SERIAL COMMUNICATION_QUICK RECAP

serialEvent_read()

ARDUINO CODE:

```
int val = 0;
void setup()
{
 Serial.begin(9600);
}
void loop()
{
  val = analogRead(0);
  Serial.print(val);
  Serial.println();
}
```

```
import processing.serial.*;
Serial myPort;        // Create object from Serial class
int val;              // Data received from the serial port
int lf = 10;          // ascii code for linefeed
void setup()
{
 size(200, 200);
 // I know that the first port in the serial list on my mac
 // is always my  FTDI adaptor, so I open Serial.list()[0].
 // On Windows machines, this generally opens COM1.
 // Open whatever port is the one you're using.
 String portName = Serial.list()[0];
 myPort = new Serial(this, portName, 9600);
}
void draw()
{
 background(255);                    // Set background to white
 fill(val);
 rect(50, 50, 100, 100);
}

void serialEvent(Serial p)
{
 myPort.bufferUntil(lf);
 val = p.read();
 println("received: "+ val);
}
```

serialEvent_read()

ARDUINO: 0 - 1023

```
long map(long x, long in_min, long in_max, long out_min, long out_max)
{
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
```

PROCESSING: 0 - 255

arduino:

169 (int)
168

processing:

without myPort.bufferUntil(lf):

received: 53 (int)
received: 13
received: 10
received: 49
received: 56
received: 53
received: 13
received: 10
received: 49
received: 56

with myPort.bufferUntil(lf):

received: 10(int)
received: 49
received: 56
received: 53
received: 13
received: 10
received: 49
received: 56
received: 54
received: 13

serialEvent_readString()

ARDUINO CODE:

```
int val = 0;
void setup()
{
 Serial.begin(9600);
}
void loop()
{
  val = analogRead(0);
  Serial.print(val);
  Serial.println();
}
```

PROCESSING CODE:

```
import processing.serial.*;
Serial myPort;                  // Create object from Serial class
String inString;                // Data received from the serial port
float val;
int lf = 10;                    // ascii code for linefeed
void setup()
{
 size(200, 200);
 String portName = Serial.list()[0];
 myPort = new Serial(this, portName, 9600);
 myPort.clear();
}
void draw()
{
 background(255);               // Set background to white
 val = map(val, 0, 1023, 0, 255);
 fill(int(val));                //typecast float to int
 rect(50, 50, 100, 100);
}
/*built in function within processing called when data is available
which can be set with buffer() to only trigger after a certain number
of data elements are read and can be set with bufferUntil() to only
trigger after a specific character is read to be used with one of the
Read() functions*/
void serialEvent(Serial p)
{
 myPort.bufferUntil(lf);        //buffer until linefeed (println)
 inString = p.readString();
 print("received: "+inString);
 val = float(inString);         //typecast string type to float
 println( val);
}
```

serialEvent_readString()

arduino print:

186 (int)
186
186

without myPort.bufferUntil(lf):

received: (string)

NaN(not a number) = null/linefeed
received: 1
1.0 (float)
received: 8
8.0
received: 6
6.0
received:

NaN
received:

with myPort.bufferUntil(lf):

received: 186 (string)
186.0 (float)
received: 186
186.0
received: 186
186.0

```
void serialEvent(Serial p)
{
  myPort.bufferUntil(lf);
  inString = p.readString();
  print("received: "+inString);
  val = float(inString);
  println( val);
}
```

serialEvent_readBytes()

ARDUINO CODE:

```
int val = 0;
void setup()
{
 Serial.begin(9600);
}
void loop()
{
  val = analogRead(0);
  Serial.print(val);
  Serial.println();
}
```

PROCESSING CODE:

```
import processing.serial.*;
Serial myPort;                  // Create object from Serial class
byte[] inBuffer;
float val;
int lf = 10;                    // ascii code for linefeed
void setup()
{
 size(200, 200);
 String portName = Serial.list()[0];
 myPort = new Serial(this, portName, 9600);
 myPort.clear();
}
void draw()
{
 background(255);               // Set background to white
 val = map(val, 0, 1023, 0, 255);
 fill(int(val));                //typecast float to int
 rect(50, 50, 100, 100);
}
//readByte
void serialEvent(Serial p)
{
 myPort.bufferUntil(lf);
 inBuffer = myPort.readBytes();
 println("received: "+ inBuffer);
 if (inBuffer != null)
 {
   String myString = new String(inBuffer);
   val = float(myString);
   println(val);
 }
}
```

serialEvent_readBytes()

arduino print:

180(int)
181
181
180

without myPort.bufferUntil(lf):

received: [B@1113ac8 (hex)
NaN(not a number) = null/linefeed
received: [B@6f26bb
1.0 (float)
received: [B@157c76a
8.0
received: [B@1e53a48
0.0
received: [B@556aa9
NaN

with myPort.bufferUntil(lf):

received: [B@1fc25e5 (hex)
180.0 (float)
received: [B@11c19e6
181.0
received: [B@1e88b35
181.0

```
void serialEvent(Serial p)
{
 myPort.bufferUntil(lf);
 inBuffer = myPort.readBytes();
 println("received: "+ inBuffer);
 if (inBuffer != null)
 {
   String myString = new String(inBuffer);
   val = float(myString);
   println(val);
 }
}
```

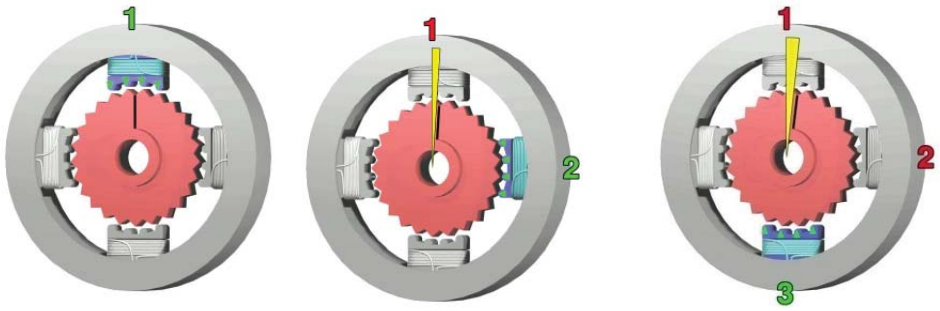
MOTORS

Steppers
DC motors
Servos

TYPES

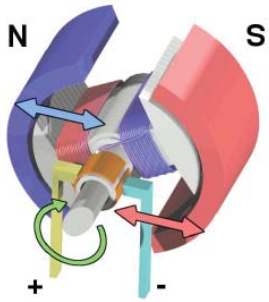
Steppers

_stepped movement



DC motors

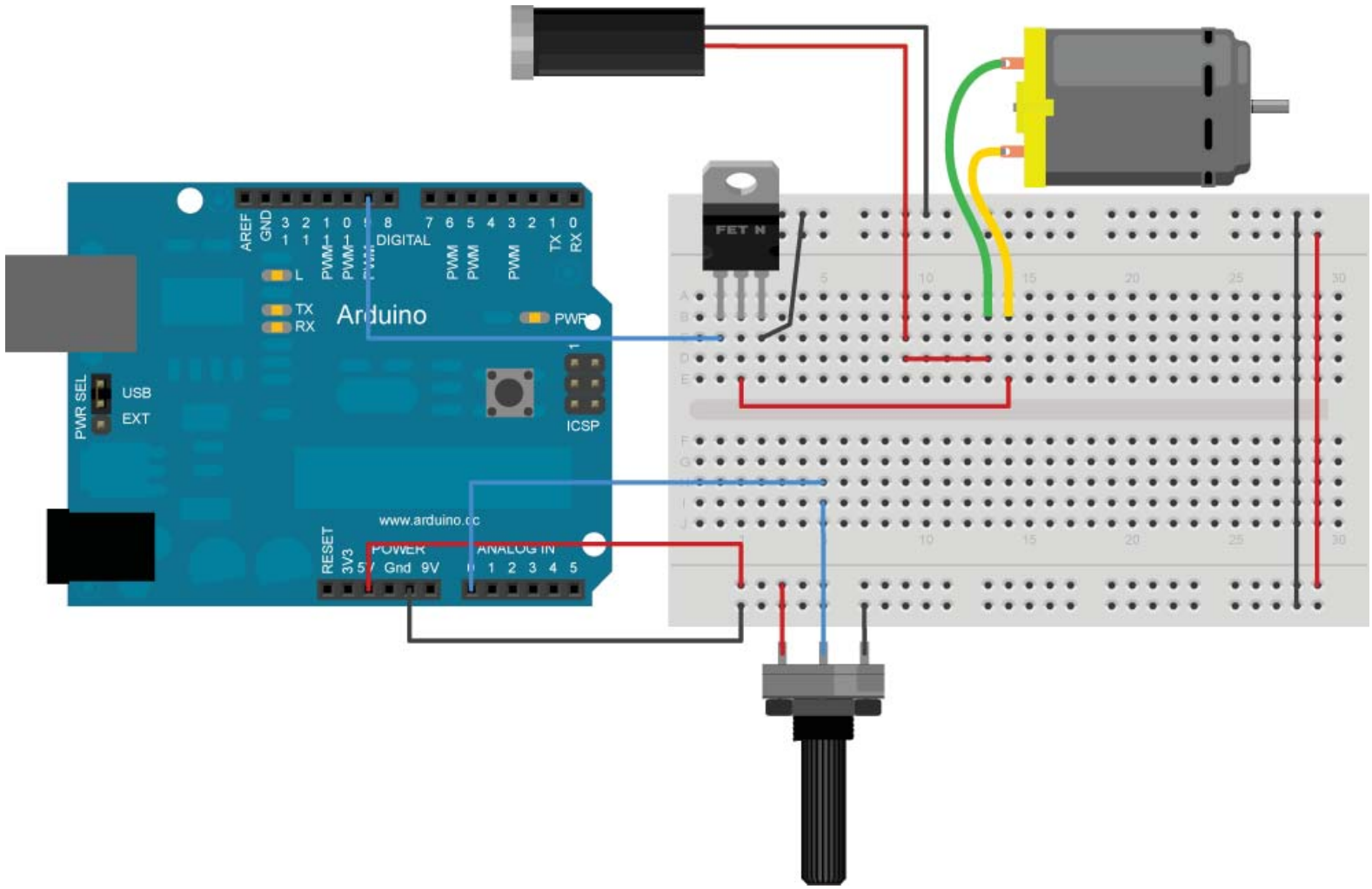
_continuous movement



Servos

_tracked stepped movement

DC Motor


FET N
Arduino
PWR
TX
RX
L
USB
EXT
PWR SEL
ICSP
DIGITAL
AREF
GND
PWM
www.arduino.cc
RESET
3V3
Gnd
9V

DC Motor

```
const int transistorPin = 9;    // connected to the base of the transistor

void setup() {
  // set  the transistor pin as output:
  pinMode(transistorPin, OUTPUT);
}

void loop() {
  digitalWrite(transistorPin, HIGH);
  delay(1000);
  digitalWrite(transistorPin, LOW);
  delay(1000);
}
```

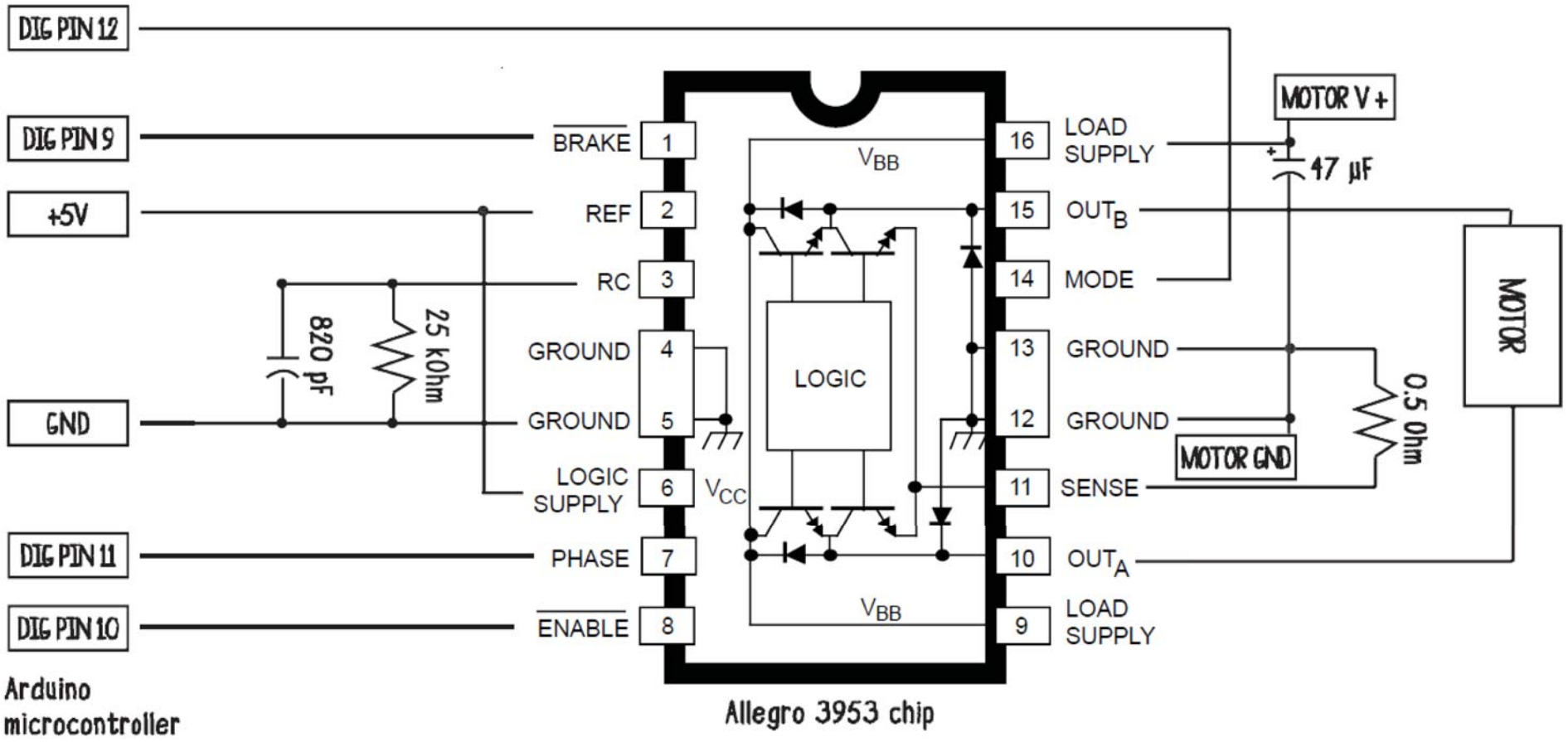
Now that you see it working, try changing the speed of the motor using the potentiometer. Try this code:

```
const int potPin = 0;          // Analog in 0 connected to the potentiometer
const int transistorPin = 9;    // connected to the base of the transistor
int potValue = 0;              // value returned from the potentiometer

void setup() {
  // set  the transistor pin as output:
  pinMode(transistorPin, OUTPUT);
}

void loop() {
  // read the potentiometer, convert it to 0 - 255:
  potValue = analogRead(potPin) / 4;
  // use that to control the transistor:
  analogWrite(9, potValue);
}
```

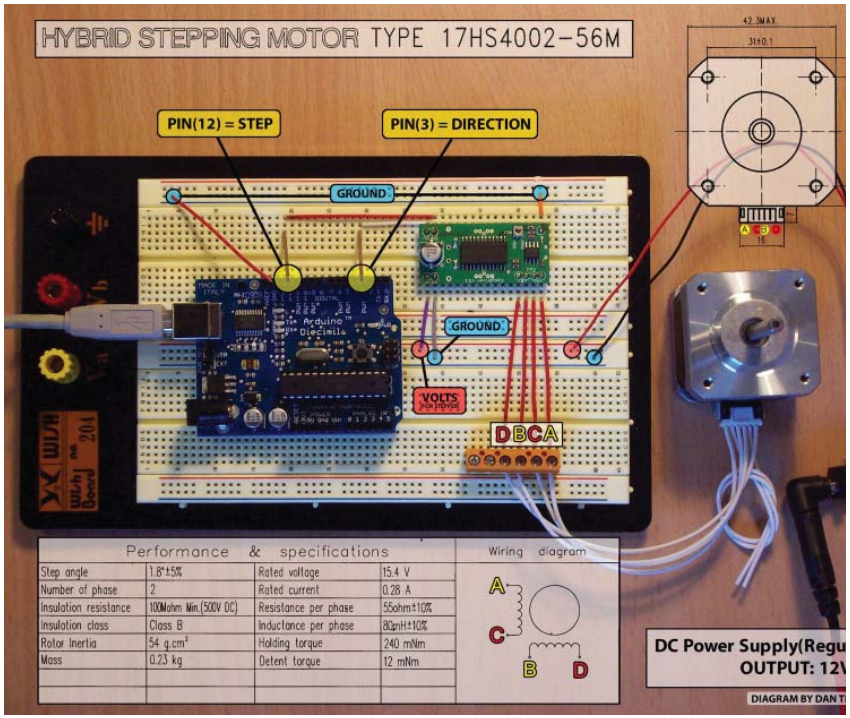
MICROCONTROLLER A3953SBT

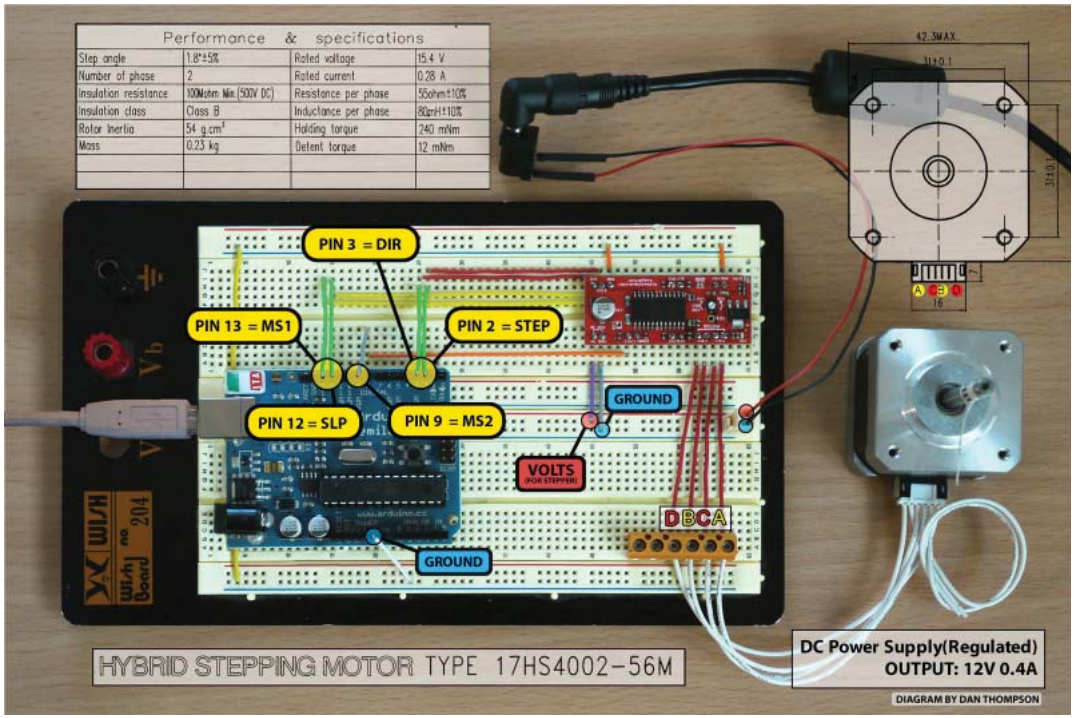


DC MOTOR CODE

```
/*
 Stepper Motor Control
A simple example of how to use the A3955 with an Arduino
*/
int brake = 9;
int enable = 10; // Must be a PWM pin
int phase = 11;
int mode = 12;
int mstatus;
void setup() // Set all of our pins as outputs
{ // Alternatively, we can address this
pinMode(brake, OUTPUT); // bank of pins as a port, as shownat
pinMode(enable, OUTPUT); // < http://www.arduino.cc/en/
pinMode(phase, OUTPUT); // Reference/PortManipulation >
pinMode(mode, OUTPUT); // This would be port B.
}
int stop()
{
        digitalWrite(brake, LOW); // Turn on the dynamic brake
        digitalWrite(mode, LOW); // Use Phase pin for current protection
        // The remaining pins are irrelevant
        return 0;
}
int forward(int mspeed, int mstat)
{
        if (mstat == -1)
        {
                return 1;
        } // Do not change directions w/o abrake
        digitalWrite(brake, HIGH); // Disable the dynamic brake
        analogWrite(enable, mspeed); // Send a PWM signal to enable the motor
        digitalWrite(phase, HIGH); // Drive the motor forward
        digitalWrite(mode, HIGH); // Fast decay mode
        return 1;
}
int reverse(int mspeed, int mstat)
{
        if (mstat == 1)
        {
                return -1;
        } // Do not change directions w/o a brake
        digitalWrite(brake, HIGH); // Disable the dynamic brake
        analogWrite(enable, mspeed); // Send a PWM signal to enable the motor
        digitalWrite(phase, LOW); // Drive the motor in REVERSE
        digitalWrite(mode, HIGH); // Fast decay mode
        return -1;
}
void loop()
{
        for (int i = 0; i <= 100; i++)
        {
                // Drive the motor forward from 0-100%
                mstatus = forward(i,mstatus); // call the forward function
                delay(50);
        } // wait between cycles to make this last 5 sec
        mstatus = stop(); // sink both motor leads low, making it stop
        delay(1000); // wait one second
        for (int i = 100; i>= 0; i--)
        {
                // Drive the motor backwards from 100-0%
                mstatus = forward(i,mstatus); // call the reverse function
                delay(50); // wait between cycles to make this last 5 sec
        }
        mstatus = stop(); // sink both motor leads low, making it stop
        delay(1000); // wait one second
}
```

STEPPERS_EASYDRIVER





STEPPERS_EASYDRIVER

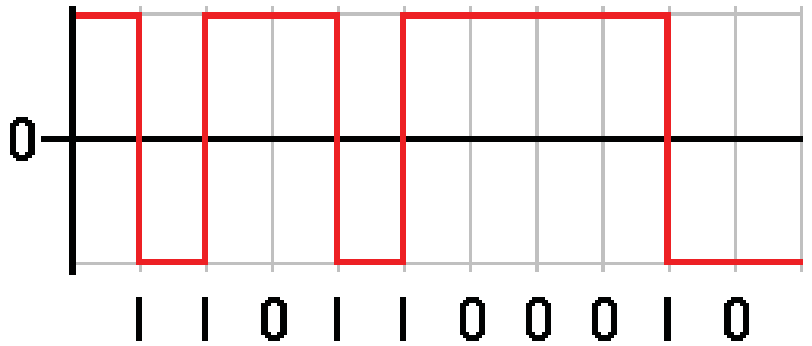
```
int dirpin = 3;
int steppin = 12;
void setup()
{
  Serial.begin(9600);
  pinMode(dirpin, OUTPUT);
  pinMode(steppin, OUTPUT);
}
void loop()
{
  int i;
  digitalWrite(dirpin, LOW);              // Set the direction.
  delay(100);
  Serial.println(“>>”);
  for (i = 0; i<4000; i++)                // Iterate for 4000 microsteps.
  {
    digitalWrite(steppin, LOW);           // This LOW to HIGH change is what creates the
    digitalWrite(steppin, HIGH);          // “Rising Edge” so the easydriver knows to when to step.
    delayMicroseconds(200);               // This delay time is close to top speed for this
  }                                       // particular motor. Any faster the motor stalls.
  digitalWrite(dirpin, HIGH);             // Change direction.
  delay(100);
  Serial.println(“<<”);
  for (i = 0; i<4000; i++)                // Iterate for 4000 microsteps
  {
    digitalWrite(steppin, LOW);           // This LOW to HIGH change is what creates the
    digitalWrite(steppin, HIGH);          // “Rising Edge” so the easydriver knows to when to step.
    delayMicroseconds(200);               // This delay time is close to top speed for this
  }                                       // particular motor. Any faster the motor stalls.
}
```

RISING/FALLING EDGE

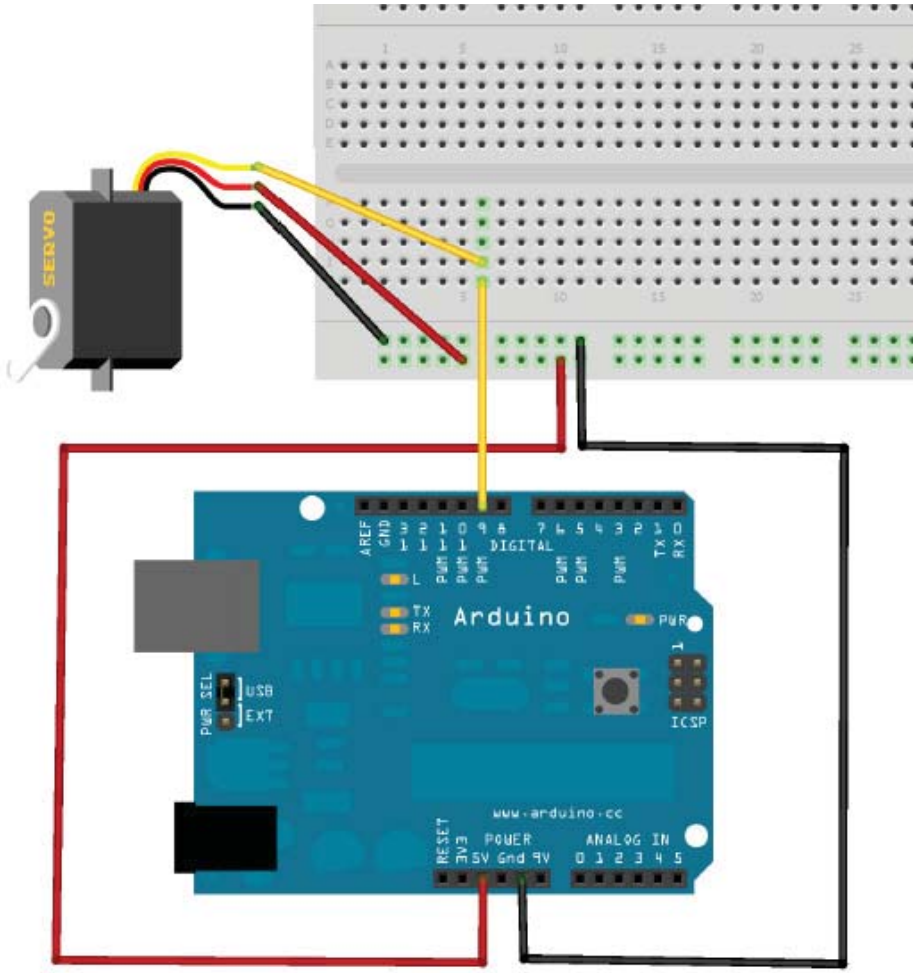
In electronics, a **clock edge** is a transition in a clock signal from either low to high (0 to 1) or high to low (1 to 0). It is called an “edge” because the square wave which represents a clock has edges at those points.

A **rising edge** is the transition of a digital signal from low to high. It is also named positive edge. When a circuit’s rising edge triggered, it becomes active when the clock signal goes from **low to high, and ignores the high to low transition.**

A **falling edge** is the high to low transition. It’s also known as the negative edge. When a circuit’s falling edge triggered, it becomes active when the clock signal goes from **high to low, and ignores the low to high transition.**



SERVO


SERVO
AREF
GND
DIGITAL
PWM
TX
RX
L
Arduino
PWR
PWR SEL
USB
EXT
ICSP
www.arduino.cc
RESET
3V3
5V
POWER
Gnd
9V
ANALOG IN
0 1 2 3 4 5

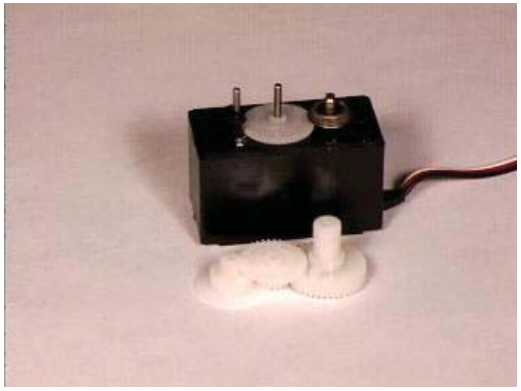
SERVO MOTOR (old version code)

```
/* Servo Example
* Code based on AnalogInput by DojoDave <http://www.0j0.org>
*/
int potPin = 2; // select the input pin for the potentiometer
int val = 0; // variable to store the value coming from the sensor
int servoPin = 7; // Control pin for servo motor
int pulseWidth = 0; // Amount to pulse the servo
long lastPulse = 0; // the time in millisecs of the last pulse
int refreshTime = 20; // the time in millisecs needed in between pulses
int minPulse = 600; // minimum pulse width
void setup()
{
        pinMode(potPin, INPUT);
        pinMode(servoPin, OUTPUT);
}
void loop()
{
        val = analogRead(potPin); // read the value from the sensor
        val = val * .175; // convert 10 bit val to derees (1024/180 = .175 )
        pulseWidth = (val * 10) + minPulse; // convert angle to microseconds
        updateServo(); // update servo position
        delay(10);
}
void updateServo()
 {
        // pulse the servo again if rhe refresh time (20 ms) have passed:
        if (millis() - lastPulse >= refreshTime)
         {
                digitalWrite(servoPin, HIGH); // Turn the motor on
                delayMicroseconds(pulseWidth); // Length of the pulse sets the motor position
                digitalWrite(servoPin, LOW); // Turn the motor off
                lastPulse = millis(); // save the time of the last pulse
        }
}
```

HACKING SERVO_180 to 360

1) 

Disassamble by using a screw driver to open housing

2) 

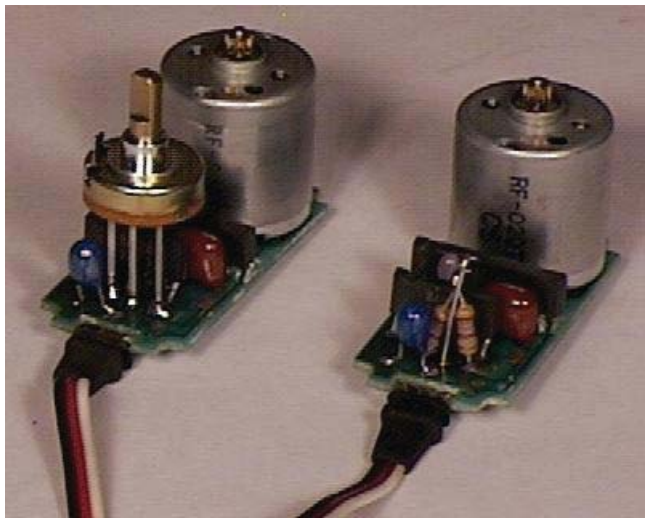
Remove the gears to reach the servo motor and potentiometer

3) 

Desolder the potentiometer

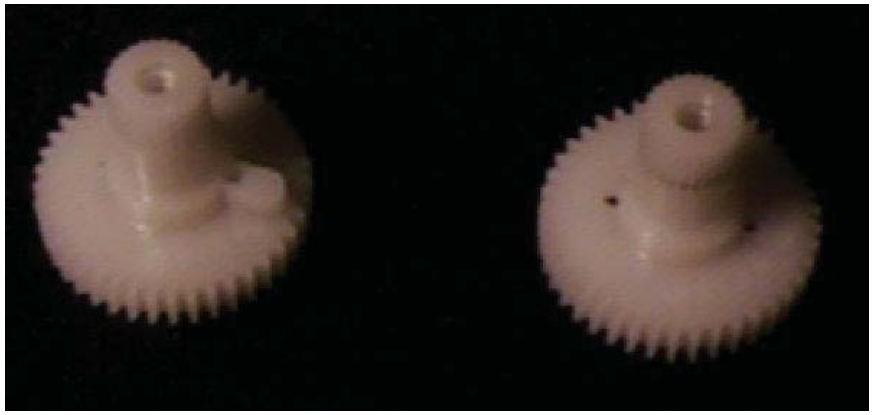
HACKING SERVO_180 to 360

4)



Replace the potentiometer with a resistor network by soldering the resistors that were attached to the potentiometer together

5)



Cut the stopper of the gear and reassemable