ESCUELA DE ROBÓTICA

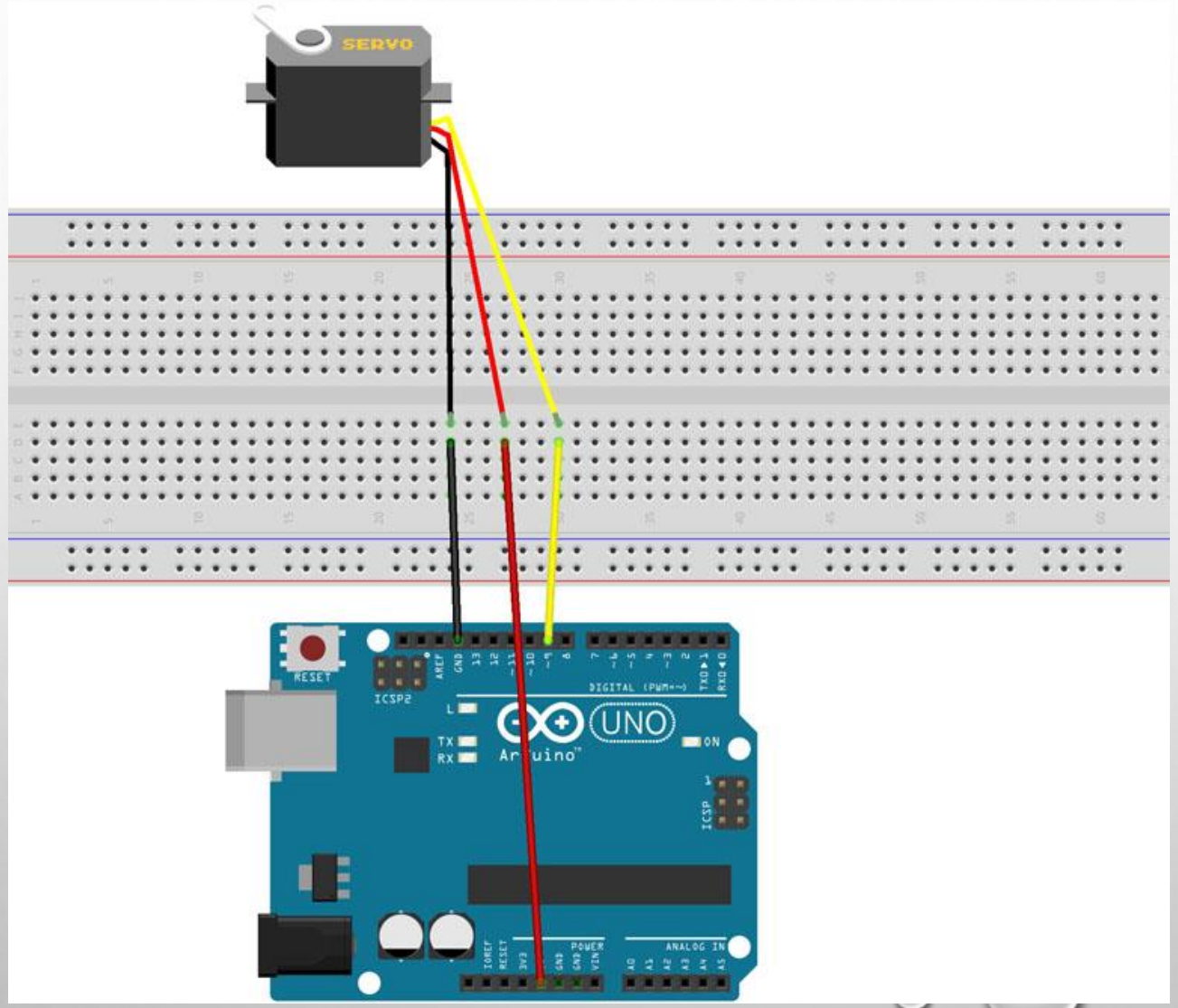
SERVOMOTOR

PARA QUE SIRVE?. NOS PERMITE **MANTENER UNA POSICIÓN** QUE INDIQUEMOS, SIEMPRE QUE ESTÉ DENTRO DEL RANGO DE OPERACIÓN DEL PROPIO DISPOSITIVO. POR OTRO LADO NOS PERMITE **CONTROLAR LA VELOCIDAD DE GIRO**, PODEMOS HACER QUE ANTES DE QUE SE MUEVA A LA SIGUIENTE POSICIÓN ESPERE UN TIEMPO

ESTOS MOTORES FUNCIONAN CON UNA SEÑAL **PWM**, CON UN PULSO DE TRABAJO ENTRE 1 MS Y 2 MS Y CON UN PERIODO DE 20 MS (50 HZ). ¿QUÉ QUIERE DECIR TODO ESTO? ESTE DATO NOS INDICA LA **VELOCIDAD MÁXIMA** A LA QUE PODEMOS MOVER EL SERVOMOTOR CON ARDUINO. SOLO PODREMOS CAMBIAR DE POSICIÓN CADA 20 MS. ESTO DEPENDERÁ DEL TIPO Y MARCA DE NUESTRO SERVO.

EL ELEGIR UNA SALIDA PWM U OTRA DA LO MISMO, TODAS LAS SALIDAS DE ESTE TIPO FUNCIONAN IGUAL.

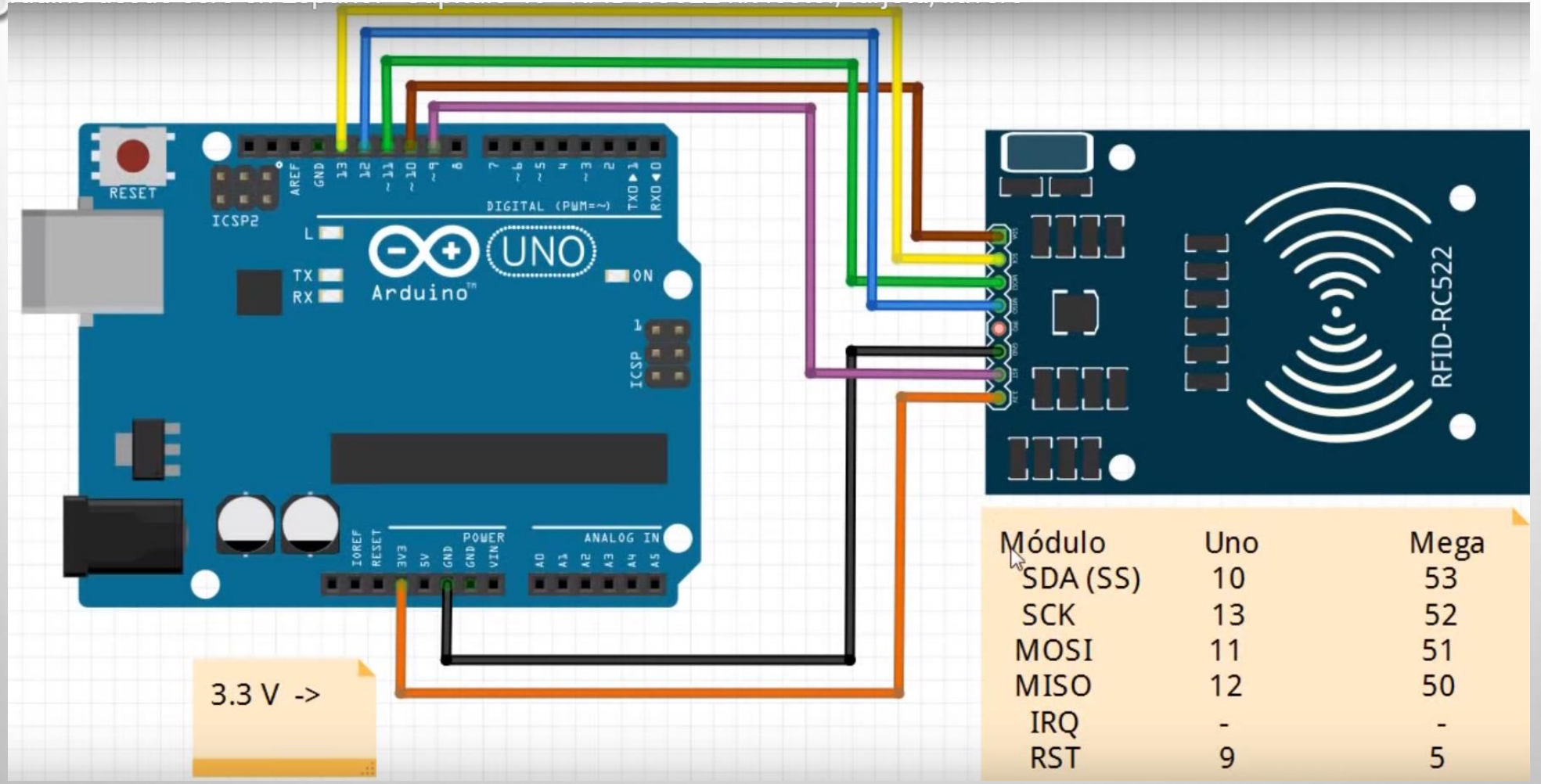
CIRCUITO ARDUINO



PROGRAMA

Programa de Arduino
por siempre
fijar ángulo del pin 9 del servo a 0
esperar 1 segundos
fijar ángulo del pin 9 del servo a 45
esperar 1 segundos
fijar ángulo del pin 9 del servo a 90
esperar 1 segundos
fijar ángulo del pin 9 del servo a 135
esperar 1 segundos
fijar ángulo del pin 9 del servo a 180
esperar 1 segundos

CIRCUITO


RESET
ICSP2
AREF
GND
DIGITAL (PWM=~)
TXO▲1
RXO▼0
L
TX
RX
UNO
Arduino
ON
ICSP
POWER
ANALOG IN
IOREF
RESET
3V3
5V
GND
GND
VIN
A0
A1
A2
A3
A4
A5
RFID-RC522
3.3 V ->
Módulo Uno Mega
SDA (SS) 10 53
SCK 13 52
MOSI 11 51
MISO 12 50
IRQ - -
RST 9 5

TARJETA DE ACCESO

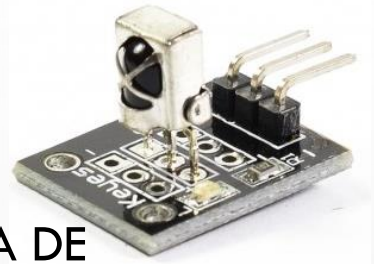
```
#include <SPI.h>
#include <MFRC522.h>
#define RST_PIN  9    //Pin 9 para el reset del RC522
#define SS_PIN  10   //Pin 10 para el SS (SDA) del RC522
MFRC522 mfrc522(SS_PIN, RST_PIN); ///Creamos el objeto
para el RC522
int ledSi = 4;
int ledNo = 3;
```

```
void setup() {
  Serial.begin(9600); //Iniciamos La comunicacion serial
  SPI.begin();        //Iniciamos el Bus SPI
  mfrc522.PCD_Init(); // Iniciamos el MFRC522
  Serial.println("Control de acceso:");
  pinMode(ledSi, OUTPUT);
  pinMode(ledNo, OUTPUT);
  }

byte ActualUID[4]; //almacenará el código del Tag leído
byte Usuario1[4]= {0x22, 0x59, 0xF5, 0x28} ; //código del
usuario 1
byte Usuario2[4]= {0x6E, 0x3F, 0x62, 0xA2} ; //código del
usuario 2
```

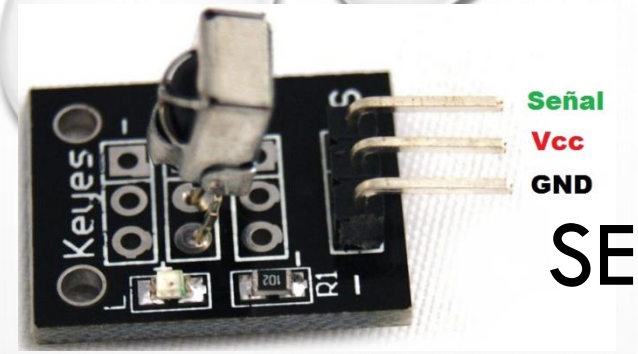
```
void loop() {
  // Revisamos si hay nuevas tarjetas  presentes
  if ( mfrc522.PICC_IsNewCardPresent())
       {
      //Seleccionamos una tarjeta
          if ( mfrc522.PICC_ReadCardSerial())
          {
                // Enviamos serialemente su UID
                Serial.print(F("Card UID:"));
                for (byte i = 0; i < mfrc522.uid.size; i++) {
                        Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
                        Serial.print(mfrc522.uid.uidByte[i], HEX);
                        ActualUID[i]=mfrc522.uid.uidByte[i];
                }
                Serial.print("     ");
                //comparamos los UID para determinar si es uno de nuestros usuarios
                if(compareArray(ActualUID,Usuario1)) {
                  Serial.println("Hola Anonimo");
                  digitalWrite(ledSi,HIGH);
                  digitalWrite(ledNo,LOW);
                }
                else{
                  if(compareArray(ActualUID,Usuario2)){
                  Serial.println("Hola Dinosaurio");
                  digitalWrite(ledSi,HIGH);
                  digitalWrite(ledNo,LOW);
                  }
                  else
                  {
                  Serial.println("Acceso denegado...");
                  digitalWrite(ledSi,LOW);
                  digitalWrite(ledNo,HIGH);
                  }
                 }
```

```
// Terminamos la lectura de la tarjeta tarjeta  actual
                mfrc522.PICC_HaltA();

          }

  }

}

//Función para comparar dos vectores
 boolean compareArray(byte array1[],byte array2[])
{
  if(array1[0] != array2[0])return(false);
  if(array1[1] != array2[1])return(false);
  if(array1[2] != array2[2])return(false);
  if(array1[3] != array2[3])return(false);
  return(true);
}
```

SENSOR IR CON CONTROL REMOTO



- ESTE SENSOR TIENE UN FILTRO INTERNO PARA DETECTAR SOLO FRECUENCIAS INFRARROJOS CERCANAS A 38KHZ, LO QUE LO HACE COMPATIBLE CON LA MAYORÍA DE MANDOS INFRARROJOS, POSEE 3 PINES DE CONEXIÓN GND, VCC Y DATA , EL CUAL NOS PERMITE CONECTAR DIRECTAMENTE A UN PIN DIGITAL DE NUESTRO ARDUINO O CUALQUIER MICROCONTROLADOR QUE DESEEMOS USAR.
- COMO MANDO USAREMOS UNO QUE VIENE JUNTO AL SENSOR EN EL KIT DEL MANDO INFRARROJO
- ESTE MANDO USA EL PROTOCOLO NEC QUE TRABAJA A 38KHZ DE FRECUENCIA, EL FORMATO DEL TREN DE PULSOS QUE ENVÍA AL PRESIONAR UNA TECLA SE MUESTRA EN LA SIGUIENTE GRÁFICA:

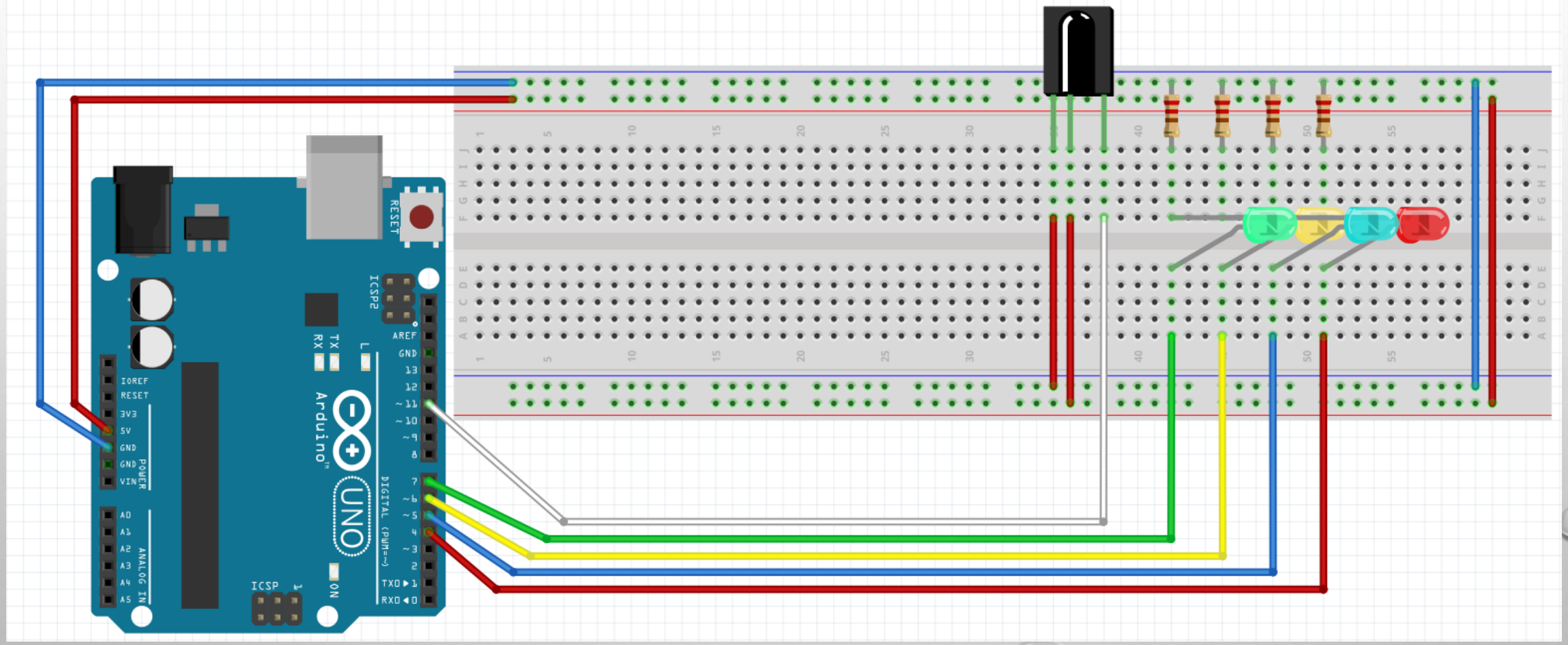


SENSOR IR CON CONTROL REMOTO

- LO PARTICULAR DE ESTE PROTOCOLO ES QUE ENVÍA DOBLE VEZ TANTO LA DIRECCIÓN COMO EL COMANDO, DE FORMA NORMAL Y NEGADO, CON ESTO POSTERIORMENTE SE PUEDE VALIDAR LOS DATOS.
- LA DIRECCIÓN ESTÁ ASOCIADA A UN DISPOSITIVO, POR EJEMPLO UN TV, UNA EQUIPO DE SONIDO, UN VCR, ETC. Y EL COMANDO ESTÁ ASOCIADO A LA ACCIÓN O FUNCIÓN DEL BOTÓN. POR EJEMPLO SUBIR EL VOLUMEN, APAGAR, EL NÚMERO 1 O 2, ETC.

PARA ESTE TUTORIAL VAMOS A TRABAJAR COMO SI SE TRATASE DE UN SOLO BLOQUE DE DATOS 32 BITS.

SENSOR IR CON CONTROL REMOTO



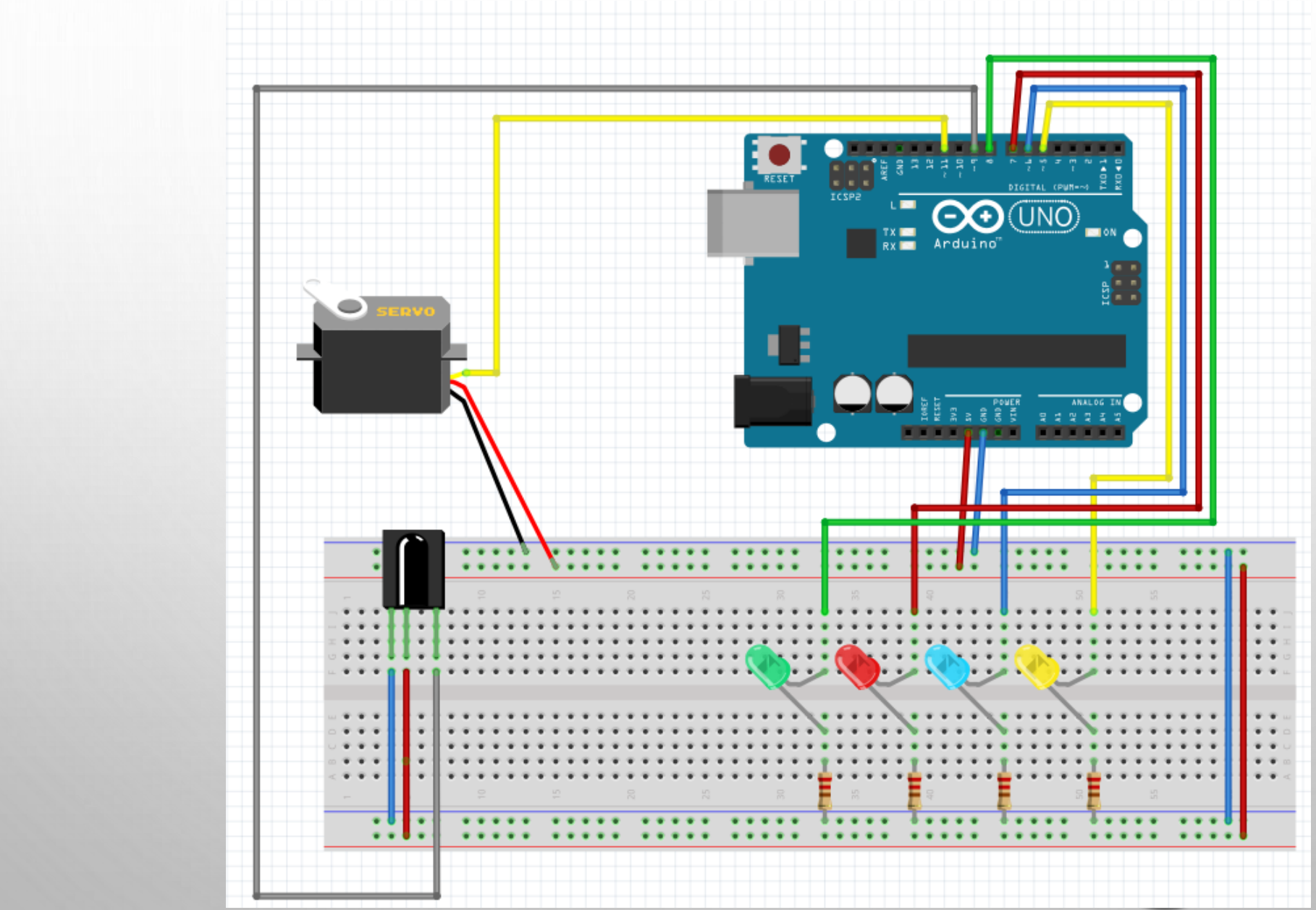
SENSOR IR CON CONTROL REMOTO

```
#include "IRremote.h"
int receiver = 11;

int in1 = 4;
int in2 = 5;
int in3 = 6;
int in4 = 7;

IRrecv irrecv(receiver);
decode_results results;

void setup()
{
  Serial.begin(9600);
  irrecv.enableIRIn();

  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);

}
```

```
void loop()
{
  if (irrecv.decode(&results))

  {
    Serial.println(results.value, HEX);
    switch(results.value)
    {

      case 0xE17A807F: digitalWrite(in1, HIGH);
                       digitalWrite(in2, LOW);
                       digitalWrite(in3, LOW);
                       digitalWrite(in4, LOW);
                  break;
      case 0xE17A40BF:  digitalWrite(in1, LOW);
                        digitalWrite(in2, HIGH);
                        digitalWrite(in3, LOW);
                        digitalWrite(in4, LOW);
                   break;
      case 0xE17AC03F: digitalWrite(in1, LOW);
                       digitalWrite(in2, LOW);
                       digitalWrite(in3, HIGH);
                       digitalWrite(in4, LOW);
                  break;

      case 0xE17A20DF:  digitalWrite(in1, LOW);
                        digitalWrite(in2, LOW);
                        digitalWrite(in3, LOW);
                        digitalWrite(in4, HIGH);
                   break;
```

```
      case 0xE17AA05F:  digitalWrite(in1, LOW);
                        digitalWrite(in2, LOW);
                        digitalWrite(in3, LOW);
                        digitalWrite(in4, LOW);
                  break;
      }
    irrecv.resume();
  }
  delay(300);
}
```

CONTROL REMOTO Y SERVOMOTOR



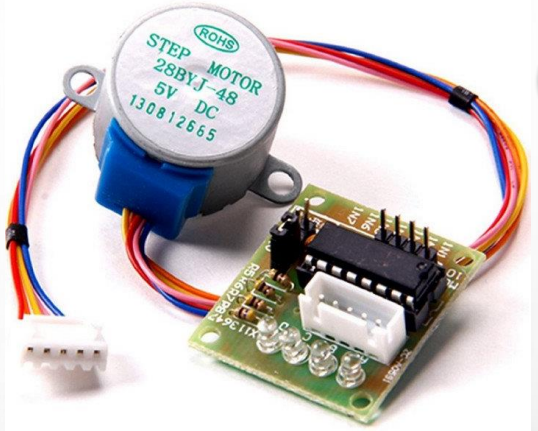
CONTROL REMOTO Y SERVOMOTOR

```
#include <IRremote.h>
#include <Servo.h>


const int receptor = 11; // Declaración de variable
int led1 = 5;
int led2 = 6;
int led3 = 7;
int led4 = 8;


IRrecv irrecv(receptor); //pin 11 sera el receptor
decode_results codigo;
void setup()
{
  Serial.begin(9600);
  irrecv.enableIRIn(); // INICIA LA RECEPCIÓN
servoMotor.attach(9);
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
  pinMode(led4, OUTPUT);

}
void loop()
{
 if (irrecv.decode(&codigo))
 {
Serial.println(codigo.value, HEX);
if (codigo.value==0x00FF6897)  // Codigo 1
    {
      digitalWrite(led1,HIGH);
      digitalWrite(led2,LOW);
      digitalWrite(led3,LOW);
      digitalWrite(led4,LOW);
      servoMotor.write(0);
    }
if (codigo.value==0x00FF9867) // Codigo 2
    {
      digitalWrite(led1,LOW);
      digitalWrite(led2,HIGH);
      digitalWrite(led3,LOW);
      digitalWrite(led4,LOW);
      servoMotor.write(25);
    }
 if (codigo.value==0x00FFB04F) // Codigo 3
    {
      digitalWrite(led1,HIGH);
      digitalWrite(led2,HIGH);
      digitalWrite(led3,LOW);
      digitalWrite(led4,LOW);
      servoMotor.write(45);
    }
 if (codigo.value==0x00FF30CF) // Codigo 4
    {
      digitalWrite(led1,LOW);
      digitalWrite(led2,LOW);
      digitalWrite(led3,HIGH);
      digitalWrite(led4,LOW);
      servoMotor.write(75);
    }
 if (codigo.value==0x00FF18E7) // Codigo 5
    {
      digitalWrite(led1,HIGH);
      digitalWrite(led2,LOW);
      digitalWrite(led3,HIGH);
      digitalWrite(led4,LOW);
      servoMotor.write(90);
    }
if (codigo.value==0x00FF7A85) // Codigo 6
    {
      digitalWrite(led1,LOW);
      digitalWrite(led2,HIGH);
      digitalWrite(led3,HIGH);
      digitalWrite(led4,LOW);
       servoMotor.write(110);
     }
    if (codigo.value==0x00FF10EF) // Codigo 7
    {
      digitalWrite(led1,HIGH);
      digitalWrite(led2,HIGH);
      digitalWrite(led3,HIGH);
      digitalWrite(led4,LOW);
       servoMotor.write(145);
     }

if (codigo.value==0x00FF38C7) // Codigo 8
    {
      digitalWrite(led1,LOW);
      digitalWrite(led2,LOW);
      digitalWrite(led3,LOW);
      digitalWrite(led4,HIGH);
       servoMotor.write(150);
    }
    if (codigo.value==0x00FF5AA5) // Codigo 9
    {
      digitalWrite(led1,HIGH);
      digitalWrite(led2,LOW);
      digitalWrite(led3,LOW);
      digitalWrite(led4,HIGH);
       servoMotor.write(165);
     }
if (codigo.value==0x00FF4AB5) // Codigo 10
    {
      digitalWrite(led1,LOW);
      digitalWrite(led2,HIGH);
      digitalWrite(led3,LOW);
      digitalWrite(led4,HIGH);
       servoMotor.write(180);
     }
    delay(500);
    irrecv.resume();
 }
```

ALTERNATIVA DE DISEÑO

Proponer alternativas de programación En lenguaje C++

MOTOR PASO A PASO

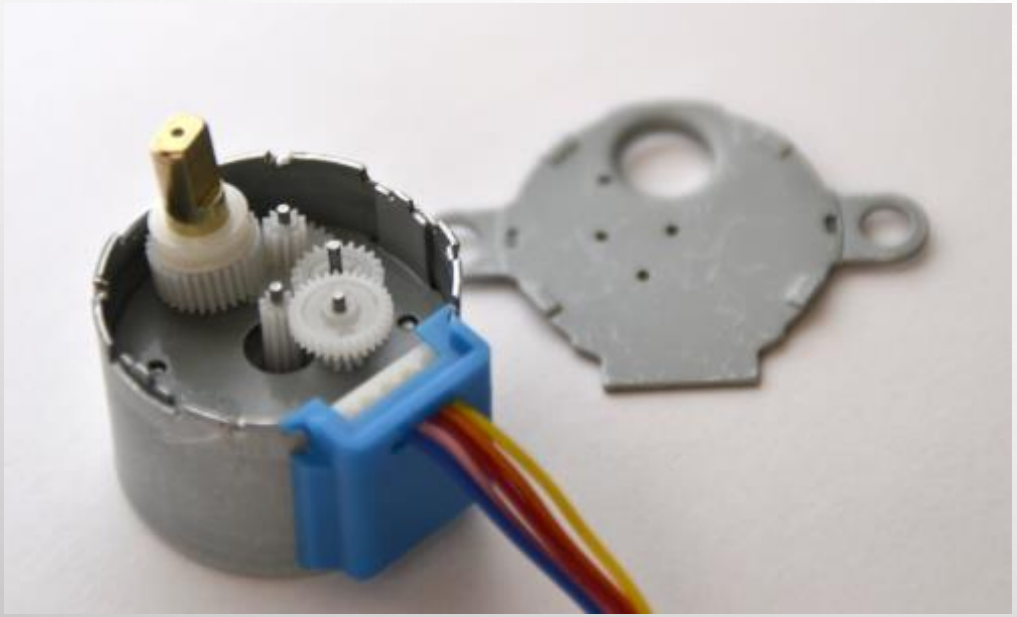


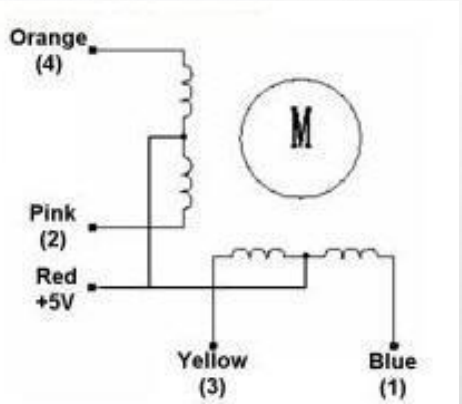
UN MOTOR PASO A PASO ES UN DISPOSITIVO ELECTROMECÁNICO QUE CONVIERTE PULSOS ELÉCTRICOS EN MOVIMIENTOS MECÁNICOS DISCRETOS. EL EJE DE UN MOTOR PASO A PASO GIRA EN INCREMENTOS DISCRETOS CUANDO IMPULSOS DE MANDO ELÉCTRICO SE APLICAN A ÉL EN LA SECUENCIA CORRECTA.

LA SECUENCIA DE LOS PULSOS APLICADOS SE RELACIONA DIRECTAMENTE CON LA DIRECCIÓN DE ROTACIÓN DE EJES MOTOR. LA VELOCIDAD DE LA ROTACIÓN DE LOS EJES MOTOR ESTÁ DIRECTAMENTE RELACIONADA CON LA FRECUENCIA DE LOS PULSOS DE ENTRADA Y LA DURACIÓN DE LA ROTACIÓN ESTÁ DIRECTAMENTE RELACIONADA CON EL NÚMERO DE PULSOS DE ENTRADA APLICADA.

UNA DE LAS VENTAJAS MÁS IMPORTANTES DE UN MOTOR PASO A PASO ES SU CAPACIDAD PARA SER CONTROLADO CON PRECISIÓN EN UN SISTEMA DE LAZO ABIERTO. CONTROL DE LAZO ABIERTO SIGNIFICA QUE NINGUNA INFORMACIÓN DE RETROALIMENTACIÓN DE POSICIÓN ES NECESARIO. ESTE TIPO DE CONTROL ELIMINA LA NECESIDAD DE COSTOSOS DISPOSITIVOS DE DETECCIÓN Y REGENERACIÓN COMO CODIFICADORES ÓPTICOS.

MOTOR PASO A PASO

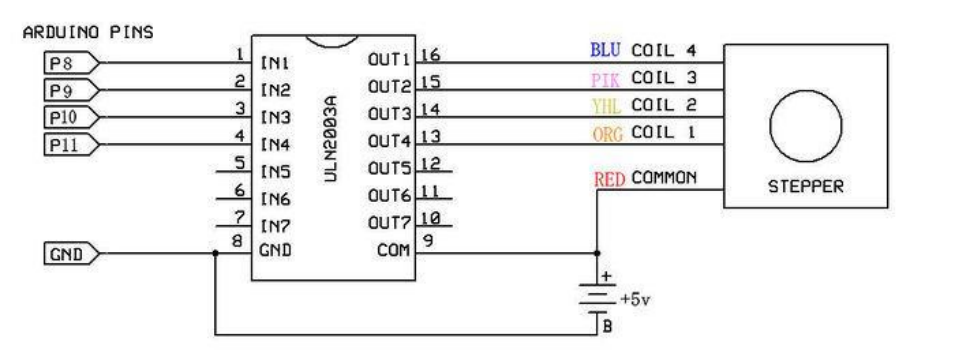




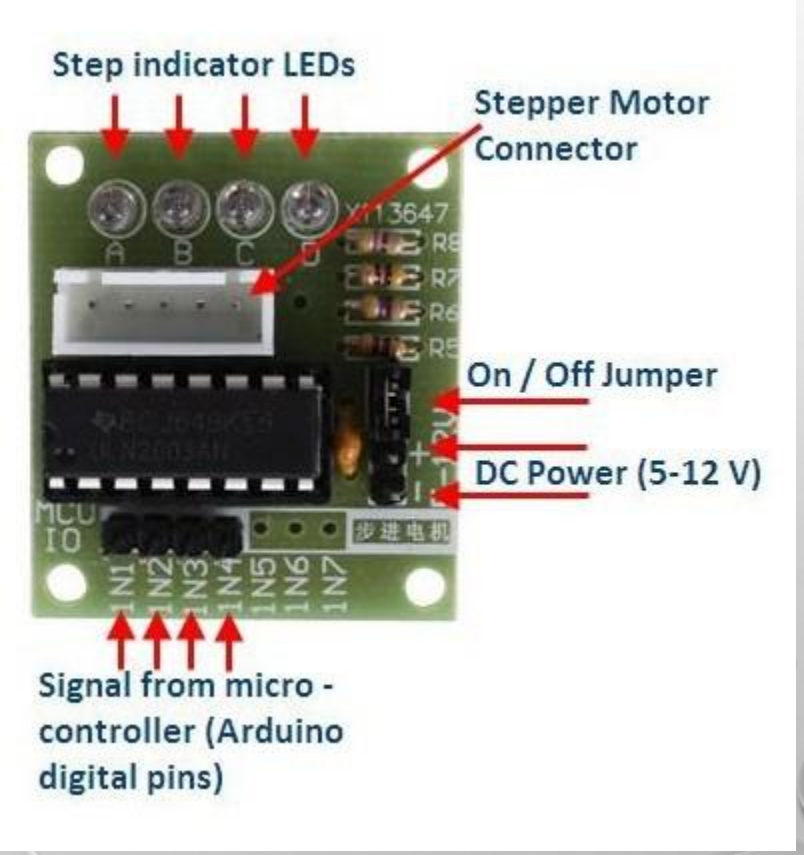
Half-Step Switching Sequence

Lead Wire Color	---> CW Direction (1-2 Phase)							
	1	2	3	4	5	6	7	8
4 Orange	-	-						-
3 Yellow		-	-	-				
2 Pink				-	-	-		
1 Blue						-	-	-

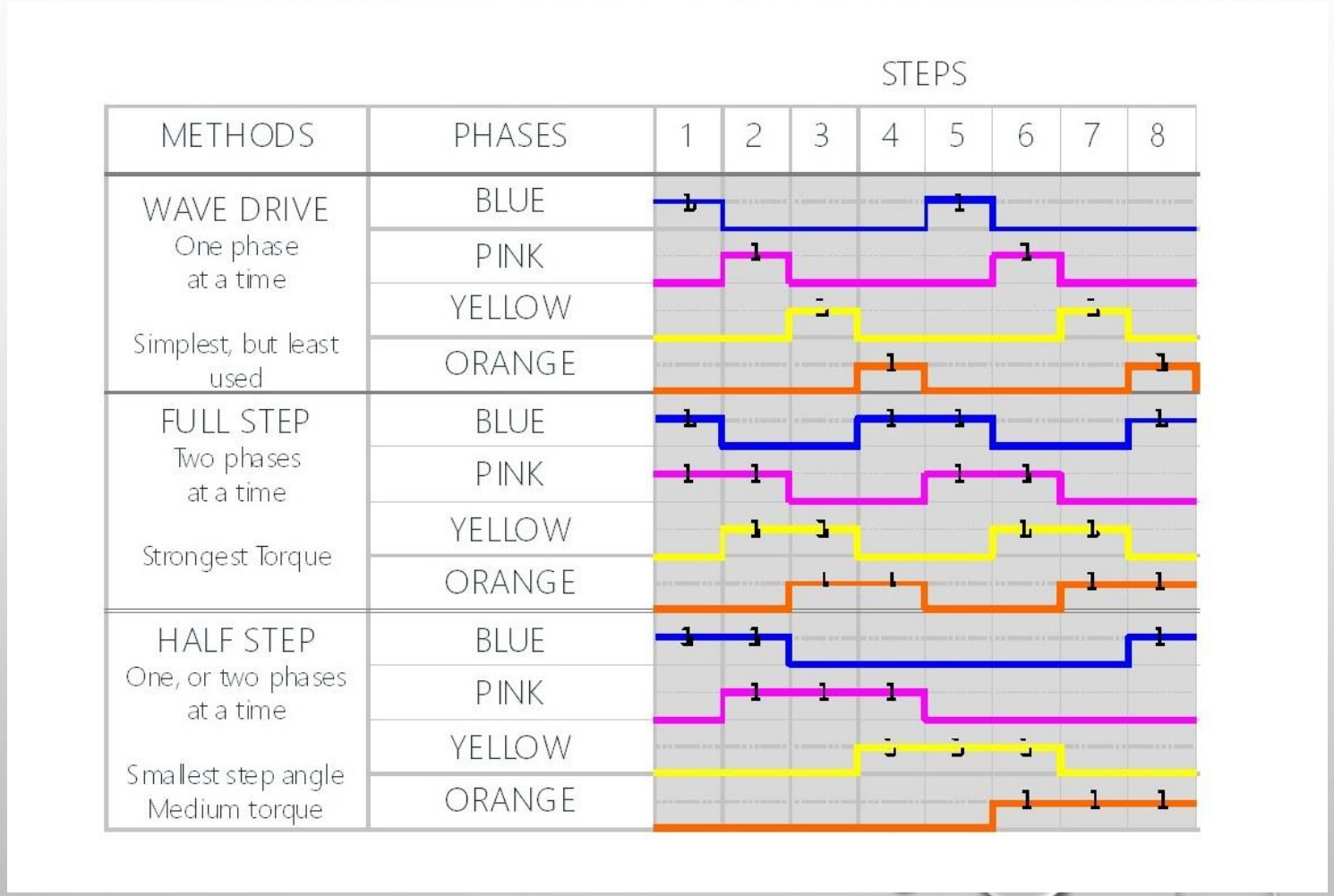
MOTOR PASO A PASO



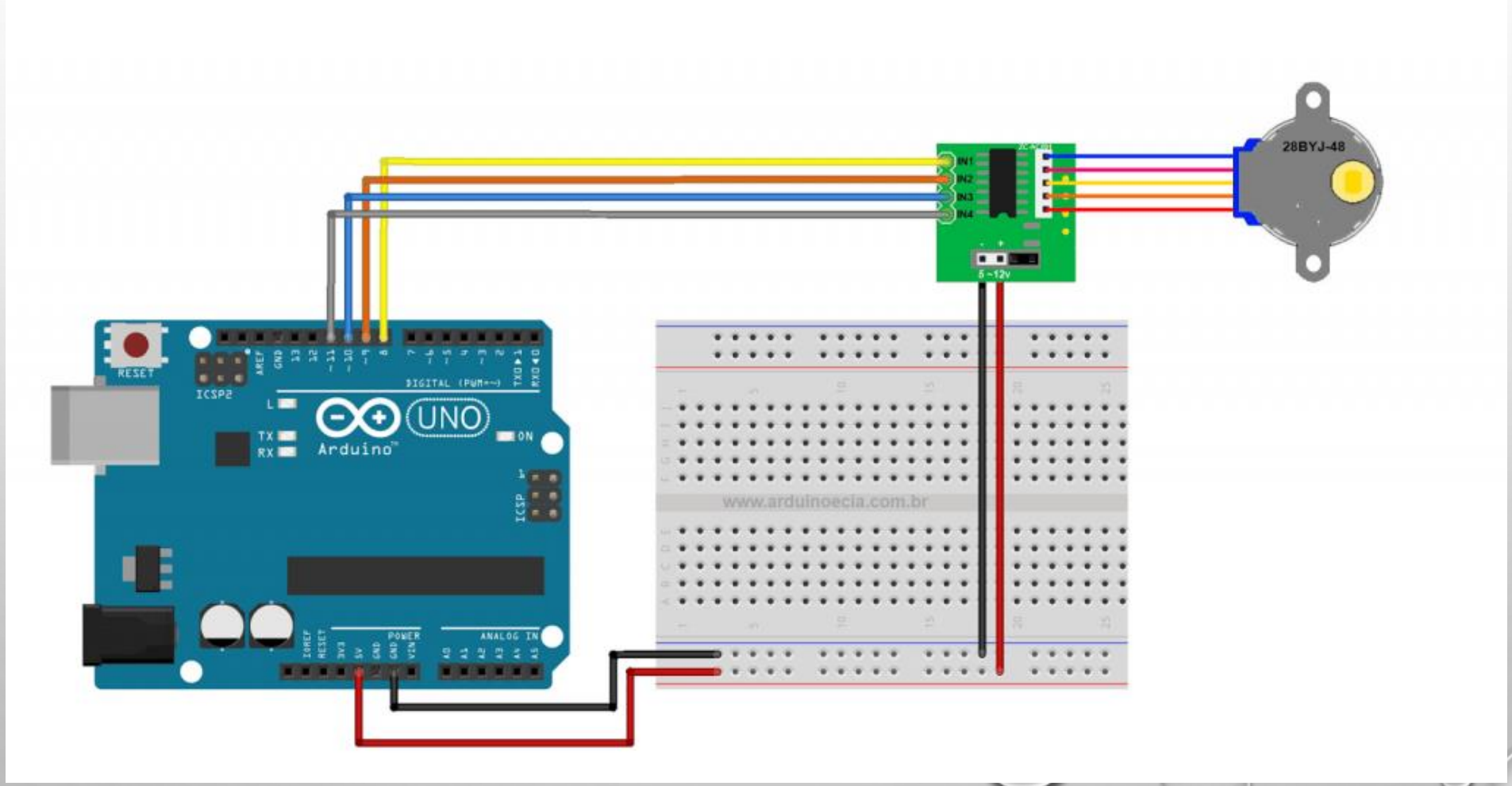
El motor de pasos bipolar tiene generalmente cuatro cables que salen de él. A diferencia de los motores PAP unipolares, steppers bipolares no tienen ninguna conexión común de centro. Tienen dos juegos independientes de bobinas en lugar de otro. Se pueden distinguir de steppers unipolares midiendo la resistencia entre los cables. Debe encontrar dos pares de cables de igual resistencia. Si tienes las puntas de su medidor conectado a dos cables que no están conectados (es decir, no conectada a la bobina del mismo), debería ver resistencia infinita (o sin continuidad).



MOTOR PASO A PASO – METODO DE CONTROL



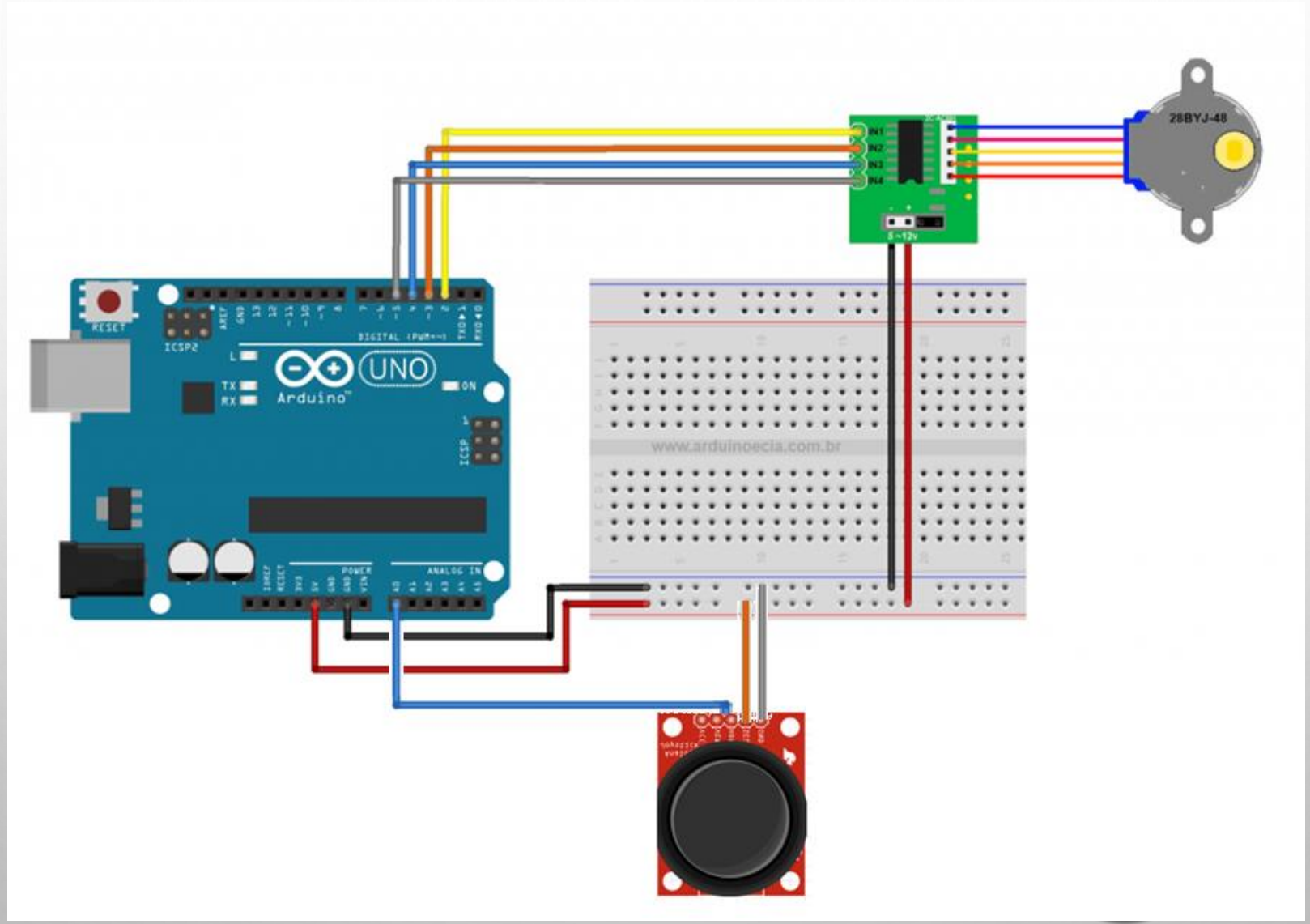
MOTOR PASO A PASO- CONEXIÓN



MOTOR PASO A PASO- PROGRAMA

```
#include <Stepper.h>
#define STEPS 2048

Stepper stepper(STEPS, 8, 9, 10, 11);

void setup() {
  stepper.setSpeed(10);
}

void loop() {
  stepper.step(2048);
}
```

MOTOR PASO A PASO CON JOISTICK



MOTOR PASO A PASO CON JOISTICK

```
#INCLUDE <STEPPER.H>

STEPPER STEPPER_1(4,2,4,3,5);

VOID SETUP()

{

SERIAL.BEGIN(9600);

PINMODE(A0, INPUT);

PINMODE(2, OUTPUT);

PINMODE(3, OUTPUT);

PINMODE(4, OUTPUT);

PINMODE(5, OUTPUT);

}
```

```
void loop()
{
    if ((analogRead(A0) == 1023)) {
      stepper_1.step((-1));
      delay(2);
    } else {
      digitalWrite(2, LOW);
      digitalWrite(3, LOW);
      digitalWrite(4, LOW);
      digitalWrite(5, LOW);
    }
    if ((analogRead(A0) == 0)) {
      stepper_1.step(1);
      delay(2);
    }
}
```

DESARROLLO DE PROGRAMA LIBRE

MyRoboTech
Robotica aplicada para todas las edades