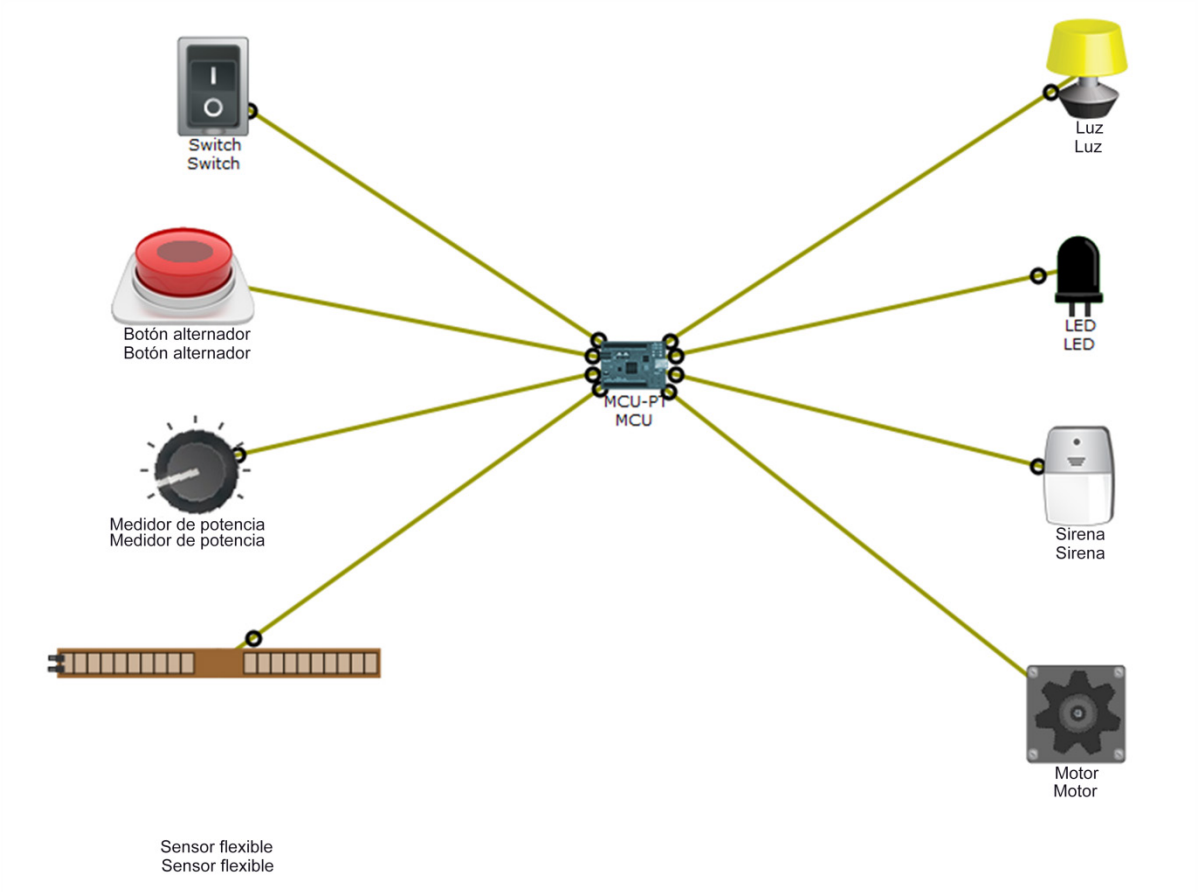
Packet Tracer: sensores y microcontrolador de PT

Topología



Objetivos

Parte 1: familiarícese con los sensores en PT 7.0

Parte 2: familiarícese con el microcontrolador de PT (PT-MCU) en PT 7.0

Parte 3: familiarícese con la programación del microcontrolador de PT (PT-MCU) en Packet Tracer 7.0

Aspectos básicos/situación

Una unidad de microcontrolador (MCU) es una pequeña computadora incorporada a un sistema en un chip (SoC). Contiene un núcleo de procesador, una memoria y unidades periféricas programables de entrada/salida. Los microcontroladores están diseñados para las aplicaciones incorporadas o las aplicaciones que requieren pocos recursos de computadora. Por otro lado, los microprocesadores utilizados en las computadoras personales por lo general se usan para admitir otras aplicaciones genéricas que requieren más recursos informáticos.

Ejemplos de aplicaciones que dependen de microcontroladores, como sistemas de control de motores automotrices, dispositivos médicos, controles remotos, máquinas de oficinas, electrodomésticos, herramientas eléctricas, juegos y otros sistemas integrados. Microcontroladores de señales combinadas, como componentes analógicos integrados necesarios para controlar los sistemas electrónicos no digitales.

Packet Tracer 7.0 brinda soporte al emulador de MCU. El usuario puede programar la MCU de PT para que realice tareas similares a las MCU del mundo real. Para simplificar el proceso, la MCU de PT puede programarse con Java y Python.

En esta actividad, lanzará MCUDEMO.PKT en Packet Tracer 7.0 y se familiarizará con el emulador de MCU de PT y su aspecto de programación.

Recursos necesarios

- 1 PC con Packet Tracer 7.0 instalado
- Archivo MCUDEMO.PKT

Parte 1: Introducción

La MCU de PT es una placa con un puerto USB, seis puertos de E/S digital y cuatro puertos de E/S analógica. Los puertos de E/S digital en la MCU de PT permiten que un usuario conecte accionadores y sensores digitales. Los puertos de E/S analógicos permiten que un usuario conecte accionadores y sensores analógicos. En esta demostración, hay una sola MCU de PT conectada a cuatro sensores y cuatro accionadores. Los cuatro sensores incluyen un interruptor digital, un botón alternador digital, un medidor de potencia analógico y un sensor flexible analógico. Los cuatro accionadores incluyen una luz, un LED, una sirena y un motor analógico. La MCU de PT está programada en Python para leer continuamente los valores del sensor y escribir los accionadores regidos por la lógica condicional. La relación entre los sensores y los accionadores puede resumirse de la siguiente manera:

- El interruptor controla la luz
- El pulsador controla el LED
- El medidor de potencia controla la sirena
- El sensor flexible controla el motor

Parte 2: Los sensores y la MCU de PT

a. Tómese un momento para analizar la topología. La **MCU** se coloca en el centro. Los dispositivos de entrada (sensores e interruptores) se colocan a la izquierda y los dispositivos de salida a la derecha.

b. Seleccione la **MCU**, los sensores, los interruptores y los dispositivos de control para abrir las ventanas de configuración. Tenga en cuenta que los diferentes dispositivos tienen distintas fichas.

   En la ventana **MCU**, la ficha **Programación** contiene el código en ejecución de Python. El código de Python define el comportamiento del dispositivo.

c. ALT + clic (mantenga presionada la tecla ALT mientras hace clic en el dispositivo) para interactuar con el dispositivo.

   ALT + clic en el **interruptor** para encender/apagar la **luz**.

   ALT + clic en el **pulsador** para encender/apagar el **LED**.

   ALT + clic (presione y arrastre el **medidor de potencia** para controlar el volumen de la **sirena**).

   ALT + clic (presione y arrastre el **sensor flexible** para controlar la velocidad del **motor**).

d. Anote a qué puertos de la **MCU** se conectan los sensores, los interruptores y los dispositivos.

Parte 3: Programación de la MCU

a. Abra la **MCU** ubicada en el centro de la topología.

b. Seleccione la ficha **Programación** para acceder al código en ejecución de Python en la **MCU**.

c. Observe el código e intente comprenderlo. A continuación hay un resumen de las tareas que realiza:

En las líneas 1 y 2, todas las clases en las bibliotecas **tiempo** y **gpio** se importan al programa. Esto es importante para brindar acceso a las funciones de tiempo y gpio.

Se declaran e inician cuatro variantes globales en las líneas 4, 5, 6 y 7. Estas variables se utilizarán para mantener los valores del sensor.

Se crea la función **readFromSensors()** (líneas 9 a 18). Primero, la función prepara las variables del sensor (líneas 10 a 13). La función **readFromSensors()** convoca a otras dos funciones, **digitalRead()** y **analogRead()**, para capturar el estado de los sensores y almacenarlo en las variables adecuadas (líneas 15 a 18). Tenga en cuenta que **digitalRead()** y **analogRead()** toman un número de pin como parámetro. Mediante la conexión de sensores específicos en los pines, el programa puede capturar el estado de los sensores específicos.

Se crea otra función en las líneas 20 a 39: **writeToActuators()** se usa para cambiar el estado de los accionadores en función del estado de los sensores. En las líneas 21 a 24, el programa prueba los contenidos de la variable **switchValue**. Debido a que **switchValue** almacena el estado del pin 0 (vea la línea 15), el programa puede decidir si enciende la luz; si el valor almacenado en **switchValue** es equivalente a ALTO (se aplica voltaje o el interruptor está ENCENDIDO), el programa enciende la luz escribiendo el valor 2 en el accionador 2. Por otro lado, si **switchValue** es equivalente a BAJO (no se aplica voltaje o el interruptor está APAGADO), el programa apaga la luz escribiendo 0 en el accionador 2.

Asimismo, las líneas 26 a 39 prueban y modifican otros accionadores en función de los sensores de control respectivos.

Las líneas 41 a 54 definen la función **main()**. Como lo indica su nombre, la función **main()** se ejecuta automáticamente cuando se enciende la MCU por primera vez. Las líneas 42 a 46 inician los pines; los pines 0 a 1 se configuran como ENTRADA (líneas 42 y 43) mientras que los pines 2, 3 y 4 se configuran como SALIDA. Esto es importante porque los pines de ENTRADA reciben voltaje y los pines de SALIDA emiten voltaje generado por la MCU.

Se crea un bucle **while** infinito en las líneas 48 a 51. Dado que la condición del bucle simplemente indica **true**, la MCU ejecutará las líneas 49, 50 y 51 continuamente. Este bucle **while** infinito fuerza la MCU:

1. Ejecuta la función **readFromSensors()** en la línea 49.
2. Ejecuta la función **writeToActuators()** en la línea 50.
3. Espere 1 segundo en la línea 50 (1000 ms = 1 segundo).
4. Reinice el bucle desde la parte superior regresando a la línea 49 y ejecutando la función **readFromSensors()** nuevamente.

Tenga en cuenta que, si bien los bucles infinitos no son deseados, son útiles en este programa; el bucle infinito garantiza que la **MCU** verifique constantemente el estado de los sensores y los interruptores (mediante la ejecución de **readFromSensors()** cada segundo) y tome siempre la medida adecuada para los accionadores (mediante la ejecución de **writeToActuators()**) en función del estado de los sensores.

d. Actualmente, la luz se controla mediante el interruptor y el LED se controla mediante el pulsador. Modifique el código para que el interruptor controle el LED y el pulsador controle la luz.

Parte 4: Reflexión

La introducción de la MCU de PT programable en Packet Tracer 7.0 permite un potente entorno de simulación de IdC. El uso de Python como lenguaje de programación además contribuye a una plataforma robusta.

a. DESAFÍO 1: transfiera el circuito 1 del kit básico de SparkFun, “Hacer parpadear el LED”, a Packet Tracer 7.0 con la MCU de PT como microcontrolador.

Sugerencia: deberá transferir el código presentado en el SIK a Python.

Además deberá modificar los pines utilizados en el SIK para adaptar el sistema numérico del pin de la MCU de PT.

b. DESAFÍO 2: con los conceptos presentados en el circuito 1 del **kit básico de SparkFun**, **Hacer parpadear el LED**, el circuito 4, **LED múltiples**, y el circuito 5, **Pulsador**, use Packet Tracer 7.0 o posterior para crear un circuito que ilumine uno de los ocho LED en secuencia, cada vez que se presiona el pulsador.

   Requisitos:

   Debe usar 8 LED alineados.

   Cada vez que se presiona el **pulsador**, el LED actualmente iluminado se oscurece y el próximo se ilumina.

   Solo debe haber un LED iluminado en un momento determinado.

   Sugerencia: use la MCU de PT como microcontrolador.