SISTEMAS DE INFORMACIÓN PARA LA INDUSTRIA

PRACTICA 1

IMPLANTACIÓN DE UNA RTU CON ARDUINO

POR:

JAVIER ANDRÉS JIMÉNEZ JIMÉNEZ

SERGIO CAÑETE LINARES

INDICE:

ANALISIS DEL PROBLEMA.

El objetivo de esta práctica es implantar una RTU basada en Arduino, haciendo uso del simulador TinkerCAD. Se nos imponen unos requisitos concretos, siendo el uso de un mínimo de componente específicos y algunos de nuestra elección.

Los componentes que tenemos que usar son: un Arduino, un sonar ya usado en ejercicios anteriores, un servomotor también usado ya (estos elementos de uso obligatorio); además de otro tipo de motor (nuestra elección) y algún otro tipo de sensor (nosotros hemos elegido un sensor de gas), además hemos introducido adicionalmente un buzzer para señales sonoras y un transistor para simular un control de potencia con relé.

El problema a resolver es la falta de seguridad frente a acumulación de gases o escapes de gas en una nave industrial, donde se trabaja con gases CO2 constantemente y se puede llegar a acumular a niveles tóxicos para la salud. Usaremos los siguientes elementos, que vienen descritos acontinuación.

Usaremos un sensor de gas que detecte por niveles cuando hay acumulación de gas, un servo motor que abrirá una escotilla 45º cuando el nivel de gas sea bajo, y a 90º cuando el nivel de gas sea medio.

En paralelo cuando el nivel de gas sea medio, un motor de CC (que simula un ventilador) entrará en funcionamiento cuando el nivel de gas sea alto. El control de este se hace con un transistor que BJT(NPN) que simula un relé que hace de controlador de potencia, ya que en un entorno industrial habrá un ventilador industrial que dependiendo de cómo sea de grande, su motor, trabajará con altos voltajes e intensidades.

Además, como medida extra cuando el sensor de proximidad detecte a alguien en la estancia se emitirá una señal sonora a través del buzzer que simula un altavoz a distintas frecuencias e intensidades dependiendo del nivel de gas. Y un led RGB que simularía luces de emergencia, que dependiendo del nivel de gas se encenderían en distintos colores dependiendo del nivel de gas: verde (no hay gas o nivel muy bajo y no peligroso), amarillo (nivel medio), rojo (nivel alto).

SOLUCIÓN PROPUESTA: CIRCUITO.

La solución propuesta es usar una placa Arduino hará de RTU interconectando los distintos elementos y dotándolos de funcionalidad dependiendo de la situación dada.

El sensor de gas será el eje central y el que provocará los disparos para la puesta en funcionamiento de los demás elementos. Este cuando detecta gas, genera un pulso analógico en el pin A2 que es recogido por el Arduino en el puerto analógico A1 en nuestro caso (el puerto puede ser cualquiera de los analógicos a conveniencia).

La conexión del sensor se realiza conectando los pines B1, H2 y B2 a VCC(5v), el pin H1 a tierra directo y el pin A1 a tierra en serie con una resistencia de 4,7K.

Hemos buscado en varios sitios, pero no he logrado encontrar el datasheet del sensor ni algún esquema de conexiones de este, por lo que las conexiones la hemos hecho así fijándonos en varios ejemplos que hay por la red, pero en ninguno explican por qué se conecta de esta forma.

Luego tenemos el motor de CC que está controlado con un transistor BJT(NPN) que simula un relé. El motor tiene conectado el emisor del transistor a su patilla de VCC, donde recibirá una alimentación mayor (Desde el colector conectado la pila de 9v) y la otra patilla a GND.

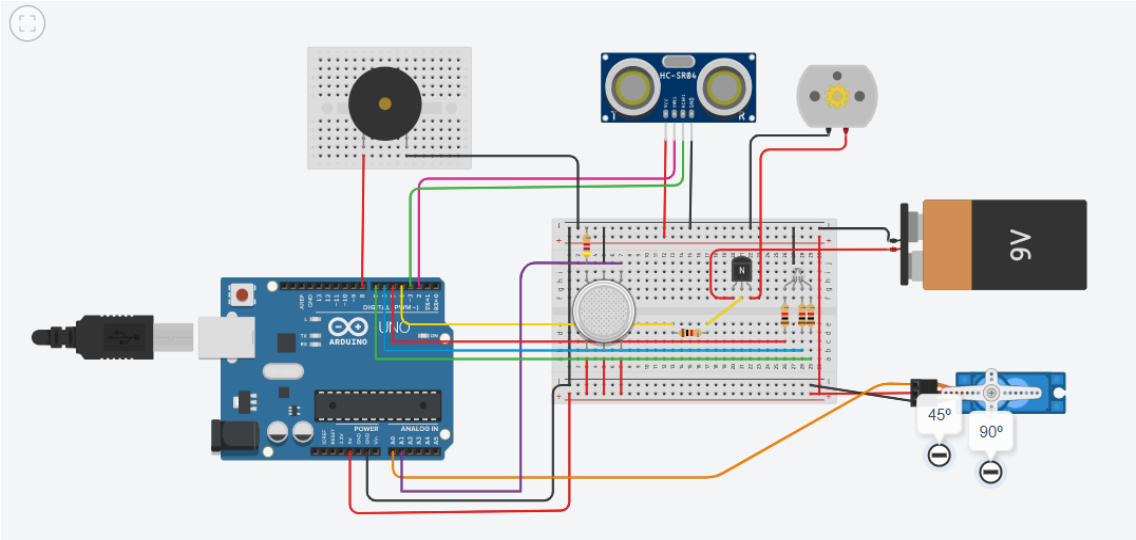
El transistor tiene conectado el colector a una pila de 9v (proporciona un mayor voltaje que el Arduino para simular una fuente de energía más potente). En la base del transistor tenemos una resistencia 1k que va conectada en serie e la base y al pin 4 del Arduino que activara o desactivara el motor. Y el emisor que estará controlado por la base, va conectado al pin VCC del motor para así poder controlarlo con el transistor.

El servo motor para controlar la apertura de una escotilla de ventilación va conectado el pin de tierra a GND (negro o marrón), el de potencia a VCC (rojo) y el de señal(naranja) va al pin A0 configurado como salida analógica del Arduino.

Por último, tenemos el buzzer conectado al pin 8 y a tierra que junto con el sensor de proximidad componen el módulo de aviso acústico en caso de que se detecte alguna presencia.

El sensor de distancia va conectado a VCC y GND por los cables rojo y negro respectivamente, luego el pin echo al pin 3 del A que será la entrada que determinará la distancia y la presencia. Y el pin TRIG al pin 2 del Arduino que será una salida para controlar el sensor, ya que este pin es el de disparo al que se le aplica un pulso de 10us para comenzar con la medición.

DISEÑO DEL CIRCUITO.



SOLUCIÓN PROPUESTA: CÓDIGO.

Para el desarrollo del código se han llevado a cabo una serie de prácticas, que aún no estén relacionadas directamente con este problema, son muy recomendables:

Sustituir int por byte cuando sea posible.

Es muy importante tener en cuenta las capacidades del Arduino Uno, una de ellas muy importante es el espacio disponible para almacenar variables, la SRAM, ya que únicamente dispone de 2kbytes.

Existen casos en los que simplemente vamos a usar una variable numérica para establecer los pines de E/S que vamos a utilizar, por lo tanto, resulta mucho más óptimo pensar en variables de tipo byte, de 1 byte, que variables de tipo int, de 2 bytes.

Evitar delay() cuando sea posible.

Para desarrollar programas de "juguete" o para otros programas donde el Arduino no vaya a desempeñar un papel de cierta importancia, delay() es usado por todos para introducir retardos/esperas. Sin embargo, en sistemas donde nuestro microcontrolador juega un papel más importante, el uso de delay() nos introduce el siguiente problema:

El sistema dejará de funcionar completamente y se mantendrá "congelado" hasta que el delay() termine.

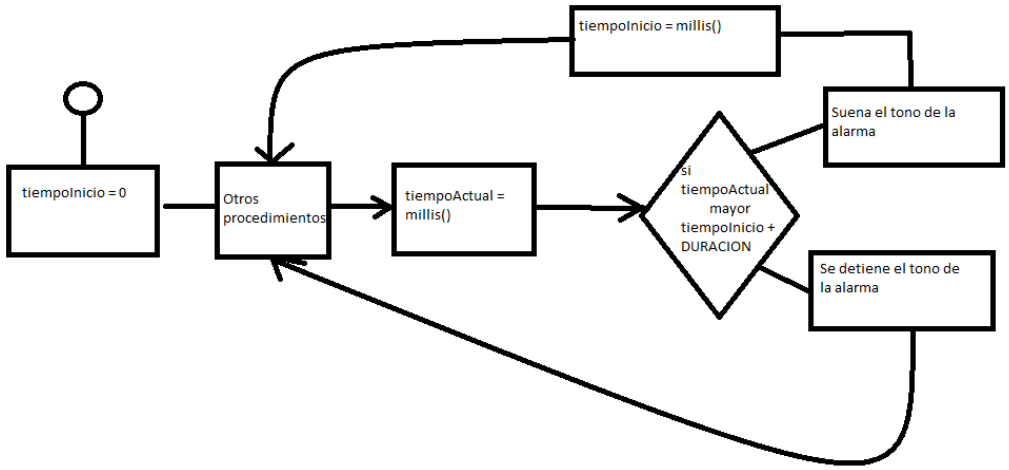
Es por eso que en este caso no es conveniente, ya que, si hacemos sonar una alarma de peligro, estableciendo un delay de 5000ms, no se volverá a comprobar ningún otro parámetro del sistema en ese tiempo, pudiendo ocurrir alguna desgracia.

Para solventar este problema, **introducimos millis()**.

Millis() no es más que una función que devuelve el tiempo en microsegundos que tiene de vida la ejecución del programa en el instante en el que se ejecuta dicha función.

Siguiendo el mismo problema de la alarma de peligro y el delay():

Esta vez, podríamos solucionar el problema con 2 variables usando esta función y un if.



La variable DURACION no es más que un parámetro que indica el tiempo de duración del tono de la alarma.

De esta manera, como nos encontramos en el loop(), en cada iteración se comprobará si ha pasado el tiempo DURACION para modificar o no, el tono de la alarma.

Para calcular la distancia de un objeto al sonar se ha utilizado una función extraída de (3).

Para controlar el servo se ha importado la librería Servo.h (4), que introduce ciertos elementos de POO y facilita su manejo.

Todas las variables han sido parametrizadas para facilitar las modificaciones en el código. Algunos valores se han tomado arbitrariamente o en función del simulador, por lo que pueden no corresponder con la realidad.

ACCESO A TINKERCAD.

https://www.tinkercad.com/things/3on4rmxS0Zg

BIBLIOGRAFÍA.

1. https://aprendiendoarduino.wordpress.com/2016/11/08/memoria-arduino/
2. https://hetpro-store.com/TUTORIALES/arduino-millis/
3. https://www.hwlibre.com/hc-sr04/#Pinout_y_datasheets
4. https://manueldelgadocrespo.blogspot.com/p/bibilioteca-servo.html
5. https://www.instructables.com/Arduino-Gas-Detecting-Alarm-System/
6. https://aprendiendoarduino.wordpress.com/2017/09/05/puertos-analogicos-arduino-avanzado/
7. https://www.arduino.cc/reference/en/language/functions/analog-io/analogwrite/
8. https://www.tinkercad.com/things/bXDdGggOCcp-sensor-de-gas
9. https://www.youtube.com/watch?v=jsB8-E2KppI
10. https://www.youtube.com/watch?v=uaro5WvJowY
11. https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf
12. https://leantec.es/wp-content/uploads/2019/06/Leantec.ES-HC-SR04.pdf