Experiments with Arduino

Swagat Kumar

Overview

- Introduction
  - Few Basics about Resistors, Diodes, Transistors, PWMs etc.
- Lab Experiments
  - Blinking LED
  - Using a Push Button - Two experiments
  - Serial Communication
  - Serial Com + LED + Push Button – Two experiments
  - Fading LED: PWM + LED
  - Servo Control
  - Speed Control using PWM – Two experiments
  - Multiplexing – Two experiments.

Introduction

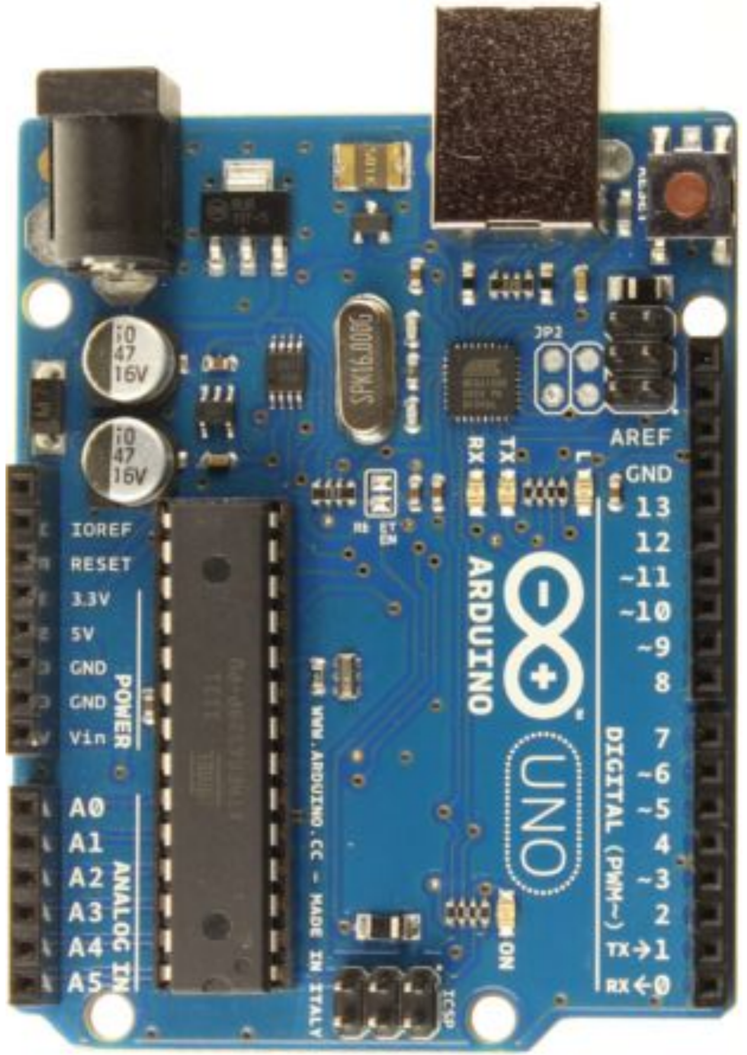
- What is Arduino?
  Arduino is an open-source electronics platform based on easy-to-use hardware and software.

- Why Arduino?
  - Platform independent
  - Easy to use: Simple and clear programming environment
  - Low cost
  - Simplifies the process of working with microcontrollers
  - Open source and extensible software & hardware

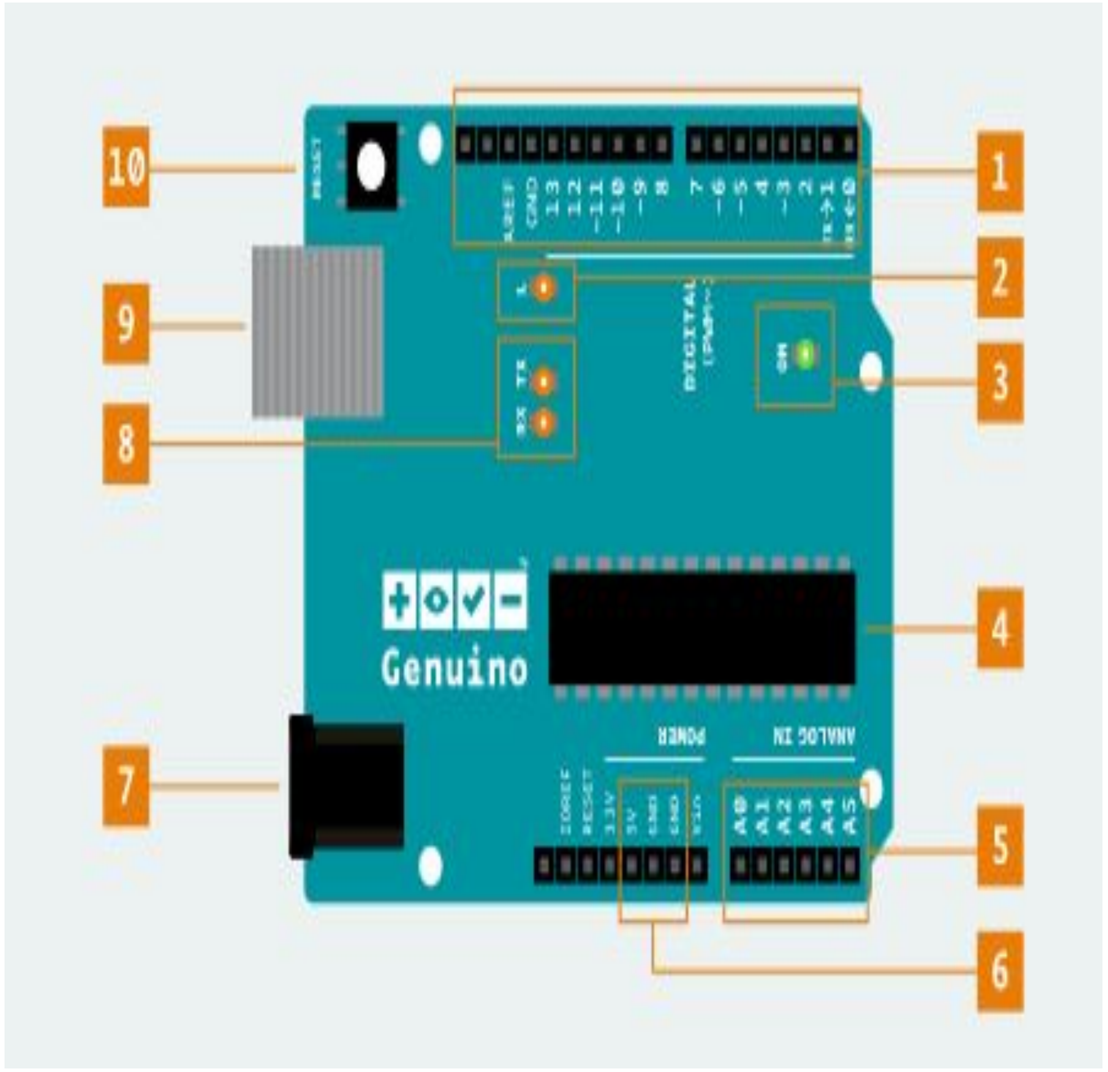
- Arduino homepage: https://www.arduino.cc/
- Arduino IDE Installation Link.
- Resistor Color Code Link.

Arduino UNO Anatomy

- 1. Digital pins Use these pins with digitalRead(), digitalWrite(), and analogWrite(). analogWrite() works only on the pins with the PWM symbol.
- 2. Pin 13 LED The only actuator built-in to your board. Besides being a handy target for your first blink sketch, this LED is very useful for debugging.
- 3. Power LED Indicates that your Genuino is receiving power. Useful for debugging.
- 4. ATmega microcontroller The heart of your board.
- 5. Analog in Use these pins with analogRead().
- 6. GND and 5V pins Use these pins to provide +5V power and ground to your circuits.
- 7. Power connector This is how you power your Genuino when it’s not plugged into a USB port for power. Can accept voltages between 7-12V.
- 8. TX and RX LEDs These LEDs indicate communication between your Genuino and your computer. Expect them to flicker rapidly during sketch upload as well as during serial communication. Useful for debugging.
- 9. USB port Used for powering your Genuino Uno, uploading your sketches to your Genuino, and for communicating with your Genuino sketch (via Serial. println() etc.).
- 10. Reset button Resets the ATmega microcontroller.



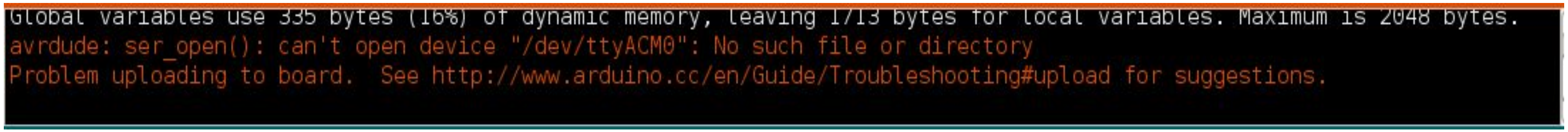
ARDUINO
UNO
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
IOREF
RESET
3.3V
5V
GND
GND
Vin
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
RX
TX
L
ON
ICSP
JP2
WWW.ARDUINO.CC - MADE IN ITALY



10
9
8
7
1
2
3
4
5
6
Genuino
RESET
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
L
TX
RX
ON
POWER
ANALOG IN
IOREF
RESET
3.3V
5V
GND
GND
Vin
A0
A1
A2
A3
A4
A5

Microcontroller	ATmega328P – 8 bit AVR family microcontroller
Operating Voltage	5V
Recommended Input Voltage	7-12V
Input Voltage Limits	6-20V
Analog Input Pins	6 (A0 – A5)
Digital I/O Pins	14 (Out of which 6 provide PWM output)
DC Current on I/O Pins	40 mA
DC Current on 3.3V Pin	50 mA
Flash Memory	32 KB (0.5 KB is used for Bootloader)
SRAM	2 KB
EEPROM	1 KB
Frequency (Clock Speed)	16 MHz

In Linux, If you are getting error about not able to open a device, you need provide read / write permission to this serial port.

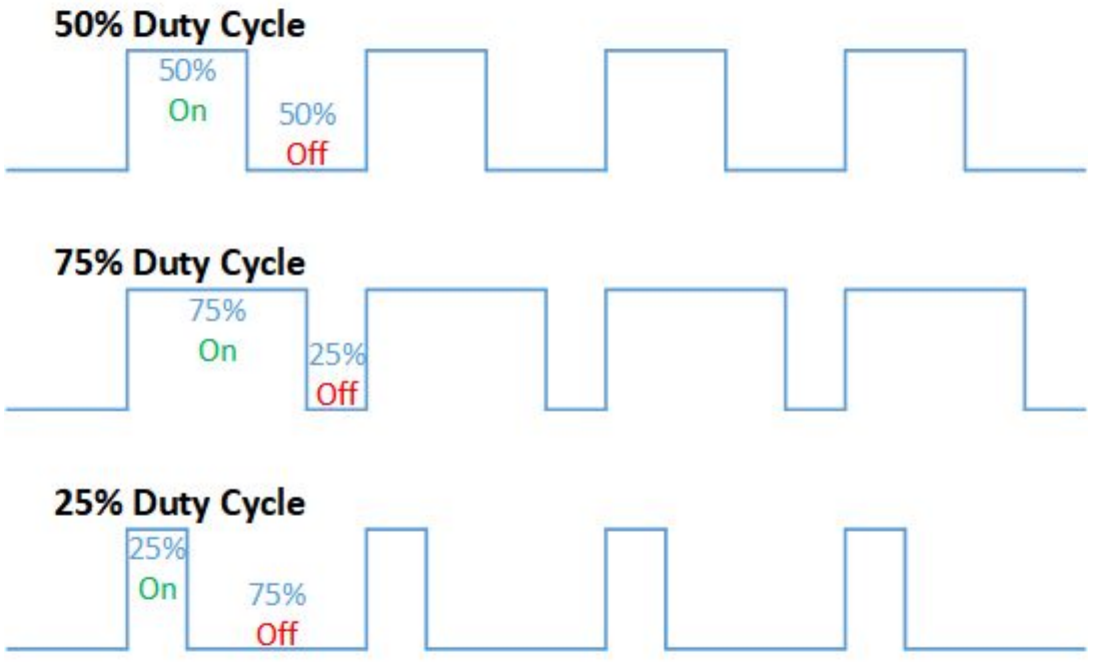


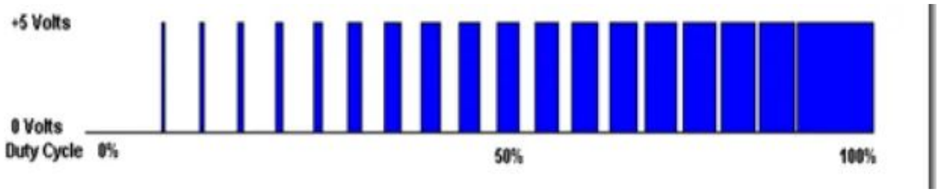
```
Global variables use 335 bytes (16%) of dynamic memory, leaving 1713 bytes for local variables. Maximum is 2048 bytes.
avrdude: ser_open(): can't open device "/dev/ttyACM0": No such file or directory
Problem uploading to board.  See http://www.arduino.cc/en/Guide/Troubleshooting#upload for suggestions.
```

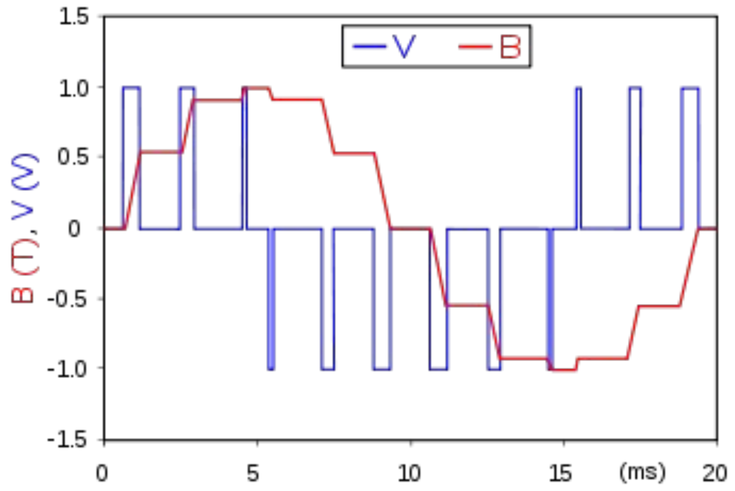
```
$ sudo chmod a+rw /dev/ttyACM0
```

Pulse-Width-Modulation (PWM)

- Pulse width modulation (PWM), or pulse-duration modulation (PDM), is a method of reducing the average power delivered by an electrical signal, by effectively chopping it up into discrete parts.
- The average value of voltage (and current) fed to the load is controlled by turning the switch between supply and load on and off at a fast rate.
- The longer the switch is on compared to the off periods, the higher the total power supplied to the load.
- PWM is particularly suited for running inertial loads such as motors, which are not as easily affected by this discrete switching, because they have inertia to react slow.
- The PWM switching frequency has to be high enough not to affect the load, which is to say that the resultant waveform perceived by the load must be as smooth as possible.







An example of PWM in an idealized inductor driven by a (red) voltage source modulated as a series of pulses, resulting in a blue sine-like current in the inductor. The rectangular voltage pulses nonetheless result in a more and more smooth current waveform, as the switching frequency increases. Note that the current waveform is the integral of the voltage waveform.

- The term ***duty cycle*** describes the proportion of 'on' time to the regular interval or 'period' of time.
- A low duty cycle corresponds to low power, because the power is off for most of the time.
- Duty cycle is expressed in percent, 100% being fully on.
- When a digital signal is on half of the time and off the other half of the time, the digital signal has a duty cycle of 50% and resembles a "square" wave.

PWM Applications

- Speed control of motors
  - Variable speed fan controllers.
  - VRF HVAC compressor drives.
  - Hybrid and electric vehicle motor drive circuits.

- LED Dimmers.

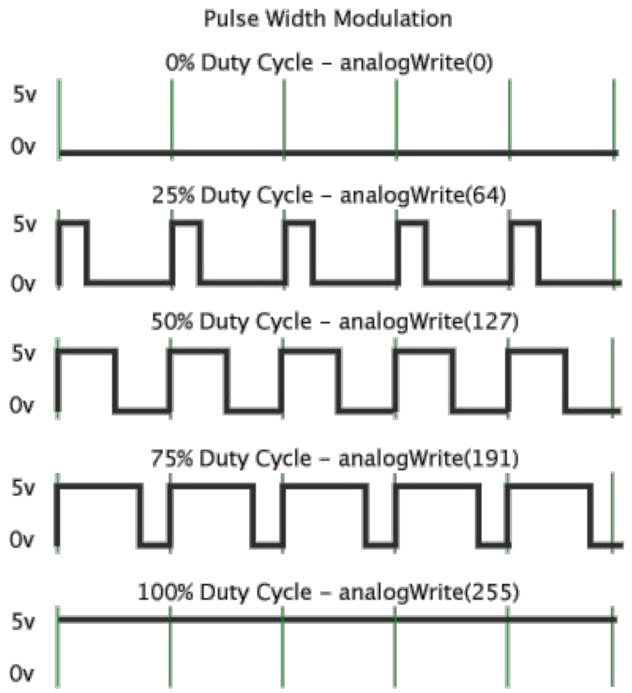
- Pulse width modulation has changed the world by slashing the power consumption of appliances utilizing motors such as inverter air conditioners, inverter refrigerators, inverter washing machines, among many others.
  - For example, inverter air conditioners can consume less than half the energy that their non-inverter counterparts do in some cases.

PWM Control using Arduino

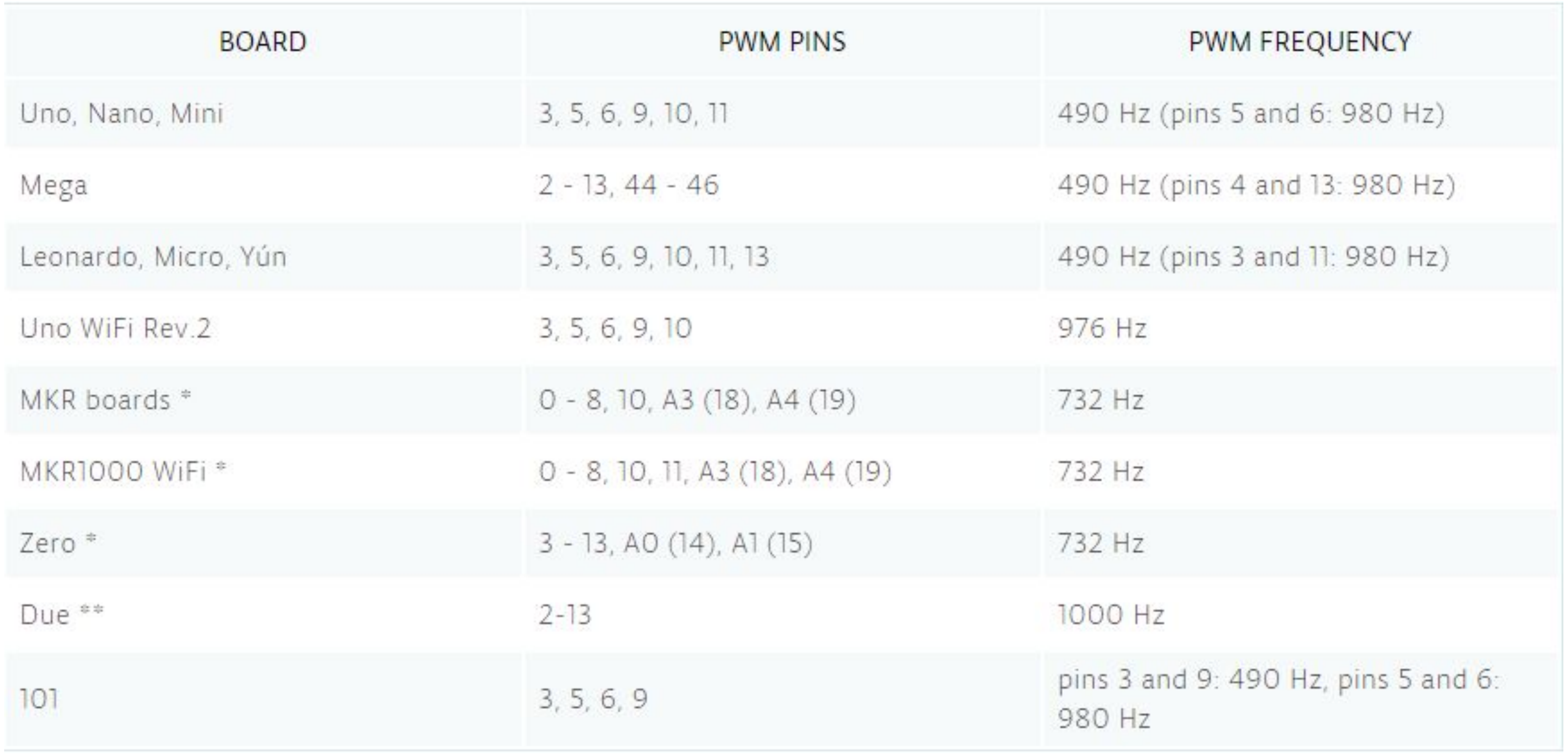
- Arduino UNO PWM Frequency is about 500 Hz - 1 KHz. This gives a time interval of 2 milliseconds.

$$f = \frac{1}{T}$$

- A call to analogWrite() is on a scale of 0 - 255.
    - `analogWrite(255)` requests a 100% duty cycle (always on),
    - `analogWrite(127)` is a 50% duty cycle (on half the time) for example.
- Various PWM Implementations of Arduino is available at the following link: https://www.arduino.cc/en/Tutorial/SecretsOfArduinoPWM



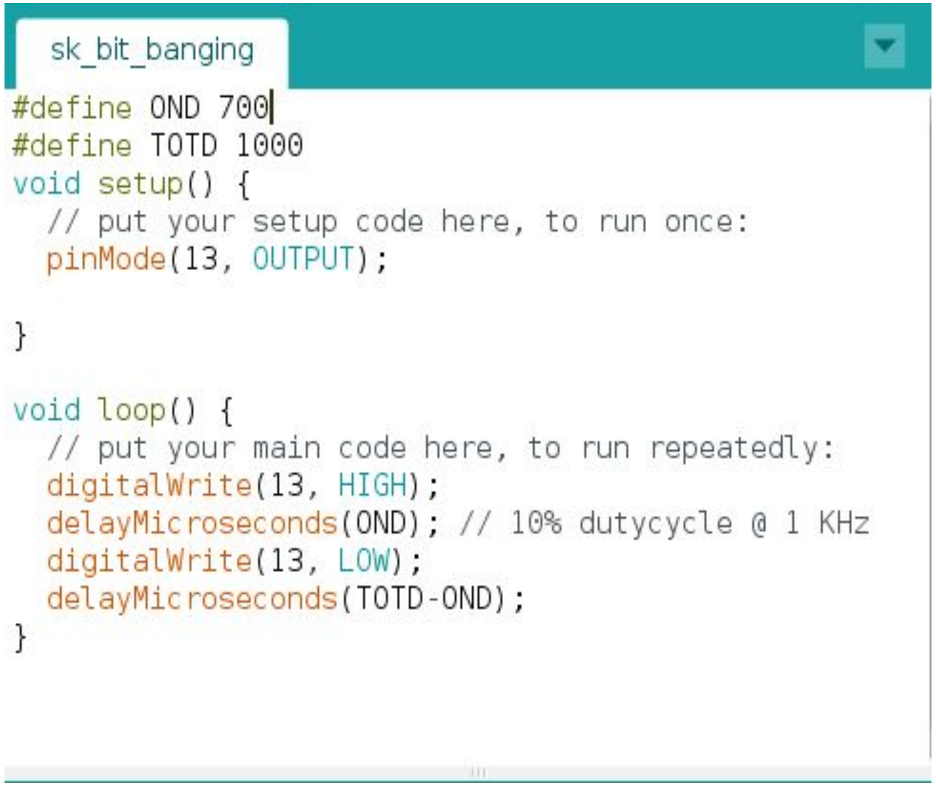
Pulse Width Modulation
0% Duty Cycle – analogWrite(0)
25% Duty Cycle – analogWrite(64)
50% Duty Cycle – analogWrite(127)
75% Duty Cycle – analogWrite(191)
100% Duty Cycle – analogWrite(255)
5v
0v



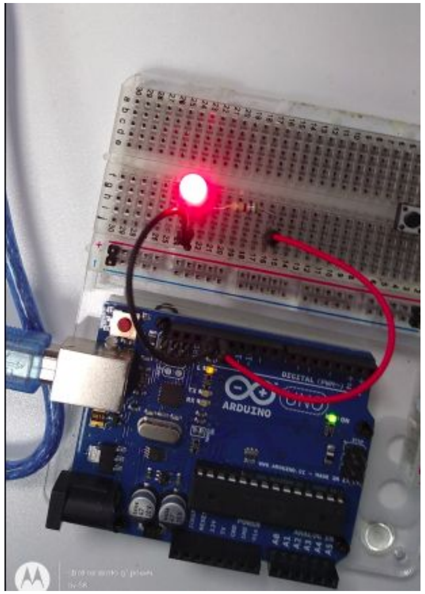
BOARD	PWM PINS	PWM FREQUENCY
Uno, Nano, Mini	3, 5, 6, 9, 10, 11	490 Hz (pins 5 and 6: 980 Hz)
Mega	2 - 13, 44 - 46	490 Hz (pins 4 and 13: 980 Hz)
Leonardo, Micro, Yún	3, 5, 6, 9, 10, 11, 13	490 Hz (pins 3 and 11: 980 Hz)
Uno WiFi Rev.2	3, 5, 6, 9, 10	976 Hz
MKR boards *	0 - 8, 10, A3 (18), A4 (19)	732 Hz
MKR1000 WiFi *	0 - 8, 10, 11, A3 (18), A4 (19)	732 Hz
Zero *	3 - 13, A0 (14), A1 (15)	732 Hz
Due **	2-13	1000 Hz
101	3, 5, 6, 9	pins 3 and 9: 490 Hz, pins 5 and 6: 980 Hz

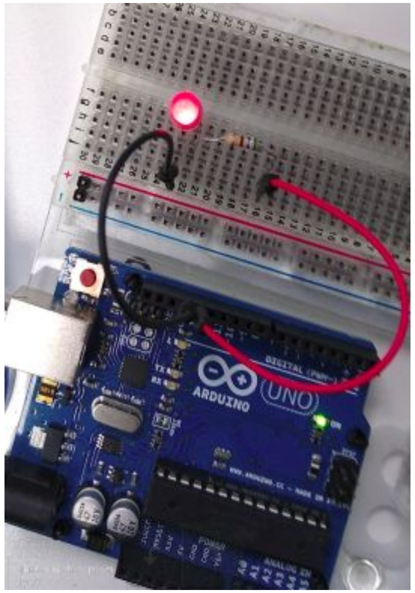
It is also possible to change some of the PWM frequencies.

A simple implementation of PWM: Bit-Banging


sk_bit_banging

```
#define OND 700
#define TOTD 1000
void setup() {
  // put your setup code here, to run once:
  pinMode(13, OUTPUT);

}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(13, HIGH);
  delayMicroseconds(OND); // 10% dutycycle @ 1 KHz
  digitalWrite(13, LOW);
  delayMicroseconds(TOTD-OND);
}
```

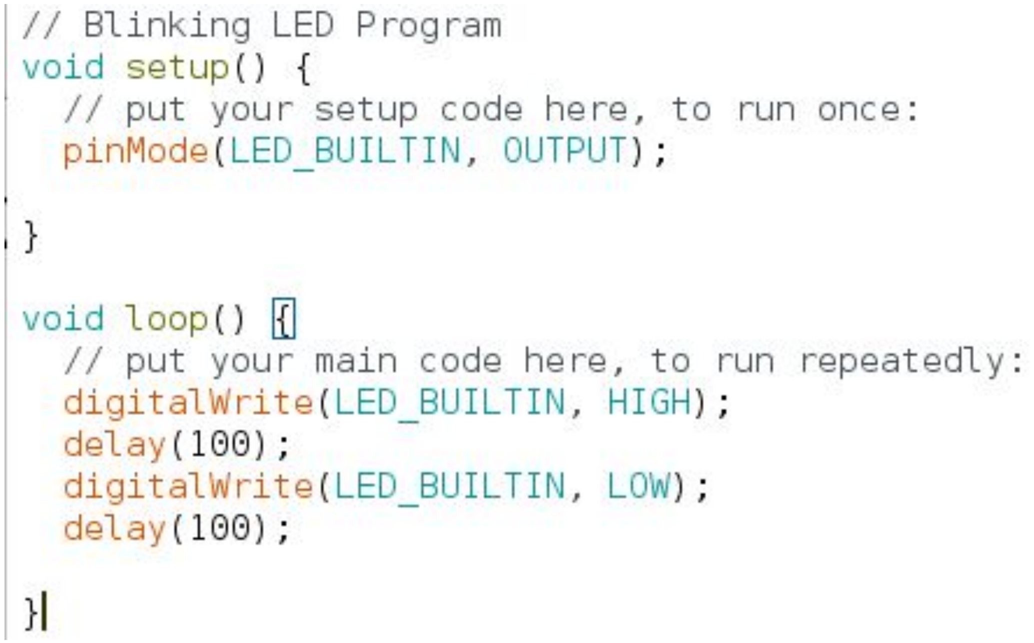

OND = 700


OND = 10

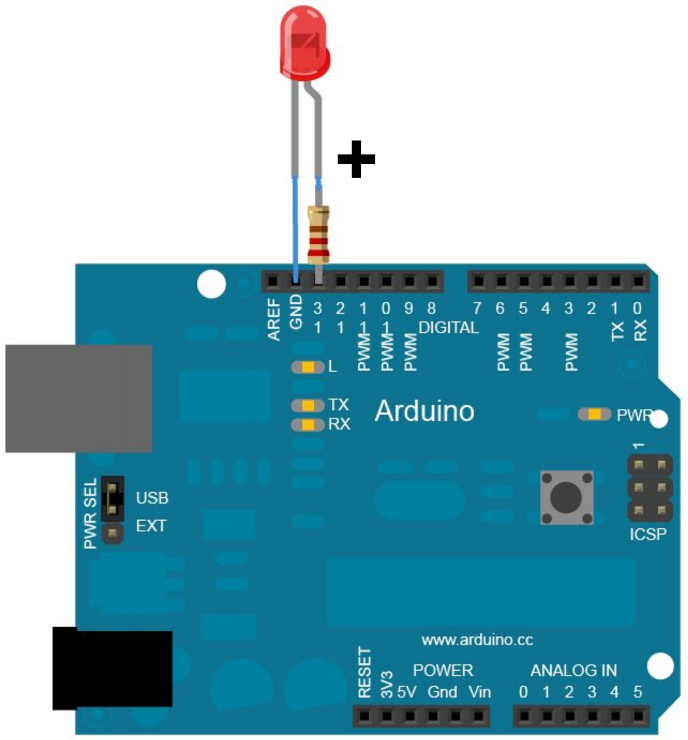
You can play around the value of OND to get different levels of illumination.

Blinking LEDs

Resistor: 200 – 500 ohm

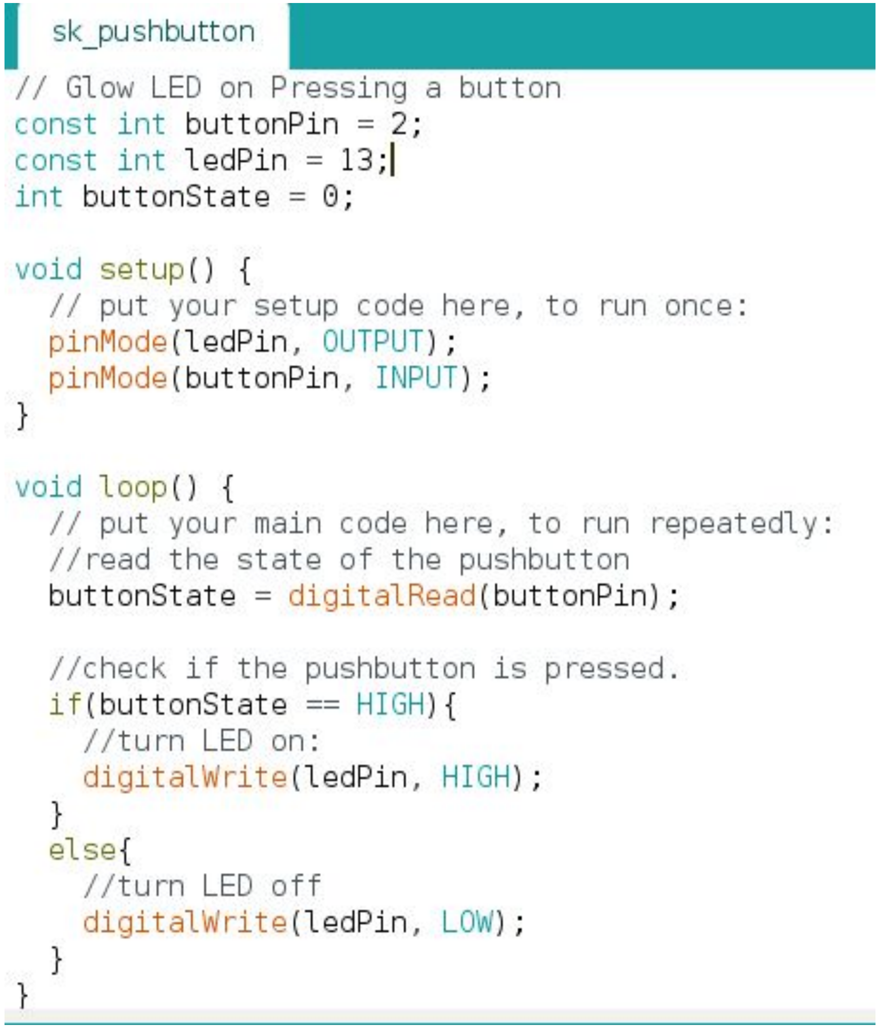


```
// Blinking LED Program
void setup() {
  // put your setup code here, to run once:
  pinMode(LED_BUILTIN, OUTPUT);

}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(LED_BUILTIN, HIGH);
  delay(100);
  digitalWrite(LED_BUILTIN, LOW);
  delay(100);

}
```



D13 – LED Output – Anode leg (long)
GND – Cathode leg (short)

Push Button to Control LED



sk_pushbutton

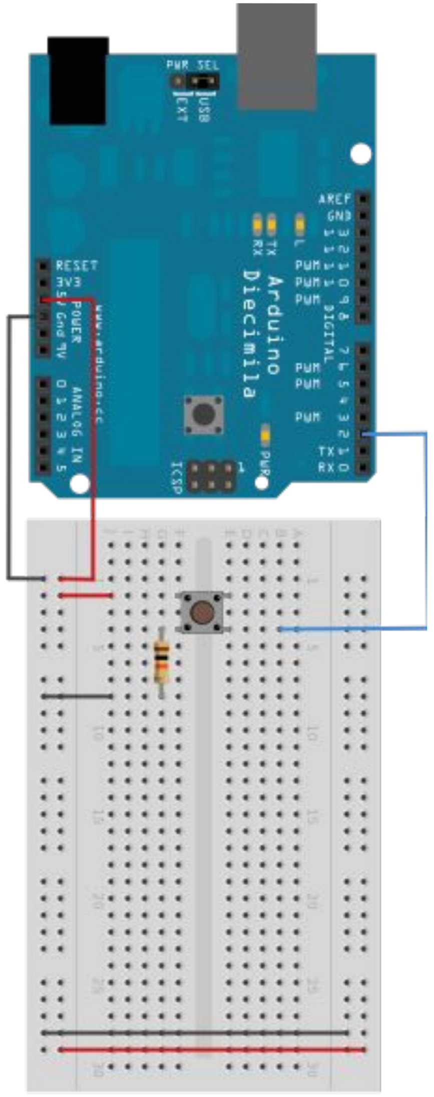
```
// Glow LED on Pressing a button
const int buttonPin = 2;
const int ledPin = 13;
int buttonState = 0;

void setup() {
  // put your setup code here, to run once:
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  //read the state of the pushbutton
  buttonState = digitalRead(buttonPin);

  //check if the pushbutton is pressed.
  if(buttonState == HIGH){
    //turn LED on:
    digitalWrite(ledPin, HIGH);
  }
  else{
    //turn LED off
    digitalWrite(ledPin, LOW);
  }
}
```

Power PINS:
- 5V
- GND

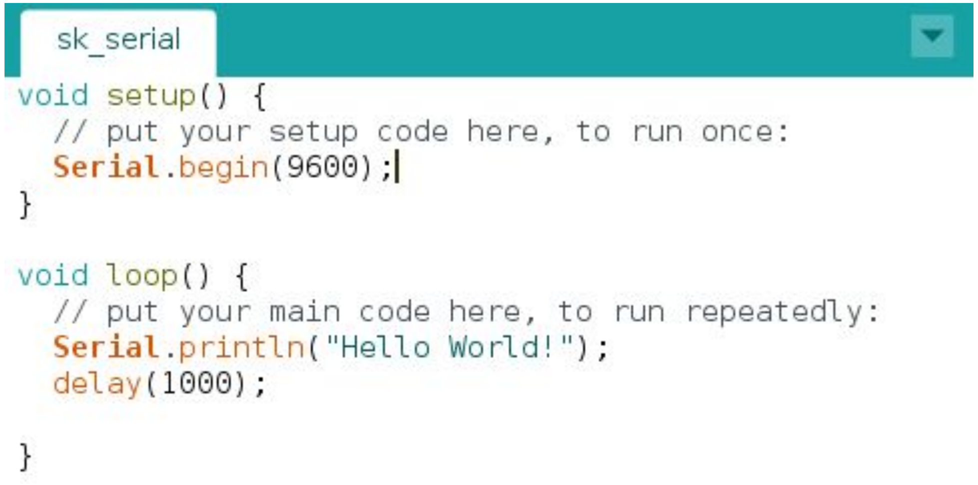
Resistor: 10KΩ

Digital PIN: 2 (input)

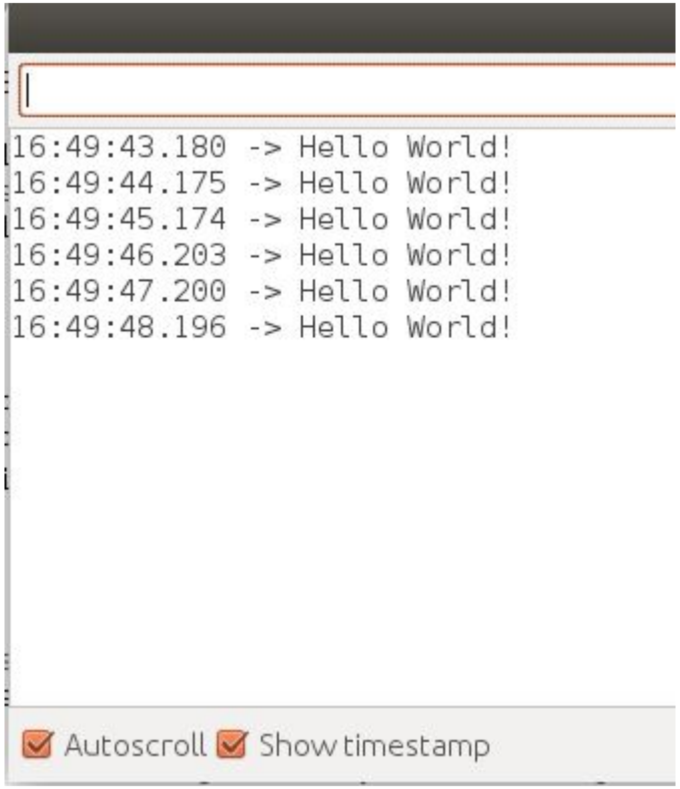
Digital PIN: 13 (output)



Serial Communication



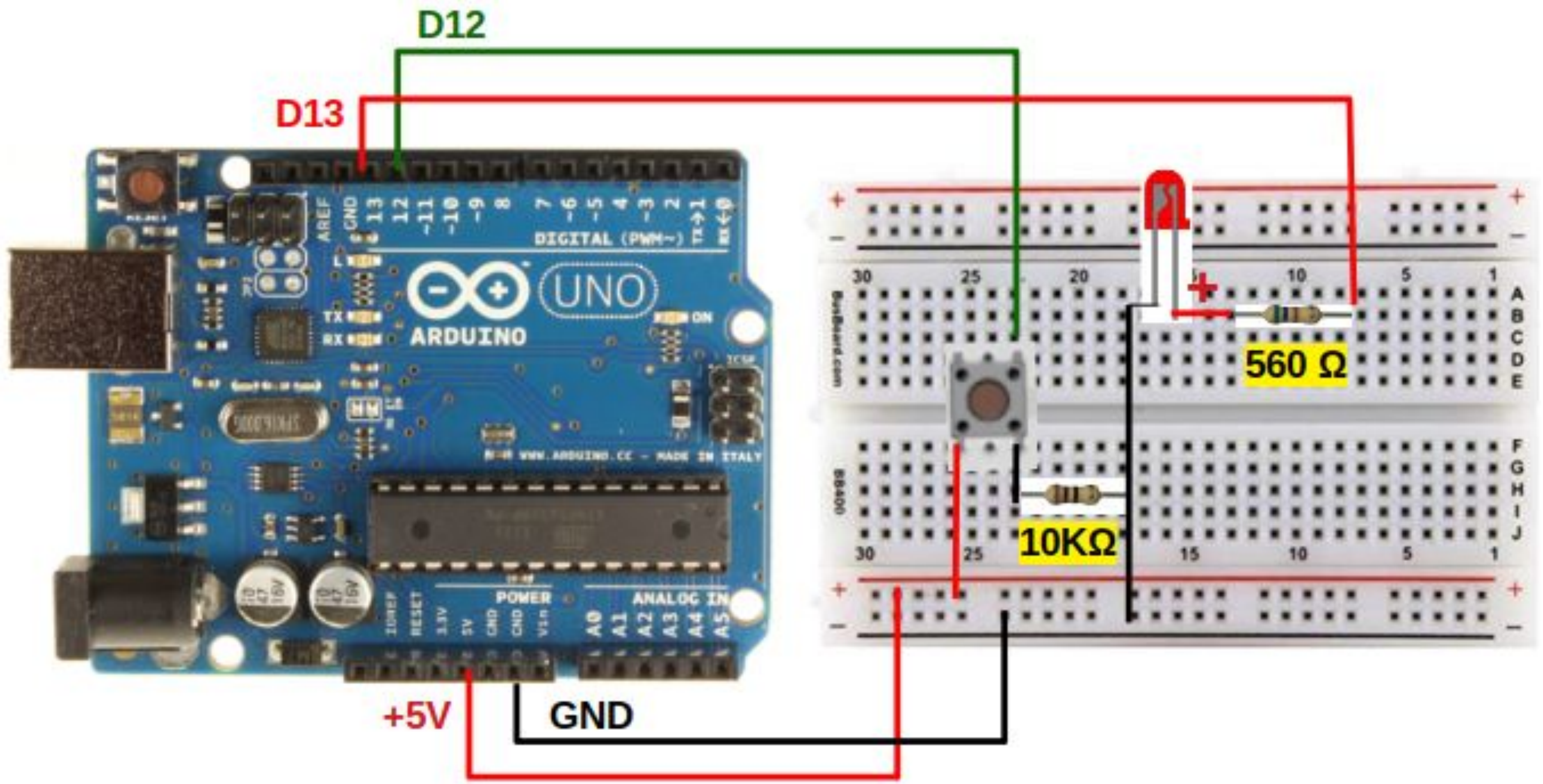
```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  Serial.println("Hello World!");
  delay(1000);

}
```



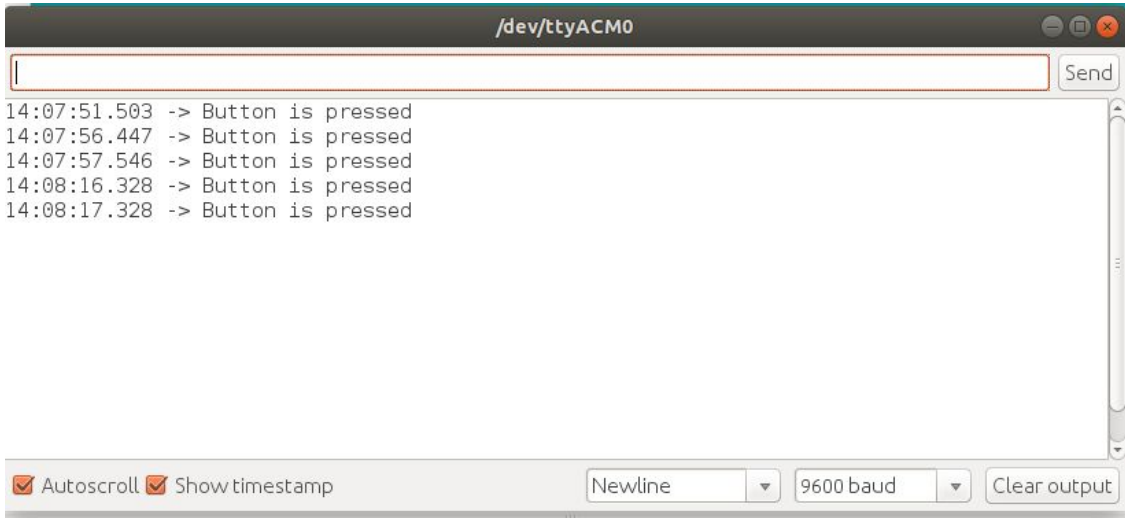
```
16:49:43.180 -> Hello World!
16:49:44.175 -> Hello World!
16:49:45.174 -> Hello World!
16:49:46.203 -> Hello World!
16:49:47.200 -> Hello World!
16:49:48.196 -> Hello World!
```

Click on Serial Monitor to see the output.

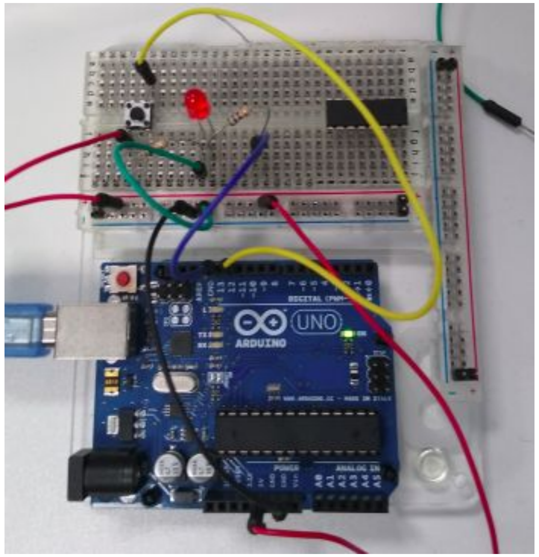
LED Blinking with Serial I/O

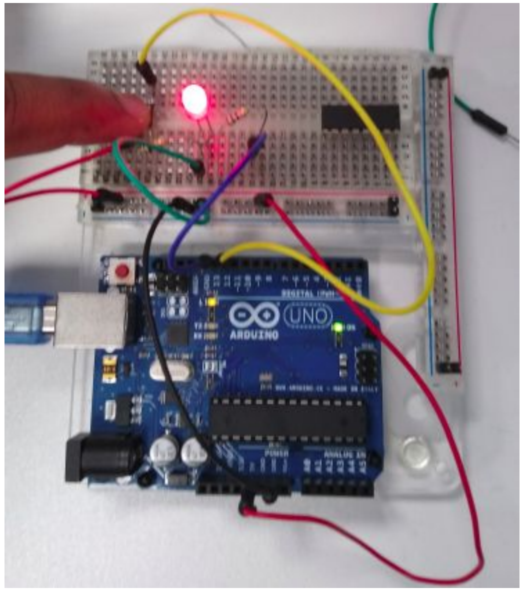


```
int led = 13;
int button = 12;
void setup() {
    pinMode(led, OUTPUT);
    pinMode(button, INPUT);
    Serial.begin(9600);
}
void loop() {
int buttonState = digitalRead(button);
  if ( buttonState == HIGH){
    digitalWrite(led, HIGH);
    Serial.println("Button is pressed");
    delay(500);
  }
  else{
    digitalWrite(led, LOW);
    delay(100);
  }
}
```

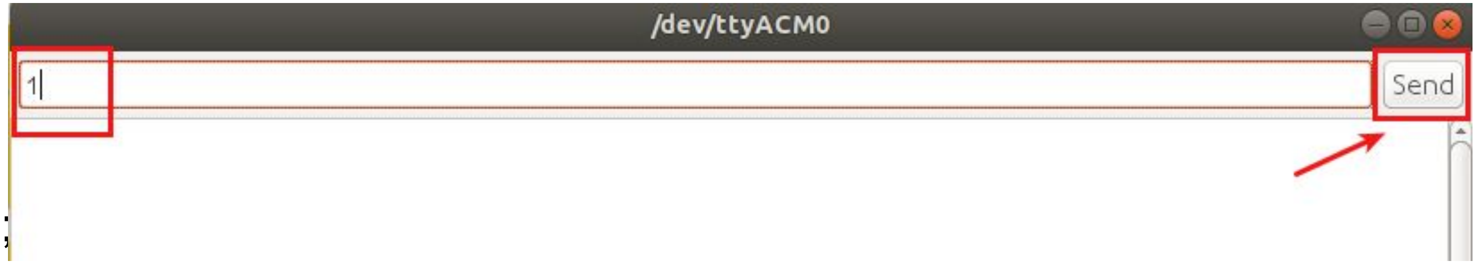


```
14:07:51.503 -> Button is pressed
14:07:56.447 -> Button is pressed
14:07:57.546 -> Button is pressed
14:08:16.328 -> Button is pressed
14:08:17.328 -> Button is pressed
```





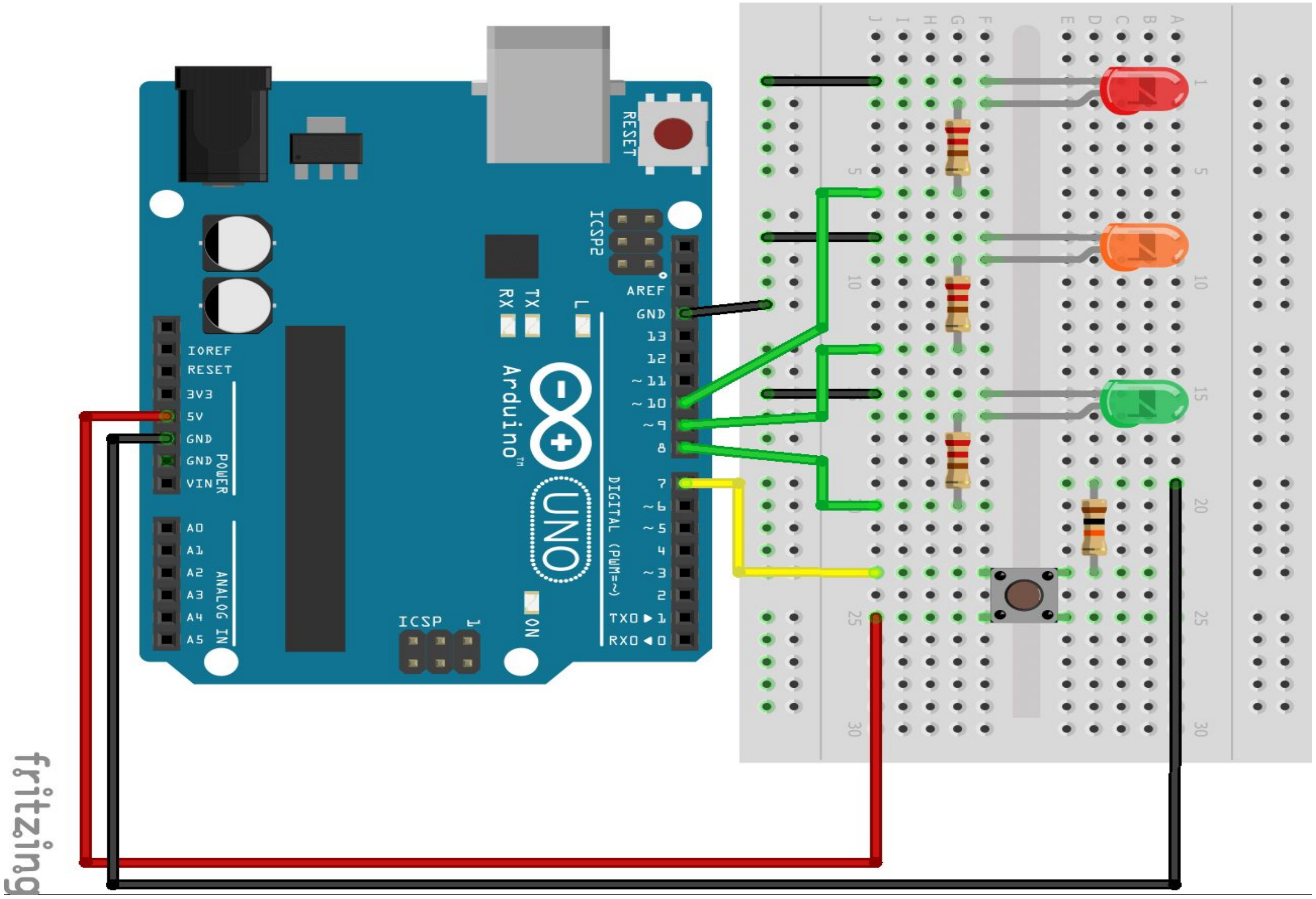
```
int led = 13;
int button = 12;
void setup() {
pinMode(led, OUTPUT);
pinMode(button, INPUT);
Serial.begin(9600);
}
void loop(){
  if(Serial.available() > 0) {
    char ledState = Serial.read();     // Reads from  serial port
    if(ledState == '1'){
      digitalWrite(led, HIGH);
    }
    if(ledState == '0'){
      digitalWrite(led, LOW);
    }
  }
```



LED glows when 1 is sent through the serial port. It turns off when 0 is sent.

Controlling three LEDs with a Push Button

- Objective is to control 3 LEDs through a single push button.
- The LEDs should glow in a sequence on each press of the push button.
- It should rotate through these three LEDs as the button is repeatedly pressed.
- You can display the button pressed status and count through the serial monitor.

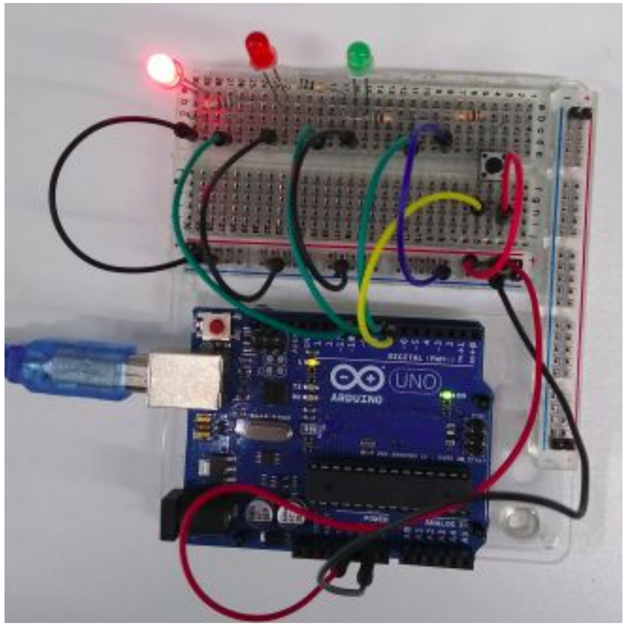


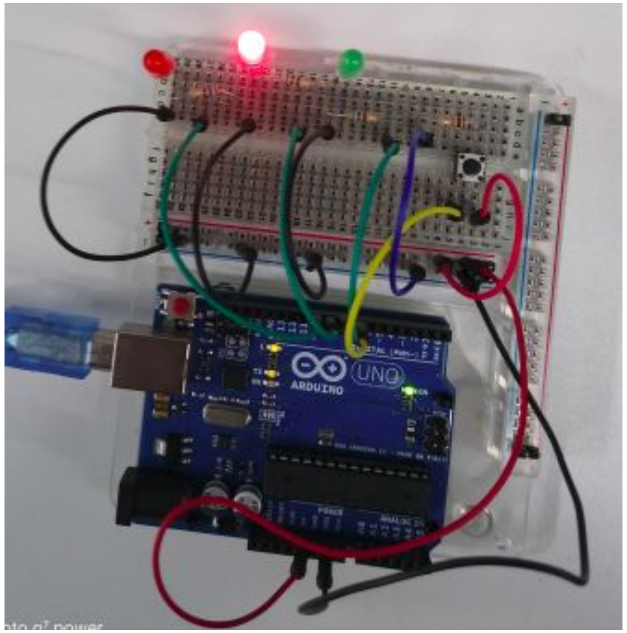
RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX0 ▶ 1
RX0 ◀ 0
DIGITAL (PWM=~)
TX
RX
L
Arduino
UNO
ON
ICSP
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
fritzing

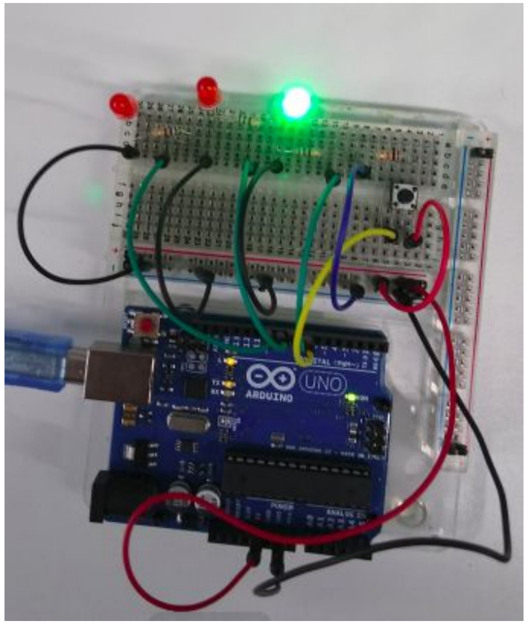
```
#define btnPin 7
#define led1Pin 8
#define led2Pin 9
#define led3Pin 10
int buttonState = 0;
int ledcount = 0;


void setup() {

  pinMode(btnPin, INPUT);
  pinMode(led1Pin, OUTPUT);
  pinMode(led2Pin, OUTPUT);
  pinMode(led3Pin, OUTPUT);

  Serial.begin(9600);
}
```

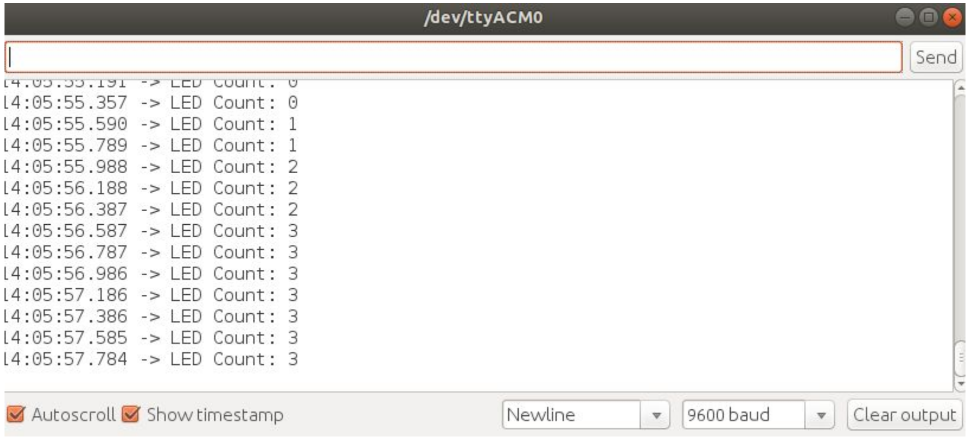
```
void loop() {
  // put your main code here, to run
repeatedly:

  if(buttonPressed())
    set_mode();

  if(ledcount > 3)
    ledcount = 0;

  Serial.print("LED Count: ");
  Serial.println(ledcount);
  delay(200);
}

bool buttonPressed(){
  buttonState = digitalRead(btnPin);
  if(buttonState == HIGH){
    ledcount += 1;
    return true;
  }else{
    return false;
  }
}
```

```
void set_mode()
{
  switch(ledcount){
    case 0:
      digitalWrite(led1Pin, LOW);
      digitalWrite(led2Pin, LOW);
      digitalWrite(led3Pin, LOW);
      break;
    case 1:
      digitalWrite(led1Pin, HIGH);
      digitalWrite(led2Pin, LOW);
      digitalWrite(led3Pin, LOW);
      break;
    case 2:
      digitalWrite(led1Pin, LOW);
      digitalWrite(led2Pin, HIGH);
      digitalWrite(led3Pin, LOW);
      break;
    case 3:
      digitalWrite(led1Pin, LOW);
      digitalWrite(led2Pin, LOW);
      digitalWrite(led3Pin, HIGH);
      break;
  }
}
```





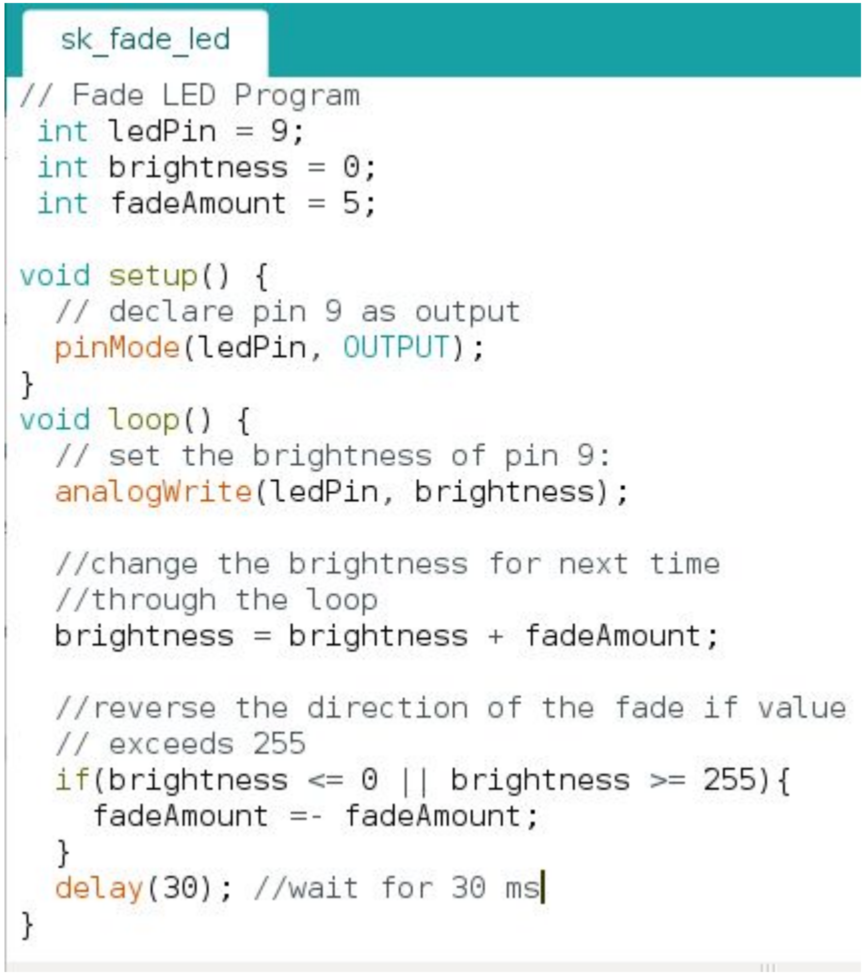




```
/dev/ttyACM0
                                                            Send
14:05:55.191 -> LED Count: 0
14:05:55.357 -> LED Count: 0
14:05:55.590 -> LED Count: 1
14:05:55.789 -> LED Count: 1
14:05:55.988 -> LED Count: 2
14:05:56.188 -> LED Count: 2
14:05:56.387 -> LED Count: 2
14:05:56.587 -> LED Count: 3
14:05:56.787 -> LED Count: 3
14:05:56.986 -> LED Count: 3
14:05:57.186 -> LED Count: 3
14:05:57.386 -> LED Count: 3
14:05:57.585 -> LED Count: 3
14:05:57.784 -> LED Count: 3
[✓] Autoscroll [✓] Show timestamp      Newline   9600 baud   Clear output
```

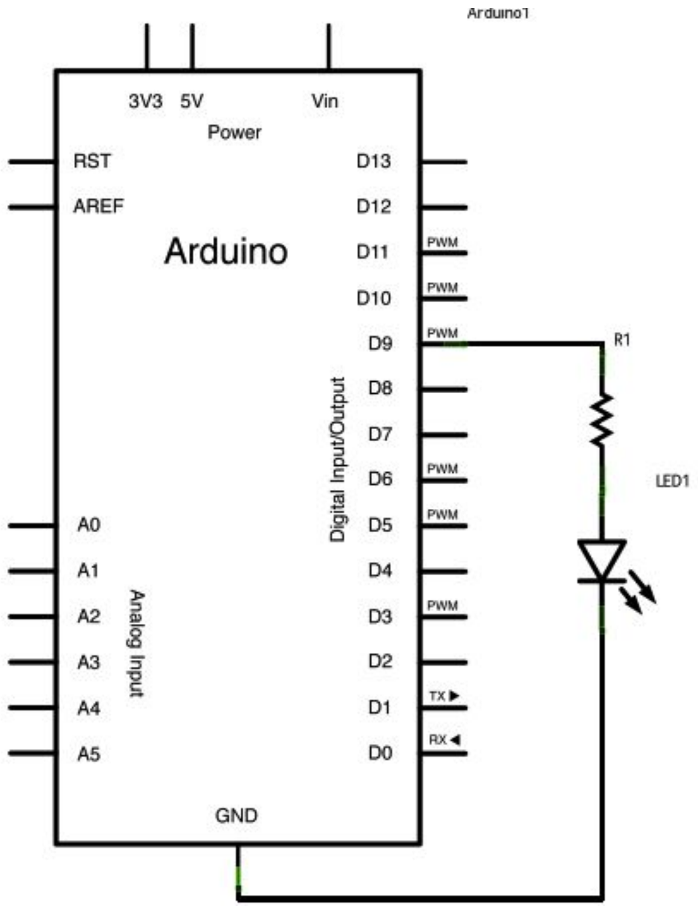
Serial Monitor Output

Fading LED

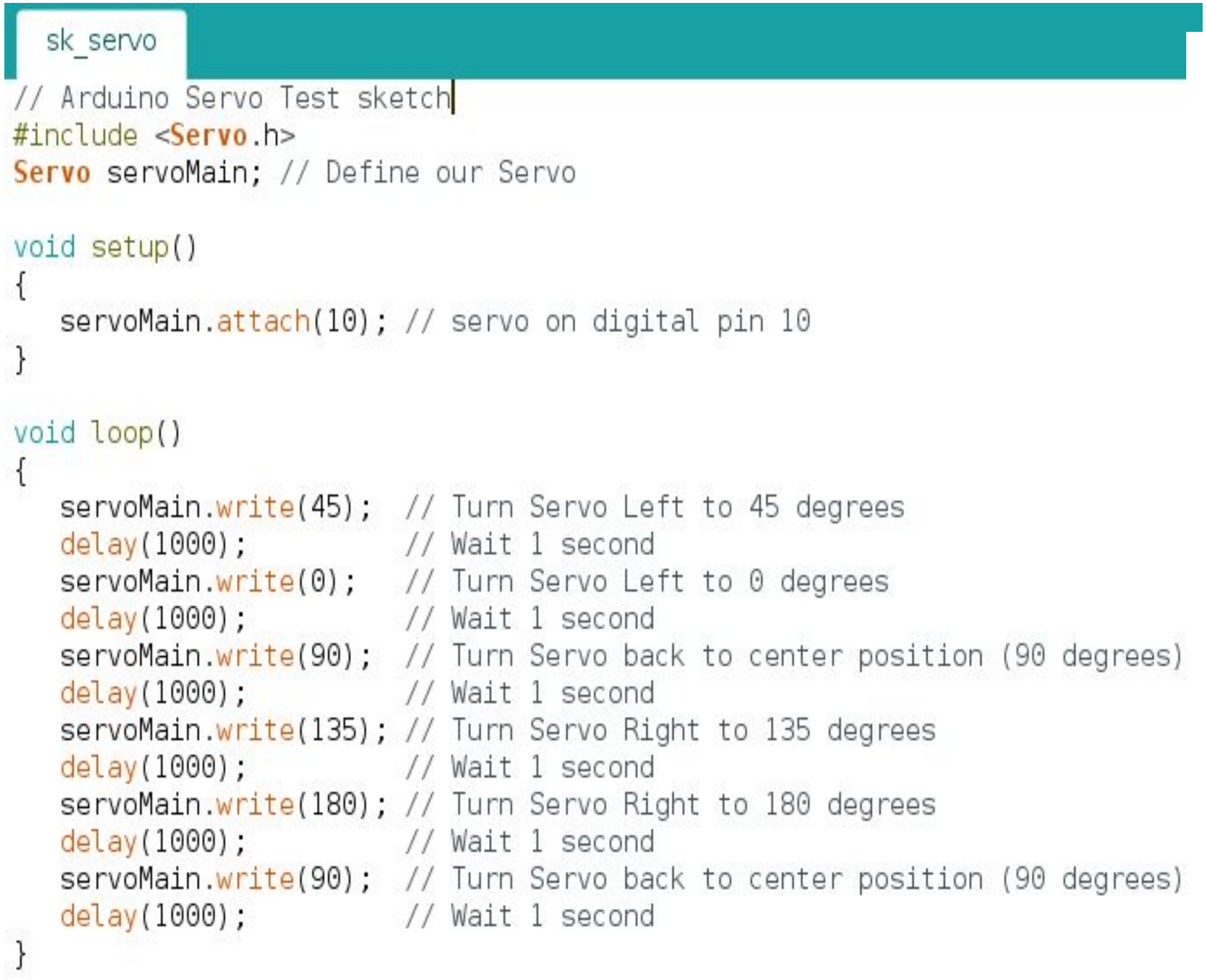


sk_fade_led

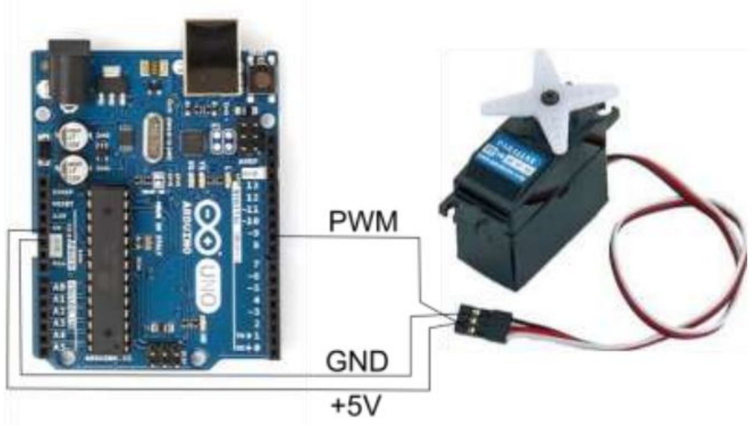
```
// Fade LED Program
 int ledPin = 9;
 int brightness = 0;
 int fadeAmount = 5;

void setup() {
  // declare pin 9 as output
  pinMode(ledPin, OUTPUT);
}
void loop() {
  // set the brightness of pin 9:
  analogWrite(ledPin, brightness);

  //change the brightness for next time
  //through the loop
  brightness = brightness + fadeAmount;

  //reverse the direction of the fade if value
  // exceeds 255
  if(brightness <= 0 || brightness >= 255){
    fadeAmount =- fadeAmount;
  }
  delay(30); //wait for 30 ms
}
```



Driving a Stepper Motor



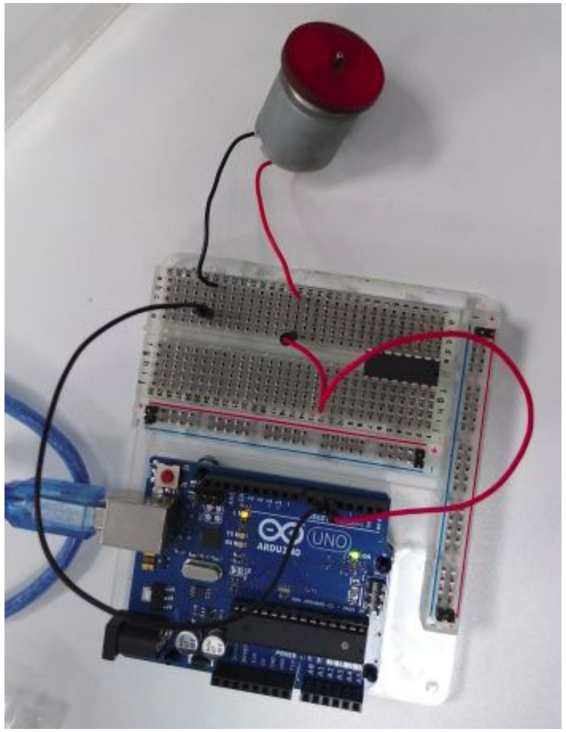
```
// Arduino Servo Test sketch
#include <Servo.h>
Servo servoMain; // Define our Servo

void setup()
{
   servoMain.attach(10); // servo on digital pin 10
}

void loop()
{
   servoMain.write(45);  // Turn Servo Left to 45 degrees
   delay(1000);          // Wait 1 second
   servoMain.write(0);   // Turn Servo Left to 0 degrees
   delay(1000);          // Wait 1 second
   servoMain.write(90);  // Turn Servo back to center position (90 degrees)
   delay(1000);          // Wait 1 second
   servoMain.write(135); // Turn Servo Right to 135 degrees
   delay(1000);          // Wait 1 second
   servoMain.write(180); // Turn Servo Right to 180 degrees
   delay(1000);          // Wait 1 second
   servoMain.write(90);  // Turn Servo back to center position (90 degrees)
   delay(1000);          // Wait 1 second
}
```



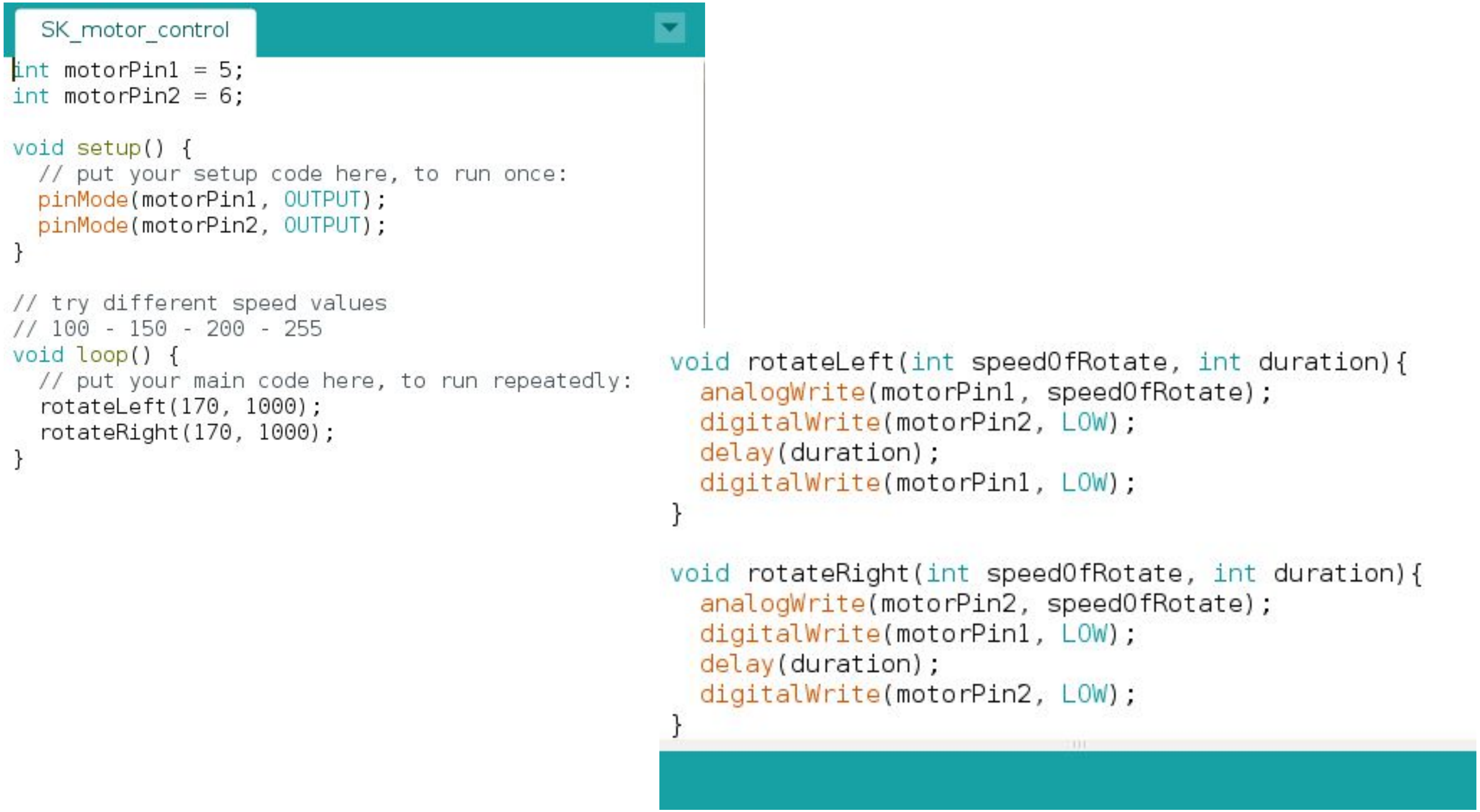
- Black Wire to GND
- Red wire to 5V
- White wire to PWM PIN 10

Bi-directional Speed Control using PWM

- Experiment 1:
  - D6 to one terminal of the dc motor
  - D5 to another terminal of dc motor
- Experiment 2:
  - Using transistor as a switch

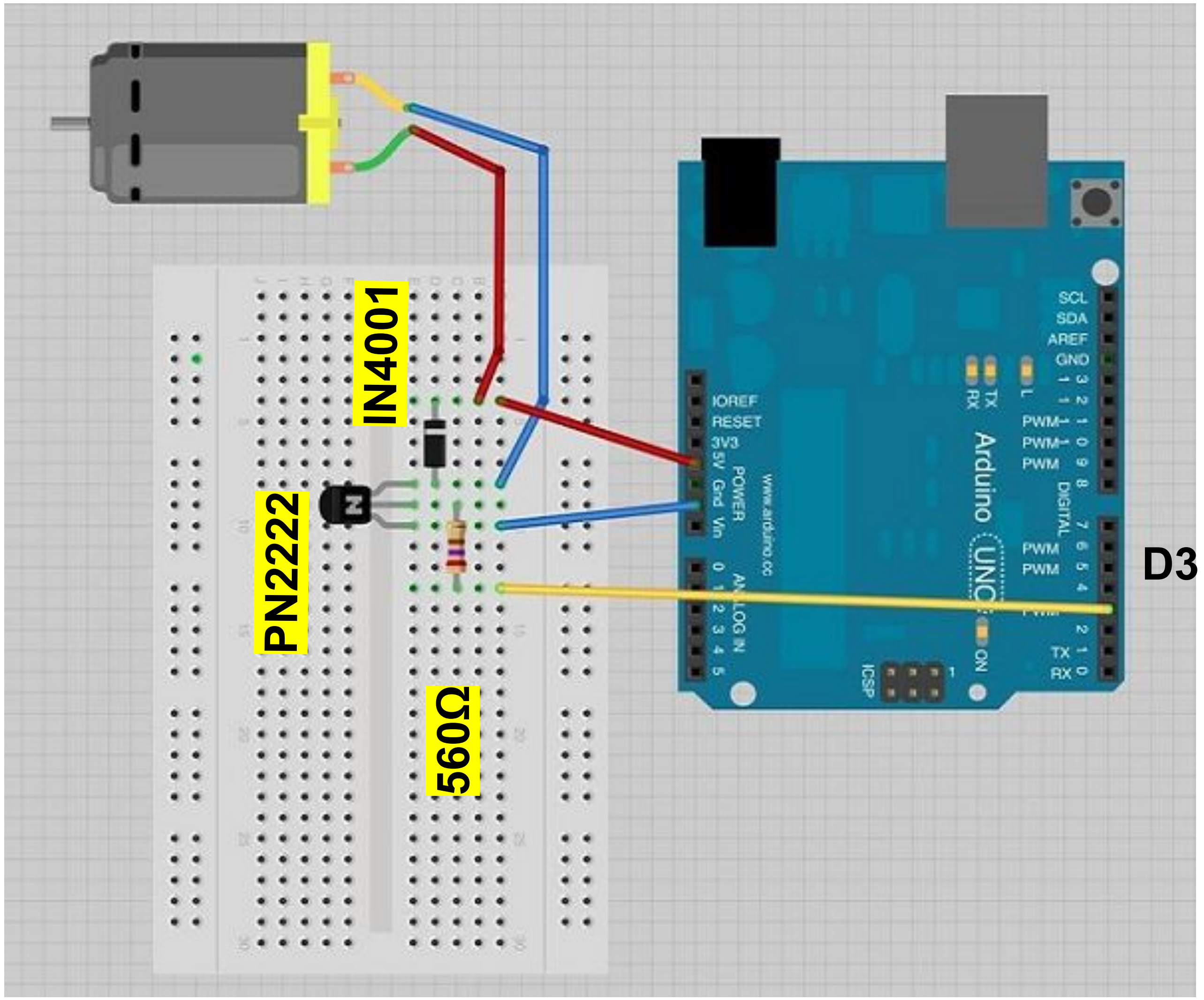


Setup for Experiment 1


SK_motor_control

```
int motorPin1 = 5;
int motorPin2 = 6;

void setup() {
  // put your setup code here, to run once:
  pinMode(motorPin1, OUTPUT);
  pinMode(motorPin2, OUTPUT);
}

// try different speed values
// 100 - 150 - 200 - 255
void loop() {
  // put your main code here, to run repeatedly:
  rotateLeft(170, 1000);
  rotateRight(170, 1000);
}
```

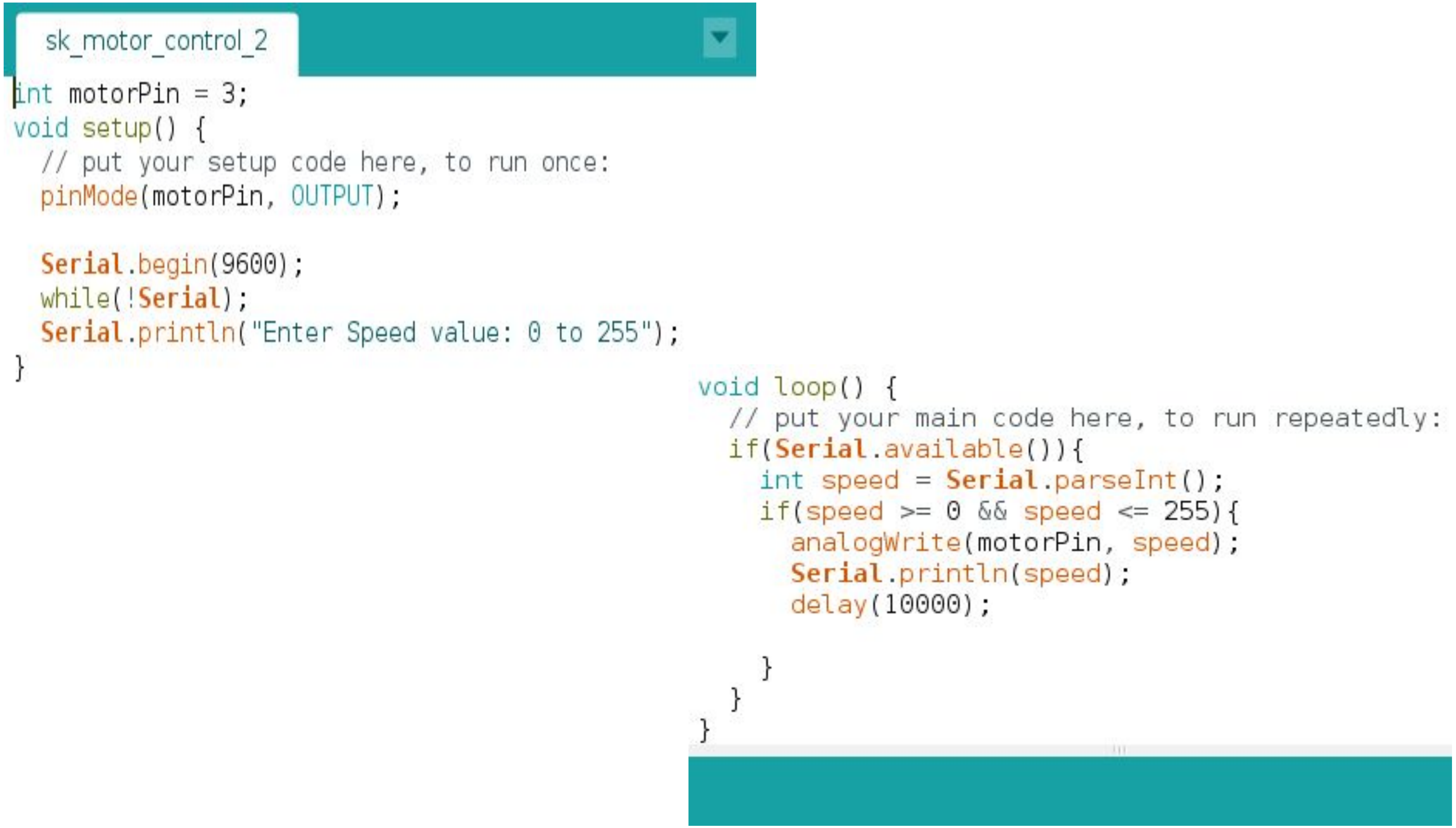
```
void rotateLeft(int speedOfRotate, int duration){
  analogWrite(motorPin1, speedOfRotate);
  digitalWrite(motorPin2, LOW);
  delay(duration);
  digitalWrite(motorPin1, LOW);
}

void rotateRight(int speedOfRotate, int duration){
  analogWrite(motorPin2, speedOfRotate);
  digitalWrite(motorPin1, LOW);
  delay(duration);
  digitalWrite(motorPin2, LOW);
}
```



Setup for Experiment 2

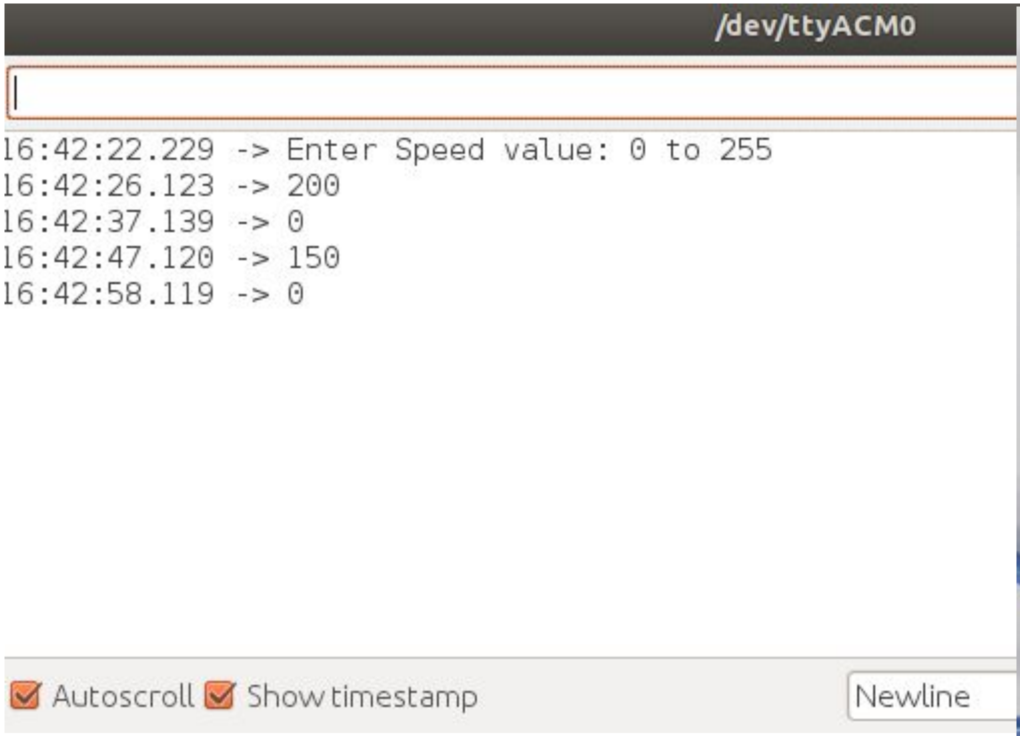
Precautions

- First, make sure that the transistor is connected in the right way. The flat side of the transistor should face the Arduino board as shown in the arrangement.
- Second, the striped end of the diode should be towards the +5V power line according to the arrangement shown in the image.

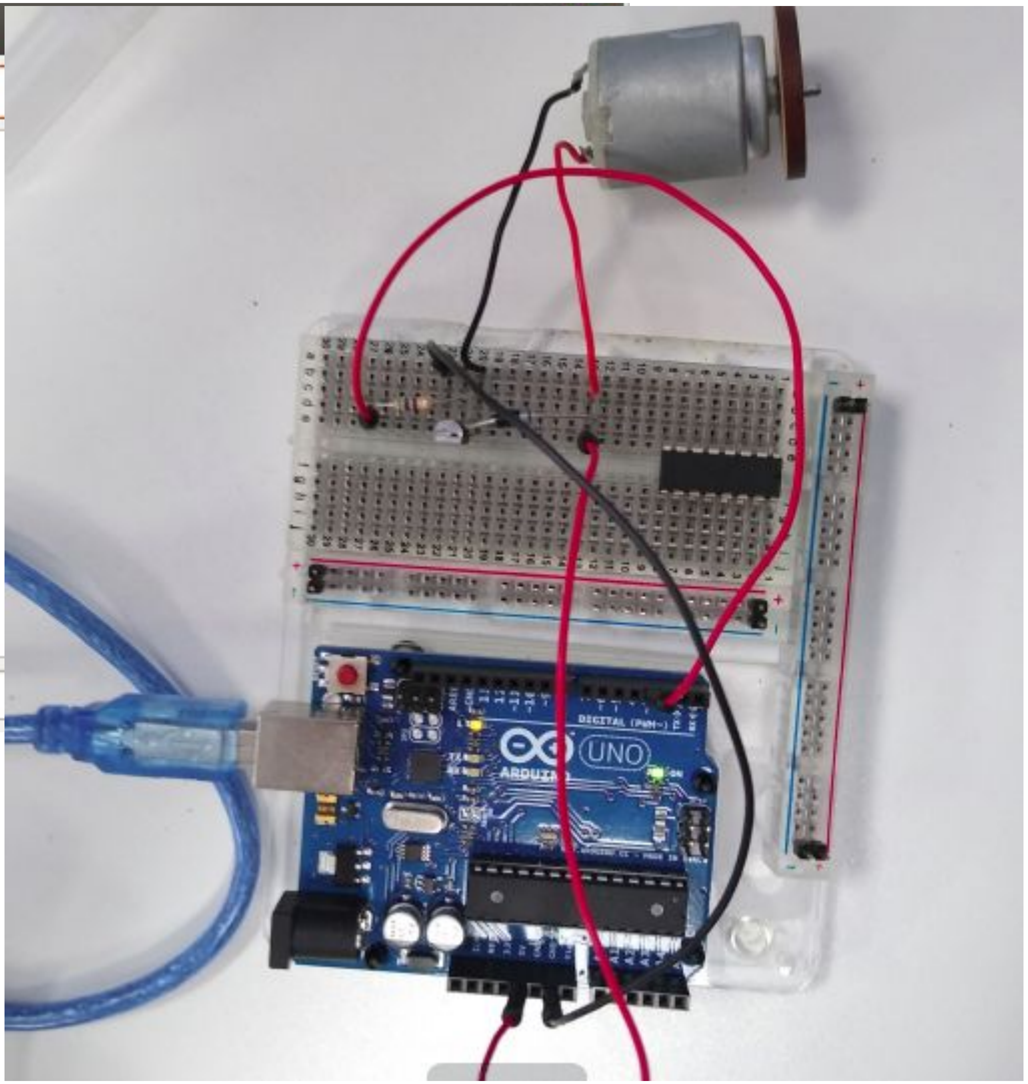

sk_motor_control_2

```
int motorPin = 3;
void setup() {
  // put your setup code here, to run once:
  pinMode(motorPin, OUTPUT);

  Serial.begin(9600);
  while(!Serial);
  Serial.println("Enter Speed value: 0 to 255");
}
```

```
void loop() {
  // put your main code here, to run repeatedly:
  if(Serial.available()){
    int speed = Serial.parseInt();
    if(speed >= 0 && speed <= 255){
      analogWrite(motorPin, speed);
      Serial.println(speed);
      delay(10000);

    }
  }
}
```


/dev/ttyACM0

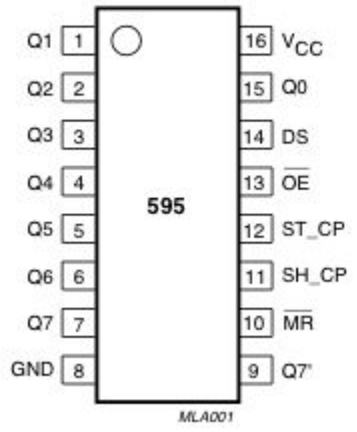
```
16:42:22.229 -> Enter Speed value: 0 to 255
16:42:26.123 -> 200
16:42:37.139 -> 0
16:42:47.120 -> 150
16:42:58.119 -> 0
```



Multiplexer:74HC595

- Required for extending the number of output pins of your microcontroller.

- 74HC595 is a shift register which works on Serial IN Parallel OUT protocol.
- It receives data serially from the microcontroller and then sends out this data through parallel pins.
- We can increase our output pins by 8 using the single chip.
- We can also connect more than 1 shift register in parallel.
- So, let's say I have connected three shift registers with our microcontroller then our output pins are increased by 8 x 3 = 24.



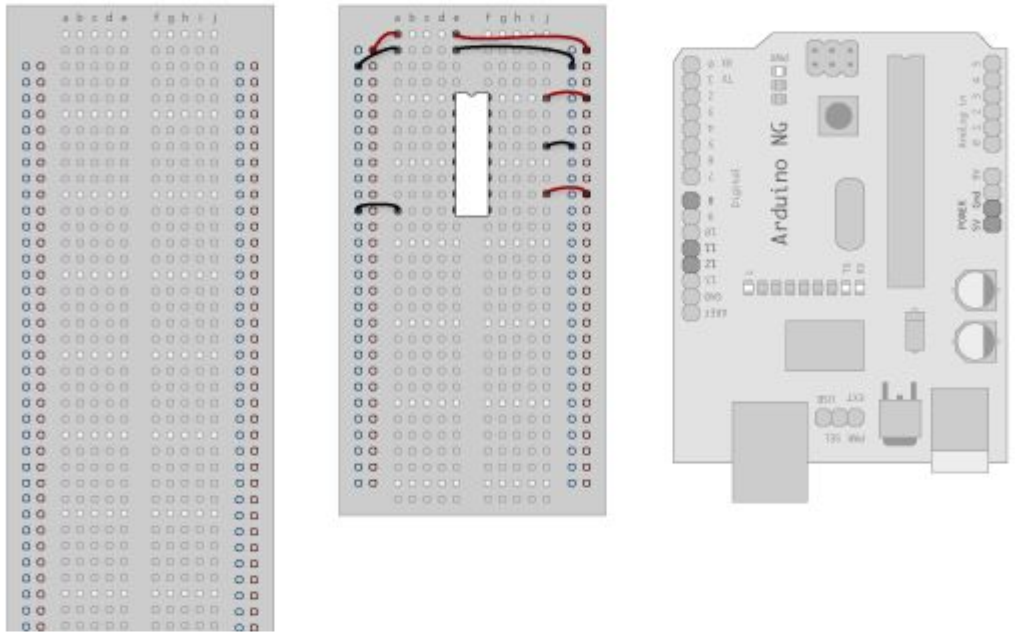
PINS 1-7, 15	Q0 " Q7	Output Pins
PIN 8	GND	Ground, Vss
PIN 9	Q7"	Serial Out
PIN 10	MR	Master Reclear, active low
PIN 11	SH_CP	Shift register clock pin
PIN 12	ST_CP	Storage register clock pin (latch pin)
PIN 13	OE	Output enable, active low
PIN 14	DS	Serial data input
PIN 16	Vcc	Positive supply voltage

- Pin # 1 to Pin # 7 are Output Pins Q1 - Q7.
- Pin # 15 is also Output Pin Q0.
- Pin # 8 is Ground.
- Pin # 9 is Q7' (OutPut Serial Data).
- Pin # 10 is Master Reset.
- Pin # 11 is SHCP which is short for Shift Register Clock Input.
- Pin # 12 is STCP which is short for Storage Register Clock Input.
- Pin # 13 is OE which is Output Enable.
- Pin # 14 is DS which is Serial Data input.
- Pin # 16 is Vcc where we have to supply the power +5V.

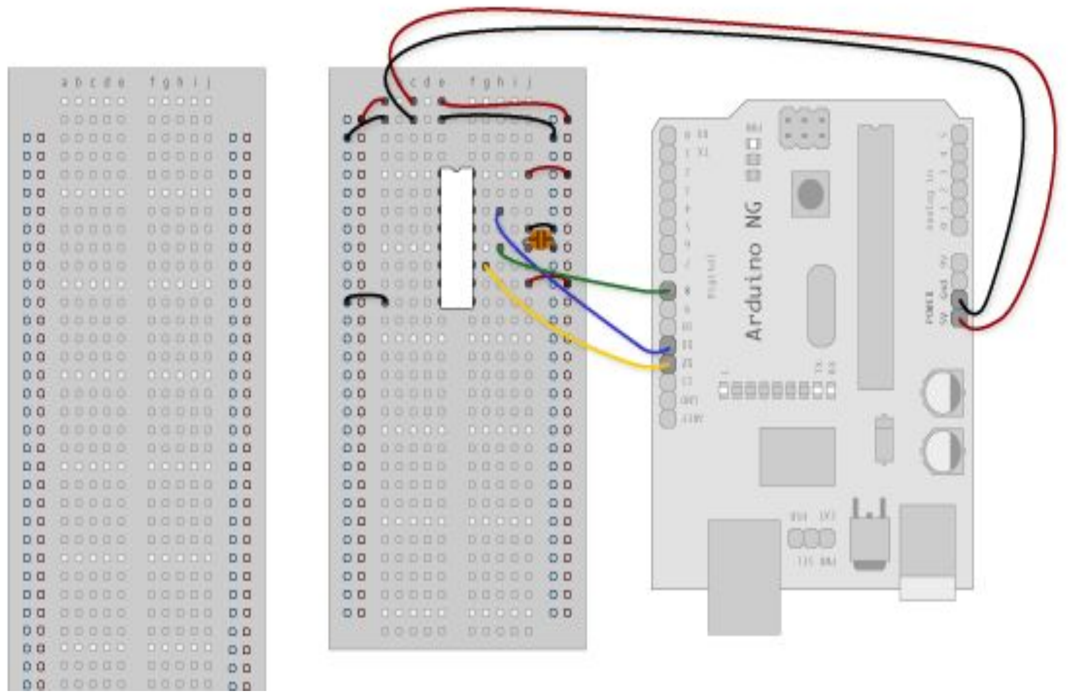
Pin Number	Pin Name	Description
1-7	Parallel Output Pins Q1 to Q7	The 74hc595 has 8 output pins out of which 7 are these pins. They can be controlled serially.
8	GND	Connected to the Ground of the circuit
9	Q7’ (Serial Output)	This pin is used to connect more than one 74hc595 as cascading
10	(MR) Master Reset	Resets all outputs as low. Must be held high for normal operation
11	SHCP (Clock)	This is the clock pin to which the clock signal has to be provided from MCU/MPU
12	STCP (Latch)	The Latch pin is used to update the data to the output pins. It is active high
13	OE (output Enable)	The Output Enable is used to turn off the outputs. Must be held low for normal operation
14	DS (Serial Data)	This is the pin to which data is sent, based on which the 8 outputs are controlled
15	Q0	The first output pin

Configure the Chip

- GND (pin 8) to ground,
- Vcc (pin 16) to 5V
- OE (pin 13) to ground
- MR (pin 10) to 5V

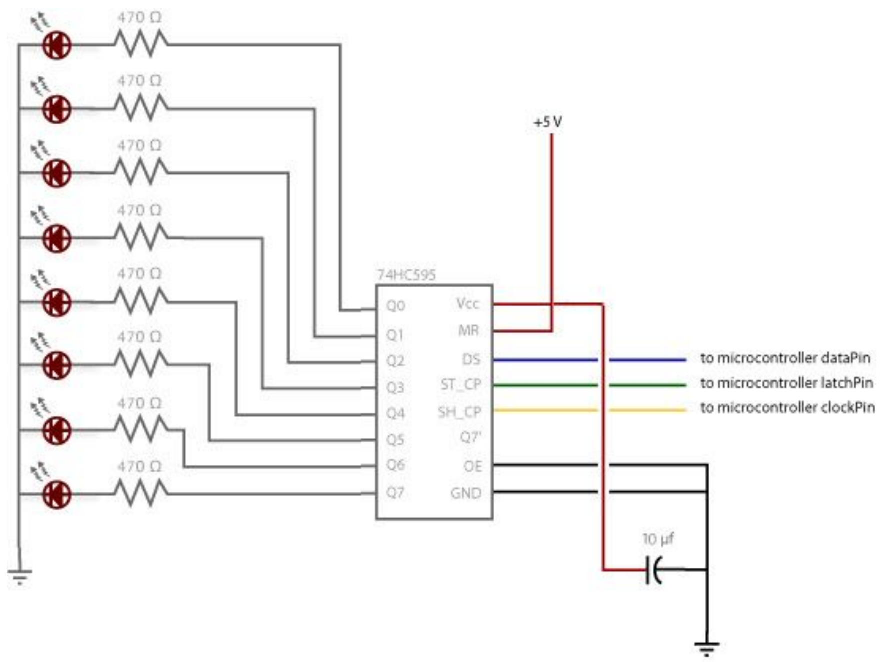


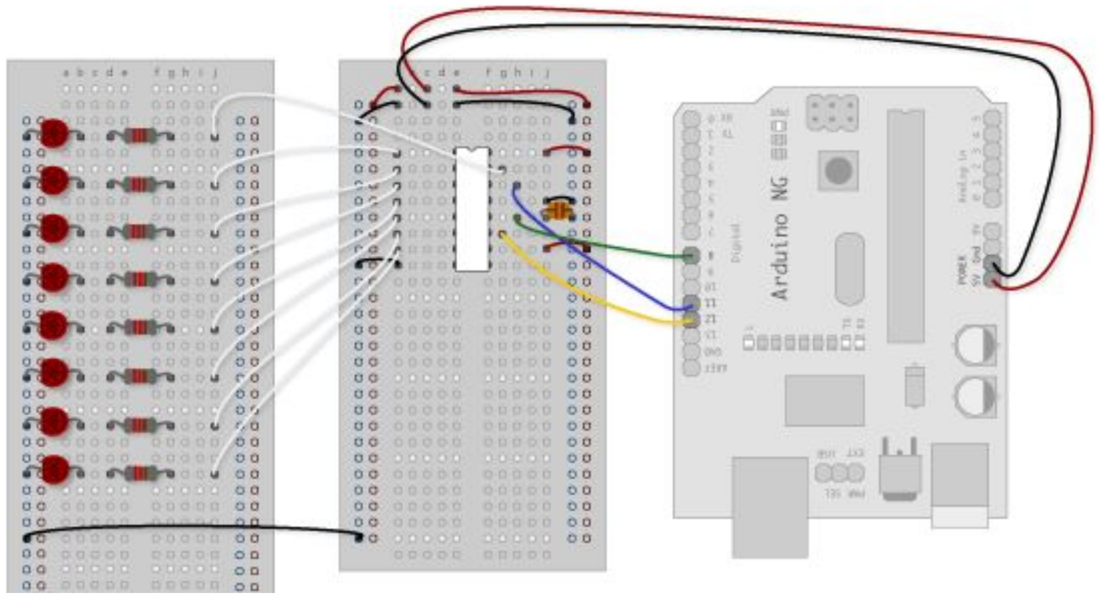
Connect to Arduino



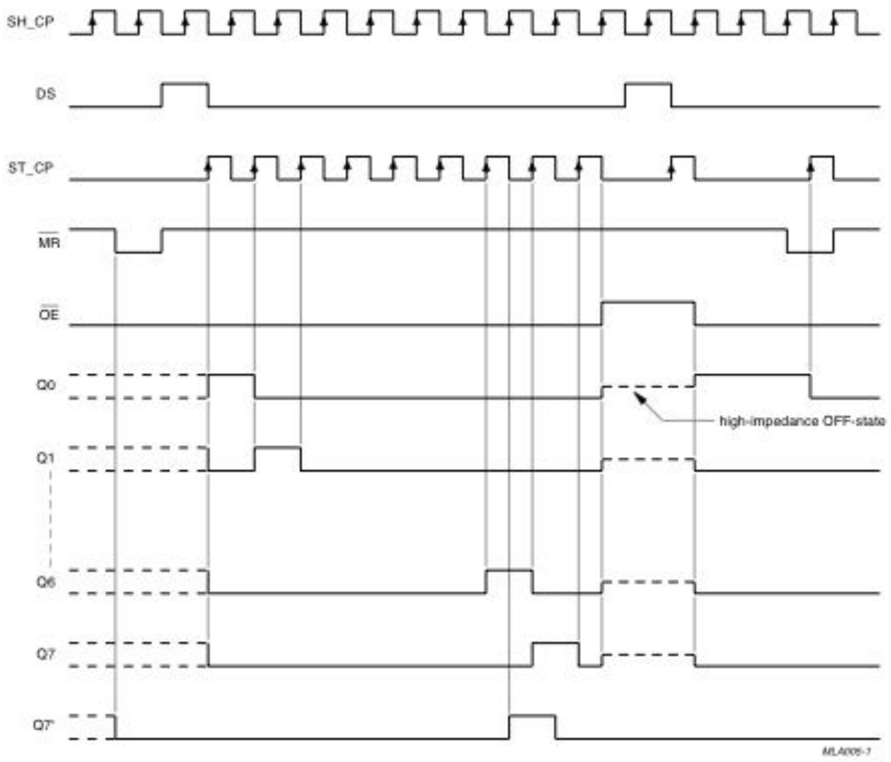
- DS (pin 14) to Ardunio DigitalPin 11 (blue wire) – Data PIN
- SH_CP (pin 11) to to Arduino DigitalPin 12 (yellow wire) – Clock PIN
- ST_CP (pin 12) to Ardunio DigitalPin 8 (green wire) – Latch PIN

Add 8 LEDs


470 Ω
+5 V
74HC595
Q0
Q1
Q2
Q3
Q4
Q5
Q6
Q7
Vcc
MR
DS
ST_CP
SH_CP
Q7'
OE
GND
to microcontroller dataPin
to microcontroller latchPin
to microcontroller clockPin
10 µf



- connect the cathode (short pin) of each LED to a common ground.
- The anode (long pin) of each LED to its respective shift register output pin



FUNCTION TABLE

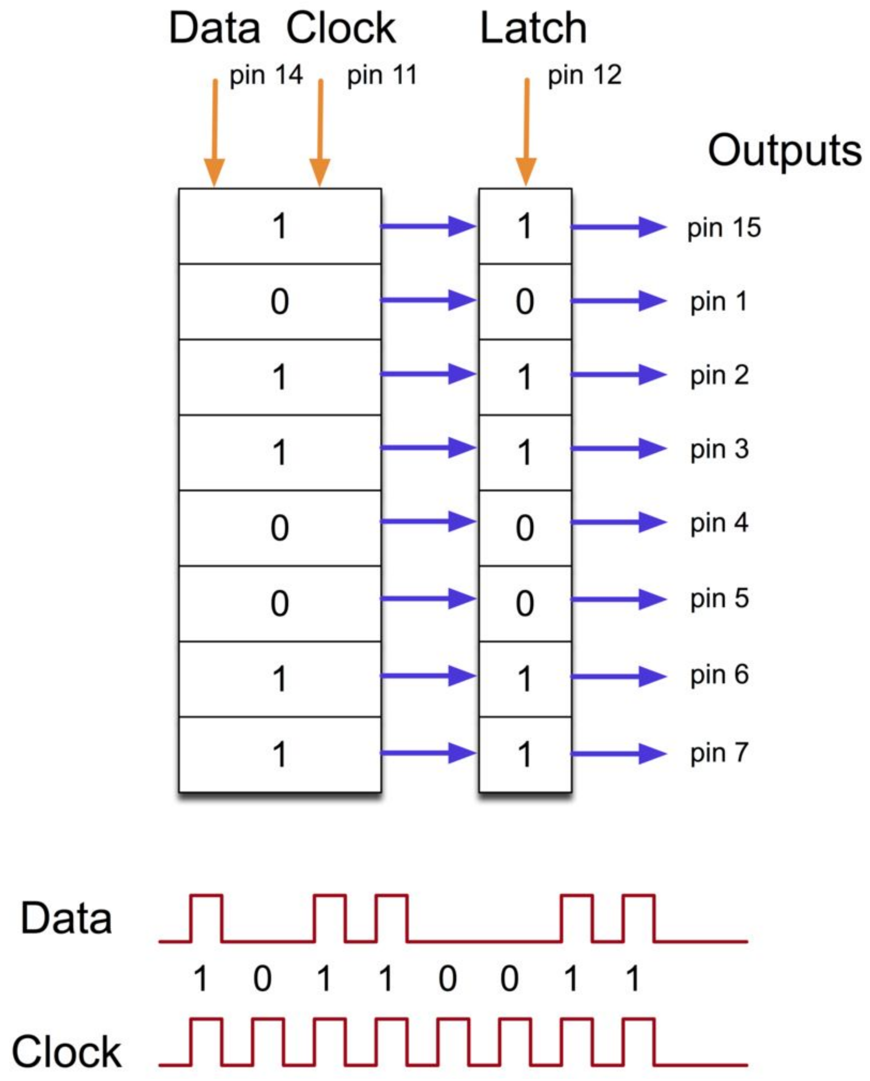
See note 1.

INPUT					OUTPUT		FUNCTION
SH_CP	ST_CP	$\overline{OE}$	$\overline{MR}$	DS	Q7'	Qn	
X	X	L	L	X	L	n.c.	a LOW level on $\overline{MR}$ only affects the shift registers
X	↑	L	L	X	L	L	empty shift register loaded into storage register
X	X	H	L	X	L	Z	shift register clear; parallel outputs in high-impedance OFF-state
↑	X	L	H	H	Q6'	n.c.	logic high level shifted into shift register stage 0; contents of all shift register stages shifted through, e.g. previous state of stage 6 (internal Q6') appears on the serial output (Q7')
X	↑	L	H	X	n.c.	Qn'	contents of shift register stages (internal Qn') are transferred to the storage register and parallel output stages
↑	↑	L	H	X	Q6'	Qn'	contents of shift register shifted through; previous contents of the shift register is transferred to the storage register and the parallel output stages

Note

1. H = HIGH voltage level;
   L = LOW voltage level;
   ↑ = LOW-to-HIGH transition;
   ↓ = HIGH-to-LOW transition;
   Z = high-impedance OFF-state;
   n.c. = no change;
   X = don't care.

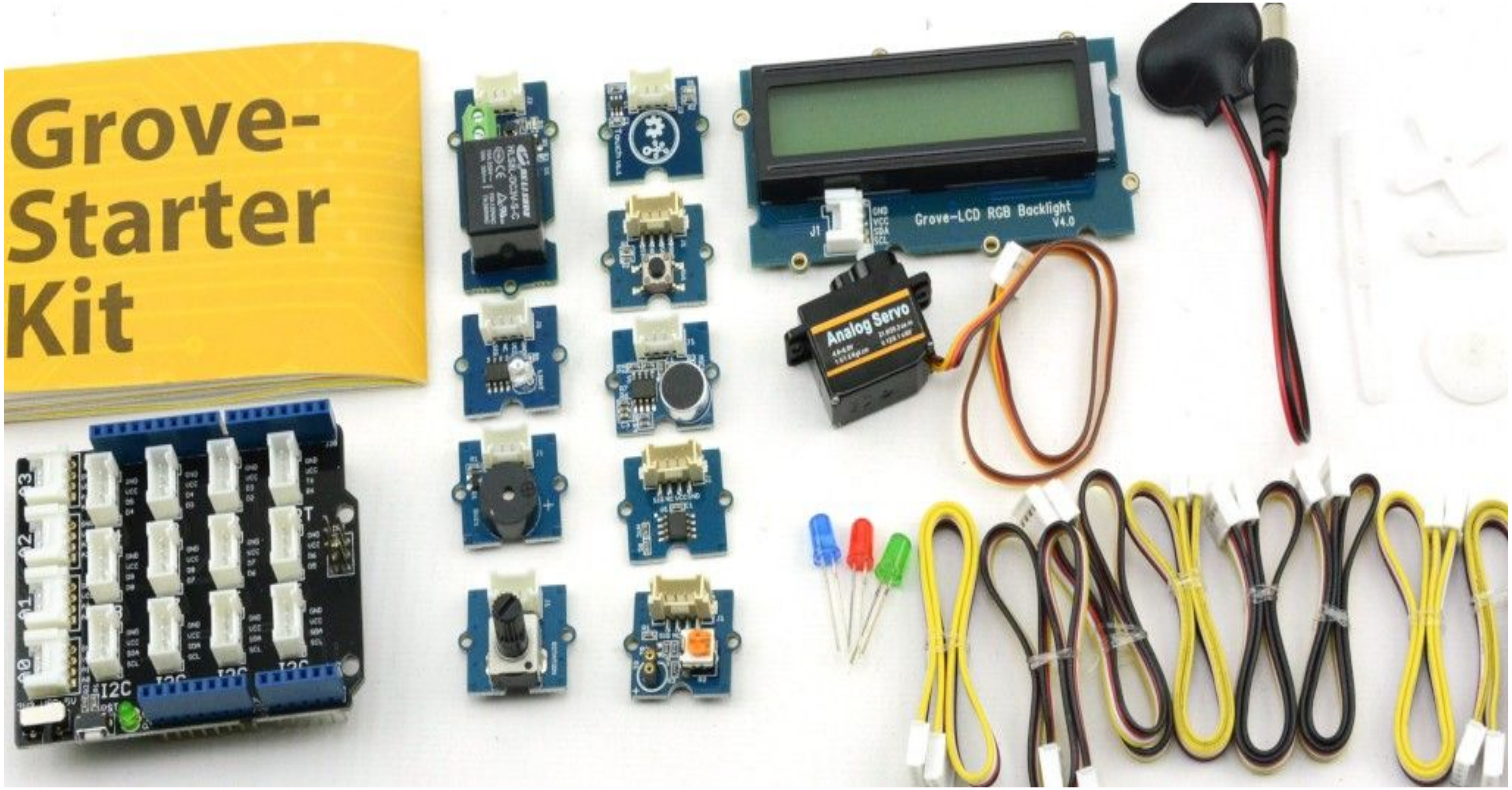
- The shift register holds what can be thought of as eight memory locations, each of which can be a 1 or a 0.
- To set each of these values on or off, we feed in the data using the 'Data' and 'Clock' pins of the chip.
- The clock pin needs to receive eight pulses. At the time of each pulse, if the data pin is high, then a 1 gets pushed into the shift register. Otherwise, it is a 0.
- When all eight pulses have been received, then enabling the 'Latch' pin copies those eight values to the latch register.
- The chip also has an OE (output enable) pin, this is used to enable or disable the outputs all at once.



```
/*
Blinking 8 LEDs one-by-one using a shift
register
*/

int latchPin = 8;
int clockPin = 12;
int dataPin = 11;

byte leds = 0; // 8-bits

void setup()
{
  pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
}

void loop()
{
  leds = 0;
  updateShiftRegister();
  delay(500);
  for (int i = 0; i < 8; i++)
  {
    bitSet(leds, i);
    updateShiftRegister();
    delay(500);
  }
}

void updateShiftRegister()
{
   digitalWrite(latchPin, LOW);
   shiftOut(dataPin, clockPin, LSBFIRST, leds);
   digitalWrite(latchPin, HIGH);
}
```

```
/* Hello World */
//Pin connected to ST_CP of 74HC595
int latchPin = 8;
//Pin connected to SH_CP of 74HC595
int clockPin = 12;
////Pin connected to DS of 74HC595
int dataPin = 11;


void setup() {
  //set pins to output so you can control
the shift register
  pinMode(latchPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
}
```

```
void loop() {
  // count from 0 to 255 and display the number
  // on the LEDs
  for (int numberToDisplay = 0; numberToDisplay <
256; numberToDisplay++) {
    // take the latchPin low so
    // the LEDs don't change while you're sending in
bits:
    digitalWrite(latchPin, LOW);
    // shift out the bits:
    shiftOut(dataPin, clockPin, MSBFIRST,
numberToDisplay);

    //take the latch pin high so the LEDs will light up:
    digitalWrite(latchPin, HIGH);
    // pause before next value:
    delay(200);
  }
}
```


Grove-Starter Kit
Grove-LCD RGB Backlight V4.0
Analog Servo
I2C

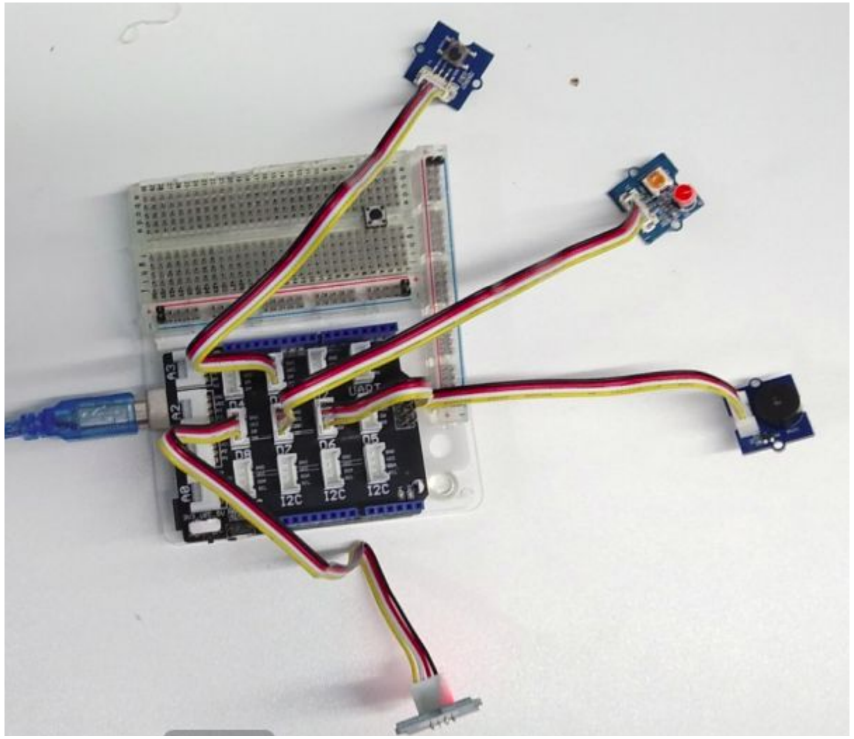
Grove Starter KIT

What is Grove starter KIT?

Grove is a modular electronics platform for convenient and rapid prototyping.

The advantages of Grove Starter Kit

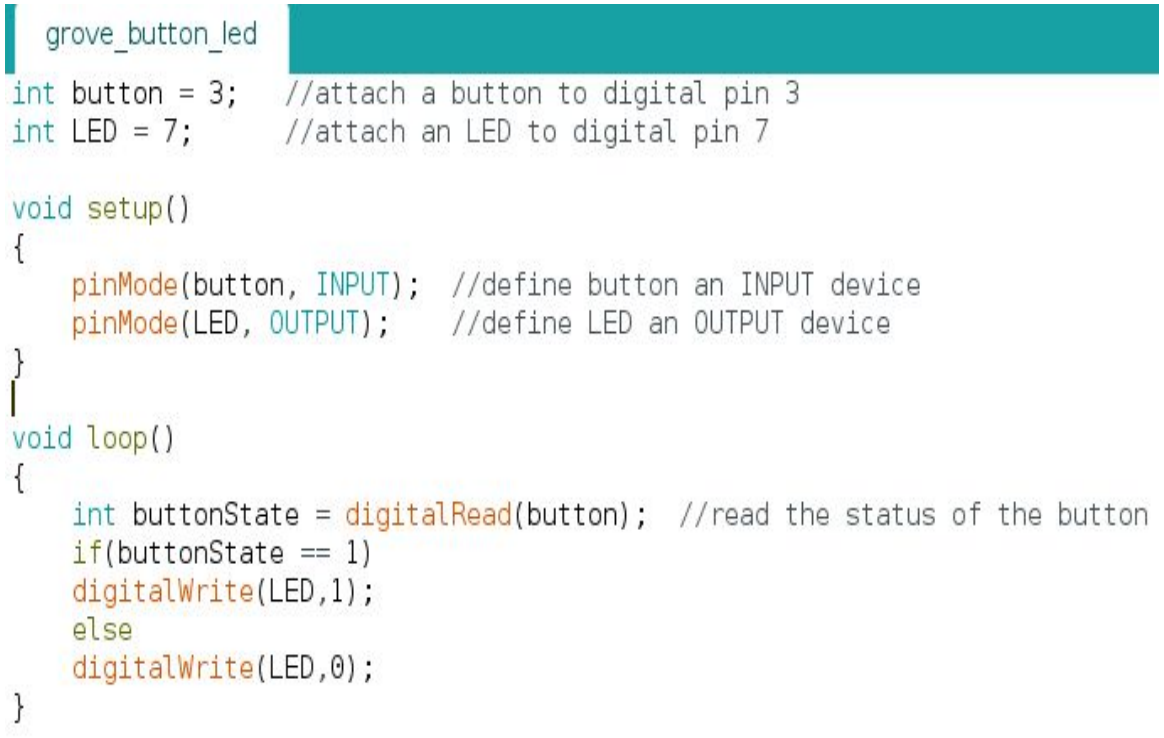
- Unified interface for different kinds of sensors and actuators.
- PIN Layout and Basic Arduino API can be used for most ports.
- Higher-level library for advanced components such as LCD panel is available.
- Shorter development time for building hardware application.
  - No soldering or breadboard required.
- Source codes are available at https://github.com/Seeed-Studio/Sketchbook_Starter_Kit_V2.0



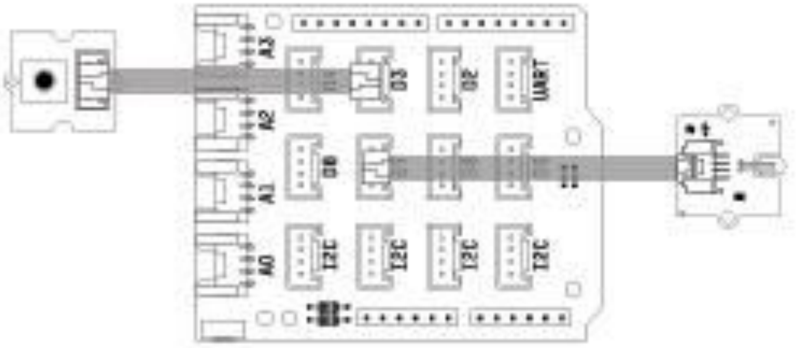
Tutorial Wiki: http://wiki.seeedstudio.com/Grove_Starter_Kit_v3/

Various Experiments with Grove Starter Kit

- Controlling LED with a Push button
- Controlling LED blinking with a Touch Sensor
- Buzzer
- Rotary Potentiometer (Angle Sensor)
- Relay Switch
- Sound Sensor - Controlling LED indicating noise level in the room
- Light Sensor
- Temperature Sensor - Measure the room temperature
- Controlling Servo Motor using Angle Sensor
- Controlling a RGB LCD Screen

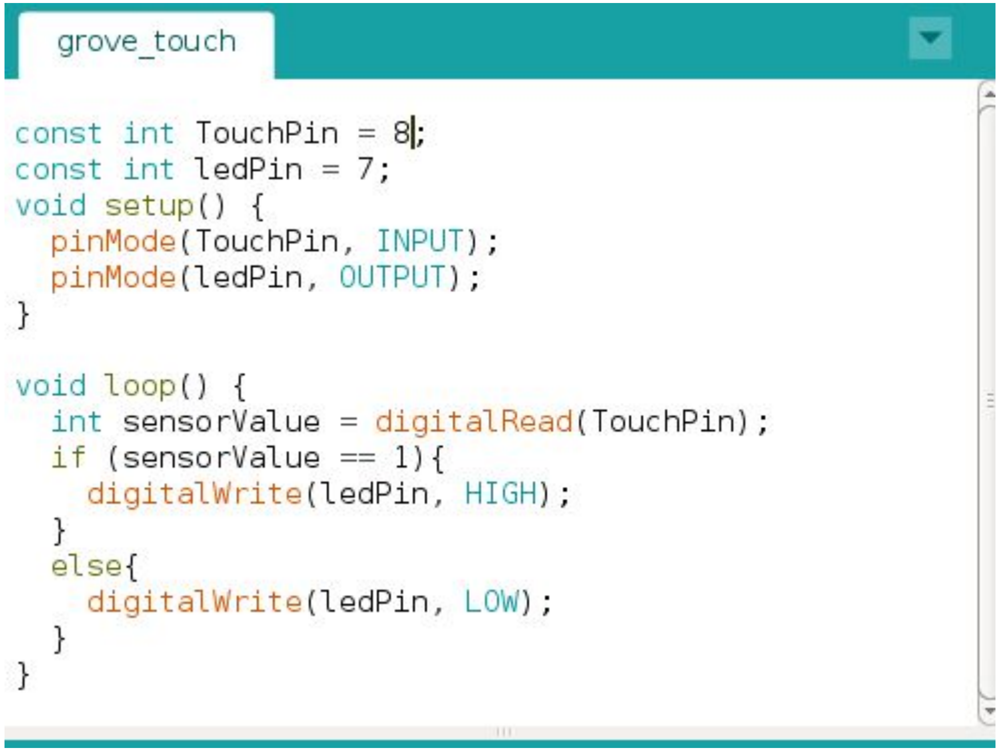


```
grove_button_led
int button = 3;    //attach a button to digital pin 3
int LED = 7;       //attach an LED to digital pin 7

void setup()
{
    pinMode(button, INPUT);  //define button an INPUT device
    pinMode(LED, OUTPUT);    //define LED an OUTPUT device
}

void loop()
{
    int buttonState = digitalRead(button);  //read the status of the button
    if(buttonState == 1)
    digitalWrite(LED,1);
    else
    digitalWrite(LED,0);
}
```



- Connect D3 to Button
- Connect D7 to LED

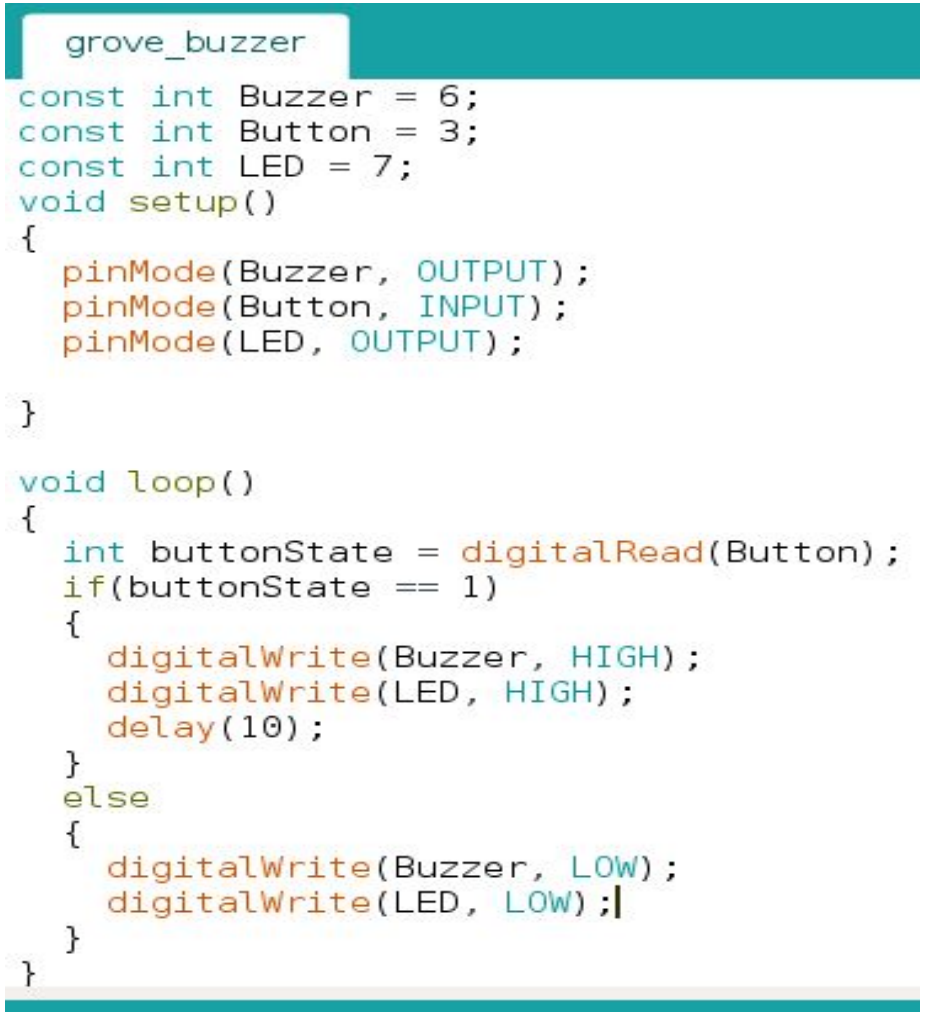
Touch Button



```
const int TouchPin = 8;
const int ledPin = 7;
void setup() {
  pinMode(TouchPin, INPUT);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  int sensorValue = digitalRead(TouchPin);
  if (sensorValue == 1){
    digitalWrite(ledPin, HIGH);
  }
  else{
    digitalWrite(ledPin, LOW);
  }
}
```

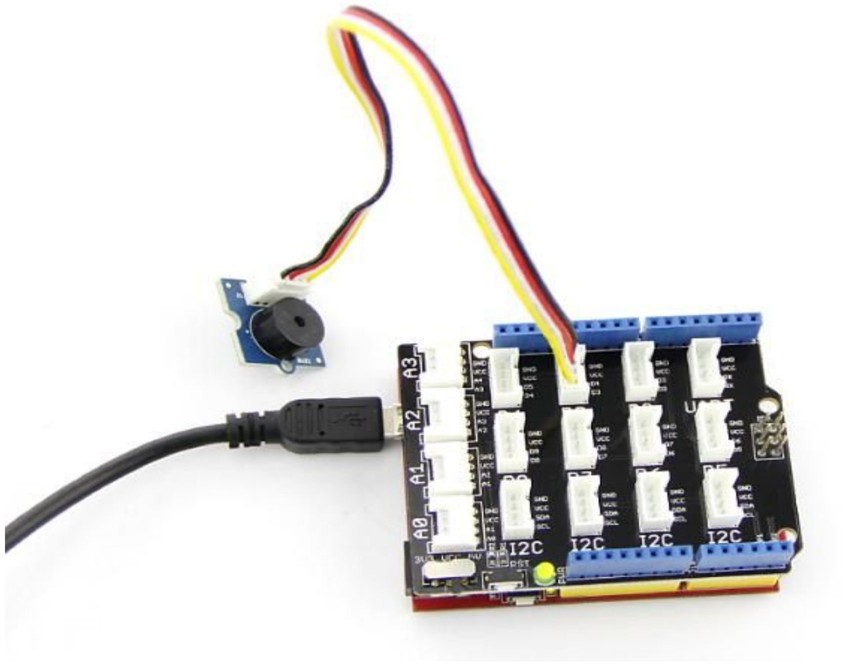
- Connect D8 to Touch button
- Connect LED to D7

Buzzer



grove_buzzer

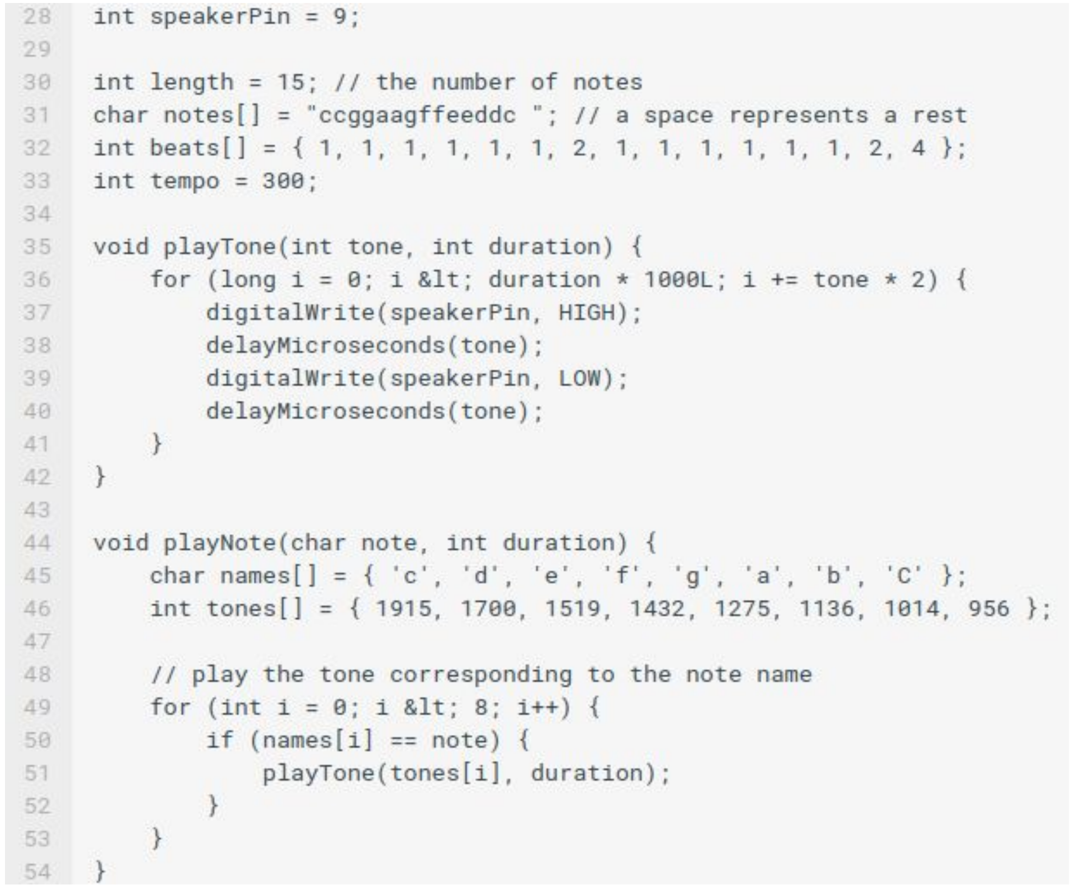
```
const int Buzzer = 6;
const int Button = 3;
const int LED = 7;
void setup()
{
  pinMode(Buzzer, OUTPUT);
  pinMode(Button, INPUT);
  pinMode(LED, OUTPUT);

}

void loop()
{
  int buttonState = digitalRead(Button);
  if(buttonState == 1)
  {
    digitalWrite(Buzzer, HIGH);
    digitalWrite(LED, HIGH);
    delay(10);
  }
  else
  {
    digitalWrite(Buzzer, LOW);
    digitalWrite(LED, LOW);
  }
}
```



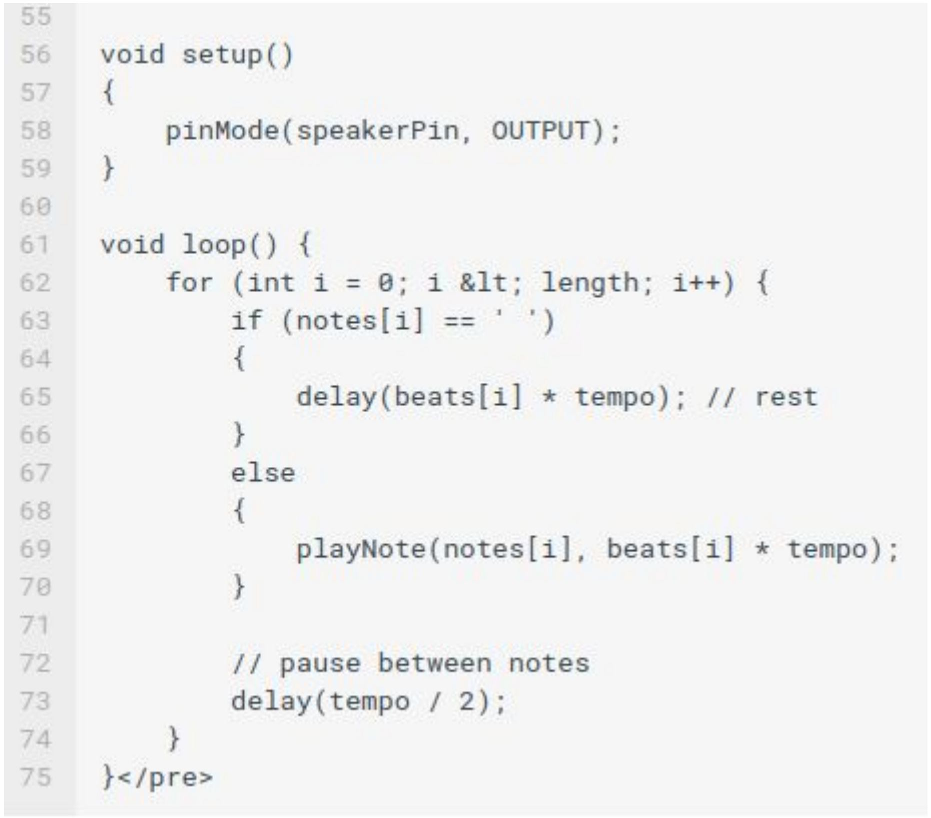
- Connect Buzzer to D6
- Connect Push Button to D3
- Connect LED to D7

Connect Buzzer to Port D6.

Buzzer Melody “Twinkle Twinkle Little Star”

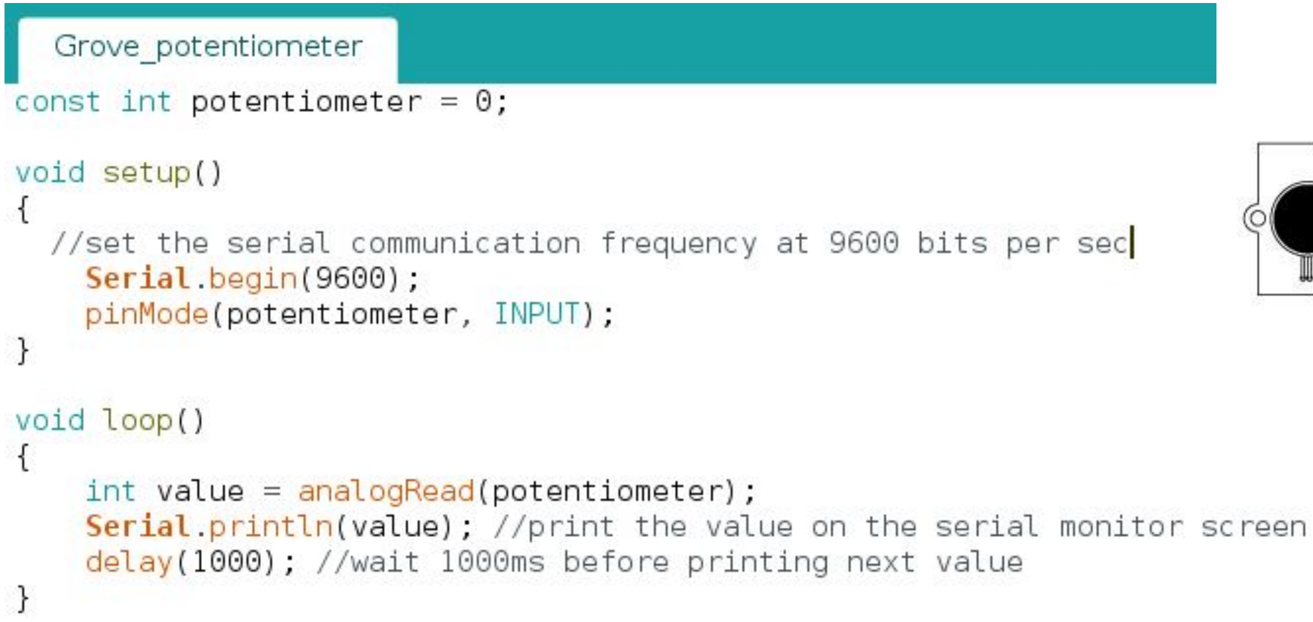


```
int speakerPin = 9;

int length = 15; // the number of notes
char notes[] = "ccggaagffeeddc "; // a space represents a rest
int beats[] = { 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 4 };
int tempo = 300;

void playTone(int tone, int duration) {
    for (long i = 0; i &lt; duration * 1000L; i += tone * 2) {
        digitalWrite(speakerPin, HIGH);
        delayMicroseconds(tone);
        digitalWrite(speakerPin, LOW);
        delayMicroseconds(tone);
    }
}

void playNote(char note, int duration) {
    char names[] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C' };
    int tones[] = { 1915, 1700, 1519, 1432, 1275, 1136, 1014, 956 };

    // play the tone corresponding to the note name
    for (int i = 0; i &lt; 8; i++) {
        if (names[i] == note) {
            playTone(tones[i], duration);
        }
    }
}
```

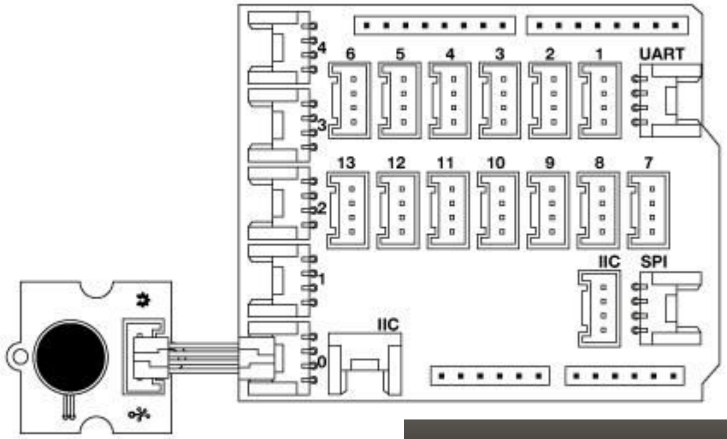


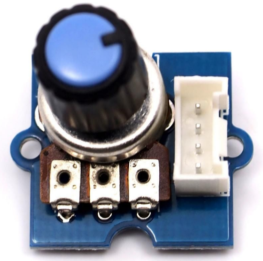
```

void setup()
{
    pinMode(speakerPin, OUTPUT);
}

void loop() {
    for (int i = 0; i &lt; length; i++) {
        if (notes[i] == ' ')
        {
            delay(beats[i] * tempo); // rest
        }
        else
        {
            playNote(notes[i], beats[i] * tempo);
        }

        // pause between notes
        delay(tempo / 2);
    }
}</pre>
```

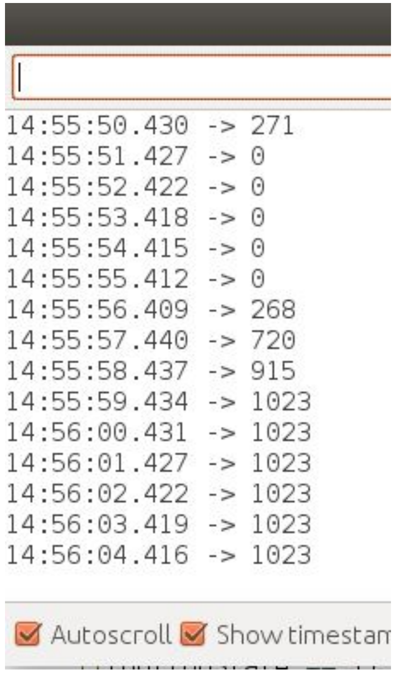
Rotary Angle Sensor



```
const int potentiometer = 0;

void setup()
{
  //set the serial communication frequency at 9600 bits per sec
    Serial.begin(9600);
    pinMode(potentiometer, INPUT);
}

void loop()
{
    int value = analogRead(potentiometer);
    Serial.println(value); //print the value on the serial monitor screen
    delay(1000); //wait 1000ms before printing next value
}
```

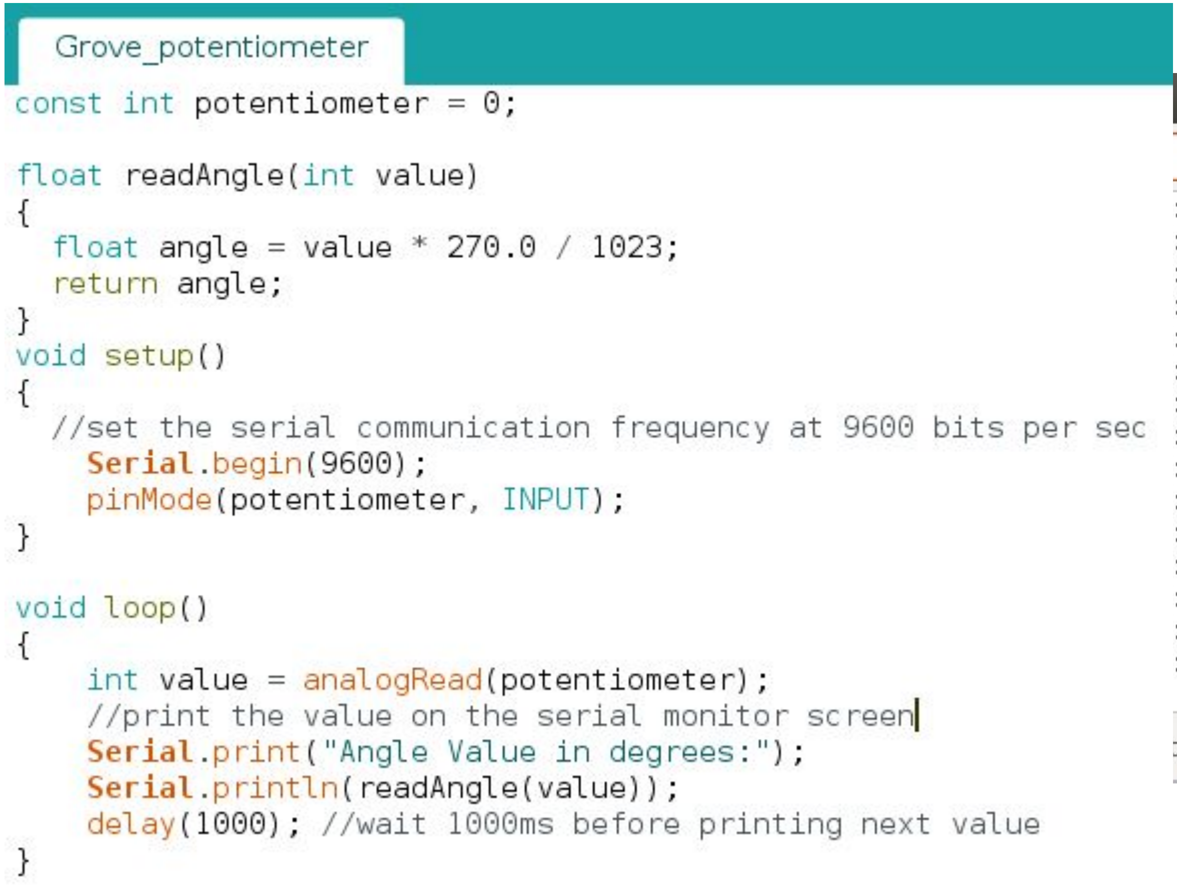




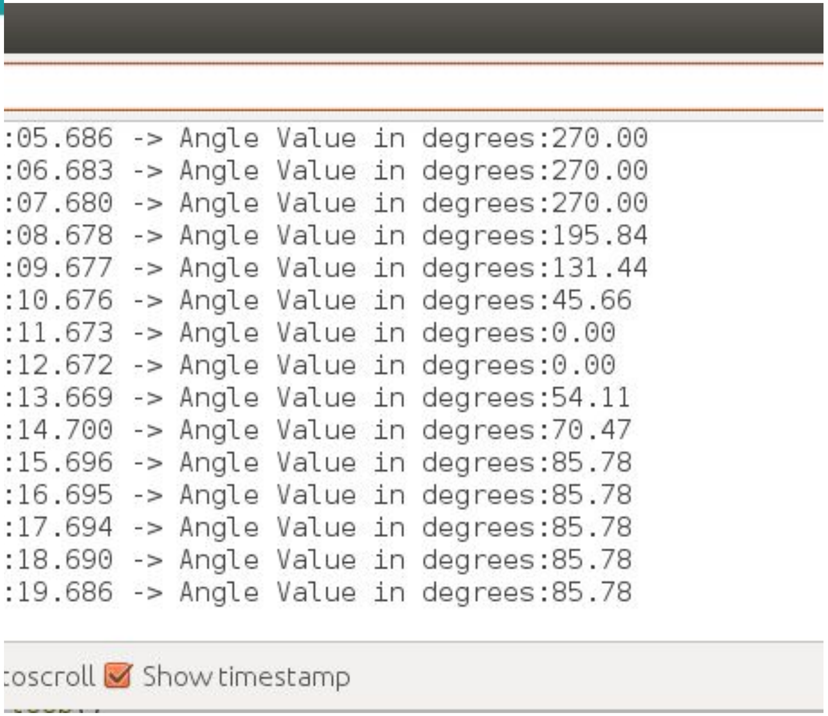


```
14:55:50.430 -> 271
14:55:51.427 -> 0
14:55:52.422 -> 0
14:55:53.418 -> 0
14:55:54.415 -> 0
14:55:55.412 -> 0
14:55:56.409 -> 268
14:55:57.440 -> 720
14:55:58.437 -> 915
14:55:59.434 -> 1023
14:56:00.431 -> 1023
14:56:01.427 -> 1023
14:56:02.422 -> 1023
14:56:03.419 -> 1023
14:56:04.416 -> 1023
```

Connect Rotary sensor to Port A0


Grove_potentiometer

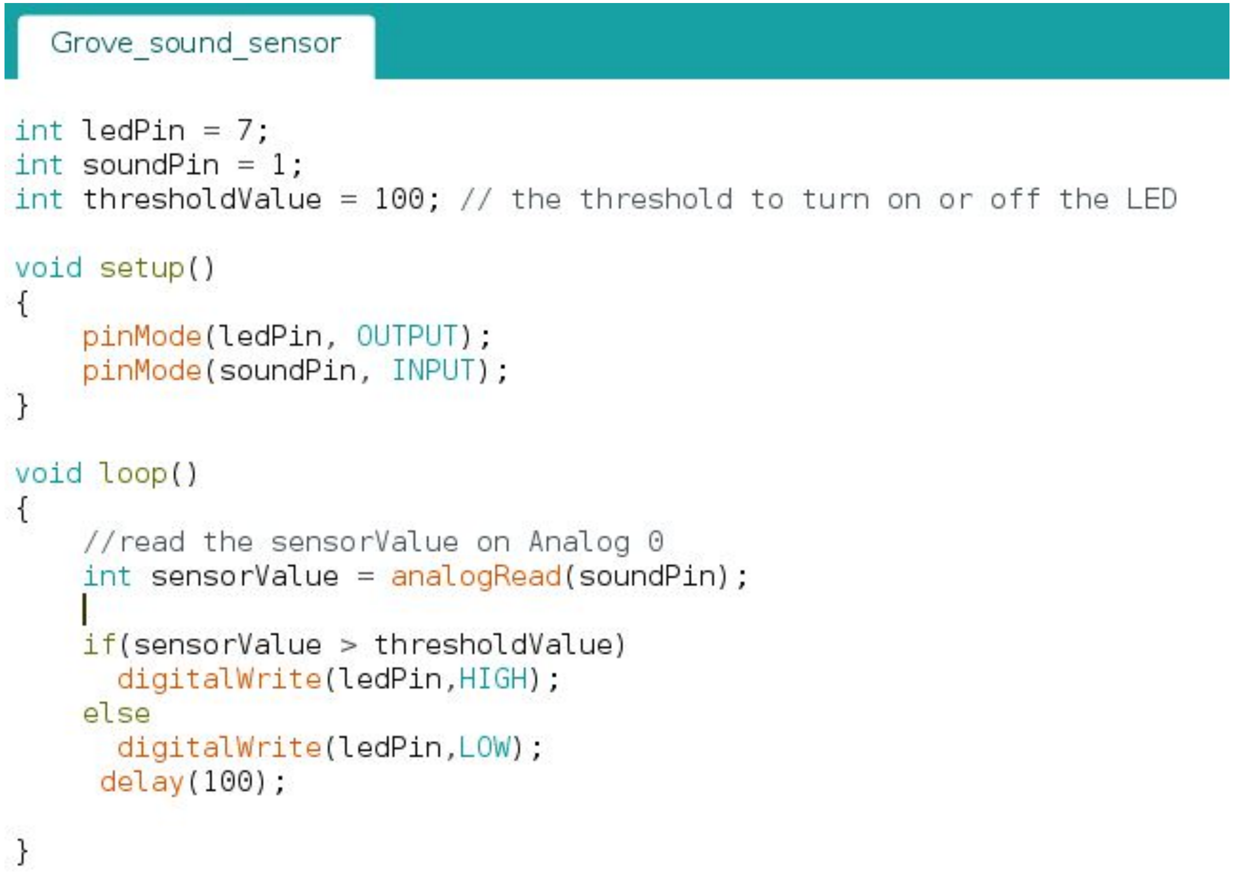
```
const int potentiometer = 0;

float readAngle(int value)
{
  float angle = value * 270.0 / 1023;
  return angle;
}
void setup()
{
  //set the serial communication frequency at 9600 bits per sec
    Serial.begin(9600);
    pinMode(potentiometer, INPUT);
}

void loop()
{
    int value = analogRead(potentiometer);
    //print the value on the serial monitor screen
    Serial.print("Angle Value in degrees:");
    Serial.println(readAngle(value));
    delay(1000); //wait 1000ms before printing next value
}
```



```
:05.686 -> Angle Value in degrees:270.00
:06.683 -> Angle Value in degrees:270.00
:07.680 -> Angle Value in degrees:270.00
:08.678 -> Angle Value in degrees:195.84
:09.677 -> Angle Value in degrees:131.44
:10.676 -> Angle Value in degrees:45.66
:11.673 -> Angle Value in degrees:0.00
:12.672 -> Angle Value in degrees:0.00
:13.669 -> Angle Value in degrees:54.11
:14.700 -> Angle Value in degrees:70.47
:15.696 -> Angle Value in degrees:85.78
:16.695 -> Angle Value in degrees:85.78
:17.694 -> Angle Value in degrees:85.78
:18.690 -> Angle Value in degrees:85.78
:19.686 -> Angle Value in degrees:85.78
```

Convert the readings into Angle Values in Degrees

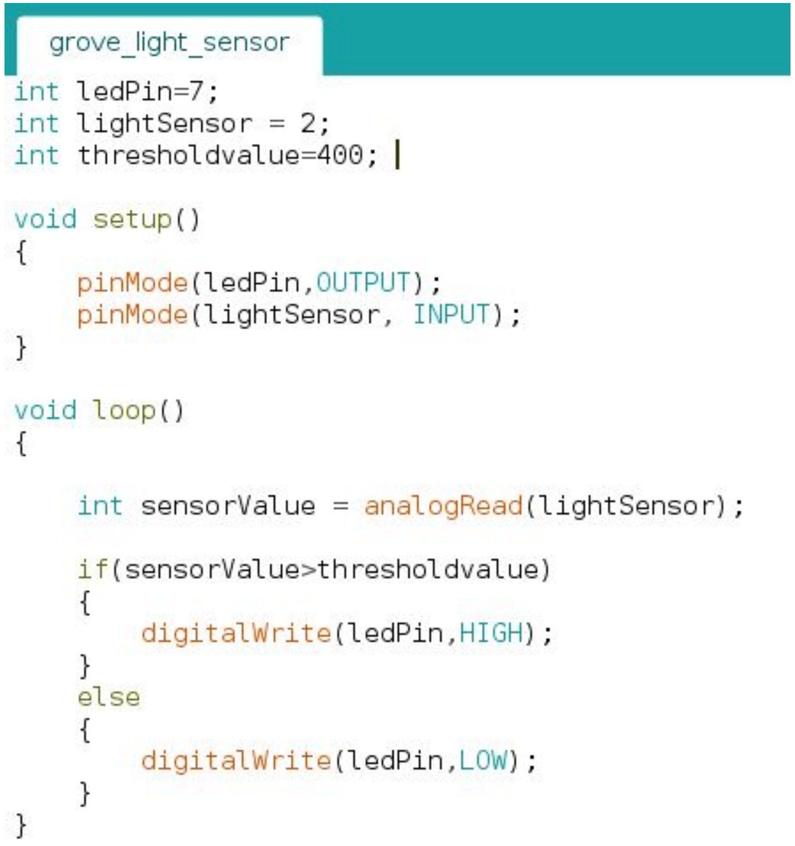
Sound Sensor



```
int ledPin = 7;
int soundPin = 1;
int thresholdValue = 100; // the threshold to turn on or off the LED

void setup()
{
    pinMode(ledPin, OUTPUT);
    pinMode(soundPin, INPUT);
}

void loop()
{
    //read the sensorValue on Analog 0
    int sensorValue = analogRead(soundPin);
    
    if(sensorValue > thresholdValue)
      digitalWrite(ledPin,HIGH);
    else
      digitalWrite(ledPin,LOW);
     delay(100);

}
```

- Connect Sound sensor to A1
- Connect LED to Port D7

Light Sensor

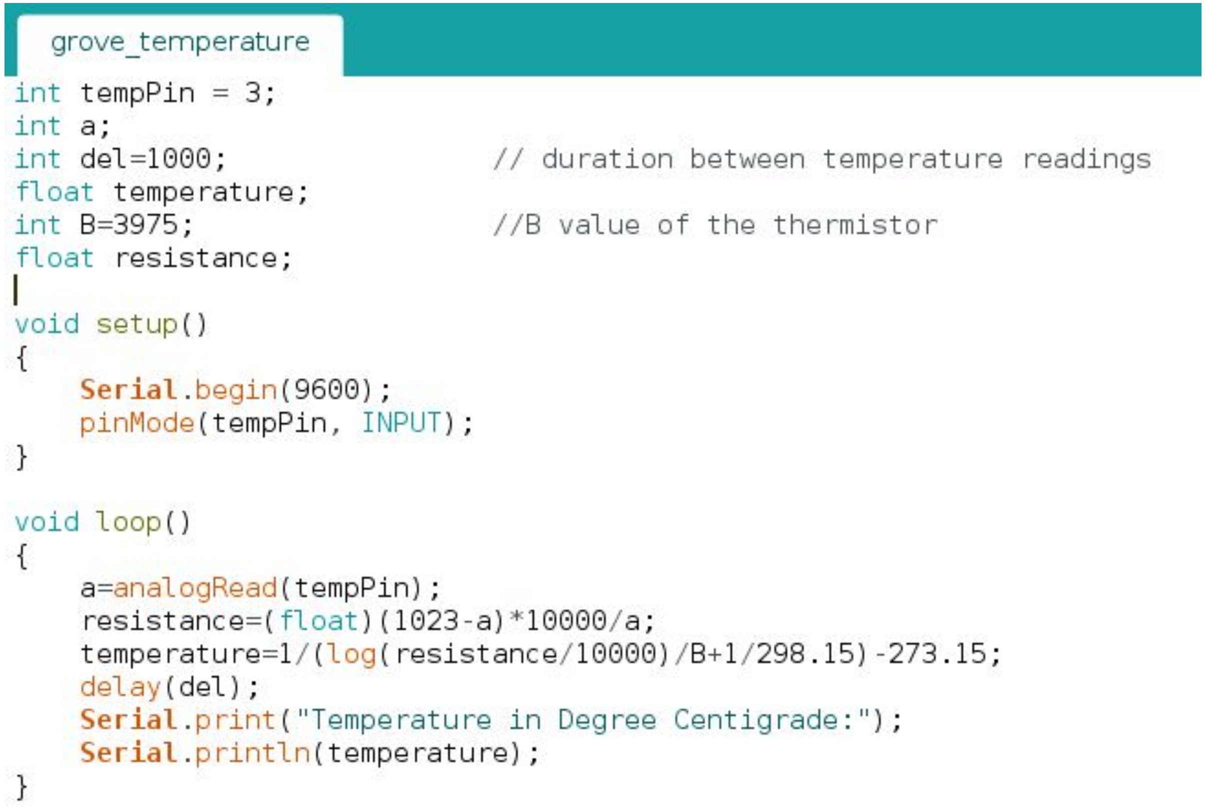


grove_light_sensor

```
int ledPin=7;
int lightSensor = 2;
int thresholdvalue=400;

void setup()
{
    pinMode(ledPin,OUTPUT);
    pinMode(lightSensor, INPUT);
}

void loop()
{

    int sensorValue = analogRead(lightSensor);

    if(sensorValue>thresholdvalue)
    {
        digitalWrite(ledPin,HIGH);
    }
    else
    {
        digitalWrite(ledPin,LOW);
    }
}
```

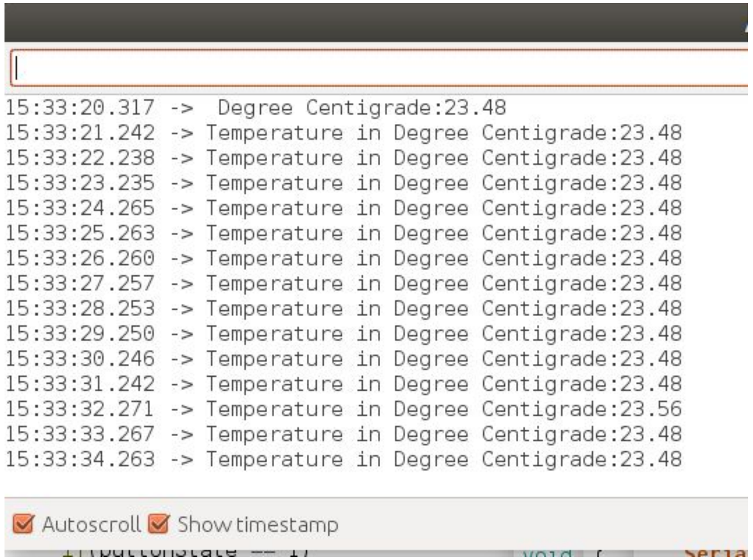
- Connect Light Sensor to Analog Port A2
- Connect LED to Digital Port D7

Temperature Sensor



grove_temperature

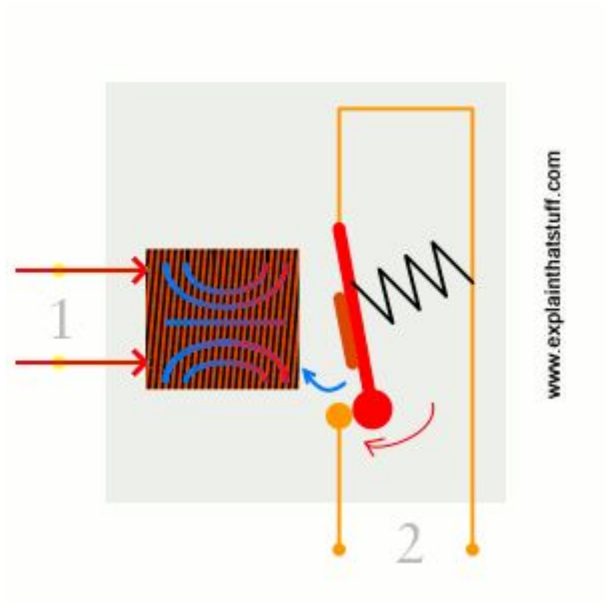
```
int tempPin = 3;
int a;
int del=1000;              // duration between temperature readings
float temperature;
int B=3975;                //B value of the thermistor
float resistance;

void setup()
{
    Serial.begin(9600);
    pinMode(tempPin, INPUT);
}

void loop()
{
    a=analogRead(tempPin);
    resistance=(float)(1023-a)*10000/a;
    temperature=1/(log(resistance/10000)/B+1/298.15)-273.15;
    delay(del);
    Serial.print("Temperature in Degree Centigrade:");
    Serial.println(temperature);
}
```

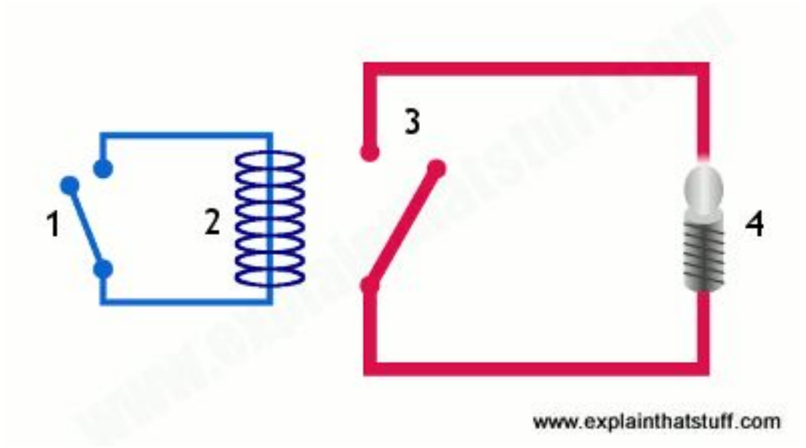


```
15:33:20.317 ->  Degree Centigrade:23.48
15:33:21.242 -> Temperature in Degree Centigrade:23.48
15:33:22.238 -> Temperature in Degree Centigrade:23.48
15:33:23.235 -> Temperature in Degree Centigrade:23.48
15:33:24.265 -> Temperature in Degree Centigrade:23.48
15:33:25.263 -> Temperature in Degree Centigrade:23.48
15:33:26.260 -> Temperature in Degree Centigrade:23.48
15:33:27.257 -> Temperature in Degree Centigrade:23.48
15:33:28.253 -> Temperature in Degree Centigrade:23.48
15:33:29.250 -> Temperature in Degree Centigrade:23.48
15:33:30.246 -> Temperature in Degree Centigrade:23.48
15:33:31.242 -> Temperature in Degree Centigrade:23.48
15:33:32.271 -> Temperature in Degree Centigrade:23.56
15:33:33.267 -> Temperature in Degree Centigrade:23.48
15:33:34.263 -> Temperature in Degree Centigrade:23.48
```

Connect Temperature Sensor to Analog Port A3

How does a Relay work?

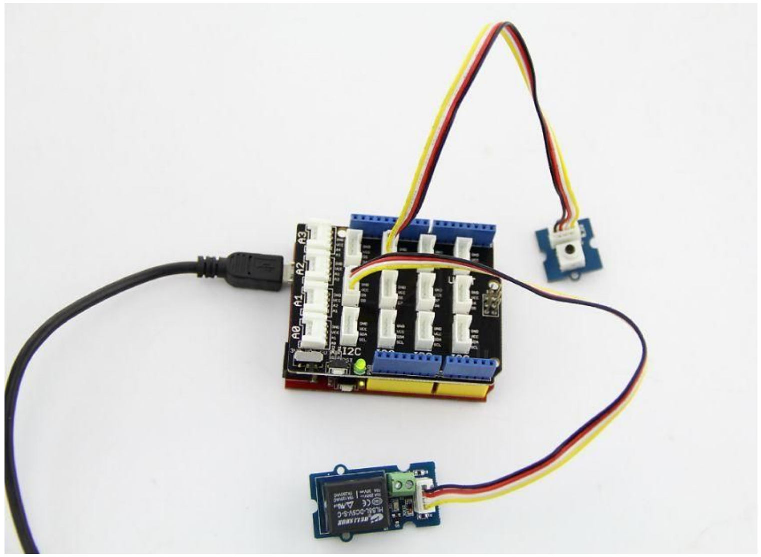




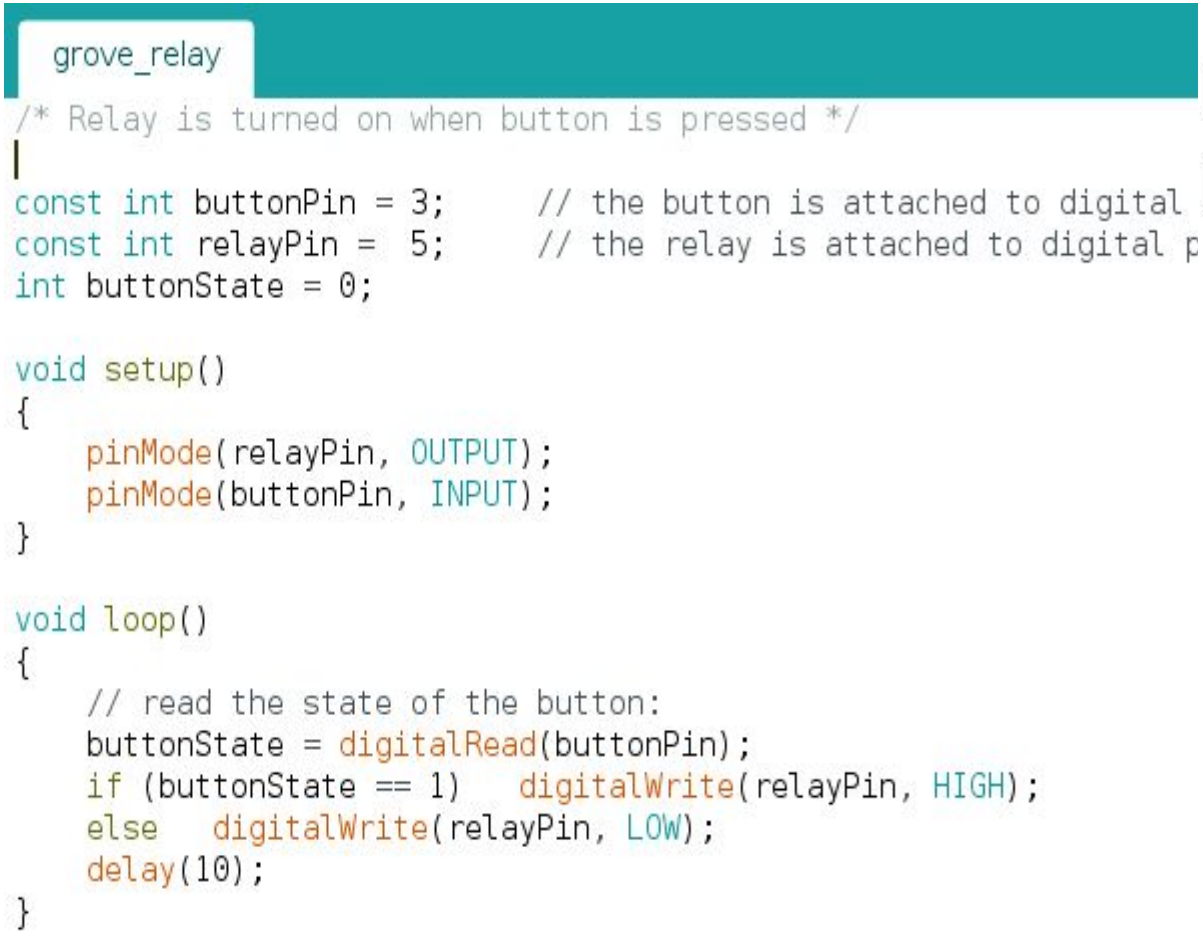
When current flows in one circuit, it controls (closes or opens) the switch in the second circuit. These two circuit could be operating at different voltage and power levels.

Relay Switch

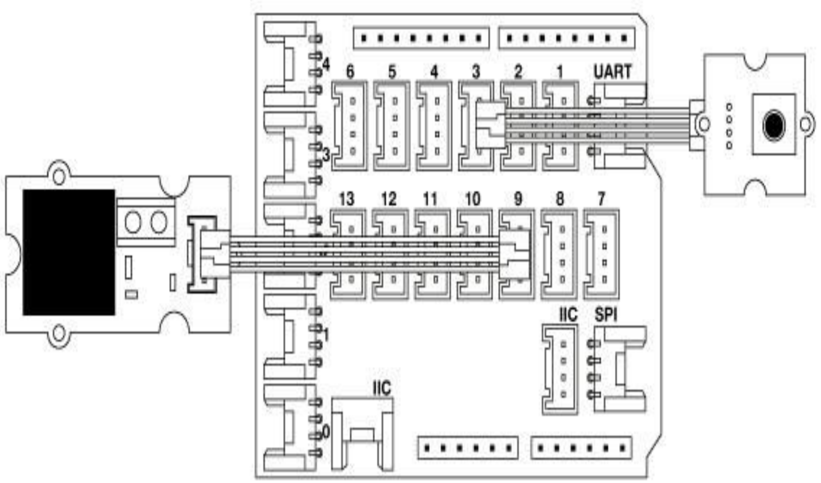
- A relay is an electromagnetic switch operated by a relatively small electric current that can turn on or off a much larger electric current.
- So, it can be used to control high voltage systems (eg. 220 V fan) using Arduino running on a mere 5V supply.
- In this experiment, We will use a Push-button to turn on and off a relay switch.



Relay Switch

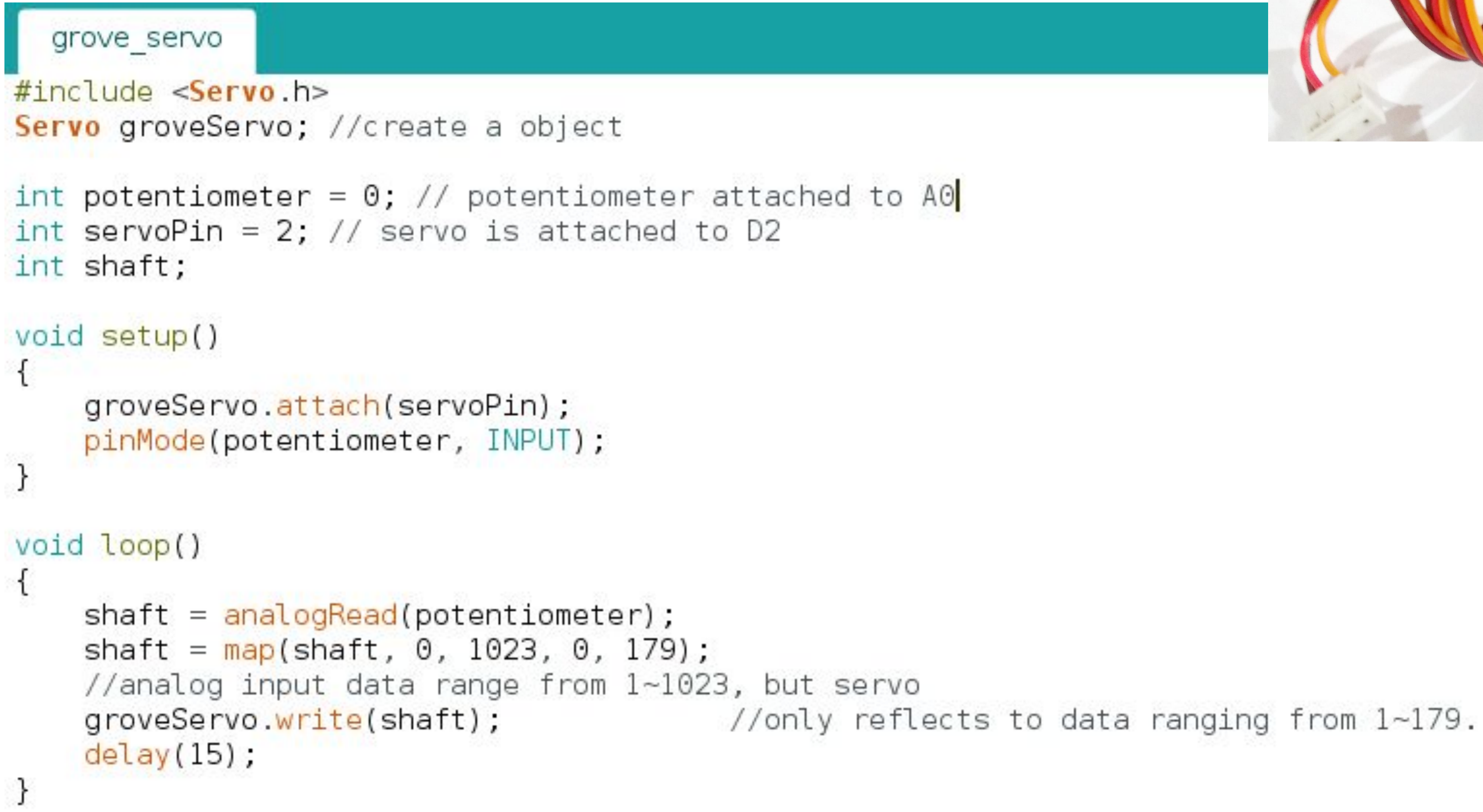

grove_relay

```
/* Relay is turned on when button is pressed */

const int buttonPin = 3;     // the button is attached to digital
const int relayPin =  5;     // the relay is attached to digital p
int buttonState = 0;

void setup()
{
    pinMode(relayPin, OUTPUT);
    pinMode(buttonPin, INPUT);
}

void loop()
{
    // read the state of the button:
    buttonState = digitalRead(buttonPin);
    if (buttonState == 1)   digitalWrite(relayPin, HIGH);
    else   digitalWrite(relayPin, LOW);
    delay(10);
}
```

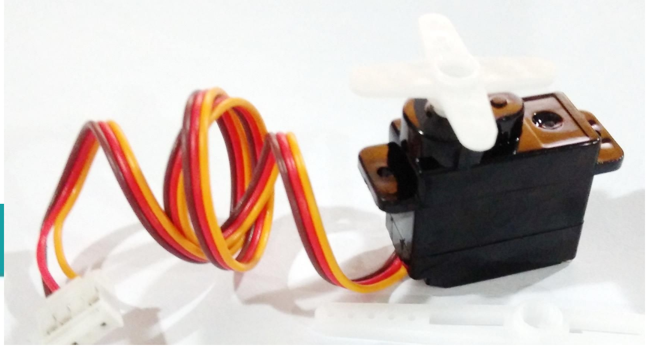


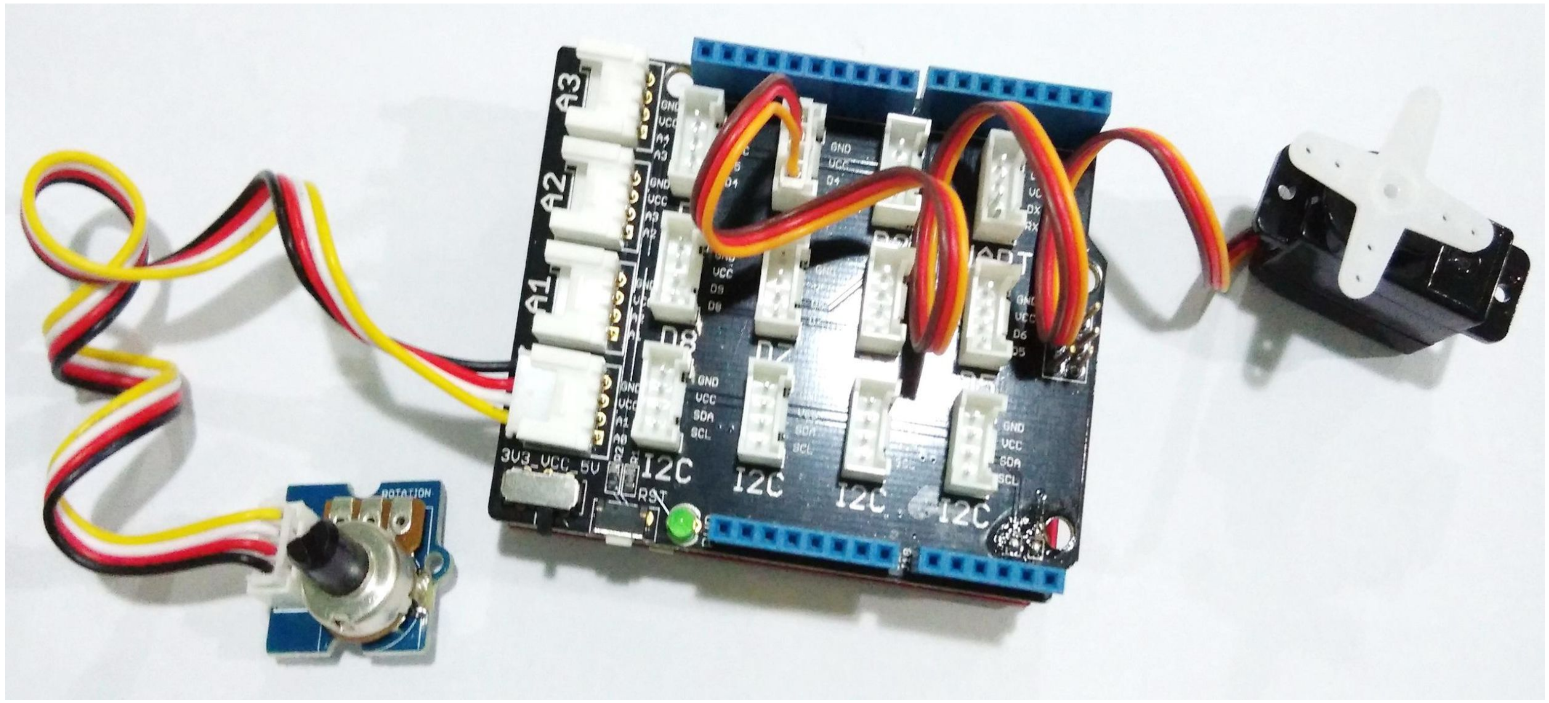
- Connect Relay to D5
- Connect Button to D3

Servo Motor


grove_servo

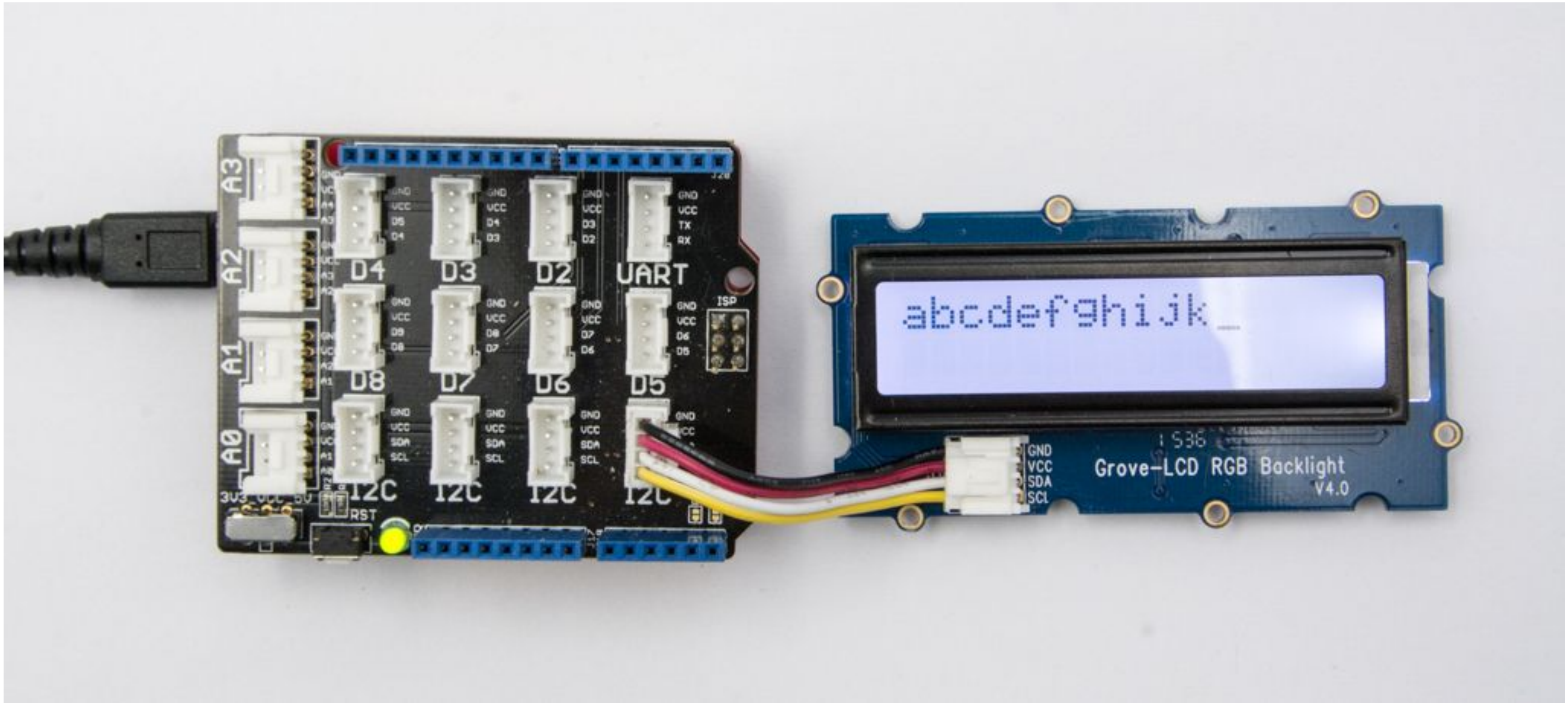
```
#include <Servo.h>
Servo groveServo; //create a object

int potentiometer = 0; // potentiometer attached to A0
int servoPin = 2; // servo is attached to D2
int shaft;

void setup()
{
    groveServo.attach(servoPin);
    pinMode(potentiometer, INPUT);
}

void loop()
{
    shaft = analogRead(potentiometer);
    shaft = map(shaft, 0, 1023, 0, 179);
    //analog input data range from 1~1023, but servo
    groveServo.write(shaft);              //only reflects to data ranging from 1~179.
    delay(15);
}
```





- Servo Motor connected to D2
- Rotational Potentiometer Connected to Analog Port A0

RGB LCD Backlit



Software Library for LCD Experiment

- Install the LCD Library from this link:

  http://wiki.seeedstudio.com/Grove-LCD_RGB_Backlight/

- Instructions for installing Library:

  http://wiki.seeedstudio.com/How_to_install_Arduino_Library/

- More examples with LCD is available at:

  File -> Examples -> Grove LCD RGB Backlit ->

```
#include <Wire.h>
#include "rgb_lcd.h"

rgb_lcd lcd;

const int colorR = 255;
const int colorG = 0;
const int colorB = 0;

void setup()
{
    // set up the LCD's number of columns and rows:
    lcd.begin(16, 2);

    lcd.setRGB(colorR, colorG, colorB);

    // Print a message to the LCD.
    lcd.print("hello, world!");

    delay(1000);
}

void loop()
{
    // set the cursor to column 0, line 1
    // (note: line 1 is the second row, since counting begins with 0):
    lcd.setCursor(0, 1);
    // print the number of seconds since reset:
    lcd.print(millis()/1000);

    delay(100);
}
```