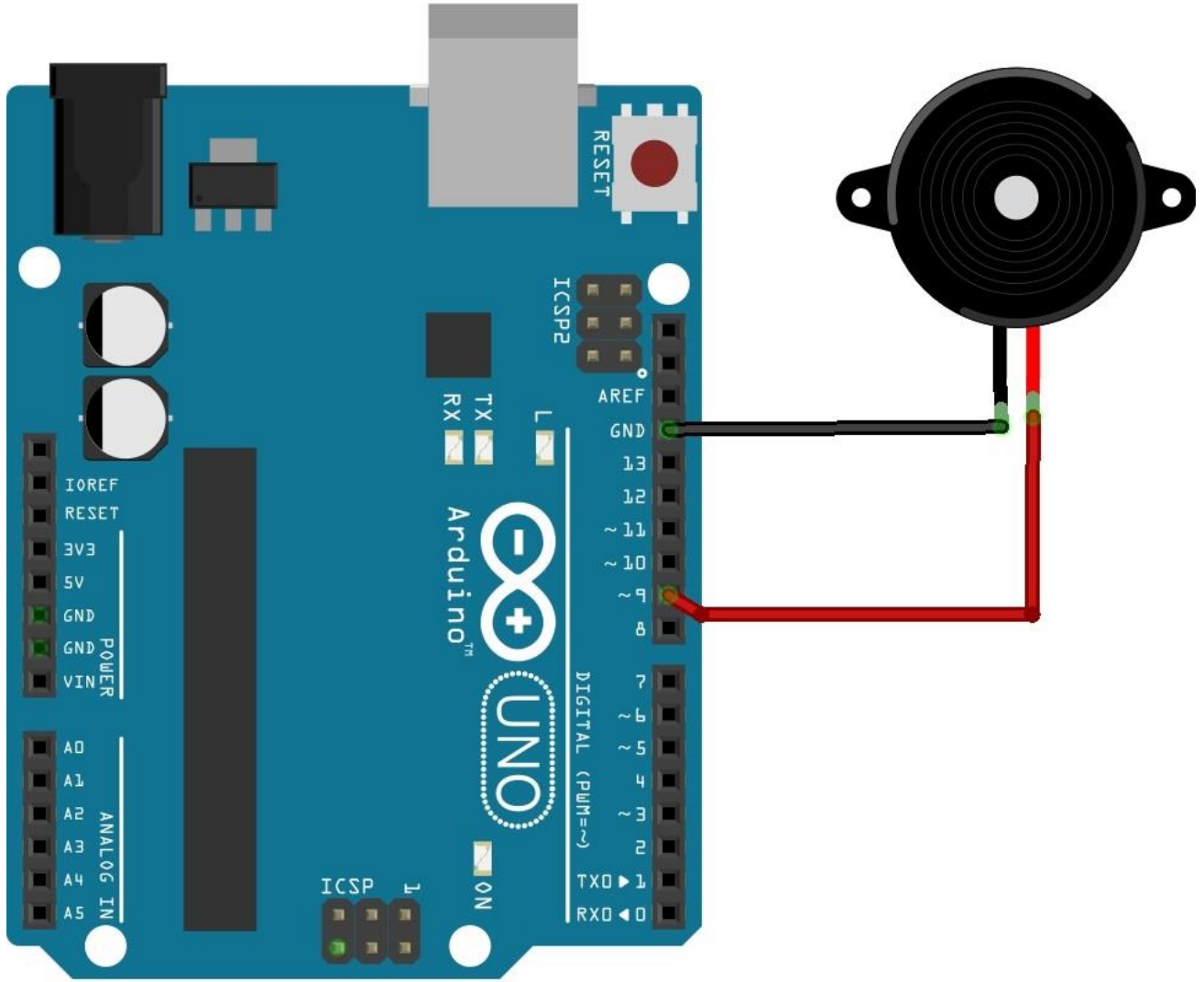
Interfacing Buzzer with Arduino

- Piezzo Buzzer
- No switching Circuit required
- Can be made on / off
- Tone generation
- Used for sound generation
- Audio feedback

Buzzer



Programmed as Digital Output
Write High = +5v on Pin = Buzzer HIGH
Write Low = 0v on Pin = Buzzer LOW

tone library

- Generates a square wave of specified frequency
- 50% duty Cycle, on time = off time
- tone(pin, frequency)
- tone(pin, frequency, duration in ms)
- noTone()
- It is not possible to generate tones lower than 31Hz.
- PWM operation issue with pin 3 and 11 with tone library

Serial Port

Simplest output Device for Arduino

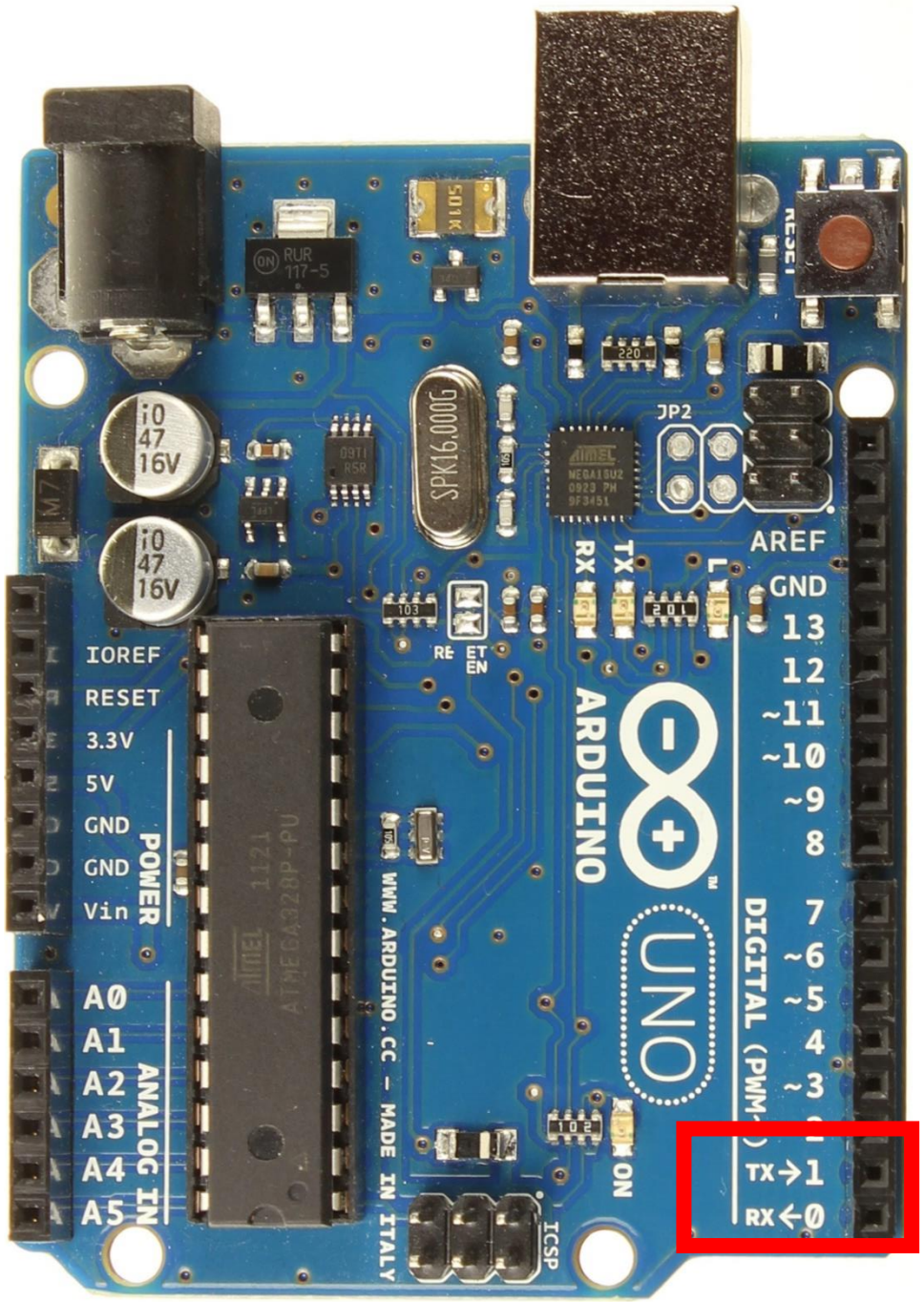
Serial Communication

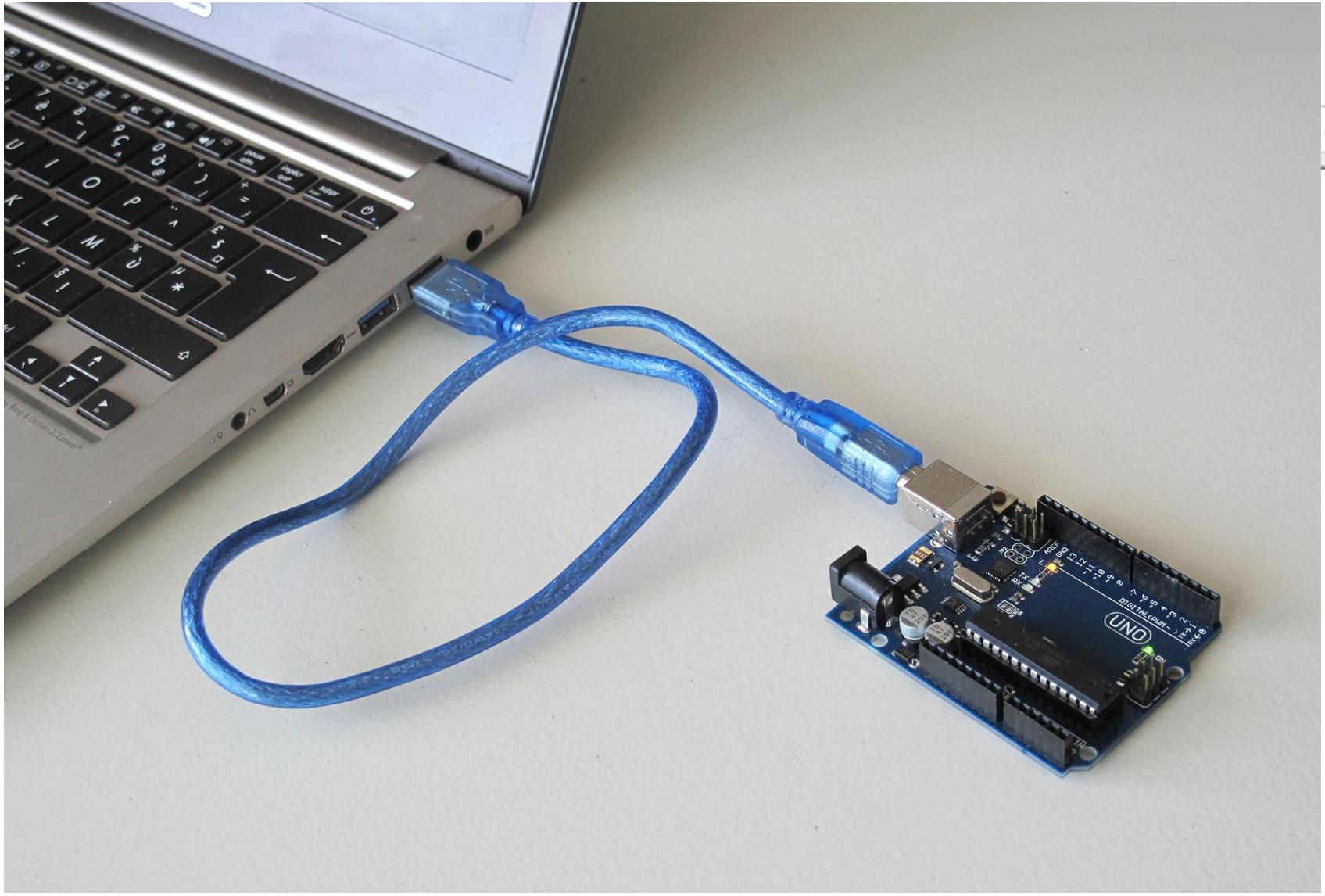
Serial Communication is the transferring and receiving of information between two machines.

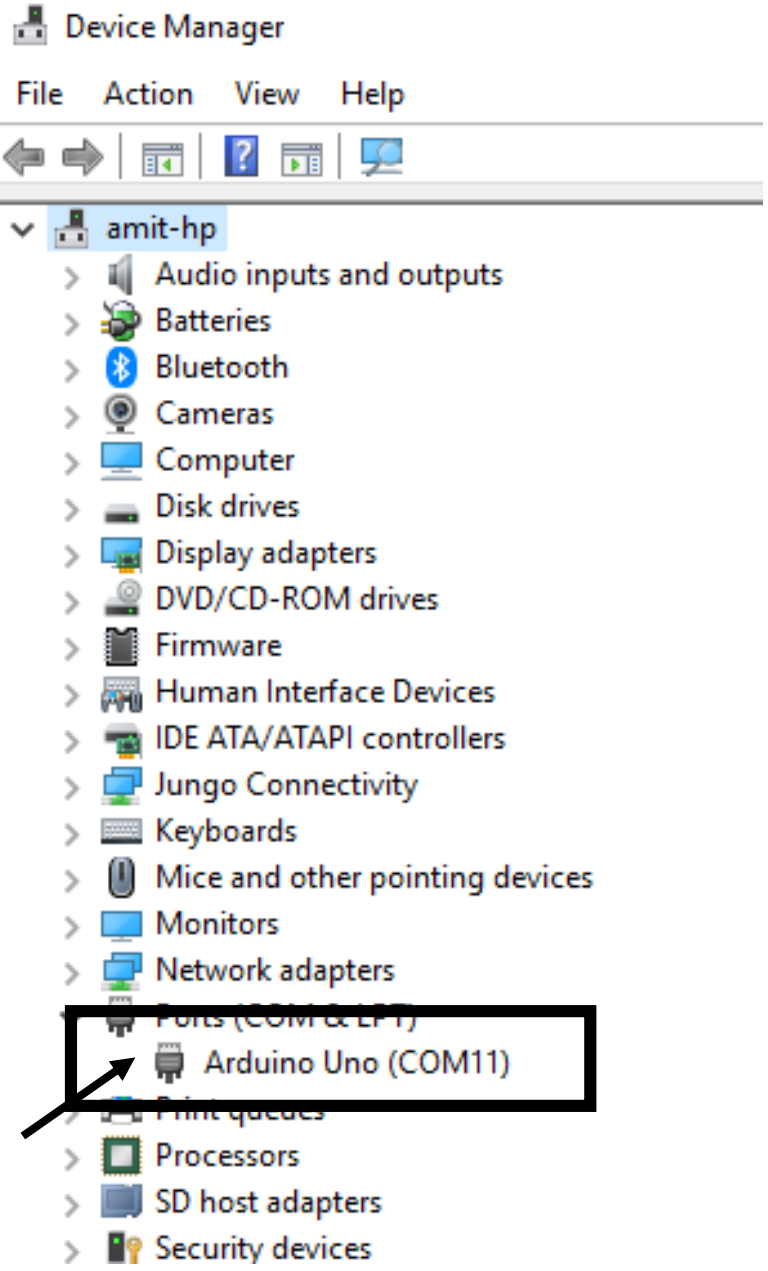
The Arduino has pin # 0 to receive information

and

pin 1 to transmit information



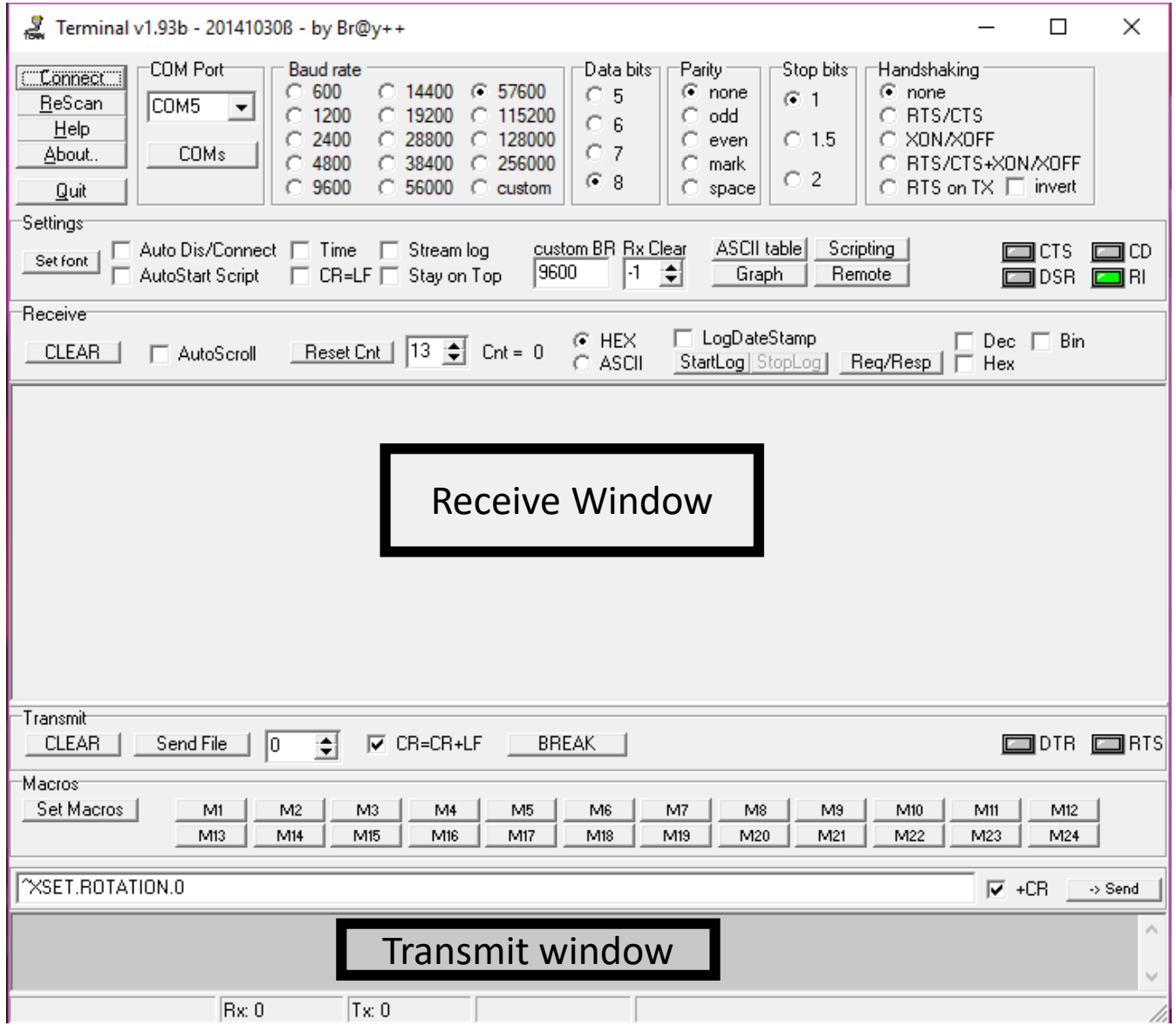




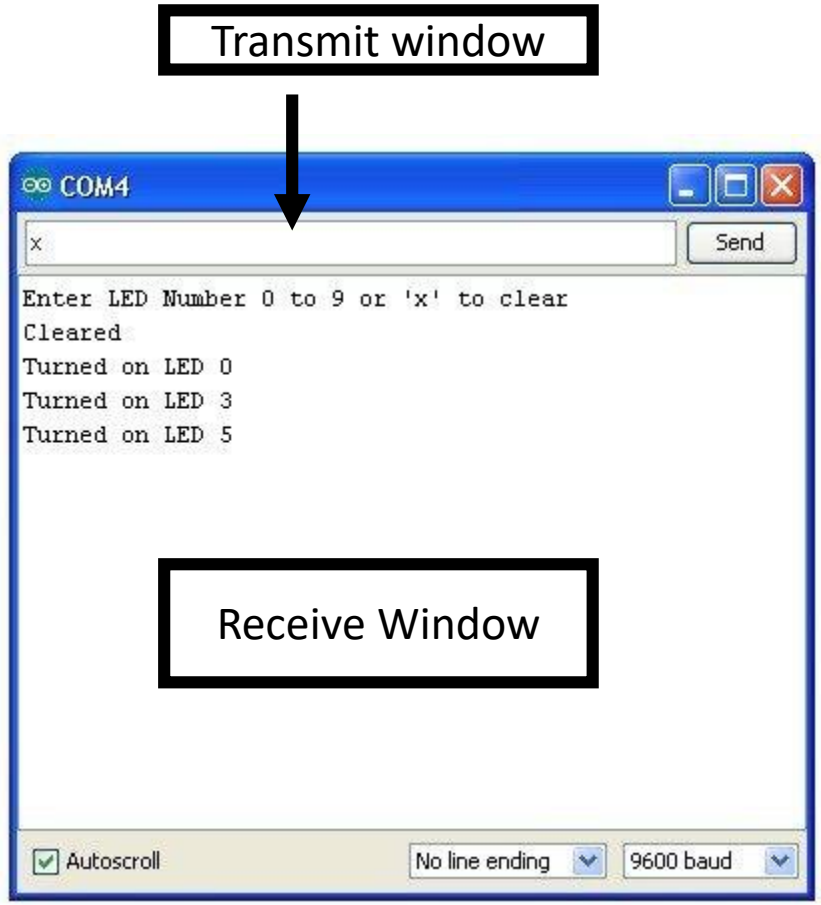
USB Connection to PC
Serial Communication with Computer

Check the COM
No

Serial Terminal PC

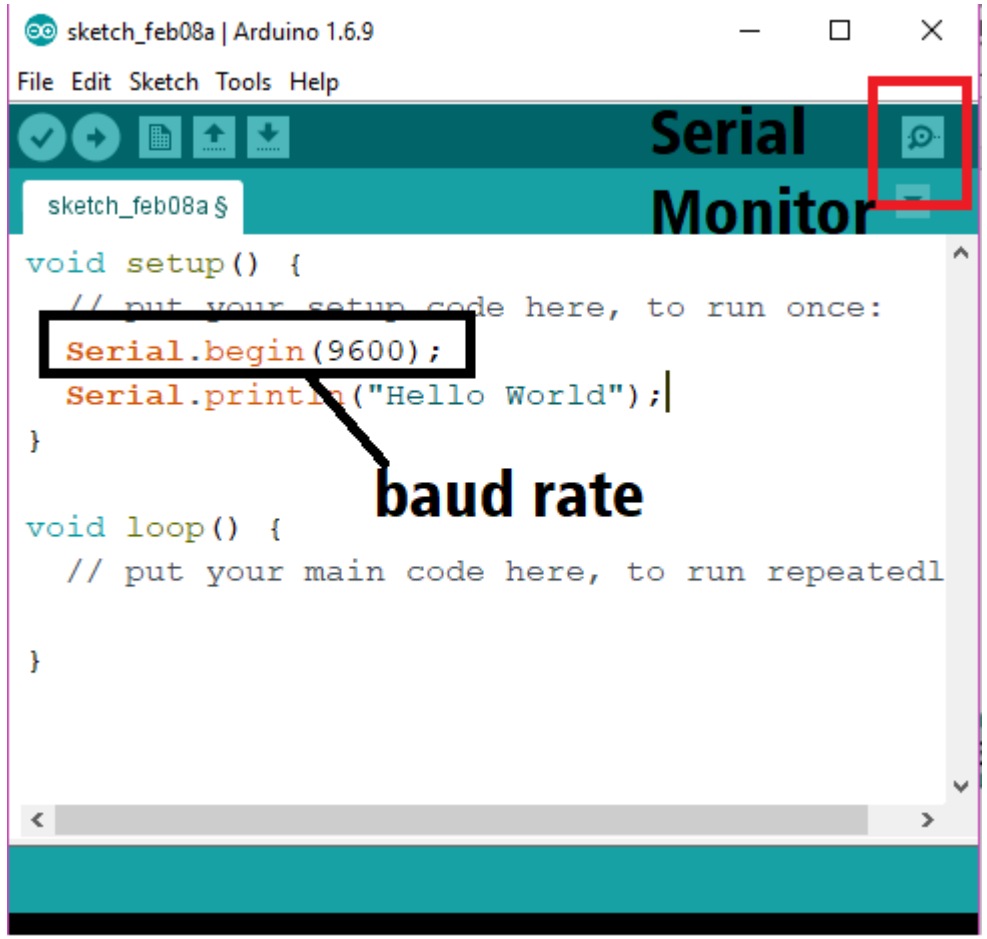


Terminal by Br@y



Arduino Terminal

Serial in Setup



Serial Port of Arduino

- USB Connection of Arduino goes to PC
- PC Terminal software can read and write
- Easiest and simplest output device
- Data sent serially
- Need to fix the baud rate
- Pin 0 and 1 used

- Supported Baud rates
- 300
- 1200
- 2400
- 4800
- 9600
- 19200
- 38400
- 57600
- 115200 …

Serial Functions

- if(Serial)
  available()
  availableForWrite()
  begin()
  end()
  find()
  findUntil()
  flush()
  parseFloat()
  parseInt()
  peek()

- print()
  println()
  read()
  readBytes()
  readBytesUntil()
  readString()
  readStringUntil()
  setTimeout()
  write()
  serialEvent()

Mostly Used Serial Functions

- Serial.begin(baud)
- Serial.print()
- Serial.println()
- Serial.write()
- If(Serial.available() > 0)
- Serial.read

Serial Communication: Serial Setup

```
void setup ( )
{
    Serial.begin(9600);
}
```

In this case the number 9600 is the baud rate at which the computer and Arduino communicate

2400, 4800, 9600,19200, … 115200

Serial Communication: Sending a Message / Variable

```
void loop ( )
{
  Serial.print("Hello");
  Serial.println(a);
}
```

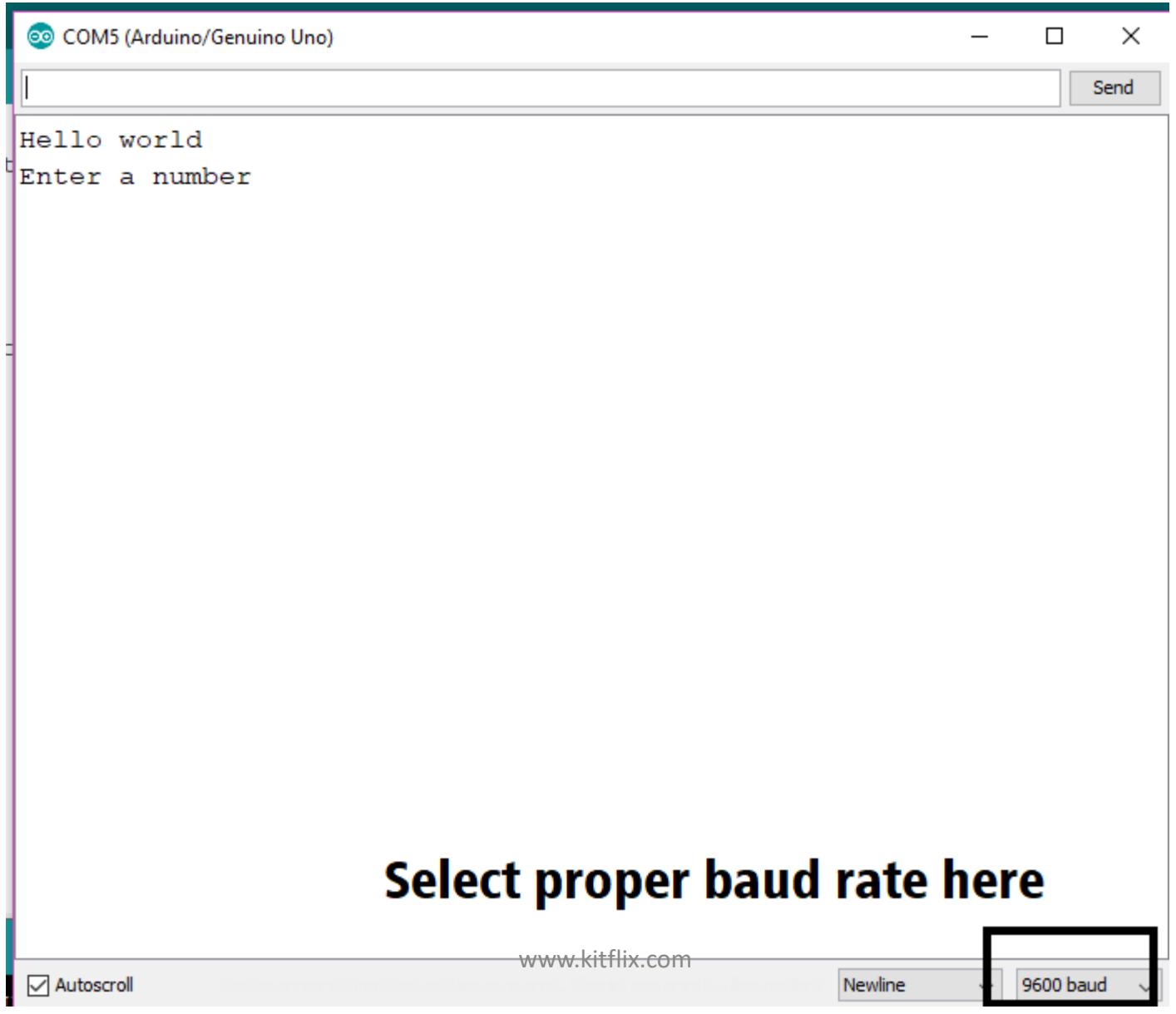
Arduino Code

Serial Communication: Receive a Message / Variable

```
void loop ( )
{
   if(Serial.available() > 0)
   {
      int rxd = Serial.read();
   }
}
```

Arduino Code

Serial Communication



Serial Hello World example

Liquid Crystal Display

- Text LCD
- Graphics LCD
- TFT
- OLED

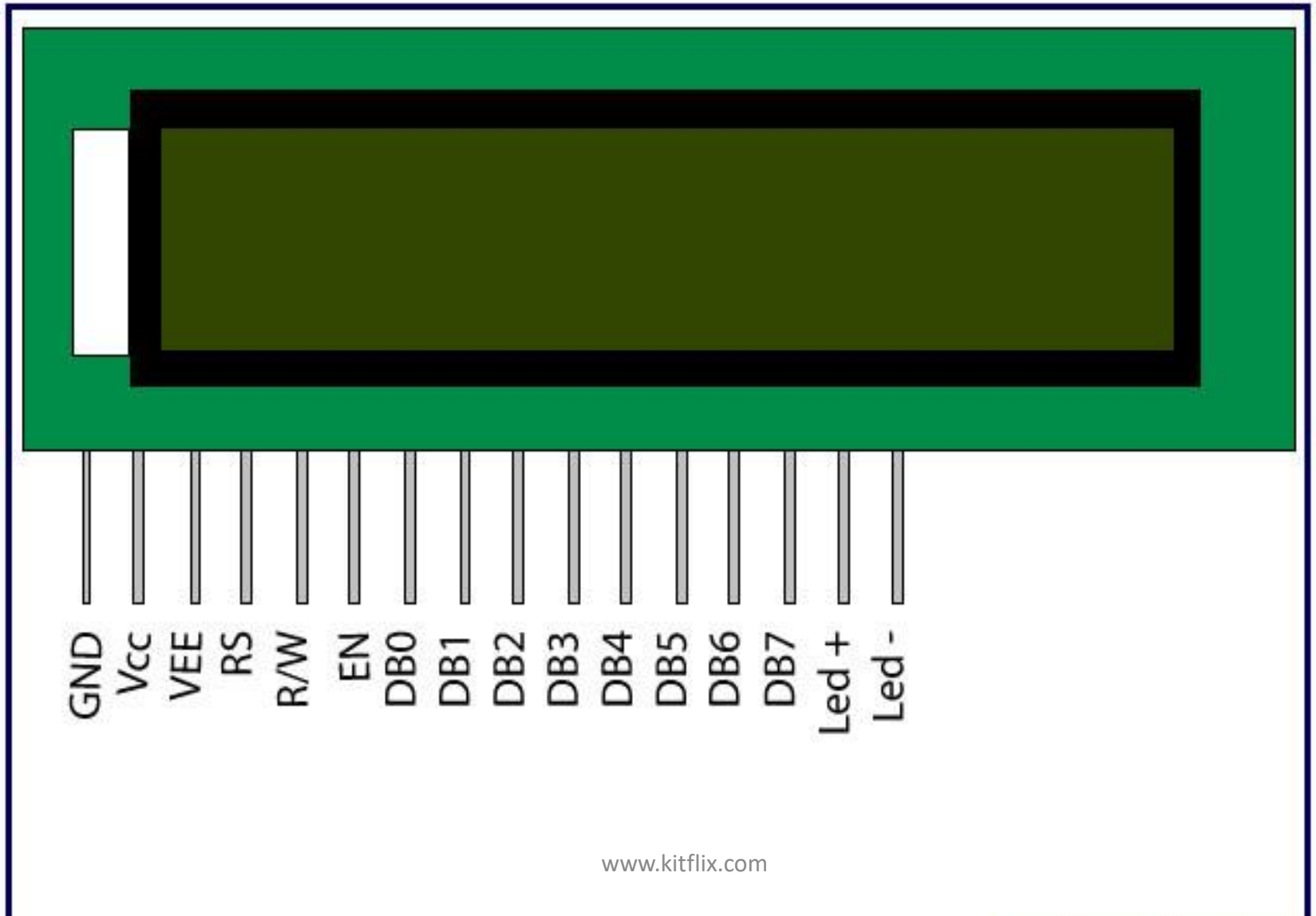
Types of LCD

- LCD Comes in majorly two flavors
  - Alphanumeric LCD
    - Capable of Displaying Alphanumeric characters only
    - Available options are 16×1, 16×2, 20×2, 40×2
  - Graphical LCD
    - Capable of Displaying all type of Graphical Characters
    - Available options are 122×32, 128×64, 240×64, 240×128

Basic of LCD

- LCD (Liquid Crystal Display) screen is an electronic display module and is used in wide range of applications.
- A 16x2 LCD display is popular
- A 16x2 LCD means it can display 16 characters per line and there are 2 such lines.
- In this LCD each character is displayed in 5x7 pixel matrix.
- This LCD has two registers, namely, Command and Data.
- The command register stores the command instructions given to the LCD
- The data register stores the data to be displayed on the LCD.

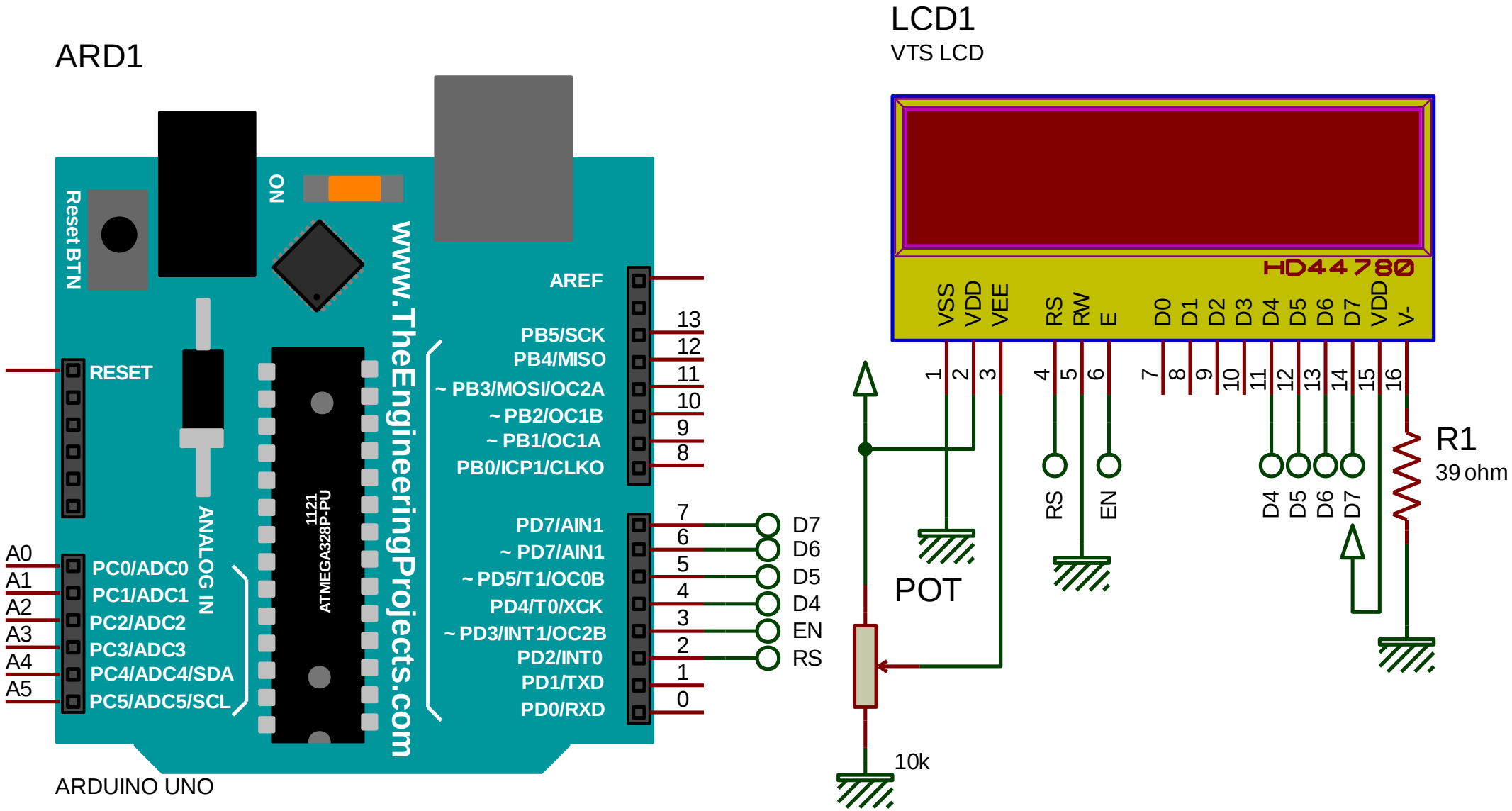
Liquid Crystal Display



LCD Pin-Out

Pin No	Function	Name
1	Ground (0V)	Ground
2	Supply voltage; 5V (4.7V – 5.3V)	Vcc
3	Contrast adjustment; through a variable resistor	V_{EE}
4	Selects command register when low; and data register when high	Register Select
5	Low to write to the register; High to read from the register	Read/write
6	Sends data to data pins when a high to low pulse is given	Enable
7	8-bit data pins	DB0
8		DB1
9		DB2
10		DB3
11		DB4
12		DB5
13		DB6
14		DB7
15	Backlight V_{CC} (5V)	Led+
16	Backlight Ground (0V)	Led-

Interfacing of Arduino and LCD



LiquidCrystal

- #include <LiquidCrystal.h>
- LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
- lcd.begin(16, 2);
- lcd.print("hello, world!");
- lcd.setCursor(colomn, row); // 0 / 15, 0-1
- lcd.clear();
- lcd.home();

LiquidCrystal Library of Arduino

- LiquidCrystal()
- begin()
- clear()
- home()
- setCursor()
- write()
- print()
- cursor()
- noCursor()
- blink()
- noBlink()
- display()
- noDisplay()
- scrollDisplayLeft()
- scrollDisplayRight()
- autoscroll()
- noAutoscroll()
- leftToRight()
- rightToLeft()
- createChar()

LiquidCrystal Library of Arduino

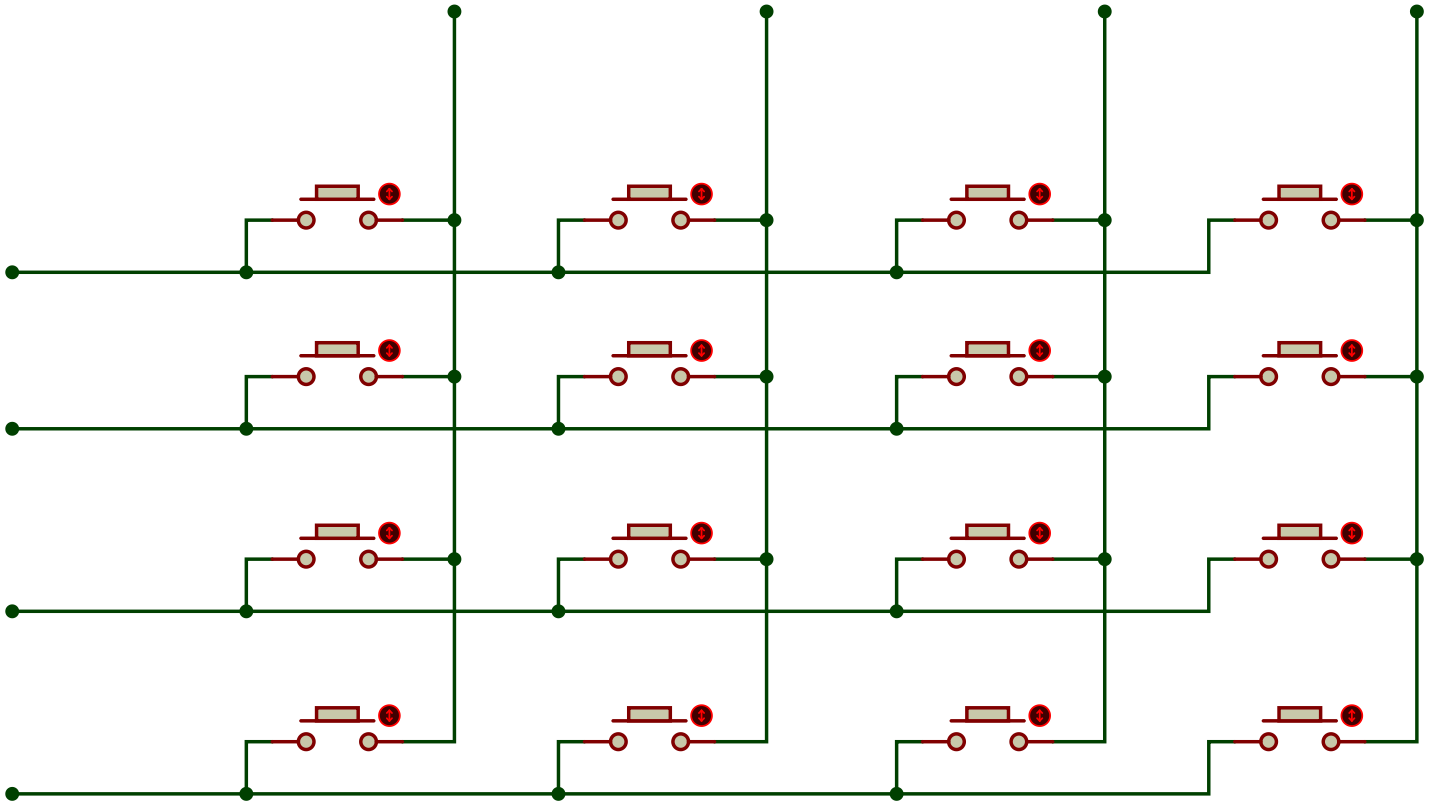
```
#include <LiquidCrystal.h>
const int rs = 2, en = 3, d4 = 4, d5 = 5, d6 = 6, d7 = 7;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
void setup() {
lcd.begin(16, 2);
lcd.setCursor(0,0);
lcd.print("Vidya Robotics");
}
void loop()
{
lcd.setCursor(0, 1);
lcd.print(millis() / 1000);
}
```

Tasks with LCD

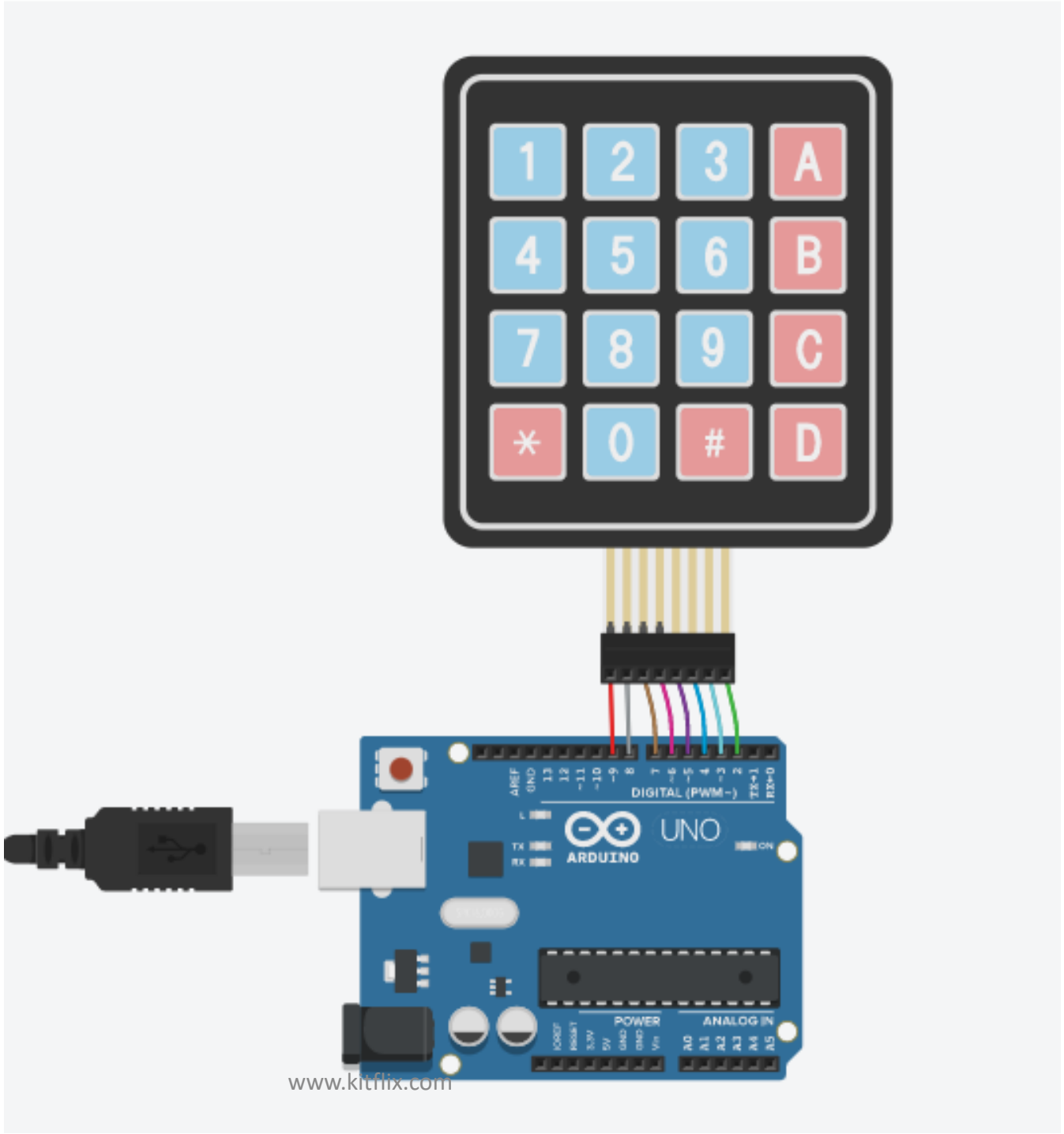
- Simple Test of millis() function
- Up counter using LCD
  - Single switch, and whenever that switch is pressed, count incremented and printed on lcd
- 2 switches and up – down counter using LCD

Keypad

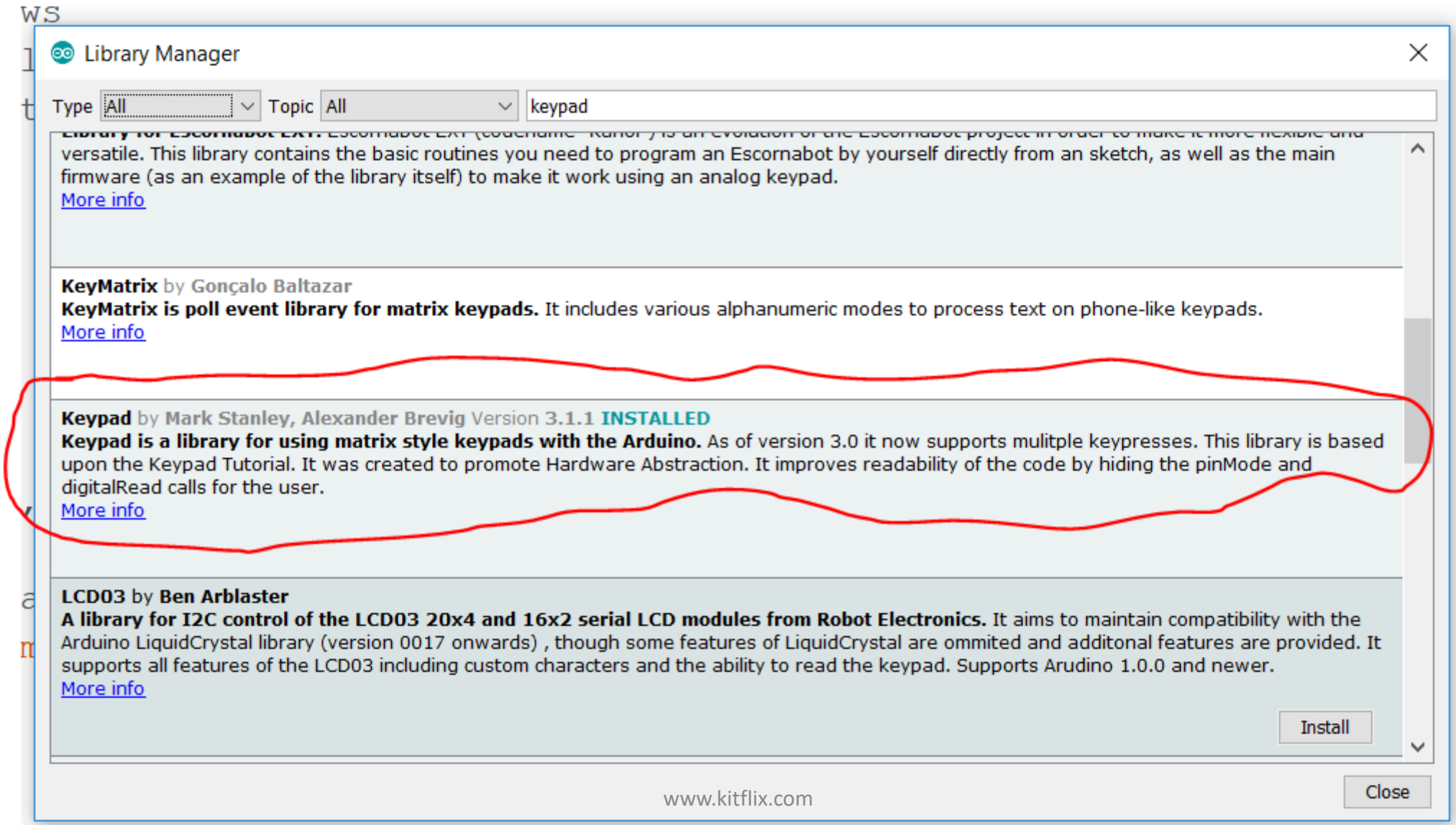
Keypad



Keypad Interfacing with Arduino



Keypad Library



Keypad Code

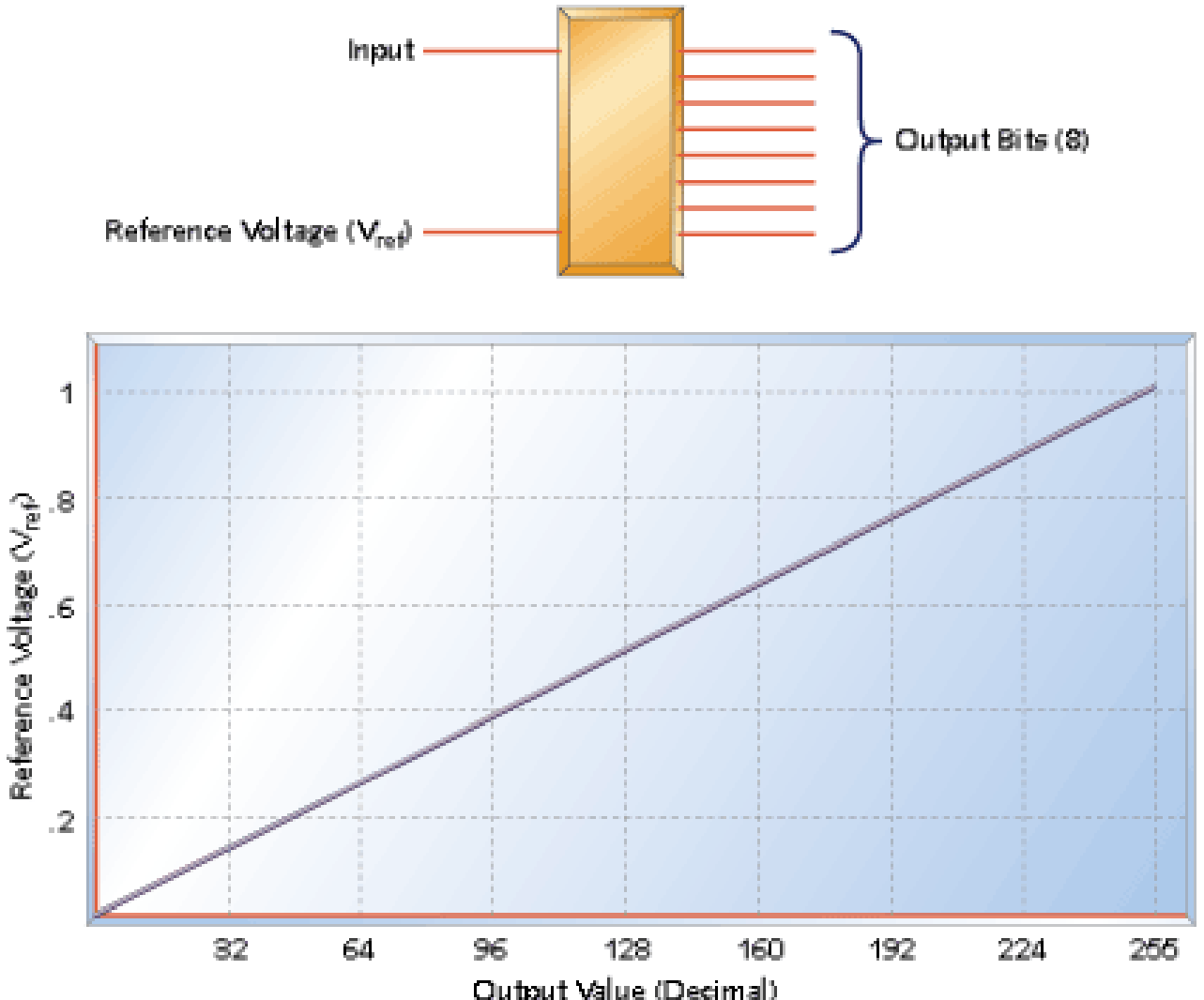
- Accept a key and print on LCD
- Accept a key and print on Serial Port
- #include <Keypad.h>
- byte rowPins[ROWS] = {7, 6, 5, 4}; //connect to the row pinouts of the keypad
- byte colPins[COLS] = {3, 2, A4, A5}; //connect to the column pinouts of the keypad
- Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);
- char customKey = customKeypad.getKey();

Analog Inputs

- What is analog signal?
- Differentiate Analog and Digital Signals
- Why ADC is required at all
- How to eliminate need of ADC
- ADC Concepts

ADC (cont)

Figure 1: Simple ADC

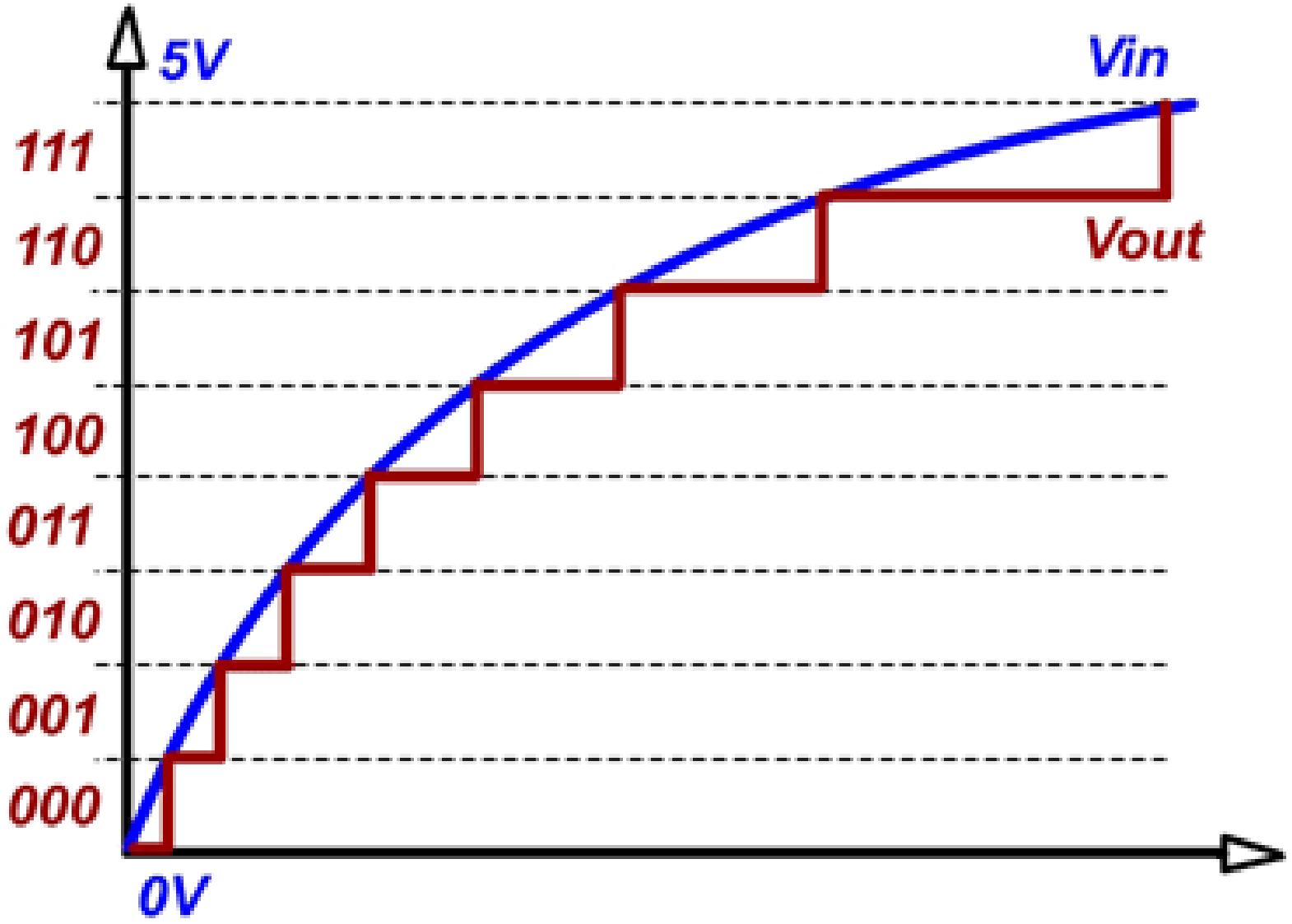


ADC Concepts

- Resolution
  - No of output bits of ADC
  - Generally 8-bit, 10-bit ADC are available
  - 8-bit ADC means resolution is 256
  - 10-bit ADC means resolution is 1024
- Vref
  - Vref is the maximum voltage that has to be converted
  - Most ADC's do have selectable Vref
- Step Size
  - Depends upon the Resolutions of ADC

ADC formula

- Step Size = Vref /Resolution
  - For Arduino 10- bit adc and 5v ref,
    - Step = 5 / 1024= 4.88mV
- Digital Out = Analog in / step size
- Analog in = Digital Out x Step size


5V
Vin
111
110
Vout
101
100
011
010
001
000
0V

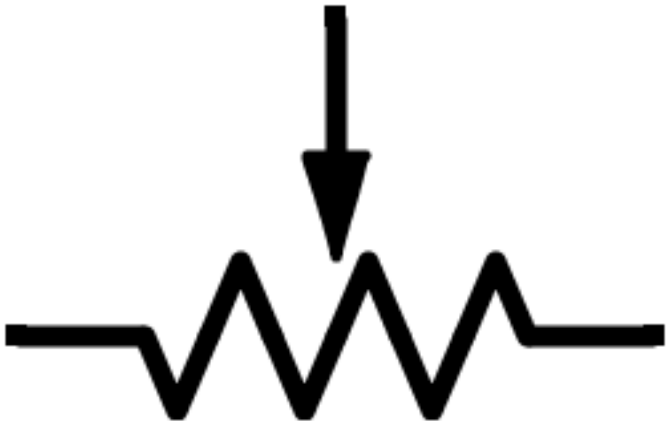
ADC of Arduino

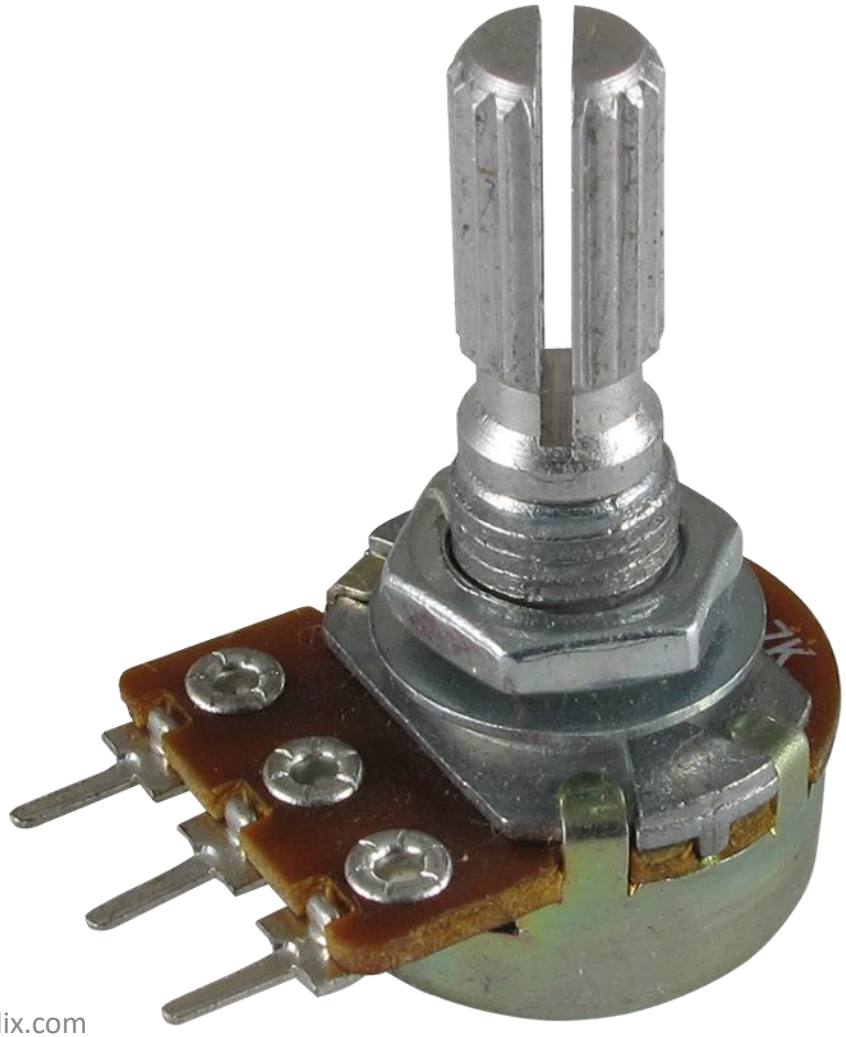
- 10- Bit ADC
- 0-5V Input signal
- 0-1023 Output

Project

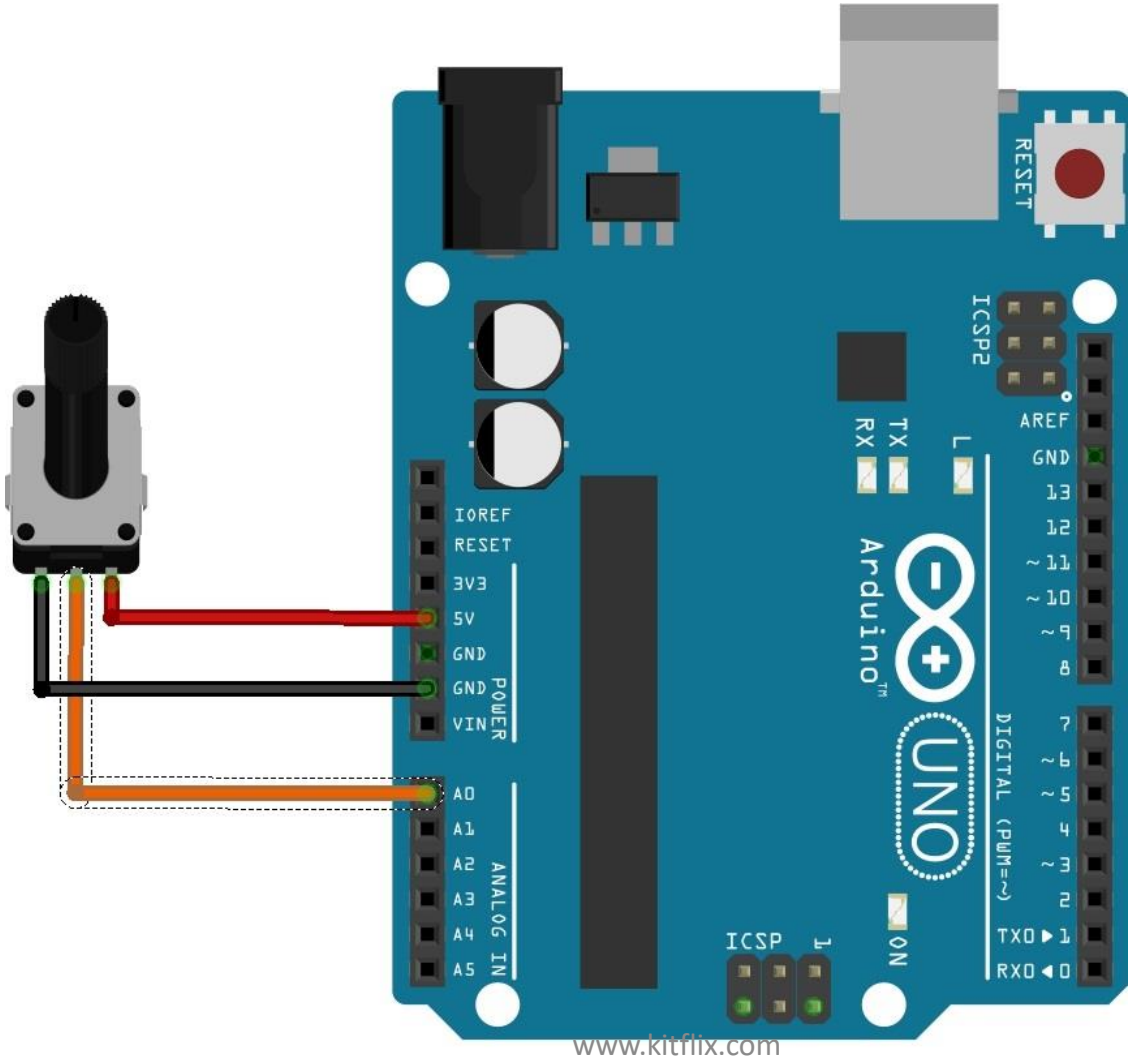
Digital Voltmeter

Potentiometer





With Arduino



Algorithm

- Declare a variable
- Read Analog Read into this
- Convert analog read to millivolts by
- millivolts = analog read * 4.88
- Print millivolts on serial port

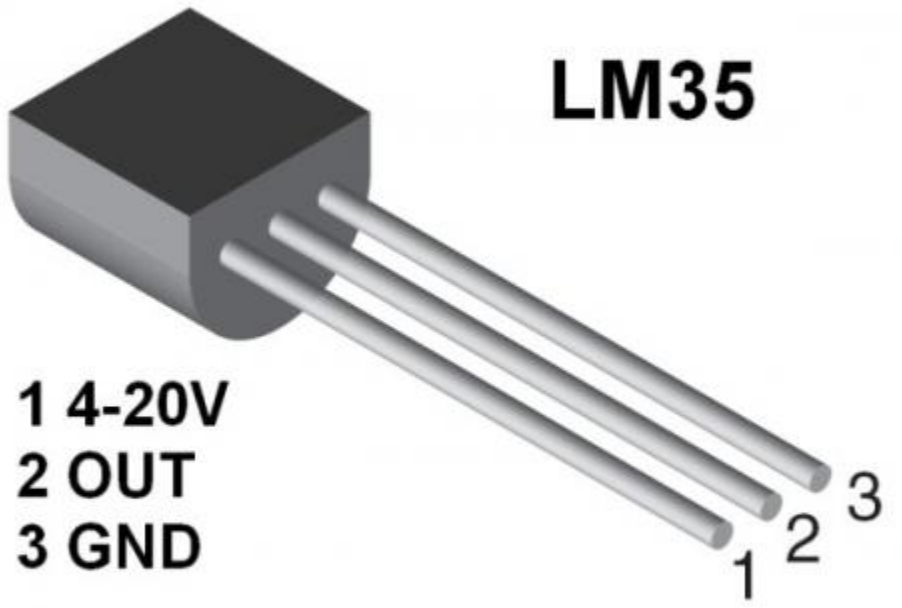
Program

Project

Digital Thermometer

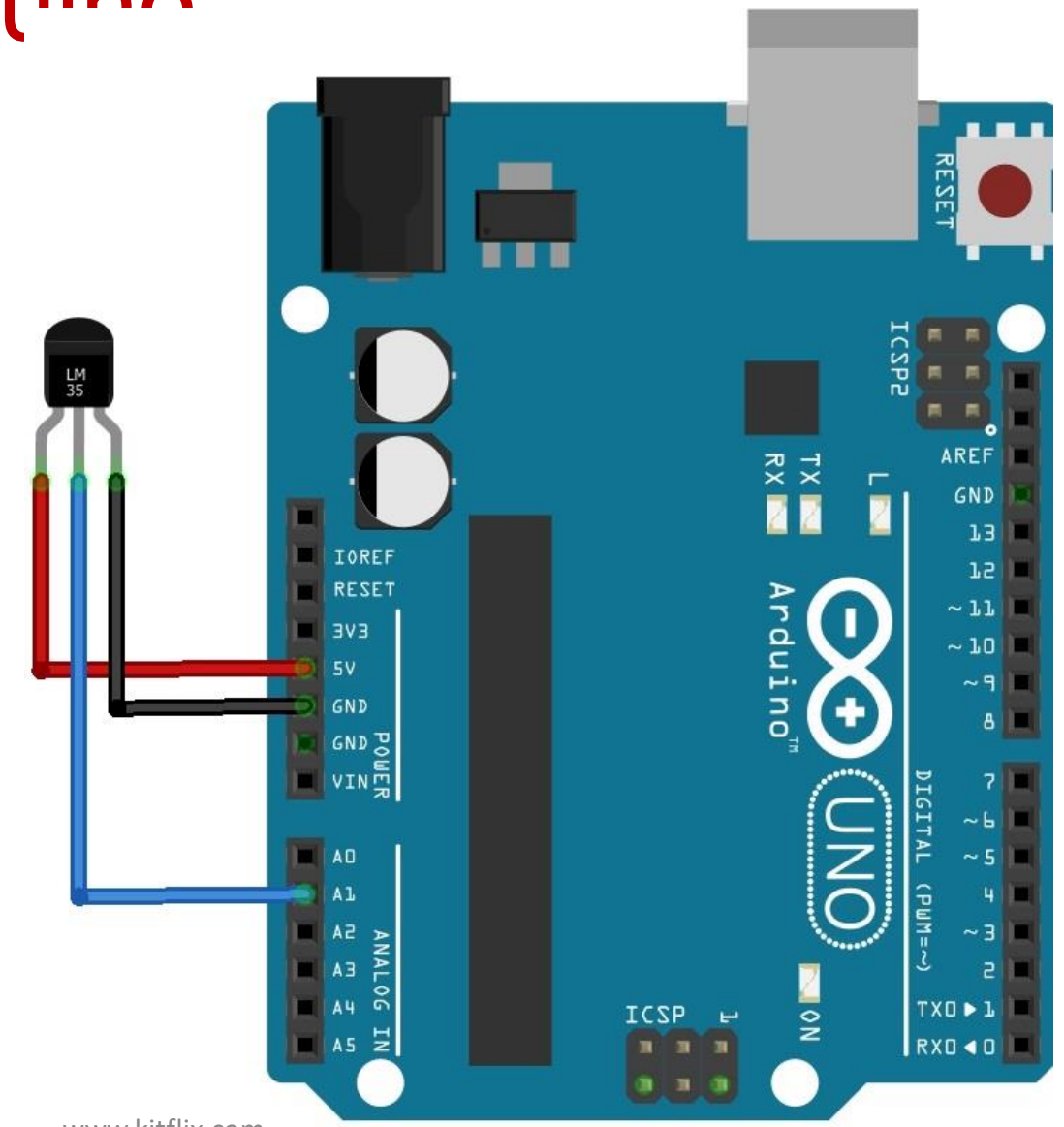
Temperature Sensor LM35

- Factory Calibrated Sensor
- 1 Deg C = 10mV
- For room temp of 25 degC, output will = 250mV



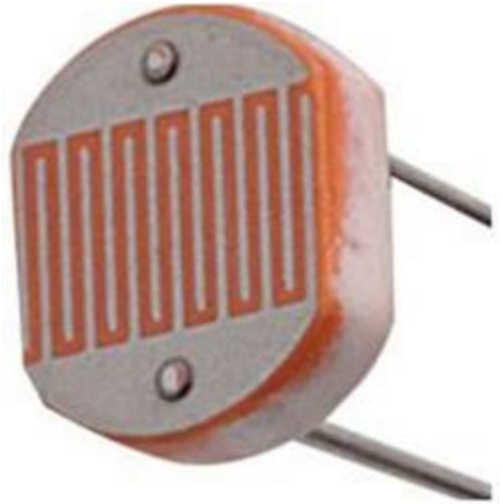
Interfacing with Arduino

- Analog Read
- Convert to mV
- Convert to temp
- Print on Serial



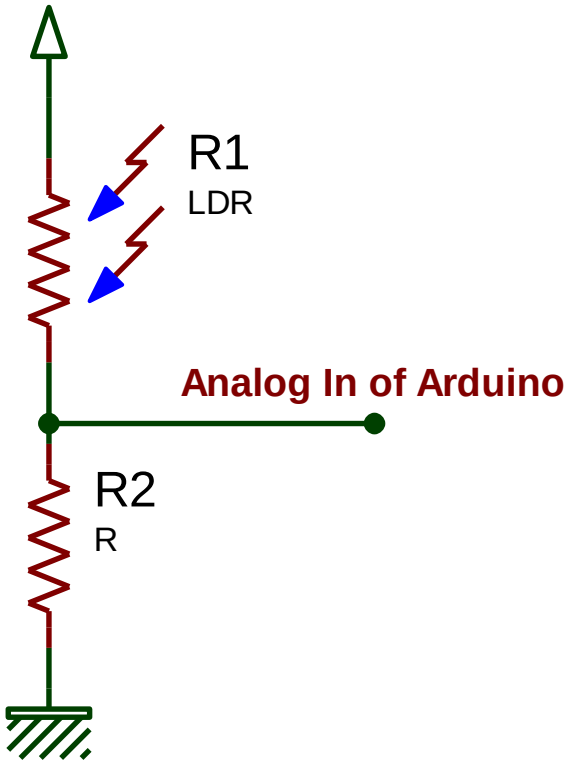
Light Dependent Resistor / Photoresistor

- Offers Changes in resistance based on light
- Need to be used with a circuit
- Create analog voltage
- Use ADC
- Display output
- Control Devices

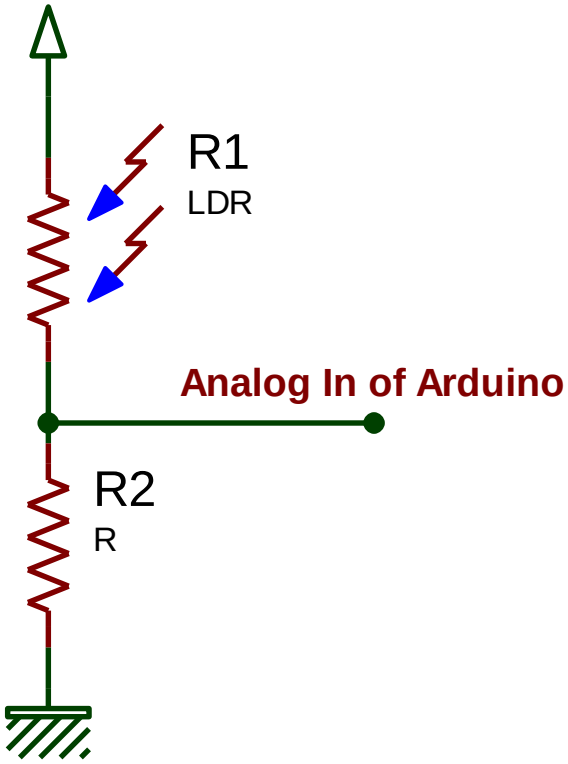


LDR with Arduino

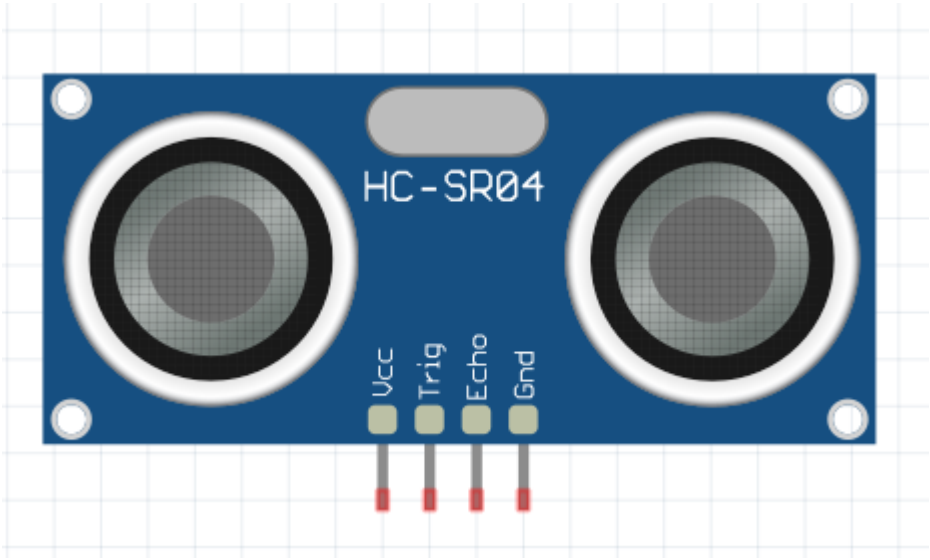
- When there is complete Darkness due to heavy resistance, the voltage will be very low
- When there is full light, due to very low resistance offered by LDR, the voltage to pin of Arduino will be high, near to VCC or 5v ideally
- Practically voltage will swing between 0-5v
- Reading this voltage will give an idea of amount of light



LDR Code



Ultrasonic sensor



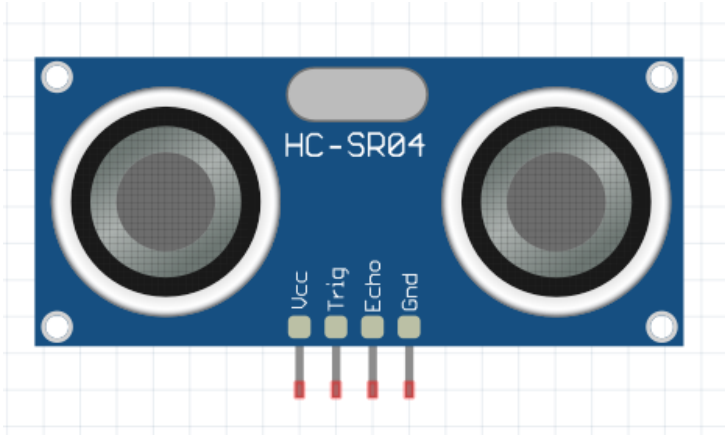
Applications

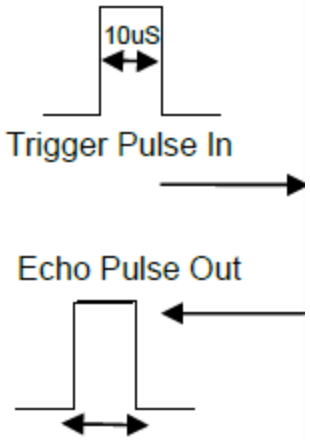
- Distance measurement
- Water level indicator
- Obstacle avoiding Robot
- Car parking
- Dimensions of object

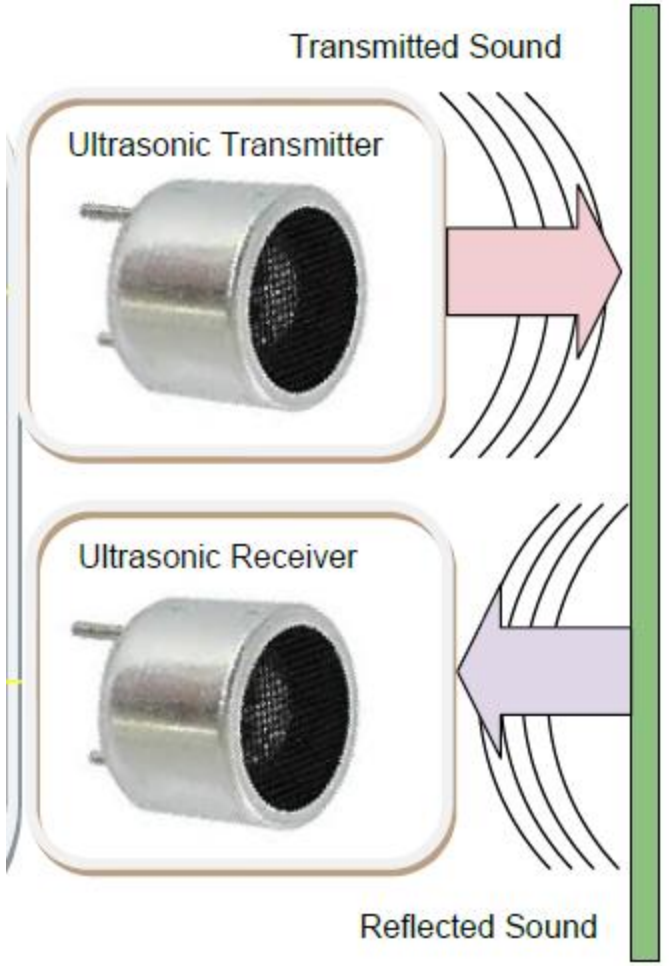
HC-SR 04

- Operating Voltage: 5V DC
- Operating Current: 15mA
- Measure Angle: 15°
- Ranging Distance: 2cm - 4m

Ultrasonic Sensor

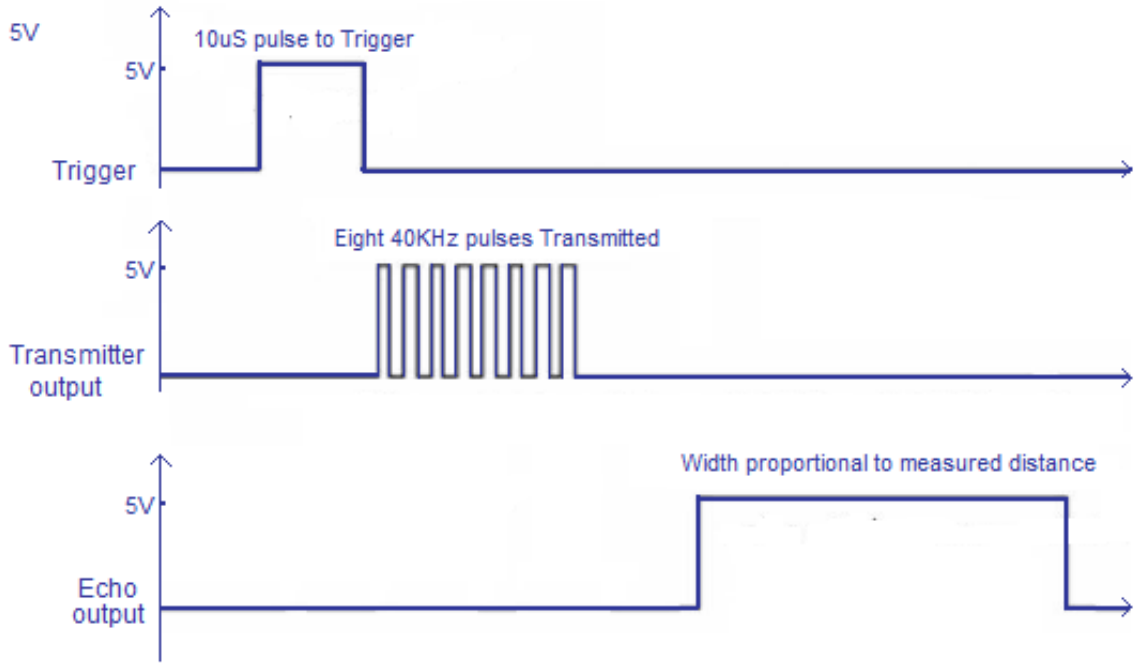






Distance Calculations

- Trig -→ 10 u second pulse
- Echo → HIGH
- Read echo pulse duration

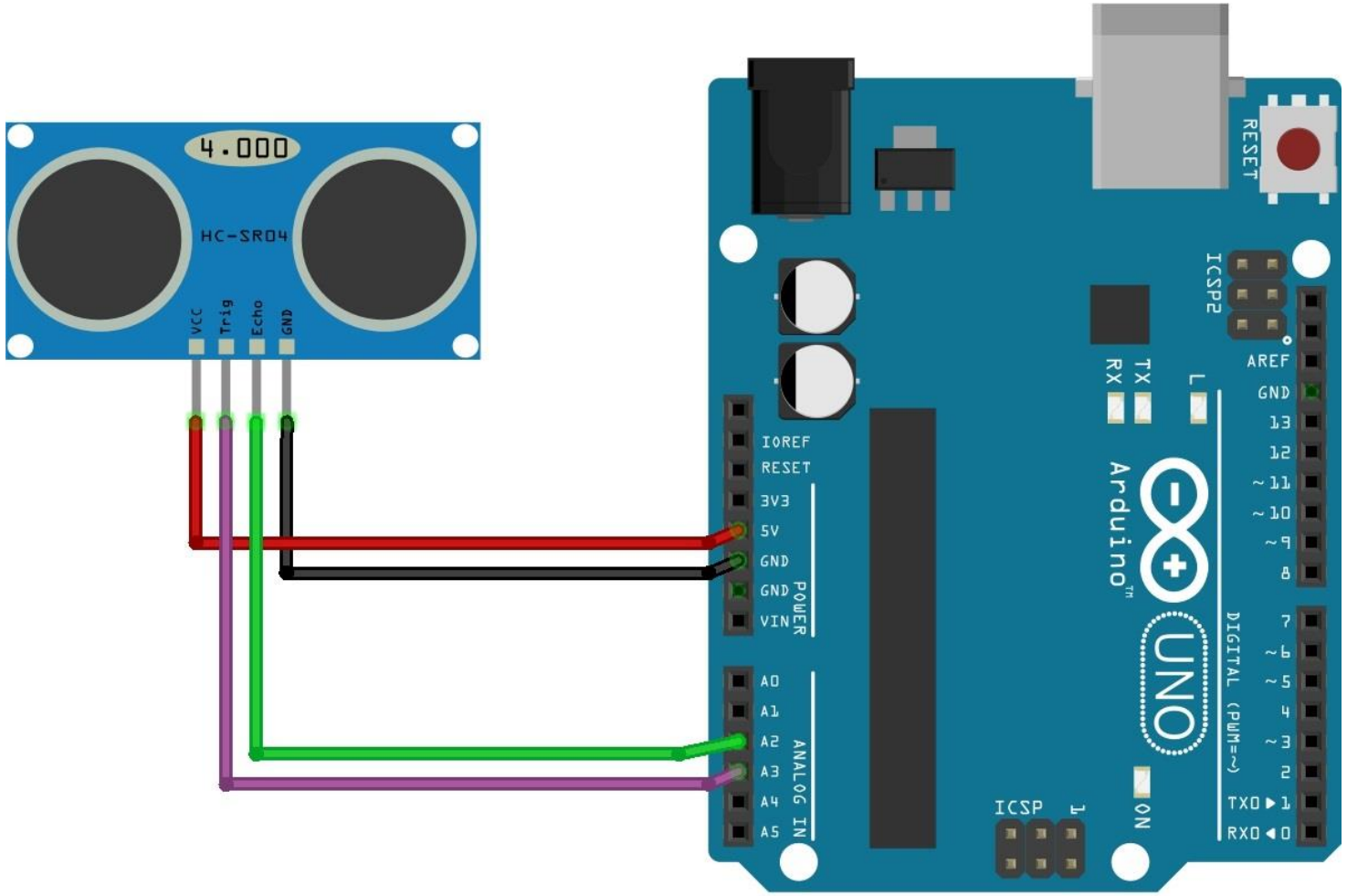


- Speed of Sound = 344 m/sec = 34400 cm / sec
- Time required for 1 cm =1/34400 = 29 micro second
- We have pulse duration in microSeconds
- Distance = (duration / 29) / 2
- Distance = duration / 58 cm

Formula

- Distance = (Speed of sound * Time delay) / 2
- Speed of sound at sea level = 343 m/s or 34300 cm/s
- Distance = 17150 * Time

Interfacing with Arduino



Algorithm

- Initialize serial port with 9600 bps
- Make trig pin output
- Make echo pin input
- Make trig pin zero
- Make trig pin high, wait 10 microseconds, make trig pin low
- Now monitor pulse length of echo pin (HIGH) duration
- Calculate the distance
- Print on Serial Port

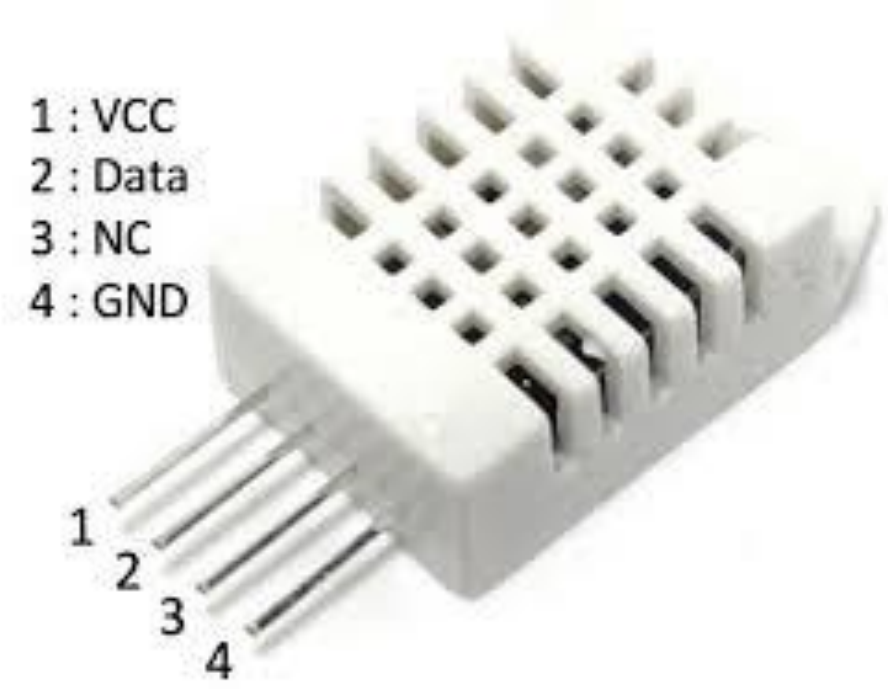
Projects

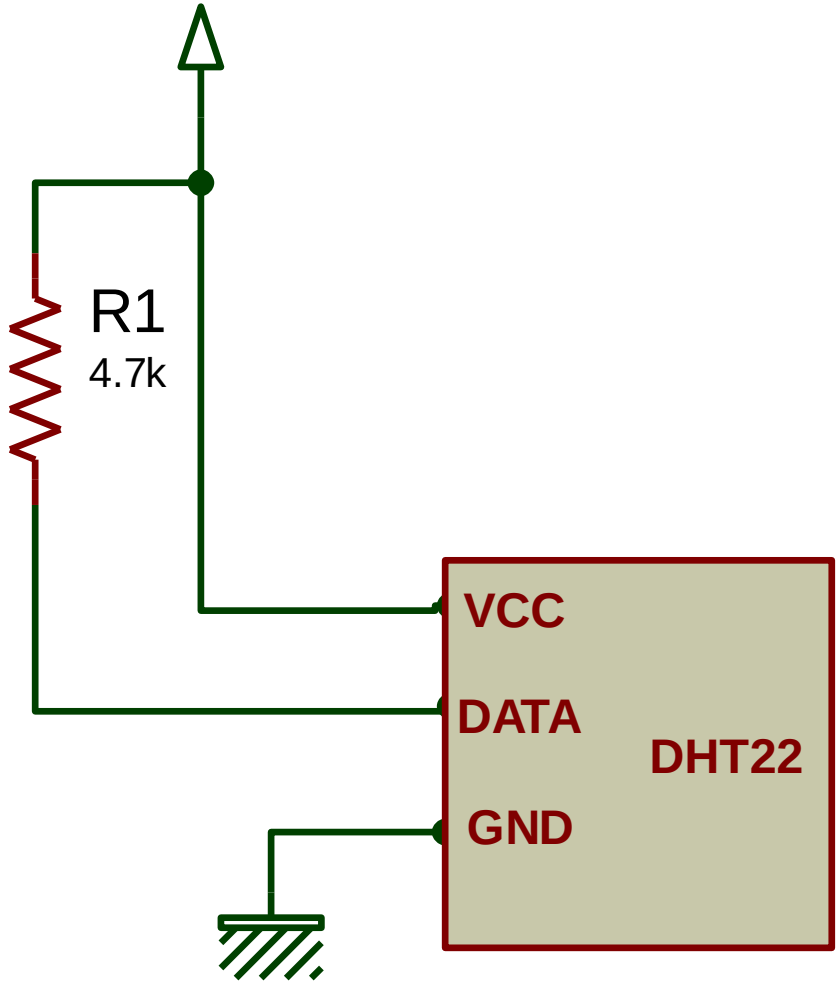
- Distance meter
- Water Level Indicator

Temperature Sensor DHT22

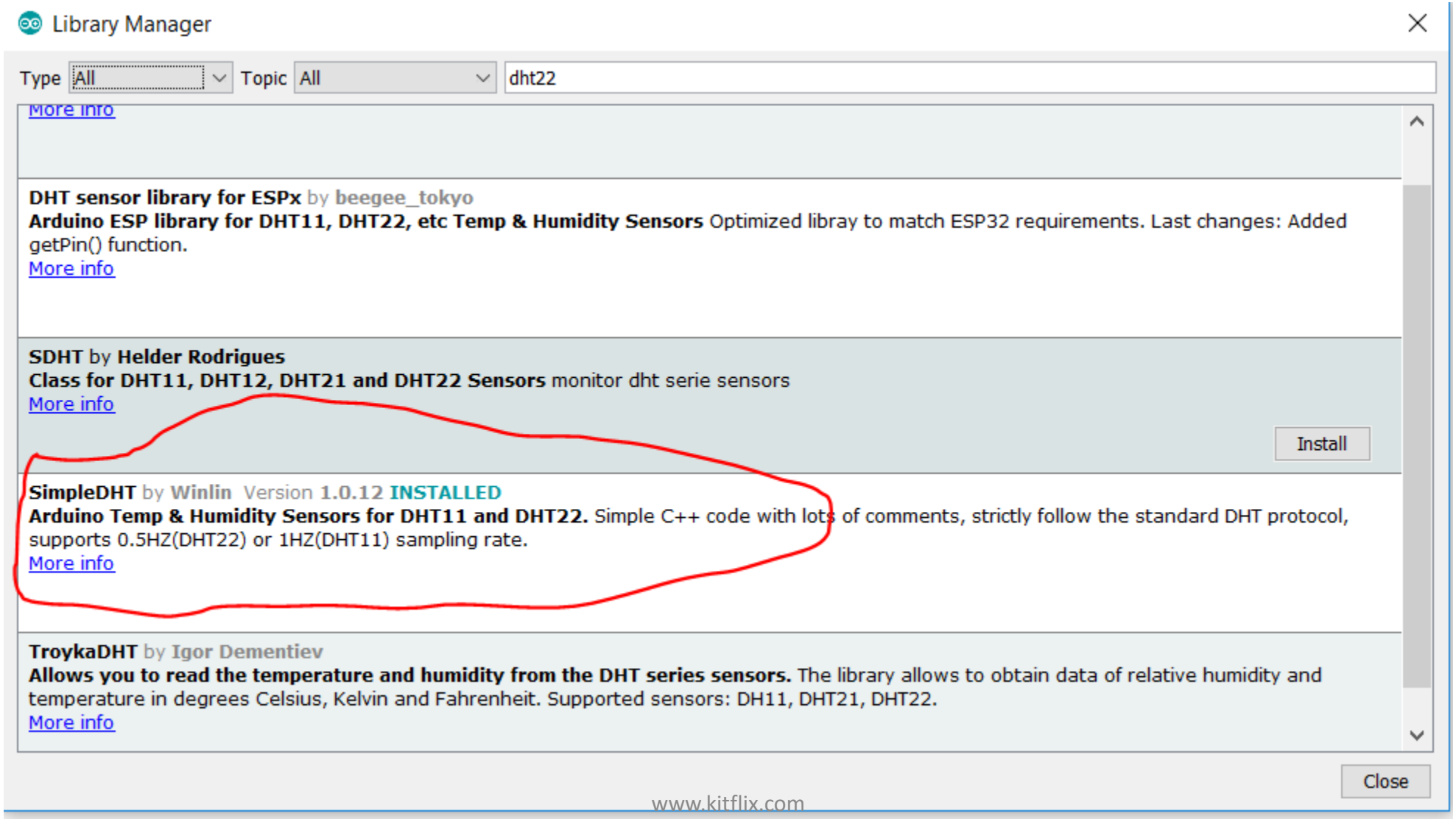
DHT22 Sensor (or DHT11)

- One Wire Digital Temperature sensor
- Requires library installation with Arduino
- Provides both temperature and humidity
- Only one pin of Arduino needed
- 3.3v to 6v
- -40 to 80 degree C




R1
4.7k
VCC
DATA
DHT22
GND

DHT22 Library



DHT11 Sensor Code

Switching Devices with Arduino

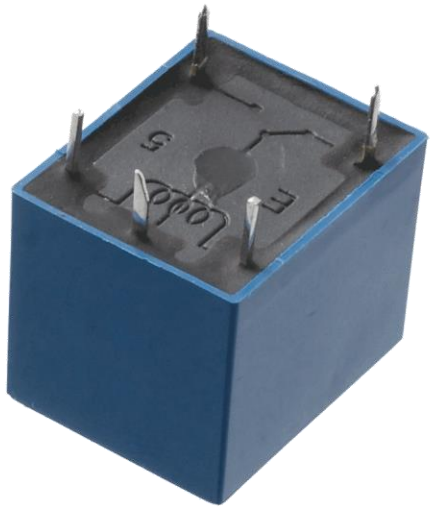
Basics of Relay

A relay is an electrically operated switch.

- Relays are used where it is necessary to control a circuit by a low-power signal (with complete electrical isolation between control and controlled circuits), or where several circuits must be controlled by one signal.

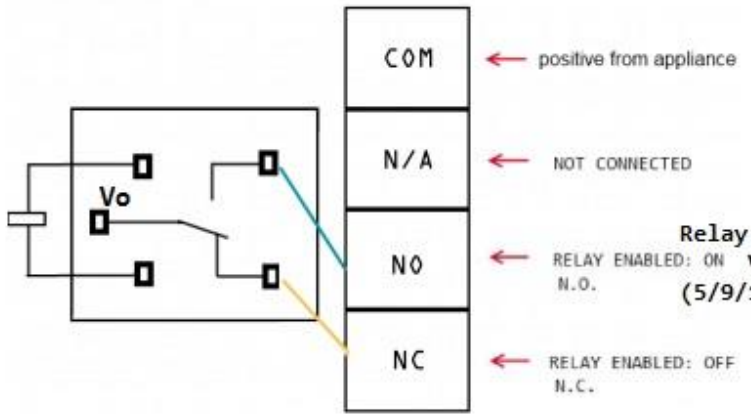
Relay

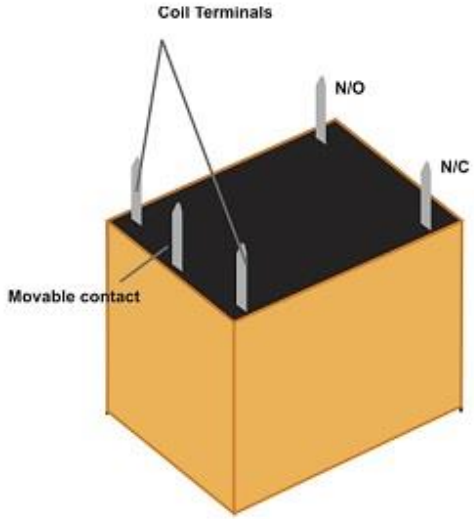

GOODSKY
RW-SS-124D
5A/250VAC
12A/120VAC
10A/120VAC
10A/24VDC
COIL:24VDC



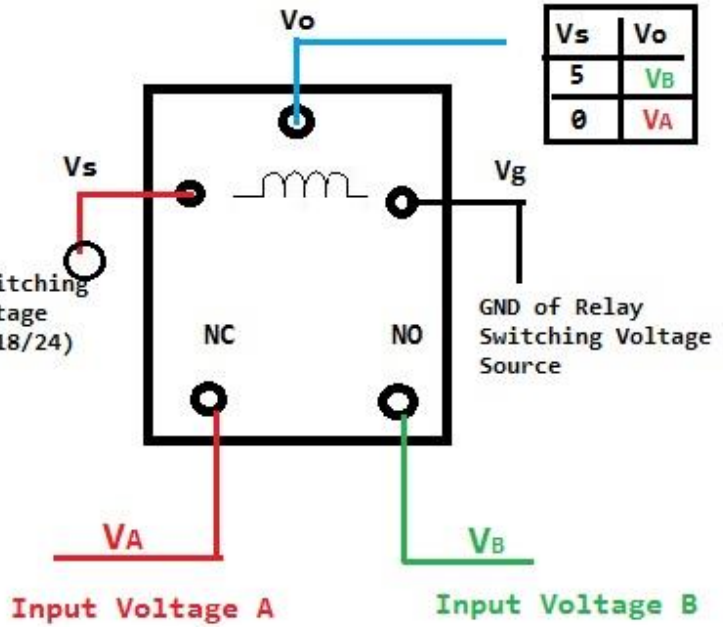

HFD2/012-0(555)
2A 30VDC
1A 125VAC

Relay Pin-Out



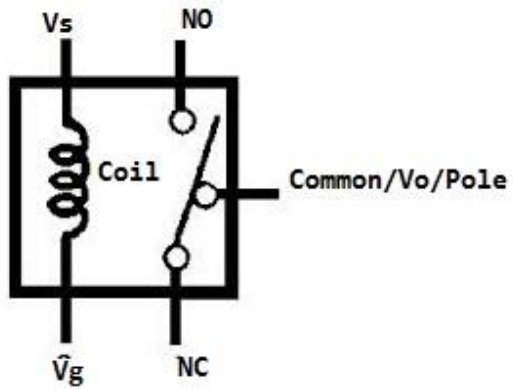


B) Physical Relay



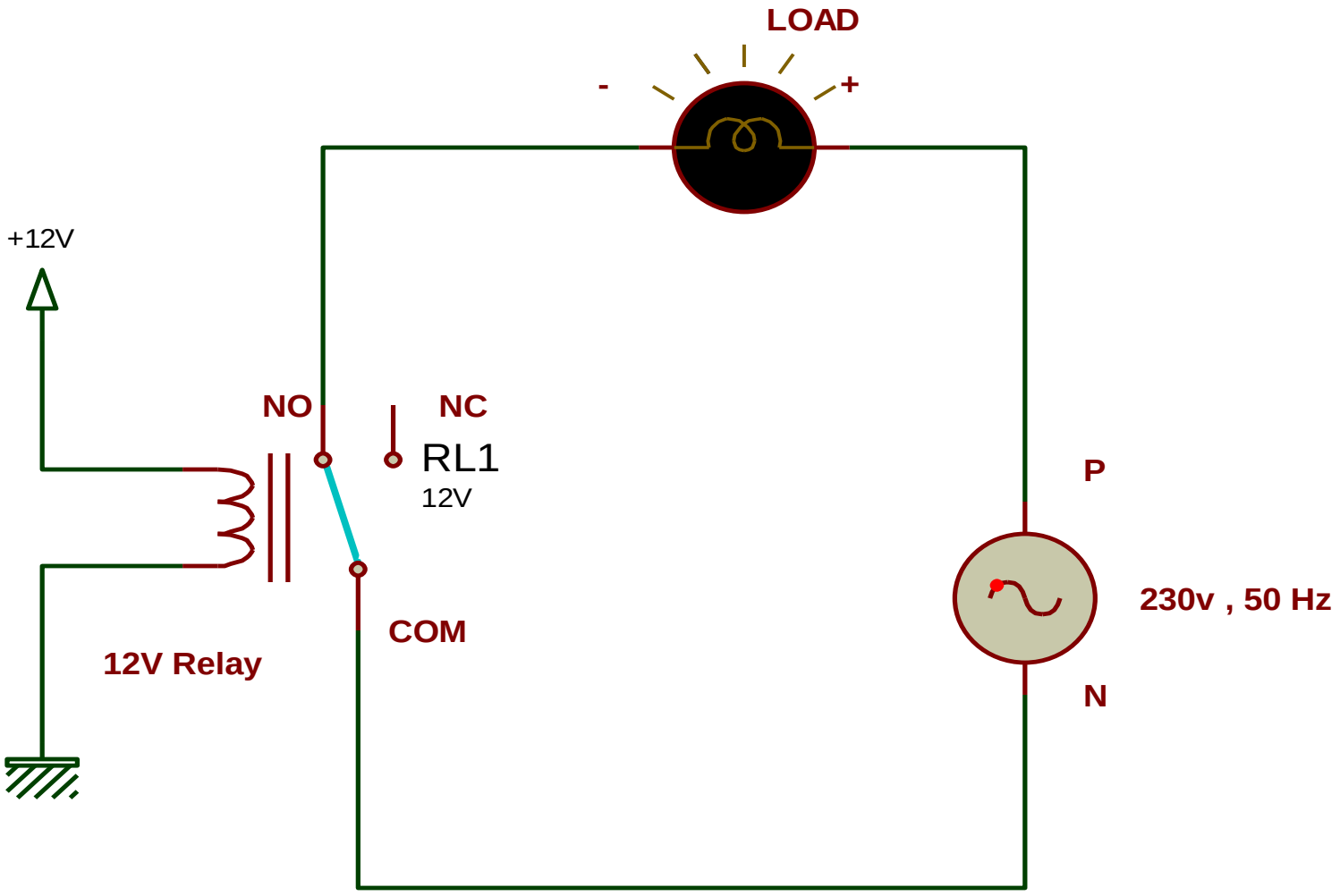
Vs	Vo
5	VB
0	VA

A) Relay Connection Principles



C) Internal Schematics

Load Connection



Basic Of Transistor

- Transistors are three terminal active devices made from different semiconductor materials that can act as either an insulator or a conductor by the application of a small signal voltage.

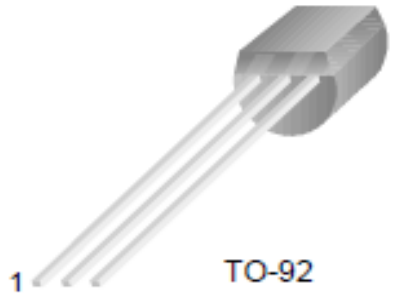
- The transistor’s ability to change between these two states enables it to have two basic functions: “switching” (digital electronics) or “amplification” (analogue electronics).

FAIRCHILD
SEMICONDUCTOR®

BC546/547/548/549/550

Switching and Applications

- High Voltage: BC546, V_{CEO}=65V
- Low Noise: BC549, BC550
- Complement to BC556 ... BC560



TO-92

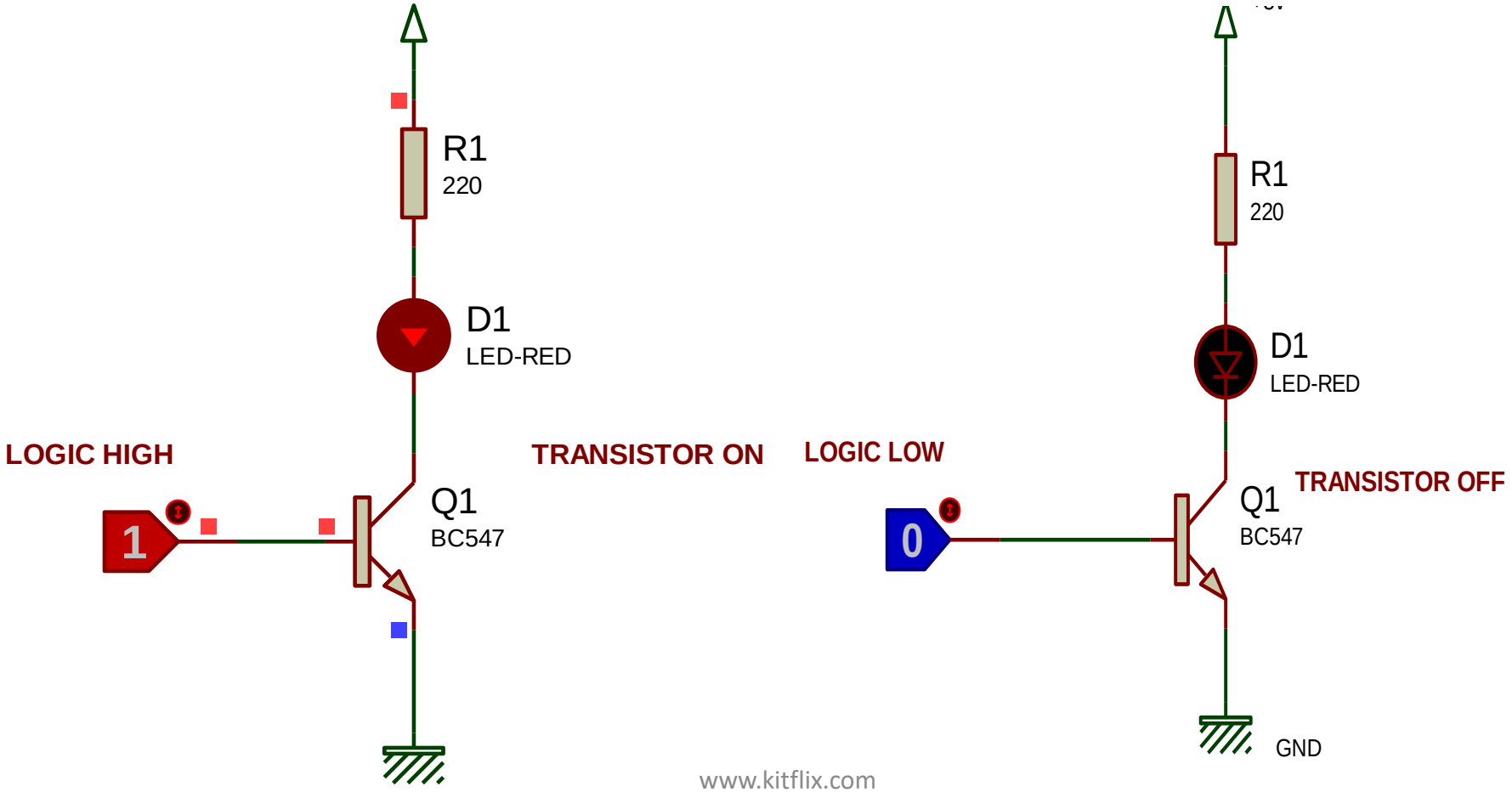
1. Collector 2. Base 3. Emitter

NPN Epitaxial Silicon Transistor

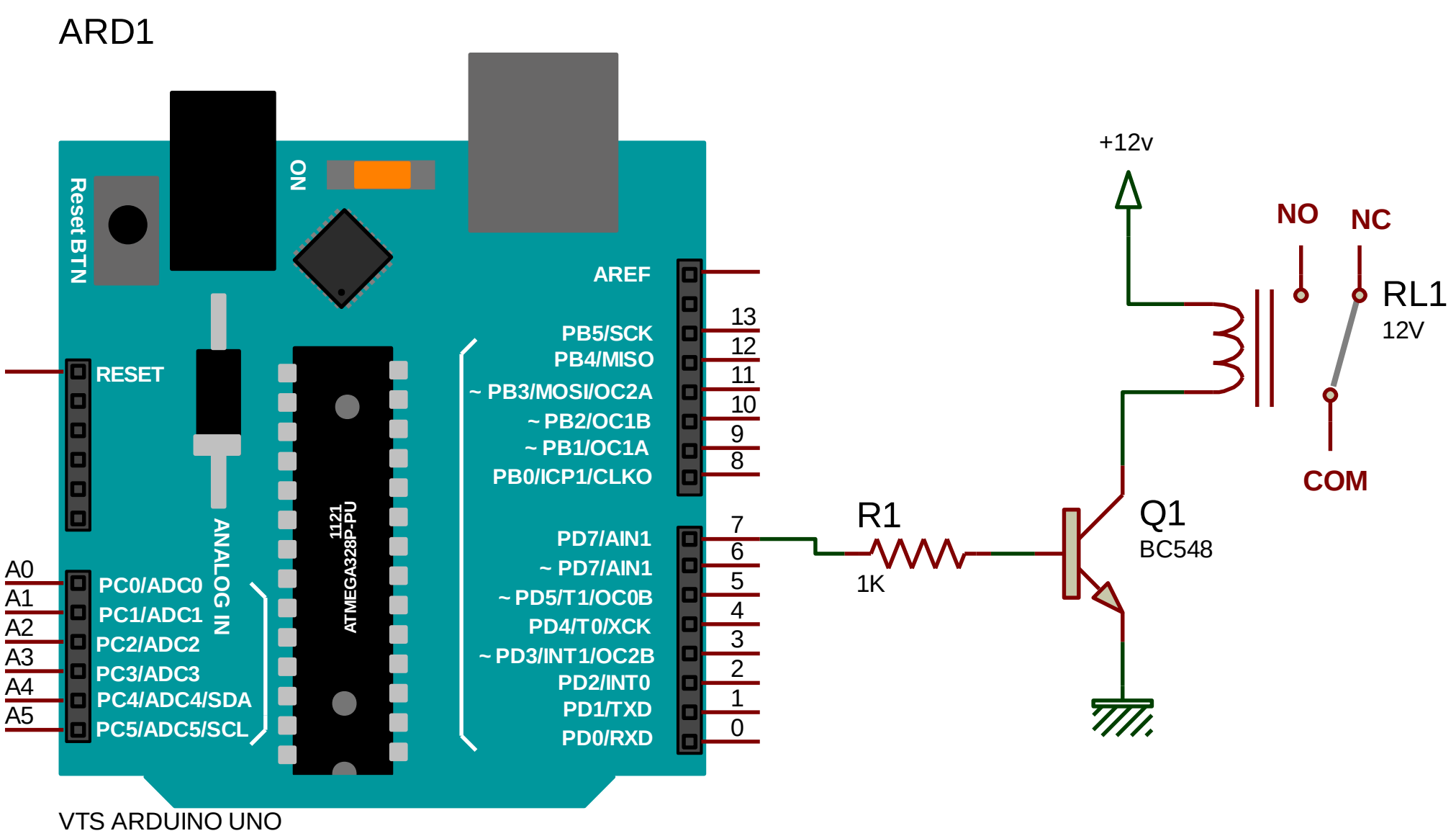
Absolute Maximum Ratings T_a=25°C unless otherwise noted

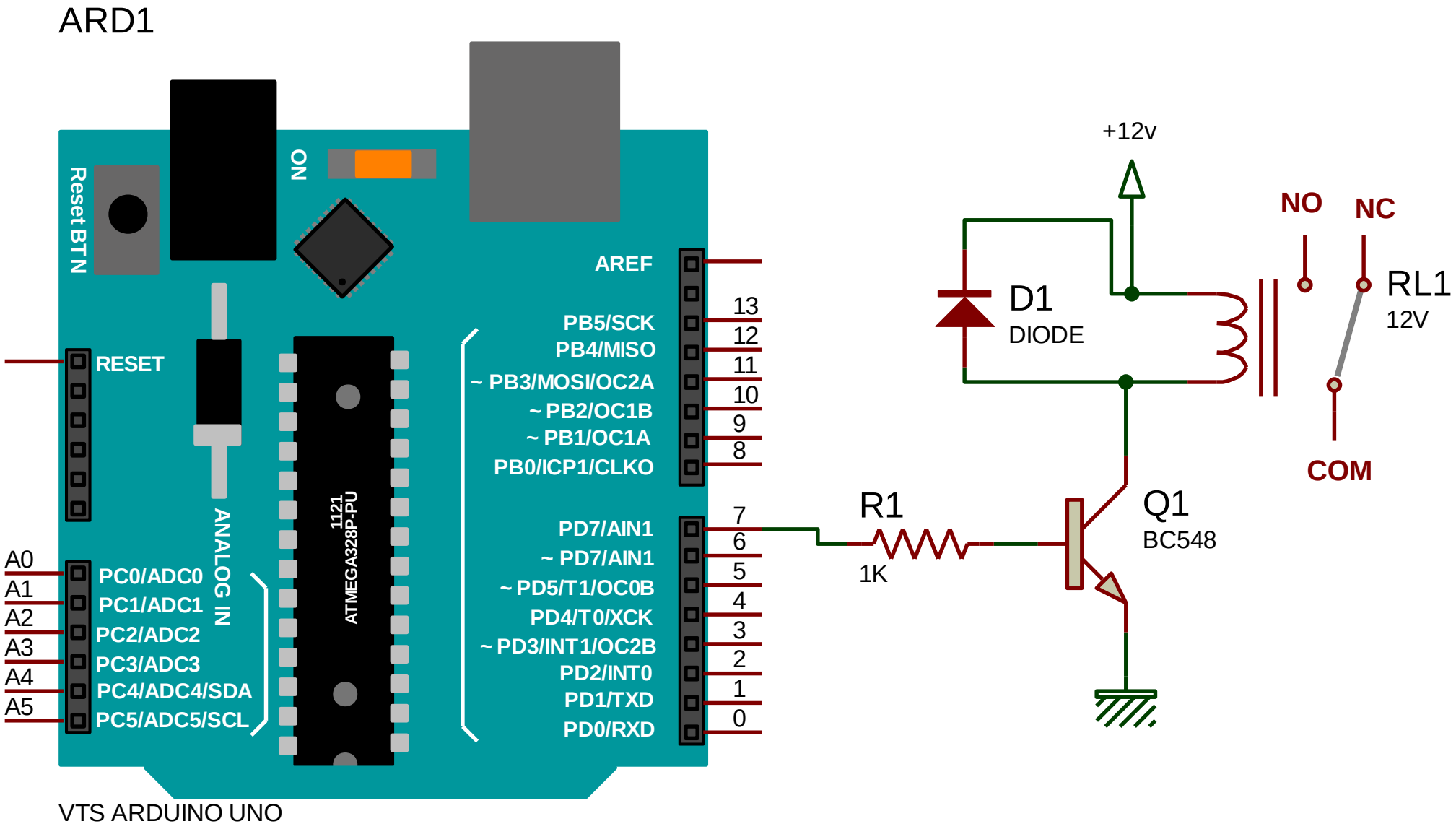
Symbol	Parameter	Value	Units
V_{CBO}	Collector-Base Voltage : BC546	80	V
	: BC547/550	50	V
	: BC548/549	30	V
V_{CEO}	Collector-Emitter Voltage : BC546	65	V
	: BC547/550	45	V
	: BC548/549	30	V
V_{EBO}	Emitter-Base Voltage : BC546/547	6	V
	: BC548/549/550	5	V
I_C	Collector Current (DC)	100	mA
P_C	Collector Power Dissipation	500	mW
T_J	Junction Temperature	150	°C
T_{STG}	Storage Temperature	-65 ~ 150	°C

Transistor Working



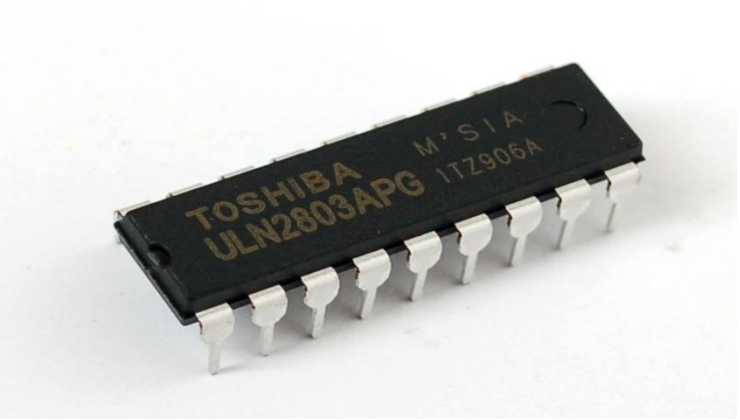
Transistorized Relay




ARD1
Reset BTN
ON
RESET
ANALOG IN
A0
A1
A2
A3
A4
A5
PC0/ADC0
PC1/ADC1
PC2/ADC2
PC3/ADC3
PC4/ADC4/SDA
PC5/ADC5/SCL
1121
ATMEGA328P-PU
AREF
PB5/SCK
PB4/MISO
~ PB3/MOSI/OC2A
~ PB2/OC1B
~ PB1/OC1A
PB0/ICP1/CLKO
13
12
11
10
9
8
PD7/AIN1
~ PD7/AIN1
~ PD5/T1/OC0B
PD4/T0/XCK
~ PD3/INT1/OC2B
PD2/INT0
PD1/TXD
PD0/RXD
7
6
5
4
3
2
1
0
R1
1K
+12v
D1
DIODE
Q1
BC548
NO
NC
RL1
12V
COM
VTS ARDUINO UNO

ULN2803

- A ULN2803 is an Integrated Circuit (IC) chip with a High Voltage/High Current Darlington Transistor Array.
- It allows you to interface TTL signals with higher voltage/current loads.
- On the output side the ULN2803 is generally rated at 50V/500mA, so it can operate small loads directly.
- Used to interface Relay and stepper motors with microcontrollers



TOSHIBA BIPOLAR DIGITAL INTEGRATED CIRCUIT SILICON MONOLITHIC

ULN2803AP,ULN2803AFW,ULN2804AP,ULN2804AFW
(Manufactured by Toshiba Malaysia)

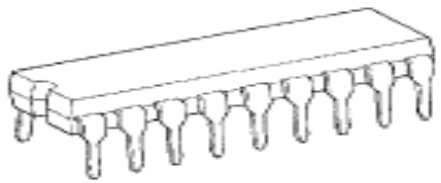
8CH DARLINGTON SINK DRIVER

The ULN2803AP / AFW Series are high−voltage, high−current darlington drivers comprised of eight NPN darlington pairs.
All units feature integral clamp diodes for switching inductive loads.
Applications include relay, hammer, lamp and display (LED) drivers.

FEATURES

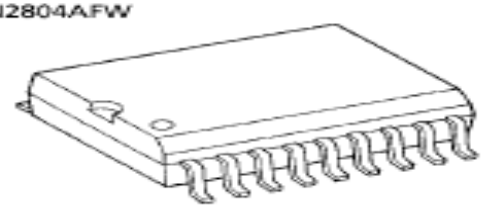
- Output current (single output)
  500 mA (Max.)
- High sustaining voltage output
  50 V (Min.)
- Output clamp diodes
- Inputs compatible with various types of logic.
- Package Type−AP : DIP−18pin
- Package Type−AFW : SOL−18pin

ULN2803AP
ULN2804AP



DIP18-P-300-2.54F

ULN2803AFW
ULN2804AFW

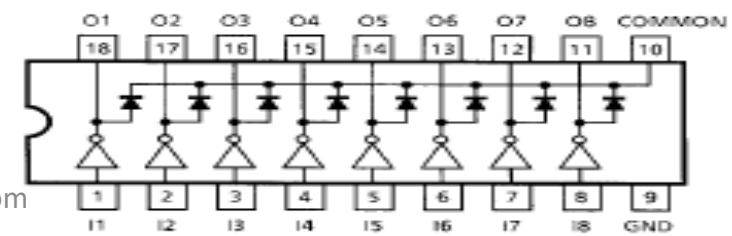


SOL18-P-300-1.27

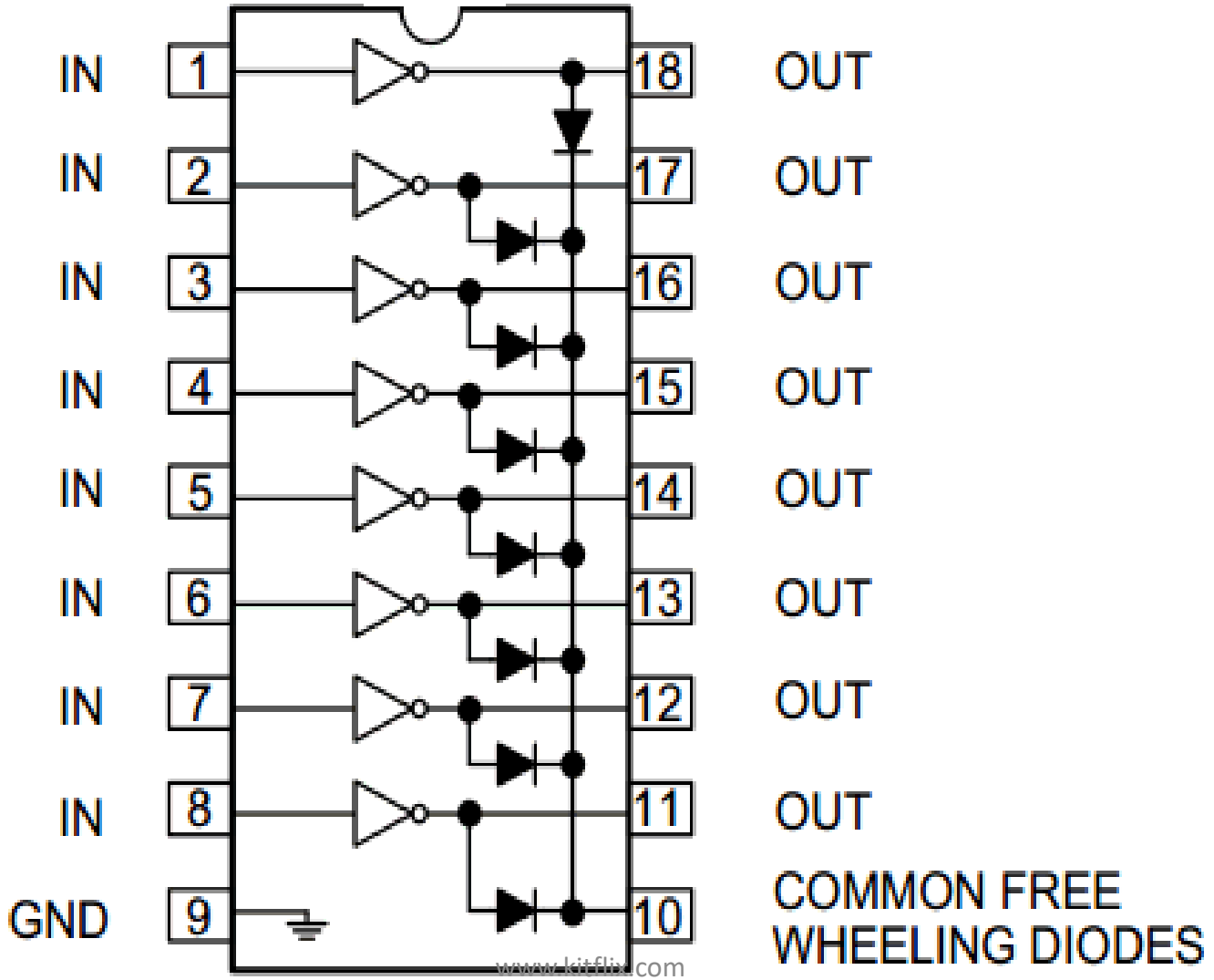
Weight
DIP18−P−300−2.54F: 1.478 g (Typ.)
SOL18−P−300−1.27 : 0.48 g (Typ.)

TYPE	INPUT BASE RESISTOR	DESIGNATION
ULN2803AP / AFW	2.7 kΩ	TTL, 5 V CMOS
ULN2804AP / AFW	10.5 kΩ	6~15 V PMOS, CMOS

PIN CONNECTION (TOP VIEW)



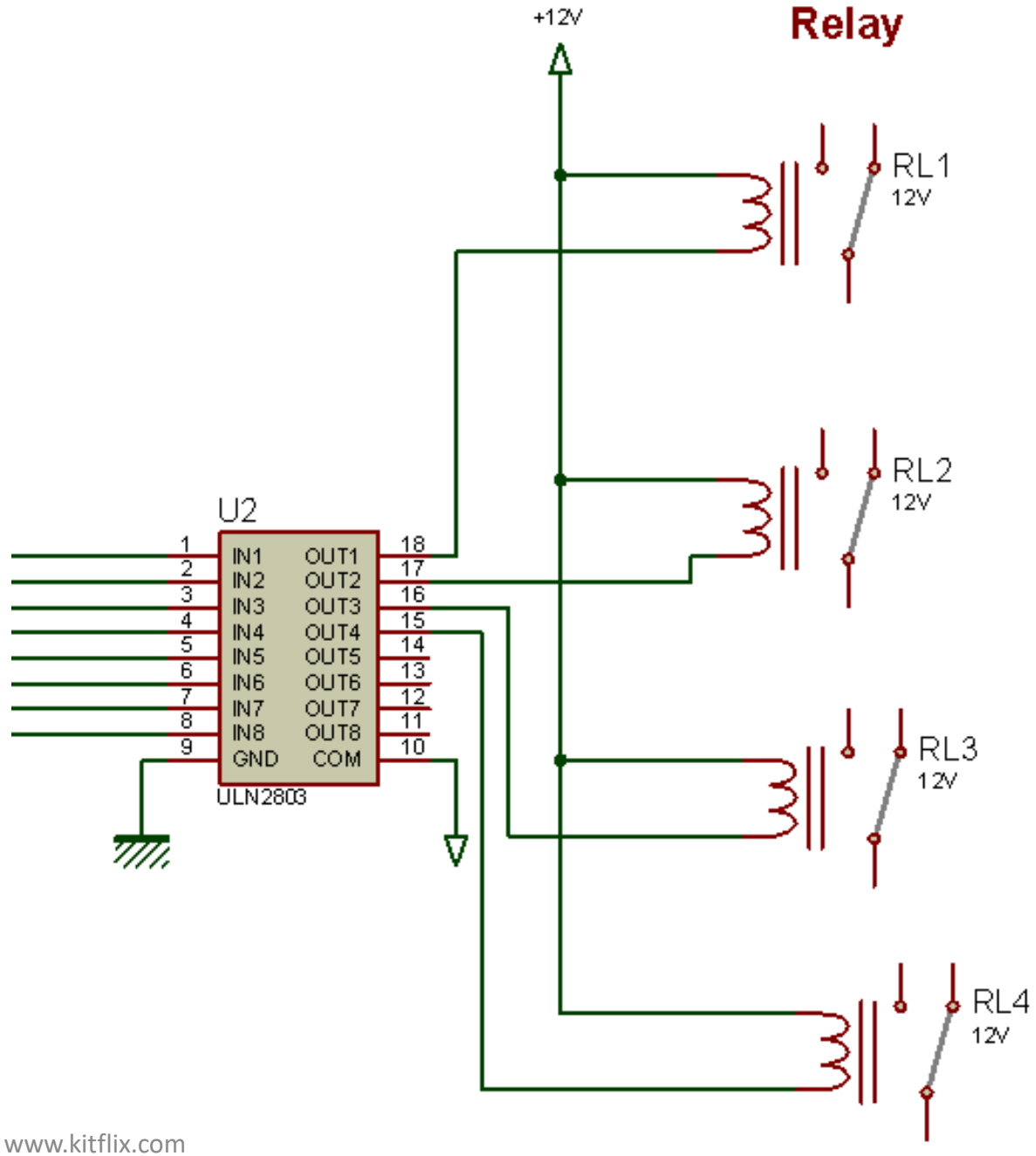
ULN2803 Pin-Out



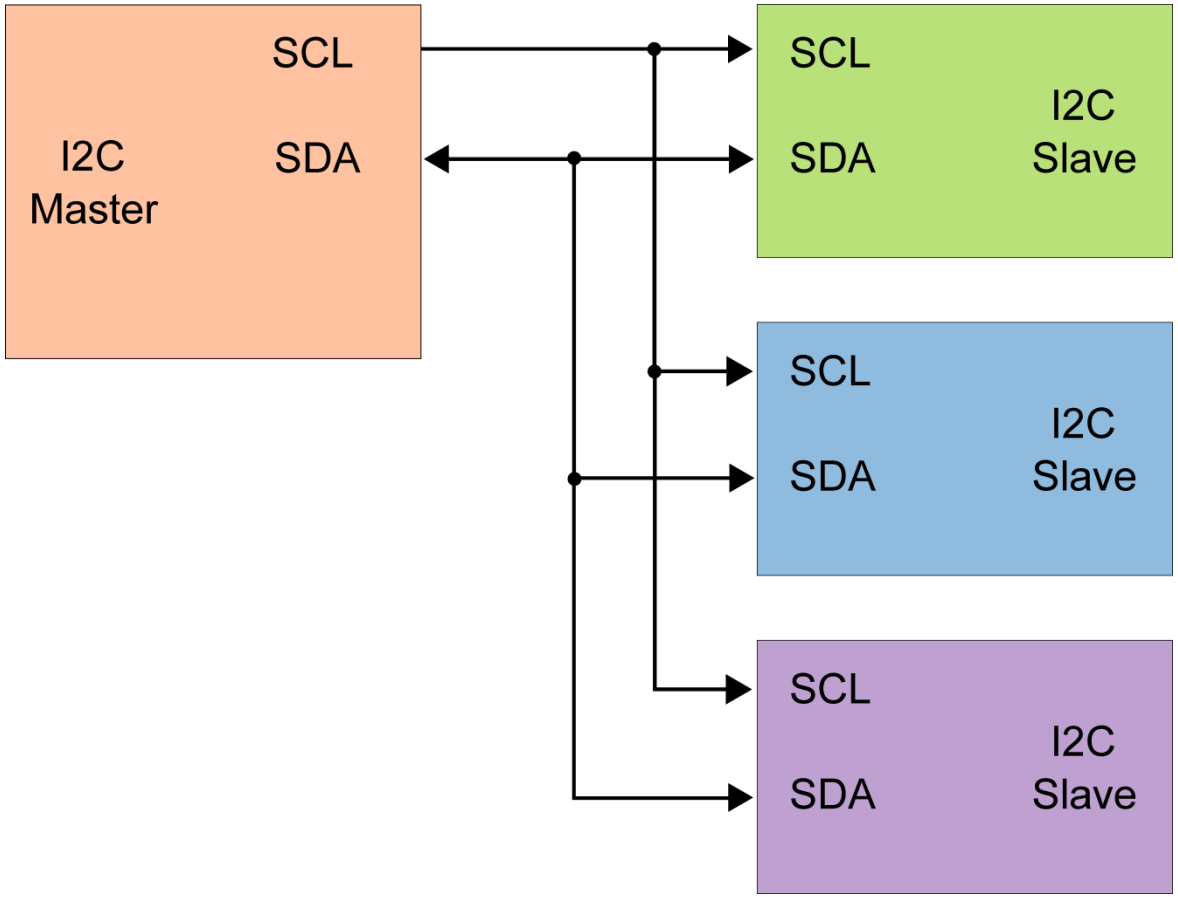
Multiple Relay Driving Using ULN2803

Interfacing Diagram

Arduino Pins



I2C Wiring


SCL
I2C
Master
SDA
SCL
SDA
I2C
Slave
SCL
SDA
I2C
Slave
SCL
SDA
I2C
Slave

I2C / TWI

- Serial Communication Protocol
- Bus type communication (one transmits, multiple receives)
- All the devices in the bus have a specific 7-bit address
- Requires 2 wires (SDA & SCL)
- SDA = Serial Data Line
- SCL = Serial Clock Line

Hardware TWI Pins on Arduino

Board	I2C / TWI pins
Uno, Ethernet	A4 (SDA), A5 (SCL)
Mega2560	20 (SDA), 21 (SCL)
Leonardo	2 (SDA), 3 (SCL)
Due	20 (SDA), 21 (SCL), SDA1, SCL1

Wire Library

- begin()
- requestFrom()
- beginTransmission()
- endTransmission()
- write()
- available()
- read()
- SetClock()
- onReceive()
- onRequest()

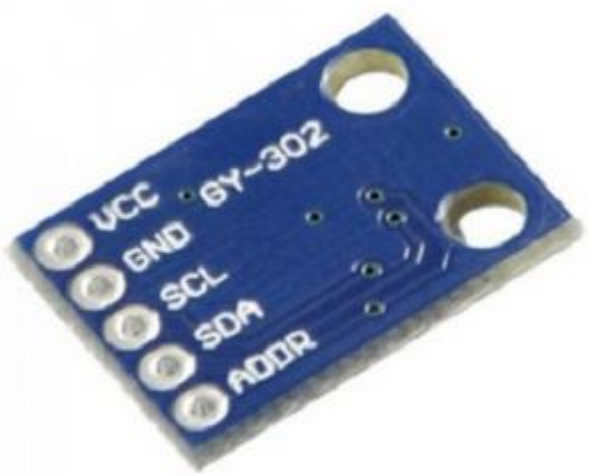
Example 1 – I2C communication

Example of a generic I2C communication

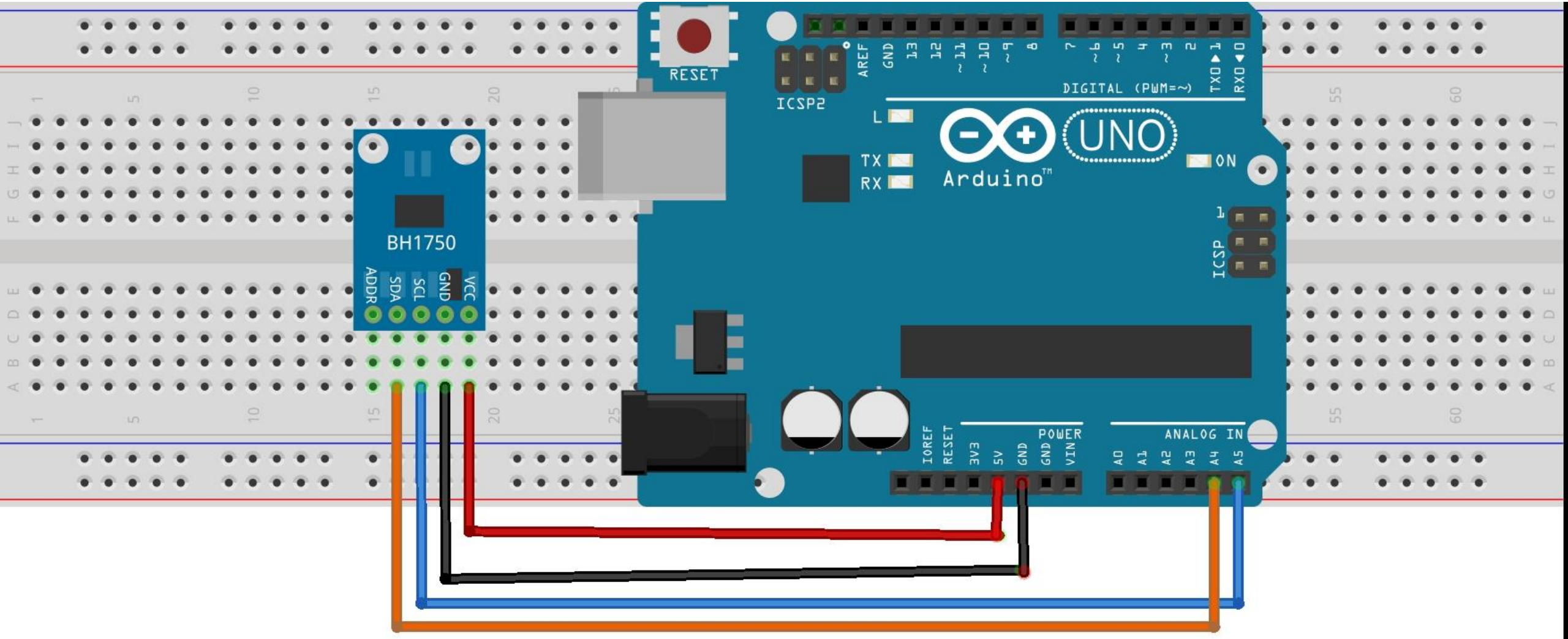
```
#define SENSOR_I2CADD 0x45 //Address for the sensor. Different for each device connected.
#include <Wire.h> //I2C library

char dataByte;

void setup(){
  Wire.begin();
  Serial.begin(9600);
}

void loop(){
  Wire.beginTransmission(SENSOR_I2CADD); //Begin transmission to address 0x45
  Wire.write(0x22); //Register address within the sensor where the data is to be read from
  Wire.endTransmission();

  Wire.requestFrom(SENSOR_I2CADD, 1); //Get a byte(1) from the register address 0x22
  if(Wire.available()) //If the buffer has data
  {
    dataByte = Wire.read(); //Save the data to a variable
  }
  Serial.println(dataByte); //print the received byte to Serial
  delay(100);
}
```

BH1750

- Digital Light Sensor
- Gives output in lumens (lux)
- I2C
- +3.3 v Operation
- Module operates on +5v
- Connect to Arduino with SDA and SCL



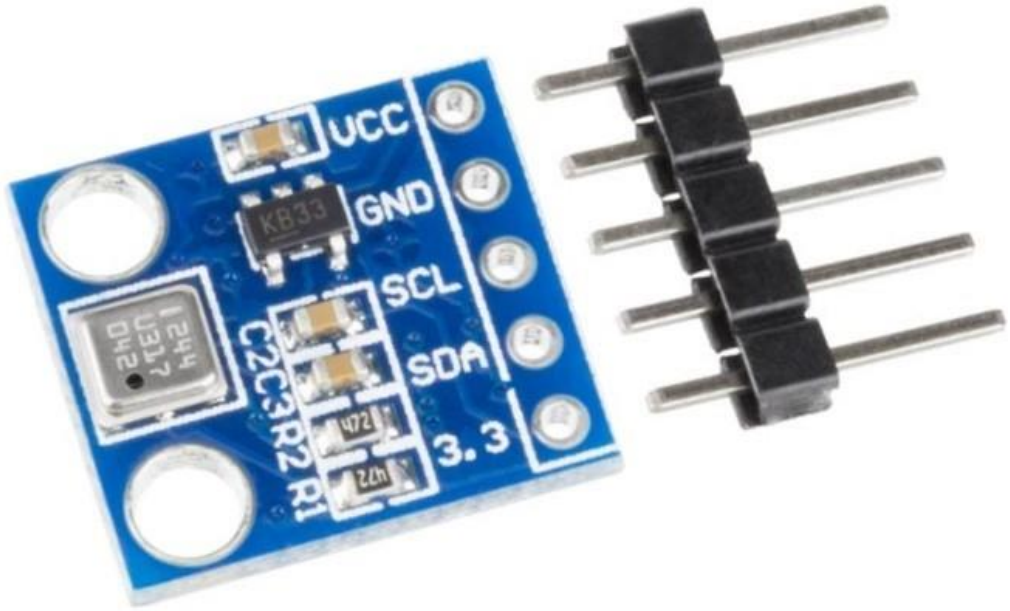
Arduino interfacing with BH1750



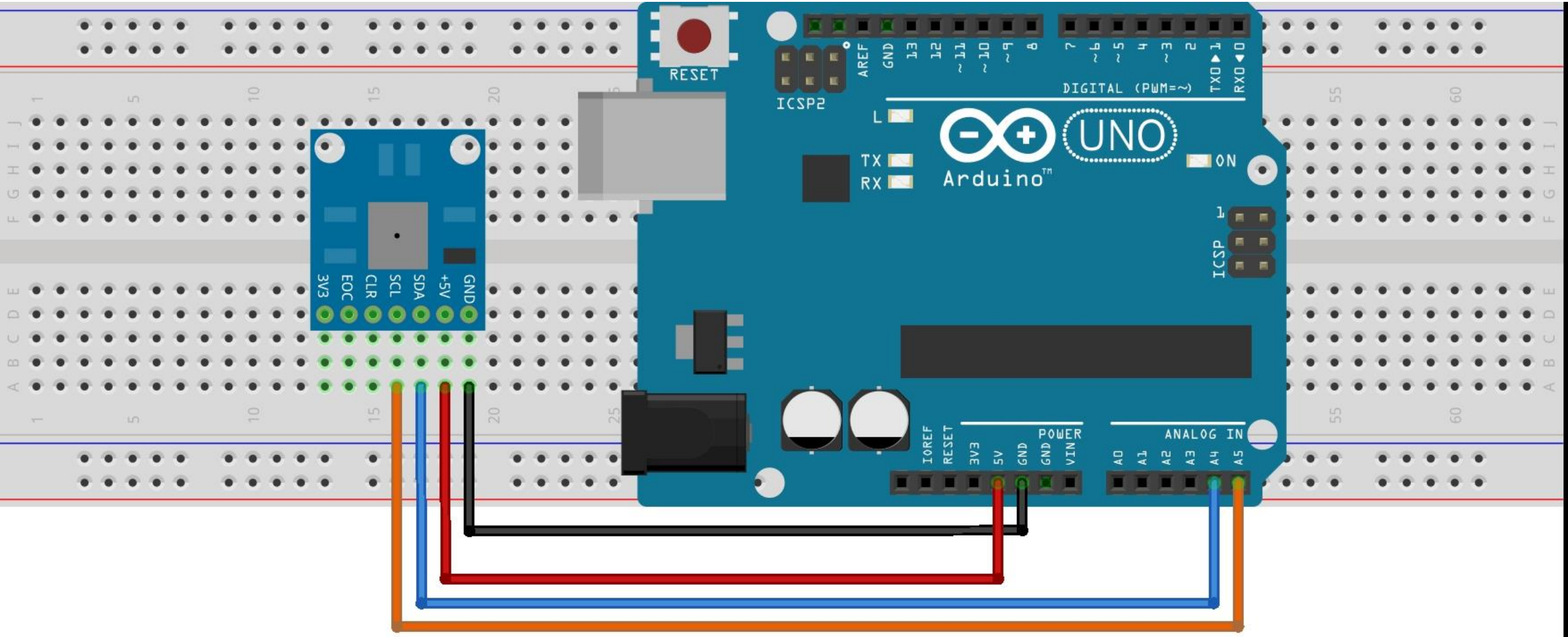
Code

BMP180

- Barometric pressure/temperature/altitude sensor by bosch
- I2C Compatible
- Logic: 3 to 5V compliant
- This board/chip uses I2C 7-bit address 0x77.



Arduino Interfacing BMP180



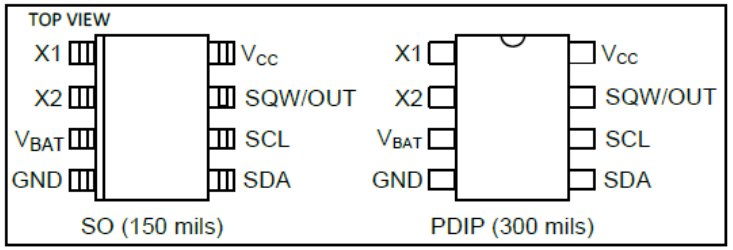
Real Time Clock

I2C based RTC Chip Interface with Arduino

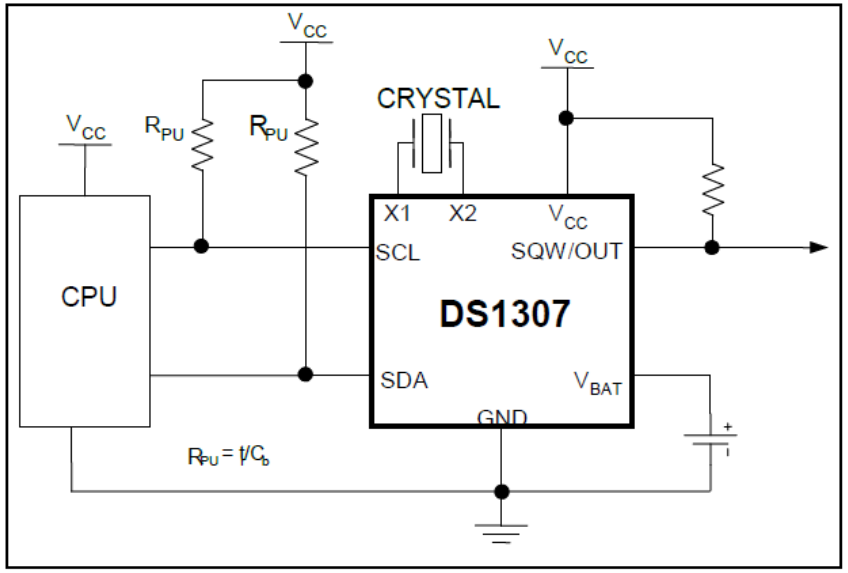
RTC with Arduino

- Real Time Clock
- I2C Serial interface
- Can keep Date, month, year, day of week, hour, minute, seconds with leap year compensation
- +5V operation
- Requires external Battery
- 24 hour / 12 hour mode
- 32.768KHz

PIN CONFIGURATIONS



TYPICAL OPERATING CIRCUIT



Applications

- Time keeping for Arduino
- Any kind of timer based project, school bell timer
- Data logger to keep track of time of data capturing
- Attendance system
- Any application requiring knowledge of precise time

DS1307 Internal Registers (BCD format)

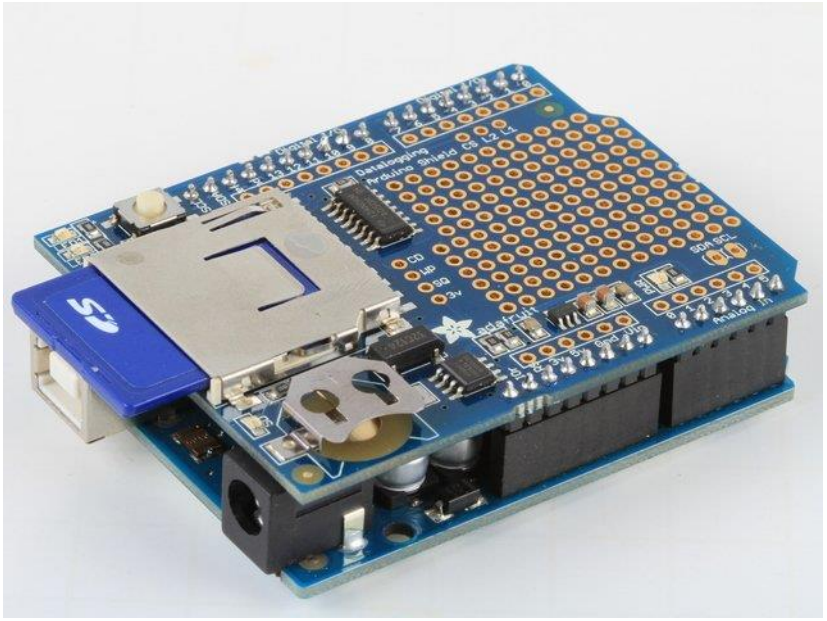
Table 2. Timekeeper Registers

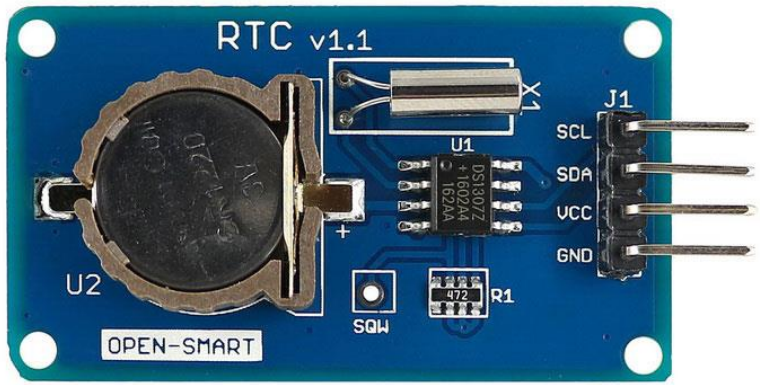
ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE
00h	CH	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12	10 Hour	10 Hour	Hours				Hours	1–12 +AM/PM 00–23
		24	PM/ AM							
03h	0	0	0	0	0	DAY			Day	01–07
04h	0	0	10 Date		Date				Date	01–31
05h	0	0	0	10 Month	Month				Month	01–12
06h	10 Year				Year				Year	00–99
07h	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08h–3Fh									RAM 56 x 8	00h–FFh

0 = Always reads back as 0.

DS1307

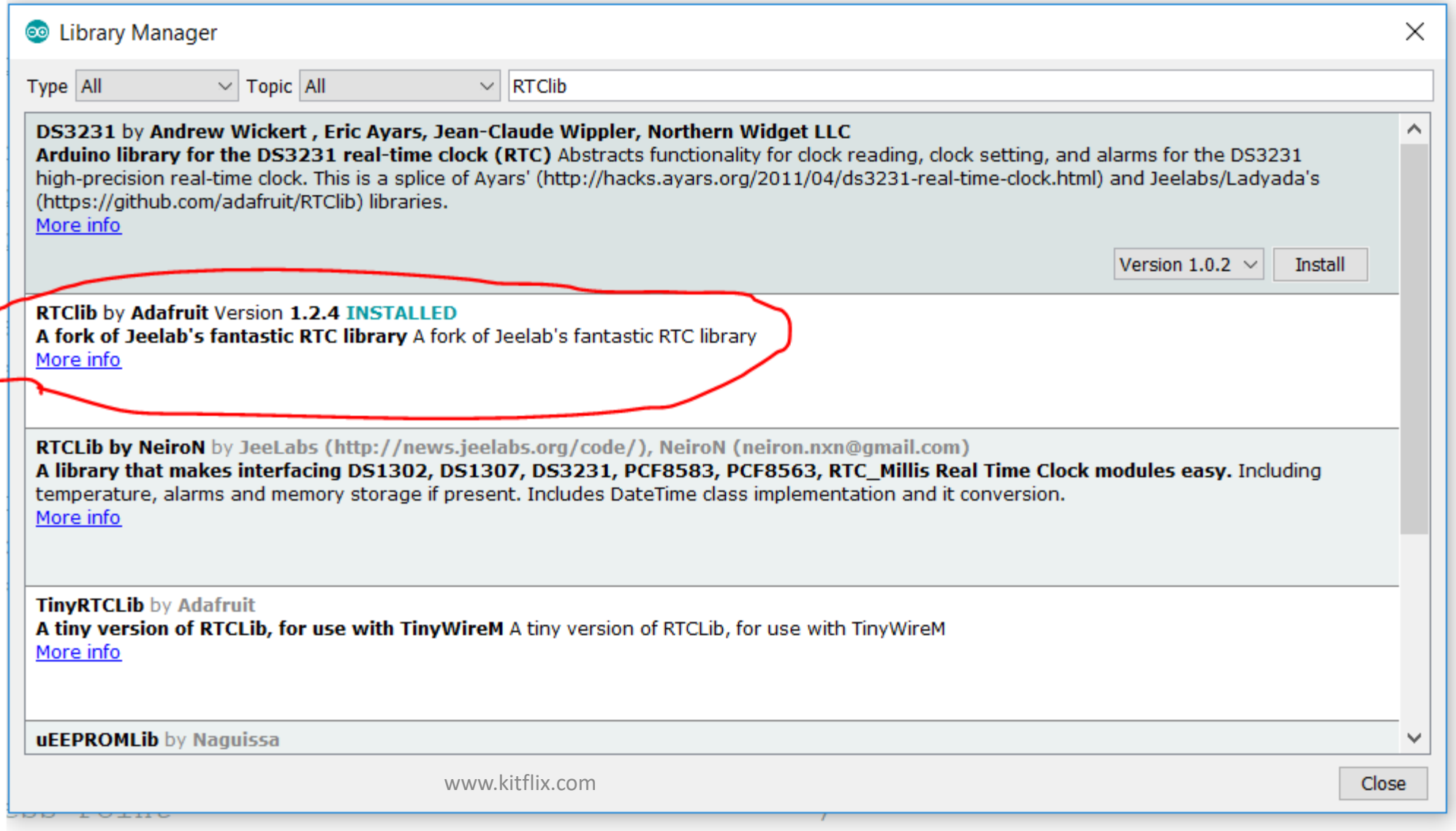
- Requires external crystal 32.768khz
- Requires external 3v battery for time keeping
- I2C address 0xD0 (write) 0xD1 (Read)
- Module / shield available
- https://github.com/adafruit/RTClib
- SCL →)A5 (SCL)
- SDA →A4 (SDA





Install Library

Adafruit's RTCLib



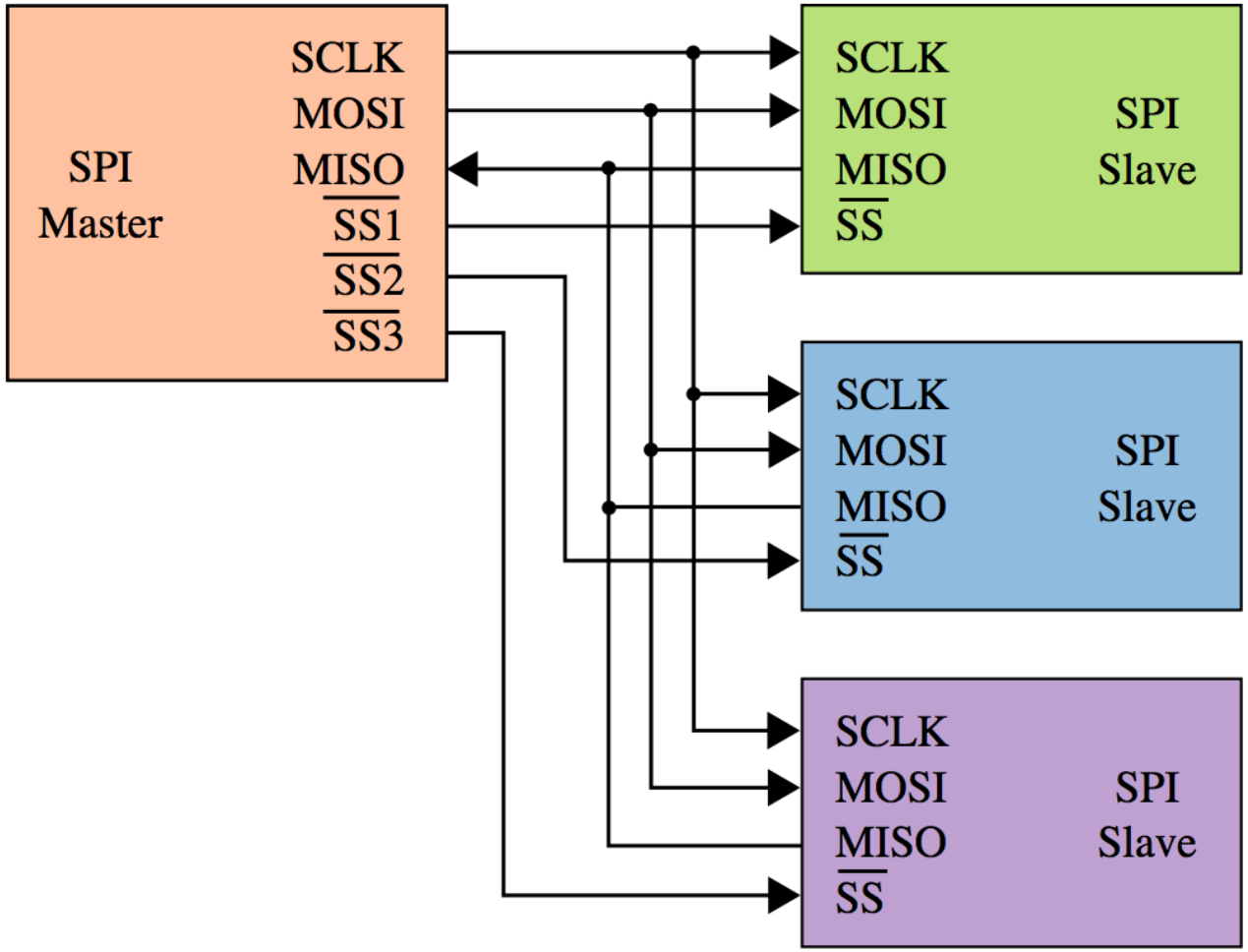
Interfacing RTC with Arduino

- #include "RTClib.h“
- RTC_DS1307 rtc;
- rtc.begin() // returns ZERO if no RTC detected
- rtc.isrunning() // returns ZERo if no RTC detected
- rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));
- DateTime now = rtc.now();
- now.
  - year()
  - month()
  - day()
  - dayoftheweek ()
  - hour ()
  - minute ()
  - second ()

Memory Card Interface

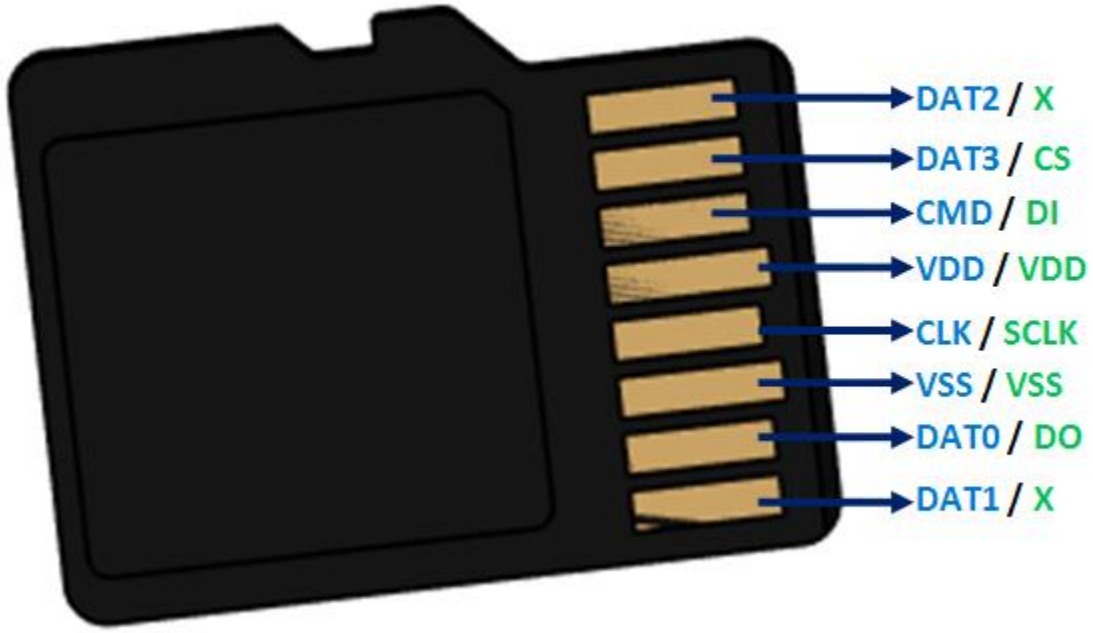
SPI Based microSD Card Interface with Arduino

SPI (Serial Peripheral Interface)



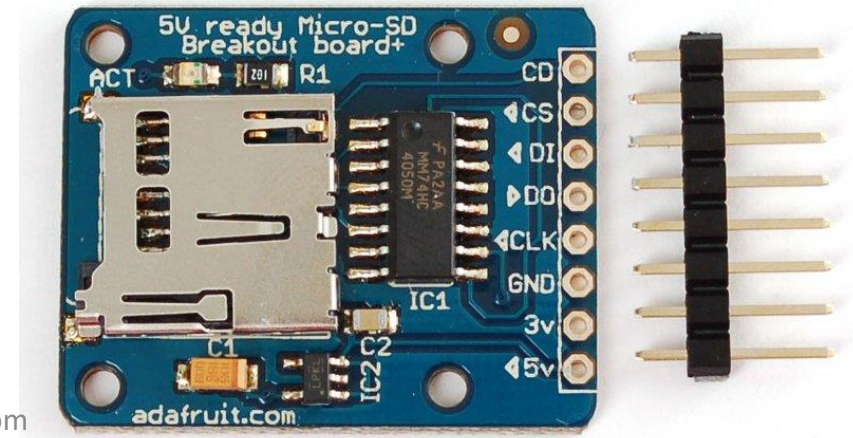
SD Card Module with Arduino

- Permanent Storage Device for Arduino
- Works on 3.3 volt
- Requires voltage level converter
- Works on SPI Protocol



Recording Data to a File

- The Arduino UNO has no data recording capability
- You can output to a serial monitor and capture that as a file
- The answer – add a MicroSD Card Breakout board!
- MicroSD Card provides GBs of storage for files
- Supports extended deployments



Some Notes on the MicroSD Cards

- They are strictly 3.3 volt devices!
- SD cards are raw storage, but can be formatted with a file system
- The SD cards in your kit should work with Arduino, Windows, and Mac (but some devices require a specific file system)
- Don't format unless you use the "official" formatter from: https://www.sdcard.org/downloads/formatter_3/
- FAT32 is a good option for the file system with Arduino

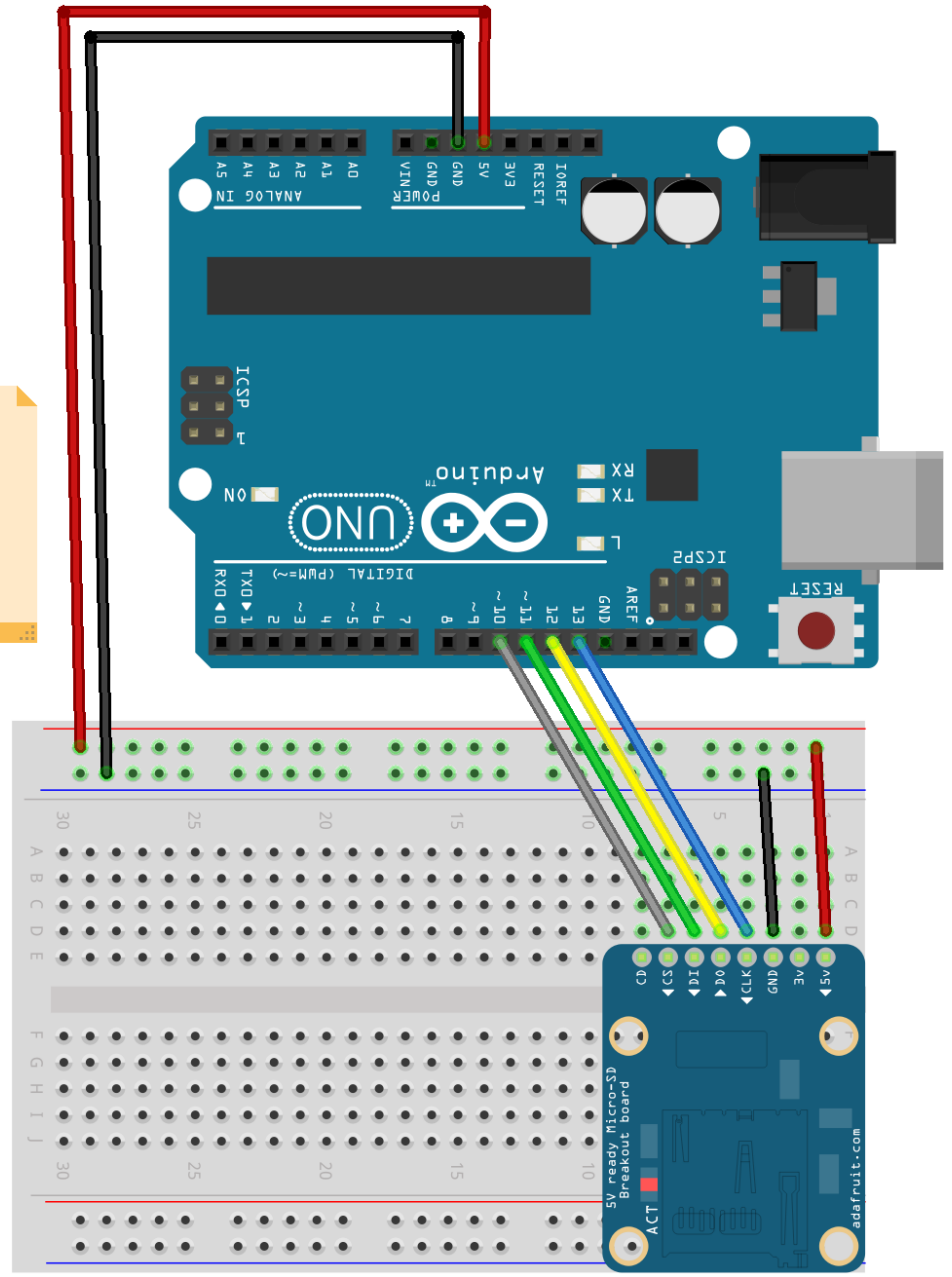
SD Card Library for Arduino

- Already installed by default
- Enables reading and writing contents of an SD or MicroSD card
- Examples of use: https://learn.adafruit.com/adafruit-micro-sd-breakout-board-card-tutorial/library

Micro SD Breakout Wiring

Apply the 5V Power to the (+) and GND to the (-) on the rail

Then, we can use it for both the SD Card breakout and for the sensor



Breakout CS --> Arduino Digital Port 10
Breakout DI --> Arduino Digital Port 11
Breakout DO --> Arduino Digital Port 12
Breakout CLK --> Arduino Digital Port 13
Breakout GND --> Arduino GND
Breakout 5V --> Arduino 5V

Adafruit Shield

- CS → 10

Creating Data Logger

Using Internal EEPROM

In Arduino

EEPROM in Arduino

- Permanent storage of Data
- Easy retrieval
- Easy deletion
- Limited Space
- 1024 Bytes on Uno
- 4096 Bytes on MEGA

Internal EEPROM

- Permanent Data storage in Arduino
- Uno has 1024 bytes (1K)
- 4k on Arduino Mega

- read()
- write()
- update()
- get()
- put()
- EEPROM[]

Code

```
#include <EEPROM.h>
void setup()
{
  /** Empty setup. **/
}

void loop() {
  int val = analogRead(0) / 4;
  EEPROM.write(addr, val);
  delay(100);
}
```

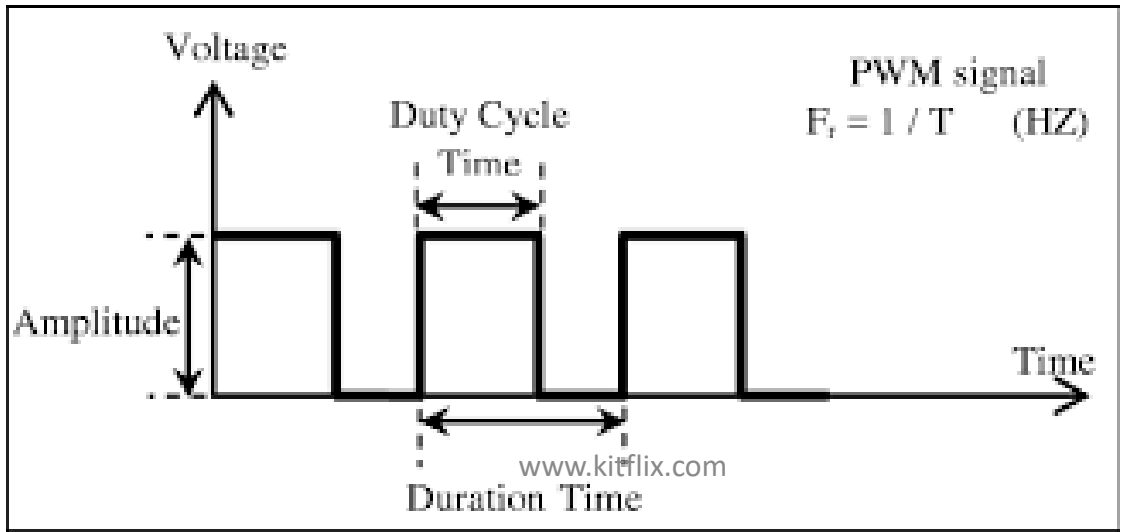
Project

- Accept input from Serial (0 – 9)
- Store in eeprom if number is larger than 6
- Display on boot (largest no is : xyz)

PWM with Arduino

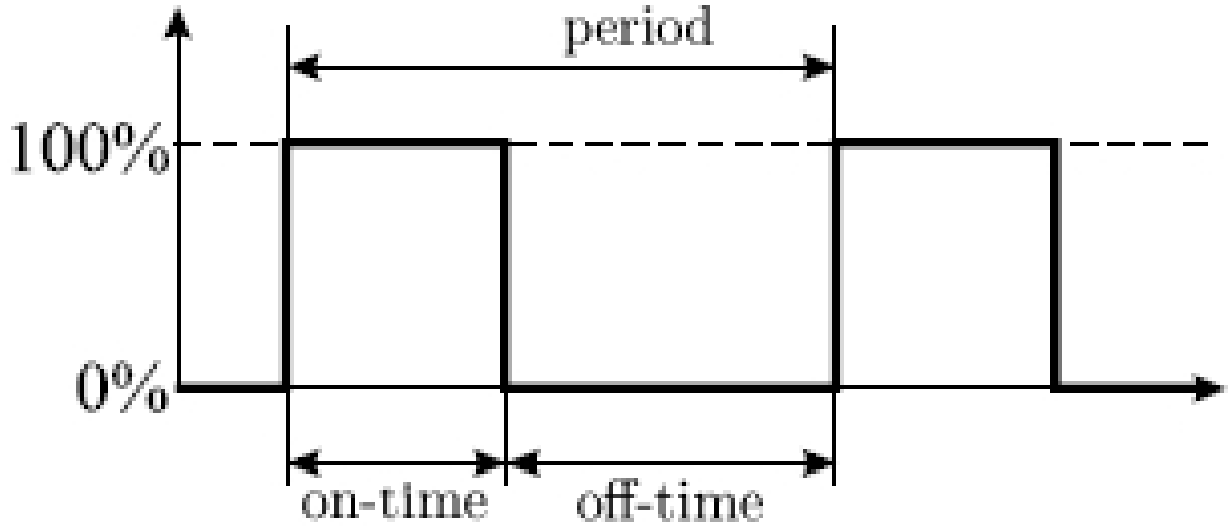
What is PWM?

- Definition: **P**ulse **W**idth **M**odulation is a technique that generates a square wave with varied Pulse Width
- The general purpose of Pulse Width Modulation is to control power delivery.
- The on-off behavior changes the average power of signal.
- Output signal alternates between on and off within a specified period.
- We can do digitalWrite(pin, HIGH) delay() digitalWrite(pin,LOW) delay() to generate pwm



Definitions

- Duty Cycle: average ON time
- Ton
- Toff
- Total Time T = Ton + Toff
- Duty cycle = (Ton / Total)
- Duty Cycle = (Ton / Ton+Toff) x 100 %
- Generate Without Writing Code
- P.W.M. is available on Arduino pins 3, 5, 6, 9, 10 and 11



PWM on Arduino

- In arduino software command is used to generate PWM Effect.
- analogWrite(analog pin number,value)

- PWM pins : 3,5,6,9,10,11 on Uno
- Value: 0-255

PWM on Arduino

- analogWrite(pin, duty_cycle) duty 0 – 255

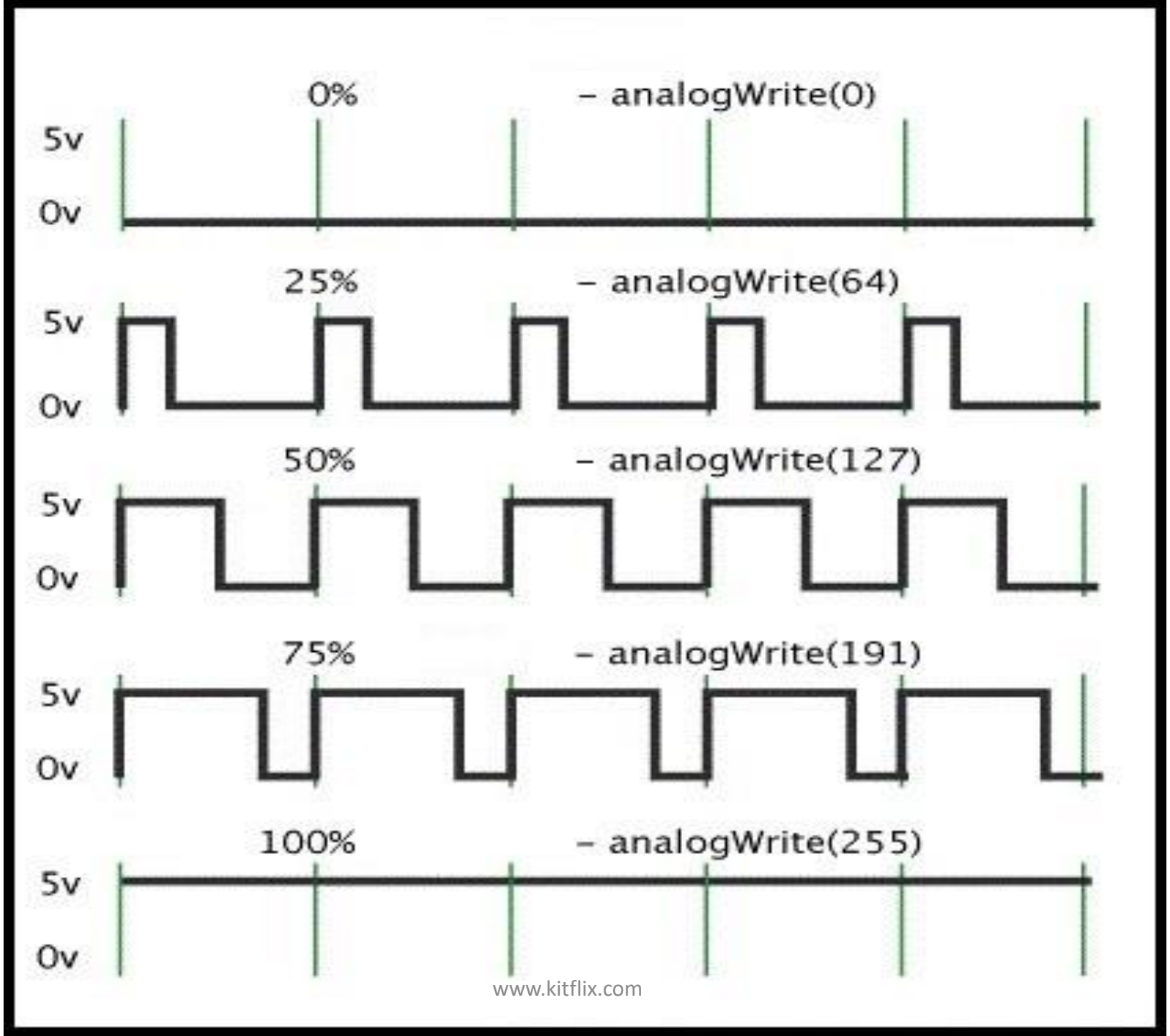
analogWrite()

[Analog I/O]

Description

Writes an analog value (PWM wave) to a pin. Can be used to light a LED at varying brightnesses or drive a motor at various speeds. After a call to `analogWrite()`, the pin will generate a steady rectangular wave of the specified duty cycle until the next call to `analogWrite()` (or a call to `digitalRead()` or `digitalWrite()`) on the same pin.

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)



LED Fading Effect

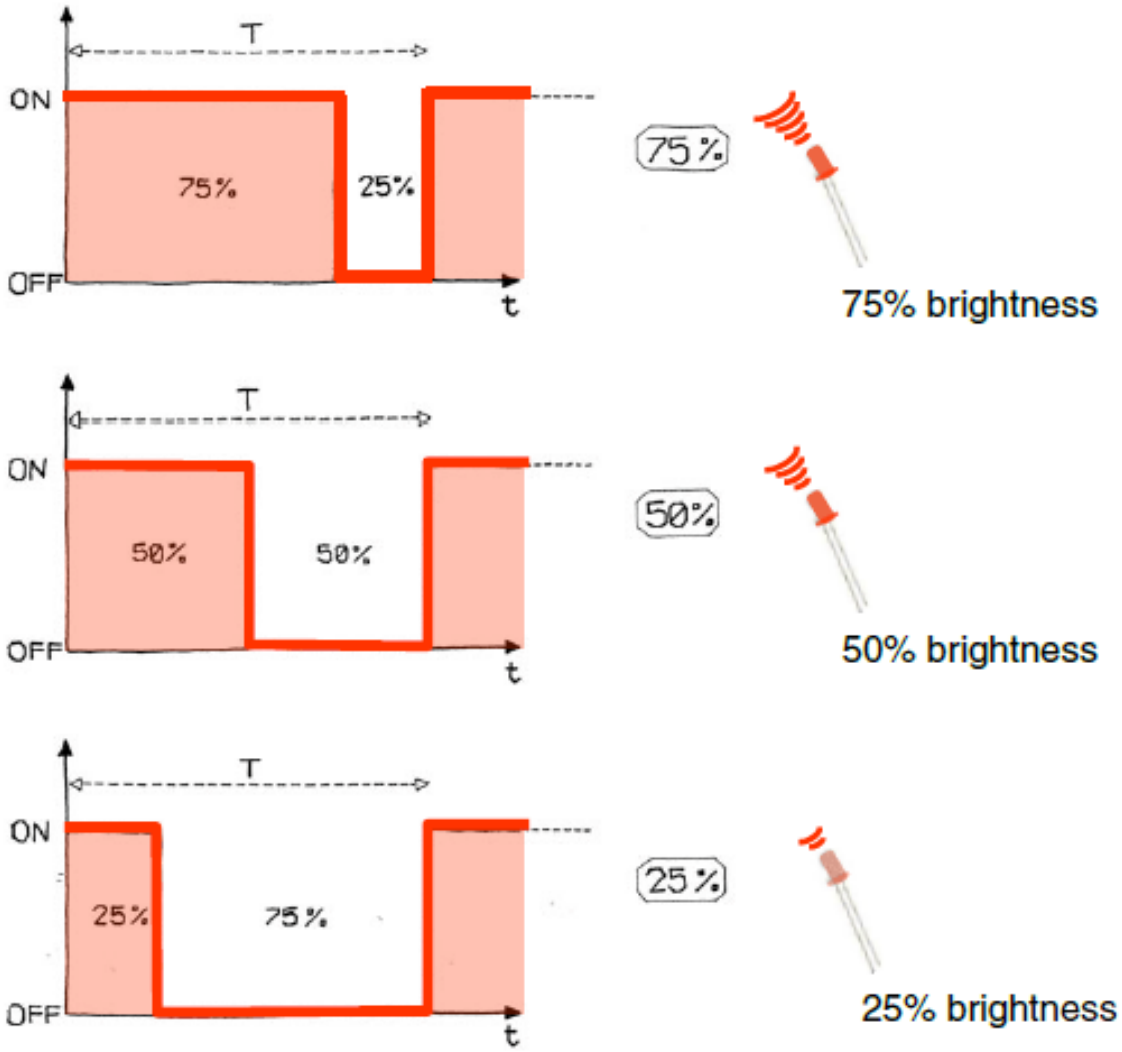


Image from *Theory and Practice of Tangible User Interfaces* at UC Berkley

LED Fading

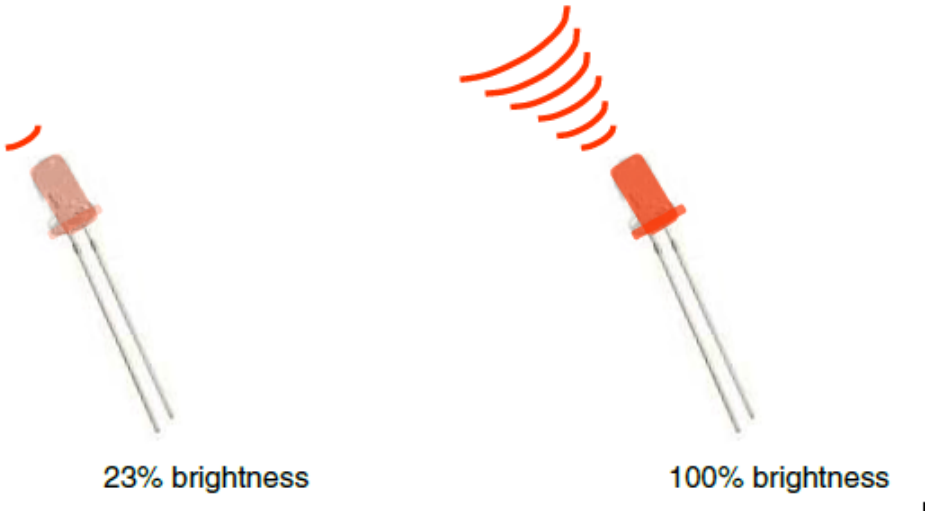
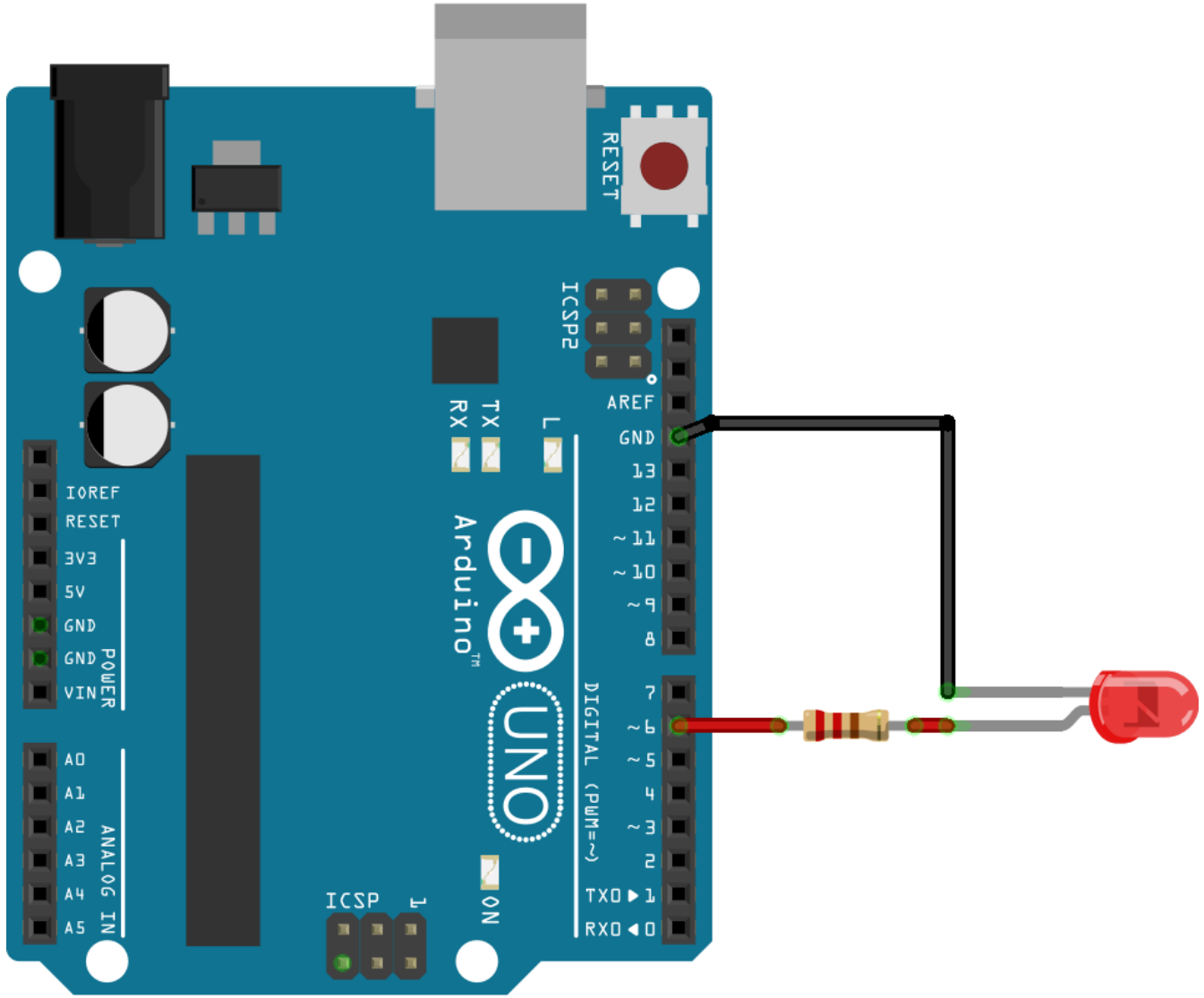


Image from *Theory and Practice of Tangible User Interfaces* at UC Berkley

Applications

- Dimming an LED
- Generating audio signals
- Providing variable speed control for motors.
- Servo motor : Rotational angle depend upon the incoming wave duty cycle



DC Motor Interfacing with Arduino

DC MOTOR

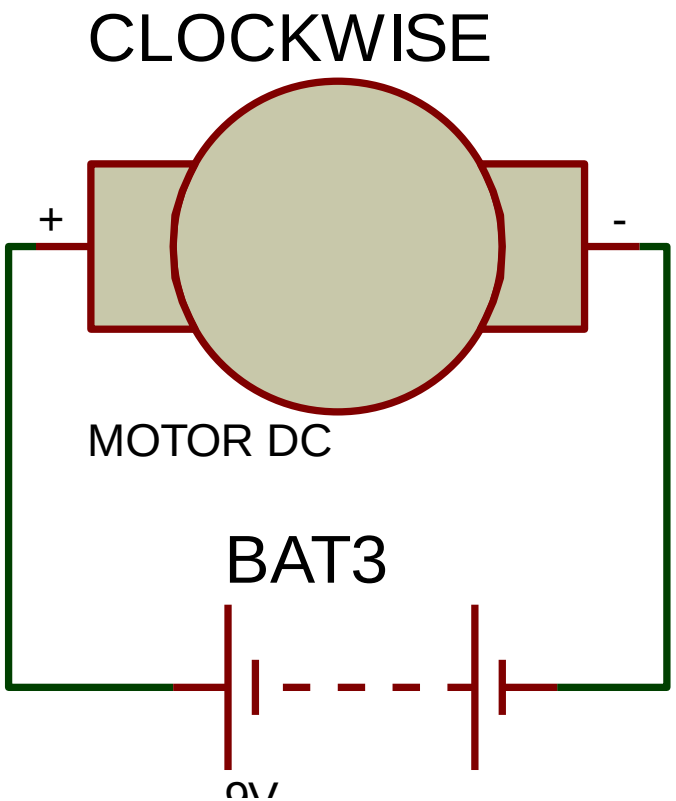
A DC motor is any of a class of electrical machines that converts direct current electrical power into mechanical power.

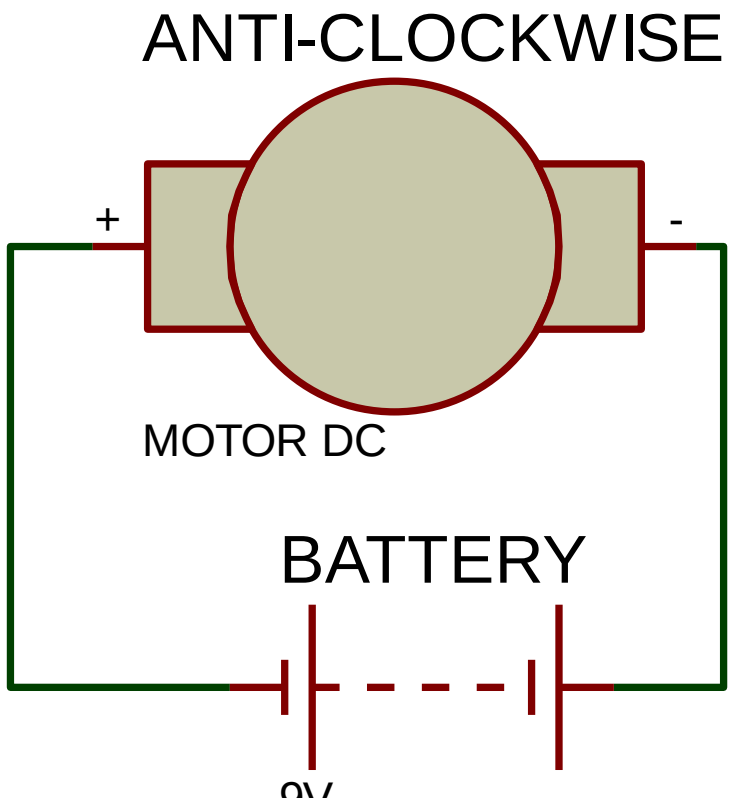
DC Motors

Driving DC Motor

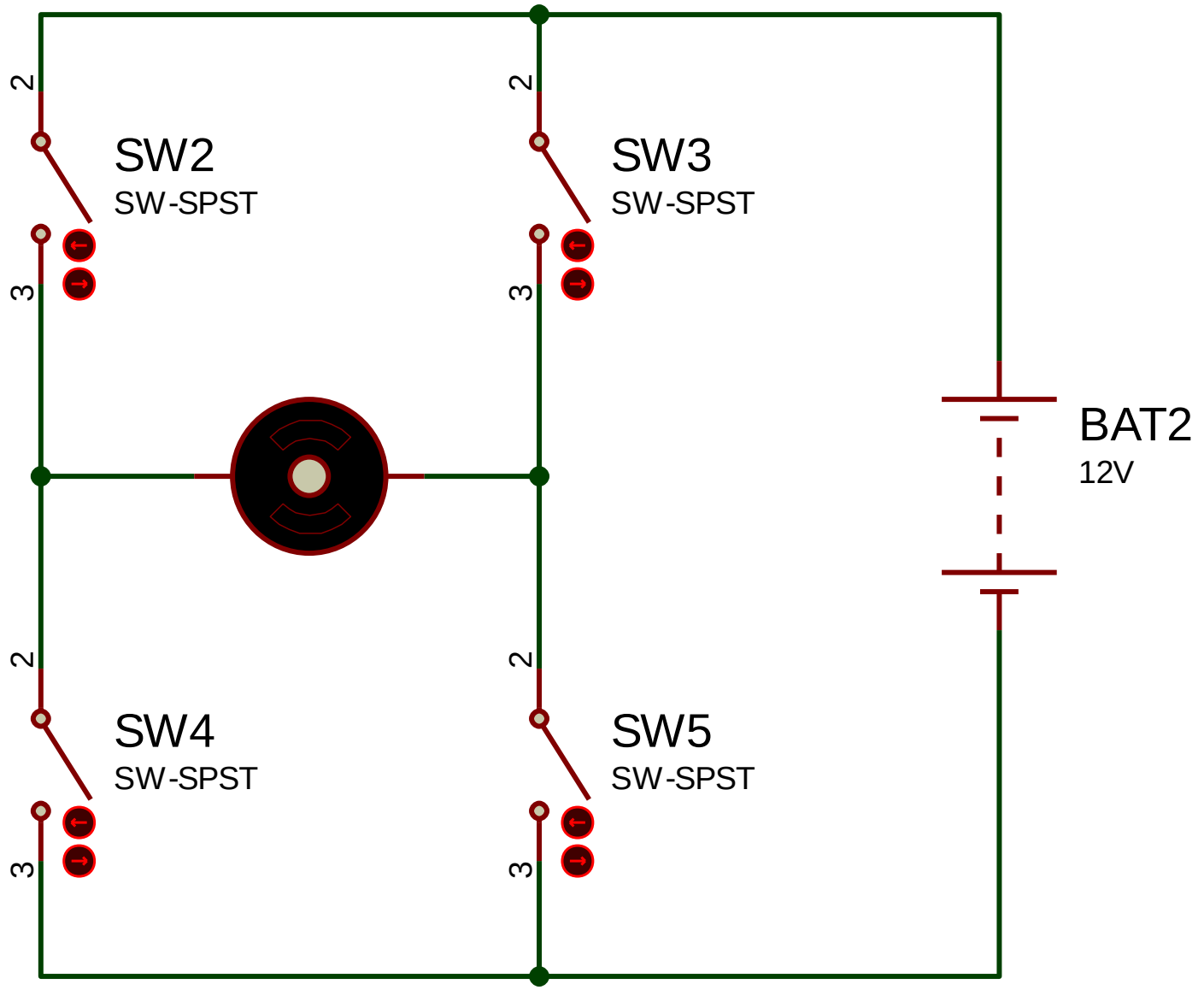
- Control the movement
- Clockwise
- Anticlockwise
- Control the Speed of motor
- +5v, 6v, 12v DC motors
- Geared or non-geared dc motors

DC motor Connection





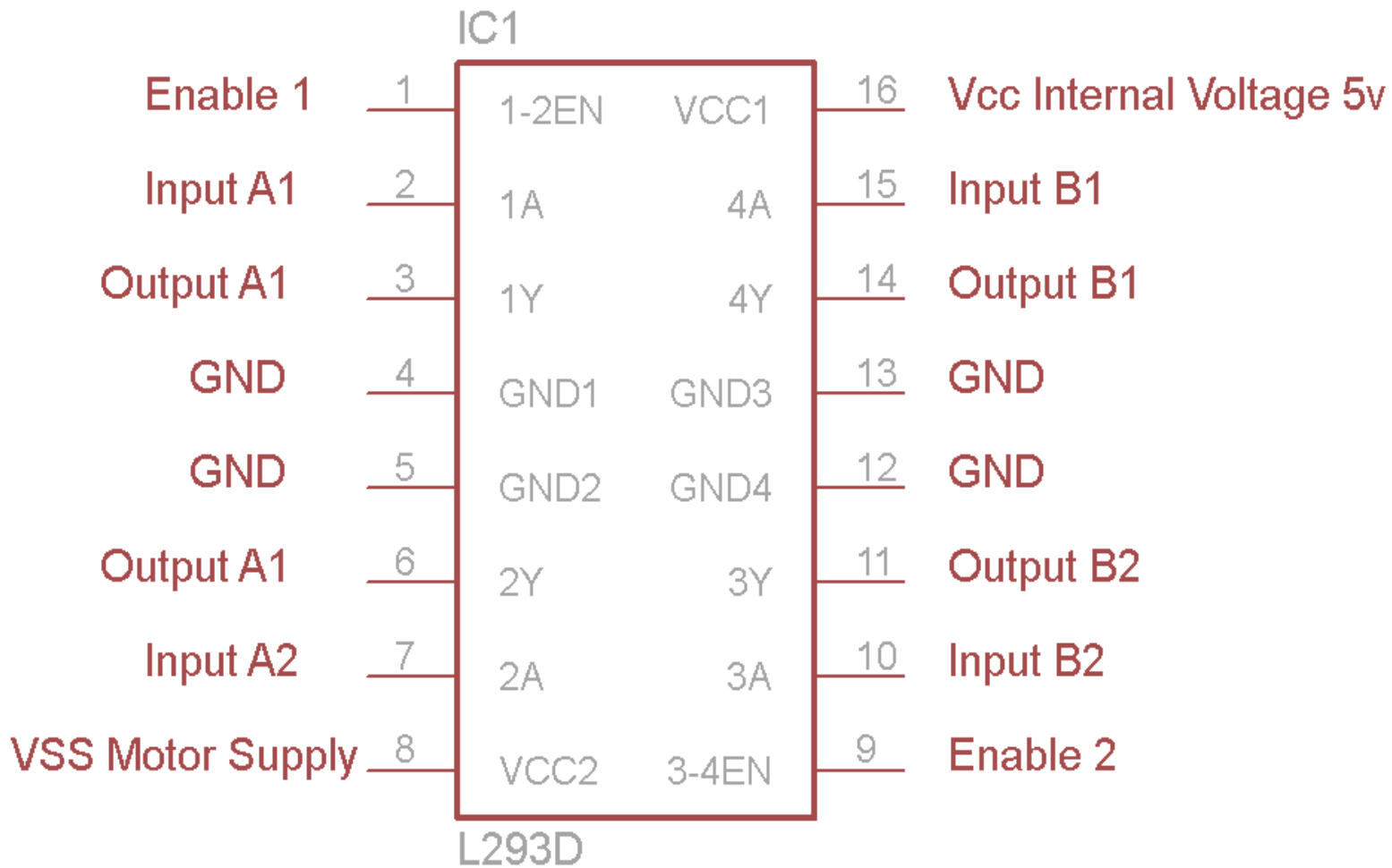
H-Bridge Concept



Basics of L293D

- Dual DC Motor Driver
- Takes low voltage signal from microcontroller
- Supports driving of 2 motors
- Speed control and direction control
- Based on H bridge driver circuits
- Needs no external components
- 600mA per channel

L293D Pin-Out



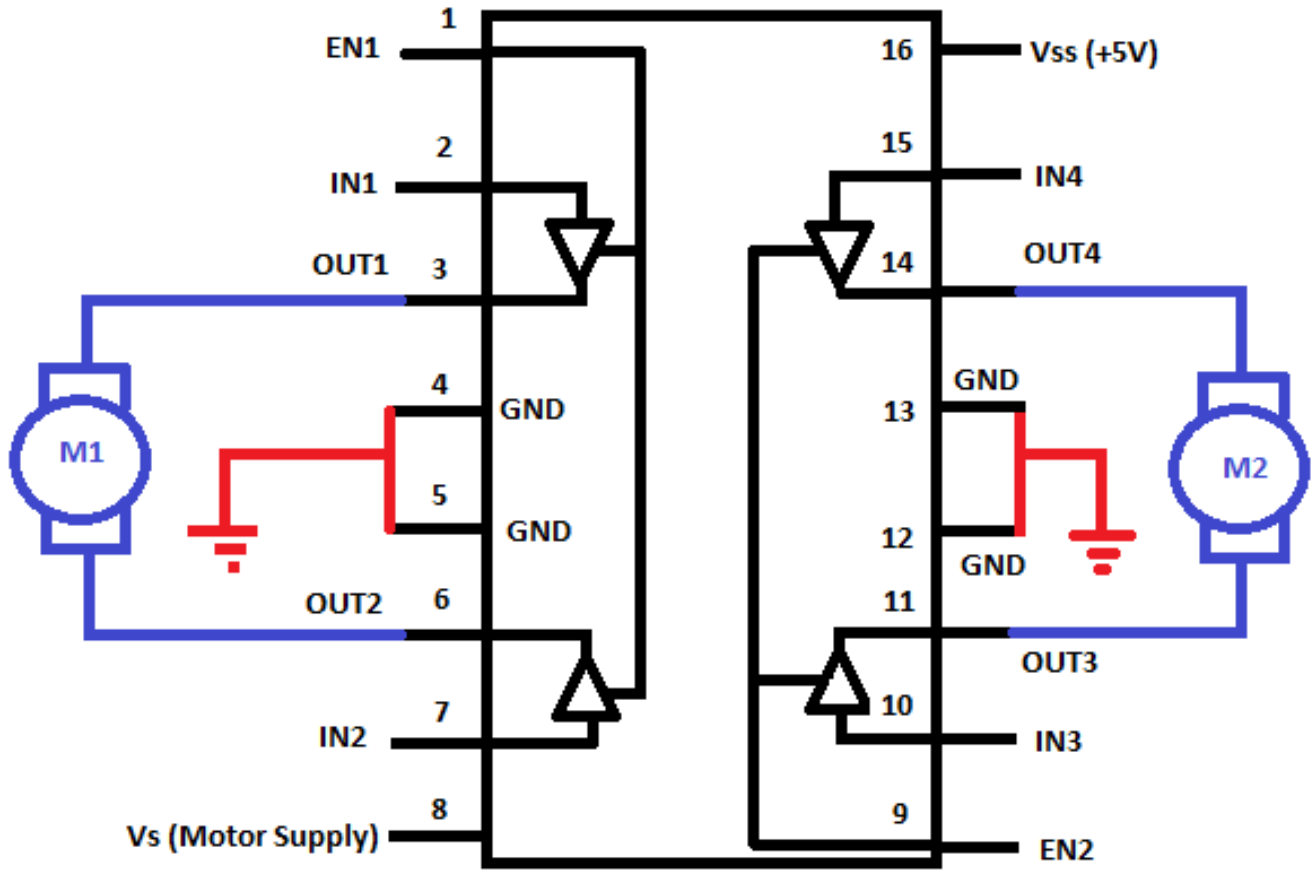
Pin Description

Pin Number	Function
1	Enable pin for Motor 1. It Should be active high
2	Input 1 for Motor 1
3	Output 1 for Motor 1
4	Ground (0V)
5	Ground (0V)
6	Output 2 for Motor 1
7	Input 2 for Motor 1
8	Supply voltage for Motors. In between 9-12V

Pin Number	Function
9	Enable pin for Motor 2. It Should be active high
10	Input 1 for Motor 2
11	Output 1 for Motor 2
12	Ground (0V)
13	Ground (0V)
14	Output 2 for Motor 2
15	Input 2 for Motor 1
16	Supply voltage. 5V

Truth Table

Input 1 (I3)	Input 2 (I4)	Motor1 (M2)
LOW	LOW	OFF
LOW	HIGH	Clockwise
HIGH	LOW	Anti-Clockwise
HIGH	HIGH	OFF


1
EN1
2
IN1
OUT1
3
4
GND
M1
5
GND
OUT2
6
7
IN2
8
Vs (Motor Supply)
16
Vss (+5V)
15
IN4
14
OUT4
GND
13
M2
12
GND
11
OUT3
10
IN3
9
EN2

Sample Experiment

Servo Motor with Arduino

What is a servo?

A servo-motor is an actuator with a built-in feedback mechanism that responds to a control signal by moving to and holding a position, or by moving at a continuous speed.

DC Motors and Servos

DC Motor

- Motion is continuous
- Speed controlled by applied voltage
- No control circuit

Servo

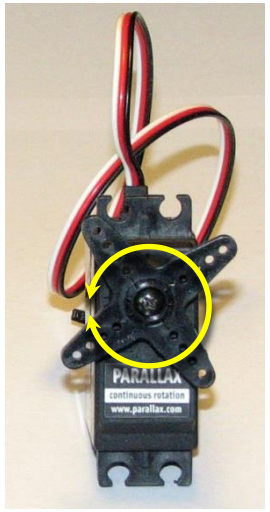
- Capable of holding a position
- Can be continuous or fixed angle
- Speed controlled by delay between position updates
- Hybrid of motor, gears and controller.
- Includes its own control signal

Conventional and Continuous Rotation

Two types of servos

Continuous

can rotate all the way around in either direction

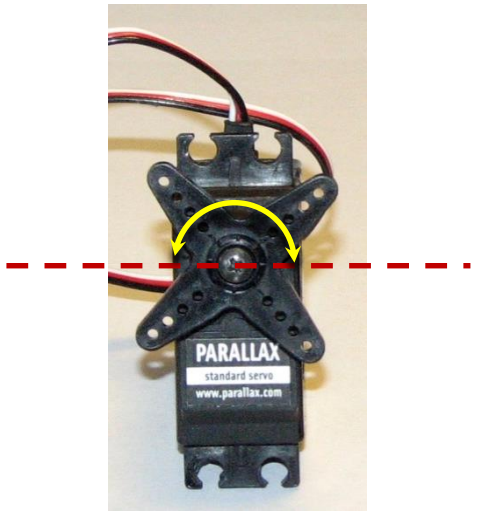


pulse tells servo

which way to spin & how fast to spin

Standard

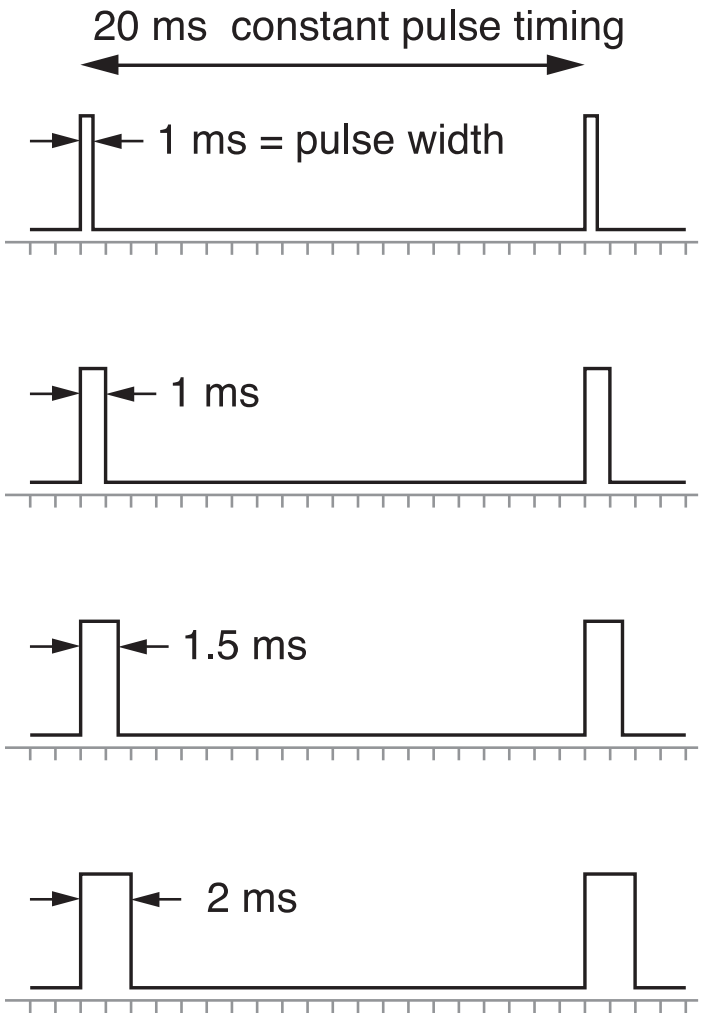
can only rotate 180 degrees



pulse tells servo

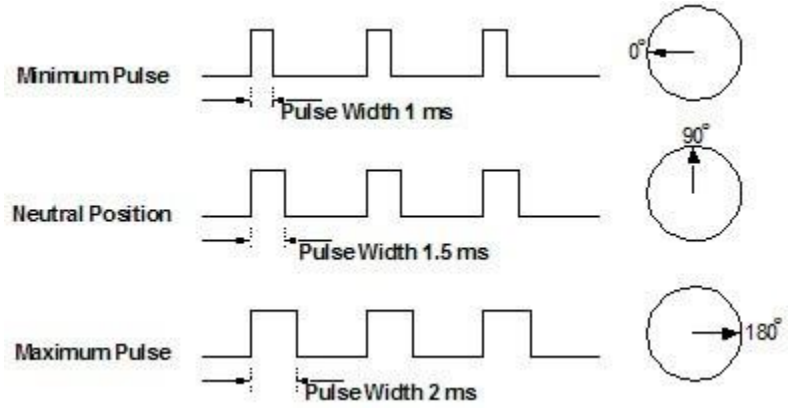
which position to hold

Control signal is a pulse train



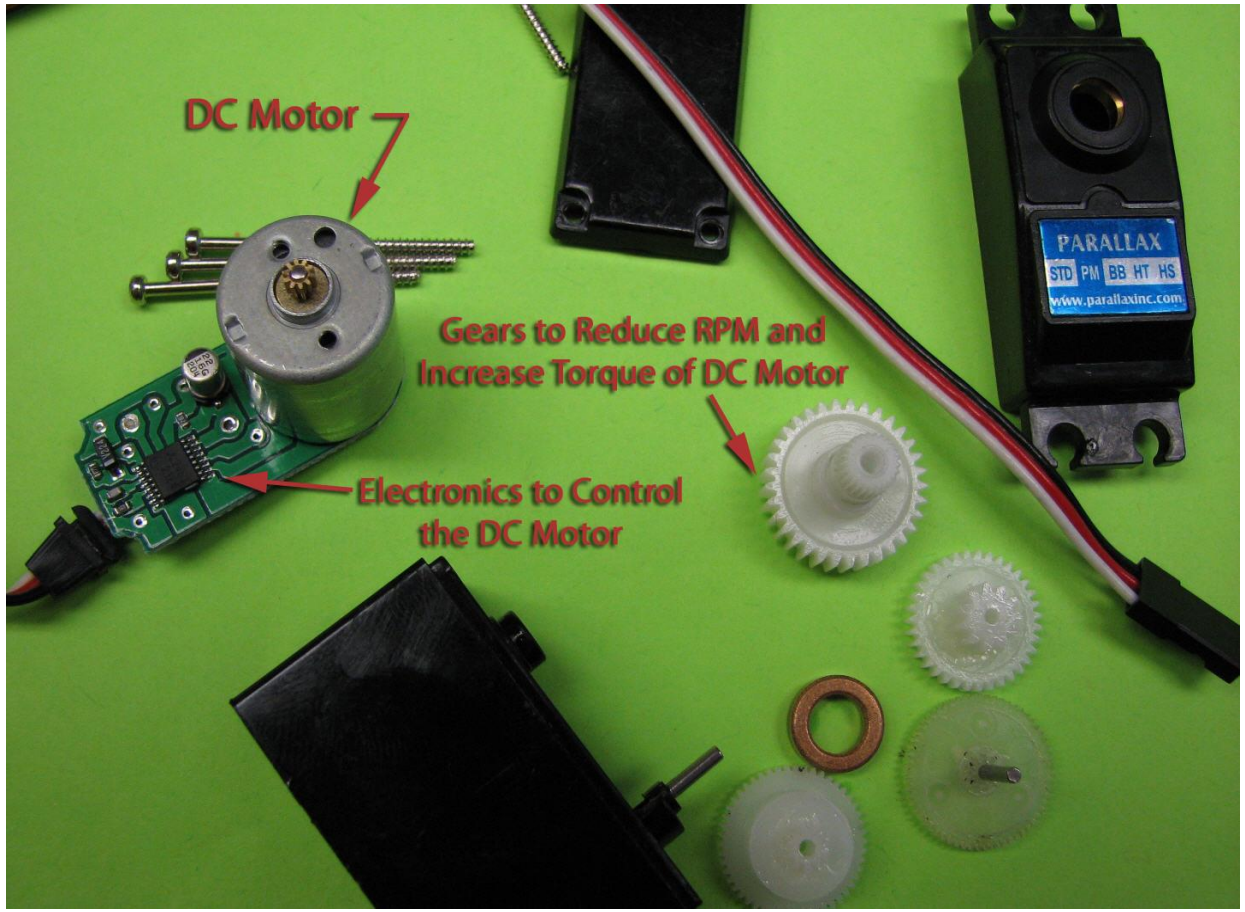
Pulse frequency is fixed
Typical: 20 ms

Pulse width determines position
Typical: 1ms to 2 ms

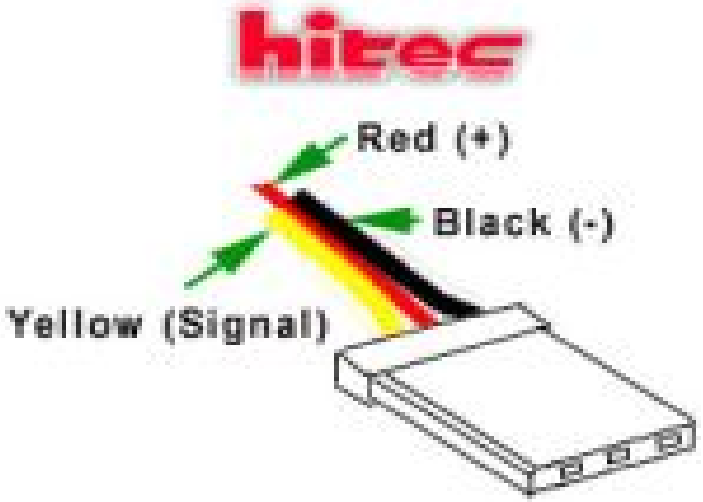


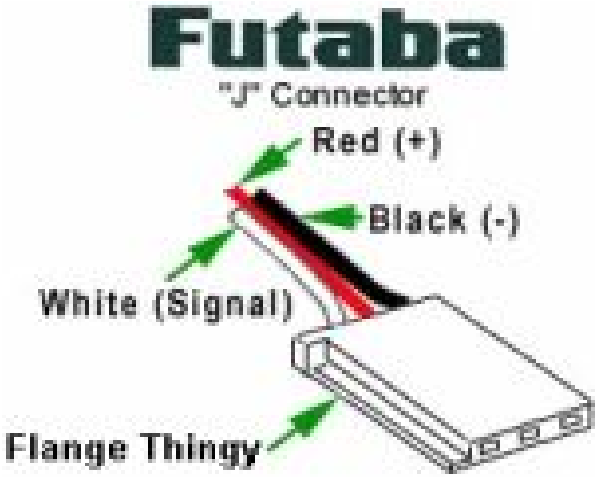
Servo components

1. Small DC motor
2. Gearbox with small plastic gears to reduce the RPM and increase output torque
3. Special electronics to interpret a pulse signal and deliver power to the motor



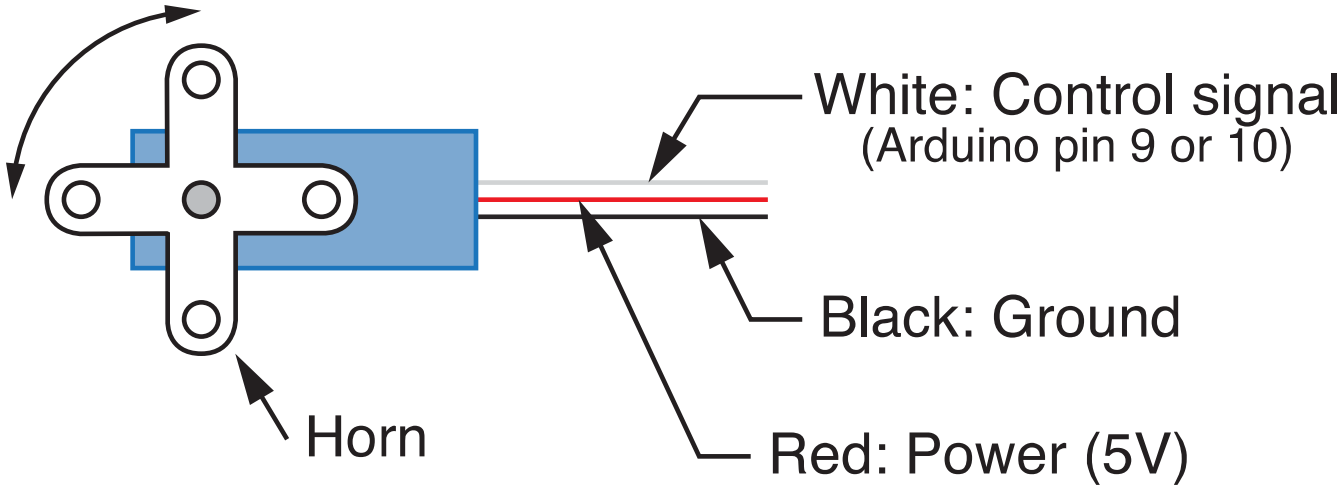
Servo Wiring





Common Servo Connection

Here are the common wiring connections of common servo motor



Arduino Servo library handles the details

- Connect Servo on any pin of Arduino
- analogWrite won’t work on pin 9 and 10 with servo being used
- Upto 12 Servo’s can be controlled with Most Arduino boards like Uno
- Upto 48 servo can be controlled with MEGA

Arduino Servo library handles the details

- Three components of the Servo Library
  - Create the servo object

```
Servo my_servo_object;
```

Name of the object is like a variable name.

  - Attach the object

```
my_servo_object.attach(servo_pin);
```

  - Send control signal

```
my_servo_object.write(pos);
```

`attach` and `write` are pre-defined methods that act on the servo object.

Sweep Code

- Move servos in clock wise direction
- Move servo full in counter clock wise direction

Sample Code

```
#include <Servo.h>   // Make code in Servo.h available to this sketch
Servo myservo;       // Create servo object called "myservo"
int servo_pin=9;     // The servo must be attached to pin 9 or pin 10

void setup()
{
  myservo.attach(servo_pin);   // attaches the servo pin to myservo object
}

void loop()
{
  int pos = 0;           // variable to store the servo position
  int dtwait=15;         // duration of wait at the end of each step
  for(pos = 0; pos < 180; pos++) {
    myservo.write(pos);               // Move to position in variable 'pos'
    delay(dtwait);                    // wait dtwait for the servo to reach the position
  }
  for(pos = 180; pos>=0; pos--) {
    myservo.write(pos);              // Move to position in variable 'pos'
    delay(dtwait);                   // wait dtwait for the servo to reach the position
  }
}
```

Project

- Use 2 switches and control servo
- Switch 1 → Servo Clockwise direction by 1 degree
- Switch 2 → Servo Counter Clockwise direction by 1 degree

Experiment

- What happens when you adjust dtwait?
- Can adjust the sweep angle?
  - Make new variable to define end angle of the loop
- Open the Knob demo from the Arduino IDE
  - Connect a potentiometer to an analog input
  - Use the potentiometer to control the servo position

MQ Series Gas Sensors

With Arduino

MQ Series Sensors

- Works on heating coil principal
- Sensitive to wide no of gases
- MQ-7, CO Sensor
- MQ135, Air Quality (Benzene, Alcohol, smoke)
- MQ-2, Smoke, LPG, Butane
- MQ-3, Alcohol Sensor

Analog Output

- Analog output
- Sensitivity depends on the coil resistor
- Use sensor module
- Analog Read
- Print on serial monitor
- Take action!!!

Gas Detector Project

- MQ-6
- LPG Sensor
- Analog / digital Output
- +5v operating voltage
- DO not use on USB, it requires about 250mA current
- Use external power adapter before powering on
- Wait for at least 2 minutes for sensor to stabilize