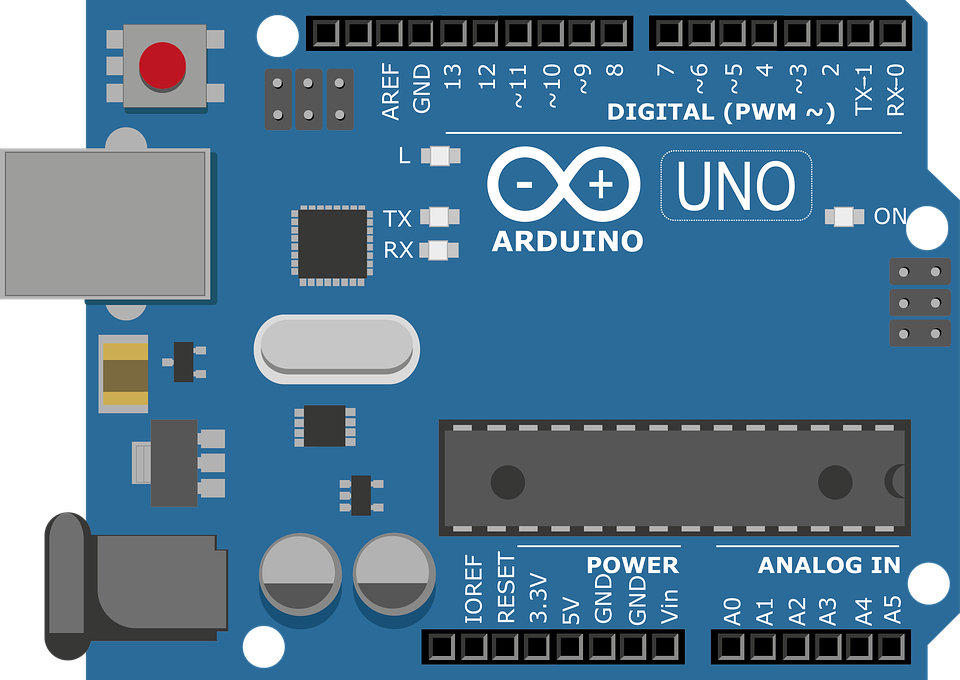
REPORTES INICIALES SOBRE EL ANALISIS DE APLICACIONES EN ARDUINO

Amador Sánchez Hernández

Andree Miguel Luna Solano



Fecha de Entrega: jueves 22 de agosto de 2019

**1.-Temporizador Digital (Digital Timer)**

**Objetivos:**

En esta ocasión implementamos un temporizador apoyado de un display múltiple de 7 segmentos, un circuito integrado (74HC595), una placa arduino uno, y demás componentes. Este dispositivo tiene varias aplicaciones, desde la cocina (calcular el tiempo de horneado o de algún otro proceso) en laboratorios (al momento de requerir medir un tiempo específico para algún experimento) entre otras más, y cuenta con la ventaja de no tener que estar supervisándolo visualmente todo el tiempo, ya que incorpora una alarma sonora (Buzzer) para cuando termine la cuenta regresiva programada por el usuario.

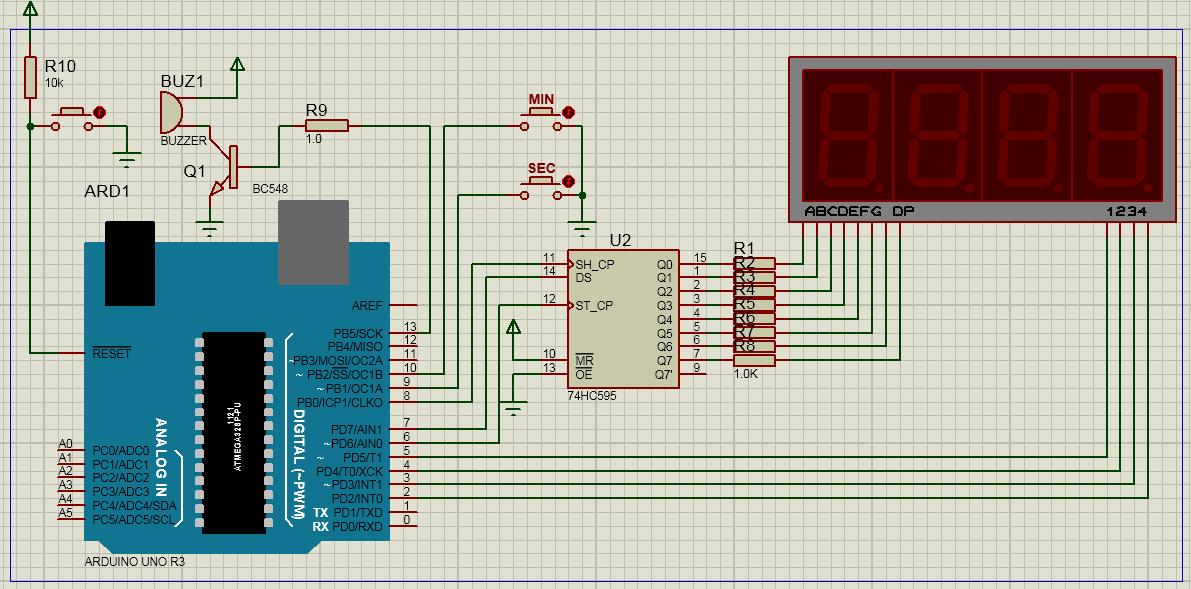
Como entradas, tendremos dos pulsadores, de los cuales, uno sirve para agregar minutos, y el otro para agregar segundos, haciendo de esta manera más precisa la forma en la que configuramos nuestro temporizador (no siempre requeriremos minutos exactos). Y nuestras salidas serán los dígitos mostrados en nuestro display.

El proceso consta de lograr una cuenta regresiva de nuestros dígitos en tiempo real el cual se logra gracias al código de nuestro arduino. Pero para explicarlo de manera más detallada lo veremos de la siguiente manera; al momento de encender en el display se mostrarán los 4 dígitos en “0” ya que esa es la manera de inicializarlo, después, al momento de presionar cualquiera de los dos pulsadores, este iniciara con su tarea principal, el cual es iniciar la cuenta regresiva, tomando como punto de partida el número de minutos/segundos agregados (Cabe mencionar que los dos primeros dígitos son para minutos y los últimos dos para segundos, dando a entender que nuestro dispositivo solo puede hacer una cuenta regresiva de una hora máximo.) Al momento de terminar la cuenta regresiva, la placa Arduino mandara una señal para encender el Buzzer, y de esta manera avisarnos que ha acabado.

Por último, pero no menos importante, cabe mencionar que, aunque parece un proceso sencillo, tiene un trabajo con cierto nivel de complejidad detrás, ya que el multiplexado en displays de este tipo suele ser una tarea compleja, ya que se utilizó un circuito integrado (75HC595) para lograr tener las conexiones necesarias en el arduino. Sin duda a pesar de ser sencillo contiene elementos que con practica y concentración son muy útiles de abordar.

**Materiales:**

* Placa Arduino UNO
* Display múltiple de 7 segmentos
* Buzzer
* 1 Circuito integrado 75HC595
* 9 Resistencias de 1K
* 3 Pulsadores
* 1 Transistor BC548

**Diagrama Eléctrico::**

**Codigo:**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Temporizador Digital. Usa 4 dígitos de Siete Segmentos de cátodo común

// Tiene controles para Minutos y Segundos

//==============================================================================

#include <TimerOne.h>

const int Data = 7; // Conexiones entre el CI 74HC595 y la placa Arduino

const int Clock = 8; //

const int Latch = 6; //

const int SEG0 = 5; // Conexiones para manejar los cátodos

const int SEG1 = 4; //

const int SEG2 = 3; //

const int SEG3 = 2; //

const int Buzzer = 13; // Bocina o Buzzer

const int Min\_key = 10; // Teclas

const int Sec\_key = 9; //

//------------------------------------------------------------------------------

//-----[ VARIABLES ]-----

//------------------------------------------------------------------------------

int cc = 0;

char Value[4];

// Patrones para desplegar los dígitos (verificarlos):

const char SegData[] = {0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F};

int MIN = 0, SEC = 0, count = 10; // Variables y valores iniciales

//==============================================================================

//-----[ AJUSTES GENERALES ]-----

//==============================================================================

void setup() {

Serial.begin(9600); // Comunicación serial

pinMode(Data, OUTPUT); // Terminales para manejar arreglo 7 segmentos

pinMode(Clock, OUTPUT); //

pinMode(Latch, OUTPUT); //

pinMode(SEG0, OUTPUT); //

pinMode(SEG1, OUTPUT); //

pinMode(SEG2, OUTPUT); //

pinMode(SEG3, OUTPUT); //

pinMode(Buzzer, OUTPUT); //

digitalWrite(Buzzer, LOW); // Apagado de la bocina o buzzer

pinMode(Min\_key, INPUT); // Tecla de Minutos

pinMode(Sec\_key, INPUT); // Tecla de Segundos

digitalWrite(Min\_key, HIGH); // Ajuste de "Pull-Ups" para las entradas

digitalWrite(Sec\_key, HIGH); //

cc = 0;

// Lo siguiente es un recurso proporcionado por una biblioteca: TimerOne

// Ajuste de un temporizador de 100,000 microsegundos (o de 0.1 segundo,

// o de 10 Hertz) => Un LED parpadearía 5 veces por segundo

Timer1.initialize(10000);

Timer1.attachInterrupt( timerIsr ); // Subrutina que atiende la Interrupción

}

//==============================================================================

//-----[ LAZO PRINCIPAL ]-----

//==============================================================================

void loop() {

char cMIN[4], cSEC[4];

if (count == 0) {

count = 10; // Esto es 1 segundo. 10 retardos de 100 milisegundos

//

if (MIN > 0) { // Si minutos son mayores a cero...

if (SEC == 0) { // Si se llegó al segundo "00"

MIN--;

SEC = 60;

}

}

SEC--; // Decremento de Segundos

// Revisar que el Temporizador está en cero

if (MIN == 0 && SEC == 1) { // Segundos mantenidos para evitar sonido

digitalWrite(Buzzer, HIGH); // Encendido del sonido

}

}

//----------------------------------------

if (MIN > 0 || SEC > 0) { // Si los Minutos y Segundos son mayores a "0"

count--; //

}

//----------------------------------------

if (digitalRead(Min\_key) == LOW) { // ¿Se oprimió la tecla "Minutos"?

delay(10); // Para evitar "rebotes"

MIN++; // Incrementa minutos

digitalWrite(Buzzer, LOW); // Apagar bocina

}

//----------------------------------------

if (digitalRead(Sec\_key) == LOW) { // ¿Se oprimió la tecla de "Segundos"?

delay(10); // Para evitar "rebotes"

SEC++; // Incrementar segundos

if (SEC > 59) { // 60 segundos - 1 minuto

MIN++; // Ajuste...

SEC = 0; // ...para reciclar

}

digitalWrite(Buzzer, LOW); // Apagar bocina o buzzer

}

//----------------------------------------

// Si se oprimieran simultáneamente ambas teclas, se Limpia el Timer

if (digitalRead(Sec\_key) == LOW && digitalRead(Min\_key) == LOW) {

MIN = 0;

SEC = 0;

digitalWrite(Buzzer, HIGH); // Prender bocina o buzzer

delay(500);

digitalWrite(Buzzer, LOW);

}

//----------------------------------------

// Muestra de conteo en el desplegador de 7 segmentos

sprintf(cMIN, "%02d", MIN); // Arreglo en ASCII

sprintf(cSEC, "%02d", SEC); // Arreglo en ASCII

Serial.println(cMIN); // Imprimir el conteo en terminal serial

// (depuración)

Value[0] = cMIN[0] & 0x0F; // Remover el "nibble" superior (AND con 0x0F)

Value[1] = cMIN[1] & 0x0F; // Por ejemplo, de 0x32 (ASCII)...

Value[2] = cSEC[0] & 0x0F; // ...sólo necesitamos el "2"

Value[3] = cSEC[1] & 0x0F;

delay(100);

}

//==============================================================================

//-----[ SUBRUTINAS ]-----

//==============================================================================

// INTERRUPCIÓN CAUSADA POR EL DESBORDAMIENTO DE TIMER 1

// (Para el desplegador de 7 segmentos)

//==============================================================================

void timerIsr() {

cc++; // Incremento del contador de "cátodos", asignados a dígitos

if (cc == 5) { // De momento sólo se tienen 4 dígitos

cc = 1;

}

Scanner(); // Realiza el rastreo

TCNT0 = 0xCC; // Reinicia "Timer 1" para ocurrencia próxima Interrupción

}

//==============================================================================

// GENERACIÓN DE LOS DÍGITOS

//==============================================================================

void DisplayDigit(char d) {

int i;

for (i = 0; i < 8; i++) { // Corrimiento, bit a bit, en...

if ((d & 0x80) == 0x80) { // ... registro de corrimiento

digitalWrite(Data, HIGH);

}

else {

digitalWrite(Data, LOW);

}

d = d << 1; // Corrimiento...

// Pulsos de Reloj

digitalWrite(Clock, LOW);

digitalWrite(Clock, HIGH);

}

// Captura del dato

digitalWrite(Latch, LOW);

digitalWrite(Latch, HIGH);

}

//==============================================================================

// FUNCIÓN DE RASTREO DEL DESPLEGADOR

//==============================================================================

void Scanner() {

switch (cc) // Selección del dígito que se actualiza

{

case 1:

digitalWrite(SEG3, HIGH);

DisplayDigit(SegData[Value[0]]);

digitalWrite(SEG0, LOW);

break;

case 2:

digitalWrite(SEG0, HIGH);

DisplayDigit(SegData[Value[1]] | 0x80); // Punto Decimal

digitalWrite(SEG1, LOW);

break;

case 3:

digitalWrite(SEG1, HIGH);

DisplayDigit(SegData[Value[2]]);

digitalWrite(SEG2, LOW);

break;

case 4:

digitalWrite(SEG2, HIGH);

DisplayDigit(SegData[Value[3]]);

digitalWrite(SEG3, LOW);

break;

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Conclusión:**

El llevar a cabo una tarea tan sencilla como la de hacer una cuenta regresiva con cadencia de tiempo real y mostrarla es un trabajo arduo. Gracias a las herramientas y a la tecnología que hoy en día tenemos es posible esto y muchas cosas más, además de no solo quedarse así, sino que también podemos agregarle muchas cosas más al dispositivo. Como valores predeterminados de tiempo, indicadores led, u alguna otra opción adicional.

**2.- Medidor de RPM (RPM meter)**

**Objetivo:**

En este circuito lo que se implementa es un sistema en el cual se lee un medidor de vueltas por segundo.

Se implementa con un arduino, un registro de desplazamiento y un display.

En este caso se utiliza un generador de reloj para simular un objeto en movimiento. Esta información ira a la entrada digital, la cual registrara el valor que reciba, en el mismo código de arduino recibirá el valor, lo almacenara para su uso posterior. Se ajustará el tiempo de interrupciones. Siguiente donde calcularemos las revoluciones por segundo. esto haciendo que tenga interrupciones y evitar errores. Una vez llegamos a la generación de datos Se irá cambiando la información en el bit de registro, una vez dado esto se dará la información del pulso de reloj. Se establecerá lo que se mostrara en el display el cual recibirá la información del 74HC595 y se mostrara en el display.

**Lista de partes:**

\*Microcontrolador

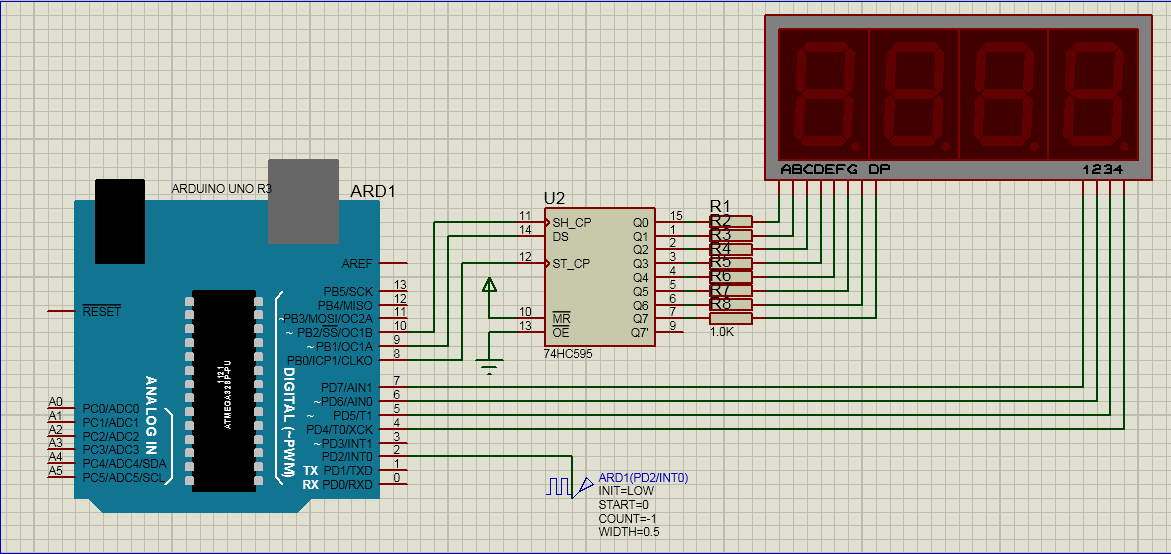
\*74HC595

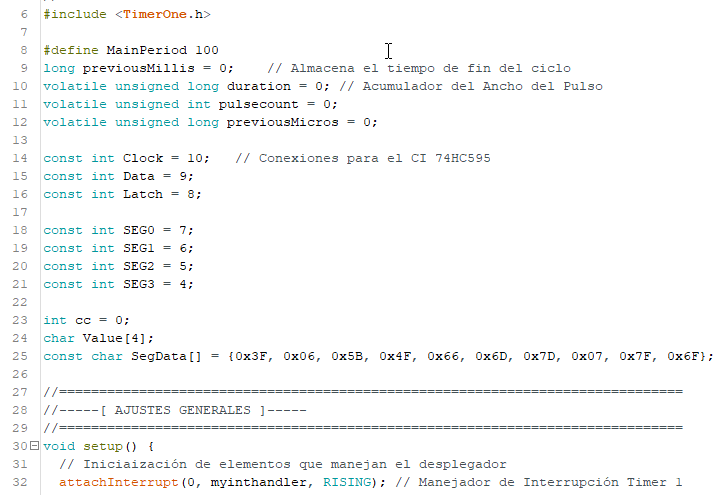
\*Resistencias

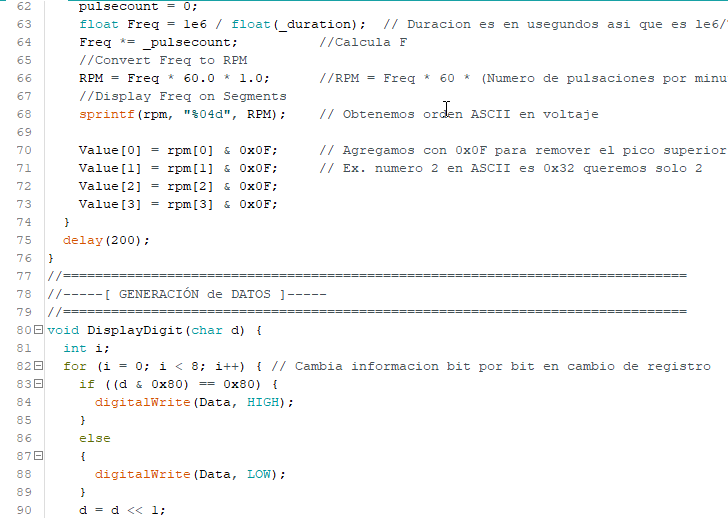
\*Display de 8 segmentos

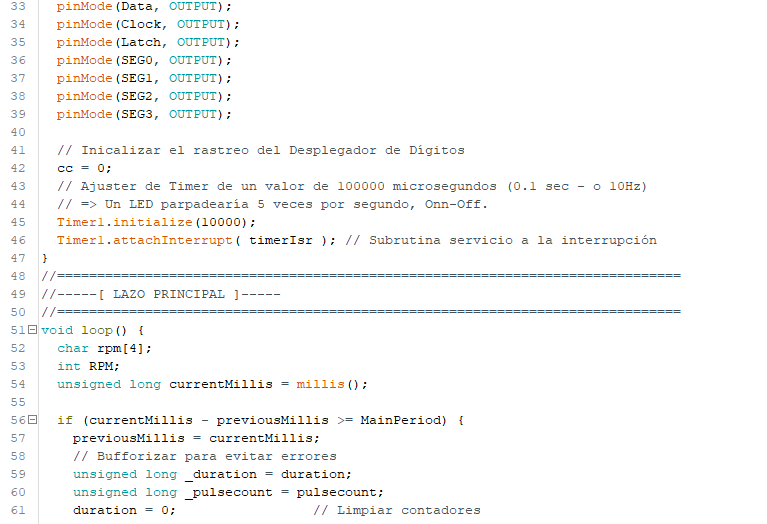
\*Señal

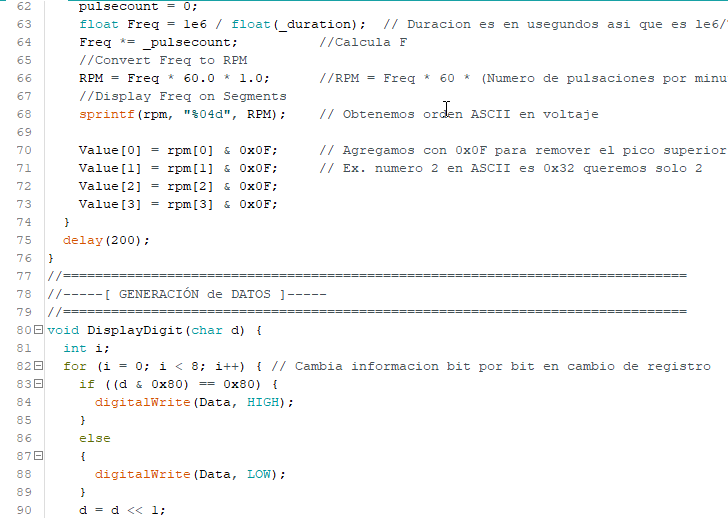
**Diagrama eléctrico:**

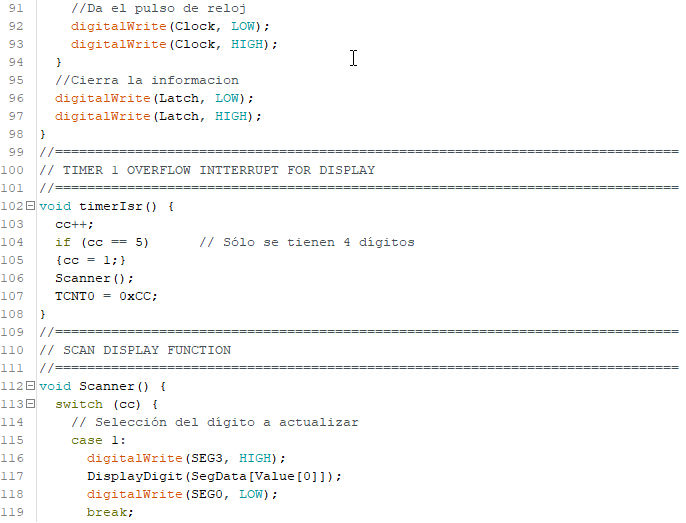
****

**Código:**











**Conclusiones:**

Como hemos mencionado antes, el tener un conocimiento exacto de lo que estamos visualizando en pantalla siempre ha sido de mucha utilidad si lo que tenemos es un sistema que tiene que estar bajo control en todo sentido. El medir las revoluciones por minuto solo con mirar un motor girar no sería posible, ya que estas suelen ser muy rápidas para el ojo humano, sin embargo, apoyándonos de herramientas como esta, es posible.

**3.- Light Controller:**

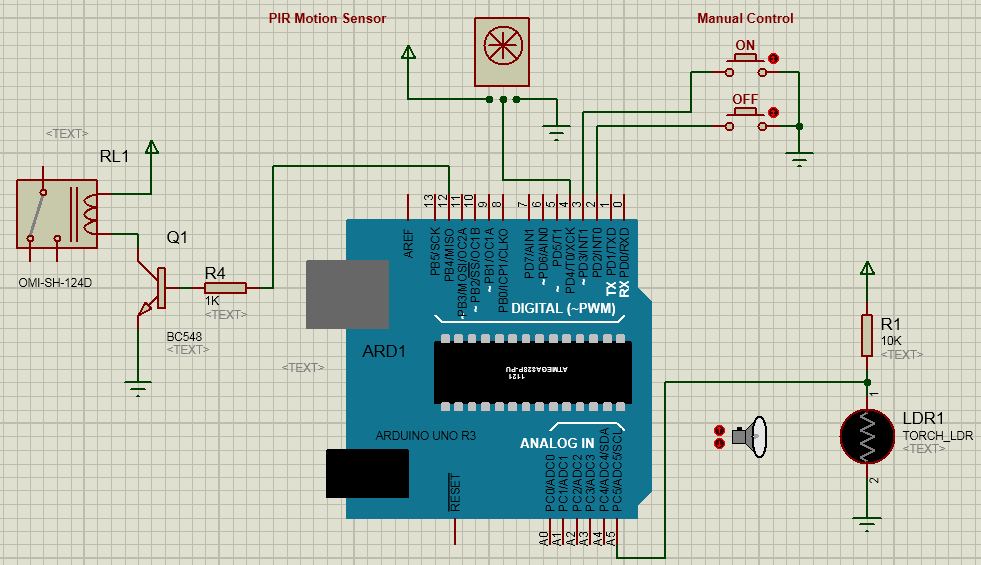
**Objetivos:**

Se realiza un circuito de control de luz. Se tiene 2 pulsos de ajuste manual el cual ayudara a ir ajustando. 2 sensores y un relay. Se busca que a medida de movimiento del motor se vaya ajustando la luz que se produce. Una vez conectado los sensores, teclas y el relé en sus puertos, comenzara a iniciar. recibirá el primer valor una vez el sensor detecte movimiento y la intensidad de luz sea baja. Se tendrá un tiempo de registro de 10 segundos mientras que se detectara el movimiento en 1.5 seg., si la intensidad es menor al límite inferior dado, se encenderá la iluminación. el usuario podrá ajustar la iluminación de este circuito activándolo manualmente.

**Lista de partes:**

* Placa Arduino
* LDR
* Sensor de movimiento
* 2 pulsadores
* 1 relevador (OMI-SH-124D)
* 2 Resistencias de 1K
* 1 Transistor BC548

**Circuito Eléctrico:**

****

**Código:**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Controlador de Iluminación.

// Control por nivel de iluminación (día - noche) y sensado de presencia.

//------------------------------------------------------------------------------

const int RELAY = 12; // Relevdor o Motor

// Teclas asociadas

const int on\_key = 3;

const int off\_key = 2;

int counter = 0, manual = 0;

// Conexiones de los sensores

const int LDR = A5;

const int PIR = 4;

//-----[ AJUSTES GENERALES ]-----

void setup() {

pinMode(RELAY, OUTPUT);

pinMode(on\_key, INPUT);

pinMode(off\_key, INPUT);

pinMode(PIR, INPUT);

// Ajuste de "Pull up" para teclas de ajuste

digitalWrite(on\_key, HIGH);

digitalWrite(off\_key, HIGH);

digitalWrite(PIR, HIGH);

digitalWrite(RELAY, LOW); // Apagado del Motor o Relevador

}

//-----[ LAZO PRINCIPAL ]-----

void loop() {

// Enciende las luces si se detecta movimiento y la intensidad de iluminación es baja

if (digitalRead(PIR) == HIGH) {

counter = 1000; // Ajusta el tiempo para 10 segundos

if (counter > 15) // Movimiento detectado por 1.5 segundos

{

if (analogRead(LDR) > 512) // La intensidad de la luz es baja

{

digitalWrite(RELAY, HIGH); // Enciende las iluminación

}

}

}

counter--;

if (counter == 0) {

if (manual == 0) // Revisa esta opción.

{

digitalWrite(RELAY, LOW);

}

}

// Obtención del punto de ajuste del usuario

if (digitalRead(on\_key) == LOW)

{

digitalWrite(RELAY, HIGH); // Encendido de las luces

manual = 1; // Encendido manual

}

if (digitalRead(off\_key) == LOW)

{

digitalWrite(RELAY, LOW); // Apagado de las luces

manual = 0;

}

delay(10); // ACtualización cada 10 milisegundos

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Conclusiones:**

Los sistemas automatizados nos hacen la vida más fácil, ya que no hay necesidad de estar supervisando algún proceso para lograr hacerlo manualmente, si no que si ya existe una herramienta que este controlando una acción (como la intensidad de la luz, por ejemplo) no es necesaria la intervención humana, a menos que se le requiera.

Sin duda, aunque sea un pequeño ejemplo, nos muestra como un proceso automatizado puede ayudarnos a ahorrar tiempo y recursos en una acción cotidiana, haciendo así un proceso más ameno, desde algo local, a algo industrial.