = !flag;
39   digitalWrite(13, LOW);
40   // Enable power for STEP milliseconds
41   if(pfactor < 8)
42     digitalWrite(3, HIGH);
43   delay(pfactor);
44   // Disable power
45   digitalWrite(3, LOW);
46   // Increment/reset power factor every 20 cycles
47   if(++step >= 19){
48     step = 0;
49     if(++pfactor > 8) pfactor = 0;
50   }
51 }
52
53 void zxhandle(){
54   flag = true;
55   digitalWrite(13, HIGH);
56 }
```

B. Programa Ejemplo: `arduino-code-i2c.cpp`

src/arduino–code–i2c.cpp

```cpp
#include <Wire.h>

// Constants
#define I2C_SLAVE_ADDR 0x0A
#define BOARD_LED 13

// Global variables
float power = 0;

// Prototypes
void i2c_received_handler(int count);
void i2c_request_handler(int count);

/**
* Setup the Arduino
*/
void setup(void){
  // Configure I2C to run in slave mode with the defined address
  Wire.begin(I2C_SLAVE_ADDR);
  // Configure the handler for received I2C data
  Wire.onReceive(i2c_received_handler);
  // Configure the handler for request of data via I2C
  Wire.onRequest(i2c_request_handler);

  // Setup the serial port to operate at 56.6kbps
  Serial.begin(56600);

  // Setup board led
  pinMode(BOARD_LED, OUTPUT);
}

/**
* Handles data requests received via the I2C bus
* It will immediately reply with the power stored
*/
void i2c_request_handler(){
  Wire.write((byte*) &power, sizeof(float));
}

/**
* Handles received data via the I2C bus.
* Data is stored in local variable power.
*/
void i2c_received_handler(int count){
  float f;
  byte *fp;
  if(count != 4) return;
  fp = &f;
  for(byte i = 0; i < count; ++i)
    fp[i] = (byte)Wire.read();
  power = f;
}

void loop(){
  char buffer[20];
  sprintf(buffer, "Power = %.2f\n", power);
  Serial.write(buffer)
  delay(1000);
}
```

C. Programa Ejemplo: `raspberry-code-i2c.py`

src/raspberry–code–i2c.py

```python
import smbus
import struct
import time

# Arduino's I2C device address
SLAVE_ADDR = 0x0A # I2C Address of Arduino 1

# Initialize the I2C bus;
# RPI version 1 requires smbus.SMBus(0)
i2c = smbus.SMBus(1)

def readPower():
  try:
    # Creates a message object to read 4 bytes from SLAVE_ADDR
    msg = smbus2.i2c_msg.read(SLAVE_ADDR, 4)
    i2c.i2c_rdwr(msg)  # Performs write
    data = list(msg)   # Converts stream to list
    pwr = struct.unpack('<f', ''.join([chr(c) for c in data]))
    # print('Received temp: {} = {}'.format(data, pwr))
    return pwr
  except:
    return None

def writePower(pwr):
  try:
    data = struct.pack('<f', pwr) # Packs number as float
    # Creates a message object to write 4 bytes from SLAVE_ADDR
    msg = smbus2.i2c_msg.write(SLAVE_ADDR, data)
    i2c.i2c_rdwr(msg)  # Performs write
  except:
    pass

def main():
  while True:
    try:
      power = input("Power? ")
      power = float(power)
      if power >= 0 and power <= 100:
        writePower(power)
        print("\tPower set to {}".format(readPower()))
      else:
        print("\tInvalid!")
    except:
      print("\tInvalid!")

if __name__ == '__main__':
  main()
```