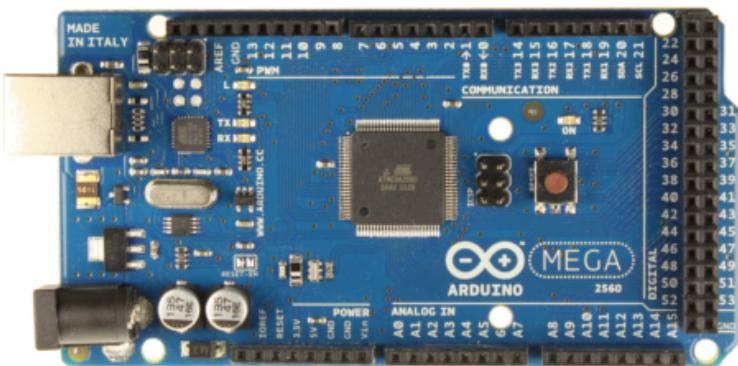


GUIA – MODIFICACION DE MAQUINA TRAGAMONEDAS A MAQUINA DE FERIA DE \$1



PARTE 000: Preparación



Este tutorial será el [Arduino Mega](#).

El [Arduino Mega](#) es probablemente el micro controlador más capaz de la familia Arduino. Posee 54 pines digitales que funcionan como entrada/salida; 16 entradas análogas, un cristal oscilador de 16 MHz, una conexión USB, un botón de [reset](#) y una entrada para la alimentación de la placa. La comunicación entre la computadora y [arduino](#) se produce a través del puerto serie, sin embargo posee un convertidor usb-serie, por lo que sólo se necesita conectar el dispositivo a la computadora utilizando un cable USB como el que utilizan las impresoras.

Arduino Mega posee las siguientes especificaciones:

Micro-controlador: ATmega2560

Corriente DC por cada Pin Entrada/Salida: 40 mA

Voltaje Operativo: 5V

Corriente DC entregada en el Pin 3.3V: 50 mA

Voltaje de Entrada: 7-12V

Memoria Flash: 256 KB

(8KB usados por el bootloader)

Voltaje de Entrada(límites): 6-20V

SRAM: 8KB

Pines digitales de Entrada/Salida: 54 (de los cuales 15 proveen salida PWM)

EEPROM: 4KB

Pines análogos de entrada: 16

Clock Speed: 16 MHz

Alimentación

Arduino Mega puede ser alimentado mediante el puerto USB o con una fuente externa de poder. La alimentación es seleccionada de manera automática.

Cuando se trabaja con una fuente externa de poder se debe utilizar un convertidor AC/DC y regular dicho voltaje en el rango operativo de la placa. De igual manera se puede alimentar el micro mediante el uso de baterías. Preferiblemente el voltaje debe estar en el rango de los 7V hasta los 12V.

Arduino Mega posee algunos pines para la alimentación del circuito aparte del adaptador para la alimentación:

VIN: A través de este pin es posible proporcionar alimentación a la placa.

5V: Podemos obtener un voltaje de 5V y una corriente de 40mA desde este pin.

3.3V: Podemos obtener un voltaje de 3.3V y una corriente de 50mA desde este pin.

GND: El ground (0V) de la placa.

¿Qué es Arduino?

>[Arduino](#), es el nombre de la compañía que creo estas cosas. [Arduino](#) es un nombre Italiano masculino que significa “Amigo Fuerte”. Pero en si la cosa es un micro controlador basado en una familia de micro controladores de PIC. Tienen un pequeño programa pre cargado que hace posible programarlas desde una computadora, en un lenguaje como C++.

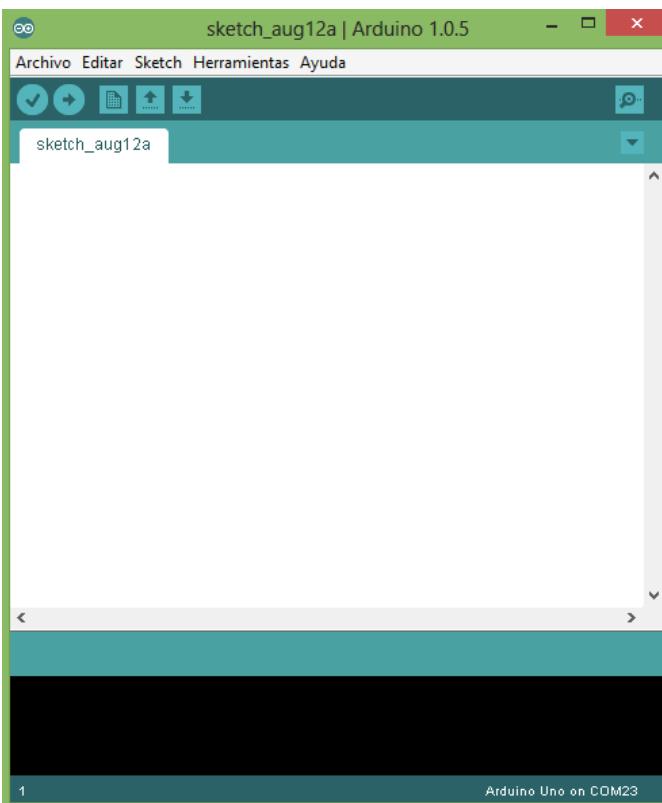
Hay diferentes versiones de este micro controlador. La versión que usaremos en este

Arduino IDE

Arduino puede ser programado de una manera muy fácil utilizando el lenguaje propio de Arduino junto con la interfaz Arduino IDE. Para poder subir el código con el que programaremos nuestro micro-controlador Arduino, se requiere descargar Arduino IDE. Liga para descarga: <http://arduino.cc/en/Main/Software>. Una vez descargado, descomprimimos el contenido del archivo y encontramos una carpeta donde se encuentran todos los archivos necesarios para que Arduino interactúe con el computador.

Recomiendo copiar todo el contenido del archivo descomprimido en una carpeta especial en la ruta C:/Arduino/.

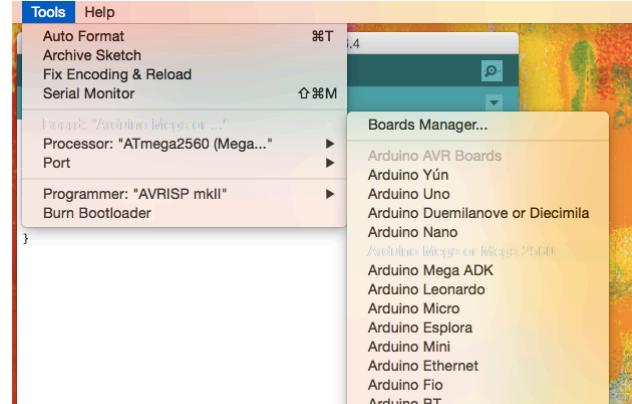
Si ejecutamos el archivo Arduino.exe, nos aparecerá la siguiente ventana:



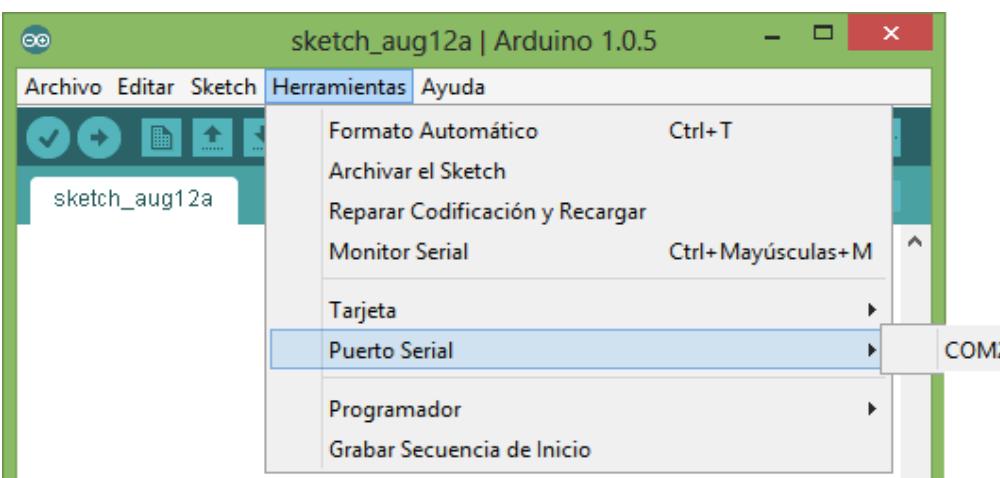
Conectamos nuestra placa Arduino al puerto USB. Si los drivers están en orden no habrá problemas con esto.

Ahora vamos al menú Herramientas/Tarjeta y nos aparecerá toda la gama de productos Arduino. El usuario debe escoger el modelo que posee.

Selecciona el que dice **Arduino ATmega2560**



En el mismo menú de herramientas escogemos el puerto serie en el cual se encuentra el micro conectado. Para que esta opción se habilite es necesario que la placa esté conectada mediante USB y que los drivers funcionen correctamente.



Con estas configuraciones es posible subir código desde Arduino IDE hasta el micro-controlador.

```
sketch_nov26a | Arduino 1.6.4
sketch_nov26a
void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

Cuando programemos, el primer ícono de la izquierda a derecha (la palomita) se usa para comprobar si la estructura del programa que hemos hecho es la correcta.

El botón con el ícono → sirve para subir el programa del IDE hasta el micro-controlador. El tercer ícono es para crear un nuevo archivo, el cuarto es para abrir un archivo, el quinto es para guardar un archivo, y el último (la lupa) es para acceder a la consola.

Al crear un nuevo archivo este viene con 2 funciones por default.

“void setup(){...}”

Esta función es invocada al inicio, se usa para inicializar variables, establecer los modos de los pins, el uso de alguna librería, etc. Esta función solo corre una vez, al inicio, o después de que se reseteo el Arduino.

“void loop(){...}”

Después de crear una función `setup ()`, que inicializa y establece los valores iniciales, la función `loop ()` hace exactamente lo que su nombre indica, bucles de forma consecutiva, lo que permite a el programa cambiar y responder. Se usa para controlar activamente la placa Arduino.

Una ultima cosa antes de empezar



Serial Monitor

Abre la ventana del “monitor de serie” e inicia el intercambio de datos con cualquier placa conectada en el puerto seleccionado. Esto por lo general restablece el tablero, si la placa es compatible Restablecer sobre la apertura del puerto serie.

Utilizaremos esta herramienta para comunicarnos con la placa Arduino Mega directamente en algunas partes donde correremos pruebas individuales.

El **Arduino Mega** tiene tres puertos seriales adicionales: `Serial_1` en los pines 19 (RX) y 18 (TX), `Serial_2` en los pines 17 (RX) y 16 (TX), `Serial_3` en los pines 15 (RX) y 14 (TX). Para utilizar estos pines para comunicarse con el ordenador personal, usted necesitará un adaptador adicional de USB a serie, ya que no están conectados al adaptador de los Mega-USB a serie. Para utilizarlos para comunicarse con un dispositivo serie TTL externa, conecte el pin TX al pin RX del dispositivo, el RX a TX pin de su dispositivo, y el suelo de su Mega a tierra del dispositivo.

Material/requisitos a usar que no vienen incluido en la Maquina Tragamonedas

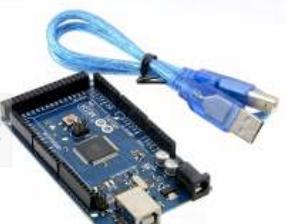
- + Arduino IDE: <http://arduino.cc/en/Main/Software>
- + Conocimientos básicos de programación
- + Monedero Electrónico de Múltiples Monedas
- + Arduino MEGA
- + Relevador de 5v
- + Lector de Billetes NV10USB
- + Fuente 12v 3A para Lector de Billetes NV10USB
- + Cable (1-2 mts)



Fuente Wei-ya Para Maquinas Vending

\$ 234⁹⁹
12 meses de \$ 23²²

Artículo nuevo - 2 vendidos
Distrito Federal



Arduino Mega 2560

\$ 250⁰⁰
12 MSI de \$ 20⁸³

Artículo nuevo - 66 vendidos
Sinaloa



Modulo Relevador Relé Para Arduino

\$ 30⁰⁰
12 MSI de \$ 2⁵⁰

Artículo nuevo
Distrito Federal



Hopper O Despachador De Plástico Para Perla, Vending O Genie

\$ 309⁹⁹
12 meses de \$ 30⁶²

Artículo nuevo
Distrito Federal



Resbaladilla De Plastico Para Maquinas Vending

\$ 49⁹⁹
12 meses de \$ 4⁹⁴

Artículo nuevo
Distrito Federal



Charola De Plastico Para Maquinas Vending

\$ 49⁹⁹
12 meses de \$ 4⁹⁴

Artículo nuevo - 1 vendido
Distrito Federal



Cono De Plastico Para Maquinas Vending O Perla De Oriente

\$ 49⁹⁹
12 meses de \$ 4⁹⁴

Artículo nuevo
Distrito Federal



Lector De Billetes Nv10 Maquinas De Casino, Gamers, Vending.

\$ 2,850⁰⁰
12 meses de \$ 281⁵⁶

Artículo nuevo - 2 vendidos
Estado De México



Monedero Electronico Multimoneda -el Mejor Al Mejor Precio-

\$ 640⁰⁰
12 meses de \$ 63²³

Artículo nuevo - 3 vendidos
Veracruz



Porta Candados Generico Para Todo Tipo De Maquinas

\$ 24⁹⁹
12 meses de \$ 2⁴⁷

Artículo nuevo
Distrito Federal



Mueble Para Maquinas Vending Tipo K

\$ 1,150⁰⁰
12 meses de \$ 113⁶¹

Artículo nuevo
Distrito Federal



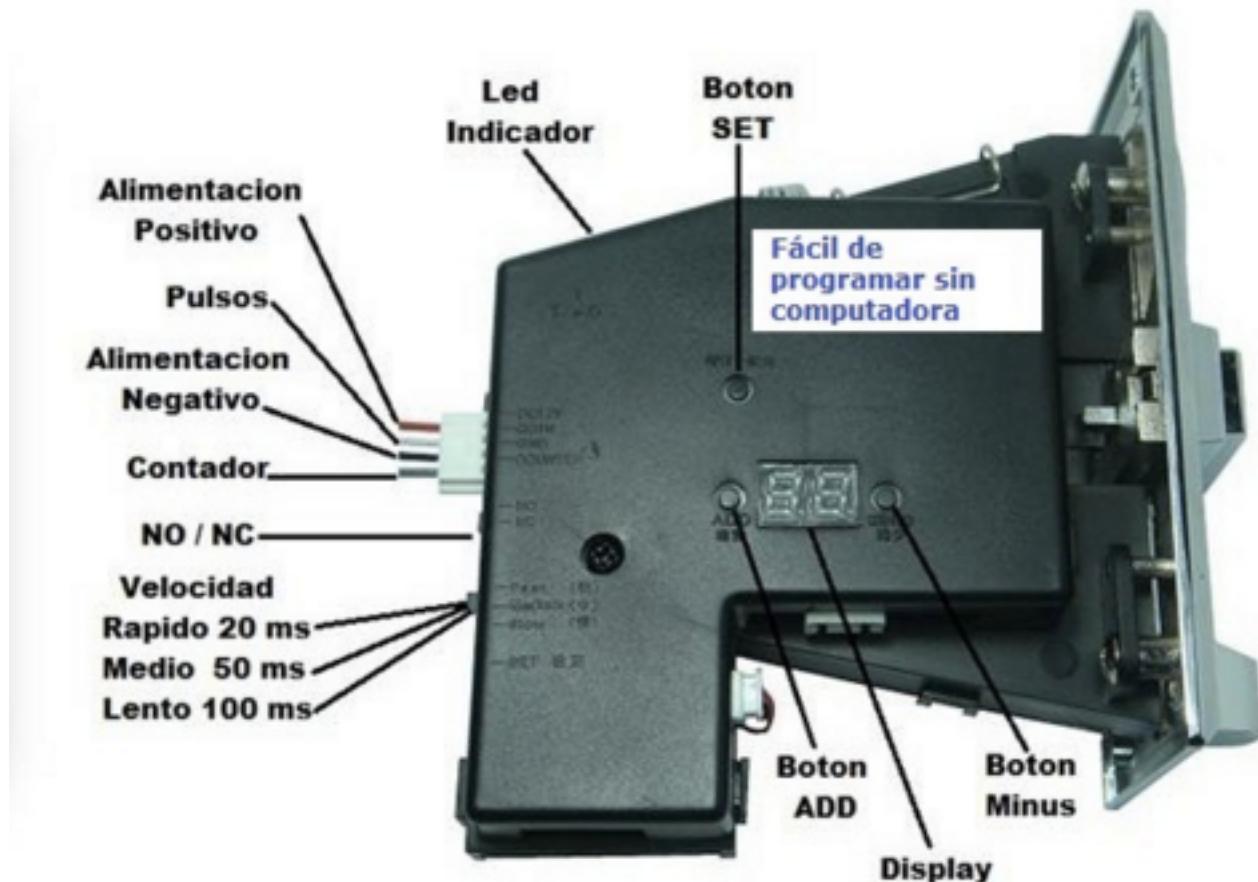
Sensor Para Hopper O Despachador De Metal

\$ 34⁹⁹
12 meses de \$ 3⁴⁶

Artículo nuevo
Distrito Federal

PARTE 001: Leer Pulses del Monedero Electrónico

Queremos que la maquina lea distintas monedas y según estas monedas, expulse una cantidad de pesos. El monedero que viene con la maquina de tragamonedas solo lee un tipo de moneda, **este monedero no nos sirve**, a menos que metamos uno de estos para cada tipo de moneda. Por el precio de \$650, podemos emplear un solo monedero electrónico que se le puede configurar para que lea hasta 4 diferentes tipos de moneda.



Hay que configurar el monedero con las monedas que queremos que lea, así como la cantidad de pulsos que queremos que emita al leer dicha moneda, no entrare en detalle de cómo hacer esto, dejare una liga a un video-tutorial para configurar el monedero: https://www.youtube.com/watch?v=G_ub03hjXhI

Vamos a configurar la moneda de \$10 para que emita #8 pulsos, la moneda de \$5 para que emita #7 pulsos, y la moneda de \$2 para que emita #4 pulsos. La cuarta moneda puede ser ya sea de \$1 o de \$0.50. Como esta maquina dará feria de \$1 opte por configurar la cuarta moneda como \$0.50 y que esta emita #3 pulsos. P.D. Esto lo hago pues en el lector de billetes tenemos ocupado los números #2,#5,#10,#20,#50 para billetes. Una vez vea como cambiar estos pulsos, sugiero emplear del #(2-9) pulsos para monedas y de #(10-19) pulsos para billetes. Dejando un espacio de dos a 3 pulsos de diferencia entre valores para evitar que por ruido o fallas, algún usuario que introduzca una moneda se lleve feria de un billete.

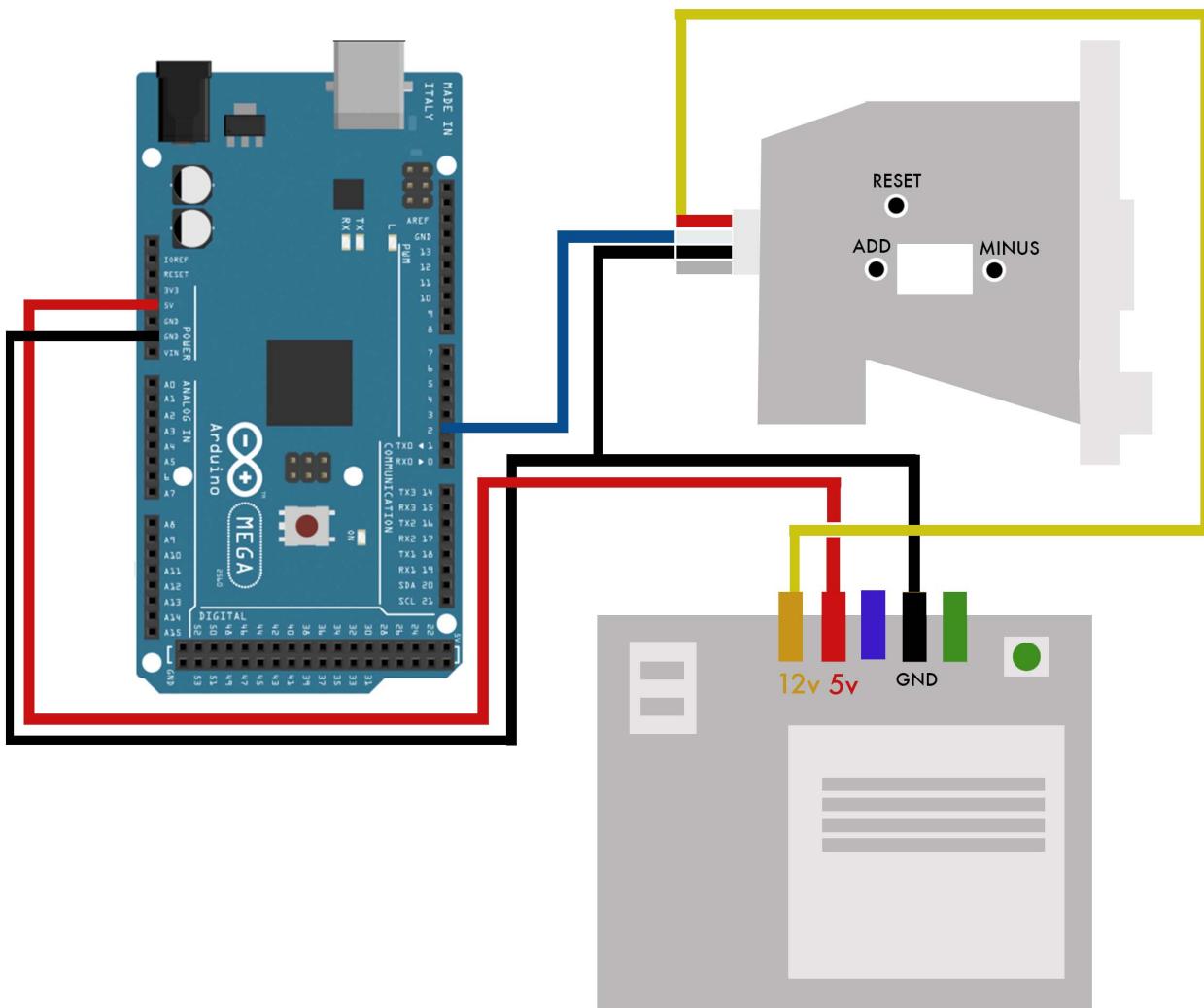
MONEDA	\$0.50	\$2.00	\$5.00	\$10.00
PULSOS	#3	#4	#7	#8
BILLETE	\$20.00	\$50.00	\$100.00	\$200.00
PULSOS	#2	#5	#10	#20

MONEDA	\$0.50	\$2.00	\$5.00	\$10.00
PULSOS	#2	#4	#6	#8
BILLETE	\$20.00	\$50.00	\$100.00	\$200.00
PULSOS	#12	#15	#18	#21

Una vez configurado el monedero múltiple, necesitamos calar que este monedero funcione, introduzca una moneda de \$10, el monedero deberá de mostrar en el "Display" el numero de pulsos que usted le designo al configurarlo, en este caso #8 para esa moneda.

Repita este procedimiento para cada moneda, de no leer un tipo de moneda o no mostrar los pulsos deseados, regrese al video y repita el procedimiento.

Una vez seguro de que el monedero esta pre configurado correctamente, falta hacer que podamos leer y contar estos pulsos con el Arduino Mega. A continuación conectaremos el monedero como sigue, empleando la fuente de la maquina de tragamonedas, conectaremos el cable rojo del monedero a la salida de la fuente de 12v, conectaremos el cable blanco del monedero a el pin #2 del arduino mega, conectaremos el cable negro a GND de la fuente de la maquina de tragamonedas, los cables grises, vienen 2, son para un contador electrónico, estos los dejaremos solos. Hay que conectar el arduino mega a la fuente de la maquina tragamonedas con salida de 5v, al pin que dice "5v" y el cable de tierra GND de la fuente de poder de la maquina tragamonedas al pin del arduino mega que dice GND. Dibuje un diagrama de cómo debería de quedar.



Empleo la fuente que venia con la maquina tragamonedas, pero aun falta darle poder al lector de billetes, y este requiere mayor amperaje que los que requieren el arduino, el monedero, y el relevador. No se si se pueda emplear una fuente especial para conectar todos a una, mientras yo ocupo 2 fuentes. OJO: los pins de la fuente varian de orden, asi vienen en las fuentes que yo tengo, estas pueden no ser las mismas para su fuente, normalmente vienen etiquetadas, pero si no, hay que medir cuanto voltaje emite cada una, no se como xd creo que con un multímetro.

Una vez este conectado el hardware, hay que escribir un programa con el Arduino IDE para leer los pulsos emitidos por el monedero al introducir una moneda.

Yo escribi este programa "test_leer_pulsos_monedero.ino" se lo adjuntare en el correo junto con este archivo si falta pídamelo yo se lo vuelvo a enviar, espero le resulte útil, usted esta libre de escribir el suyo, le advierto este código a veces cuenta pulsos de mas, y este bug hasta hoy no lo eh podido arreglar. Espero que compartiendo y comentando el código pueda ver la falla, o me pueda ayudar usted a generar otro código para leer los pulsos.

Copie y pegue el siguiente programa en el IDE del arduino, guárdelo en su computadora, verifique que este bien, compilandolo con el icono de la palomita, luego conecte el arduino mega, verifique que el COM y el tipo de placa sean los correctos, para poder cargar el programa al arduino mega.

Una vez cargado, abra la consola (icono lupa), espere a que cargue el programa, encienda el monedero electrónico, (asegurese de que este todo conectado como en la imagen anterior) e introduzca una moneda, este programa debería de mostrarle el siguiente mensaje en consola "Nueva moneda detectada" y luego la cantidad de pulsos asignados a esa moneda.

```

/////////////////////////////INICIO DE PROGRAMA/////////////////////////////
//Variables Globales
///Lo pongo como variables de tipo constante para guardar espacio en memoria
//*********************************************************************
const int limite_pulsos = 500; // EDITAR ESTE VALOR PARA CAMBIAR EL DELAY ENTRE DETECCION DE PULSOS-nesecito
poner un DELAY es como una pausa entre la detección de pulsos para que el arduino me lea los pulsos sin
arrojar basura, eh calado multiples valores y 500 es el mas estable
//*********************************************************************
const int _pin = 2; //Asigno el valor de 2 para la variable donde conectare el pin que va a recibir los
pulsos del monedero
///// Una variable debe ser declarada como "volátil" cuando su valor puede ser cambiado por algo fuera del
control de la sección de código en el que aparece, como un hilo al mismo tiempo de ejecución. En el Arduino,
el único lugar que es probable que ocurra en secciones de código asociados a las interrupciones, llama una
rutina de servicio de interrupción.
volatile byte cont_pulsos_moneda = 0;//Variable para guardar el valor de los pulsos contados
volatile unsigned long tiempo_pulso; //Variable para guardar el tiempo en el que se leyó el pulso
byte pulsos_total; // lugar para guardar el total de pulsos detectados
byte nueva_moneda=0;// lugar para guardar el valor de la moneda basandome en el total de pulsos detectados

void setup() {
    Serial.begin(9600); // Establece la velocidad de datos en bits por segundo (baudios) para la transmisión de
datos en serie. Para comunicarse con la pc
    Serial.println("Cargando Setup, por favor esperar..."); //Para mostrar en consola que ya inicio la función
setup()
//pinMode: Configura el pin especificado para comportarse ya sea como una entrada o una salida: pines
digitales se pueden utilizar como entrada, INPUT_PULLUP o SALIDA. Cambiar el pin con la función pinMode () cambia el
comportamiento eléctrico de la clavija.///*input: de afuera hacia el arduino - output*del arduino
ahcia afuera - a pull-up* es la resistencia interna del arduino, que asegura que el pin este ya sea en un
estado ALTO o BAJO, mientras emplea un nivel bajo de corriente.
    pinMode(coin_pin, INPUT_PULLUP); //establecemos el pin 2 (coin_pin) como tipo INPUT y le adjuntamos una
resistencia interna para evitar rebotes
/// El procesador en el corazón de cualquier Arduino tiene dos tipos diferentes de interrupciones: "externo"
y "Cambiar PIN".
/// Si usted quiere asegurarse de que un programa siempre capte los pulsos de un encoder rotativo, de manera
que nunca se pierde un pulso, sería difícil escribir un programa que no hiciera otra cosa más que eso, ya que
el programa tendría que sondear constantemente las líneas de sensores para el codificador, con el fin de
detectar pulsos solo cuando ocurran emplearemos la función de attachInterrupt()*
/// Las interrupciones son útiles formas de hacer que las cosas sucedan de forma automática en los programas
de microcontroladores, y puede ayudar a resolver problemas de tiempo.
/// Estas interrupciones pueden configurarse para activar una función específica en flancos ascendentes o
descendentes de la señal, o bajo nivel.
/// * OJO: Dentro de la función de "attachInterrupt()", la función "delay()" no funcionará y el valor
devuelto por "millis()" no se incrementará. Los datos recibidos mientras dentro de la función podrán ser
perdidos. Usted debe declarar como volátiles cualquier variable que modifique dentro de la función adjunta.
Vea la sección de ISR a continuación para obtener más información.
    attachInterrupt(digitalPinToInterrupt(coin_pin), sumar_contador_moneda, FALLING);
//SYNTAXIS: attachInterrupt(pin,ISR,mode);
/**pin: parámetro para decir a la función que pin desea adjuntar el interruptor (Arduino_MEGA cuenta con x6
pins que admiten esta posibilidad de actuar como interruptores estos son [2, 3, 18, 19, 20, 21]) los pins
específicos con capacidad para interrupciones, y su mapeo para interrumpir llevan un número diferente a su
número de pin, estos números varían según el modelo de arduino, por esta razón empleamos la función
"digitalPinToInterrupt()" para obtener este valor para el pin en específico y evitar confusiones
/**ISR: función a invocar cuando se produce una interrupción; esta función no debe recibir parámetros y debe
regresar nada
/**mode: define cuándo se debe producir una interrupción, hay 4 constantes predefinidas como valores y
estos son;
    /**LOW: acciona la interrupción cuando el pin se encuentre en estado bajo (0v)
    /**CHANGE: acciona la interrupción cada vez que el pin cambie de valor
    /**RISING: acciona la interrupción cuando el pin vaya de estado bajo a estado alto (0v a 5v)
    /**FALLING: acciona la interrupción cuando el pin vaya de estado alto a estado bajo (5v a 0v)
    Serial.println("Morralla Setup Cargado correctamente."); //Imprimo en consola para notificarme que la
función setup() llegó a su fin
}

void loop() {
    //dejar de contar pulsos
    if (cont_pulsos_moneda >0 && millis()-tiempo_pulso > limite_pulsos)

```

```

//si hubo una interrupcion y el contador se incremento entonces hubo una moneda detectada
(cont_pulsos_moneda>0) && si el tiempo actual menos el tiempo del ultimo pulso es mayor a los 500
milisegundos entonces hacer lo siguiente
{
    nueva_moneda = cont_pulsos_moneda; //Guardo el valor del total de pulsos en esta variable para liberar
espacio en la variable cont_pulsos_moneda
    Serial.print("Nueva moneda detectada ");//me alerta en consola que hubo una moneda detectada
    Serial.println(nueva_moneda);           // imprime en consola el valor de esa moneda

    cont_pulsos_moneda = 0;                //reseteo a 0 el contador de pulsos del interruptor

}
switch (nueva_moneda) {

case 3://$0.50
    Serial.println("$0.50");   //muestro en consola el valor de la moneda segun los pulsos leidos
    nueva_moneda = 0;        //reseteo - no se si este paso es nesesario?
    break;

case 4://$2.00
    Serial.println("$2.00 ");  //muestro en consola el valor de la moneda segun los pulsos leidos
    nueva_moneda = 0;        //reseteo - no se si este paso es nesesario?
    break;

case 7://$5.00
    Serial.println("$5.00");  //muestro en consola el valor de la moneda segun los pulsos leidos
    nueva_moneda = 0;        //reseteo - no se si este paso es nesesario?
    break;

case 8://$10.00
    Serial.println("$10.00"); //muestro en consola el valor de la moneda segun los pulsos leidos
    nueva_moneda = 0;        //reseteo - no se si este paso es nesesario?
    break;
}

} //*****INTERUPCION al detectar pulsos del monedero
void sumar_contador_moneda()      //funcion invocada para incrementar el contador cont_pulsos_moneda
{
    cont_pulsos_moneda++;
    tiempo_pulso = millis(); //guardo el tiempo ACTUAL y se lo asigno a la variable como el tiempo cuando leyo
el pulso
}
///////////////////////////////FIN DE PROGRAMA///////////////////////////////

```

PARTE 002: Leer pulsos del Lector NV10USB

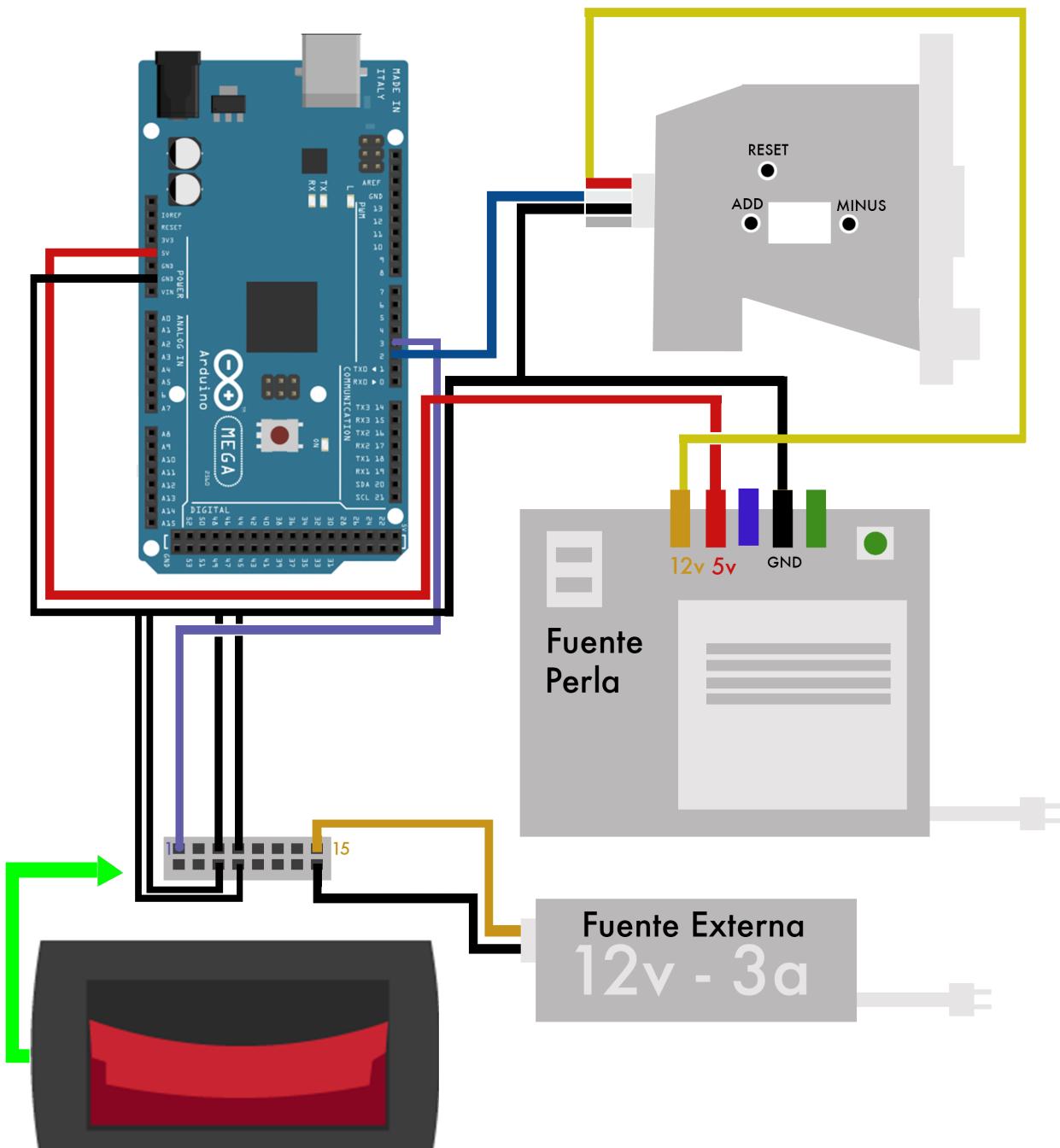
Para leer los billetes, hay que poner el lector de billetes en modo de pulsos. Una vez en modo de pulsos hay que poner los pins como sigue.

-El pin #1 del NV10USB al pin #3 del Arduino Mega (para que se le pueda agregar la función de AttachInterrupt() este debe de estar en pins específicos según el modelo de Arduino ya empleamos el #2, para este emplearemos el siguiente pin #3)

-Los Pins #5 NV10USB, #6 NV10USB, #7 NV10USB, #8 NV10USB van a tierra, (para que el NV10USB lea los 4 tipos de billetes, \$20,\$50,\$100,\$200 en modo de pulsos sino se ponen estos no va a leer ningún billete en modo de pulsos, este detalle me falto cuando adqueri el NV10USB para que funcionara sin uso de la compu)

-El pin #15 NV10USB a los 12v y 3amps de la fuente externa

-El pin #16 NV10USB a GND de la fuente externa.



Yo uso dos fuentes, pero creo que usted me podría ayudar a solo emplear 1 sola fuente para alimentar todo. Aun no eh calado conectar el arduino a la fuente de PC. Creo que tiene demasiado Amperaje y podría quemar el arduino. No estoy seguro.

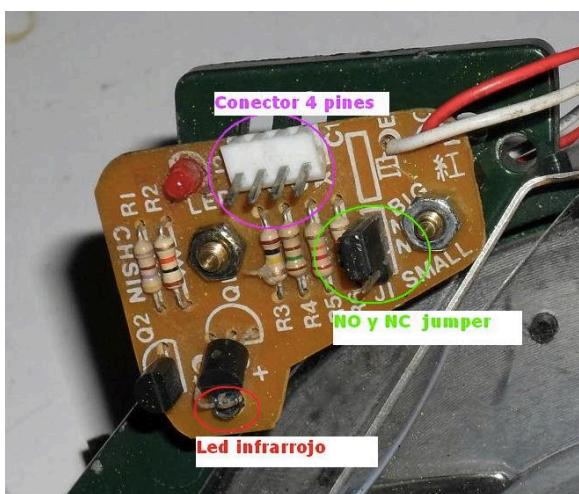
```

/////////////////////////////INICIO DE PROGRAMA////////////////////////////
//Programa para Leer y contar pulsos del NV10USB /////
//Variables
volatile byte billPulseCount = 0; // contador para guardar cuantos pulsos ah contado
volatile unsigned long pulseTime; //guarda el tiempo del pulso
volatile unsigned long billTime; //guarda el ultimo tiempo de pulso
byte newBillInserted; // Variable para guardar la cantidad de pulsos una vez terminado de contar
byte pulseThreshold = 500; //EDITAR este valor para el DELAY entre pulsos
//pins empleados
int billSelectorPin = 3; // pin #3 como input para NV10USB
//Funcion Setup() que acuta como constructor, solo corre una vez al inicio
void setup(){
    Serial.begin(9600); //inicia comunicacion con la pc
    //establezco los modes iniciales de los pins
    pinMode(billSelectorPin, INPUT_PULLUP); //poner el pin #03 como input
    digitalWrite(0,HIGH); //poner ese valor inicialmente como HIGH
    attachInterrupt(digitalPinToInterrupt(billSelectorPin), billacceptor, FALLING); // Agrego Interruptor al
pin y si detecta algo en modo FALLING llamar a funcion "billacceptor"
}//fin de funcion setup()
void loop(){
    if (billPulseCount > 0 && millis() - pulseTime > pulseThreshold && millis() - pulseTime !=0)
    //si hubo una interrupcion y el contador se incremento entonces hubo una moneda detectada
    (cont_pulsos_moneda>0) && si el tiempo actual menos el tiempo del ultimo pulso es mayor a los 500
    milisegundos entonces dejar de contar y hacer lo siguiente
{
    billTime = millis() - pulseTime; //guardo el tiempo detencion de billete(tiempo actual - t_ultimo pulso)
    newBillInserted = billPulseCount; // guardo el total de pulsos en esta variable
    Serial.println(".");
    Serial.print("Tiempo: ");
    Serial.println(millis());
    Serial.print("Ultimo tiempo de pulso: ");
    Serial.println(pulseTime);
    Serial.print("Dif de Tiempo: ");
    Serial.println(billTime);
    Serial.println(".");
    Serial.print("Billete detectado ->Cantidad de pulsos: ");
    Serial.println(newBillInserted); // imprime en consola la variable
    billPulseCount = 0; //Reseteo a 0 el contador de pulsos
} //fin del IF
//Codigo que muestra en consola el tipo de billete segun la cantidad de pulsos en variable
switch (newBillInserted) {
    case 2:
        Serial.println("$20.00"); //4 pulsos del NV10USB significa que detecto un billete de $20
        newBillInserted = 0; //reset
        Serial.println("SE FERIO $18");
        break;
    case 5:
        Serial.println("$50.00"); //4 pulsos del NV10USB significa que detecto un billete de $50
        newBillInserted = 0; //reset
        Serial.println("SE FERIO $45");
        break;
    case 10:
        Serial.println("$100.00"); //4 pulsos del NV10USB significa que detecto un billete de $100
        newBillInserted = 0; //reset
        Serial.println("SE FERIO $90");
        break;
    case 20:
        Serial.println("$200.00"); //4 pulsos del NV10USB significa que detecto un billete de $200
        newBillInserted = 0; //reset
        Serial.println("SE FERIO $180");
        break;
}
}

//funcion para incrementar contador si es que se detecta una interrupcion en el pin #03 del arduino
void billacceptor()
{
    pulseTime = millis(); //guarda el tiempo actual como tiempo del pulso
    billPulseCount++; //incremento la variable del contador
    //Serial.print("b:");
    // (omitar estas lineas las uso para ver en consola si entro la funcion)
    //Serial.print(billPulseCount);
} //End of billacceptor()
/////////////////////////////FIN DE PROGRAMA////////////////////////////

```

PARTE 003: Leer Pulsos del Sensor de Hopper

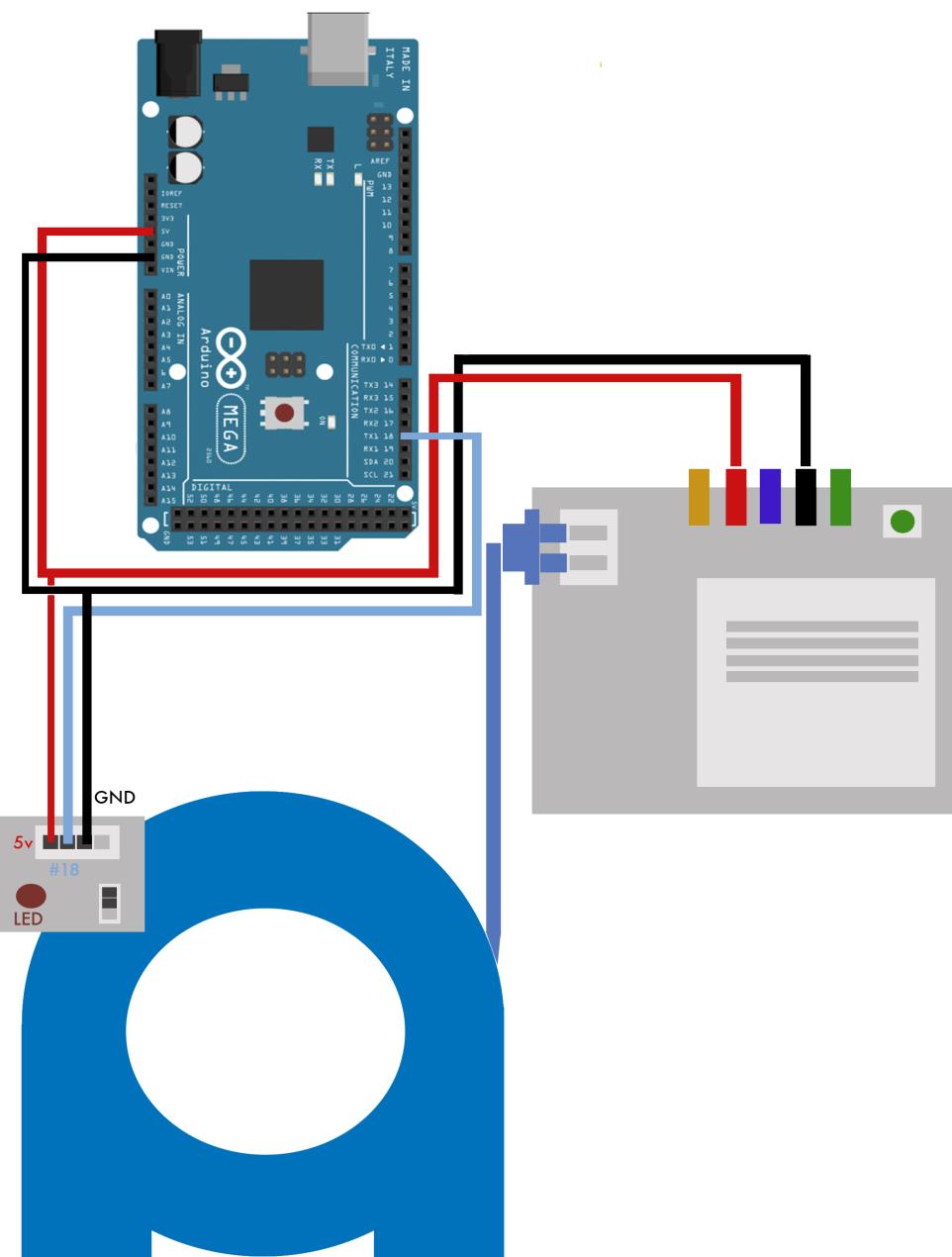


Ok, entonces ya podemos leer y contar pulsos tanto del lector de billetes como del monedero electrónico. Ya tenemos 2 fuentes de input, ahora para hacer el output, emplearemos el Hopper de la maquina de apostar, prenderemos este motorsito y con los pulsos del sensor en el Hopper sabremos cuando apagar el motorsito.

Este sensor emite un pulso cuando pasa una moneda eh interrumpe la señal del **Led infrarrojo**. Los 4 Pins en morado van de Izquierda a derecha 1---→4

1. Entrada de 5v para encender sensor
2. Salida de pulsos → #18 Arduino Mega
3. A Tierra GND
4. Este no le pongo nada
5. NC [-][+][] NO

OJO: estos valores los saque a modo de prueba y error, pueden estar mal jajajaja pero asi detecta pulsos y asi los deje.



Primero leeremos y contaremos los pulsos del sensor del Hopper de la misma forma que hicimos con los pulso del lector de monedas y billetes. Pero como aun no conectamos el relevador de 5v al Hopper para accionar remotamente y en automático el motor del Hopper, emplearemos el botón verde (en la imagen, en la fuente es negro) que dice SSR. Este botón al presionarlo y teniendo el cable del Hopper conectado a la fuente Perla, accionara el motor del Hopper manualmente.

Este motor prenderá siempre y cuando ese botón SSR este presionado.

Para contar y leer los pulsos del sensor del Hopper, hay que descargar el siguiente programa al Arduino MEGA, dejar mínimo 15 a 20 monedas en el Hopper y manualmente presionar el botón SSR para accionar el motor.

Si todo funciona bien, el programa leerá y mostrara en consola las veces que la función `AttachInterrupt` detecto un cambio, llamara a la función `pulsoCont++()` y aumentara el contador de pulsos para el sensor. Habremos completado esta parte.

```

/////////////////////////////INICIO DEL PROGRAMA////////////////////////////
//Variables Globales
const int limite_pulsos = 500; // EDITAR ESTE VALOR PARA CAMBIAR EL DELAY ENTRE DETECCION DE PULSOS-nesecito
poner un DELAY es como una pausa entre la deteccion de pulsos para que el arduino me lea los pulsos sin
arrojar basura, eh calado multiples valores y 500 es el mas estable, es una forma de evitar rebotes o contar
pulsos de mas
const int sensor_pin = 18; //Asigno el valor de 18 para la variable donde conectare el pin que va a recibir
los pulsos del sensor del Hopper
volatile byte cont_pulsos_sensor = 0; //Variable para guardar el valor de los pulsos contados
volatile unsigned long tiempo_pulso_sensor; //Variable para guardar el tiempo en el que se leyó el pulso
byte pulsos_sensor_total; // lugar para guardar el total de pulsos detectados

void setup() {
    Serial.begin(9600);
    Serial.println("Cargando Setup, por favor esperar..."); //Para mostrar en consola que ya inicio la funcion
setup()
    pinMode(sensor_pin, INPUT_PULLUP); //establecemos el pin 18 (coin_pin) como tipo INPUT y le adjuntamos una
resistencia interna para evitar rebotes

    attachInterrupt(digitalPinToInterrupt(sensor_pin), sumar_contador_sensor, FALLING); //asignamos un
attachInterrupt al pin especial del pin 18, llamaremos la función sumar_contador_sensor cuando este detecte un
cambio de alto a bajo
    Serial.println("Morralla test de sensor de hopper Cargado correctamente."); //Imprimo en consola para
notificarme que la función setup() llegó a su fin
}

void loop() {
    //no haremos nada aquí pues el objetivo es mostrar en consola que el pin 18 está leyendo los pulsos
emittedos por las monedas saliendo del hopper, y luego contarlas
}

//*****INTERUPCIÓN al detectar pulsos del monedero
void sumar_contador_sensor() //función invocada para incrementar el contador cont_pulsos_moneda
{
    cont_pulsos_sensor++;
    tiempo_pulso_sensor = millis(); //guardo el tiempo ACTUAL y se lo asigno a la variable como el tiempo
cuando leyó el pulso
    Serial.println(cont_pulsos_sensor); //imprimo en consola el valor de la variable cont_pulsos_sensor
    Serial.print("tiempo:");Serial.print(tiempo_pulso_sensor); //imprimo en consola el tiempo en el que leyó el
pulso
}
//////////////////////////FIN DE PROGRAMA////////////////////////////

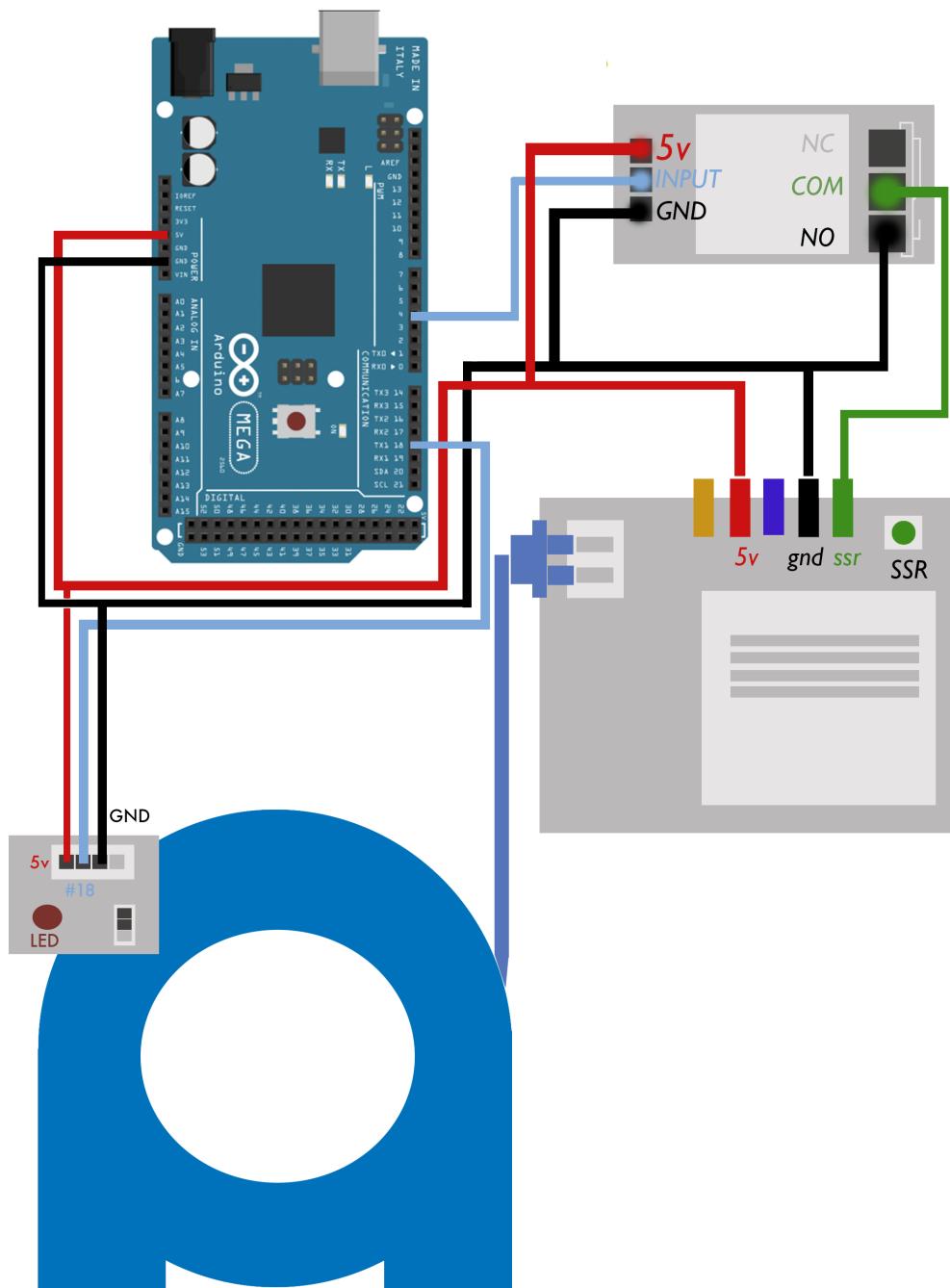
```

PARTE 004: Activar Relevador para Accionar Hopper con la ayuda de la fuente

Bueno ahora que ya podemos leer y contar todos los pulsos entrantes al arduino, los del monedero, los del lector de billetes, y los del sensor del Hopper, toca poder prender el Hopper con el arduino para que este expulse las monedas en función a los pulsos detectados por el sensor del Hopper.

Para esto emplearemos un relevador de 5v, como el que le mande por correo. Este no lo usaremos para conectar la luz hacia el Hopper, y accionarlo con el arduino como debería de usarse un relevador, sino mas bien lo emplearemos para actuar como un botón que jale a GND el pin del SSR de la fuente, que hará lo mismo que usted estaba haciendo manualmente al presionar el botón. Esto por que? Pues no supe como hacer prender el Hopper sin la fuente, intente empleando el relevador y conectándolo a la luz pero el Hopper se calentaba como plancha y solo funcionaba 3 de cada 5 veces. Eso no es bueno, así que decidí hacer lo de antes porque funciona.

Entonces conectamos el relevador de 5v como sigue, el pin de 5v a los 5v de la fuente perla, el pin de IN1 al pin del arduino Mega #4, el pin de GND del relevador al pin de GND de la fuente perla, y del lado derecho conectamos el pin NO (normalmente abierto) a GND de la fuente perla, y el pin COM (común) a el pin SSR del la fuente perla.



ayudara a pulir esta parte en especial, o a crear una nueva forma para controlar mejor las salidas de las monedas.

El primer programa solo es para calar que el relevador funcione, lo único que hace es accionar el relevador, con accionar me refiero a conectar COM con NO para jalar el pin SSR de la fuente a GND, es como si aplana el botón. El programa ocupara de que usted esté en consola, tecle la letra "p" para prender, y "a" para apagar.

El siguiente programa empleara la consola del arduino mega, como input, es decir haremos que esta recibiendo pulsos del monedero o del lector de billetes, introduciendo manualmente un numero en la consola del Arduino ide, para que este se meta a una función switch y según el numero de pulsos, accione el motor del Hopper hasta que el arduino lea y cuente pulsos del sensor del hopper menores a los que ocupa expulsar en monedas. Esto para meterle un delay, al final para que agarre suficiente momentum para expulsar la ultima moneda pero no demasiado para que expulse una moneda de mas. Por eso lo paramos 1 moneda antes. Este es el gran problema que no me permite trabajar con estas maquinas aun, a veces da de menos, pues el Hopper se detiene y expulsa una moneda de menos, y a veces tarda en detenerse y expulsa monedas de mas. Una moneda, rara vez expulsa mas de una o menos de una.

Quisiera que lo probara y me


```

void setup() {
    Serial.begin(9600); // Establece la velocidad de datos en bits por segundo (baudios) para la transmisión de
    // datos en serie. Para comunicarse con la pc
    Serial.println("Cargando Setup, por favor esperar..."); //Para mostrar en consola que ya inicio la función
    //pinMode: Configura el pin especificado para comportarse ya sea como una entrada o una salida: pines
    // digitales se pueden utilizar como entrada, INPUT_PULLUP o SALIDA. Cambiar el pin con la función pinMode () cambia el comportamiento eléctrico de la clavija.///*input: de afuera hacia el arduino - output*del arduino
    // hacia afuera - a pull-up* es la resistencia interna del arduino, que asegura que el pin este ya sea en un
    // estado ALTO o BAJO, mientras emplea un nivel bajo de corriente.
    pinMode(relevador_pin,OUTPUT); //establecemos el pin 4 del relevador como output
    pinMode(billetero_pin, INPUT_PULLUP); //establecemos el pin 3 (billetero_pin) como tipo INPUT y le
    // adjuntamos una resistencia interna para evitar rebotes
    pinMode(monedero_pin, INPUT_PULLUP); //establecemos el pin 2 (monedero_pin) como tipo INPUT y le
    // adjuntamos una resistencia interna para evitar rebotes
    //valores iniciales
    digitalWrite(relevador_pin, HIGH); //Apagamos el relevador por default
    //digitalWrite(sensor_hopper_pin, HIGH); //ponemos el valor inicial de los pins con interrupciones como
    ALTOS
    //digitalWrite(monedero_pin, HIGH); //ponemo el valor inicial de los pins con interrupciones como ALTOS
    //digitalWrite(billetero_pin, HIGH); //ponemo el valor inicial de los pins con interrupciones como ALTOS
    // El procesador en el corazón de cualquier Arduino tiene dos tipos diferentes de interrupciones: "externo"
    // y "Cambiar PIN".
    // Si usted quiere asegurarse de que un programa siempre capte los pulsos de un encoder rotativo, de manera
    // que nunca se pierde un pulso, sería difícil escribir un programa que no hiciera otra cosa más que eso, ya que
    // el programa tendría que sondear constantemente las líneas de sensores para el codificador, con el fin de
    // detectar pulsos solo cuando ocurran emplearemos la función de attachInterrupt()*
    // Las interrupciones son útiles formas de hacer que las cosas sucedan de forma automática en los programas
    // de microcontroladores, y puede ayudar a resolver problemas de tiempo.
    // Estas interrupciones pueden configurarse para activar una función específica en flancos ascendentes o
    // descendentes de la señal, o bajo nivel.
    // * OJO: Dentro de la función de "attachInterrupt()", la función "delay()" no funcionará y el valor
    // devuelto por "millis()" no se incrementará. Los datos recibidos mientras dentro de la función podrán ser
    // perdidos. Usted debe declarar como volátiles cualquier variable que modifique dentro de la función adjunta.
    // Vea la sección de ISR a continuación para obtener más información.
    attachInterrupt(digitalPinToInterrupt(sensor_hopper_pin), sumar_contador_pulsos, FALLING); //interupcion
    // para el sensor del hopper
    attachInterrupt(digitalPinToInterrupt(monedero_pin), sumar_contador_pulsos, FALLING); //interupcion para el
    monedero
    attachInterrupt(digitalPinToInterrupt(billetero_pin), sumar_contador_pulsos, FALLING); //interupcion para
    el Nv10USB
    //SYNTAX: attachInterrupt(pin,ISR,mode);
    //**pin: parámetro para decir a la función que pin desea adjuntar el interruptor (Arduino_MEGA cuenta con x6
    // pins que admiten esta posibilidad de actuar como interruptores estos son [2, 3, 18, 19, 20, 21] ) los pins
    // específicos con capacidad para interrupciones, y su mapeo para interrumpir llevan un número diferente a su
    // número de pin, estos números varían según el modelo de arduino, por esta razón empleamos la función
    // "digitalPinToInterrupt()" para obtener este valor para el pin en específico y evitar confusiones
    // **ISR: función a invocar cuando se produce una interrupción; esta función no debe recibir parámetros y debe
    // regresar nada
    //**mode: define cuando se debe de producir una interrupción, hay 4 constantes predefinidas como valores y
    // estos son;
    //****LOW: acciona la interrupción cuando el pin se encuentre en estado bajo (0v)
    //****CHANGE: acciona la interrupción cada vez que el pin cambie de valor
    //****RISING: acciona la interrupción cuando el pin vaya de estado bajo a estado alto (0v a 5v)
    //****FALLING: acciona la interrupción cuando el pin vaya de estado alto a estado bajo (5v a 0v)
    Serial.println("Morralla_1_moneda Setup Cargado correctamente."); //Imprimo en consola para notificarme que
    // la función setup() llegó a su fin
} //FIN DEL FUNCION SEUP()
*****void loop() {
    //dejar de contar pulsos
    if (cont_pulsos >0 && millis()-tiempo_pulso > limite_pulsos && millis()-tiempo_pulso != 0)
        //si hubo una interrupción y el contador se incrementó entonces hubo una moneda detectada
        (cont_pulsos_moneda>0) && si el tiempo actual menos el tiempo del último pulso es mayor a los 500
        milisegundos entonces hacer lo siguiente
    {
        pulsos_total = cont_pulsos; //Guardo backup el valor del total de pulsos en esta variable para liberar
        espacio en la variable cont_pulsos_moneda
}

```

```

Serial.print("Nuevos Pulsos detectados: ");//me alerta en consola que hubo una moneda detectada
Serial.println(pulsos_total);           // imprime en consola el valor de esa moneda
cont_pulsos = 0;                      //reseteo a 0 el contador de pulsos del interruptor
}
switch (pulsos_total) {
case 3://$0.50
    Serial.println("$0.50");   //muestro en consola el valor de la moneda segun los pulsos leidos
    toston++;
    if(toston==3){
        valor_expulsar=1;
        dispence();
    }else{
        Serial.println("Introduzca otra moneda de $0.50");
    }
    break;
case 4://$2.00
    Serial.println("$2.00 ");   //muestro en consola el valor de la moneda segun los pulsos leidos
    valor_expulsar=1;
    dispence();
    Serial.println("Se ferio $1.00");
    break;
case 7://$5.00
    Serial.println("$5.00");   //muestro en consola el valor de la moneda segun los pulsos leidos
    valor_expulsar=4;
    dispence();
    Serial.println("Se ferio $4.00");
    break;
case 8://$10.00
    Serial.println("$10.00");  //muestro en consola el valor de la moneda segun los pulsos leidos
    valor_expulsar=9;
    dispence();
    Serial.println("Se ferio $9.00");
    break;
case 2://$20.00
    Serial.println("$20.00");  //muestro en consola el valor de la moneda segun los pulsos leidos
    valor_expulsar=18;
    dispence();
    Serial.println("Se ferio $18.00");
    break;
case 5://$50.00
    Serial.println("$50.00 "); //muestro en consola el valor de la moneda segun los pulsos leidos
    valor_expulsar=45;
    dispence();
    Serial.println("Se ferio $45.00");
    break;
case 10://$100.00
    Serial.println("$100.00"); //muestro en consola el valor de la moneda segun los pulsos leidos
    valor_expulsar=90;
    dispence();
    Serial.println("Se ferio $90.00");
    break;
case 20://$200.00
    Serial.println("$200.00"); //muestro en consola el valor de la moneda segun los pulsos leidos
    valor_expulsar=180;
    dispence();
    Serial.println("Se ferio $180.00");
    break;
case 50://$500.00
    Serial.println("$500.00"); //muestro en consola el valor de la moneda segun los pulsos leidos
    valor_expulsar=450;
    dispence();
    Serial.println("Se ferio $450.00");
    break;
}
//FIN DEL FUNCION LOOP()
*****void dispence()//Funcion para prender el relevador hasta que detecte el arduino pulsos del sensor del hopper
menor o igual a los indicados por coinValue
{

```

```

digitalWrite(relevador_pin, LOW); //turn on relay - active LOW.
//delay(VarDelayA);
cont_pulsos =0; //Variable para
while (cont_pulsos < valor_expulsar)
{
    //do nothing and wait with the relay on dispensing coins until it hits the "coinValue"
}
//delay(VarDelayB); //wait to ensure the coin has enough momentum to leave hopper but not long enough
for another coins to dispence!
digitalWrite(relevador_pin, HIGH); //turn off relay - active LOW.
//*********************************************************************
delay(180);
cont_pulsos = 0; // ERROR IN SPIKES - hopper pulsing effects coin acceptor pulse line!
valor_expulsar=0;
pulsos_total=0;
//*********************************************************************
}//FIN DE FUNCION DISPENCE()
*****INTERUPCION al detectar pulsos del monedero
void sumar_contador_pulsos() //funcion invocada para incrementar el contador cont_pulsos_moneda
{
    cont_pulsos++;
    tiempo_pulso = millis(); //guardo el tiempo ACTUAL y se lo asigno a la variable como el tiempo cuando leyo
el pulso
} //FIN DEL FUNCION SUMAR_CONTADOR_PULSOS()
//*********************************************************************

```

Muy bien si todo salio bien ya tienes una maquina que lee billetes, monedas y expulsa pesos cobrando un 10% de comision.