SNS COLLEGE OF TECHNOLOGY

An Autonomous Institution
Coimbatore-35

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A+' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

19ECT308-WIRELESS TECHNOLOGIES FOR IoT

III YEAR/ VI SEMESTER

UNIT 4 – PROTOTYPING AND DESIGNING SOFTWARE FOR IOT APPLICATIONS

TOPIC Reading data from sensors and devices, Devices, Gateways

Fundamentals of Arduino Programming

Structure

```
void setup()                    //Preparation function used to declare variables
{                               //First function that runs only one in the
program
        Statement(s);           //used to set pins for serial communication
}
void loop()                     //Execution block where instructions are
executed repeatedly
{                               //this is the core of the Arduino programming
        Statements();           //Functionalities involve reading inputs,
triggering outputs etc.
}
```

Fundamentals of Arduino Programming

setup()

```
void setup()
{
        pinMode(pin, INPUT);        //'pin' configure as input
}
```

loop()

```
void loop()                         //After calling setup(),loop() function does its task
{
        digitalWrite(pin, HIGH);            //sets 'pin' ON
        delay(10000);                       //pauses for ten thousand mili second
        digitalWrite(pin, LOW);             //sets 'pin' OFF
        delay(10000);                       //pauses for ten thousand mili second
}
```

Fundamentals of Arduino Programming

Functions

Syntax: *type functionName(parameters)*

{

Statement(s);

}

Example:

```
int delayvar()
{
        int var;                                //create temporary variable var
        var=analogRead(potent);     //read from potentiometer
        var=var/4;                              //convert the value of variable var

        return var;                             //return var
}
```

Fundamentals of Arduino Programming

{ } curly braces

They define beginning and end of function blocks, unbalanced braces may lead to compile errors.

semicolon

It is used to end a statement and separate elements of a program.
Syntax: *int x=14;*

/*…..*/ block comments

Multiline comments begin with /* with a description of the block and ends with */.
Syntax: */*This is an enclosed block of comments*
Use the closing comment to avoid errors/*

Fundamentals of Arduino Programming

//line comments

Single line comment begins with // and ends with next instruction followed.

Syntax: *//This is a Single line comment*

Variables

Example:

int var; //variable 'var' visible to all functions

Data Types

Data type	Syntax	Range
Byte	byte x=100;	0-255
Int	int y=200;	32767 to -32768
Long	long var=8000;	2147483647 to -2147483648
Float	float x=3.14;	3.4028235E+38 to -3.4028235E+38
arrays	int myarray[]={10,20,30,40}	Size depends on the data type associated with declaration.

Fundamentals of Arduino Programming

Operators

Operator	Syntax and its usage

Arithmetic operators
(+,-,/,*)
	x=x+5;
	y-y-6;
	z=z*2;
	p=p/q;

Assignment operators
(=,++,--,+=, =,*=,/=)
	x++; //same as x=x+1
	x+=y; //same as x=x+y
	x-=y; //same as x=x-y
	x*=y //same as x=x*y
	x/=y //same as x=x/y

Comparison operators
(==,!=,<,>,<=,>=)
	x==y //x is equal to y
	x!=y //x is not equal to y
	x<y //x is less than y
	x>y //x is greater than y
	x!=y //x is not equal to y

Logical operators (&&,||,!)	x>2 && x<5 //Evaluates to true only if both expression are true x>2 || y>2 //Evaluates to true if any one expression is true !x>2 //true if only expression is false

Fundamentals of Arduino Programming

Constants

Constants	Usage

TRUE/FALSE	Boolean constants true=2 and false=0 defined in logic levels.
	if(b==TRUE)
	{
	//do something
	}

INPUT/OUTPUT	Used with pinMode () function to define levels. pinMode(13,OUTPUT):
HIGH/LOW	Used to define pin levels HIGH-1, ON, 5 volts LOW-0,OFF, 0 volts digitalWrite(13,HIGH);

Fundamentals of Arduino Programming

Flow control Statements

if

```
if(some_variable == value)
{
  Statement(s);    //Evaluated only if comparison results in a true value
}
```

if…else

```
if(input==HIGH)
{
  Statement(s);    //Evaluated only if comparison results in a true value
}
else
{
  Statement(s);  //Evaluated only if comparison results in a false value
}
```

Fundamentals of Arduino Programming

for

```
for(initialization;condition;expression)
{
    Dosomething;   //Evaluated till condition becomes false
}

for(int p=0;p<5;p++)       //declares p, tests if less than 5, increments by 1
{
digitalWrite(13,HIGH);  //sets pin 13 ON
delay(250);                // pauses for ¼ second
digitalWrite(13,LOW);  //sets pin 13 OFF
delay(250);                //pause for ¼ second
}
```

while

```
while(some_variable ?? value)
{
  Statement(s);    //Evaluated till comparison results in a false value
}
```

Fundamentals of Arduino Programming

do...while

```
do
{
Dosomething;
}while(somevalue);
```

Fundamentals of Arduino Programming

Digital and Analog input output pins and their usage

Digital i/o

Methods	Usage

pinMode(pin, mode)	Used in setup() method to configure pin to behave as INPUT/OUTPUT
	pinMode(pin, INPUT) //'pin' set to INPUT
	pinMode(pin, OUTPUT) //'pin' set to OUTPUT

digitalRead(pin)	Read value from a specified pin with result being HIGH/LOW Val=digitalRead(pin); //Val will be equal to input 'pin'
digitalWrite(pin,value)	Outputs to HIGH/LOW on a specified pin. digitalWrite(pin, HIGH); //'pin' is set to HIGH

Example	int x=13; //connect 'x' to pin 13
	int p=7; //connect pushbutton to pin 7
	int val=0; //variable to store the read value
	void setup()
	{
	pinMode(x,OUTPUT); //sets 'x' as OUTPUT
	pinMode(p,INPUT); //sets 'p' as input
	}
	void loop()
	{
	val=digitalRead(p); // sets 'value' to 0
	digitalWrite(x,val); //sets 'x' to button value
	}

Fundamentals of Arduino Programming

Analog i/o

Methods	Usage
analogRead(pin)	Reads value from a specified analog pin works on pins 0-5. val=analogRead(pin); //'val' equal to pin
analogWrite(pin,value)	Writes an analog value using pulse width modulation (PWM) to a pin marked PWM works on pins 3, 5, 6,9,10.

Example	int x=10; //connect 'x' to pin 13
	int p=0; //connect potentiometer to analog pin 7
	int val; //variable for reading
	void setup() { } // No setup is needed
	void loop()
	{
	val=analogRead(p); // sets 'value' to 0
	val/=4;
	analogWrite(x,val); //outputs PWM signal to 'x'
	}

Fundamentals of Arduino Programming

time

Methods	Usage
delay(ms)	Pauses for amount of time specified in milliseconds. delay(1000); //waits for one second
millis()	Returns the number of milliseconds since Arduino is running. val=millis(); //'val' will be equal to millis()

math

Methods	Usage
min(x,y)	Calculates minimum of two numbers val=min (val,10); //sets 'val' to smaller than 10 or equal to 10 but never gets above 10.
max(x,y)	val=max(val, 10); // sets 'val' to larger than 100 or 100.

Fundamentals of Arduino Programming

random

Methods	Usage
randomSeed(value)	Sets a value/seed as starting point for random () function.
random(min,max)	Allows to return numbers within the range specified by min and max values. val=random(100,200); //sets 'val' to random number between 100-200

Example	int rnumber; // variable to store random value
	int x=10;
	void setup()
	{
	randomseed(millis()); //set millis() as seed
	rnumber=random(200); //random number from 0-200
	analogWrite(x,rnumber); //outputs PWM signal
	delay(500);
	}

Fundamentals of Arduino Programming

Serial

Methods	Usage

Serial.begin(rate)	Opens serial port and sets the baud rate for serial data transmission.
	
	void setup()
	{
	Serial.begin(9600); //sets default rate to 9600 bps
	}

Serial.println(data)	Prints data to the serial port Serial.println(value); //sends the 'value' to serial monitor

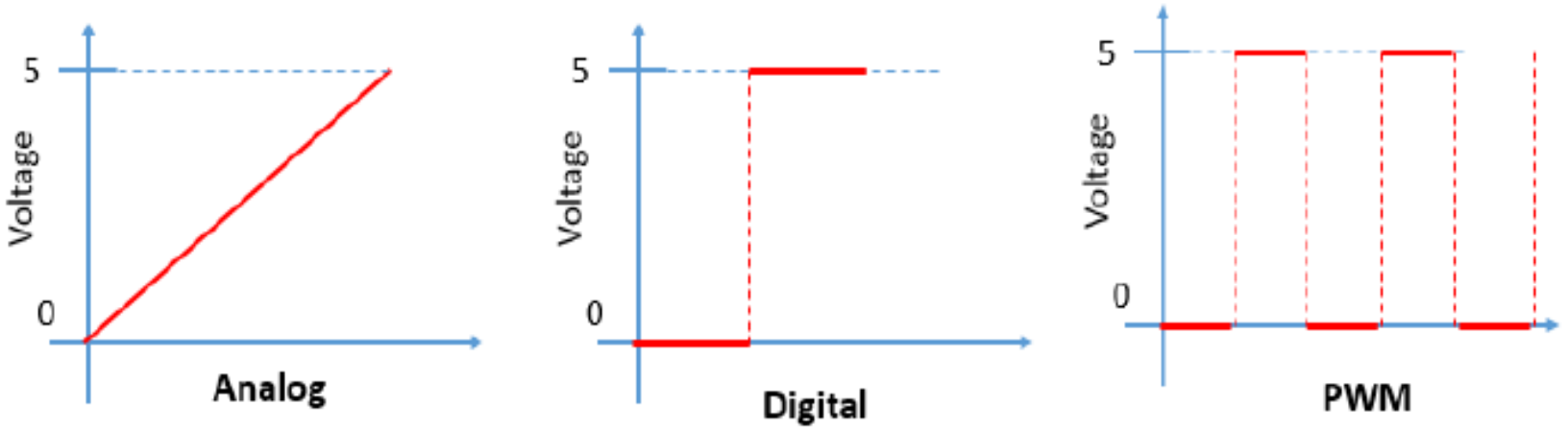
Difference between Analog, Digital and PWM Pins

In **analog pins,** you have unlimited possible states between 0 and 1023. This allows you to read sensor values. For example, with a light sensor, if it is very dark, you'll read 1023, if it is very bright you'll read 0 If there is a brightness between dark and very bright you'll read a value between 0 and 1023.

In **digital pins**, you have just two possible states, which are on or off. These can also be referred as High or Low, 1 or 0 and 5V or 0V. For example, if an LED is on, then, its state is High or 1 or 5V. If it is off, you'll have Low, or 0 or 0V.

PWM pins are digital pins, so they output either 0 or 5V. However these pins can output "fake" intermediate voltage values between 0 and 5V, because they can perform "Pulse Width Modulation" (PWM). PWM allows to "simulate" varying levels of power by oscillating the output voltage of the Arduino.

Fundamentals of Arduino Programming

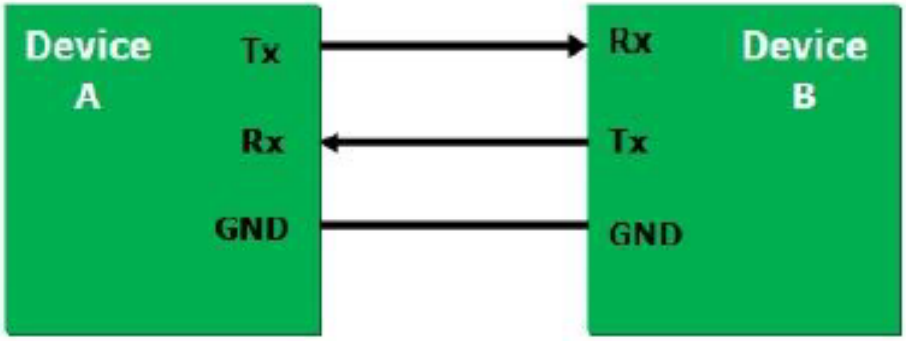


Difference between Analog, Digital and PWM Pins

Introduction to Communications

Serial (UART) communications:

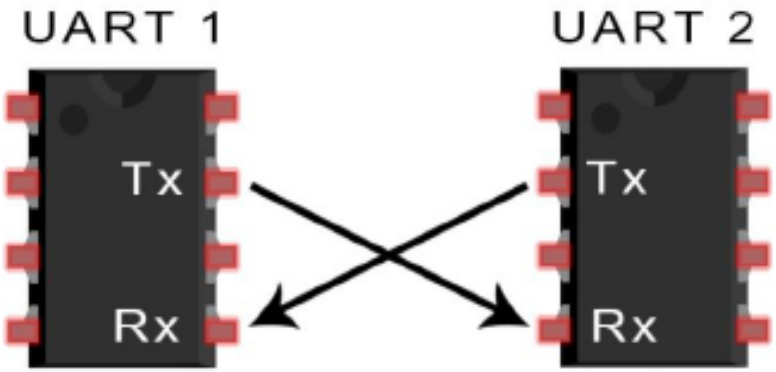
- Serial communication on Arduino pins Tx/Rx uses TTL logic levels which operates at either 5V/3.3V depending the type of the board used.
- Tx/Rx pins should not be connected to any source which operates more than 5V which can damage the Arduino board.
- Serial communication is basically used for communication between Arduino board and a computer or some other compatible devices.



Introduction to Communications

Serial (UART) communications:

- Every Arduino board will have at least one serial port known as UART.
- Serial communicates on digital pins Rx(pin 0) and Tx(pin 1) with the computer via USB, pin 0 and pin 1 cannot be used for digital input or output.
- The built in serial monitor can be used to communicate with an Arduino board by selecting same baud rate that is used in the call to begin () which will come across in the later part of the chapter



Introduction to Communications

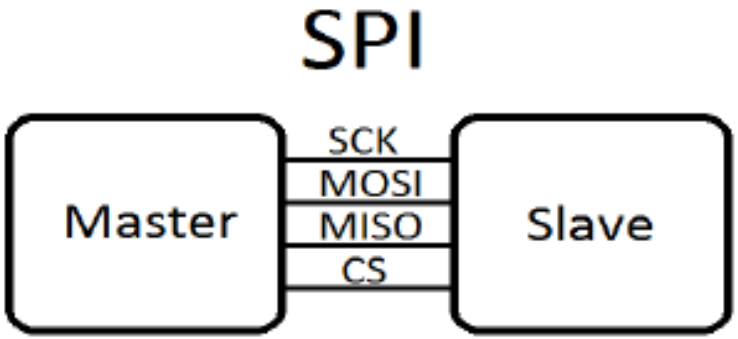
SPI communications

- Serial communication Interface (SPI) is a synchronous data protocol used by large microcontrollers for communicating with one or more peripheral devices for a shorter distance and also used for communication between two devices.
- With SPI there will be always one master device which is a microcontroller like Arduino which controls the functionalities of other peripheral devices.
- Devices have three lines in common which are as follows
  - MISO (Master in Slave Out)- Slave line for sending data to the master.
  - MOSI (Master Out Slave In)- Master sending data to peripherals
  - SCK Serial clock) - clock pulses which synchronize data transmission generated by the master And one of the specific line for every device is
  - SS (slave select) - pin on each device that the master can use to enable and disable specific devices.

Introduction to Communications

SPI communications

- When device SS pin is low, communication happens with the master, if SS pin is high device ignores the maser. This allows multiple SPI devices sharing the the same MISO, MOSI and CLK lines.
- To program a new SPI device some key points to be noted which are
  - Maximum SPI speed of the device used?
  - How data is shifted like MSB/LSB?
  - Data clock is idle when high/low.



Introduction to Communications

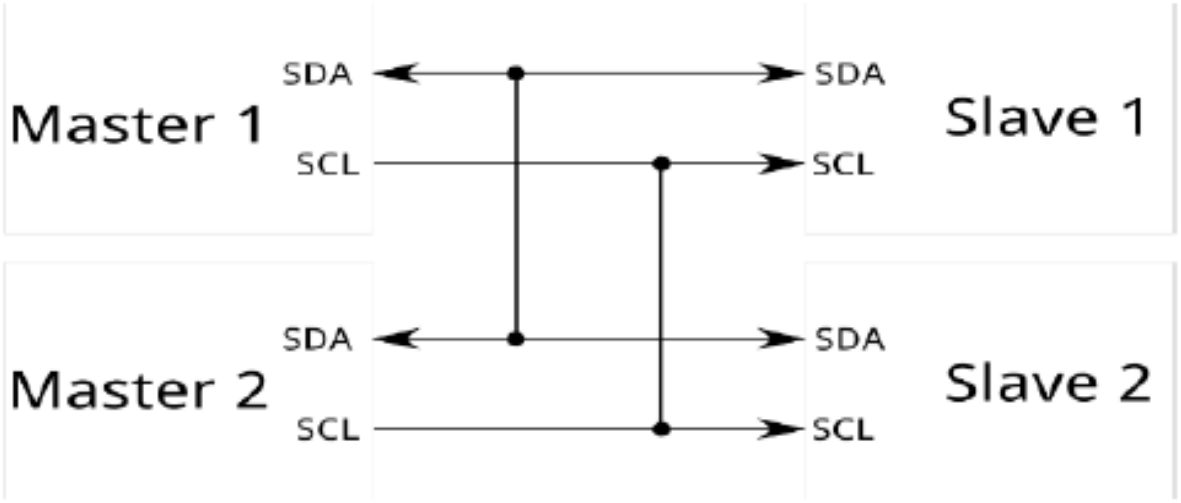
I^2C communications

- Inter-Integrated circuit or I^2C (I squared C) is one of the best protocol used when a workload of one Arduino (Master Writer) is shared with another Arduino (Slave receiver).
- The I^2C protocol uses two lines to send and receive data which are a serial clock pin (SCL) which writes data at regular intervals and a serial data pin (SDA) over which data sent between devices.
- When the clock signal changes from LOW to HIGH the information, the address corresponds to a specific device and a command is transferred from board to the I^2C device over the SDA line.
- This information is sent bit by bit which is executed by the called device, executes and transmits the data back.

Introduction to Communications

I²C communications

- If the device require execution from another a slave device, the data is transferred to the board on the same line using pulse generated from Mater on SCL as timing.
- Each slave should have unique identity and both Master and slave turns out communicating on the same data line. In this way many of the Arduino boards are communicated using just two pins of microcontroller with each unique address of a device.



Example modules on Arduino

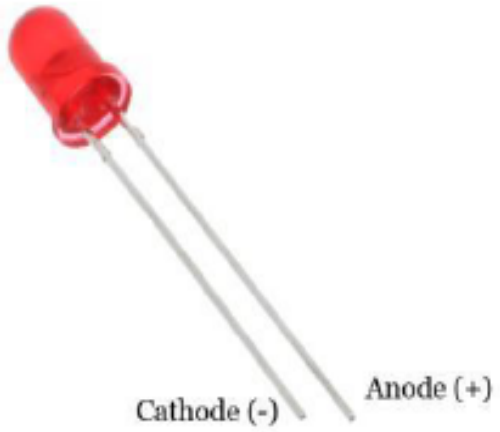
- Interfacing programs on Arduino using LED
- Programs to interact with Serial Monitor of our Computer Screen
- Interfacing Sensors
- Interfacing Display, GSM, GPS
- Interfacing Motors

Interfacing programs on Arduino using LED

1. Blinking an LED
2. Toggle the state of LED using Switch
3. Traffic light simulation for pedestrians
4. Create Dimmable LED using Potentiometer

Blinking an LED

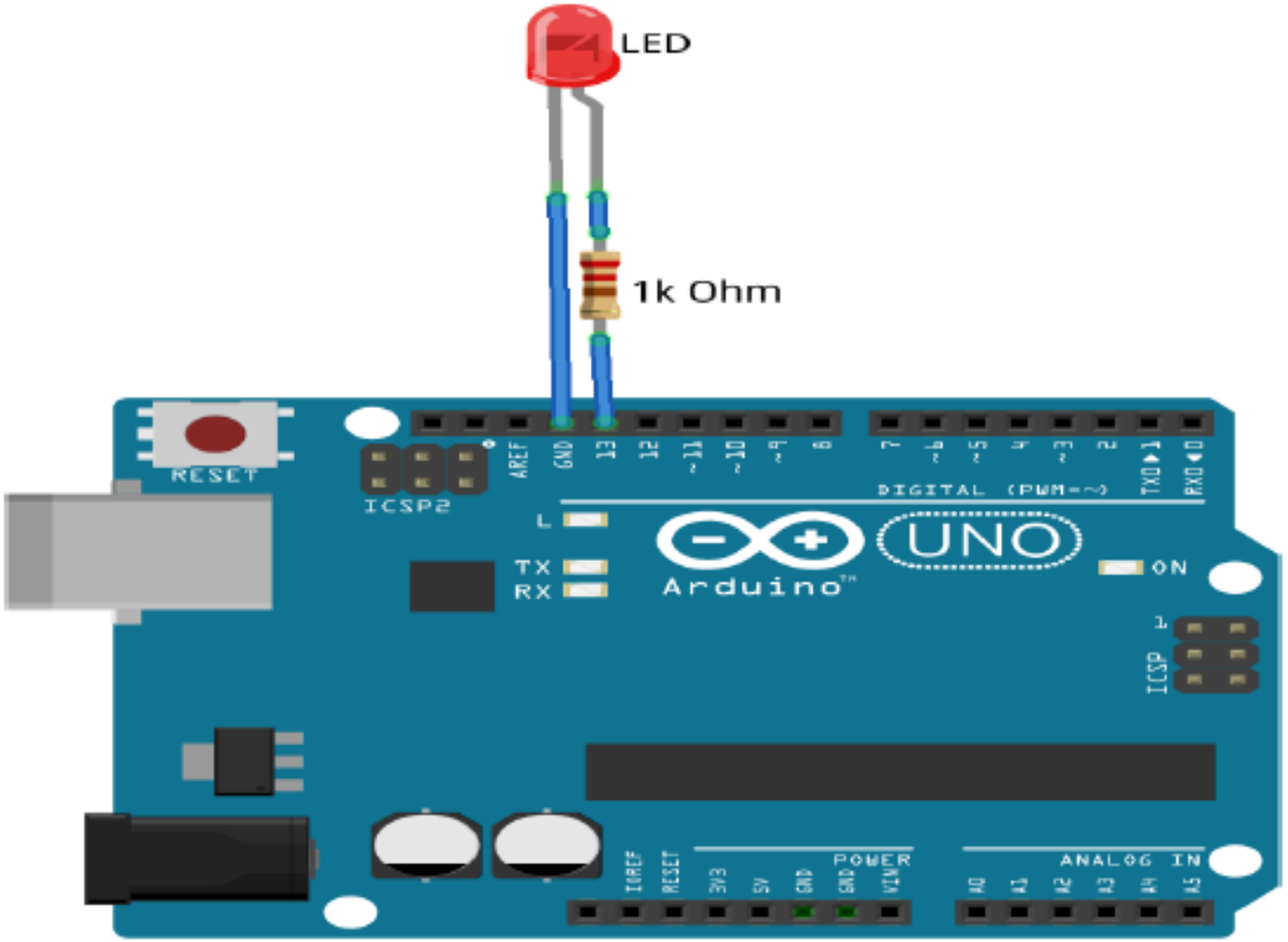
Components required	1-LED, 1-KΩ resistor, Jumper wires, Breadboard



The longest lead is the anode and the shortest is the cathode.

Blinking an LED

Circuit Diagram



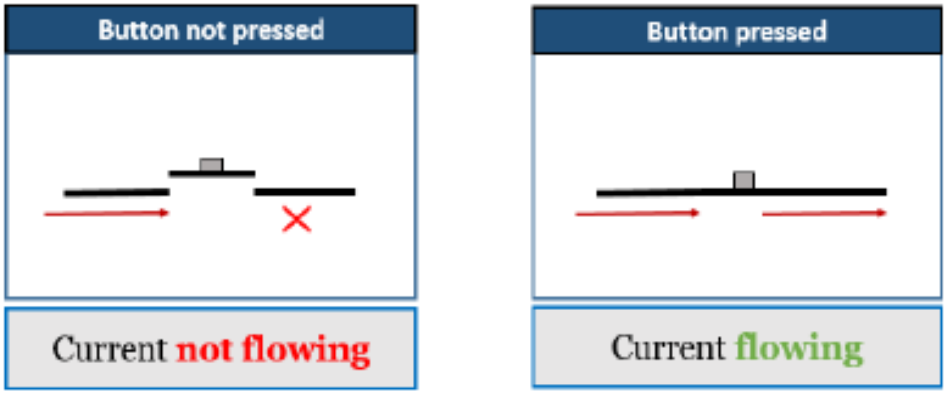
Blinking an LED

Code

```
/*The Function setup runs only once when Arduino board is first powered up
or a rest button the board is pressed */
void setup()
{
pinMode(13, OUTPUT); //pin 13 is set as an OUTPUT pin
}
//loop function iterates forever
void loop() {
digitalWrite(13, HIGH); //Sets LED to HIGH voltage
delay(1000); //delay by a second
digitalWrite(13, LOW); //Sets LED to LOW voltage
delay(1000); //delay by a second
}
```

Toggle the state of LED using Switch

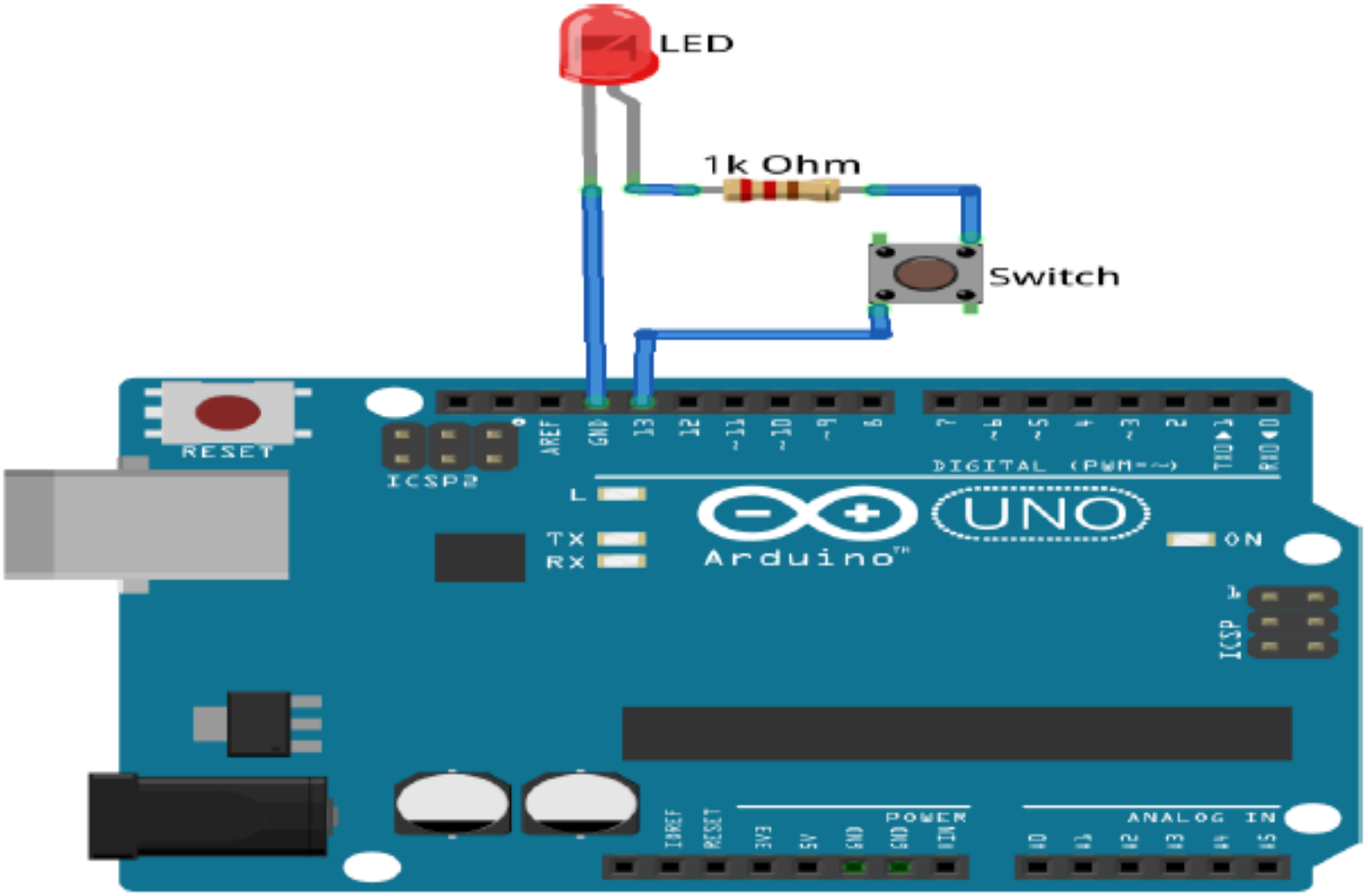
Components required	1-LED, 1-KΩ resistor, 1-push button, Jumper wires, Breadboard



Here an **open pushbutton** mechanism is used. In Normal state(not pushed) of the button current doesn't flow, only when button is pushed flow of current is allowed

Toggle the state of LED using Switch

Circuit diagram



Toggle the state of LED using Switch

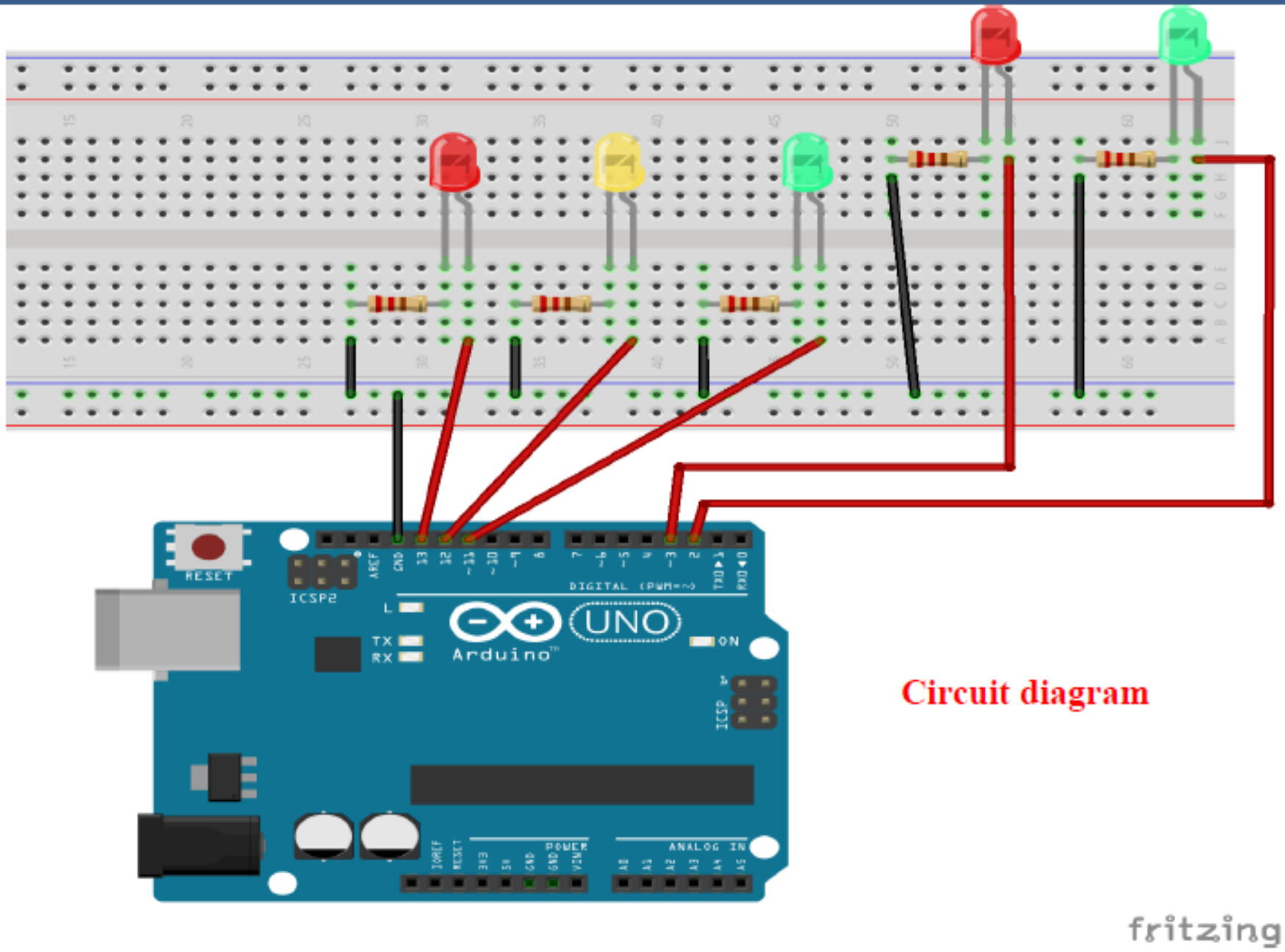
Code

```
/*The Function setup runs only once when Arduino board is first
powered up or a rest button the board is pressed */
void setup()
{
pinMode(13, OUTPUT); //pin 13 is set as an OUTPUT pin
}
//loop function iterates forever
void loop()
{
digitalWrite(13, HIGH); //Sets LED to HIGH voltage when a button is
//pressed else it remains LOW
//delay by a second
delay(1000);
}
```

Traffic light Simulation for Pedestrians

Components required	2-Red LED, 2-Green LED, 1-Yellow LED, 5-220Ω resistor, Jumper wires, Breadboard

Traffic light Simulation for Pedestrians



Circuit diagram

Traffic light Simulation for Pedestrians

Code

```
// Declare the variables for different colors of LEDs.
int red_vehicle = 13;
int yellow_vehicle = 12;
int green_vehicle = 11;
int green_Pedestrian =2;
int red_Pedestrian= 3;
void setup( )
{
// Initialize the pins for output
pinMode(red_vehicle, OUTPUT);
pinMode(yellow_vehicle, OUTPUT);
pinMode(green_vehicle, OUTPUT);
pinMode(red_Pedestrian, OUTPUT);
pinMode(green_Pedestrian, OUTPUT);
}
```

Traffic light Simulation for Pedestrians

```
void loop( )
{
digitalWrite(green_Vehicle, HIGH); // green LED turns ON
digitalWrite(red_Pedestrian, HIGH);
delay(5000);
digitalWrite(green_Vehicle, LOW); // green LED turns OFF
digitalWrite(yellow_Vehicle, HIGH); // Yellow LED turns ON for 2second.
delay(2000);
digitalWrite(yellow_Vehicle, LOW); // yellow LED will turn OFF
digitalWrite(red_Pedestrain, LOW);
digitalWrite(red_Vehicle, HIGH); // Red LED turns ON for 5 seconds
digitalWrite (green_Pedestrian, HIGH);
delay(5000);
digitalWrite(red_Vehicle, LOW); // Red LED turns OFF
digitalWrite(green_Pedestrian, LOW);
}
```

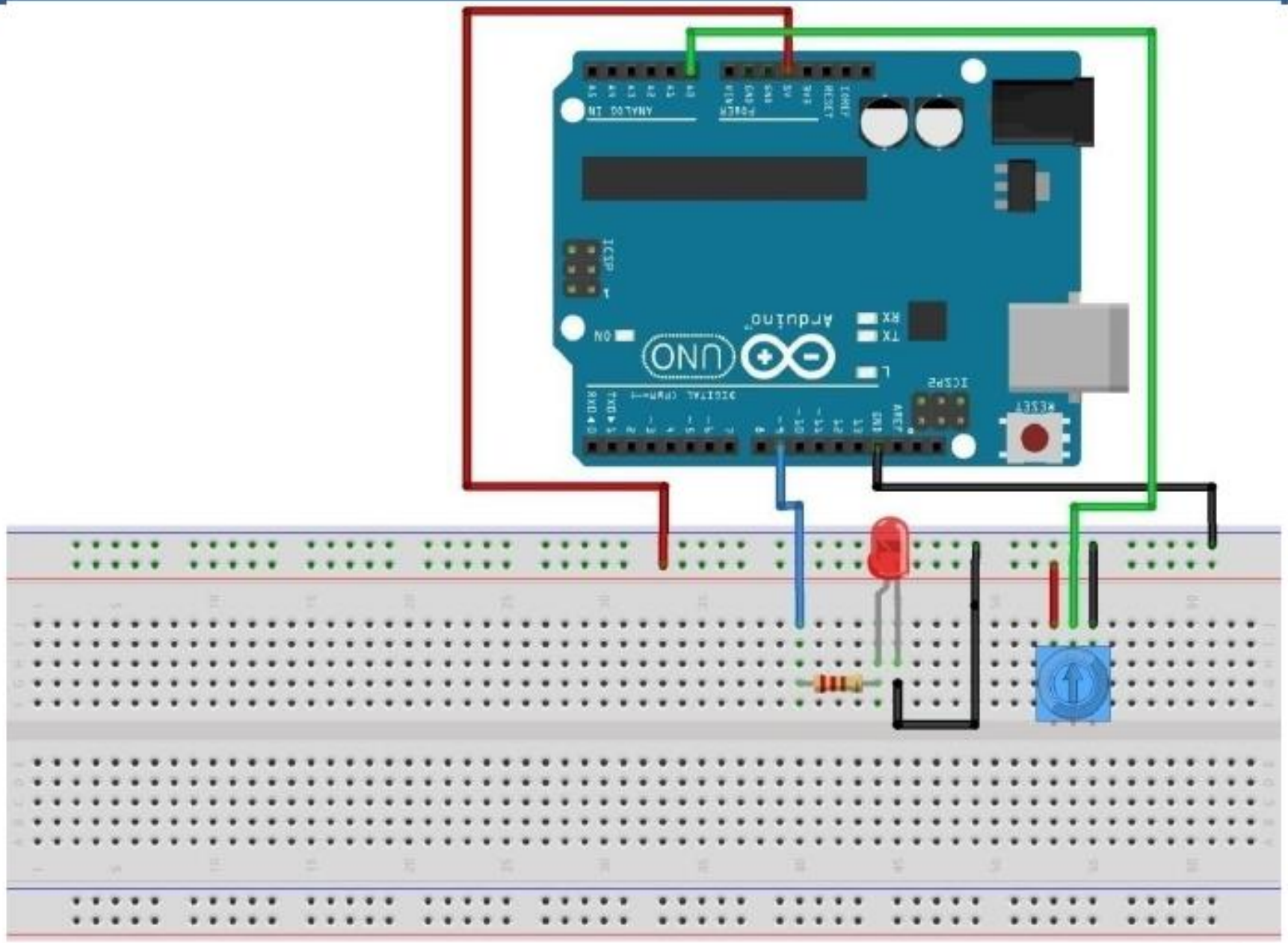
Creating a Dimmable LED using Potentiometer

Components Required	1-LED, 220Ω resistor, 1-Potentiometer, Jumper wires, Breadboard

In this program we dim the LED based on the value read from the potentiometer. A "**0**" value from potentiometer is a "**0V**" and a value "**1023**" from potentiometer is a "**5V**", which means we need to write a value of **255**. Hence we need to scale our read values from the potentiometer which falls between 0 to 1023 to suitable write values to be between 0 to 255 using the below given formulae.

write value=(255/1023)* read_value

Creating a Dimmable LED using Potentiometer



Creating a Dimmable LED using Potentiometer

```
//Declaring the pins corresponds to an LED-to pin 9 and a Potentiometer- to
//pinA0
int pot_Pin= A0;
int LED_Pin= 9;
int read_Value;  // To store the value read by potentiometer
int write_Value; // To write the value to LED
void setup( )
{ pinMode(pot_Pin, INPUT);
  pinMode(LED_Pin, OUTPUT);
  Serial.begin(9600);       }
void loop( )
{ read_Value = analogRead(pot_Pin);  //Potentiometer reading
 write_Value = (255./1023.) * readValue; //Write value for LED is calculated
 analogWrite(LEDPin, writeValue);      //Write to the LED
 Serial.print("The writing vlues to the LED is ");  //Debugging purpose
 Serial.println(write_Value); }
```

Programs to interact with Serial Monitor of our Computer Screen

To print the status of our computer Screen

Now, let's introduce the interaction with the **Serial monitor**. In this program we perform Arithmetic operations on the variables defined in the program, variables are initialized inside the program. Serial monitor communication will be processed when we call the method **Serial.begin( )** with appropriate **Baud rate**. Serial monitor displays the desired message of a program using the method **Serial.print( )** method.

Syntax:

Serial.begin(speed) /* to communicate between your computer and Serial monitor */

Serial.begin(speed, config)

Serial.print() #To print desired message on the Serial monitor

Programs to interact with Serial Monitor of our Computer Screen

```
//In this program we compute basic aritmetic operations to print the result on
//to the Serial monitor.
int a = 5, b = 10, c = 20;
void setup( )                    // run once, when the sketch starts
{ Serial.begin(9600);            // set up Serial library at 9600 bps
  Serial.println("Here is some math: ");
  Serial.print("a = ");
  Serial.println(a);
  Serial.print("b = ");
  Serial.println(b);
  Serial.print("c = ");
  Serial.println(c);
```

Programs to interact with Serial Monitor of our Computer Screen

```
Serial.print("a + b = ");       // add
  Serial.println(a + b);
  Serial.print("a * c = ");       // multiply
  Serial.println(a * c);
  Serial.print("c / b = ");       // divide
  Serial.println(c / b);
  Serial.print("b - c = ");       // subtract
  Serial.println(b - c);
}
void loop( )  {  }
```

Interfacing Sensors to the Arduino

- **Temperature Sensor**
- **Light Sensor**
- **Ultrasonic distance sensor**
- **Line sensor** (infrared).

Interfacing Temperature Sensor

Components Required	Buzzer, LM35 Temperature Sensor, Jumper wires, Breadboard

LM35 Temperature Sensor:

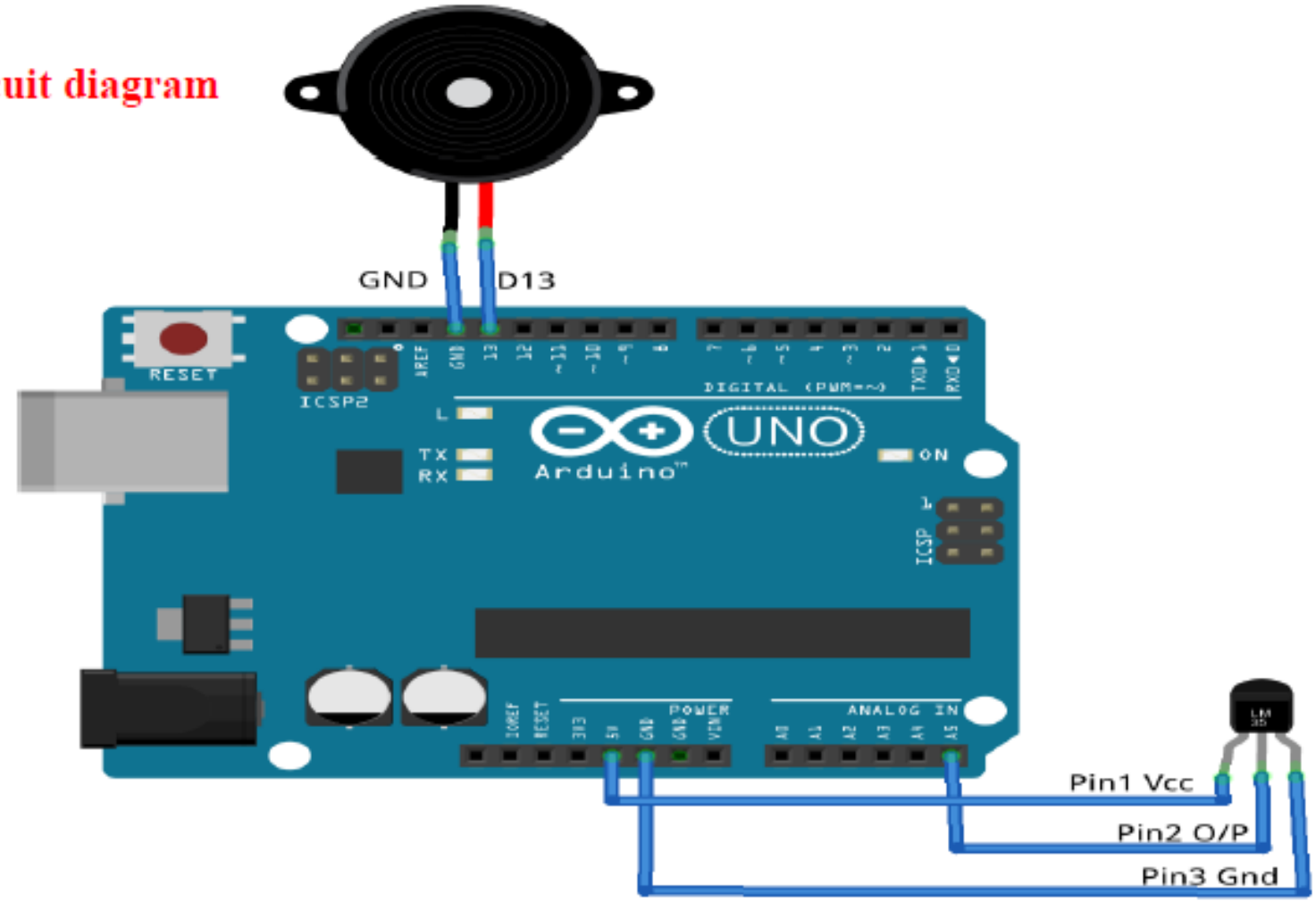
The LM35 series are the gadgets with precision integrated circuit temperature whose yield voltage falls directly corresponding to the Centigrade temperature.

- Calibrated Directly in Celsius (Centigrade)
- Operates from 4 V to 30 V
- Ranges are evaluated from Full −55°C to 150°C.
- Suitable for Remote Applications
- Used in Battery Management

Pin No	Function	Name
1	Supply voltage; 5V (+35V to -2V)	Vcc
2	Output voltage (+6V to -1V)	Output
3	Ground (0V)	Ground

Interfacing Temperature Sensor

Circuit diagram



Interfacing Temperature Sensor

```
//initialize a variable temPin to Analog pin A%
int temPin = A5;
//Set buzzer to pin 13 as OUTPUT
int buzzer = 13;
//Variable to store the temperature read
int value;
void setup()
{
//Initialize Serial baud rate to 9600
Serial.begin(9600);
//sets buzzer as an OUTPUT
pinMode(buzzer, OUTPUT);
}
```

Interfacing Temperature Sensor

```
void loop()
{
//Read temperature value on pin A5 by analogRead() method
value = analogRead(temPin);
//Conversion of temperature value read
float mvalue = ( value/1024.0)*5000;
//Conversion of Temperature to celsius
float celsius = mvalue/10;
//conversion of temperature to Fahrenheit
float fahrenheit = (celsius*9)/5 + 32;
//print the celsius value onto the serial monitor
Serial.print(cel);
//check if the read temperature is greater than 32 degree celsius
if(cel>32)
{
//trigger HIGH value on buzzer
digitalWrite(buzzer, HIGH);
delay(1000);
```

Interfacing Temperature Sensor

```
// trigger LOW value on buzzer
digitalWrite(buzzer, LOW);
//delay for 2 second
delay(2000);
//trigger HIGH value on buzzer
digitalWrite(buzzer, HIGH);
//delay for 1 second
delay(1000);
// trigger LOW value on buzzer
digitalWrite(buzzer, LOW);
//delay for 2 second
delay(2000);
}
//Print the temperature onto a serial monitor
Serial.print("TEMPRATURE = ");
Serial.print(cel);
Serial.print("*C");
Serial.println(); }
```

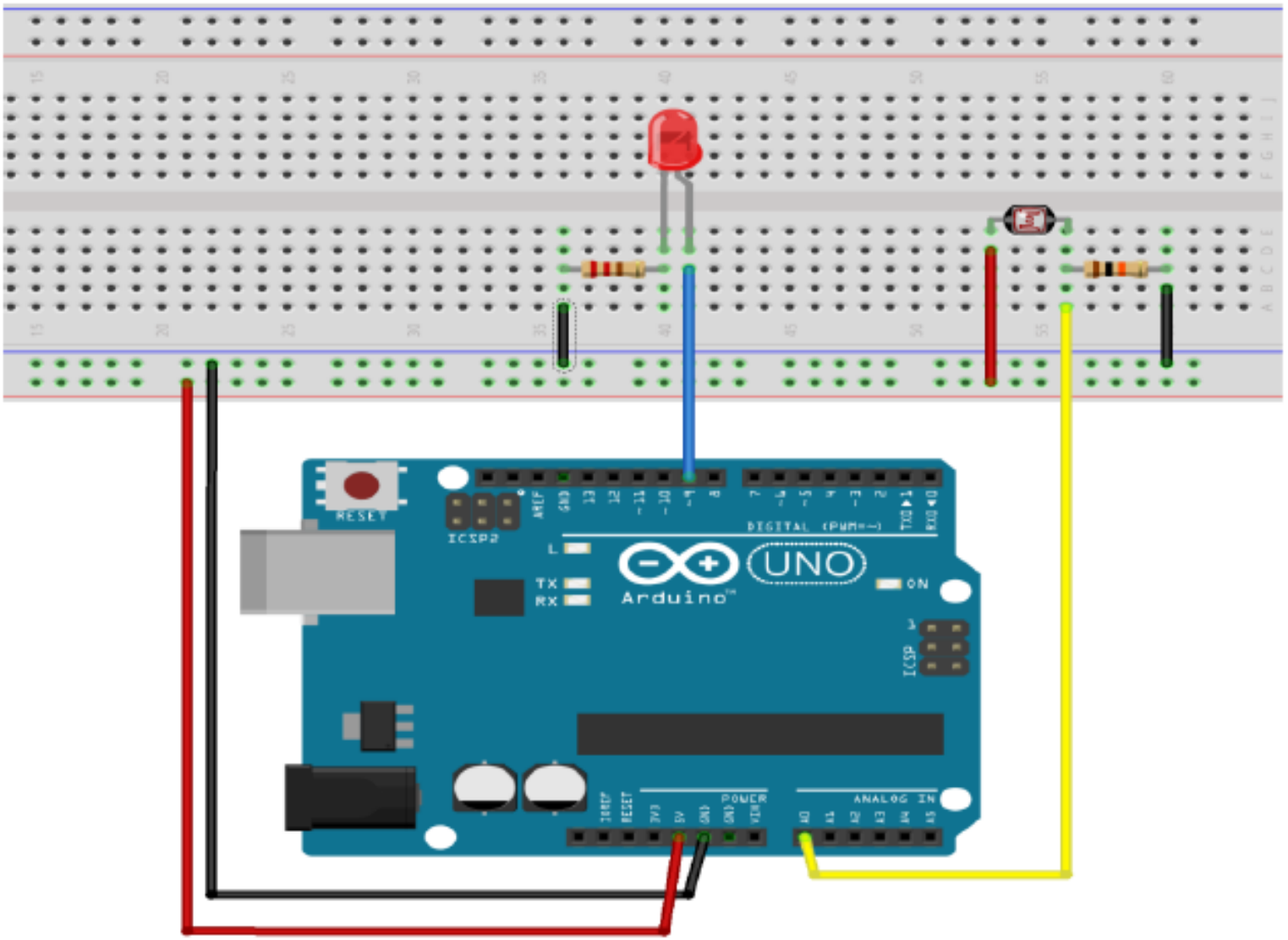
Automatic lights with light sensor

Components Required	1x LED , 1x 220Ω resistor , 1x photoresistor , 1x 10kΩ resistor, Jumper wires, Breadboard

A **photoresistor** is a light-dependent resistor. The resistance of a photoresistor decreases with increasing of light intensity. So:

- When there is light, the resistance decreases, we will have more current flowing.
- When there is no light, the resistor increases, we will have less current flowing.

Automatic lights with light sensor



Automatic lights with light sensor

```
int led_Pin = 9;
int led_Brightness = 0;
int sensor_Pin = A0;
int sensor_Value = 0;
void setup(void) {
pinMode(led_Pin, OUTPUT);
// Send some information to Serail  monitor
Serial.begin(9600);
}
```

Automatic lights with light sensor

```
void loop(void) {
sensor_Value = analogRead(sensor_Pin);
Serial.print("Sensor reading: ");
Serial.println(sensor_Value);
// LED gets brighter the darker it is at the
sensor
// that means we have to -invert- the reading
from 0-1023 back to 1023-0
sensorValue = 1023 - sensorValue;
//now we have to map 0-1023 to 0-255 since
thats the range analogWrite //uses
ledBrightness = map(sensorValue, 0, 1023, 0,
255);
analogWrite(ledPin, ledBrightness);
delay(50);
}
```

To Measure Speed of Sound using Ultrasonic Sensor

Components Required	1- HC-SR04 -ultrasonic sensor, Jumper wires, Breadboard

Working of Ultrasonic sensor?

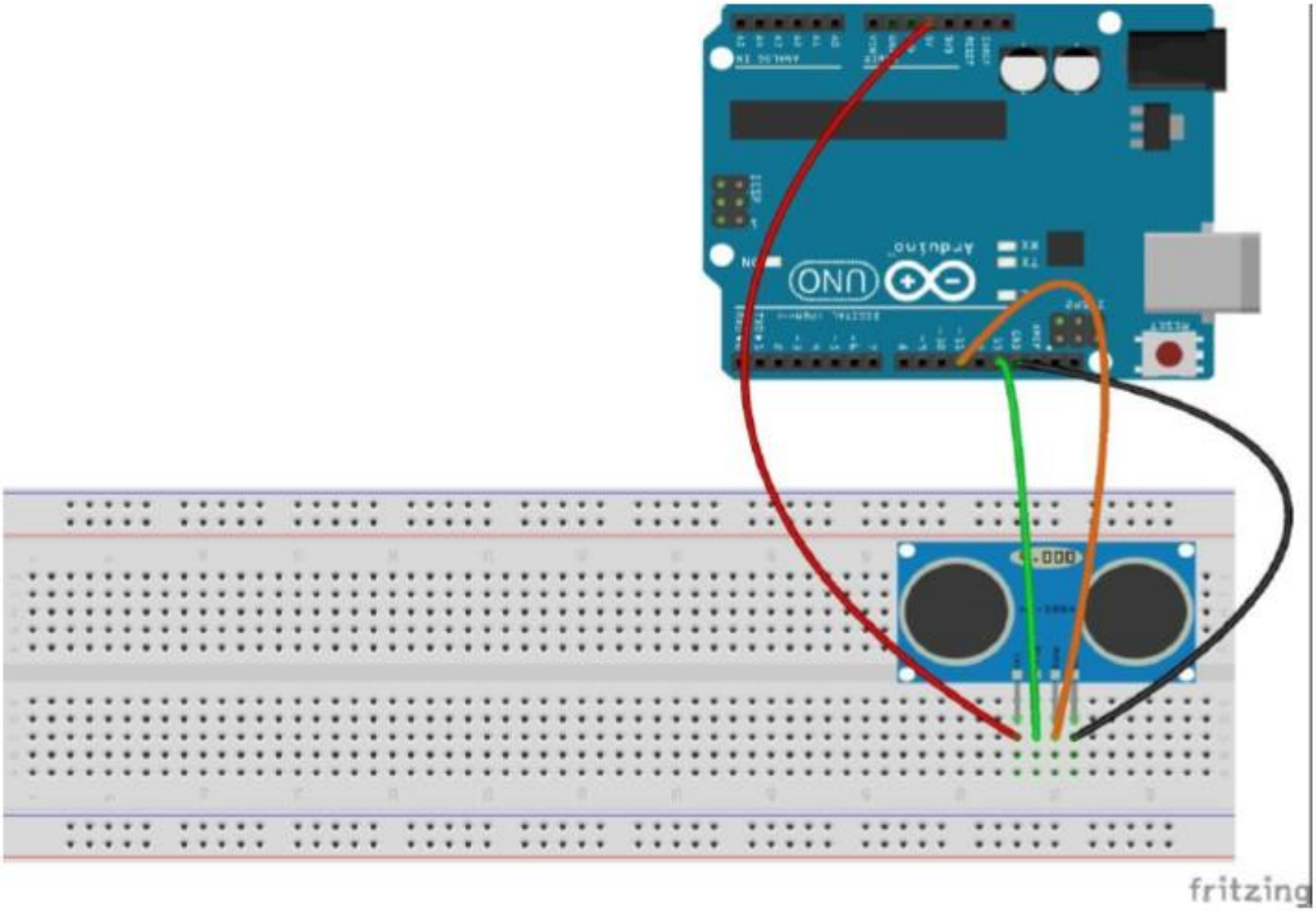
- Trigger LOW-HIGH-LOW sequence on the pin which creates a high pitched ultrasonic tone which sent out from the sensor, which will go out and bounce off the first thing in front of it and back to the sensor.
- The sensor will output HIGH on the pin and length of pulse in microseconds indicates time it took the ping to travel to target and return.
- Measure the length of the pulse using pulseIn command.
- Calculate the speed of sound by

  distance= rate * time

  rate = time/distance
- convert this to miles per hour as follows:

  (rate in inches/mircrosecond)*(1000000 microsecond/second)*
(3600 seconds/hour)*(1 mile/63360 inches)

To Measure Speed of Sound using Ultrasonic Sensor



To Measure Speed of Sound using Ultrasonic Sensor

```
int trig_Pin=13; //Connect Trip pin of sensor to
13 pin of Arduino
int echo_Pin=11;  //Connect sensor echo pin to
11 pin of Arduino
float pinging_Time;
float speed_Of_Sound;
int  target_Distance=6;  //Target  distance  in
inches
void setup() {
  Serial.begin(9600);
  pinMode(trig_Pin, OUTPUT);
  pinMode(echo_Pin, INPUT);
}
```

To Measure Speed of Sound using Ultrasonic Sensor

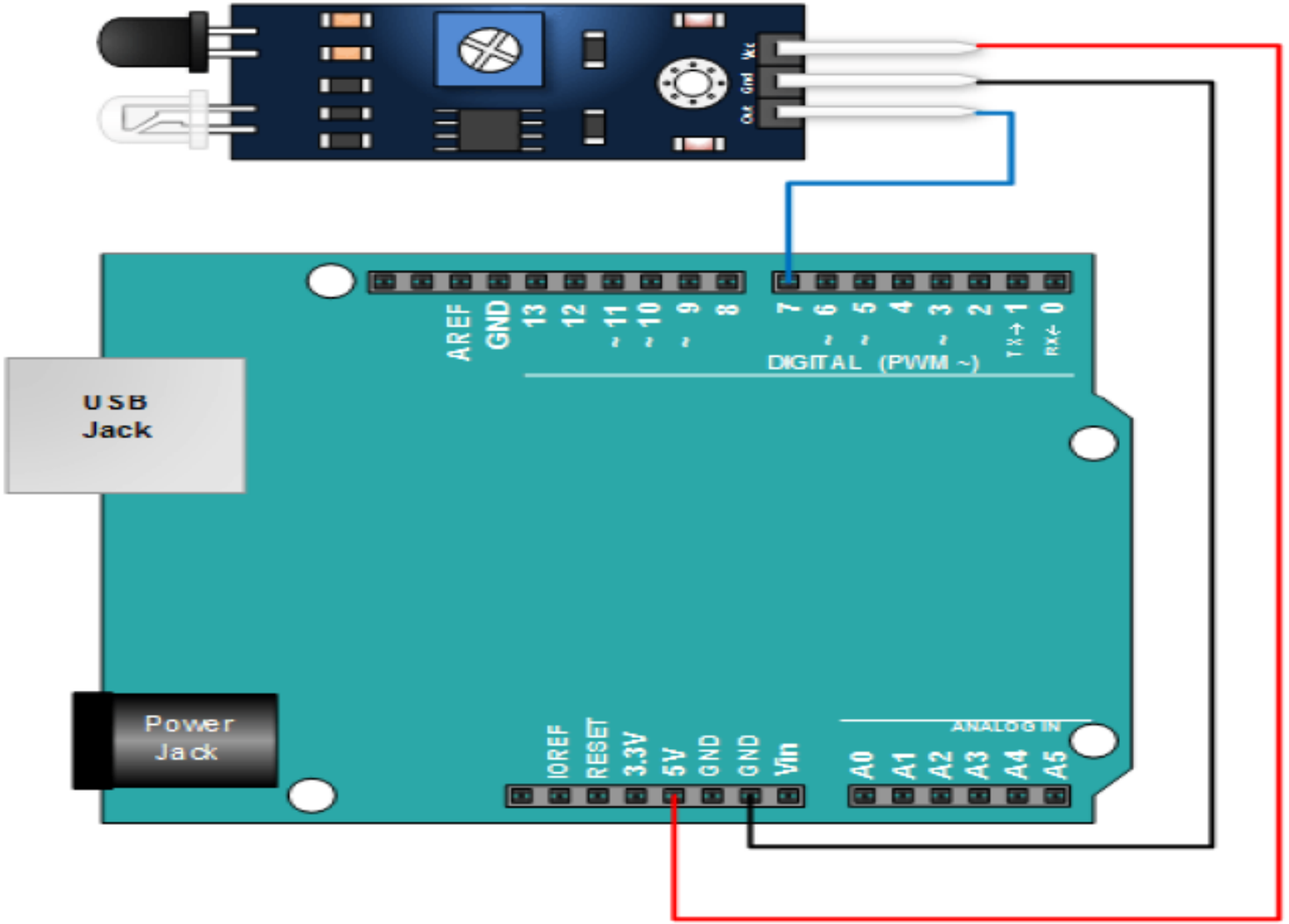
```
void loop() {
  digitalWrite(trig_Pin, LOW); //trigpin set to LOW
  delayMicroseconds(2000);
  digitalWrite(trig_Pin, HIGH); //trigPin to high
  delayMicroseconds(10);
  digitalWrite(trig_Pin, LOW); //Send ping
  pingTime = pulseIn(echo_Pin, HIGH);  /*pingTime is presented
in microceconds */
  speedOfSound                                                    =
(targetDistance*2)/pinging_Time*(1000000)*3600/63360;
//converts to miles per hour
  Serial.print("The Speed of Sound is: ");
  Serial.print(speed_Of_Sound);
  Serial.println(" miles per hour");
  delay(1000);
}
```

Obstacle collision module using IR(Infrared) sensor

Components Required	1-IR sensor, Jumper wire, Breadboard

An **Infrared sensor** is an electronic instrument which is used to sense certain characteristics of its surroundings by either emitting and/or detecting infrared radiation. Infrared sensors are also capable of measuring the heat being emitted by an object and detecting motion.

Obstacle collision module using IR(Infrared) sensor



Obstacle collision module using IR(Infrared) sensor

```
// IR Obstacle Collision Detection Module

int LED = 13;
int is_Obstacle_Pin = 7;  // input pin for ostacle
int is_Obstacle = HIGH;   // value HIGH tells
there's no obstacle

void setup() {
  pinMode(LED, OUTPUT);
  pinMode(is_Obstacle_Pin, INPUT);
  Serial.begin(9600);

}
```

Obstacle collision module using IR(Infrared) sensor

```
void loop() {
  is_Obstacle = digitalRead(is_Obstacle_Pin);
  if (is_Obstacle == LOW)
  {
    Serial.println("OBSTACLE!!,
OBSTACLE!!");
    digitalWrite(LED, HIGH);
  }
  else
  {
    Serial.println("clear");
    digitalWrite(LED, LOW);
  }
  delay(200);
}
```

More Examples Refer Textbook

- **Interfacing Display, GSM, GPS to Arduino**
  - Temperature and LCD Display
  - Custom Characters in LCD
  - 7 Segment Display on Arduino
- **GSM Interface**
- **GPS Interface**
- **Interfacing Motors**
  - Servo motor

Self Test Questions

- What is a Arduino?
- Can I connect a mouse and keyboard to Arduino Uno?
- What SOC Arduino using?
- What is a SOC?
- Does Arduino Uno overclock?
- Does Arduino uno need a heat sink?
- Does Arduino Uno has any hardware interfaces?
- Does Arduino Uno need an External power source?
- Which IDE environment does Arduino Uno use?
- Does the Arduino supports networking?
- Define a Microcontroller?
- State the use of Serial Monitor in Arduino IDE?
- Define the term Baud Rate?

Review Questions

- How is Arduino Uno is different from the other available Microcontrollers?
- What is the use of GPIO pins?
- What is the use of I2C interfaces on Raspberry Pi?
- How many pins does the Atmega328P MCU used on the standard Arduino have? Over what range of voltages will it operate?
- Assume that you have an LED connected to each of the 14 digital-only I/O pins on the Arduino.
- If all of the LEDs could possibly be on at the same time, what must the current be limited to through each of the LEDs?
- Assume that a project requires that a high-brightness LED be on any time that the Arduino is powered-on, and that this LED requires 350mA. What is the best way to supply power/current to this LED?