

¿Qué es un Microcontrolador?

Guía del Estudiante para Experimentos 1 al 6

Versión en Castellano 1.1

PARALLAX 

Garantía

Parallax garantiza sus productos contra defectos en sus materiales o debidos a la fabricación por un periodo de 90 días. Si usted descubre un defecto, Parallax según corresponda, reparará, reemplazará o regresará el valor de la compra. Simplemente pida un número de autorización de regreso de mercadería (Return Merchandise Authorization, "RMA"), escriba el número en el exterior de la caja y envíela a Parallax. Por favor incluya su nombre, número telefónico, dirección, y una descripción del problema. Nosotros le regresaremos su producto o el reemplazo, usando el mismo método de correo que usted usó para enviar el producto a Parallax.

Garantía de 14 días de regreso de dinero

Si dentro de los 14 días en que usted recibió su producto, encuentra que no es conveniente para sus necesidades, puede regresarlo, recibiendo un reembolso. Parallax regresará el precio de compra del producto, excluyendo los costos de manipuleo y correo. Esto no se aplica si el producto a sido alterado o dañado.

Derechos de Copia y Marcas Registradas

Esta documentación tiene derechos de copia Copyright 1999 por Parallax, Inc. BASIC Stamp (Estampilla BASIC) es una marca registrada de Parallax, Inc. Otros nombres de productos son marcas registradas de sus respectivos dueños.

Desvinculación de Responsabilidad

Parallax, Inc. no es responsable de daños por consecuencias, incidentes o daños especiales que resulten de cualquier violación de la garantía, bajo cualquier teoría legal, incluyendo pérdida de beneficio, tiempo, daño o reemplazo de equipo o propiedad y cualquier costo, recuperando, reprogramando o reproduciendo cualquier dato guardado o usado dentro de los productos Parallax. Parallax tampoco es responsable de cualquier daño personal, incluyendo vida o muerte, resultado del uso de cualquiera de nuestros productos.

Acceso en Internet

Mantenemos sistemas de internet para su uso. Estos pueden ser usados para obtener software, comunicarse con miembros de Parallax, y comunicarse con otros clientes. Las rutas de acceso a la información se muestran a continuación:

E-mail: stampsinclass@parallaxinc.com

Ftp: [ftp.parallaxinc.com](ftp://ftp.parallaxinc.com) y [ftp.stampsinclass.com](ftp://ftp.stampsinclass.com)

Web: <http://www.parallaxinc.com> y <http://www.stampsinclass.com>

Lista de Discusión de BASIC Stamp en Internet (En Inglés)

Mantenemos una lista de discusión por e-mail para gente interesada en el BASIC Stamp. La lista trabaja así: mucha gente se suscribe a la lista y luego todas las preguntas y respuestas son distribuidas a todos los suscriptos. Es una forma rápida, divertida y gratis de discutir temas sobre el BASIC Stamp y obtener respuestas a preguntas técnicas. Para suscribirse a la lista de BASIC Stamp mande un e-mail a majordomo@parallaxinc.com y escriba *subscribe stamps* en el cuerpo del mensaje.

También mantenemos una lista exclusiva para educadores que usan el BASIC Stamp en el aula. Los educadores discuten temas como protección del BASIC Stamp contra daños, creación de sus propias lecciones, usando el material Parallax Stamps in Class (Stamp en Clase), y robótica. Usted puede unirse a esta lista en el sitio web <http://www.stampsinclass.com>.

Prefacio	3
Guías para estudiantes y Profesores	3
Experimentos Futuros	4
Derechos de Copia y Reproducción	4
Agradecimientos Especiales	4
Experimento 1: ¿Qué es un Microcontrolador?	7
Partes Requeridas	9
¡Constrúyalo!	9
¡Prográmelo!	13
Preguntas.....	20
¡Desafío!.....	21
¿Qué aprendí?.....	22
¿Por qué aprendí esto?	23
¿Cómo puedo aplicarlo?.....	23
Experimento 2: Detectando el Mundo Exterior	25
Partes Requeridas	25
¡Constrúyalo!	26
¡Prográmelo!	27
Preguntas.....	33
¡Desafío!.....	34
¿Qué aprendí?.....	35
¿Por qué aprendí esto?	36
¿Cómo puedo aplicarlo?.....	36
Experimento 3: Movimiento Micro-controlado:	37
Partes Requeridas	38
¡Constrúyalo!	38
¡Prográmelo!	40
Preguntas.....	47
¡Desafío!.....	48
¿Qué aprendí?.....	49
¿Por qué aprendí esto?	50
¿Cómo puedo aplicarlo?.....	50
Experimento 4: Automatización Simple	51
Partes Requeridas	52
¡Constrúyalo!	52
¡Prográmelo!	54
Preguntas.....	60
¡Desafío!.....	61
¿Qué aprendí?.....	62

Contenido

¿Por qué aprendí esto?	63
¿Cómo puedo aplicarlo?.....	63
Experimento 5: Midiendo una Entrada.....	65
Partes Requeridas.....	66
¡Constrúyalo!	66
¡Prográmelo!.....	69
Preguntas.....	73
¡Desafío!.....	74
¿Qué aprendí?.....	75
¿Por qué aprendí esto?	76
¿Cómo puedo aplicarlo?.....	76
Experimento 6: Manual a Digital	77
Partes Requeridas.....	78
¡Constrúyalo!	78
¡Prográmelo!.....	81
Preguntas.....	87
¡Desafío!.....	88
¿Qué aprendí?.....	89
¿Por qué aprendí esto?	90
¿Cómo puedo aplicarlo?.....	90
Lista de Partes	91
Apéndice A: Lista de Componentes y Suministros.....	91
Apéndice B: Guía de Referencia Rápida de PBASIC.....	99
Apéndice C: Leyendo el Código de Colores de los Resistores	111

Prefacio

En 1979 Chip Gracey tuvo su primera introducción a la programación y la electrónica: la computadora Apple II. Chip se interesó instantáneamente en la nueva máquina, escribiendo código BASIC para mostrar gráficos y quitando la tapa para mostrar los componentes electrónicos. Esta experiencia lo llevó rápidamente a dismantelar el código fuente de los videojuegos y electrodomésticos hogareños y a tratar de usar estos artefactos para propósitos distintos de los que originalmente tenían, transformando el hobby en un negocio. A la vez que se recibía en el colegio, Chip comenzaba un pequeño negocio desde su pieza.

Los colegios no ofrecían clases sobre montajes electrónicos ni programación en 1982, y cuando Chip se graduó en 1986 no se veía como el lugar correcto para comenzar un negocio. En lugar de eso, él y un amigo, Lance Walley comenzaron "Parallax" desde su apartamento. Los primeros productos incluyeron digitalizadores de sonido para la Apple II y programadores 8051.

El negocio creció lentamente hasta 1992 cuando Parallax realizó el primer BASIC Stamp. Parallax reconoció que el BASIC Stamp debería ser especial – era la herramienta que ellos necesitaban para hacer sus propios proyectos de hobby. El hecho que el BASIC Stamp crearía su propia industria era probablemente desconocido por los fundadores de Parallax, pero rápidamente se volvió aparente que la pequeña computadora tenía su propio grupo de entusiastas. Este le permitía a la gente común programar un microcontrolador de una sola vez y les daba poderosos comandos de entrada-salida que lo hicieron fácil de conectar a otros componentes electrónicos. Hacia el final de 1998 Parallax vendió más de 125,000 módulos de BASIC Stamp y distribuyó una serie completa de periféricos a través de 40 canales de venta mundiales.

¿Qué significa toda ésta historia dentro de un curriculum de Stamps en Clase? El curriculum de Stamps en Clase está diseñado para introducir a los estudiantes y profesores al uso de los microcontroladores usando BASIC y circuitería simple, integrando los dos sin un gran esfuerzo (el curriculum es gratis y Parallax tiene precios educacionales para los circuitos). Este comienza desde abajo con proyectos y programación prácticas. Esta es la forma en que Chip aprendió microcontroladores y ganó suficiente experiencia en ingeniería para diseñar el BASIC Stamp.

Guías para estudiantes y Profesores

El curriculum "¿Qué es un Microcontrolador?" fue creado para edades entre 15-18 años con el propósito de proveer un entorno de bajo nivel a la programación e interconexión de microcontroladores, pero también es apropiado para clases en los colegios con material suplementario para el instructor adicional. El curriculum combina la programación y circuitería, primero mostrando al estudiante cómo construir el circuito, luego programar el microcontrolador, y finalmente desafiándolo a mejorar el diseño. La guía del profesor para cada lección está disponible solicitándola por correo electrónico e-mail a stampsinclass@parallaxinc.com.

Por supuesto, alguna base de electrónica sería de ayuda. Los principios de electrónica no están explicados con detenimiento debido a que hay grandes textos a ser leídos junto con éste currículum. La mejor compañía es "Comenzando en electrónica" de Radio Shack Forrest Mimm. Otros recursos más avanzados incluyen "Programando y manejando la computadora BASIC Stamp" de Scott Edwards' y el "Manual del BASIC Stamp Versión 1.9". Ver Apéndice A para más bibliografía.

Experimentos Futuros

La serie “¿Qué es un microcontrolador?”, está compuesta por los primeros seis experimentos incluidos en éste libro. Lecciones futuras también serán publicadas en juegos de seis. El contenido de éstos experimentos está basado en la realimentación de estudiantes e instructores. Si usted tiene ideas para experimentos futuros, por favor envíelas al grupo editorial a stampsinclass@parallaxinc.com. También aceptamos contribuciones a experimentos, sólo asegúrese de ver las pautas editoriales de nuestro sitio de web de Stamps en Clase.

Derechos de Copia y Reproducción

Las lecciones de Stamps en Clase tienen derechos de copia por Parallax 1999. Parallax garantiza a cualquier persona un derecho condicional de descargar, duplicar y distribuir éste texto sin su permiso. La condición es que éste texto o cualquier porción de él, no debería ser duplicada para uso comercial, resultando un cargo al usuario más allá del costo de impresión. Esto quiere decir que nadie debería lucrar por la duplicación de éste texto. Preferiblemente la duplicación no debería tener costo para el estudiante. Cualquier institución educativa que desee producir duplicados para sus estudiantes, puede hacerlo sin nuestro permiso. Este texto está también disponible en formato impreso por Parallax. Debido a que imprimimos el texto en cantidades, el precio al consumidor es bastante menos que una fotoduplicación. Este texto puede ser traducido a cualquier lengua con el permiso de Parallax, Inc.

Agradecimientos Especiales

Este curriculum fue escrito por I-Four de Grass Valley, California. El principal autor de I-Four es Matt Gilliland. Parallax se tropezó con Matt en internet. Matt es un verdadero educador práctico. Él, personalmente probó estos experimentos con un grupo de vecinos entusiastas. Su realimentación inicial fue importante debido a que resultó en revisiones sustanciales (si usted encuentra cualquier error o áreas que deban ser desarrolladas, por favor háganoslo saber por e-mail a stampsinclass@parallaxinc.com). I-Four también escribió porciones del programa de Electrónica del Nivel Secundario de Ciencia (FOSS), con “The Lawrence Hall of Science at University of California, Berkeley”.

Agradecemos especialmente también al grupo de Parallax que nos proveyó de ideas y contenidos para el programa. Los miembros de Parallax que diseñan, fabrican, reciben los pedidos y envían los productos de Stamps en Clase, son una parte clave del programa Stamps en Clase.

Por supuesto, se necesita un BASIC Stamp para hacer el programa posible. Gracias a Chip por crear éste producto único y una nueva industria.

Traducción

La Versión en Castellano 1.1 de "¿Qué es un Microcontrolador?", es la traducción correspondiente a "What's a Microcontroller?" Version 1.5.

Traducido y adaptado al castellano por Aristides Alvarez y Ana Lusi, Argentina.

e-mail: aristides@copetel.com.ar

Sitio web en castellano www.cursoderobotica.com.ar.

Prefacio



Experimento 1: ¿Qué es un Microcontrolador?

Muchos de nosotros sabemos qué apariencia tiene una computadora. Usualmente tiene teclado, monitor, CPU (Unidad de Procesamiento Central), impresora y mouse. Este tipo de computadoras, como la Mac o PC, son diseñadas principalmente para comunicarse con humanos.

Manejo de base de datos, análisis financieros o incluso procesadores de textos, se encuentran todos dentro de la "gran caja", que contiene CPU, la memoria, el disco rígido, etc. El verdadero "cómputo", sin embargo, tiene lugar dentro de la CPU.

¿Qué es un CPU

Central Processing Unit. (Unidad de procesamiento Central). Este término se refiere específicamente al circuito integrado (contenido dentro de la caja de la computadora), que hace los "verdaderos cómputos". Sin embargo, a veces el término es usado (aunque incorrectamente), para incluir todo lo que está dentro de la caja, incluyendo el disco rígido, la disquetera, el CD ROM, la fuente y la motherboard.

Microcontrolador

Es un circuito integrado que contiene muchas de las mismas cualidades que una computadora de escritorio, tales como la CPU, la memoria, etc., pero no incluye ningún dispositivo de "comunicación con humanos", como monitor, teclados o mouse. Los microcontroladores son diseñados para aplicación de control de máquinas, más que para interactuar con humanos.

Si piensa sobre esto, el único propósito del monitor, teclado, mouse e incluso la impresora, es "conectar" a la CPU con el mundo exterior.

¿Pero usted sabía que hay computadoras alrededor de nosotros, corriendo programas y haciendo cálculos silenciosamente sin interactuar con ningún humano? Estas computadoras están en su auto, en el transbordador espacial, en un juguete, e incluso puede haber uno en su secador de pelo.

Llamamos a éstos dispositivos "microcontroladores". Micro porque son pequeños, y controladores, porque controlan máquinas o incluso otros controladores. Los Microcontroladores, por definición entonces, son diseñados para ser conectados más a máquinas que a personas. Son muy útiles porque usted puede construir una máquina o artefacto, escribir programas para controlarlo, y luego dejarlo trabajar para usted automáticamente.

Hay un número infinito de aplicaciones para los microcontroladores. ¡Su imaginación es el único factor limitante!

Cientos (sino miles) de variaciones diferentes de microcontroladores están disponibles. Algunos son programados una vez y producidos para aplicaciones específicas, tales como controlar su horno microondas. Otros son "reprogramables", que quiere decir que pueden ser usados una y varias veces para diferentes aplicaciones. Los Microcontroladores son increíblemente versátiles, el mismo dispositivo puede controlar un aeromodelo, una tostadora, o incluso el ABS de su auto (sistema antibloqueo).

Este experimento nos introduce en un microcontrolador muy popular llamado **BASIC Stamp**. El BASIC Stamp es un conjunto sofisticado de circuitos, todos ensamblados en una pequeña plaqueta de circuito impreso (**PCB**). En realidad, el PCB tiene el mismo tamaño de muchos otros tipos de "circuitos integrados". El BASIC Stamp es mostrado en la Figura 1.1.

Experimento 1: "¿Qué es un Microcontrolador?"

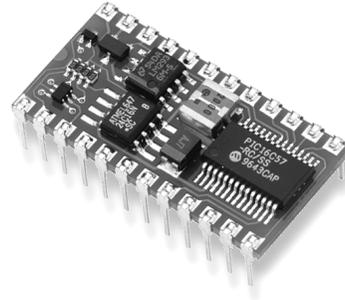
¿Qué es Un

PCB:

Printed Circuit Board. (Plaqueta de circuito impreso). Los circuitos electrónicos complejos requieren muchas conexiones eléctricas entre componentes. Una PCB es simplemente una pieza rígida, normalmente de fibra de vidrio, que tiene muchos cables de cobre sobre su superficie (o algunas veces dentro). Estos cables llevan las señales entre los componentes del circuito.

Figura 1.1: BASIC Stamp II

Esta es la imagen del módulo BASIC Stamp II. El módulo real tiene aproximadamente el tamaño de una estampilla.



La escritura de programas para el BASIC Stamp se realiza con la versión especial del lenguaje BASIC (se llama PBASIC). La mayoría de los otros microcontroladores requieren alguna forma de programación que puede ser muy difícil de aprender. Con el BASIC Stamp, usted puede crear circuitos simples y programas en minutos, (que es lo que estamos por hacer). Sin embargo, no debe pensar erróneamente que todo lo que el BASIC Stamp puede hacer son cosas simples. Muchos productos comerciales sofisticados han sido creados y vendidos usando un BASIC Stamp como "cerebro".

Cuando creamos dispositivos que tienen un microcontrolador actuando como un "cerebro", en muchas formas estamos tratando de imitar cómo actúa nuestro cuerpo.

Su cerebro necesita cierta información para tomar decisiones. Esta información es obtenida a través de varios sensores, como la vista, el oído, el tacto, etc. Estos sensores detectan lo que nosotros llamamos el "mundo real" o mundo exterior, y envían esa información al cerebro para "procesamiento". Recíprocamente, cuando su cerebro toma una decisión, manda señales a través de su cuerpo para hacer algo en el "mundo exterior". Utilizando las "entradas" de sus sentidos, y las "salidas" de sus piernas, brazos, manos, etc., su cerebro se está comunicando e interactuando con el mundo exterior.

Cuando usted va manejando por la ruta, sus ojos detectan un ciervo corriendo frente a usted. Su cerebro analiza esta "entrada", toma una decisión y "emite" instrucciones a sus brazos y manos, girando el volante para evitar golpear al animal. Esta "entrada/decisión/salida" es de lo que se tratan los microcontroladores. Nosotros llamamos a esto "entrada/salida" o "e/s", para abreviar.

Esta primera lección lo introduce en la función de salida del BASIC Stamp, y en cada lección siguiente introduce nuevas ideas y experimentos para que usted realice. Usted será capaz de usar las ideas de éstas lecciones para inventar sus propias aplicaciones para programas de microcontroladores y circuitos.

Experimento 1: "¿Qué es un Microcontrolador?"

¿Qué es Un...

LED:

Light Emitting Diode. (Diodo Emisor de Luz). Es un tipo especial de diodo semiconductor, que cuando es conectado a un circuito electrónico con un resistor limitador de corriente, emite luz visible. Los LED usan muy poca energía y son ideales para ser conectados a dispositivos tales como el Stamp.

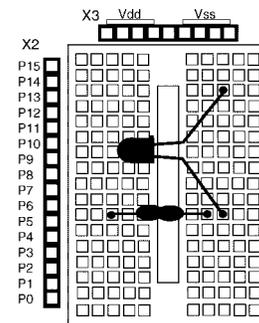
En este experimento conectaremos dos diodos emisores de luz (**LEDs**) al BASIC Stamp. Los LEDs son formas especiales de lámparas que por varias razones son fácilmente conectadas a dispositivos microcontroladores.

Hay dos cosas muy importantes de recordar cuando conecta LEDs al BASIC Stamp. La primera es que *siempre se debe asegurar que haya una resistencia conectada*, como muestra la figura 1.3. En este experimento el resistor debe tener un valor de 470 ohms, ¼ watt. Ver Apéndice C para información adicional.

Segundo, esté seguro que la *polaridad del LED* es la correcta. Hay una zona lisa en un costado del LED que debería ser conectada como en la Figura 1.3. Si la polaridad es invertida, el LED no trabaja. El lado liso también tiene la pata más corta del LED.

Figura 1.3: LED en la Protoboard

Muestra el LED y el resistor enchufados en la protoboard. Ninguna conexión ha sido hecha aún a las entradas/salidas del BASIC Stamp.



Cuando inserte un LED en la protoboard no doble las patas en ángulos rectos a corta distancia del cuerpo del LED, debido a que algunos no soportan un esfuerzo sobre el plástico.

Entendiendo la Protoboard.

El BASIC Stamp tiene un total de 24 patas o pines, como se muestra en la Figura 1.1. Algunas de estas señales son usadas para conectar el BASIC Stamp a la PC y a la batería de 9 volts (o la fuente externa). Dieciséis de estas señales (P0 a P15) están disponibles para que nosotros las conectemos al "mundo exterior".

En la Plaqueta de Educación, usted puede seguir un "trazo" desde el módulo del BASIC Stamp a la línea de fichas en el costado izquierdo de la protoboard. Cada pin de entrada/salida del BASIC Stamp es traído hasta el borde de la protoboard y con cables, usted puede unir desde las fichas a la protoboard.

Conectando un LED:

Nunca conecte un LED al Stamp, sin tener un resistor (del valor apropiado) en el circuito. El resistor limita la cantidad de corriente que fluye en el circuito a un nivel seguro, protegiendo así al LED y al Stamp.

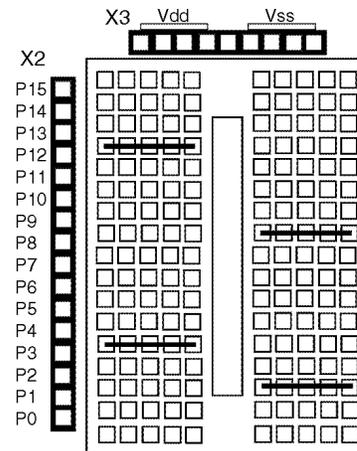
Es importante entender cómo trabaja la protoboard. La protoboard tiene muchas tiras metálicas que pasan por debajo en fila. Estas tiras conectan los huecos unos a otros; esto hace fácil conectar componentes juntos para construir un circuito eléctrico.

Para usar la protoboard, las patas del LED y el resistor, serán puestas en los huecos. Estos huecos son hechos de forma que tendrán al componente en su lugar. Cada hueco es conectado a una de las tiras metálicas que corren por debajo de la plaqueta. Usted puede conectar diferentes componentes enchufándolos dentro de nodos comunes. La figura 1.4 ilustra éste concepto.

Cada pin de BASIC Stamp tiene un "nombre de señal" asociado a él. Por ejemplo el pin 24 es VIN (que quiere decir "entrada de voltaje"). Esta es una de las conexiones para la batería de 9 volts. Cuando usted enchufa la batería, una conexión es hecha de la batería a éste pin mediante una pista de cobre que está sobre la Plaqueta de Educación.

Figura 1.4: Conexiones de la Protoboard.

Las líneas negras horizontales muestran cómo los huecos están conectados por debajo de la protoboard. Esto significa que usted no tiene que enchufar dos cables en un mismo hueco, debido a que ese hueco está unido con el de la izquierda o el de la derecha.



Los pines que usaremos en éste experimento son los siguientes:

Pin	Nombre de señal
5	P0
6	P1
21	Vdd (+5 volts)

Cuando programemos el Basic Stamp, nos referiremos al **Nombre de Señal**, más que al verdadero número de pin.

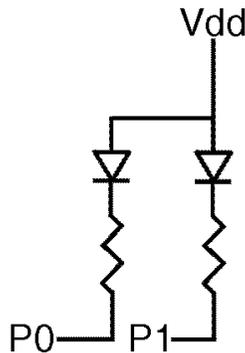
Experimento 1: "¿Qué es un Microcontrolador?"

¿Qué es un...

Esquema:

Un diagrama eléctrico que muestra conexiones entre componentes que no necesariamente se ve, como el circuito físico. Usamos diagramas esquemáticos debido a que ellos ayudan a entender el flujo de la señal, a través de circuitos complejos.

Figura 1.5: Esquema
Diagrama eléctrico del experimento 1

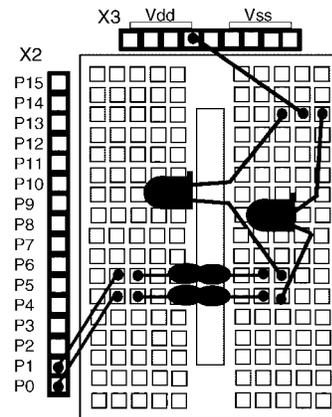


Bien, ¡ construyamos el circuito! No conecte la fuente de alimentación aún (la batería de 9 volts o el adaptador externo).

Las Figuras 1.5 y 1.6 son dos métodos diferentes de mostrar un diagrama eléctrico. La Figura 1.5 es un diagrama "esquemático" del circuito. La Figura 1.6 es el mismo circuito, pero dibujado como una foto de cómo se vería el circuito físico. En cada experimento mostraremos un diagrama esquemático y un dibujo físico hasta que progrese a lecciones más avanzadas.

Figura 1.6: Diagrama Físico:

Muestra como se ve el circuito físicamente, después de que lo construya. El lado liso del LED es el más cercano al resistor.



Conecte el primer LED:

1. Conecte un cable en P0 y luego en la protoboard como se muestra. Luego conecte un resistor en la protoboard al lado del cable y enchufe el otro extremo del resistor del otro lado de la protoboard.
2. Enchufe el LED en la protoboard al lado del resistor. Asegúrese de conectar al lado del resistor la pata del lado liso del LED.
3. Enchufe la otra pata del LED a Vdd (+5v) en la Plaqueta de Educación.

Conecte el segundo LED:

1. Enchufe un cable en la posición P1 y conéctelo a la protoboard. Luego enchufe un resistor en la protoboard al lado del cable y enchufe el otro extremo del resistor en el costado derecho de la protoboard.
2. Enchufe el LED en la protoboard al lado del resistor. Asegúrese de conectar la pata del lado liso del LED junto al resistor.
3. Conecte la otra pata del LED a Vdd (+5v) de la Plaqueta de Educación, usando un alambre conector.

Conecte la Plaqueta de Educación a la PC:

1. Enchufe un extremo del cable de programación en la Plaqueta de Educación.
2. Enchufe el otro extremo del cable de programación en un conector del puerto serial disponible en la PC.

**¡Programémoslo!****¿Qué es Un Programa?****Programa:**

Una secuencia de instrucciones que son ejecutadas por una computadora o un microcontrolador, en una secuencia específica, para realizar una tarea. Los programas son escritos en diferentes tipos de lenguajes, tales como "C", Fortran o BASIC

¡Listo! Ya hemos creado un circuito ("hardware"). Pero esto no es nada aún. Así es que necesitamos...

¿Cuántos de ustedes ya saben escribir un programa de computadora? Si lo ha hecho antes, entonces la primera parte de ésta sección puede ser un repaso. Pero si es nuevo en el tema, no se preocupe, en realidad no es tan difícil.

Un programa de computadora no es nada más que una lista de instrucciones que una computadora ejecuta (o en nuestro caso un microcontrolador). Creamos un programa para el microcontrolador escribiéndolo en una PC (utilizando el teclado y el monitor), luego descargamos este "código" a través del cable de programación, al microcontrolador. Este programa (o lista de instrucciones), entonces se "ejecuta" dentro del BASIC Stamp.

¿Qué es Un Error?**Error:**

Un error en su programa o hardware. Aplicar "debug" (depurar) en su programa es revisar y eliminar errores en su código. También puede haber errores de hardware tales como un LED puesto al revés, que causa que el sistema no funcione.

Si hemos escrito el programa correctamente, hará lo que nosotros esperábamos. Sin embargo, si hemos cometido un error, entonces el dispositivo no trabajará (o trabajará mal), y necesitamos depurarlo (debug). Depurar el programa puede ser una de las experiencias más estresantes de todo el proceso, por lo tanto, cuánto más cuidadoso es en crear el programa, teóricamente más fácil será depurarlo. Un "debugging" es el arte de remover errores (bug).

PBASIC para el BASIC Stamp, tiene muchos comandos de donde elegir, 36 para ser exactos. Una lista y descripción completas de cada uno de éstos comandos

Experimento 1: "¿Qué es un Microcontrolador?"

puede ser obtenida del Manual del Basic Stamp Versión 1.9, pero cada comando usado en éstas lecciones está descrito en el apéndice B, referencia rápida del PBASIC.

Para éste experimento usaremos sólo cuatro comandos.

Estos son: **OUTPUT, PAUSE, GOTO, Y OUT.**

Como se mencionó arriba, un programa es una lista de instrucciones que son ejecutadas en una secuencia determinada por la estructura del programa en sí mismo. Por lo tanto, como escribimos el programa, es muy importante para tener en mente la secuencia de la ejecución que nosotros deseamos.

Por ejemplo, si queremos comprar una gaseosa en una máquina expendedora, nuestro cerebro ejecuta una lista de comandos para realizar esto. Tal vez algo así...

1. Insertar \$1.00 en la ranura.
2. Esperar que se encienda la luz verde.
3. Presionar el botón para el tipo de gaseosa.
4. Ver salir la gaseosa.
5. Agarrar la gaseosa.
6. Abrir la gaseosa.
7. Beber la gaseosa.
8. Burp.

Ahora, esto se ve bastante lógico, pero sólo porque ya lo hemos hecho antes.

Si, sin embargo, su cerebro envía el programa siguiente:

1. Presionar el botón para el tipo de gaseosa.
2. Abrir la gaseosa.
3. Insertar \$1.00 en la ranura.
4. Agarrar la gaseosa.
5. Burp
6. Beber la gaseosa.
7. Esperar que se encienda la luz verde.
8. Ver salir la gaseosa.

No pasará mucho. Todos los comandos apropiados están ahí, pero en un orden incorrecto. Una vez que usted ha apretado el botón para tipo de gaseosa (botón 1), su cerebro (programa), se "colgará" o se detendrá, porque no puede ejecutar "abrir la gaseosa", que es el punto 2, ¡porque no tiene gaseosa que abrir!.

Experimento 1: "¿Qué es un Microcontrolador?"

Este es un error de programación (bug). Los humanos podemos modificar nuestro programa cerebral a medida que la situación transcurre, y podemos, por supuesto imaginarnos cómo obtener la gaseosa.

Los microcontroladores, sin embargo, no tienen la capacidad de adaptar y modificar su propio conjunto de instrucciones; ellos son solamente capaces de ejecutar la secuencia exacta de las instrucciones que nosotros les damos.

Bien, suficiente teoría, ¡ programemos éste microcontrolador para hacer algo!

Conecte la batería de 9 volts a la plaqueta de educación. Conecte el cable serial a su PC. Enchufe el BASIC Stamp II a la plaqueta de Educación, con el integrado grande hacia la parte inferior de la plaqueta.

Encienda su PC. El software del BASIC Stamp funciona en DOS y Windows 95/98/NT4.0. Asumiremos que usted está usando una computadora con Windows 95. Usted primero necesita copiar el contenido del disco en una carpeta en su PC.

¿Qué es un Stamp?

Descarga:

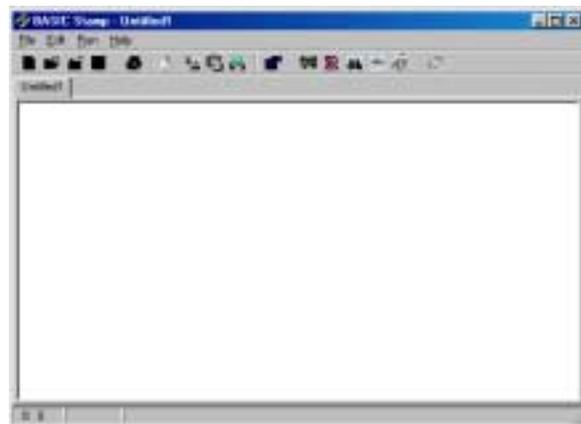
Después que un programa de un microcontrolador ha sido creado en la PC, es enviado desde la PC por un cable y cargado en la memoria del microcontrolador. El programa es ejecutado dentro del Stamp.

Doble click en el icono BASIC Stamp

Usted ahora, debe estar corriendo un programa llamado "Stamp Editor". Este es un programa que fue creado para ayudarlo a escribir y descargar programas al microcontrolador BASIC Stamp. La pantalla se verá como en la Figura 1.7:

Figura 1.7: Software del BASIC Stamp

Doble click sobre el icono BASIC Stamp para hacer correr el software. La pantalla que se abre se verá como ésta.



La pantalla, excepto por unas pocas palabras en el título, está en blanco. Acá es donde usted creará su programa. Ahora recuerde, vamos a escribir nuestro programa usando un equipo de "comunicación humano" (monitor, teclado, etc.),

INFORMACIÓN

El Editor del Stamp

Si usted está usando la versión de DOS, apretando la tecla F1, primero le mostrará cuantas variables ha usado. Presionando la barra espaciadora, se mueve entre (1) variable, (2) memoria de programa general y (3) memoria de programa detallada. Para averiguar qué tan grande es su programa, simplemente tenga apretada la tecla ALT y presione "m".

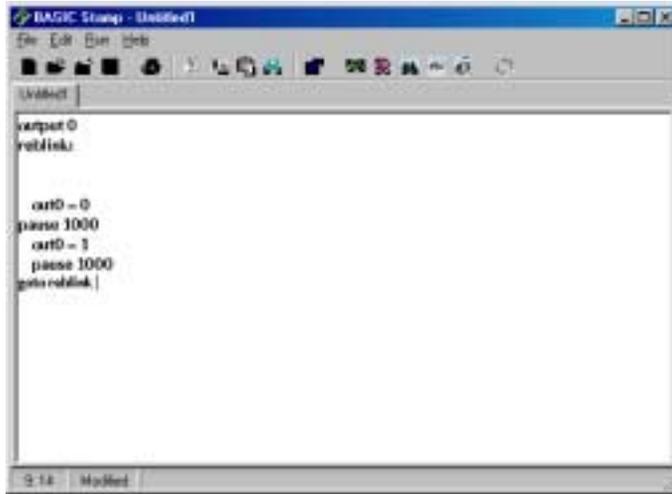
Experimento 1: "¿Qué es un Microcontrolador?"

que es parte de su PC. El programa que escribiremos, no correrá en su PC, sino que será descargado o enviado al microcontrolador. Una vez que el programa ha sido recibido, el BASIC Stamp ejecutará las instrucciones exactamente como nosotros las hemos creado.

Escriba el siguiente programa en el editor del BASIC Stamp, y se verá como en la figura 1.8:

```
output 0
titilar:
  out0 = 0
  pause 1000
  out0 = 1
  pause 1000
goto titilar
```

Figura 1.8: Software del BASIC Stamp
Escriba el código en el editor, y se verá como en ésta pantalla.



Ahora, mientras tiene apretada la tecla "ALT" presione "R" (de "run" correr), y luego presione "ENTER" cuando el menú muestre el comando RUN. Si todo está correcto, el LED que está conectado a P0 (pin 5 de la Plaqueta de Educación), debería titilar.

Si usted obtiene un mensaje que dice "Hardware not found", (no se encuentra el circuito), revise las conexiones del cable entre la PC y la plaqueta de educación, y también asegúrese que una fuente de alimentación está conectada a la plaqueta de educación. Si aún no funciona, revise bajo el menú EDIT la opción PREFERENCIAS y EDITOR OPERATION. El ajuste del puerto COM por defecto, debería estar en AUTO (automático).

Trate de descargar el programa nuevamente (tenga presionada la tecla ALT y presione "r"). Si aún no funciona, usted debe tener un error (bug). Revise nuevamente su programa para asegurarse que lo haya escrito correctamente.

Si después de tratar esto, usted aún tiene problemas, pídale ayuda a su instructor. Ahora analicemos nuestro programa.

El primer comando usado es "**output**". Cada señal (P0 y P15) puede ser ajustada como "entrada" o "salida". Debido a que nosotros queremos que el microcontrolador encienda y apague un LED, el microcontrolador está *manipulando* el "mundo exterior". Por lo tanto, por definición, nosotros queremos que P0 sea una salida "**output**".

Resultado del primer comando: "**output 0**" hace P0 una salida. (Aclaración: si nosotros buscáramos hacer P1 una salida, el comando debería haber sido "output 1").

El siguiente ítem en el programa **"titilar:"**, no es realmente un comando. Es sólo una etiqueta, una marca en cierto punto del programa. Volveremos a esto más adelante.

Ahora, el pin 5 o P0 como lo llamamos, es una salida. En el mundo de las computadoras, voltajes en éstos pines, pueden ser tanto "altos" o "bajos", que significa que pueden tener un voltaje alto o bajo. Otra forma de referirnos a alto y bajo es "1" y "0". "1" significa alto y "0" significa bajo.

INFORMACIÓN

Salida:

Hablando técnicamente, "Out" no es realmente un comando. Usamos el registro "out" para hacer una salida tanto alta o baja. En un experimento próximo, exploraremos los registros en más detalle.

Piense en una llave de luz en la pared, cuando la llave está en una posición, la lámpara se enciende, y cuando está en otra posición, se apaga. Es binaria, hay sólo dos combinaciones: encendida o apagada, o "1" o "0". No importa cuánto insista, usted nunca podrá poner la llave "en medio" de las dos posiciones.

Si queremos encender el LED necesitamos hacer que P0 vaya abajo (que tenga un 0). P0 está actuando como un interruptor, que puede ser cambiado a encendido o apagado bajo un control de programa. Circuitos simplificados son mostrados en la Figura 1.9 (LED apagado) y en la Figura 1. 10 (LED encendido). El flujo de corriente es el voltaje positivo a través del resistor, el LED, y entrando a P0, donde P0 es conectado abajo.

Figura 1.9: LED apagado

Cuando P0 está en "nivel alto", no hay flujo de corriente.

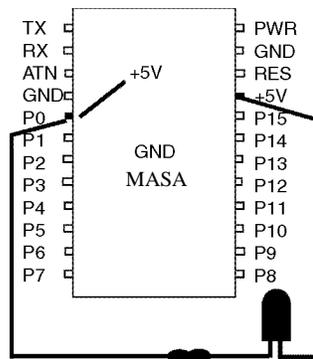
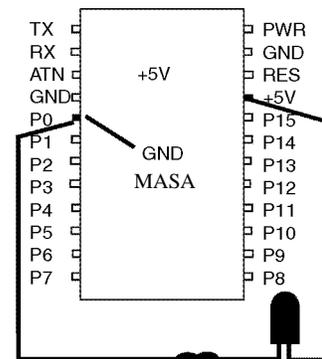


Figura 1.10: LED encendido

Cuando P0 está en "nivel bajo", y la corriente fluye, el LED está encendido.



Este es el propósito para el segundo comando: **"Out0=0"**. Este causará que P0 vaya a nivel bajo, lo que hace que el LED se encienda.

Tenga en cuenta que los microcontroladores ejecutan su programa muy rápidamente. En realidad, el BASIC Stamp ejecutará alrededor de 4000 instrucciones *por segundo*.

Experimento 1: "¿Qué es un Microcontrolador?"

Si apagáramos el LED en el siguiente comando, esto pasaría demasiado rápido para que pudiéramos verlo. Por lo tanto necesitamos "aletargar" el programa; de ésta forma podemos ver si está operando correctamente o no.

Es el propósito del siguiente comando: "**Pause 1000**". Este comando hace que el programa espere por 1000 milisegundos, o sea 1 segundo.

El siguiente comando es "**out0=1**". Este comando hace que P0 vaya a nivel alto y apague el LED, debido a que no hay flujo de corriente.

A continuación hacemos una pausa con "**pause 1000**" (otro segundo). El LED está aún apagado.



"Goto" (ir) es muy simple de entender. Durante el curso de la ejecución del programa, cuando el comando "goto" es encontrado, el programa "va" a algún punto específico en el programa. En nuestro ejemplo, le decimos al programa "ir a titilar". Donde sea que esté "titilar", es donde el programa irá.

En nuestro programa, la etiqueta "titilar" está en la segunda línea. Por lo tanto, cuando la instrucción "**goto titilar**" es encontrada, el programa salta hacia la segunda línea y lo repite nuevamente. (El programa regresa a la segunda línea cada vez que encuentra el comando "**goto titilar**". Esto es lo que causa que el LED continuamente parpadee.

Un buen hábito al que conviene acostumbrarse, es "**remark**" (comentar) sus programas. Comentar o documentar sus programas, los hace más fáciles de seguir o de depurar si hay algún problema.

El apóstrofe (') es usado para decirle al microcontrolador que ignore la siguiente información, es sólo para beneficio humano. En otras palabras, cualquier cosa que esté escrita en una línea del programa después de un apóstrofe, no es parte del código de la instrucción.

Así, nuestro programa puede ser "comentado":

output 0	' hace PO una salida
titilar:	' acá es donde el bucle comienza
out0 = 0	' enciende el LED
pause 1000	' espera por 1 segundo, con el LED encendido
out0 = 1	' ahora apaga el LED
pause 1000	' deja el LED apagado por 1 segundo
goto titilar	' regresa y enciende el LED otra vez

El programa aún operará exactamente de la misma manera, los comentarios después de los apóstrofes, están solamente para nuestro beneficio, para entender qué hemos creado.

Observe que a través de éste experimento, hemos usado el comando "PAUSE" para esperar por "x" milisegundos. Tenga en cuenta que las instrucciones también requieren de un tiempo de ejecución. Por ejemplo, el tiempo para la ejecución de los comandos **LOW**, **HIGH**, y **PAUSE** es de alrededor de 0.15 milisegundos cada uno. El BASIC Stamp ejecuta en promedio 4,000 instrucciones por segundo.

INFORMACIÓN:

Un modo más simple

Recuerde que cada uno de los pines del Stamp (P0-P15) puede ser configurado como entrada o salida. Para hacer el pin una salida, use el comando output. Una vez que el pin es una salida, nosotros podemos hacerlo ir a nivel bajo (un nivel lógico 0), o alto (un nivel lógico 1), con la instrucción `OUT0=0` (para abajo), o `OUT0=1` (para alto). Usando estos comandos, toma dos líneas en nuestro programa para hacer el PIN una salida y luego hacerlo ir a alto o bajo.

PBASIC tiene una forma más simple de hacer esto. Si usted desea hacer P0 una *salida y en nivel alto* (al mismo tiempo), simplemente use el comando: **HIGH 0**; y recíprocamente, para hacer P0 una *salida y nivel bajo* (al mismo tiempo) use: **LOW 0**.

Nuestro programa de ejemplo ahora podría verse de ésta manera:

```
titilar:  
  low 0  
  pause 1000  
  high 0  
  pause 1000  
  goto titilar
```

El programa funciona exactamente igual, sólo que los comandos nuevos no sólo hacen que el pin vaya a nivel alto o bajo (como "OUT0=0" y "OUT 1=1") sino que ellos también hacen que el pin se convierta en una salida. En casos simples como éste programa, cualquier método bastará, pero en programación más complicada, un método puede ser más apropiado que otro. Exploraremos esto en una lección futura.

Experimento 1: "¿Qué es un Microcontrolador?"



Preguntas

1. ¿En qué se diferencia un microcontrolador de una computadora?
2. ¿Cuál es la diferencia entre hardware y software?
3. ¿Por qué un microcontrolador es como nuestro cerebro?
4. ¿Qué significa "debug" (depurar)?
5. El siguiente programa debería encender el LED en P0 por 2 segundos, luego apagarlo por 2 segundos, y luego repetir. ¿Cuántos errores (bugs) hay en el programa y qué correcciones son necesarias?

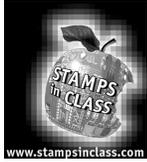
```
output 0
titilo:
out0 = 0
pause 200
out1 = 1
pause 2000
goto titilar
```

**¡Desafío!**

Reescriba el programa de la pregunta 5 anterior, para hacer lo siguiente. Cada programa debería ser cargado en el BASIC Stamp y probado.

1. Haga que ambos LED se enciendan y apaguen parpadeando al mismo tiempo, Cuando termine de hacer el programa, cárguelo en la PC (como lo ha hecho antes), y pruébelo.
2. Haga encender y apagar los LED alternativamente; en otras palabras, mientras un LED está encendido el otro está apagado, y viceversa.
3. Encienda el primer LED por 2 segundos, luego apáguelo. Espere 5 segundos y encienda el segundo LED por 1 segundo y luego apáguelo. Espere 3 segundos y repita.
4. Encienda el primer LED por 1.5 segundos, luego apáguelo. Espere 2 segundos y luego encienda el segundo LED por 1.5 segundos, Luego apáguelo. Espere 2 segundos, luego encienda ambos LED por 0.5 segundos y apáguelos por 2 segundos. Repita ésta última acción de 0.5 segundos encendido y 2 segundos apagado.

Experimento 1: "¿Qué es un Microcontrolador?"



¿Qué aprendí?

Complete las siguientes oraciones con las palabras de la lista.

software

ejecutado

microcontroladores

adaptará

re-programados

secuencia

descargamos

Los _____ están a nuestro alrededor. Incluso cuando no se ven como una computadora. (¿Quién alguna vez se imaginó que un juguete tendría una computadora incluida en su interior?)

Los Microcontroladores consisten de hardware y _____. Creamos programas en una PC, una computadora que está diseñada para interactuar con humanos (con teclado, monitor, etc.) y luego _____ el programa en un microcontrolador, donde es realmente _____ ("run").

Un programa de microcontrolador es solo tan inteligente como el que lo programó. Al contrario que el cerebro humano, el programa del microcontrolador no se _____ por sí mismo, ni cambiará el orden de las instrucciones del programa. El microcontrolador ejecutará un conjunto de instrucciones en la misma _____ en la que fue creado.

Muchos microcontroladores son versátiles y fáciles de usar, porque pueden ser re-utilizados, _____, y pueden usarse en un sinnúmero de productos e innovaciones, desde robots a tostadores.



¿Por qué aprendí esto?

La gran versatilidad de los microcontroladores está en que pueden ser programados para controlar cualquier cosa que la mente humana pueda concebir. Aeromodelos, controladores de casas inteligentes o sistemas de colección de datos climáticos remotos operados por batería, son sólo ejemplos.

Los microcontroladores deben tener dos componentes trabajando juntos, para que el dispositivo funcione. El primer componente es el hardware (el circuito). Muchas personas pasan su vida diseñando hardware para microcontroladores para una infinidad de variedad de productos. El segundo componente es el software. Los programadores se especializan en escribir (código de control) para teléfonos celulares, juguetes o incluso equipamiento industrial.



¿Cómo puedo aplicarlo?

La ventaja de los microcontroladores (y algo que usted puede considerar como una carrera futura) es que el mundo de los dispositivos inteligentes se está expandiendo a una velocidad increíble y no muestran ningún signo de disminuir su velocidad. A medida que la tecnología avanza en todas las áreas de nuestras vidas, nos vemos rodeados por un creciente número de aparatos avanzados tecnológicamente. Usted puede ayudar a desarrollarlos y tal vez

inventar el próximo "gran producto", o simplemente divertirse construyendo cosas; ¡la tecnología es la misma, sólo que aplicada diferente!

Mire alrededor suyo y piense cómo podría usar un microcontrolador para crear una luz de seguridad para su bicicleta, las luces del auto, un proyecto de arte que use la luz para interactuar con los espectadores.

Piense con sus amigos en un desarrollo para comenzar a fabricar con perspectivas comerciales...

...¿Quién sabe?

Experimento 1: "¿Qué es un Microcontrolador?"



Experimento 2: Detectando el Mundo Exterior

Tomar decisiones. Nuestro cerebro lo hace todo el tiempo. Tomamos decisiones basándonos en lo que vemos, oímos, tocamos, etc. Como aprendimos en el experimento 1 (¿qué es un microcontrolador?), los microcontroladores actúan como nuestro cerebro. Ellos manipulan el mundo exterior basados en “entradas” (inputs).

El Experimento 2 se centrará en cómo podemos diseñar un sistema microcontrolado que pueda cambiar sus salidas (outputs), dependiendo de qué tipo de entradas (inputs) digitales detecta.

Los microcontroladores son dispositivos programables. Esto quiere decir que contienen una cierta lista de instrucciones (llamada programa o código), que dice qué hay que hacer bajo ciertas circunstancias.

El BASIC Stamp se programa en BASIC, un lenguaje de computadora, que es fácil de aprender y que aún tiene unas características muy poderosas. Exploreemos ahora cómo podemos hacer reaccionar al microcontrolador y controlar el “mundo exterior”.



Partes Requeridas

El experimento 2 necesita las siguientes partes:

- (1) Un cable de programación
- (2) Dos LED (diodos emisores de luz)
- (2) Dos pulsadores
- (1) Un BASIC Stamp II
- (1) Plaqueta de Educación
- (2) Dos resistores de 470 ohm, ¼ watt (amarillo, violeta, marrón)
- (2) Dos resistores de 10k ohm, ¼ watt (marrón, negro, naranja)
- (1) Una batería de 9 volts o adaptador
- (6) Seis cables conectores
- (1) Un programa editor de BASIC Stamp, tanto en versión DOS o Win95

¿Qué es Un...

Sensor:

Un sensor es un dispositivo de entrada usado para detectar o medir presencia física. Los ejemplos incluyen sensores que detectan luz, calor, temperatura, curvamiento y compuestos químicos (tales como monóxido de carbono).

Hay una infinita variedad de sensores que podemos conectar al BASIC Stamp. Este experimento incluye un pulsador (un tipo de sensor) y un LED (un dispositivo de salida).

Debido a que este experimento usa dos valores diferentes de resistores, y ambos se ven parecidos, ¿cómo los puedo separar?. Leyendo las bandas con el código de colores. Si usted no conoce cómo leer el “código”, mire el Apéndice C para averiguarlo.



¡Programélo!

Este circuito tiene un tipo de sensor (el pulsador) y un tipo de dispositivo de salida (el LED).

Una vez que usted tiene todos los componentes instalados en el área del prototipo, como se muestra en la figura, conecte el cable de programación desde la Plaqueta de Educación a su PC y conecte una fuente de alimentación.

Arranque el editor para Windows de BASIC Stamp II, haciendo click en el icono.

Como aprendimos en el Experimento 1, la pantalla, en el borde superior está en blanco, excepto por unas pocas palabras. Acá es donde escribiremos nuestro programa de control.

Además recuerde que el programa que escribiremos no funcionará en la PC, sino que será descargado (“**downloaded**”) o enviado al microcontrolador. Una vez que el programa ha sido recibido, el BASIC Stamp ejecutará las instrucciones exactamente como las hemos creado.

Escriba el siguiente programa:

```

output 0
out0=1
Input 2
revisar:
  if in2=0 then titilar
  goto revisar
titilar:
  low 0
  pause 200
  high 0
  pause 200
  goto revisar
    
```

Ahora, mientras tiene presionada la tecla “ALT” presione la letra “r” (por “run” que quiere decir correr).

Experimento 2: Detectando el Mundo Exterior

Si el programa no se descarga apropiadamente, y usted tiene un mensaje diciendo "error", puede haber un "bug" (error). Si usted obtiene un mensaje que dice "hardware not found" (no se encuentra el circuito) revise la conexión del cable entre la Plaqueta de Educación y la PC. El programa para Windows del BASIC Stamp II Windows lo ayudará a encontrar estos errores. Si encuentra un error se lo dirá mientras hace una descarga, pero usted también puede presionar ALT-M para encontrar errores de sintaxis.

Trate de descargarlo nuevamente (tenga presionada la tecla ALT y luego presione "r"). Si aún no funciona, debe tener un "bug" (error). Revise nuevamente su programa y asegúrese que ha escrito todo correctamente.

Si después de tratar esto, usted sigue teniendo problemas, pídale ayuda a su instructor.

Si su programa está trabajando apropiadamente, el LED debería titilar, mientras usted presiona el pulsador.

Ahora analicemos paso por paso nuestro programa:

INFORMACIÓN

Entrada:

Técnicamente, no necesitamos realmente darle este comando al BASIC Stamp debido a que al encenderlo (cuando usted aplica la alimentación por primera vez o presiona el botón reset) cada pin es automáticamente configurado como una entrada. Sin embargo como usted irá desarrollando circuitos cada vez más complicados, es posible que necesite cambiar el estado de los pines de entrada a salida o viceversa. Todo bajo el control del programa. Incluimos este comando para ayudar a clarificar la diferencia entre entrada y salida.

Nuestro primer comando, **output 0**, hace el pin P0 una salida ("output").

La instrucción **Out0=1**, fija el registro de salida de P0 a un valor igual a "1", o "alto" (high). Debido a que el LED está conectado a Vdd (alto) y el pin de P0 es también fijado a "alto", no hay flujo de corriente, por lo tanto el LED está apagado.

"**Input 2**" le dice al BASIC Stamp que P2 debe ser configurada como entrada.

Recuerde del Experimento 1, que un comando como "**revisar:**" realmente no es un comando, es una etiqueta (simplemente un marcador o puntero a cierto lugar de su programa). Cuando nuestro programa eventualmente encuentre el comando "**goto revisar**", el programa buscará la etiqueta "**revisar**", saltará hasta ella, y luego continuará ejecutando el programa desde ese punto.

El comando "**if in2=0 then titilar**" le dice al microcontrolador que revise el estado del pin llamado "P2". Cuando un microcontrolador revisa el estado de un pin en particular, lo que realmente está haciendo es leer el valor digital de ese pin.

El campo de la electrónica está generalmente dividido en dos diferentes dominios "del tipo de señal" –Digital y Analógico. Nuestra entrada del pulsador es tanto un "0" o un "1" (abierto o cerrado). Este es un sensor de tipo digital.

Medir el nivel de una pileta de natación o el volumen de un amplificador de audio, son ejemplos de entrada analógica. Sensores para éste tipo de aplicación, convierten una medición, por ejemplo de 10 cm, a un valor digital que el microcontrolador pueda entender. Exploraremos este mundo fascinante de la Conversión Analógico a Digital en un experimento futuro.

¿Qué es Un...

Analógico:

Un valor variable continuo. En lugar de 1 ó 0 (+5 volt), los valores analógicos pueden estar en cualquier valor intermedio de ambos extremos. Debido a que los microcontroladores solamente entienden entradas si son valores digitales, nuestros sensores y los circuitos de inter-comunicación, necesitan convertir valores analógicos (voltajes) a sus equivalentes digitales.

Binario:

El sistema de numeración usado por todos los microcontroladores (así como los sistemas de computadoras). Normalmente usamos el sistema decimal (de 0 a 9). Los sistemas electrónicos digitales sólo trabajan (en los sistemas más básicos) con dos dígitos: 0 y 1.

En electrónica digital (binaria), cualquier valor distinto de 0 y 1 es considerado inválido. Por lo tanto, cuando el microcontrolador lee el estado de P2, verá un valor de 0 o 1. Nosotros buscamos que P2 sea un "0". Si (cuando el microcontrolador lo revisa) es un "1", entonces este comando no hará nada, y el programa se continuará ejecutando en el siguiente comando (en este caso, "goto revisar" – lo que causa que el programa salte al principio y continuamente revise que P2 se convierta en un "0"). Una vez que P2 se convierta en "0" entonces las condiciones para este comando se cumplen y el programa salta a "titilar" (una etiqueta que define la ubicación de otra "rutina").

La rutina "titilar:" le debería resultar conocida. Es simplemente un pequeño programa que hace que el LED se encienda (por 0.2 segundos) y luego se apague (por 0.2 segundos). Luego de hacer parpadear el LED, el comando "goto revisar" hace que el programa regrese y revise el estado de P2 nuevamente y "repita todo nuevamente".

Nuestro programa, con comentarios, ahora se ve así:

```

output 0           'hace P0 una salida
out0=1            'hace P0 "alta"
Input 2
revisar:          'una etiqueta
if in2=0 then titilar 'se fija si P2 es "0", y si lo es hace parpadear el LED
goto revisar      'si P2 era "1", entonces regresa y se fija nuevamente

titilar:          'una etiqueta
low 0             'enciende el LED
pause 200         'espera .2 segundos
high 0           'apaga el LED
pause 200         'espera .2 segundos
goto revisar      'regresa al principio y hace todo nuevamente
    
```

Escribamos ahora un programa nuevo. Escriba lo siguiente en el Editor del BASIC Stamp :

```

output 0
output 1
    
```

Experimento 2: Detectando el Mundo Exterior

```
Input 2
Input 3
out0=1
out1=1
revisar:
if in2=0 then titilar
if in3=0 then doble_titilar
goto revisar
titilar:
low 0
pause 200
high 0
pause 200
goto revisar

doble_titilar:
low 0
low 1
pause 200
high 0
high 1
pause 200
goto revisar
```

Antes de hacer funcionar este programa, ¿puede usted decir qué va a hacer?

El programa “tomará una decisión” basado en qué botón es presionado. Una vez que cualquier botón es presionado, el programa saltará a la rutina apropiada. El microcontrolador está sensando una entrada, tomando una decisión, y luego creando una salida. La decisión (si debe encender un LED o dos), es tomada basándose en qué botón es presionado.

El comando “**if in2=0 then titilar**”, se fija directamente en el pin y luego toma una decisión basada en el estado de ese pin. El comando “**if in3=0 then doble_titilar**” revisa el otro botón. ¿Qué pasa si ambos botones son presionados?. ¿Por qué?

Puede haber veces que necesitemos obtener el estado de una entrada (alto o bajo) y luego continuar ejecutando el programa sin tomar una decisión en ese momento. Tal vez varias entradas diferentes necesiten estar en un cierto estado antes de que una acción ocurra.

Si queremos que un LED titile solamente cuando ambos botones son presionados, nuestro actual programa no trabajará de la forma que nosotros queremos.

Una **variable** nos permite almacenar cierta pieza de información (ingresada ahora), para un análisis posterior.

¿Qué es Una

Variable:

Una variable es un símbolo que contiene un cierto valor. Ese valor puede ser cambiado bajo el control del programa y por lo tanto, el valor de las variables puede cambiar, pero su nombre no. Las variables pueden guardar ciertas piezas de información ahora, para ser usadas posteriormente en el programa.

Las variables deben ser “declaradas” antes de ser usadas en un programa. Declarar una variable simplemente es una instrucción en su programa que le avisa al microcontrolador el nombre de la variable y qué tan grande es.

Modifiquemos nuestro programa para demostrar una forma de cómo puede ser usada una variable. Escriba el siguiente programa:

```
x var bit
y var bit
```

```
output 0
output 1
```

```
out0=1
out1=1
Input 2
Input 3
```

```
obtener_estado:
x=in2
y=in3
```

```
if x+y=0 then doble_titilar
goto obtener_estado
```

```
doble_titilar:
low 0
low 1
pause 200
high 0
```

Experimento 2: Detectando el Mundo Exterior

```
high 1
pause 200
goto obtener_estado
```

Las primeras líneas, "**x var bit**", le dicen al microcontrolador que vamos a usar una variable llamada x y que tendrá el tamaño de un bit. Debido a que solamente estamos interesados en cuál es el estado del pin ("0" o "1"), un simple pin es todo lo que necesitamos como variable. En la segunda línea ocurre lo mismo para "y".

En la rutina "**obtener_estado**:" ocurren dos cosas. En "**x=in2**" el microcontrolador se fijará en la entrada P2, y luego (a diferencia de nuestro programa anterior), no saltará a otro lugar del programa, sino que guardará el valor (nuevamente "0" o "1") en nuestra variable llamada "x". El programa hará lo mismo para "y" en la siguiente línea.

Ahora, en éste punto de la ejecución del programa (después que los pines P2 y P3 fueron leídos), no importa qué pasa en las entradas P2 y P3, los valores que tenían cuando fueron leídos, están guardados en las variables x e y.

Ahora podemos realizar algunas "operaciones" con éstas variables. Por ejemplo, en nuestro programa sumamos el valor de "x" más el valor de "y". Si la respuesta es cero, entonces nuestro programa saltará a la etiqueta llamada "**doble_titilar**". Por lo tanto, ambos botones deben ser presionados para que la rutina `doble_titilar` sea llamada.

Las variables no tienen por qué ser simples caracteres. En nuestro programa de ejemplo simplemente usamos las letras "x" e "y".

Pruebe esto: use diferentes nombres para x e y, por ejemplo "Pedro" & "Juan". Cuando usted crea programas más complicados, darle un nombre a la variable que signifique algo para usted puede ayudarle a entender cómo su programa está operando (¡especialmente si usted necesita depurarlo!).

INFORMACIÓN

Más sobre variables

En PBASIC, los nombres de las variables pueden tener una longitud de hasta 32 caracteres. La longitud del nombre no tiene ninguna influencia en la velocidad de ejecución del programa. Por ejemplo, la instrucción: `x = in6`, tendrá la misma velocidad de ejecución que: `éste_es_un_nombre_muy_largo = in6`.

Las variables pueden ser declaradas en 4 tamaños diferentes: **bit** (1 bit), **nib** (nibble - 4 bits), **byte** (8 bits), & **word** (16 bits)

Usted debería siempre declarar sus variables en el tamaño más pequeño que sea apropiado para los datos que contendrá. El Stamp2 tiene un límite de 208 bits para guardar variables. Estos están agrupados en 13 palabras (de 16 bits cada una). Estos bits pueden ser usados en cualquier combinación de tamaños anteriores. Por ejemplo, su programa puede tener 10 variables word (160 bits), 10 variables nibble (40 bits), y 8 variables bit (8 bits), o cualquier otra combinación mientras que el valor total no exceda los 208 bits.

Vea el Manual del BASIC Stamp Manual Version 1.9 para información adicional de cómo usar las variables eficientemente.



Preguntas

1. ¿Cómo toma una decisión un microcontrolador?

2. ¿Qué es un sensor y por qué un microcontrolador necesita uno? Mencione algunos tipos diferentes de sensores.

3. Defina una variable y describa cómo puede ser usada en un programa.

4. Escriba el código para declarar una variable llamada "status". La variable podría valer tanto "0" como "1".

5. Agregue comentarios apropiados al siguiente programa:

```

output 0
output 1
out0=1
out1=1
revisar:
  if in2=0 then titilar
  if in3=0 then doble_titilar
  goto revisar
titilar:
  low 0
  pause 200
  high 0
  pause 200
  goto revisar
doble_titilar:
  low 0
  low 1
  pause 200
  high 0
    
```

Experimento 2: Detectando el Mundo Exterior

high 1
pause 200
goto revisar



¡Desafío!

1. Escriba un programa (completo, con comentarios) que haga titilar al LED P0 (cada $\frac{1}{2}$ segundo), mientras esté presionado el pulsador P2. Cuando el botón no está presionado, el LED P1 está encendido, pero se apaga cuando el LED P0 está titilando.
2. Escriba un programa que haga titilar ambos LED (cada 1.2 segundos) cuando cualquier pulsador es presionado. Si ningún pulsador es presionado, los LED están encendidos y si ambos pulsadores son presionados, ambos LED están apagados.
3. Escriba un programa que haga titilar alternadamente los LED cada $\frac{1}{2}$ segundo, pero solamente después de que P2 haya sido presionado y liberado, y luego de que P3 sea presionado. Luego, escriba comentarios en su programa mostrando que cambios haría para invertir el orden en que presionaría los interruptores.
4. Escriba un programa que haga titilar los LED (cada .2 segundos) cada vez que el pulsador P2 es presionado. Luego, mientras el pulsador P2 está presionado, el LED P1 es apagado cuando el pulsador P3 es presionado (pero el LED P0 aún sigue titilando).



¿Qué aprendí?

Complete las siguientes oraciones con las palabras de la lista.

I/O pin
 declaradas
 entradas (o sensores)
 if in1=0
 programa
 activados
 debug (depurar)
 variables

Microcontroladores necesitan _____ para saber que está pasando en el "mundo exterior". Usando comandos PBASIC tales como " _____", nuestro programa puede determinar que tipo de respuesta es la apropiada.

Hay una variedad infinita de sensores que pueden ser conectados al BASIC Stamp. Aunque los interruptores que usamos en este experimento fueron _____ presionándolos, podrían fácilmente haber sido los interruptores de las puertas de un ascensor –aquellos que evitan que seamos aplastados al cerrarse las puertas.

_____ son usadas para retener la información, permitiéndole al programa obtener los datos ahora (tal vez de varias entradas diferentes), y más tarde tomar decisiones en el momento apropiado.

Las variables pueden ser _____, o inicializadas ("set up"), en 4 tamaños diferentes. Si queremos controlar el estado (alto o bajo) de un _____ en particular, entonces fijamos la variable como un simple "bit".

Las variables pueden tener hasta 32 caracteres de longitud. Es importante recordar cuando usamos variables, darle un nombre que tenga relación con el dato que almacena. La longitud del nombre no tiene influencia en que tan rápido su _____ se ejecuta, pero un nombre "muy descriptivo" hace mucho más fácil de _____ su programa.

Experimento 2: Detectando el Mundo Exterior



¿Por qué aprendí esto?

La verdadera importancia de los microcontroladores es su habilidad para tomar decisiones basándose en las entradas. Las entradas a los microcontroladores, deben tener un formato digital, pero muchos tipos de situaciones del “mundo exterior” son analógicas por naturaleza.

La tecnología de los sensores es una de las áreas más desafiantes de la electrónica. Hay cientos de tipos diferentes de sensores en el Transbordador Espacial y en los satélites que él pone en órbita. Mucha gente se especializa en el diseño de sensores que se comunican con microcontroladores. Si a usted le gusta más trabajar con “hardware”, en lugar de escribir programas (crear software), éste podría ser un campo muy excitante y aún desafiante.

Cualquier sistema microcontrolado (o para el caso, computado), depende de las entradas digitales para tomar decisiones correctas. Es importante recordar que las decisiones del microcontrolador son solamente tan buenas como el programa que están ejecutando, y de la calidad de los sensores de entrada.

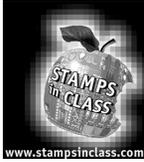
Cuanto más mire a su alrededor, más aplicaciones verá para la tecnología de sensores y microcontroladores.



¿Cómo puedo aplicarlo?

Muchas tiendas tienen una especie de “timbre” en la puerta, que suena cuando uno la atraviesa. Cada vez que el timbre suena, el dueño mira y se fija quién entró.

Usando un sensor de proximidad, que detecta la presencia de un objeto, (similar a un botón que está siendo presionado), usted podría detectar cuándo alguien atraviesa una puerta. Usando tres sensores, usted podría determinar en qué dirección la están atravesando. Entonces, usando dos tonos, podríamos saber con uno, si alguien está entrando, y con el otro, si alguien está saliendo.



Experimento 3: Movimiento Micro-controlado

Bien, ya estamos en el Experimento 3, y todo lo que hemos hecho ha sido hacer titilar unos cuantos LED. Ahora llegó el momento, ¡algo está por *moverse*!

3

Como usted sabe, un LED es un dispositivo de "salida". Un microcontrolador puede hacer titilar LEDs, así como puede controlar muchos otros tipos de dispositivos de salida (*algunas veces móviles*) bajo un programa de control.

Aunque los métodos de control son muy similares, otros tipos de dispositivos, tales como motores, pueden darnos un ejemplo más tangible de las manipulaciones del "mundo exterior".

¿Qué es UN

Circuito de interface:

Los microcontroladores operan con niveles de señal y voltaje muy pequeños. No tienen suficiente capacidad de manejo para operar dispositivos de salida grandes, o para grandes cargas.

Considere su "Walkman" como un microcontrolador. Por sí mismo puede controlar una pequeña salida (como los auriculares), pero para controlar un dispositivo más grande (como grandes bafles), usted necesitará un circuito de interface (un amplificador).

El BASIC Stamp puede controlar pequeños motores en un robot, o con el adecuado circuito de interface, puede operar los motores que abren las compuertas de una represa hidroeléctrica; todo depende de su circuito de interface.

Hay muchos tipos de motores diferentes que el BASIC Stamp puede controlar. La mayoría de los motores, sin embargo, requieren algún tipo de "circuitería de interface" externa que le permite a nuestro microcontrolador, controlarlos. En este experimento usaremos un tipo especializado de motor DC (CC, corriente continua). Es llamado "servo".

¿Qué es un servo? UN servo es un motor de corriente continua que tiene un "circuito de control" construido en su interior. Esto lo hace extremadamente simple de conectar a un microcontrolador (tal como el BASIC Stamp). El tipo de servo que usaremos fue diseñado originalmente para ser usados en aviones, botes y autos radiocontrolados.

En lugar de rotar continuamente, como los motores comunes, un servo es "posicionable". Usted puede, enviando las señales apropiadas desde el BASIC Stamp, hacer rotar al servo a un punto específico, y quedarse ahí.

Los servos tienen muchas aplicaciones, como estamos por ver.

Experimento 3: Movimiento Micro-controlado



Partes Requeridas

El Experimento 3 requiere de las siguientes partes:

- (1) Un BASIC Stamp II
- (1) Una Plaqueta de Educación
- (1) Un conector de tres pines
- (1) Un cable de programación
- (1) Un servo RC
- (1) Un LED (diodo emisor de luz)
- (1) Un resistor de 470 ohm, ¼ watt
- (1) Un capacitor electrolítico de 3000 microfaradios
- (1) Una batería de 9 volts o fuente de alimentación
- (6) Seis cables conectores
- (1) Un programa Editor de BASIC Stamp tanto en versión DOS o Win 95



¡Constrúyalo!

Una imagen de un servo típico es mostrada en la Figura 3.1. Los servos vienen en muchas formas y tamaños, dependiendo de su aplicación.

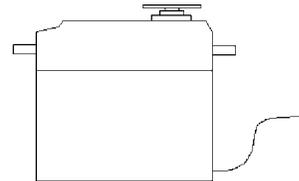
Usando la Plaqueta de Educación, cree el circuito de hardware como e muestra en las Figuras 3.2 y 3.3

¿Qué es Un...

Vdd & Vss:

Estas son las designaciones que son usadas para *voltaje positivo y masa*. En nuestro circuito (Plaqueta de Educación), Vdd es igual a +5 volts, y Vss es igual a cero volts. Este es un juego de valores bastante común para la mayoría de los sistemas de computación, sin embargo, estos valores pueden variar, dependiendo de qué tipos de dispositivos electrónicos pueden estar en el circuito.

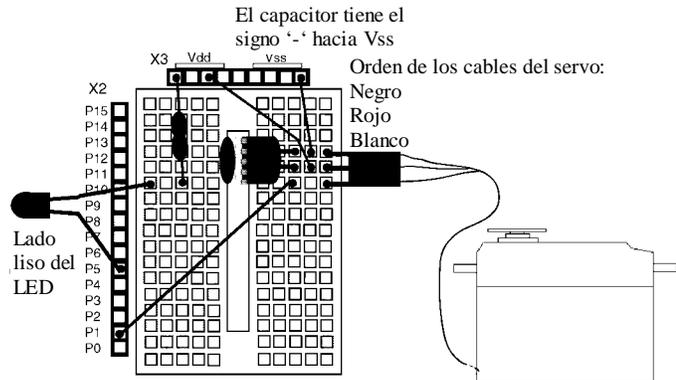
Figura 3.1: Servo
Servo de radio control (RC)



La Figura 3.2 muestra cómo se muestra físicamente el circuito y la Figura 3.3 es la representación esquemática.

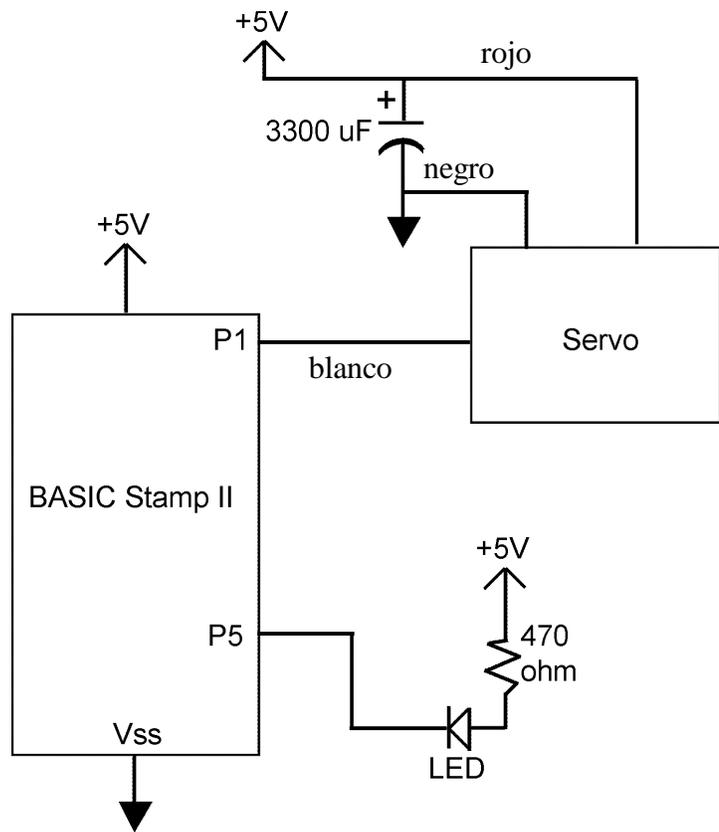
El código de color de los cables del servo que usted tiene, pueden variar, dependiendo del modelo. En todos los casos (con los servos que usted obtiene de Parallax), el cable negro es conectado a **Vss** y el cable rojo es conectado a **Vdd**. El cable restante (el tercero) puede ser blanco o amarillo (o algún otro color). Este es el cable de control que será conectado a la señal de P1 en el BASIC Stamp.

Figura 3.2: Dibujo de la conexión del servo:
Observe el orden en que se conectan los cables del servo en la protoboard.



¿Qué es un Capacitor:
Un capacitor almacena energía eléctrica. Es usado en nuestro circuito como una pequeña batería para entregar los picos de corriente (medidos en Amperes), requeridos cuando el motor del servo empieza a girar. El capacitor ayuda a entregar esta energía de "arranque", permitiéndole al circuito funcionar "suavemente", minimizando los picos, que pueden hacer funcionar erráticamente al microcontrolador

Figure 3.3: Esquema de conexión del servo: Observe el orden en que los cables del servo se conectan en la protoboard.



Experimento 3: Movimiento Micro-controlado

Asegúrese de que hay un resistor de 470 ohm en serie con el LED. Como hemos aprendido en un experimento anterior, esto limitará la corriente que fluye a través del LED a una cantidad segura. Un flujo demasiado grande de corriente por el LED lo quemará y puede también dañar el BASIC Stamp .

El **capacitor** (el cilindro con dos cables) tiene también una polaridad. Es importante que usted conecte el pin negativo (-) del capacitor al Vss y el positivo (+) a Vdd. Invertir ésta conexión puede dañar al capacitor. Ver la Figura 3.2.

Este circuito tiene dos tipos de dispositivos de salida (un LED y el Servo).

Una vez que usted tenga todos los componentes instalados en el área de prototipo, (como se muestra en las figuras), conecte el cable de programación de la Plaqueta de Educación a la PC y conecte una batería de 9 volts o una fuente de alimentación de corriente continua (CC) de 9 volts a la Plaqueta. Debido a que el servo requiere mucha corriente (mucho más que un LED), la vida de la batería será bastante limitada, así que use una fuente de alimentación, si usted tiene una.



¡Prográmelo!

Encienda su PC, y haga doble click en el icono del BASIC Stamp.

Usted debería estar ahora ejecutando un programa llamado el "BASIC Stamp Editor". Este es un programa que fue creado para ayudarle a escribir y descargar programas al microcontrolador.

Escriba el siguiente programa:

```
output 5
inicio:
out5=1
pause 200
out5=0
pause 200
goto inicio
```

Ahora, mientras tiene presionada la tecla "ALT", presione la tecla "r" (por "run", que significa ejecutar) y presione "enter".

Si su programa está trabajando apropiadamente, el LED debería estar titilando.

Pero ya hemos hecho esto antes. Es simplemente un programa que hace titilar un LED, ¿por qué estamos haciendo esto otra vez?

¿Problemas?

Si usted está usando el editor de DOS BASIC Stamp y obtiene un mensaje que dice, "Hardware not found", revise nuevamente las conexiones entre la PC y la Plaqueta y también asegúrese que la batería de 9 volts (o la fuente de alimentación) está conectada y cargada.

Trate de descargar nuevamente (tenga presionada la tecla ALT y luego presione "r"). Si aún no funciona, debe tener un bug (error). Revise nuevamente su programa para estar seguro de que lo ha escrito correctamente.

Después de revisar sus conexiones, presione ALT "r" otra vez. Si aún recibe el mensaje "hardware not found" asegúrese de que su computadora está funcionando en DOS, y no en Win95. Si está funcionando en Win 95, vaya al menú de inicio (en el monitor), y seleccione reiniciar en modo MS/DOS.

Si después de revisar esto tiene problemas, pídale ayuda a su instructor.

La respuesta es que estamos por usar un comando de PBASIC más sofisticado y ésta simple rutina nos ayudará a entender cómo trabaja el comando nuevo.

Trate de cambiar las instrucciones "pause" a valores de solamente 100 (en lugar de 200).

Ahora cambie las pausas a 50. Ahora a 30. Ahora a 20. Ahora a 5. ¿Qué está pasando?.

El LED está titilando cada vez más rápido, debido a que el tiempo de cada pausa se está volviendo más corto, cada vez que usted disminuye los valores (la Pausa). Cuando usted llega a cierta velocidad de parpadeo, nuestros ojos ven el LED como si estuviera encendido todo el tiempo. En realidad no lo está, está parpadeando a una velocidad tan alta, que nuestros ojos no pueden ver los *pulsos individuales de luz*.

Está bien, ¿y?. Bien, un servo es controlado por un chorro de pulsos que tienen una longitud de entre 1 y 2 **milisegundos**. Este pulso se repite aproximadamente cada 10 milisegundos.

Recuerde que el comando "pause" es fijado en milisegundos, y que la pausa más chica que podemos hacer, es de 1 milisegundo. El siguiente valor disponible es de 2 milisegundos (ms).

Un servo, necesita tener el chorro de pulsos (en el cable de "control" blanco o amarillo), que varíen su longitud entre 1 y 2 milisegundos. Con un chorro de pulsos que tienen una longitud constante de 1 milisegundo, el servo se posicionará en un extremo de su rotación. A medida que el ancho del pulso se incrementa, (1,1ms, 1,2ms, 1,3ms... etc.), el servo cambia su posición. Cuando el ancho del pulso alcanza los 2,0 ms, el servo está en el otro extremo de su rotación. Estos pulsos ocurren a intervalos de aproximadamente 10 ms. La Figura 3.4 es un **diagrama de tiempo** de los pulsos que el servo necesita.

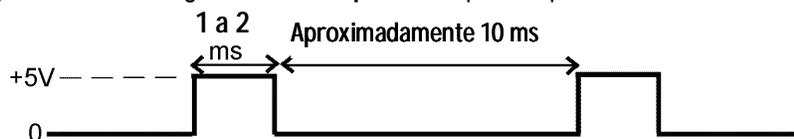


Figura 3.4: Secuencia de pulsos de un servo típico. Diagrama de tiempo de los pulsos que necesita el servo.

Experimento 3: Movimiento Micro-controlado

¿Qué es Un...

Milisegundo:

Los sistemas de microcontroladores y computadoras, operan a velocidades muy altas. Como humanos, estamos acostumbrados a usar para mediciones de tiempo, el rango de segundos, o en el caso de competencias atléticas, décimas o centésimas de segundos. Un milisegundo es 1 / 1000 de segundo, es decir hay 1000 milisegundos en 1 segundo. Esto se ve como una pequeña cantidad de tiempo, pero es realmente bastante largo en el mundo de la micro-electrónica de las computadoras. De hecho, su computadora personal (la que está usando para escribir éstos programas en PBASIC), está probablemente operando en *millonésimas* de segundos.

Diagrama de Tiempo:

Las computadoras funcionan con una serie de pulsos, normalmente entre 0 y 5 volts. Un diagrama de tiempo es simplemente una forma de visualizar los pulsos. Usted "lee" un diagrama de tiempo de izquierda a derecha. En nuestro diagrama de ejemplo, vemos que el voltaje, (en nuestro pin de salida P1) comienza a 0 volts. Después de un corto periodo de tiempo vemos que P1 pasa a estado alto por una duración de 1 a 2 milisegundos y luego regresa a 0 volt. Después de aproximadamente 10 milisegundos, P1 vuelve a subir. Salvo que se indique en el diagrama, usted puede asumir que el proceso se repite indefinidamente, es decir que cuando llega al extremo derecho del diagrama, regresa al extremo izquierdo, y comienza otra vez.

De acuerdo, armados con esta información, escribamos un programa que hará mover el servo a un extremo, quedarse ahí por un corto período de tiempo y luego moverse hacia el otro extremo, quedarse ahí por un corto tiempo, y luego repetir.

Escriba el siguiente programa:

```
x var word
output 1
inicio:
for x = 1 to 100
  pulsout 1,500
  pause 10
next
pause 500
for x = 1 to 100
  pulsout 1,1000
  pause 10
next
pause 500
goto inicio
```

Ahora, mientras tiene presionada la tecla "ALT", presione la letra "r", (por "run").

Si su programa trabaja bien, el servo debería estar rotando repetidamente de un extremo a otro.

Los servos no están diseñados para realizar giros completos (como un motor común que usted puede usar para mover las ruedas de un robot). Por ejemplo se usan servos para abrir y cerrar válvulas o en un brazo robótico.

INFORMACIÓN

Modificaciones de Servo:

Aunque no están diseñados específicamente para rotación completa, los servos pueden ser modificados para permitirles movimiento giratorio completo. Un método para ésta modificación está explicado en Programming & Customizing the BASIC Stamp", por Scott Edwards. Ver el Apéndice para más información.

Analicemos ahora el programa:

x var word

Recuerde que para que el BASIC Stamp sepa qué variables están siendo usadas, necesitamos "declararlas" en nuestro programa. Este comando le dice al BASIC Stamp que usaremos una variable llamada "x", y que tendrá el tamaño de una palabra "word". Una variable "word" de 16 bits puede tener un valor entre 0 y 65.536 en nuestro sistema numérico decimal. Debido a que usamos solamente "100" como valor máximo en nuestro programa, podríamos haber fijado esta variable como tipo *byte*, usando 8 bits y capaz de almacenar un valor entre 0 y 255 (decimal). La palabra bit viene de binary digit (dígito binario).

output 1

Como ya sabemos, hace que P1 sea una salida.

Inicio :

Es simplemente una etiqueta, que marca un lugar en el programa.

for x = 1 to 100

Para aquellos que han escrito programas en BASIC, este programa puede resultarles familiar. Es el comienzo de un bucle "FOR...NEXT". Simplemente dice que la primera vez que este comando es encontrado, la variable "x" será fijada al valor "1". El programa continúa ejecutando las siguientes líneas hasta que encuentra el comando llamado "next".

Una vez que encuentra el comando "next", el programa regresa al comando "for x = 1 to 100", e incrementa el valor de "x" en una unidad. El programa entonces repite el bucle una y otra vez (incrementando "x" cada vez), hasta que el valor de "x" = 100. Cuando "x" = 100 (es decir cuando esta parte del programa se ha repetido 100 veces) el programa saldrá del bucle y ejecutará la línea de comando inmediatamente posterior a "next".

Le enviamos un paquete de 100 pulsos al servo para darle tiempo suficiente a reaccionar mecánicamente a la señal. El microcontrolador puede operar mucho más rápido que cualquier dispositivo mecánico del mundo exterior, y al repetir 100 veces la instrucción, le damos al servo tiempo suficiente para "alcanzar" al BASIC Stamp.

Experimento 3: Movimiento Micro-controlado

pulsout 1,500

Este es un comando muy fácil de usar en el mundo de la entrada/salida. Muchas veces necesitamos obtener un pulso de salida muy estable generado por nuestro microcontrolador para controlar precisamente dispositivos de alta tecnología (tales como nuestro servo). Este comando funciona en forma similar a las técnicas que usamos para hacer parpadear un LED pero sería imposible de llevar a cabo porque nuestro servo requiere anchos de pulso entre 1 y 2 ms. "Pause" no puede proveer la resolución que nosotros necesitamos, solo puede saltar de 1 a 2 milisegundos. Esta es la razón por la que creamos el programa que hace titilar el LED. Ese programa nos llevó 4 o 5 líneas de código (con una resolución inadecuada) puede ser reemplazado con esta simple línea de comando y con una resolución que es medida en *microsegundos!*.

"Pulsout 1" (pulsar la salida 1) hace exactamente lo que su nombre indica. Crea un pulso *único* de salida en el pin de E/S P1. El número "500" es un valor que determina la duración del pulso. Como se mencionó antes, ésta duración es medida en microsegundos. Pulsout tiene una duración de 2 microsegundos, por lo tanto un valor de 500, nos dará un pulso con una longitud de *500 veces 2 microsegundos*, o 1000 microsegundos (que es igual a 1 milisegundo, el valor requerido por el servo). Un valor de 1000 creará un pulso con una longitud de 1000 x 2 microsegundos = 2 milisegundos, el ancho de pulso requerido por el servo para moverse al otro extremo.

pause 10

Ya conocemos lo que hace **pause**, pero la razón por la que pusimos esta pausa, puede no ser tan obvia. Las especificaciones para manejar al servo (por lo menos el servo que estamos usando en éste experimento), indican que el chorro de pulsos que entran al servo deben estar separados aproximadamente por 10 milisegundos. Poniendo una pausa de 10 milisegundos en éste punto controlamos el flujo de los pulsos para cumplir con las especificaciones del servo. Nuevamente vea el *diagrama de tiempo* en la Figura 3.4.

next

En este punto el programa regresa al comando "for x = 1 to 100" anterior, y envía el siguiente pulso, a menos que ya haya repetido 100 veces (en este ejemplo). Si "x" alcanza el valor 100 en éste punto, el programa continuará su ejecución en el siguiente comando.

pause 500

Este comando es ejecutado cuando el bucle For...Next anterior ha finalizado. Es simplemente una pausa, así que nosotros vemos que el servo se detiene antes de volver a empezar a girar.

for x = 1 to 100

Estamos comenzando otro bucle. Este bucle es idéntico al primero, con la excepción que la longitud del pulso de salida es de 2 milisegundos. Esto hace que el servo rote hacia su otro extremo.

pulsout 1,1000

Crea un único pulso con una duración de 2 milisegundos.

pause 10

Nuevamente, necesitamos esperar durante 10 milisegundos, antes de continuar nuestro "bucle".

next

Si "x" aún no alcanzó el valor 100, el programa regresará a la línea de comando "for x=1 to 100". Observe que regresará al comando "for" inmediatamente superior a la instrucción "next". (**No saltará hasta el primer bucle "for...next"**).

goto inicio

Regresa a la etiqueta inicio y repite todo otra vez.

¿Qué es un

Inicialización:
La primera parte de muchos programas es llamada a veces la "rutina de inicialización". Esto significa que esta porción del programa ajusta todos los parámetros que el programa usará.

Bien, recapitemos y veamos qué hace nuestro programa.

Después de la **inicialización**, el programa enviará un chorro de 100 pulsos, cada pulso con una longitud de 1 milisegundo. Esto hará que el servo rote a un extremo .

Luego, el BASIC Stamp enviará otra serie de 100 pulsos (nuevamente utilizando el bucle "for...next"), ésta vez sin embargo, el ancho de los pulsos es de 2 milisegundos. Esto hace que el servo rote a su extremo opuesto.

El programa vuelve al comienzo y repite todo otra vez.

Ahora, pruebe algo interesante. Debido a que la posición del servo es controlada por la longitud del pulso (generada por el comando pulsout), trate de cambiar el primer comando pulsout a:

pulsout 1,750

¿Qué pasó y por qué?.

Experimento 3: Movimiento Micro-controlado

Observe que para rotar el servo a una posición en particular, sólo debe cambiar el valor del ancho del pulso. Cambiando el primer valor a 750, obtenemos un ancho de pulso de 750 x 2 microsegundos, o 1,5 milisegundos. El servo girará aproximadamente al centro de su posición y al repetirse el programa se moverá entre el centro y un extremo.

Pruebe diferentes combinaciones de anchos de pulsos (en ambos extremos) para girar al servo a diferentes posiciones.

¿Entiende usted qué estamos haciendo aquí?. Su programa es capaz de mover (o en éste caso rotar) un dispositivo mecánico, *en el mundo real*. Si éste fuera un servo más grande, usted podría usarlo para mover un brazo de un robot industrial, o abrir la puerta de un supermercado ¡automáticamente!. Servos como éstos son también usados para controlar los ojos y las expresiones faciales de la mayoría de las "criaturas" construidas por expertos en efectos especiales para las películas.



Preguntas

3

1. ¿En qué se diferencia un servo de un motor?
2. ¿Qué comando usamos (en PBASIC) para controlar la rotación de un servo?
3. ¿Por qué no podemos usar el comando Pause para crear las longitudes de pulso necesarias para controlar un servo?
4. Describa cómo funciona el bucle "For...Next"
5. Agregue los *comentarios* apropiados al siguiente programa:

```
x var word _____  
output 1 _____  
inicio: _____  
for x = 1 to 100 _____  
pulsout 1,500 _____  
pause 10 _____  
next _____  
for x = 1 to 100 _____  
pulsout 1,1000 _____  
pause 10 _____  
next _____  
goto inicio _____
```

Experimento 3: Movimiento Micro-controlado



¡Desafío!

1. Escriba un programa (completo, con comentarios), que encienda un LED (en P5) cada vez que el servo alcanza un extremo de su rotación, y luego apague el LED cuando alcanza el otro extremo.
2. Escriba un programa (con comentarios), que haga girar el servo de un extremo a otro (continuamente), pero que se detenga un instante en la mitad de su rotación, cada vez que pasa por ahí.
3. Escriba un programa (con comentarios), que mueva al servo de un extremo al centro, regrese a ese extremo y luego gire directamente hasta el otro extremo y luego repite todo el ciclo.
4. Escriba un programa que haga titilar 3 veces al LED y luego gire el servo de un extremo al otro. Haga una pausa por un momento y luego repita. Esto se verá como un indicador que “advierde” que una pieza automática de maquinaria está por moverse.



¿Qué aprendí?

Complete las siguientes oraciones con las palabras de la lista.

3

- ciclos
- decimal
- motores
- circuito
- For...Next
- pulsout
- servos
- pulsos
- milisegundos
- componentes

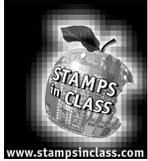
_____ son un tipo especial de _____ de CC que son convenientes para ser conectados directamente a un microcontrolador. Un servo es diseñado para reaccionar a una serie de _____ en su cable de control. A medida que el ancho de estos pulsos cambia de 1 a 2 _____, la circuitería interna del servo hace que el motor rote a la posición apropiada.

Usamos el comando _____ para enviar un ancho de pulso específico, para controlar la entrada de línea del servo. En nuestra aplicación variamos el ancho del pulso, entre 1 y 2 milisegundos, usando el comando: *Pulsout 1, X;* donde X es un valor _____ entre 500 y 1000. Debido a que el comando *pulsout* tiene una resolución de 2 microsegundos, esto nos da una salida de ancho de pulso de 1000 y 2000 microsegundos, respectivamente.

Los servos pueden ser grandes o pequeños, dependiendo de la aplicación. El _____ de control (que está construido dentro del servo) elimina la necesidad de conectar muchos otros _____ para el funcionamiento apropiado del circuito.

Un bucle _____ es un método conveniente para repetir cierta porción de nuestro programa un número predeterminado de _____. En nuestro programa de ejemplo repetimos 100 veces, pero este número pudo haber sido fácilmente cambiado, dependiendo de los requerimientos del programa y del hardware(circuito).

Experimento 3: Movimiento Micro-controlado



¿Por qué aprendí esto?

Para muchos, hacer titilar un LED con un microcontrolador no es un asunto muy importante, pero hacer mover un dispositivo mecánico o un motor *bajo el control del programa* es donde los microcontroladores se empiezan a volver interesantes.

Aunque el microcontrolador no sabe qué dispositivo de salida tiene conectado (LED o servo), hacer mover algo en el mundo real nos da un ejemplo más tangible de la *manipulación del mundo real*.

Hay microcontroladores (¡algunos de ellos BASIC Stamps!) a nuestro alrededor, controlando servos, motores de CC y CA, solenoides y otros tipos de dispositivos motrices. Por ejemplo las puertas automáticas del supermercado, los brazos robóticos usados por hobbyistas y profesionales.



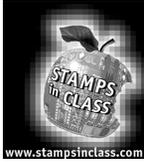
¿Cómo puedo aplicarlo?

Aunque normalmente es necesaria circuitería de interface adicional para la mayoría de los otros tipos de dispositivos motrices (para conectarlos al BASIC Stamp), los principios de este experimento usan esencialmente las mismas técnicas de control. Mucha gente pasa su vida diseñando sistemas basados en microcontroladores que

manipulan mecánicamente nuestro mundo. Aún si usted no piensa terminar haciendo éste tipo de trabajo como una carrera, usted tendrá una noción de cómo se abren las puertas del supermercado automáticamente.

Ahora que sabemos controlar un servo, puede desarrollar sistemas de control para un aeromodelo que podría ser similar a una función de "piloto automático" en una aeronave de tamaño real. Si usted agrega un altímetro digital como una "entrada" al BASIC Stamp, entonces la aeronave podría ser volada automáticamente.

De hecho, usted podría diseñar una especie de sistema de seguridad "anti colisiones" que podría permitirle a un novato volar el aeromodelo, pero cuando éste esté por chocar contra el suelo (¡y destruir el aeromodelo!). Su sistema de piloto automático podría "tomar el control" y ¡Prevenir la catástrofe!



Experimento 4: Automatización Simple

En el Experimento 3 usamos un servo (un tipo especializado de motor) para demostrar cómo un microcontrolador puede manipular un dispositivo mecánico en el “mundo real”. El programa que escribimos (y descargamos en la memoria del BASIC Stamp) controló la rotación y posición del servo.

4

¿Qué es una...

Automatización:
En este experimento el término automatización significa que algo está siendo hecho sin “intervención humana”. En nuestro ejemplo (la puerta automática del supermercado), esto no es exactamente cierto. Aunque no presionamos físicamente ningún botón, estamos, con nuestra presencia, “presionando el botón detector de luz”. Esto sin embargo, parece ser completamente automático, debido a que no tenemos que pensar en otra cosa que caminar hacia la puerta.
En un sentido estricto, automatización es la habilidad de los microcontroladores de hacer que las cosas sucedan sin interacción de nuestra parte

El programa hizo que el servo rotara una y otra vez de un extremo a otro. Este fue un ejemplo de cómo un microcontrolador puede hacer mover un dispositivo.

Sin embargo, en el Experimento 3 el BASIC Stamp estaba “ciego”. Todo lo que hizo el servo fue responder a nuestro código. Recuerde que la verdadera importancia de un microcontrolador es su habilidad para tomar decisiones basado en las entradas y entonces manipular el “mundo real” con su salida.

En este experimento haremos exactamente eso. Sí, vamos a mover el servo nuevamente, pero sólo si se reúnen las condiciones de entrada apropiadas. Usted puede pensar en éste experimento como un equivalente a escala de una puerta automática de un supermercado. La puerta está cerrada la mayor parte del tiempo, hasta que alguien o algo se acerca, entonces la puerta se abre automáticamente. Aparentemente no hay nada que tengamos que hacer para abrir la puerta; no presionamos ningún botón, sólo estar cerca de la puerta es suficiente para hacerla abrir. Esta es una forma muy básica de automatización.

Algunos de los sensores que son usados para éste tipo de aplicaciones son bastante sofisticados, y otros son bastante simples. Sin embargo, todos tienen una cosa en común, y es que cuando detectan una entrada entregan una señal al microcontrolador, quien de esa forma puede tomar una decisión, en éste caso “abrir la puerta”.

Como “sensor de detección” usaremos un dispositivo llamado “foto resistor”. Es un dispositivo diseñado para detectar niveles de luz diferentes. Es un tipo de sensor “óptico”.

Ahora ¡automaticemos! El Experimento 4 requiere las siguientes partes:

Experimento 4: Automatización Simple



Partes Requeridas

Para este experimento, usted necesitará lo siguiente:

- (1) Un BASIC Stamp II
- (1) Una "Plaqueta de Educación
- (1) Un conector de tres pines
- (1) Un Cable de Programación
- (1) Un servo R/C
- (1) Un LED
- (1) Un resistor de 470 ohm, ¼ watt
- (1) Un capacitor electrolítico de 3300 microfaradios
- (1) Un Fotoresistor, Cadmium Photocell (CdS)
- (1) Un resistor de 10K ohm, ¼ watt
- (1) Una batería de 9 volt o fuente de alimentación.
- (varios) cables de interconexión
- (1) Una P.C. con DOS 2.0 o superior, con un puerto serial disponible
- (1) Un programa BASIC Stamp Editor



¡Constrúyalo!

Usando la Plaqueta de Educación, cree el circuito de hardware, como se muestra en las figuras siguientes.

La Figura 4.1 es el esquema, y la Figura 4.2 es la imagen, (cómo se ve físicamente el circuito).

Dependiendo de qué modelo de servo usted tiene, el código de colores de los cables puede variar. En todos los casos, (con los servos que usted obtiene de Parallax), el cable negro es conectado a Vss y el cable rojo es conectado a Vdd. El cable restante (el tercero) puede ser blanco o amarillo (o algún otro color). Este es el cable de entrada de control, que será conectado a la señal de P1 en el BASIC Stamp.

Figura 4.1: Esquema
El experimento 4 también usa un fotoresistor.

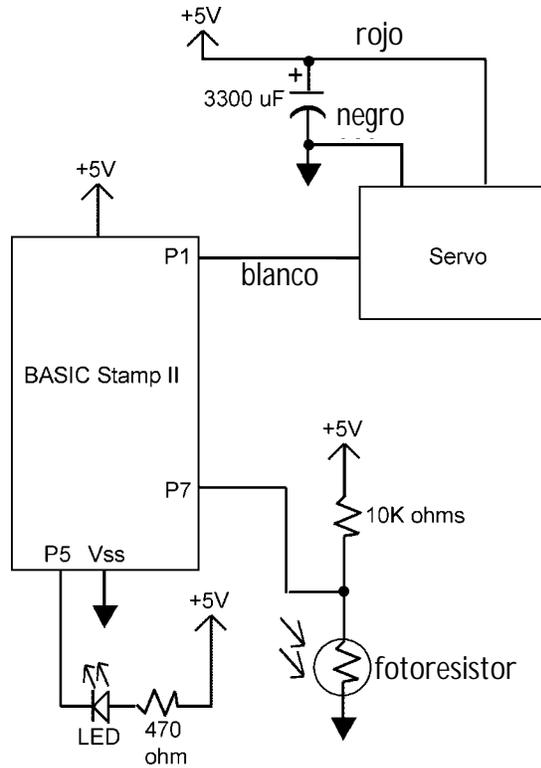
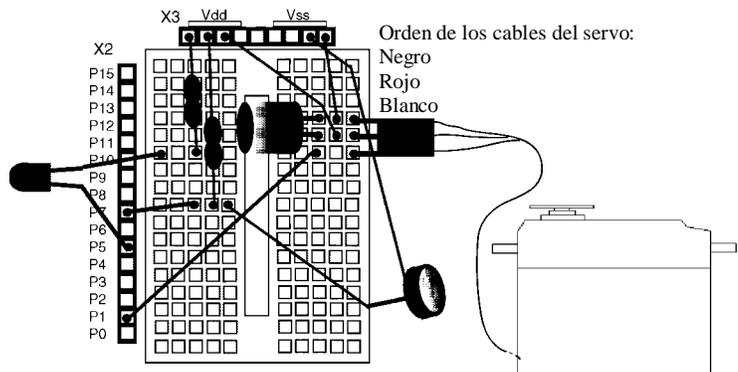


Figura 4.2: Imagen física.
Se muestra la pequeña modificación al hardware usado en el experimento 3.



Experimento 4: Automatización Simple

Este circuito tiene dos tipos de dispositivos de salida, (el servo y el LED) y un tipo de dispositivo de entrada, (el fotoreistor).

¡Recuerde conectar el LED correctamente!



¡Programémo!

Una vez que tiene todos los componentes instalados en el área de prototipo de la Plaqueta de Educación (como se muestra en las Figuras 4.1 y 4.2) conecte el cable de programación de la Plaqueta de Educación a la PC y conecte una batería de 9 volt o la fuente de alimentación a la plaqueta.

Debido a que el servo requiere mucha corriente (mucha más que un LED), la vida de la batería será bastante limitada, así que si tiene, use una fuente de alimentación.

Encienda su PC, y ejecute el editor del BASIC Stamp.

Usted debería estar ejecutando un programa llamado "Stamp Editor". Este es un programa que fue creado para ayudarle a escribir y descargar programas en el microcontrolador BASIC Stamp.

Escriba el siguiente programa:

```
este_lugar:  
high 5  
pause 200  
low 5  
pause 200  
goto este_lugar
```

Ahora, mientras tiene presionada la tecla "ALT" presione la tecla "r" (por "run") y presione "enter".

¿Otra vez un programa titilador?. Pues sí y no. Observe que no hay un comando "output 5" en el programa. Si usted ha estado leyendo los apéndices a través de éstas lecciones, usted descubrirá que los comandos "high" y "low" automáticamente hacen del pin una salida.

¿Qué es Un

Espacio de programa:

Los microcontroladores pueden tener varios tipos de memoria que son usadas para llevar a cabo su tarea. En el caso del BASIC Stamp 2, estamos limitados a 2048 bytes de almacenamiento de memoria (EEPROM). Esta cantidad de espacio es usada para almacenar datos y programas. Si usted escribe un programa que obtiene datos automáticamente en un periodo de tiempo (tal como una estación meteorológica remota), usted querrá hacer su programa tan pequeño y eficiente como sea posible para permitirle tanto lugar como pueda para almacenar datos.

EEPROM:

Esta es la sigla de 'electrically erasable, programmable, read only memory'(memoria sólo de lectura, eléctricamente borrable y programable). Aunque es un desarrollo sofisticado en la "industria de la memoria", es bastante simple de usar. Podemos grabar nuestros programas y datos en la EEPROM con comandos muy simples. Luego, cuando la alimentación es quitada, el programa y los datos son retenidos. ¿En qué se diferencia la EEPROM de los otros tipos de memoria de "estado sólido"? en que pueden ser muy fácilmente borradas ("automáticamente") y reescritas, una y otra vez.

Esto ahorra tiempo de escritura, y más importante, ahorra espacio de programa en el BASIC Stamp. Realmente no necesitamos preocuparnos por el espacio, en nuestros programas de experimentos(pequeños) pero a medida que usted haga programas más grandes y más complejos, es un buen hábito, agudizar sus habilidades de programación para obtener programas de "alta calidad". No solamente estaremos más lejos de sobrepasar el espacio del programa, sino que además el código se ejecutará más rápido dando como resultado un tiempo de ejecución más corto, (mayor velocidad de ejecución).

4

Modificar el programa para que se vea así:

```
n var bit
n=0

este_lugar:
n=0
low 5
debug ? n
pause 1000
high 5
n=1
debug ? n
pause 1000
goto este_lugar
```

Ejecute el programa.(Run)

No sólo debería estar titilando el LED sino que también debería haber un "recuadro de información" en el monitor de su PC, mostrando alternadamente "n=1" y "n=0".

Veamos cómo funciona...

n var bit
Una variable llamada "n", con el tamaño de sólo un bit .

este_lugar:
Una etiqueta en el programa.

n=0

Experimento 4: Automatización Simple

Algo nuevo. Vamos a fijar el valor de "n" a 0

low 5

Hace P5 bajo, por lo tanto enciende el LED.

¿debug ? n

¿Debug? Esta palabra suena familiar. Recuerde que "debug" (depurar) significa quitar los errores de su programa. Bien, el lenguaje PBASIC tiene un comando llamado "debug", que realmente puede ayudar a encontrar éstos errores de programa.

Normalmente nosotros enviamos el programa desde nuestra PC al BASIC Stamp a través del cable de programación. Es esencialmente un viaje en un solo sentido (hasta ahora).

Debug es un comando muy especializado que le permite al BASIC Stamp enviar información ("data") (datos), a través del cable y mostrarlo en el monitor de su PC. De ésta forma podemos "ver dentro del BASIC Stamp" y conocer los datos en los que el BASIC Stamp está trabajando. En éste caso, fijamos el valor de "n" en 0 en un comando previo. Cuando el comando debug es encontrado, imprime el valor de "n" en la ventana debug de la PC.

El "?" es una abreviación de "imprimir", así que el comando literalmente dice: "Abrir una ventana de depuración 'debug' en la PC e imprimir el valor de 'n' en la pantalla".

pause 1000

Fácil por ahora, ¿no?

¿Qué es Un

Debug (comando):
Una herramienta muy útil para "ver" qué está haciendo su programa dentro del Stamp.

El comando Debug tiene una gran flexibilidad propia. Revise el apéndice para una descripción completa de cada una de las características disponibles.

high 5

Apaga el LED.

n=1

Acá es donde cambiamos el valor de "n" a 1.

¿debug ? n

Enviamos el valor de "n" a la PC. Debido a que el valor de "n" ha cambiado, debug "imprime" el cambio en la pantalla de su PC.

pause 1000

Sí. Ya sabemos lo que hace.

goto este_lugar

Repite todo nuevamente.

Ahora, algunos de ustedes pueden estar pensando ¿Para qué necesitamos ver el valor de "n" en nuestra pantalla debug? Debido a que nuestro programa (el que escribimos) fija el valor de "n", ya sabemos cuál será el valor de "n". ¿Para qué queremos que lo muestre?

4

¿Qué es una

Constante:

Un valor o parámetro que es fijado *absolutamente* a un valor específico. Hay otras formas de fijar las constantes a un valor predeterminado, (tales como usando el comando **con**), esto lo hace más fácil de cambiar en el futuro. En muchos casos, las variables usadas en un programa son *dinámicas*, debido a que pueden cambiar constantemente su valor durante la ejecución del programa.

Es cierto, el valor de "n", en éste ejemplo es "**constante**", el programa fija el valor sin permitirle ser modificado. Sin embargo, cuando una variable es fijada por eventos externos, por ejemplo cuando una entrada cambia debido a un pulsador que está siendo presionado, entonces debug nos permite ver el cambio en el dato de entrada, y entonces observar si nuestro programa reacciona o no correctamente.

Intentémoslo. Modifique el programa:

n var bit

este_lugar:

n=in7

debug ? n

pause 100

goto este_lugar

Ahora, al valor de "n" en lugar de ser constante, le permitimos que sea igual al valor de un 7. Debido a que P7 está conectado a nuestro fotoresistor, cuando la cantidad de luz que ve el fotoresistor cambia, el voltaje de entrada (en P7) también cambia.

Mueva su mano sobre el fotoresistor (no necesita tocarlo). El valor que muestra la ventana debug en su PC debería estar cambiando de "1" a "0", dependiendo de si el fotoresistor ve un ambiente claro u oscuro.

Esta parte del circuito es llamada un divisor resistivo, y exploraremos esto en más detalle en un experimento futuro. Por ahora nos basta con saber que a medida que el fotoresistor cambia su resistencia (debido a los niveles variantes de luz), el voltaje en P7 cambia. Este cambio en el voltaje es una señal analógica. Ahora, debido a que los pines de entrada del microcontrolador, solamente reconocen señales binarias (digital) cuando el voltaje alcanza cierto punto, el valor de P7 ve un "1" (+ 5 volts) o un "0" (0 volts).

¡Lo que acabamos de crear es un "interruptor" que no necesita ser presionado! Es un sensor que reacciona automáticamente a los niveles de luz externa.

Experimento 4: Automatización Simple

Hagamos ahora algo divertido...
Cambia el programa de la siguiente forma:

```
x var word
n var bit
output 1
```

```
cerrar_la_puerta:
for x = 1 to 100
  pulsout 1,500
  pause 10
next
pause 10
```

```
ver_si_hay_alguien:
n=in7
if n=1 then abrir_la_puerta
pause 100
goto ver_si_hay_alguien
```

```
abrir_la_puerta:
for x = 1 to 100
  pulsout 1,1000
  pause 10
next
pause 10
```

```
n=in7
if n =0 then cerrar_la_puerta
goto abrir_la_puerta
```

¿Qué piensa usted que va a hacer éste programa?

No se desaliente si el programa se volvió un poco largo. Dividámoslo en pedazos más manejables.
Esta es la parte de inicialización del programa, ya sabemos qué hacen estos tres comandos.

```
x var word
n var bit
output 1
```

Usamos esta rutina en el Experimento 3. Envíe una serie de pulsos, que hacen rotar el servo a uno de sus extremos.

```
cerrar_la_puerta:  
for x = 1 to 100  
  pulsout 1,500  
  pause 10  
next  
pause 200
```

4

Esta parte del programa simplemente se fija si el fotoresistor detecta o no una “sombra”, y si la detecta hace que el programa vaya a la rutina que “abrirá la puerta”.

```
ver_si_hay_alguien:  
n=in7  
if n=1 then abrir_la_puerta  
pause 100  
goto ver_si_hay_alguien
```

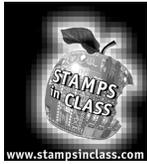
Si nuestro servo fuera más grande y estuviera conectado a la puerta de un supermercado, ésta rutina la abriría.

```
abrir_la_puerta:  
for x = 1 to 100  
  pulsout 1,1000  
  pause 10  
next  
pause 200
```

Esto es por “seguridad”. Mientras una persona esté parada en un sector cercano a la puerta, mantiene la puerta abierta. (¡Usted no querría aplastar a varios clientes, ellos comenzarían a comprar en algún otro lado!).

```
n=in7  
if n =0 then cerrar_la_puerta  
goto abrir_la_puerta
```

Experimento 4: Automatización Simple



Preguntas

1. ¿Qué es automatización?
2. ¿Qué hace el comando "debug" y por qué es tan útil?
3. ¿Qué hace el comando "n = in7"?. ¿En qué se diferencia en la ejecución el comando "hay_alguien_ahí = in7"?
4. Es éste experimento, ¿cómo sabe el microcontrolador, cuándo abrir la puerta?
5. Agregue comentarios apropiados al siguiente programa:

```
x var word _____  
n var bit _____  
output 1 _____  
cerrar_la_puerta: _____  
for x = 1 to 100 _____  
pulsout 1,500 _____  
pause 10 _____  
next _____  
pause 200 _____  
ver_si_hay_alguien: _____  
n=in7 _____  
if n=1 then abrir_la_puerta _____  
pause 100 _____  
goto ver_si_hay_alguien _____  
abrir_la_puerta: _____  
for x = 1 to 100 _____  
pulsout 1,1000 _____  
pause 10 _____  
next _____
```

```
pause 200 _____  
n=in7 _____  
if n =0 then cerrar_la_puerta _____  
goto abrir_la_puerta _____
```



¡Desafío!

1. Escriba un programa (completo, con comentarios) que encienda el LED (en P5) cada vez que el fotoresistor detecta una sombra.
2. Escriba un programa (con comentarios) que haga titilar el LED dos veces, y luego abra la puerta (rotar el servo) cuando el sensor detecte una sombra. Luego repite y hace todo otra vez.
3. Escriba un programa (con comentarios), que haga titilar dos veces el LED y abra la puerta (rote el servo) cuando el sensor detecta una sombra. Luego, mientras el sensor está detectando una sombra, haga parpadear el LED continuamente, hasta que la sombra desaparezca. Luego, que repita el ciclo y haga todo otra vez.
4. Escriba un programa que haga que el LED titile continuamente, hasta que el fotoresistor detecte una sombra. Una vez que la sombra es detectada, el LED es encendido mientras la puerta está siendo abierta. Una vez que la puerta está abierta, el LED se apaga hasta que la sombra se va. Luego se repite y hace todo otra vez.
5. Piense en condiciones donde el programa de este experimento no trabaja correctamente.

Experimento 4: Automatización Simple



¿Qué aprendí?

Complete las siguientes oraciones con las palabras de la lista.

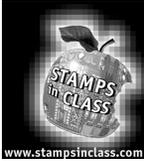
automáticas
foto-resistor
eficientemente
automatización
respuesta
entrada
óptico
ejecutarse
acción
datos
memoria

_____ es una aplicación fascinante para los microcontroladores. Sin ninguna _____ intencional por parte de los humanos, el BASIC Stamp puede hacer que las cosas sucedan, basado solamente en los sensores de entrada. Las puertas _____ de los supermercados son un gran ejemplo de una aplicación típica de microcontroladores.

Hay muchos tipos diferentes de sensores que pueden detectar movimiento en el "mundo exterior". En este experimento usamos un _____ cuyo valor depende de la cantidad de luz que detecta. Debido a que este sensor detecta la luz (o la ausencia de ella) se lo suele llamar un sensor "_____".

El comando "Debug" le permite al BASIC Stamp enviar _____ a la PC, de esa forma podemos determinar que tan bien está funcionando nuestro programa, y si está o no tomando las "decisiones" correctas. En este experimento el BASIC Stamp envió el valor de "n" a la PC, que fue asignado al valor de la _____ detectada en P7.

A medida que desarrollamos programas más grandes, se vuelve importante escribir "nuestro" código tan _____ como sea posible. Esto es importante por varias razones. Primera, la mayoría de los microcontroladores (como el BASIC Stamp) tienen una cantidad limitada de _____ para almacenar datos y el programa. Cuanto menos instrucciones podamos usar para realizar una tarea, más cualidades podremos agregarle a nuestro programa. Segunda, menos instrucciones para realizar una tarea le permiten al programa _____ más rápido, obteniendo un tiempo de _____ menor a las situaciones del "mundo exterior". Como ejemplo, si el programa no está bien escrito y la puerta no se abre rápidamente, el cliente puede chocarse contra ellas. ¡No tendríamos una muy buena relación con el cliente!



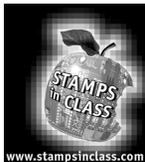
¿Por qué aprendí esto?

Sabiendo cómo “automatizar” ciertas tareas se pueden eliminar las tareas pesadas (y algunas veces peligrosas) de muchos tipos de trabajo. La automatización en la industria automovilística ha mejorado significativamente en muchas facetas del proceso de ensamble. La pintura y soldadura, por ejemplo, son actualmente realizadas por robots automatizados, con más consistencia, a menores costos y con menor

riesgo por parte de los empleados.

Como consecuencia negativa de esto, mucha gente en esta “era de la automatización”, está siendo desplazada. El reentrenamiento es necesario, pero afortunadamente, el próximo trabajo que obtengan no será ni tan peligroso ni tan tedioso.

Como consecuencia positiva, hay una gran oportunidad para nuevos productos y procesos que se realizan automáticamente. Esto ha expandido un nuevo tipo de industria llamada “innovación industrial”. Ya no es necesario que usted “ponga la tuerca A en el tornillo B”, hora tras hora, día tras día; ahora puede usar su imaginación para desarrollar nuevos y aún cambiantes productos, que mejoren la calidad de vida para todos.



¿Cómo puedo aplicarlo?

Hay muchas oportunidades para mejorar lo que podemos considerar las tareas mundanas.

Por ejemplo, usted puede diseñar una puerta de supermercado que no sólo se abra automáticamente sino también que observe cuánta gente está entrando, (o en éste caso también saliendo) de la tienda.

Usted puede observar este movimiento a través de periodos específicos durante el día, de esa forma, su microcontrolador podrá avisar al dueño de la tienda si va a necesitar cajeros adicionales, debido a que hay mucha más gente comprando en ese momento.

Esto puede mejorar el servicio al cliente, debido a que los empleados no serán tomados por sorpresa con un “cuello de botella” formado en las cajas. Su sistema sería una especie de “¡ alarma anti embotellamiento de clientes!”.

4

Experimento 5: Midiendo una Entrada



Experimento 5: Midiendo una Entrada

Como aprendimos hace mucho, cada uno de los 16 pines del “mundo real” del BASIC Stamp, puede ser configurado como entrada o como salida. Si el pin fue configurado como entrada, hay 7 instrucciones diferentes en el lenguaje PBASIC que pueden ser usadas. Cada uno de éstos comandos es conveniente para tipos específicos de condiciones de entrada.

Por ejemplo, en el Experimento 2 aprendimos a usar el comando llamado “input”. Si éste comando es usado en su programa, hace que el pin especificado se convierta en entrada. Luego, cada vez que queramos conocer el estado del pin (si está en “alto” o “bajo”), usamos la instrucción “if in2=0 then titilar”.

5

¿Qué es un

7 instrucciones “input” diferentes:

Ellas son:

Button
Count
Input
Pulsin
Rctime
Serin
Shiftin

Anteriormente ya hemos usado “input”. En éste experimento exploraremos “Pulsin”

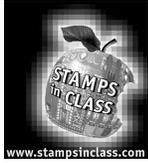
Esta línea de código “miró” en el pin, y regresó un valor. Si el valor en ese pin fue “0” entonces el código hubiera saltado a otro punto del programa, donde hubiera hecho titilar el LED. Si el valor de la entrada fue “1”, entonces hubiera continuado la ejecución del programa con la siguiente línea de código.

En cualquier caso, los valores que podían ser detectados con éste código eran binarios (“1” o “0”). Esto es conveniente para detectar si un pulsador ha sido o no presionado (Experimento 2), o incluso para detectar claridad u oscuridad con un fotoreistor, como hicimos en el Experimento 4.

El lenguaje del PBASIC tiene otros comandos, que ofrecen un gran nivel de sofisticación, cuando se tienen que detectar entradas. Si usted aún no bajó una copia gratis del BASIC Stamp Manual de www.stampsinclass.com, hágalo ahora. En él usted encontrará información de aplicación y una descripción completa de todos los comandos disponibles en el lenguaje PBASIC.

En este Experimento no sólo avanzaremos sobre la “detección de entradas” sino que también usaremos un circuito integrado muy popular, llamado el “temporizador 555”.

Experimento 5: Midiendo una Entrada



Partes Requeridas

Para éste experimento, usted necesitará lo siguiente:

- (1) BASIC Stamp II
- (1) "Plaqueta de Educación"
- (1) Cable de Programación
- (1) LED
- (1) CI temporizador CMOS 555
- (1) capacitor 10 microfaradios, 25 volt electrolítico
- (1) capacitor 1 microfaradio, 25 volt electrolítico
- (1) resistor 470 ohm, ¼ watt
- (1) potenciómetro 100K (resistor variable)
- (1) resistor 15K ohm, ¼ watt
- (1) resistor 1 K ohm, ¼ watt
- (1) batería de 9 volts o fuente de alimentación
- (varios) cables de conexión
- (1) P C con DOS 2.0 o superior, con un puerto serie disponible.
- (1) programa BASIC Stamp Editor

Distribuidores de éstos materiales son listados en el Apéndice A.



¡Constrúyalo!

Este circuito de hardware usa un Circuito Integrado llamado "temporizador 555". El '555' es realmente un "conjunto de circuitos electrónicos" (aquellos que se usan para llenar grandes áreas sobre una plaqueta de circuito impreso), que han sido miniaturizados y encapsulados dentro de un pequeño dispositivo de "8 pines dip"

como el que usaremos hoy. Aunque dentro del encapsulado plástico hay un conjunto sofisticado de circuitos, el '555' es bastante simple de usar en muchas aplicaciones diferentes. De hecho, en los años transcurridos desde su desarrollo, el '555' ha sido usado en una gran variedad de diseños y en innumerable cantidad de aplicaciones, debido a que puede hacer muchas cosas diferentes. Aunque no es "programable" como el BASIC Stamp, el '555' puede ser configurado con diferentes combinaciones de resistores y capacitores para realizar tareas diferentes.

¿Qué es Un...?

Circuito Integrado:

Normalmente llamados "CI", son circuitos electrónicos que han sido miniaturizados y combinados dentro de un pequeño encapsulado. Muchos tipos diferentes de CI han sido creados para un sinnúmero de aplicaciones. El temporizador '555' que usaremos en este experimento es un miembro de la familia de CI "lineales". El CPU Stamp es un "CI digital".

8 pin dip:

Esto se refiere al estilo de encapsulado del CI. El '555' tiene 8 pines, y éstos están dispuestos en dos líneas al costado del encapsulado (Dual Inline Package).

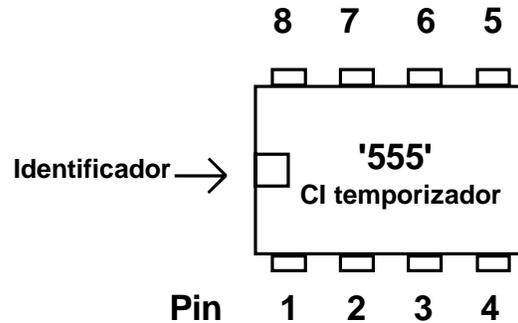
¿Qué es Un...?

Multivibrador astable:

Es un nombre extravagante para un circuito que sin "intervención externa", (de otros circuitos o dispositivos), emitirá continuamente una secuencia de pulsos. ¿Recuerda cuando creamos el chorro de pulsos para controlar el servo en los Experimentos 3 y 4? Lo mismo ocurre con éste circuito, excepto que el '555' alternará entre alto y bajo, sin que nosotros tengamos que escribir un programa. Es una versión en hardware del comando de software "Pulsout".

Un CI (**circuito integrado**) necesita tener algún tipo de identificación en su encapsulado, para decirnos cuál es el pin 1. El identificador es normalmente una ranura o marca ubicada en un extremo del encapsulado plástico. Ver Figura 5.1.

Figura 5.1: CI Temporizador 555
 Note la muesca en un extremo del encapsulado del chip.



5

Como con el BASIC Stamp, cada pin del '555' tiene una función en particular. Aunque el '555' es a "prueba de balas", conectar una señal eléctrica inadecuada al pin equivocado, puede dañar el dispositivo, así que sea cuidadoso y siga de cerca los diagramas.

El tipo de circuito que construiremos se llama "**multivibrador astable**". ¡No se asuste por este nombre tan complicado! Este significa que la salida del '555' alterna entre alto y bajo. Recuerde que en el experimento 1 usamos los comandos 'high' y 'low' para hacer titilar un LED. En realidad, eso es todo lo que el circuito '555' está haciendo. Está simplemente "oscilando". El circuito con el '555' que construiremos es el equivalente en hardware del Experimento 1.

La velocidad a la que parpadea la salida (el pin 3 del '555'), es controlada por los valores de un resistor y un capacitor. A medida que los valores de estos dispositivos cambien, la velocidad de "parpadeo" del '555' también cambiará.

Ya hemos usado resistores anteriormente. Ellos controlan la cantidad de corriente que fluye a través de un circuito dado. Debido a que queremos cambiar (convenientemente) la velocidad a la que el '555' parpadea, vamos a usar un resistor variable, también conocido como **potenciómetro**. Si usted alguna vez ajustó el volumen de una radio, usted ha usado uno.

Experimento 5: Midiendo una Entrada

¿Qué es Un...

Potenciómetro:

Es simplemente un resistor que cambia su valor a medida que es rotado (o en algunos casos deslizado). Recuerde que los resistores tienen dos conectores. Un potenciómetro tiene tres conectores. El conector central es el cursor que se desliza sobre un elemento resistivo. Cuanto más acerca el cursor a uno de sus extremos, menos resistencia hay entre el cursor y ese extremo. Los potenciómetros vienen en muchos valores diferentes, tales como 5K, 10k, 100k, 1 M ohms, y más. También vienen en muchas configuraciones físicas diferentes para acomodarse a diferentes diseños de productos. Todos operan esencialmente de la misma forma; un movimiento mecánico en el cursor cambia el valor resistivo del dispositivo.

Girando el dial del "potenciómetro", usted cambia el valor del resistor variable. Este giro cambia la velocidad a la que titila el '555'.

Cuando conecte el potenciómetro, notará que tiene tres terminales de conexión disponibles. Uno de éstos es el "cursor" y los otros dos son los extremos del elemento resistivo. Solamente necesitamos conectar uno de los extremos y el cursor a nuestro circuito, como se muestra en la Figura 5.2.

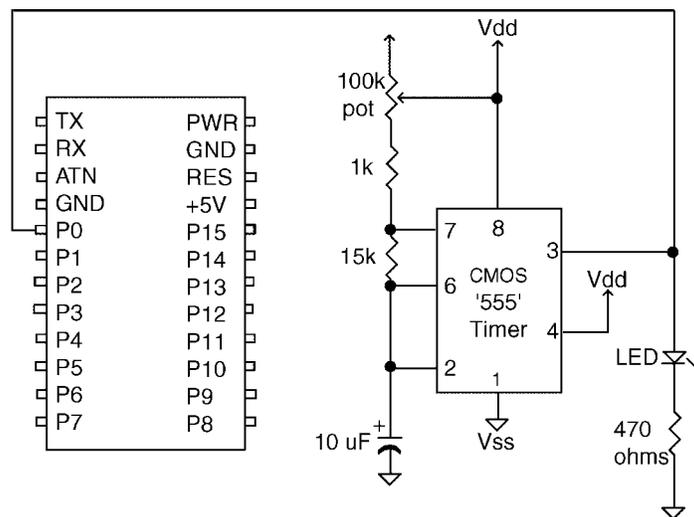


Figura 5.2: Esquema del Temporizador 555

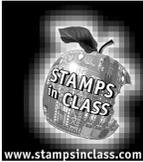
Esquema para el Experimento 5 en la Plaqueta de Educación

INFORMACION:

Sobre los esquemas:

Es una práctica común en los diagramas esquemáticos, dibujar los pines de un CI, en cualquier lugar en el que hagan más fácil de leer el diagrama.

Cuando usted inserta el CI temporizador '555' en la zona de prototipo de la Plaqueta de Educación, asegúrese de que el dispositivo quede "centrado" en la "línea divisoria", de esa forma los pines no están puestos en corto. Una vez que ha completado el circuito que se muestra en la Figura 5.2, siga adelante y alimente la Plaqueta de Educación.



¡Programelo!

Inicie el BASIC Stamp Editor. Si usted no recuerda cómo hacer esto, fíjese en un experimento anterior.

El pin 4 (en el '555') es un pin de "reset". Es una "entrada" al '555' y mientras éste pin vea un estado "alto", el '555' funcionará. Para que el circuito trabaje sin interacción con el BASIC Stamp, conectamos el pin 4 (del '555'), directamente a Vdd (alto). Esto mantiene al pin 4 en un estado alto, lo que le permite al '555' hacer titilar el LED. Ahora desconecte éste cable de Vdd y conéctelo a P1 para darle el control al BASIC Stamp.

5

¿Qué es un...

Reset:

Como se mencionó, es un pin de control en el CI temporizador '555'. Si conectamos este pin a P1 en el Stamp, y P1 es configurado como una "entrada", entonces la circuitería del '555' puede operar (aunque tal vez aleatoriamente). En éste caso, tenemos dos entradas (p1 en el Stamp y 'reset' en el '555') conectadas juntas. Un pin configurado como entrada en el Stamp tenderá a "flotar" a nivel alto. Esta es una "condición flotante" y no se garantiza que sea un verdadero valor "alto". Cuando convertimos a P1 en salida y la fijamos en nivel "alto", ésta también lleva a nivel alto al pin 4 del '555' y no nos da una condición "flotante".

Escriba el siguiente programa:

```
inicio:  
high 1  
pause 5000  
low 1  
pause 5000  
goto inicio
```

Ahora mientras presiona la tecla "ALT", escriba "r" (por "run") y presione "enter".

¿Qué pasó y por qué?. Si todo está trabajando apropiadamente, usted debería ver titilar el LED (por 5 segundos), y luego estar apagado por otros 5 segundos. Luego el programa se repite.

Debido a que P1 (en el BASIC Stamp) está conectado al pin reset del '555', cada vez que P1 va a "alto", le permite al '555' hacer titilar el LED. Y cada vez que P1 va a "bajo" (bajo el control de nuestro programa), apaga el circuito '555'.

Bien, dirá usted; ¿y qué?.

Bien, piénselo de ésta forma. Los microcontroladores son capaces de hacer sólo una cosa a la vez. Si queremos hacer titilar un LED como un "indicador de advertencia", entonces mientras el BASIC Stamp está haciendo su rutina "high - pause - low - pause - repetir" (para hacer titilar el LED), el BASIC Stamp no es capaz de hacer alguna otra cosa.

Experimento 5: Midiendo una Entrada

Ahora, como se muestra en el siguiente programa, usted puede encender “el circuito que hace titilar el LED”, y continuar (en su programa) y hacer algo “más importante”. Inténtelo.

```
x var word
low 1
inicio:
high 1          'enciende el circuito que titila
for x = 1 to 500
debug ? x      'cuenta hasta 500 en la pantalla
next           'mientras el LED parpadea
pause 3000
low 1          'apaga el circuito que titila
pause 2000
goto inicio    'regresa y repite todo otra vez
```

Lo que hemos hecho acá es quitarle la tarea al BASIC Stamp de hacer titilar el LED. La acción de hacer “titilar” el LED es cumplida por el circuito temporizador del ‘555’. Todo lo que necesita hacer el BASIC Stamp es habilitar o deshabilitar el “circuito titilador”. El BASIC Stamp puede entonces realizar una tarea más importante. En éste ejemplo, la tarea más importante es contar hasta 500 y mostrar los números en la pantalla. En el mundo real, sin embargo, usted puede estar buscando que se reúnan ciertas “condiciones de entrada” (en algún otro pin del BASIC Stamp).

¿Qué es Un...

Microfaradio:

Una unidad de medida para la cantidad de “carga” que puede ser almacenada en un capacitor. Similar al valor de “ohm” para los resistores, el microfaradio (para los capacitores), está disponible en un amplio rango de valores. 1 microfaradio es igual a 1/1.000.000 de un Faradio. Analizaremos los capacitores en un experimento posterior, pero por ahora, es necesario comprender que a menor valor, menor capacidad de carga tiene el capacitor, lo que da como resultado una oscilación más rápida en el circuito del 555.

En este circuito hemos usado un capacitor de 10 **microfaradios**. Apague la alimentación y reemplace el capacitor de 10 microfaradios con uno de 1 microfaradio. Asegúrese de observar la polaridad correcta de los capacitores. Continúe y encienda el circuito.

Reduciendo el valor del capacitor (C1) hemos aumentado la velocidad de parpadeo del LED (en éste caso, 10 veces). Aunque difícil de observar, el LED está aún parpadeando, pero a una velocidad mucho mayor.

En el Experimento 4 (donde controlamos la rotación de un servo), usamos un comando llamado “Pulsout”. Recuerde que “pulsout” genera como salida un único pulso con una longitud determinada por uno de los parámetros del comando.

Por ejemplo, para crear un pulso con una longitud de 1 milisegundo (en P1), el comando era: Pulsout 1, 500. El valor de 500 es el número de incrementos de dos microsegundos. Por lo tanto 500 veces 2 microsegundos = 1000 microsegundos, que es lo mismo que 1 milisegundo.

Ahora usaremos un comando nuevo llamado **"pulsin"**.

Pulsin es la contraparte de entrada a **pulsout**. En lugar de generar un pulso de salida de una longitud determinada, **pulsin** se fija en un pin de entrada particular y mide la longitud de un pulso entrante, y almacena ese valor en una variable.

Intente con el siguiente programa:

```
x var word
```

```
high 1
```

```
inicio:
```

```
pulsin 0,1,x
```

```
debug ? x
```

```
goto inicio
```

¿ Qué está pasando?

El pin 3 del circuito temporizador '555' está conectado al LED. El LED parpadea a una velocidad determinada por el valor del potenciómetro (y del capacitor). También conectamos la salida del '555' a P0 en el BASIC Stamp.

Analicemos el programa:

```
x var word
```

Esto simplemente declara una variable llamada "x", que tiene el tamaño word (16 bits). Esto significa que x puede valer hasta 65.536 (decimal).

```
high 1
```

Esto hace que P1 vaya a nivel alto, lo que hace que el '555' comience a oscilar.

```
inicio:
```

Una etiqueta para que el programa pueda regresar . . .

```
pulsin 0,1,x
```

Este simple comando le dice al BASIC Stamp que:

Mire un pulso de entrada en P1.

Espere que ese pulso vaya de bajo a alto.



Experimento 5: Midiendo una Entrada

Tan pronto como lo haga, comienza a "cronometrarlo", y continúa controlando el pin.

Tan pronto como el pulso vuelve a bajar, detiene el cronómetro y regresa un valor en la variable llamada "x". Este valor es en incrementos de dos microsegundos.

debug ? x

Esto muestra el valor de "x" en la PC.

goto inicio

Repite todo el programa.

Ahora, tratemos de ajustar el valor del potenciómetro. ¿ Qué pasa con el valor de "x"? ¿Puede explicar qué está pasando?.

Cada vez que cambia el valor del potenciómetro, la velocidad de parpadeo del LED es cambiada. **Pulsin** puede sólo medir hasta un máximo de 131 milisegundos, ésta es la razón por la cual aumentamos la velocidad de parpadeo del LED (disminuyendo el valor del capacitor). Con el capacitor de 10 microfaradios, la velocidad de parpadeo era demasiado lenta para que **pulsin** fuera capaz de medirla.

Usted ahora puede realmente medir el valor de parpadeo del LED. Tome el valor de "x" mostrado en la pantalla, multiplíquelo por dos (recuerde que **pulsin** mide en intervalos de 2 microsegundos) y obtendrá la longitud (en microsegundos) de cada "parpadeo".

El comando **pulsin** es un comando "detector de entrada" mucho más avanzado que una simple instrucción , como "in" (o input), pero ambos tienen sus propios usos, sólo depende de la aplicación.

Asegúrese de revisar el Apéndice B, y para una lista completa de los comandos PBASIC ver el BASIC Stamp Manual Version 1.9. Las Notas de Aplicación (Application Notes) también muestran modos diferentes de usar el comando **pulsin**.



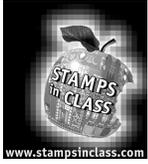
Preguntas

1. ¿Qué es un potenciómetro, y cuál es una de sus aplicaciones típica?
2. ¿Para qué podríamos querer usar un temporizador '555' para hacer titilar un LED en lugar de usar el BASIC Stamp?
3. ¿Qué hace el comando **pulsin**?
4. ¿Qué hace el pin Reset en el '555', y cómo puedo conectarlo al BASIC Stamp?
5. Agregue los comentarios apropiados al siguiente programa:

```
low 1  
inicio:  
high 1  
for x = 1 to 500  
debug ? x  
next  
pause 3000  
low 0  
pause 2000  
goto inicio
```



Experimento 5: Midiendo una Entrada



¡Desafío!

1. Escriba un programa, (completo, con comentarios), que le permita a un LED cada vez que un pulsador es conectado a P8. (Usted necesitará construir el hardware de éste circuito, si necesita una ayuda, refiérase al experimento 2 para saber cómo conectar un interruptor a un pin de entrada).
2. Dibuje el diagrama esquemático del circuito pensado en el punto 1.
3. Reemplace el pulsador del circuito del punto 1 con el circuito del fotosensor del Experimento 4. Conecte el fotosensor a P10 y escriba un programa que habilite el circuito del '555' por un período de 3 segundos, cada vez que el sensor "ve una sombra".
4. Escriba un programa que muestre (usando debug) el valor medido por **pulsin** y cada vez que el valor baje de 10.000 el pin de reset es enviado a nivel bajo, apagando el circuito que hace titilar el LED.



¿Qué aprendí?

Complete las siguientes oraciones con las palabras de la lista.

- | |
|------------------|
| microfaradios |
| salida |
| nivel de volumen |
| simultáneamente |
| valor |
| .131 seconds |
| hardware |
| debug |
| variable |
| integrado |
| pulsos |
| Pulsout |
| microsegundos |
| resistencia |
| LED |

5

El temporizador '555' es un circuito _____ que puede ser usado en aplicaciones muy diferentes. En este experimento, lo usamos para crear un chorro de _____ que hicieron titilar un LED. Luego conectamos la _____ del '555' al BASIC Stamp y pudimos medir la longitud de cada pulso en _____.

Un potenciómetro es una versión mecánica de un resistor _____. Para aumentar o disminuir la _____ del potenciómetro, usted gira físicamente la perilla, igual que cuando cambia el _____ de su equipo de música.

La "velocidad de parpadeo" del circuito temporizador '555' es determinada por el _____ de un resistor (medido en ohms) y un capacitor (medido en _____). Un microfaradio es igual a una millonésima (1 / 1.000.000) de un Faradio.

El comando Pulsin es el "equivalente de entrada" al comando _____. Pulsin puede medir pulsos de hasta _____ de longitud. En nuestro programa, usando el comando _____, podemos realmente medir la longitud de cada pulso que hace titilar al _____.

Utilizar hardware para realizar una tarea simple es, a veces, la mejor solución para realizar una tarea dada. Si usted tiene una aplicación en la que se necesita realizar dos tareas _____, usted necesitará sopesar las ventajas/desventajas del agregado de _____ adicional a su circuito.

Experimento 5: Midiendo una Entrada



¿Por qué aprendí esto?

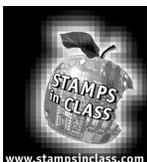
Este Experimento demuestra cómo conectar otros tipos de circuitos integrados al Microcontrolador. Los microcontroladores son capaces de hacer sólo una cosa a la vez. En muchos casos, ésta restricción no es un problema, debido a que los microcontroladores operan a velocidades muy altas. Pero, sin embargo, si usted necesita hacer más de una cosa a la vez, el problema puede ser fácilmente solucionado usando circuitería adicional como hicimos con el circuito integrado temporizador " 555".

El temporizador "555" ha sido usado en innumerables aplicaciones y productos a través de los años. En éste experimento usamos el temporizador en lo que llamamos el modo de " multivibrador astable". Este es un ejemplo relativamente simple de cómo descargar en otro componente, parte del proceso que normalmente tendría que haber realizado el microcontrolador. Muchos productos que usan un microcontrolador, como su unidad de procesamiento central (CPU), tienen circuitería adicional para cumplir ciertas tareas.

Esto no quiere decir que el microcontrolador no pueda realizar la tarea, sino que a veces es más rápido y más económico usar circuitería adicional para cumplir con una tarea dada. A medida que diseñe sus propios circuitos, necesitará tomar decisiones entre agregar código adicional o hardware adicional, para obtener la decisión más apropiada.

En algunos experimentos futuros, conectaremos muchos otros tipos de CI al BASIC Stamp que aumentarán significativamente sus capacidades de interactuar con el "mundo real".

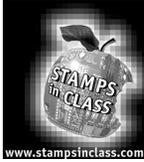
Y, por supuesto, sabiendo cómo interconectar diferentes tipos de circuitos integrados y componentes juntos, es una de las disciplinas requeridas para un Ingeniero Electrónico.



¿Cómo puedo aplicarlo?

A medida que experimente con microcontroladores, cubrirá muchas formas diferentes de interconectar cosas. Algunos de éstos métodos pueden obtenerse de alguna "nota de aplicación" que fue desarrollada por una compañía de semiconductores y otras pueden ser de su propia creación. En cualquier caso, conocer los métodos básicos de interconectar CI para obtener un producto que funcione, es una habilidad muy valiosa.

Muchos de usted pueden tener teléfonos celulares. Estos, como hemos mencionado antes, tienen microcontroladores como parte de su "cerebro básico". Pero, para que éstos dispositivos puedan desarrollar su máximo potencial, necesitan "circuitos de soporte"(no como nuestro temporizador '555'). Y la habilidad de diseñar una solución de hardware conveniente, siempre estará en demanda.



Experimento 6: Manual a Digital

En nuestro último experimento, usamos uno de los más populares circuitos integrales de todos los tiempos, el temporizador '555'. Con él construimos un "multivibrador astable", un nombre extravagante para un circuito que hace titilar un LED.

Recuerde que la velocidad de parpadeo fue controlada por los valores de dos componentes: un capacitor y un resistor. Para poder cambiar convenientemente la velocidad de parpadeo, sustituimos un resistor de valor fijo por un potenciómetro. Rotando manualmente la perilla del potenciómetro, cambiamos su resistencia.

Algunos dispositivos que actualmente están disponibles, nos permiten eliminar el elemento "manual" para cambiar el valor resistivo.

Usted puede estar familiarizado con los teléfonos celulares, que requieren que presione un botón en lugar de girar una perilla, para ajustar el volumen. En muchos casos, esto se realiza con un circuito similar al que vamos a realizar. En lugar de cambiar el volumen de un parlante, vamos a regresar a nuestro circuito titilador '555', y no sólo seremos capaces de prender y apagar un circuito, sino que también podremos variar (desde el programa), la velocidad de parpadeo del LED.

6

¿Qué es un...

Decisión Económica:

A medida que usted comience a crear sus propios circuitos, sean o no productos comerciales, el costo del "hardware electrónico" puede subir rápidamente, principalmente si hace esto como un pasatiempo fuera de una clase normal. Se vuelve realmente importante decidir cuál es la mejor aproximación para resolver una tarea particular. Algunas veces, la mejor y la más barata, no es la misma opción. Como descubriremos muchas veces, usted será capaz de resolver una tarea de ambas formas. Puede que le lleve más tiempo realizarlo en software, pero (si no contamos su tiempo), será casi siempre la decisión más barata.

Recuerde, todo esto se realiza en hardware. Es importante destacar que en cada diseño, usted tomará decisiones; tanto escribir un programa o implementar la función en hardware. No hay sólo una respuesta correcta. En muchos casos usted podría tomar cualquiera de las dos, y luego está la **decisión económica** – ¿qué método será menos caro?, y ¿podría el código de control realizar todas las funciones?.

Estas son situaciones que se plantean a través del proceso de diseño. Como descubriremos en éste experimento, hay muchos métodos diferentes para realizar una cierta tarea, y algunas veces es mejor dejarle al microcontrolador hacer la "tarea pesada", como por ejemplo cálculos, y dejar las tareas simples (hacer titilar el LED a una velocidad diferente) a un simple circuito de hardware.

¡ Saque su Plaqueta de Educación y haga que algo suceda!.



Partes Requeridas

Para éste experimento, usted necesitará lo siguiente:

- (1) BASIC Stamp II
- (1) "Plaqueta de Educación"
- (1) Cable de Programación
- (1) LED
- (1) CI temporizador 555
- (1) resistor digital X9313TP (CI)
- (1) capacitor 10 microfaradios, 25 volt electrolítico
- (1) capacitor 1 microfaradio, 25 volt electrolítico
- (1) resistor 470 ohm, ¼ watt
- (1) potenciómetro 100K (resistor variable)
- (1) resistor 15K ohm, ¼ watt
- (1) resistor 1 K ohm, ¼ watt
- (1) batería de 9 volts o fuente de alimentación
- (varios) cables de conexión
- (1) programa BASIC Stamp Editor

6



¡Constrúyalo!

Mire el circuito que se muestra en la Figura 6.1. Como usted habrá notado, éste es esencialmente el mismo circuito titilador de LED que usamos en el Experimento 5. Recuerde que la velocidad de parpadeo del LED fue cambiada girando el potenciómetro.

Ahora construya el circuito que se muestra en la Figura 6.1. Conecte el circuito integrado X9313 en el lugar del potenciómetro. También observe que la línea de reset del 555 (pin 4) está ahora conectada a P0 en el BASIC Stamp.

¿Qué es un...

X9313:

Muchos números diferentes y combinaciones de letras son usados en la industria de los semiconductores, para referirse a componentes individuales. Estos son simplemente números de referencia para tipos específicos de dispositivos.

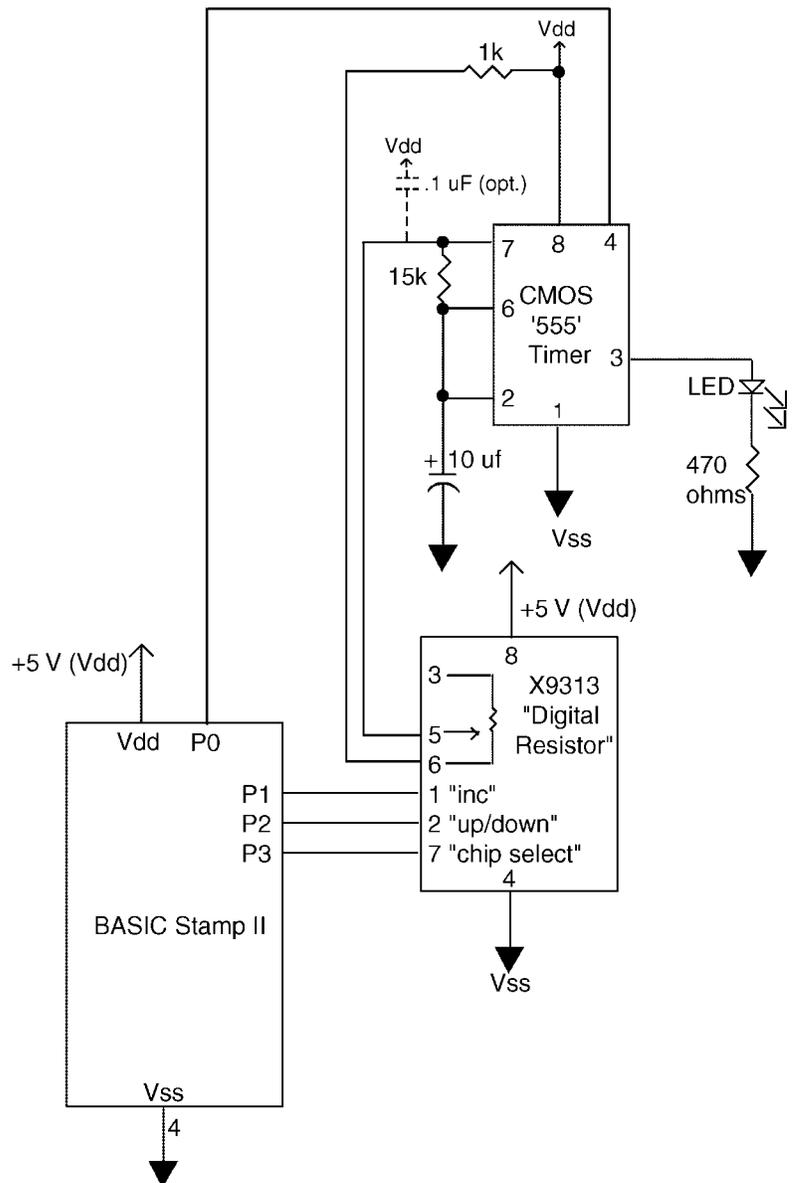
Este CI en particular es fabricado por una compañía llamada "Xicor".

Bien, ¿qué es el X9313? Recuerde que un potenciómetro es simplemente un resistor variable que cambia su resistencia cuando usted rota o mueve mecánicamente su perilla o cursor. Este cambio en la resistencia es responsable del cambio en la velocidad de parpadeo del circuito temporizador 555.

El X9313 es un potenciómetro "controlado digitalmente". Usted puede cambiar su resistencia (igual que como hacía con el potenciómetro manual), pero en lugar de mover la perilla mecánicamente, usted le enviará pulsos digitales desde el BASIC Stamp. Estos pulsos digitales cambian la ubicación de la perilla y por lo tanto cambian la velocidad de parpadeo del 555.

Figura 6.1:
Potenciómetro Digital:
 Reemplace el
 potenciómetro manual
 con el "resistor digital"
 Xicor en el Experimento 6,
 realizando un circuito
 como el mostrado.

Las líneas punteadas
 indican un capacitor
 "opcional" de 0.1 uF que
 puede ser necesitado por
 '555' que no sea fabricado
 por SGS / Thompson o TI.

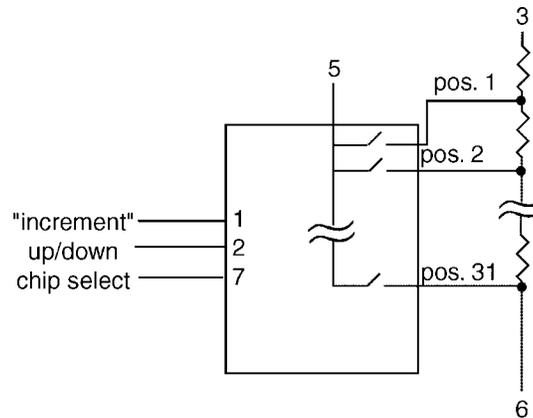


6

Experimento 6: Manual a Digital

Mire la figura 6.2. Este es un diagrama interno del circuito integrado X9313. El elemento resistivo tiene un valor total de 100k ohms. La perilla, si bien no es totalmente "continua", puede ser posicionada en cualquiera de sus 31 posiciones discretas.

Figura 6.2: Interior del X9313. Este potenciómetro "controlado digitalmente" reemplaza el lugar del potenciómetro manual.



6

INFORMACIÓN

Cero ohms:

Por varias razones, no son realmente "0" ohms. Hay resistencia en los cables, alambres y conexiones dentro del CI, así como también la resistencia propia del silicio. En el X9313, el brazo del cursor introduce aproximadamente 40 ohms en el circuito. Podemos normalmente ignorar éste tipo de errores, debido a que en el caso del X9313, la resistencia total es de 100k ohms. Cuarenta ohms realmente no afectan nuestro circuito, (por lo menos no lo suficiente para preocuparnos por ello).

Esto significa que el valor de 100k del X 9313 puede ser posicionado en 31 posiciones distintas, a lo largo del elemento resistivo. Cada paso representa aproximadamente 3230 ohms.

$$(31 \times 3230 = 100k).$$

Por lo tanto, cuando la perilla está en el "lado inferior" del elemento resistivo, la resistencia es de aproximadamente de cero ohms (medida entre el terminal "inferior" y el cursor). Cuando la perilla aumenta un lugar, la resistencia cambia a 3230 ohms. Un segundo incremento causará que el valor se vuelva aproximadamente 6460 ohms. Así sucesivamente, hasta 100k.

En el Experimento 5 creamos un circuito que podía habilitar (o desconectar) un circuito temporizador 555. Sin embargo, la única forma en que podíamos modificar la velocidad de parpadeo era mediante la rotación manual de un potenciómetro mecánico. Con el X9313 podemos lograr que el BASIC Stamp no sólo encienda o apague el circuito, sino que también podemos controlar la velocidad de parpadeo.



¡Programélo!

Asegúrese que su cable de programación esté conectado a la Plaqueta de Educación y luego aplíquela la alimentación.

¿Problemas?

Si obtiene un mensaje que dice, "Hardware not found", revise las conexiones entre la PC y la Plaqueta, y también asegúrese que la batería de 9 volt (o la fuente de alimentación), está conectada y cargada.

Trate de descargar nuevamente (tenga presionada la tecla ALT y luego presione "r"). Si aún no funciona, usted debe tener un error. Revise su programa para asegurarse que lo ha escrito correctamente.

Después de revisar sus conexiones presione ALT "r" nuevamente. Si aún recibe el mensaje "hardware not found" asegúrese de que su computadora esté funcionando en DOS, no en Win95. Si está funcionando en Win 95, entonces presione el botón Inicio (en el monitor) y elija "Reiniciar en modo MS-DOS".

Si después de intentar esto, usted aún tiene problemas, pídale ayuda a su instructor.

Escriba el siguiente programa en el editor del BASIC Stamp:

```
x var word
y var word
output 0
output 1
output 2
output 3
high 0
```

```
low 3
```

```
low 2
for x = 1 to 32
  high 1
  low 1
next
```

```
high 2
for y = 1 to 2
  high 1
  low 1
next
```

```
bucle:
goto bucle
```

Bien, ¿qué está haciendo el programa?. Si opera correctamente, su LED debería estar titilando (bastante rápido), si no revise su programa y asegúrese que lo ha escrito correctamente.

Ahora, pruebe esto: cambie el valor de "y" de 2 a 25.

Su programa (con comentarios), debería ahora verse así:

6

Experimento 6: Manual a Digital

x var word
y var word

output 0 '555 reset
output 1 'incremento
output 2 'arriba / abajo (up/down)
output 3 'chip select

high 0 'habilita el titilador
low 3 'selecciona el X9313

'Esta sección reinicia el resistor variable en cero ohms

low 2 'los pulsos hacen que el X9313 vaya a "bajo" (down)
for x = 1 to 32
 high 1 'pulso
 low 1
next 'repite 32 veces

'Esta rutina fija el valor del resistor variable a un valor determinado por 'y' en el bucle For...Next.

high 2 'los pulsos hacen que el X9313 vaya "arriba" (up)
for y = 1 to 25
 high 1
 low 1
next 'repite 25 veces (aumenta el resistor en 25 posiciones)

bucle: 'después de ajustar el resistor, se queda aquí y no hace nada
goto bucle

Ejecute el programa.

¿Qué pasó?, ¿Por qué?

Miremos más de cerca nuestro programa:

x var word
y var word

6

Nada nuevo, todo lo que estamos haciendo es inicializando un par de variables llamadas "x" e "y".

output 0 **'555 reset**

Esta salida es conectada a la línea de reset del temporizador. Un cambio en el valor de la salida de P0 puede encender o apagar el circuito titilador.

output 1 **'incremento**

Este comando lo hemos usado muchas veces, pero ¿a qué está conectado P1? La respuesta puede ser encontrada en la hoja de datos del resistor variable X9313. La señal de incremento (pin 1 de X9313) es donde enviaremos los pulsos para mover el cursor. Como verá más adelante, usaremos los comandos 'high' y 'low' para enviar los pulsos en esta línea de señal.

6

output 2 **'arriba / abajo (up/down)**

Otra salida al BASIC Stamp que está conectada a una señal llamada "arriba / abajo" del 9313. Cuando programamos el BASIC Stamp para hacer P2 alto, entonces cualquier pulso en el "pin de incremento" hará que el resistor variable aumente su valor. Inversamente, si P2 es bajo, entonces cualquier pulso en la línea de señal de incremento hará que el X9313 disminuya su valor.

output 3 **'chip select**

Con la señal de chip select seleccionamos si el X9313 puede cambiar su valor. Si la señal chip select (en el pin 7 del 9313) es alta, entonces cualquier pulso o señales "arriba/abajo" son simplemente ignoradas. La señal (que sale de P3 en el BASIC Stamp) es "activa baja".

high 0 **'habilita el titilador**

Bien, esta instrucción fija el nivel de P0 en alto, por lo tanto habilita el circuito titilador.

low 3 **'selecciona el X9313**

Debido a que P3 es "chip select" (y es activo bajo), poniendo un 0 en P3, habilita al X9313 a recibir pulsos y modificar su resistencia.

low 2 **'los pulsos hacen que el X9313 vaya hacia "abajo".**

Experimento 6: Manual a Digital

Cada vez que el X9313 es encendido por primera vez, no tenemos ni idea de cuál es la posición del cursor, debido a que no se puede observar visualmente. Por lo tanto ésta rutina, (comenzando con éste comando), enviará una cantidad suficiente de pulsos para fijar el cursor en el inicio de la excursión. Debido a que hay 32 posiciones discretas en el elemento resistivo, por lo menos, enviándole un pulso "bajo" (fijando la dirección con el comando "low 2") por lo menos 32 veces, nos aseguramos que está en el principio. No hay ningún problema si usted le envía pulsos hacia "abajo"(o hacia "arriba" según el caso) más de 32 veces, el X9313 simplemente llegará a la posición mínima o máxima y descartará los pulsos sobrantes.

for x = 1 to 32

Esta instrucción envía al cursor hasta la posición inferior del elemento resistivo.

high 1 'pulso
low 1
next 'repite 32 veces

6

Estos comandos hacen que P0 vaya a alto y luego a bajo, dando como resultado un pulso simple, y debido a que está en un bucle For...Next que se repite 32 veces, esto crea 32 pulsos. El "resistor digital" es ahora fijado en "0" ohms.

high 2 'los pulsos hacen que el X9313 vaya hacia "arriba"

La señal en P2 que había sido fijada anteriormente en nivel "bajo" (haciendo que el cursor se mueva hacia "abajo") es ahora fijada en nivel "alto", de esa forma, cualquier pulso (en el pin 1 del X9313) a partir de éste punto del programa, hará que el cursor se mueva hacia "arriba".

for y = 1 to 25

La cantidad de veces que se repite este bucle puede ser cambiada entre 1 y 32 veces. Pruebe diferentes valores. Cada vez que usted cambia el valor, ejecute el programa. Usted verá que el circuito titilador opera a una velocidad diferente, dependiendo de su valor.

high 1
low 1
next 'repite 25 veces (sube el valor del resistor 25 posiciones)

En este caso, repetiremos 25 veces, lo que hace que el cursor se mueva "25 posiciones" hacia arriba.

bucle: 'después de ajustar el resistor, se queda aquí y no hace nada.
goto bucle

En éste punto, nuestro programa simplemente se detiene y no hace nada más. En una "aplicación real", sin embargo, su programa podría continuar realizando otras tareas, mientras tanto, la velocidad de parpadeo se mantiene (habiendo sido fijada por nuestro microcontrolador), sin requerirse alguna otra interacción con el BASIC Stamp.

Bien, modifiquemos el programa para aumentar y disminuir la velocidad de parpadeo del LED.

x var word
y var word

output 0 '555 reset

output 1 'incremento

output 2 'arriba/abajo (up/down)

output 3 'chip select

high 0 'enciende el titilador

inicio:

low 2 'hace que el cursor vaya hacia "abajo" (down)

low 3 'selecciona (o habilita) el X9313 para recibir datos

for x = 1 to 32 'fija el valor del cursor en "0" lentamente

high 1

low 1

pause 200

next

high 2 'hace que el cursor vaya hacia "arriba" (up)

for y = 1 to 32 'hace que el cursor aumente 32 posiciones discretas

high 1

low 1

debug ? y

pause 200 'pausa por un corto período de tiempo, para que sea visible.

Next

goto inicio 'repite todo

6

Experimento 6: Manual a Digital

La primera parte del programa es la misma, con la excepción de la ubicación de la etiqueta "inicio", y las siguientes modificaciones:

¿Qué es Un...

Ohmetro:

Es un dispositivo que realmente mide el valor de un resistor en particular. Trabaja haciendo circular una corriente a través del resistor, y midiendo la caída de voltaje sobre él.

En muchos casos, usted necesita apagar su circuito si va a usar un óhmetro. En el caso del X9313, sin embargo, usted puede dejar encendido el circuito, siempre y cuando solamente toque los tres terminales (que representan los terminales del potenciómetro) con las puntas. De hecho, para poder medir el valor del cursor en cualquier punto, la alimentación debe estar aplicada. Este es un caso especial y no es una circunstancia normal de medición.

high 2 'hace que el cursor vaya hacia "arriba"

Esto hace que el cursor aumente su valor, siempre y cuando el chip esté seleccionado, y reciba pulsos en el pin "incremento".

for y = 1 to 32 'hace que el cursor aumente 32 posiciones discretas

Como nos imaginamos, ésta instrucción lleva al cursor hasta su máxima posición.

high 1
low 1

Estas instrucciones son las que crean el pulso.

debug ? y

Esta instrucción nos permite ver en qué posición está el cursor

pause 200 'pausa por un corto período de tiempo,
'para que sea visible.

Ahora, poniendo una "pausa", somos capaces de ver cómo es movido el cursor en dirección creciente. Se observará que el LED parpadea cada vez más lentamente, a medida que el cursor es movido (internamente), en el chip del resistor variable.

next

goto inicio 'repite todo otra vez

En lugar de detener el programa en un bucle "vacío", ahora lo enviamos hacia atrás para repetir todo nuevamente.

Experimente. Cambie algunos de los valores del bucle. ¿Qué pasa cuando usted no inicializa "reset" el cursor hasta su valor mínimo?. Intente enviar pulsos extra para enviar al cursor más allá de la posición 32 .

Si tiene acceso a un **óhmetro**, usted puede realmente medir los cambios de resistencia en el X9313 entre el cursor y cualquiera de los otros terminales del "potenciómetro".

6



Preguntas

1. ¿Qué significa "chip select" (selección de chip)?
2. ¿En qué se diferencia el X9313 del potenciómetro que usamos en el Experimento 5?
3. ¿Por qué el comando For...Next es tan útil en este Experimento?
4. ¿Por qué es tan importante saber armar el hardware, y no solamente escribir los programas para un microcontrolador?
5. Agregue los comentarios apropiados al siguiente programa:

6

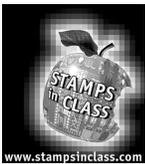
```
x var word _____  
y var word _____  
output 0 _____  
output 1 _____  
output 2 _____  
output 3 _____  
high 0 _____  
low 3 _____
```

```
low 2 _____  
for x = 1 to 32 _____  
high 1 _____  
low 1 _____  
next _____
```

```
high 2 _____
```

Experimento 6: Manual a Digital

for y = 1 to 20 _____
high 1 _____
low 1 _____
next _____
inicio: _____
goto _____
inicio _____



¡Desafío!

6

1. Escriba un programa (completo con comentarios) que varíe la velocidad de parpadeo del circuito del LED, desde una velocidad lenta a una rápida, cada 5 segundos
2. Escriba un programa que cambie la velocidad de parpadeo del circuito por cada segundo. La velocidad debe ir de rápido a lento. Una vez que el circuito haya hecho un ciclo completo (¡después de aproximadamente 31 segundos!) haga que el programa detenga el parpadeo y vaya a un bucle que "no haga nada". Muestre la posición de la "perilla" a medida que cambia, en su PC usando debug.
3. Reemplace el capacitor de 10 microfaradios en el circuito temporizador del 555 con un capacitor de 1 microfaradio. Conecte la salida del 555 al pin P5 del BASIC Stamp. Ahora dibuje el esquema completo de su circuito.
4. Usando el circuito del punto 3, modifique el programa del punto 2 para medir la longitud de los pulsos en incrementos de 2 microsegundos usando el comando **'pulsin'** en el pin 5 del BASIC Stamp. Haga que el programa se repita indefinidamente.



¿Qué aprendí?

Complete las siguientes oraciones con las palabras de la lista.

- diseño
- espacio de programa
- manual
- función
- pulsos
- hardware
- comunicarse
- descargar
- mostrar
- BASIC Stamp
- velocidad
- microcontroladores

El _____ es capaz de hacer muchas cosas diferentes. Todo depende de a que tipo de _____ está conectado. En este experimento construimos un circuito temporizador 555 y le permitimos al microcontrolador enviar una serie de _____ que no sólo habilitaron al circuito que titila, sino que también controlaron la verdadera _____ de los pulsos que salían del 555.

El control de la velocidad de parpadeo fue realizado reemplazando el potenciómetro " _____ " que usamos en el último experimento, con el "potenciómetro digital" X9313. Hay muchos dispositivos, tales como el X9313, que le permiten a los microcontroladores _____ y controlar cosas del "mundo exterior".

Al ajustar la velocidad de parpadeo en un cierto punto, el BASIC Stamp fue libre de "dedicarse a otros asuntos", tales como calcular o _____ datos en la PC.

Muchas veces es necesario liberar un valioso (y algunas veces costoso) _____ para tareas más importantes. Vimos esto inicialmente en el Experimento 4, y ahora vemos que el potencial de " _____ " parte del proceso del microcontrolador puede, en realidad, ser ilimitado.

Es posible, por ejemplo, que un sistema de control completo pueda ser construido sin ningún microcontrolador. Podría estar compuesto completamente por lógica "discreta". En realidad, hasta la aparición de los _____, ésta era la forma en que se construían los circuitos.

El microcontrolador, por lo tanto, nos brinda la opción de elegir si una _____ debería ser realizada con hardware o software. Todo es parte del proceso de _____ .

6

Experimento 6: Manual a Digital



¿Por qué aprendí esto?

Si usted elige este campo como una carrera, (tanto por primera vez, o más tarde en el transcurso de la vida) hay muchos atributos que pueden darle una ventaja sobre los demás. Como hemos aprendido en los dos últimos Experimentos, hay casi siempre más de una forma de resolver un problema. Saber cuándo resolverlo con “hardware” o

cuándo escribir el “código”, es un talento muy codiciado. Hay un número ilimitado de oportunidades para ingenieros de diseño innovadores, y especialmente, para aquellos que saben como “mezclar” mejor el hardware y el software dando como resultado un “mejor” producto.

Aún si usted no planea seguir una carrera relacionada con este campo, la habilidad de ser flexible en la aproximación a la solución del problema le ayudará a sobresalir en cualquier disciplina que usted elija.



¿Cómo puedo aplicarlo?

¿Por qué no diseñar un sistema de sonido estéreo controlado por un BASIC Stamp que detecte cuando alguien entra a la habitación? El BASIC Stamp podría detectar su presencia (similar a como usamos el fotoresistor en un Experimento anterior). Ahora, debido a que el BASIC Stamp está “siempre” mirando si alguien entra en la habitación,

no puede pasar mucho tiempo mandando señales de control continuas al control de volumen del estéreo.

Por lo tanto, utilizando un dispositivo como el X9313, el BASIC Stamp puede controlar otras entradas y, basándose en esos datos, fijar el nivel de sonido apropiado. Luego, cuando usted entra en la habitación (detectado por el BASIC Stamp), el programa hace subir gradualmente el volumen del estéreo a un nivel placentero. Si suena el teléfono, (usando otro tipo de sensor) el BASIC Stamp podría bajar automáticamente el volumen, de forma que usted pueda hablar sin ningún “ruido” de fondo.

6



Lista de Componentes

Todos los componentes (página siguiente) usados en los primeros seis experimentos están disponibles en negocios de componentes electrónicos comunes. Aquellos clientes que quieran comprar un kit completo, pueden hacerlo a través de Parallax. Parallax recarga un pequeño monto adicional por el empaque y envío de los componentes, logrando un buen precio final debido a las compras a los proveedores en grandes volúmenes. Los clientes pueden realizar un ahorro del 10% en grandes volúmenes (más de 25 unidades) en el Kit de Componentes de “¿Qué es un Microcontrolador?” construyendo sus propios Kits de componentes.

Los primeros seis experimentos requieren la Plaqueta de Educación –Kit Completo (#28102):

Parallax también comercializa el Kit de la Plaqueta de Educación (#28150), que consiste solamente de la plaqueta y los cables de interconexión. Use este kit si usted ya tiene un módulo BS2-IC y una fuente de alimentación. Piezas individuales pueden también ser ordenadas usando los códigos de stock Parallax que se muestran a continuación.

Plaqueta de Educación –Kit Completo (#28102)

Código Parallax	Descripción	Cantidad
28150	Plaqueta de Educación	1
800-00016	Cables de interconexión	6
BS2-IC	Módulo BASIC Stamp II	1
750-00008	Fuente de alimentación 300 mA 9 VCC	1
800-00003	Cable Serial	1

Kit Plaqueta de Educación (#28150)

Código Parallax	Descripción	Cantidad
28102	Plaqueta de Educación y cables de interconexión	1
800-00016	Cables de interconexión	6

Esta documentación impresa es muy útil para obtener información adicional:

Documentación BASIC Stamp (en inglés)

Código Parallax	Descripción	¿Disponibilidad en Internet?
27919	Manual BASIC Stamp Version 1.9	http://www.stampsinclass.com
28123	“What’s a Microcontroller?” Text	http://www.stampsinclass.com
27951	“Programming and Customizing the BASIC Stamp Computer”	Índice solamente en http://www.stampsinclass.com

Apéndice A: Lista de Componentes y Suministros

Las primeras seis lecciones necesitan el Kit de Componentes de ¿Qué es un Microcontrolador? (#28123)

El contenido del Kit de Componentes de “¿Qué es un Microcontrolador?” es listado a continuación. Estos componentes se necesitan para los experimentos uno al seis. En caso de necesitar un componente de reemplazo específico de Parallax el código de stock de cada componente es listado abajo. Si usted quiere comprar estos componentes en algún otro lado y necesita ayuda para encontrar el distribuidor apropiado para estos materiales, siéntase libre de contactarnos en stampsinclass@parallaxinc.com.

Kit de Componentes de ¿Qué es un Microcontrolador? (#28123)

Código Parallax	Descripción	Cantidad
150-04710	resistor 470 ohm ¼ watt 5%	6
350-00006	LED, color rojo	6
150-01030	resistor 10K ¼ watt 5%	2
400-00002	Pulsadores de 4 pines	2
800-00016	Paquete de 10 cables de interconexión	1
900-00002	Servo de hobby (Hitec HS 300 o equivalente)	1
201-03080	capacitor electrolitico de 3300 uF **	1
451-00301	Ficha de conexión de 3 pin para protoboard	1
150-01020	resistor 1K ohm ¼ watt 5%	5
350-00009	Foto-resistor (EG&G Vactec)	1
604-00006	temporizador CMOS 555 - 8 pin dip (use SGS/Thompson or TI)*	1
201-01050	capacitor electrolítico 1uF **	1
201-01062	capacitor electrolítico 10 uF **	1
150-01530	15k resistor 1/4 watt	1
152-01040	Potenciómetro 100k	1
152-01041	Potenciómetro de estado sólido de 100k (Xicor X9313TP)	1

* Si no usa el componente especificado, usted puede tener que conectar un capacitor de 0.1 uF entre el Pin 7 y Vdd en el Experimento 6, Manual a Digital.

** El voltaje máximo del capacitor debe ser mayor o igual a 16V.



Suministros

La red de distribución de Parallax funciona en aproximadamente 40 países por todo el mundo. Algunos de esos distribuidores también son distribuidores autorizados de "Stamp en Clase". Los distribuidores de Stamp en Clase normalmente tienen en stock la Plaqueta de Educación (#28102 y #28150) y, en algunos casos, el Kit de Componentes de "¿Qué es un Microcontrolador?" (#28123). También se listan algunas compañías de componentes electrónicos para aquellos clientes que deseen armar sus propios Kits de Componentes.

País	Compañía	Notas
Estados Unidos	Parallax, Inc. 3805 Atherton Road, Suite 102 Rocklin, CA 95765 USA (916) 624-8333, fax (916) 624-8003 http://www.stampsinclass.com http://www.parallaxinc.com	Suministra Parallax y Stamp en Clase. Fabricante del BASIC Stamp.
Estados Unidos	Jameco 1355 Shoreway Road Belmont, CA 94002 (650) 592-8097, fax (650) 592-2503 http://www.jameco.com	Distribuidor de Parallax Stamp en Clase. También es una gran fuente de componentes.
Estados Unidos	Peter H. Anderson 915 Holland Road Bel Air, MD 21014 (410) 838-6500, fax (410) 836 8526 http://www.phanderson.com	Distribuidor y profesor de Parallax Stamp en Clase
Estados Unidos	Digi-Key Corporation 701 Brooks Avenue South Thief River Falls, MN 66701 (800) 344-4539, fax (218) 681-3380 http://www.digi-key.com	Fuente de componentes electrónicos. Distribuidor Parallax. Excelente fuente de componentes. Puede tener en stock la Plaqueta de Educación.
Estados Unidos	Mouser Electronics 345 South Main Mansfield, TX 76203 (800) 346-6873, fax (817) 483-6899 http://www.mouser.com	Fuente de componentes electrónicos. Distribuidor Parallax. Excelente fuente de componentes. Puede tener en stock la Plaqueta de Educación en 1999.

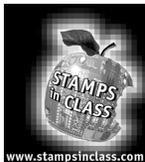
Apéndice A: Lista de Componentes y Suministros

Australia	Microzed Computers PO Box 634 Armidale 2350 Australia Phone +612-67-722-777, fax +61-67-728-987 http://www.microzed.com.au	Distribuidor Parallax. Distribuidor de Stamp en Clase. Excelente soporte técnico.
Australia	RTN 35 Woolart Street Strathmore 3041 Australia phone / fax +613 9338-3306 http://people.enternet.com.au/~nollet	Distribuidor Parallax y Stamp en Clase.
Canadá	Aerosystems 3538 Ashby St-Laurent, QUE H4R 2C1 Canada (514) 336-9426, fax (514) 336-4383	Distribuidor Parallax y Stamp en Clase.
Canadá	HVV Technologies 300-8120 Beddington Blvd NW, #473 Calgary, AB T3K 2A8 Canada (403) 730-8603, fax (403) 730-8903 http://www.hvwtech.com	Distribuidor Parallax y Stamp en Clase.
Alemania	Elektronikladen W. Mellies Str. 88 32758 Detmold Germany 49-5232-8171, fax 49-5232-86197 http://www.elektronikladen.de	Distribuidor Parallax y Stamp en Clase.
Nueva Zelanda	Trade Tech Auckland Head Office, P.O. Box 31-041 Milford, Auckland 9 New Zealand +64-9-4782323, fax 64-9-4784811 http://www.tradetech.com	Distribuidor Parallax y Stamp en Clase.

Apéndice A: Lista de Componentes y Suministros

Holanda	Antratek Kanaalweg 33 2903 LR Capelle A/S IJssel Netherlands +31-10450-4949, fax 31-10451-4955 antratek@box.nl	Distribuidor Parallax y Stamp en Clase.
Inglaterra	Milford Instruments Milford House 120 High St., S. Milford Leeds YKS LS25 5AQ United Kingdom +44-1-977-683-665 fax +44-1-977-681-465 http://www.milinst.demon.co.uk	Distribuidor Parallax y Stamp en Clase.

Apéndice A: Lista de Componentes y Suministros



Libros y Recursos de Internet

Si usted es nuevo en programación, electrónica o con los BASIC Stamps, hay varias fuentes de impresos y sitios de internet que puede querer investigar.

Libros y Publicaciones

Programming & Customizing the Basic Stamp Computer por Scott Edwards. ISBN 0-07-913684-2. Disponible en Parallax (#27905) y Amazon (<http://www.amazon.com>).

Parallax BASIC Stamp Manual Version 1.9 de Parallax (#27919) y distribuidores.

Nuts and Volts Magazine Stamp Applications. Publicada cada mes en la revista Nuts and Volts (<http://www.nutsvolts.com>), con artículos anteriores disponibles para descargar gratis en nuestro sitio web.

Internet

Sitio web Parallax <http://www.parallaxinc.com> y sitio web de Parallax Stamps en Clase <http://www.stampsinclass.com> incluyen recursos descargables gratuitamente de BASIC Stamp.

Peter H. Anderson, un educador entusiasta del BASIC Stamp y distribuidor del Stamps en Clase en <http://www.phanderson.com>.

Al Williams Consulting tiene el BASIC Stamp Project of the Month at <http://www.al-williams.com>.

Apéndice B: Guía de Referencia Rápida de PBASIC



Guía de Referencia Rápida de PBASIC

El Parallax BASIC Stamp Manual Version 1.9 (en inglés) consiste de aproximadamente 450 páginas de descripciones de comandos PBASIC, notas de aplicación, y esquemas. El documento completo está disponible para descargar en <http://www.parallaxinc.com> y <http://www.stampsinclass.com> en formato Adobe PDF, pero también puede ser adquirido por estudiantes e instituciones educativas.

Esta Guía de Referencia Rápida de PBASIC es una versión reducida de los comandos del BASIC Stamp II.

BIFURCACIÓN

IF...THEN

IF *condición* THEN *direcciónEtiqueta*

Evalúa la condición y, si es verdadera, se dirige al punto del programa marcado por *direcciónEtiqueta*.

- *Condición* es una expresión, tal como "x=7", que puede ser evaluada como verdadera o falsa.
- *DirecciónEtiqueta* es una etiqueta que especifica donde ir en el caso que la condición sea verdadera.

BRANCH

BRANCH *indicador*, [*dirección0*, *dirección1*, ...*direcciónM*]

Se dirige a la dirección especificada por el indicador (si está en el rango).

- *Indicador* es una variable / constante que especifica a cuál de las direcciones listadas dirigirse (0-N).
- *Direcciones* son las etiquetas que especifican a qué lugar dirigirse.

GOTO

GOTO *direcciónEtiqueta*

Se dirige al punto del programa especificado por *direcciónEtiqueta*.

- *DirecciónEtiqueta* es una etiqueta que especifica a qué lugar dirigirse.

GOSUB

GOSUB *direcciónEtiqueta*

Guarda la dirección de la instrucción siguiente a GOSUB, luego se dirige al punto del programa especificado por *direcciónEtiqueta*.

- *DirecciónEtiqueta* es una etiqueta que especifica a qué lugar dirigirse.

RETURN

Regresa de una subrutina. Regresa el programa a la dirección (instrucción) inmediatamente siguiente al GOSUB más reciente.

REPETICIÓN

FOR...NEXT

FOR *variable* = *inicial* to *final* {*paso*} ...NEXT

Crea un bucle repetitivo que ejecuta las líneas de programa entre FOR y NEXT, incrementando o disminuyendo el valor de la variable de acuerdo al *paso*, hasta que el valor de la variable iguale al valor *final*.

- *Variable* es una variable tipo nib, byte, o word usada como contador.
- *Inicial* es una variable o una constante que especifica el valor inicial de la *variable*.
- *Final* es una variable o una constante que especifica el valor final de la *variable*. Cuando el valor de la *variable* supera el valor *final*, end, el bucle FOR . . . NEXT detiene su ejecución y el programa continúa en la instrucción siguiente a NEXT.
- *Paso* es una variable o constante opcional que fija el valor en que aumenta o disminuye la *variable* en cada bucle de FOR / NEXT. Si *inicial* es mayor que *final*, PBASIC2 interpreta que *paso* es negativo, aunque no se use un signo menos.

FUNCIONES NUMÉRICAS

LOOKUP

LOOKUP *índice*, [*valor0*, *valor1*,... *valorM*], *variable*

Busca el valor especificado por el índice y lo guarda en la variable. Si el índice excede el máximo valor de índice de la lista, la variable no es afectada. Un máximo de 256 valores puede ser incluido en la lista.

- *Índice* es una constante, expresión o variable tipo bit, nibble, byte o word.
- *Valor0*, *valor1*, etc. son constantes, expresiones o variables tipo bit, nibble, byte o word.
- *Variable* es una variable tipo bit, nibble, byte o word.

LOOKDOWN

LOOKDOWN *valor*, {*comparador*,} [*valor0*, *valor1*,... *valorM*], *variable*

Compara un valor con los de la lista en función del *comparador* y guarda la ubicación (índice), en la variable.

- *Valor* es una constante, expresión o variable tipo bit, nibble, byte o word.
- *Comparador* es =, <>, >, <, <=, =>. (= es la opción por defecto).
- *Valor0*, *valor1*, etc. son constantes, expresiones o variables tipo bit, nibble, byte o word.
- *Variable* es una variable tipo bit, nibble, byte o word.

RANDOM

RANDOM *variable*

Genera un número pseudo-aleatorio.

- *Variable* es una variable tipo byte o word en el rango de 0..65535.

DIGITAL I/O (ENTRADA/SALIDA DIGITAL)

INPUT

INPUT *pin*

Convierte al pin especificado en entrada (escribe un 0 en el bit correspondiente de DIRS).

- *pin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15.

OUTPUT

OUTPUT *pin*

Convierte al pin especificado en salida (escribe un 1 en el bit correspondiente de DIRS).

- *pin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15.

REVERSE

REVERSE *pin*

Si el pin es de salida, lo hace de entrada. Si el pin es de entrada, lo hace de salida.

- *pin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15.

LOW

LOW *pin*

Convierte al pin en salida y la pone en estado bajo (escribe un 1 en el bit correspondiente de DIRS y un 0 en el bit correspondiente de OUTS).

- *pin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15.

HIGH

HIGH *pin*

Convierte al pin en salida y la pone en estado alto (escribe un 1 en el bit correspondiente de DIRS y OUTS).

- *pin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15.

TOGGLE

TOGGLE *pin*

Invierte el estado de un pin.

- *pin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15.

Apéndice B: Guía de Referencia Rápida de PBASIC

PULSIN

PULSIN *pin, estado, variable*

Mide un pulso de entrada (resolución de 2 μ s).

- *pin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15.
- *estado* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..1.
- *variable* es una variable tipo bit, nibble, byte o word.

Las mediciones se toman en intervalos de 2uS y el tiempo máximo es de 0.13107 segundos.

PULSOUT

PULSOUT *pin, período*

Genera un pulso de salida (resolución de 2 μ s).

- *pin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15.
- *período* es una constante, expresión o variable tipo bit, nibble, byte o word, en el rango de 0..65535 que especifica la duración del pulso en unidades de 2uS.

BUTTON

BUTTON *pin, presionado, retardo, velocidad, espaciotrabajo, estado, etiqueta*

Elimina el rebote, realiza auto-repetir, y se dirige a la etiqueta si un botón es activado.

- *pin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15.
- *presionado* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..1.
- *retardo* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..255.
- *velocidad* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..255.
- *espaciotrabajo* es una variable tipo byte o word. Se inicia con 0 antes de ser usada.
- *estado* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..1.
- *etiqueta* es la etiqueta a la que debe saltar el programa cuando se presiona el botón.

SHIFTIN

SHIFTIN *dpin, cpin, modo, [resultado{\bits} { ,resultado{\bits}... }]*

Convierte los bits recibidos de serie a paralelo y los almacena.

- *dpin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15 que especifica el pin de datos.
- *cpin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15 que especifica el pin del reloj (clock).
- *modo* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..4 que especifica el modo de transmisión. 0 o MSBPRES = msb primero, pre-reloj, 1 o LSBPRE = lsb primero, pre-reloj, 2 o MSBPOST = msb primero, post-reloj, 3 o LSBPOST = lsb primero, post-reloj.
- *resultado* es una variable tipo bit, nibble, byte o word donde el dato recibido es guardado.
- *bits* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 1..16 que especifica el número de bits a recibir en resultado. El valor predeterminado es 8.

SHIFTOUT

SHIFTOUT *dpin, cpin, modo, [datos{\bits} { ,datos{\bits}... }]*

Envía los datos en forma serial.

- *dpin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15 que especifica el pin de datos.
- *cpin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15 que especifica el pin del reloj (clock).
- *Modo* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..1 que especifica el modo de transmisión. 0 o LSBFIRST = lsb primero, 1 o MSBFIRST = msb primero.
- *datos* es una constante, expresión o variable tipo bit, nibble, byte o word que contiene el dato a ser enviado.
- *bits* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 1..16 que especifica el número de bits a enviar. El valor predeterminado es 8.

COUNT

COUNT *pin, período, resultado*

Cuenta el número de ciclos en un pin, por un *período* de milisegundos, y guarda ese número en resultado.

- *pin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15.
- *período* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..65535.
- *resultado* es una variable tipo bit, nibble, byte o word.

XOUT

XOUT *mpin*, *zpin*, [*casaclavecomando*{*ciclos*} {, *casa**clavecomando*{*ciclos*}...]}**

Genera códigos de control de línea X-10. Se usa con los módulos de interface TW523 o TW513.

- *mpin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15 que especifica el pin de modulación.
- *zpin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15 que especifica el pin de cruce por cero.
- *casa* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15 que especifica el código de casa A..P.
- *clavecomando* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15 que especifica las claves 1..16 o es uno de los comandos mostrados en BASIC Stamp Manual Version 1.9. Estos comandos incluyen encender y apagar luces.
- *ciclos* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 2..65535 que especifica el número de veces que se transmite un comando. (Predeterminado es 2).

SERIAL I/O(E/S SERIAL)

SERIN

SERIN *rpin*{\fpin}, *baudmodo*, {*petiqueta*,} {*tiempoespera*, *tetiqueta*,} [*datosentrada*]

Recibe datos asincrónicamente (como en RS-232).

- *rpin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..16.
- *fpin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15.
- *baudmodo* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..65535.
- *petiqueta* es una etiqueta a la cual saltar en caso de error de paridad.
- *tiempoespera* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..65535 que representa el número de milisegundos a esperar por un mensaje entrante.
- *tetiqueta* es una etiqueta a la cual saltar en caso de demora.
- *Datosentrada* es un conjunto de nombres de variables, expresiones o constantes, separados por comas que pueden tener los formatos disponibles para el comando DEBUG, excepto los formatos ASC y REP. Además, los siguientes formatos están disponibles:
 1. STR vector de bytes\L{E}
 2. SKIP L
 3. WAITSTR vector de bytes{L}.
 4. WAIT (valor {,valor...}) Esperar por una secuencia de hasta seis bytes.

Apéndice B: Guía de Referencia Rápida de PBASIC

SEROUT

SEROUT *tpin*{\fpin}, *baudmodo*, {*pausa*,} {*tiempoespera*, *tetiqueta*,} [*datossalida*]

Envía datos asincrónicamente (como en RS-232).

- *tpin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..16.
- *fpin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15.
- *baudmodo* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..60657.
- *pausa* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..65535 que especifica un tiempo (en milisegundos) de retardo entre los bytes transmitidos. Este valor sólo puede ser especificado si no se da un valor a *fpin*.
- *tiempoespera* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..65535 representa el número de milisegundos a esperar para la señal de transmisión del mensaje. Este valor sólo puede ser especificado si se da un valor a *fpin*.
- *tetiqueta* es una etiqueta a la cual saltar si se sobrepasa el *tiempoespera*. Se especifica con *fpin*.
- *datossalida* es un conjunto de constantes, expresiones y nombres de variables separados por comas y opcionalmente precedido por modificadores disponibles en el comando DEBUG.

ANALOG I/O (E/S ANALÓGICA)

PWM

PWM *pin*, *duty*, *ciclos*

Puede ser usado para generar voltajes de salida analógicos (0-5V) usando un capacitor y un resistor. Genera una salida por modulación de ancho de pulso, y luego el pin vuelve a ser entrada.

- *pin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15.
- *duty* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..255.
- *ciclos* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..255 que representa la cantidad de ciclos de 1ms a enviar a la salida.

RCTIME

RCTIME *pin*, *estado*, *variable*

Mide un tiempo de carga/descarga RC. Puede ser usado para medir potenciómetros.

- *pin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15.
- *estado* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..1.
- *Variable* es una variable tipo bit, nibble, byte o word.

SONIDO

FREQOUT

FREQOUT *pin*, *milisegundos*, *freq1* {*freq2*}

Genera una o dos ondas sinusoidales de las frecuencias especificadas (0 – 32.767 Hz).

- *pin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15.
- *milisegundos* es una constante, expresión o variable tipo bit, nibble, byte o word.
- *freq1* y *freq2* son constantes, expresiones o variables tipo bit, nibble, byte o word en el rango de 0..32767 que representan las frecuencias correspondientes.

DTMFOUT

DTMFOUT *pin*, {*tiempoencendido*, *tiempoapagado*}[*tono*{*tono*...}]

Genera pulsos telefónicos DTMF.

- *pin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15.
- *tiempoencendido* y *tiempoapagado* son constantes, expresiones o variables tipo bit, nibble, byte o word en el rango de 0..65535.
- *tono* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15.

ACCESO A EEPROM

DATA

{*puntero*} **DATA** {@*ubicación*}, {*WORD*} {*datos*}{(*tamaño*)} {, { *WORD*} {*datos*}{(*tamaño*)}}...

Almacena datos en la EEPROM antes de descargar el programa PBASIC.

- *puntero* es un nombre opcional de una constante no definida o variable tipo bit, nibble, byte o word al que se le asigna el valor del primer lugar de memoria en el que los datos son almacenados.
- *ubicación* es una constante, expresión o variable tipo bit, nibble, byte o word opcional que designa el primer lugar de memoria en el que los datos son escritos.
- *word* es una llave opcional que hace que los datos sean almacenados en la memoria como dos bytes separados.
- *datos* es una constante o expresión opcional a ser escrita en la memoria.
- *tamaño* es una constante o expresión opcional que designa el número de bytes de datos definidos o no, para escribir o reservar en la memoria. Si *datos* no es especificado, entonces se reserva una cantidad de espacio indefinido, y si *datos* es especificado, se reserva la cantidad especificada por tamaño.

Apéndice B: Guía de Referencia Rápida de PBASIC

READ

READ *ubicación, variable*

Lee un byte de la EEPROM y lo almacena en variable.

- *ubicación* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..2047.
- *Variable* es una variable tipo bit, nibble, byte o word.

WRITE

WRITE *ubicación, datos*

Escribe un byte en la EEPROM.

- *ubicación* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..2047.
- *datos* es una constante, expresión o variable tipo bit, nibble, byte o word.

TIEMPO

PAUSE

PAUSE *milisegundos*

Hace una pausa en la ejecución por 0–65535 milisegundos.

- *milisegundos* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..65535.

CONTROL DE ENERGÍA

NAP

NAP *período*

Descansa por un corto período. El consumo de energía es reducido a 50 uA (sin cargas conectadas).

- *período* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..7 que representa intervalos de 18ms.

SLEEP

SLEEP *segundos*

Duerme por 1–65535 segundos. El consumo de energía es reducido a 50 uA.

- *segundos* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..65535 que especifica el número de segundos a dormir.

END

END

- Duerme hasta que se interrumpa y reinicie la alimentación o se conecte a la PC. El consumo de energía es reducido a 50 μ A.

DEPURACIÓN DEL PROGRAMA

DEBUG

DEBUG *datosalida*{*datosalida*...}

Envía variables a ser vistas en la PC.

- *datosalida* es una cadena de texto, constante o variable tipo bit, nibble, byte o word. Si no se especifican modificadores DEBUG muestra caracteres en ascii sin espacios ni separación de oraciones a continuación del valor.



Leyendo el Código de Colores de los Resistores

la mayoría de los tipos comunes de resistores tienen bandas de colores que indican su valor. Los resistores que usaremos en esta serie de experimentos son normalmente "1/4 watt, de carbón, con una tolerancia del 5%". Si se fija en la secuencia de bandas observará que una de las bandas (en un extremo) es

dorada. Ésta es la cuarta banda, y el color dorado significa que tiene una tolerancia del 5%.

Las primeras tres bandas nos dicen el valor, medido en "ohms". A mayor valor, menor es el flujo de corriente a través del resistor (a un voltaje dado).

Los valores de los colores son los siguientes:

negro	0
marrón	1
rojo	2
naranja	3
amarillo	4
verde	5
azul	6
violeta	7
gris	8
blanco	9

Para determinar el valor del resistor, mire el primer color, determine su valor de la lista de arriba y escríbalo. Haga lo mismo con la segunda banda. La tercer banda es la cantidad de ceros a agregar. Por ejemplo:

Un resistor tiene las siguientes bandas de color:

- Banda 1. = rojo
- Banda 2. = violeta
- Banda 3. = amarillo
- Banda 4. = dorado

Mirando la lista de arriba vemos que el rojo vale 2.

Así que escribimos: "2".

Violeta tiene un valor de 7.

Así que escribimos: "27"

Amarillo tiene un valor de 4.

Apéndice C: Leyendo el Código de Colores de los Resistores

Así que escribimos: "27 y cuatro ceros" o "270.000".

Este resistor tiene un valor de 270,000 ohms (o 270k) y una tolerancia del 5%.