

What is Internet Of Things(IoT)?

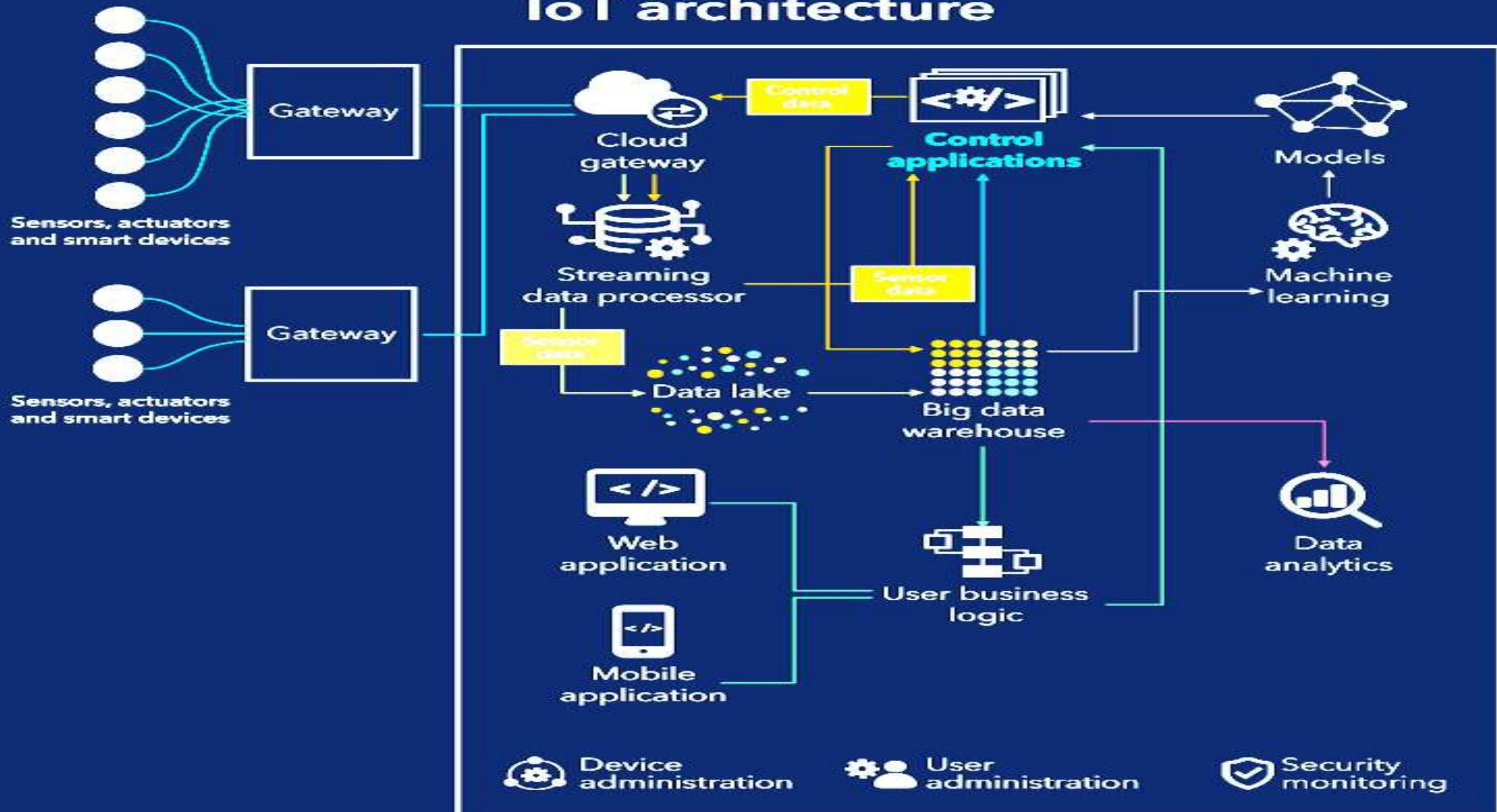
Internet of Things(IoT) is a network of physical objects or "things" that are embedded with software, electronics, network, and sensors that allows these objects to collect and exchange data.

The goal of IoT is to extend internet connectivity from standard devices like computer, mobile, tablet to relatively dumb devices like a toaster.

IoT makes virtually everything "smart," by improving aspects of our life with the power of data collection, AI algorithm, and networks.



IoT architecture



IOT Applications



TinkerCad

Introduce you to the basics of Tinkercad Circuits.

- which was a free and easy to use breadboard simulator.
- Like Fritzing, is a great design resource for makers.
- Education for introducing students to electronics and coding in a virtual low-effort, low-risk and low-cost environment.
- Even generate their actual Arduino code.

component library -

- input, output, power, Breadboards,
- Microcontrollers - **Arduino Uno R3** and an **ATtiny**
- Instruments - **multimeter**, a **power supply**, a **function generator** and an **oscilloscope**.
- Integrated Circuits – such as **timers**, an **op-amp** and **comparators**

What Is Arduino?

- Arduino is an open source programmable circuit board that can be integrated into a wide variety of makerspace projects both simple and complex.
- This board contains a microcontroller which is able to be programmed to sense and control objects in the physical world. By responding to sensors and inputs, the Arduino is able to interact with a large array of outputs such as LEDs, motors and displays.
- Because of it's flexibility and low cost, Arduino has become a very popular choice for makers and makerspaces looking to create interactive hardware projects.

What is the Arduino

The word “Arduino” can mean 3 things

A physical piece
of hardware



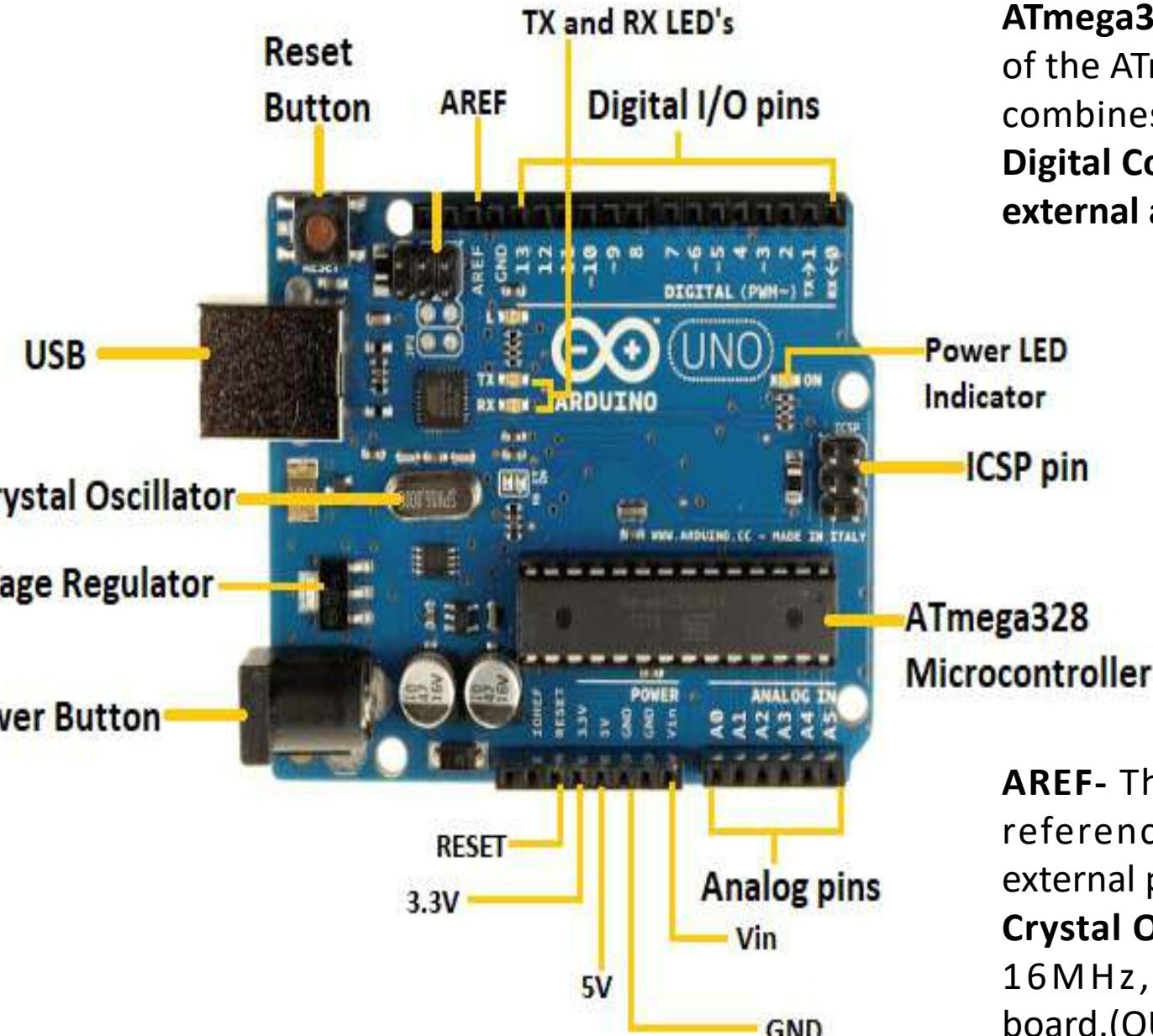
A programming
environment

A screenshot of the Arduino IDE interface. The title bar says "Arduino - 0010 Alpha". The main window shows a code editor with the following sketch:

```
int ledPin = 13; // LED connected to digital pin 13
void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
}
void loop() {
  digitalWrite(ledPin, HIGH); // sets the LED on
  delay(1000); // waits for 1 second
  digitalWrite(ledPin, LOW); // sets the LED off
  delay(1000); // waits for a second
}
```

A community
& philosophy

A screenshot of the Arduino playground website. At the top, there's a header with the text "Arduino playground". Below the header, there's a large image of a person's hands holding an Arduino Uno microcontroller board. To the right of the image is a sidebar with various links and sections. One section is titled "About the Arduino Playground ::" which contains text about the playground's purpose and how to contribute. Another section is titled "RoadMap: What Needs to be Done? ::" which lists items for improvement. The sidebar also includes links to "About", "Help", "Contact", and "Feedback".



Arduino UNO

ATmega328 Microcontroller- is a single chip Microcontroller of the ATmel family. The processor code inside it is of 8-bit. It combines **Memory (SRAM, EEPROM, and Flash)**, **Analog to Digital Converter**, **SPI serial ports**, **I/O lines**, **registers**, **timer**, **external and internal interrupts**, and **oscillator**.

ICSP pin - The In-Circuit Serial Programming pin allows the user to program using the firmware of the Arduino board.

Digital I/O pins- The digital pins D0 - D13 have the value HIGH or LOW.

Analog Pins- The pins A0 to A5 are analog pins.

AREF- The Analog Reference (AREF) pin is used to feed a reference voltage to the Arduino UNO board from the external power supply.

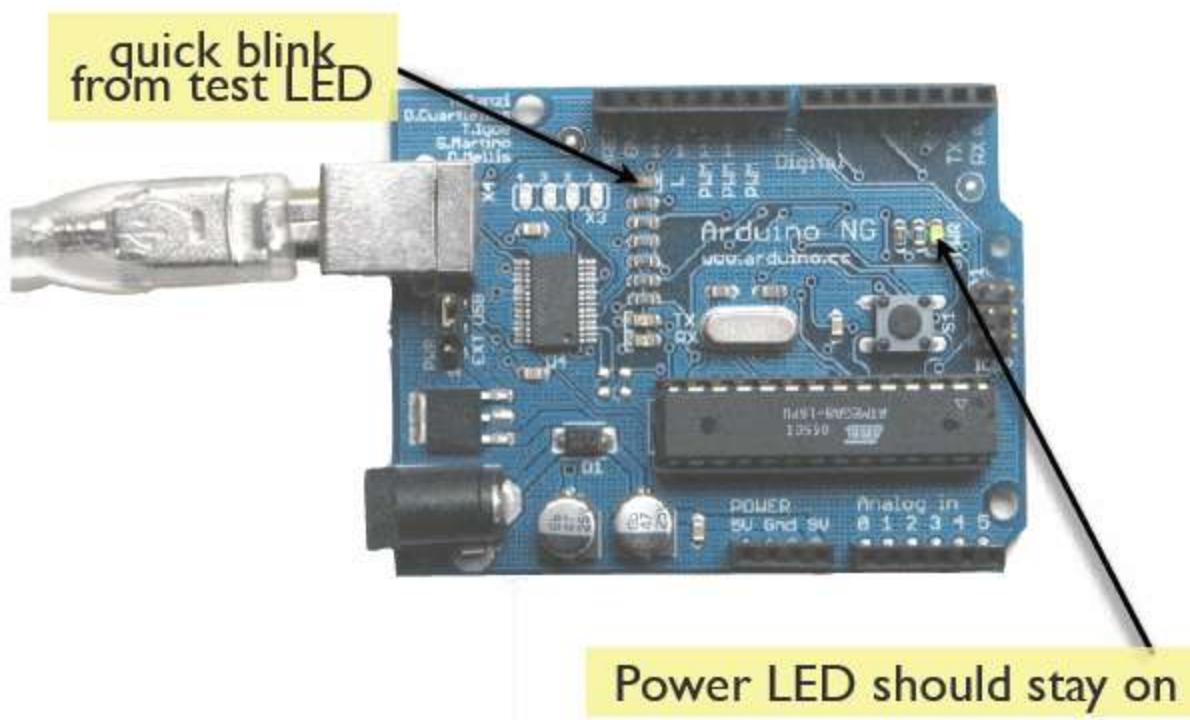
Crystal Oscillator- The Crystal oscillator has a frequency of 16MHz, which makes the Arduino UNO a powerful board.(OUTPUT STABLE FOR LONGER PERIOD)

Voltage Regulator- The voltage regulator converts the input voltage to 5V.

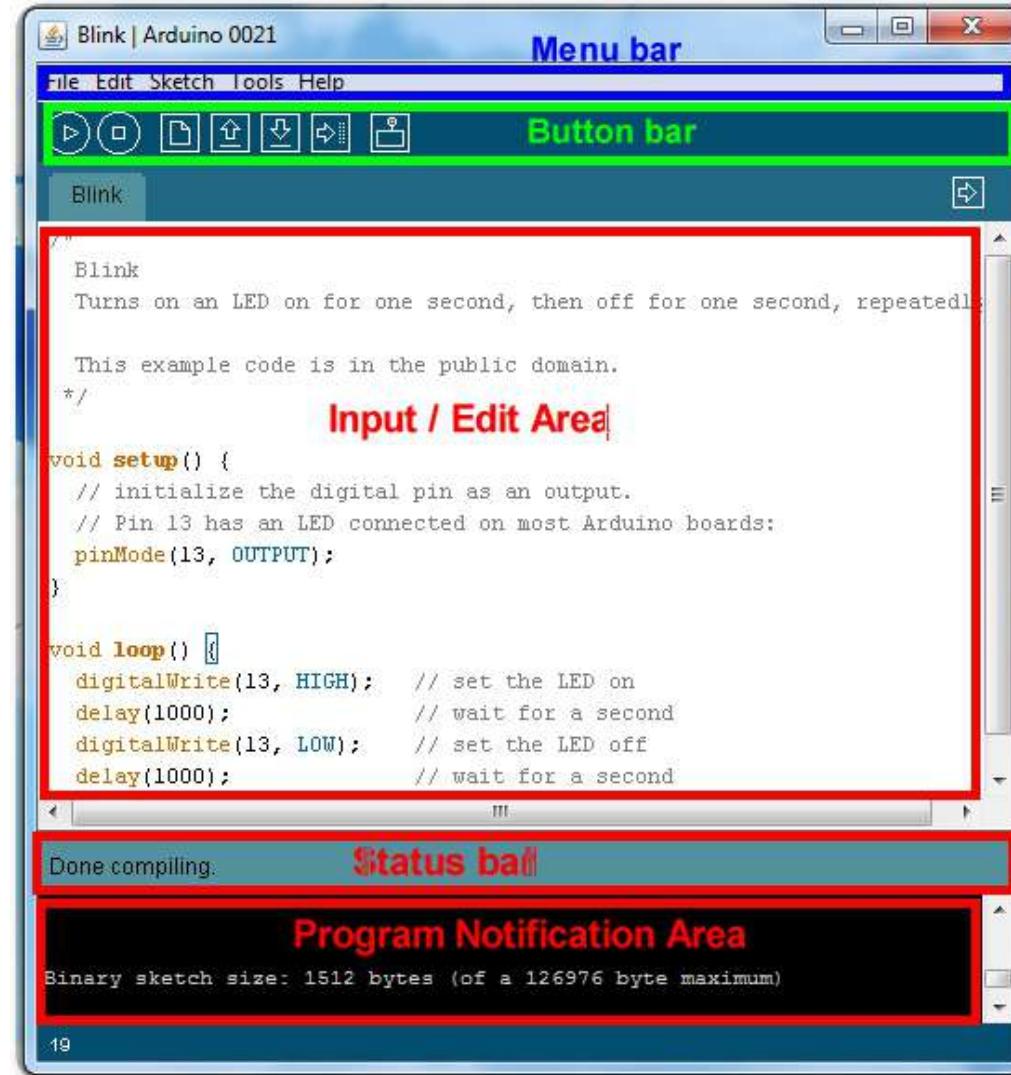
Getting Started

- Check out: <http://arduino.cc/en/Guide/HomePage>
 1. Download & install the Arduino environment (IDE)
 2. Connect the board to your computer via the UBS cable
 3. If needed, install the drivers (not needed in lab)
 4. Launch the Arduino IDE
 5. Select your board
 6. Select your serial port
 7. Open the blink example
 8. Upload the program

Try It: Connect the USB Cable

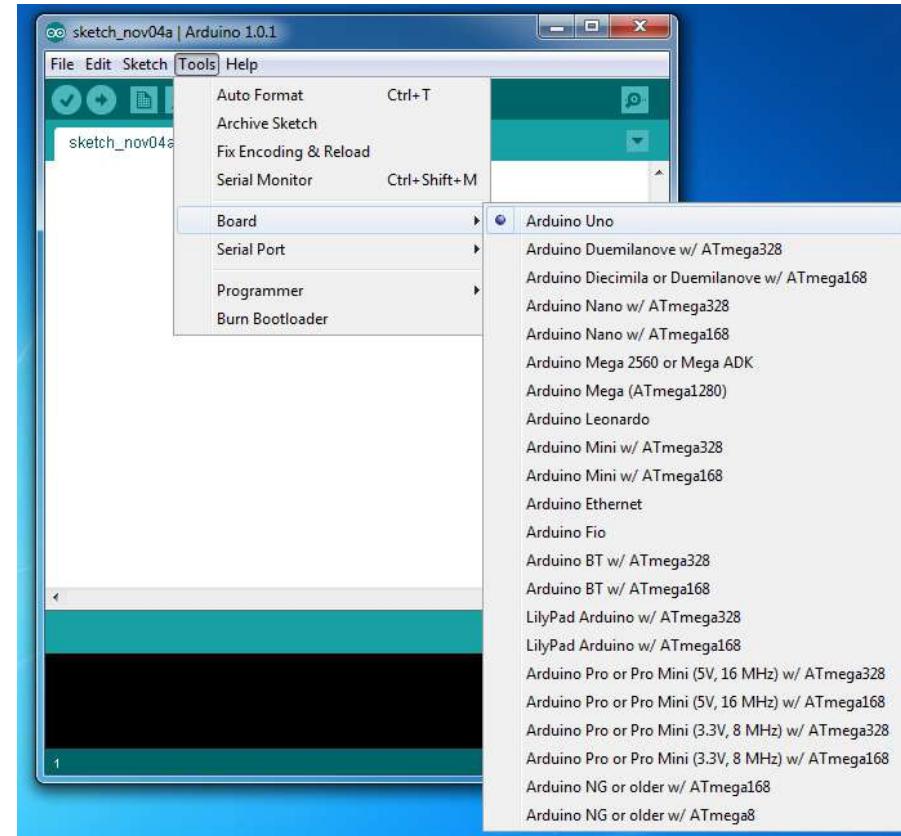
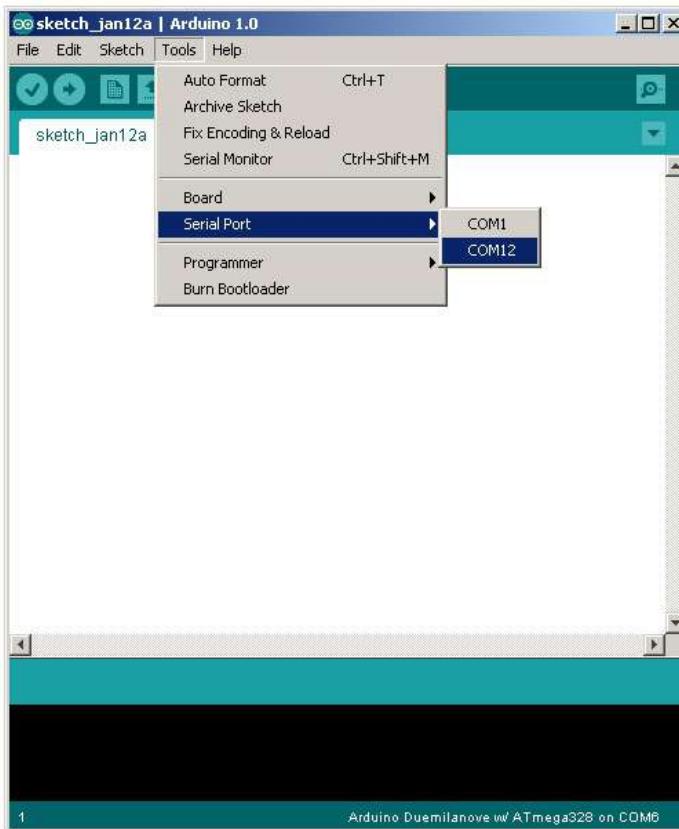


Arduino IDE



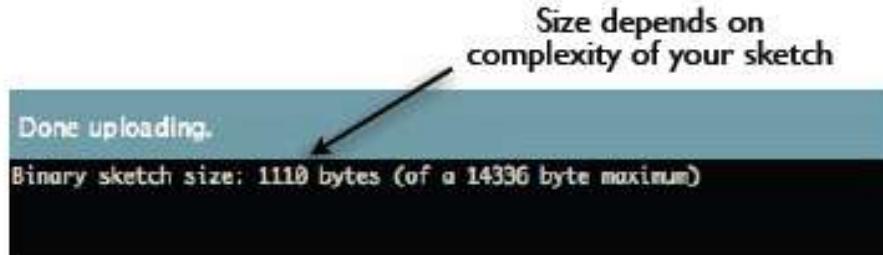
See: <http://arduino.cc/en/Guide/Environment> for more information

Select Serial Port and Board



Status Messages

Uploading worked

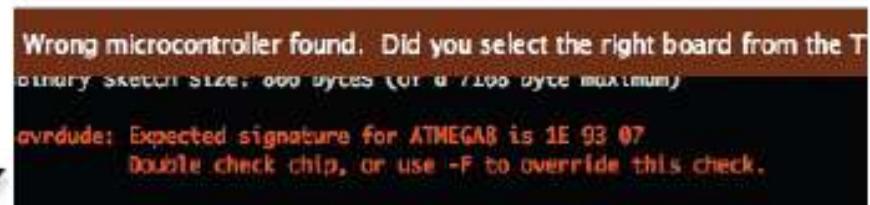


Wrong serial port selected



Wrong board selected

nerdy cryptic error messages

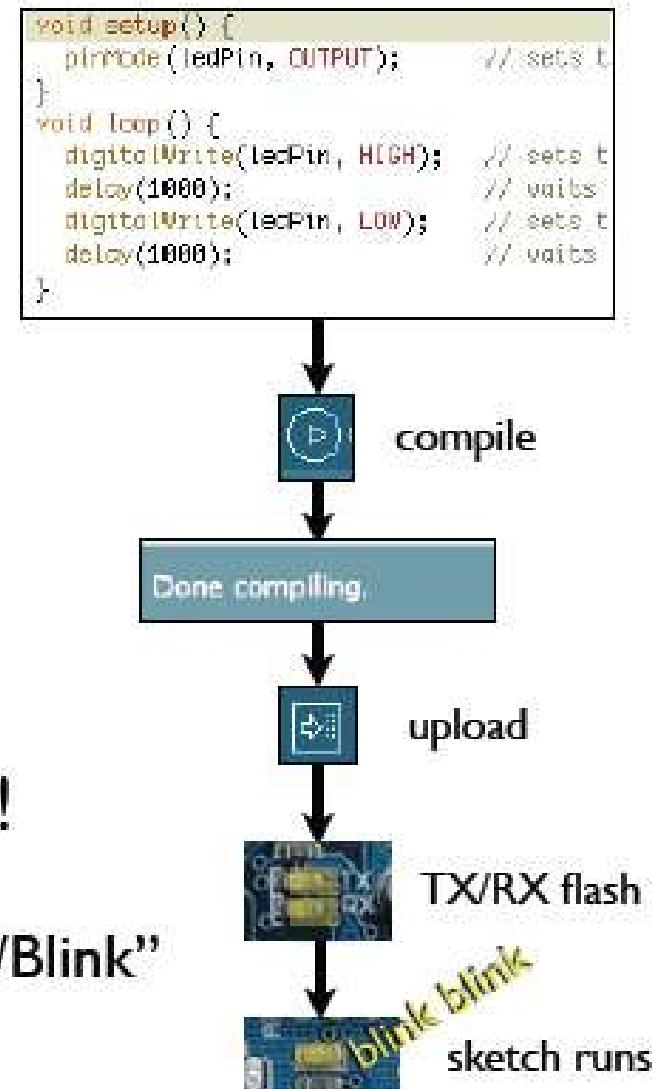


Using Arduino

- Write your sketch
- Press Compile button
(to check for errors)
- Press Upload button to program
Arduino board with your sketch

Try it out with the “Blink” sketch!

Load “File/Sketchbook/Examples/Digital/Blink”



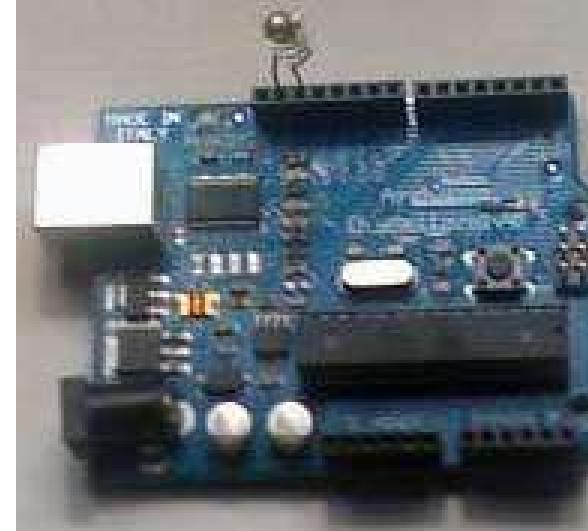
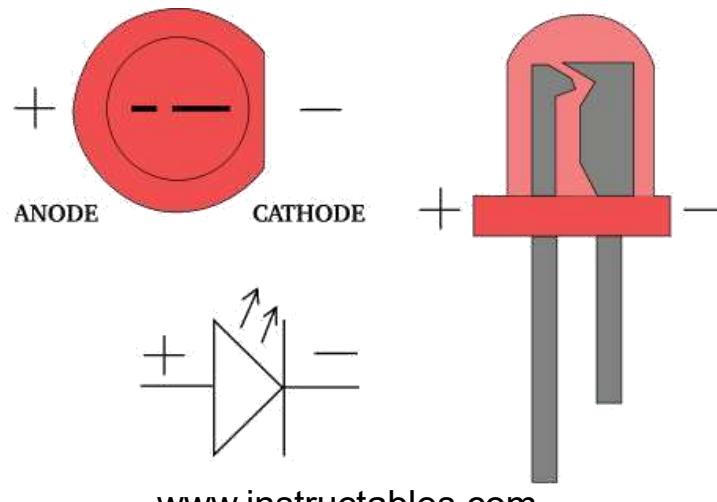
A Little Bit About Programming



- Code is case sensitive
- Statements are commands and must end with a semi-colon
- Comments follow a // or begin with /* and end with */
- loop and setup

Add an External LED to pin 13

- **File > Examples > Digital > Blink**
- LED's have polarity
 - Negative indicated by flat side of the housing and a short leg



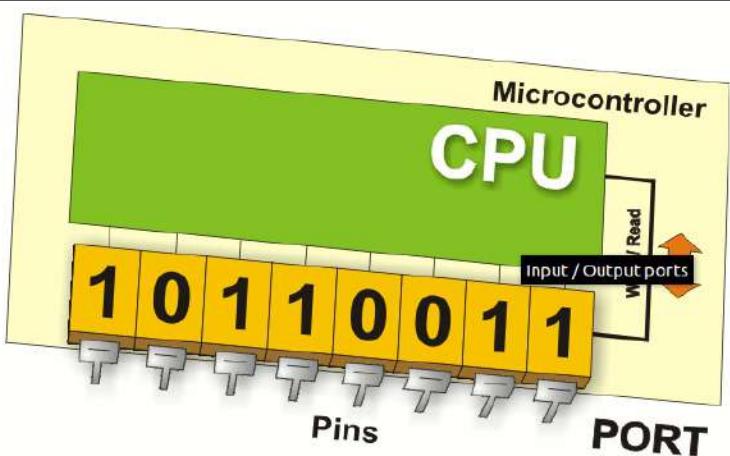
Terminology

“*sketch*” – a program you write to run on an Arduino board

“*pin*” – an input or output connected to something.
e.g. output to an LED, input from a knob.

“*digital*” – value is either HIGH or LOW.
(aka on/off, one/zero) e.g. switch state

“*analog*” – value ranges, usually from 0-255.
e.g. LED brightness, motor speed, etc.



www.mikroe.com/chapters/view/1

Digital I/O

`pinMode(pin, mode)`

Sets pin to either INPUT or OUTPUT

`digitalRead(pin)`

Reads HIGH or LOW from a pin

`digitalWrite(pin, value)`

Writes HIGH or LOW to a pin

Electronic stuff

Output pins can provide 40 mA of current

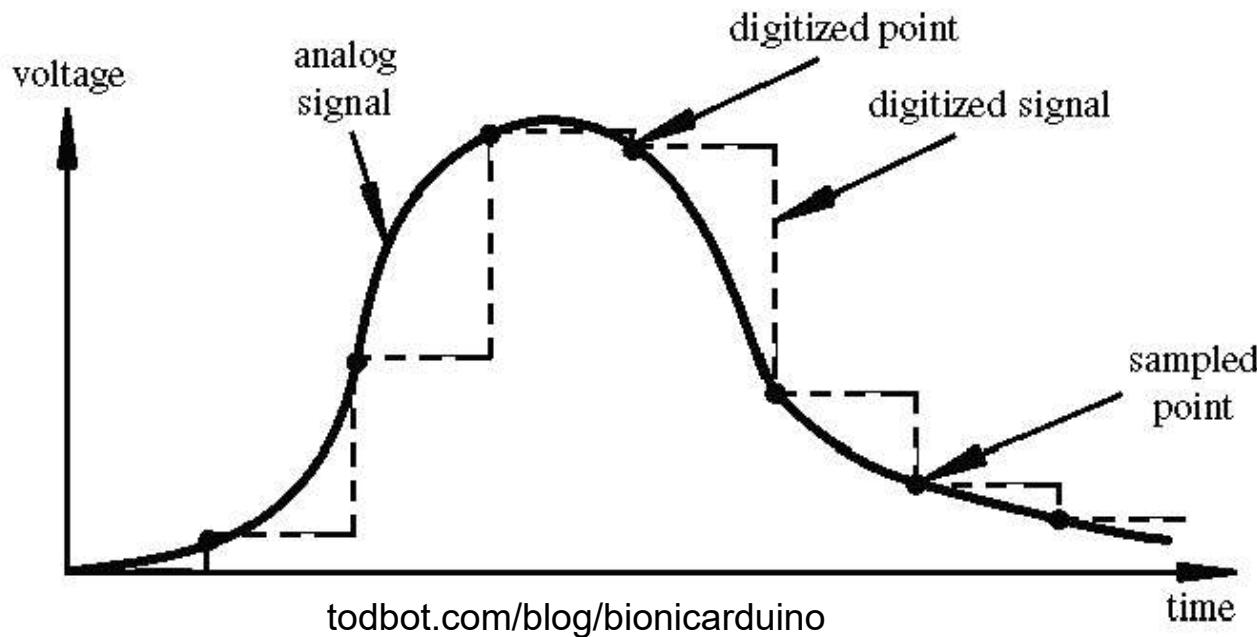
Writing HIGH to an input pin installs a 20KΩ pullup

Arduino Timing

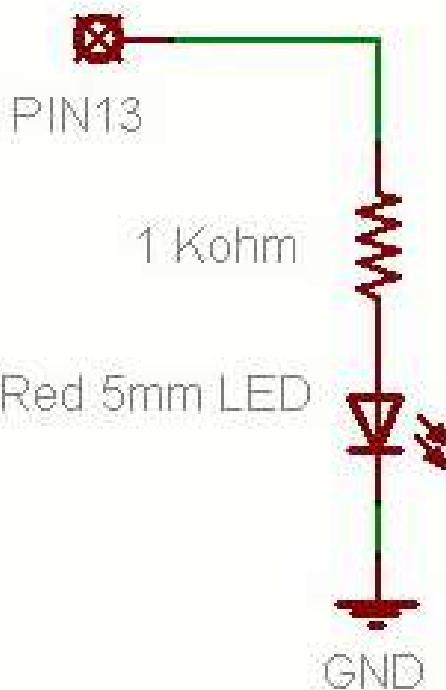
- `delay (ms)`
 - Pauses for a few milliseconds
- `delayMicroseconds (us)`
 - Pauses for a few microseconds
- More commands:
arduino.cc/en/Reference/HomePage

Digital? Analog?

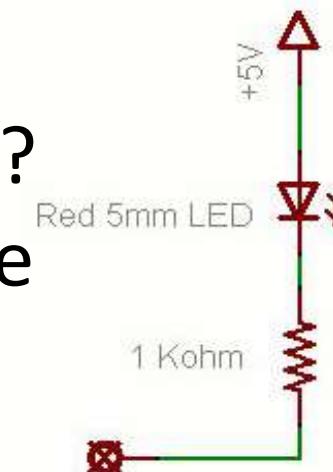
- Digital has two values: **on** and **off**
- Analog has many (infinite) values
- Computers don't really do analog, they **quantize**
- Remember the 6 analog input pins---here's how they work



Putting It Together



- Complete the sketch (program) below.
- What output will be generated by this program?
- What if the schematic were changed? →



```
void loop() // run over and over again
{
    digitalWrite(ledPin, HIGH); // sets the LED on
    delay(500); // waits for a second
    digitalWrite(ledPin, LOW); // sets the LED off
    delay(500); // waits for a second
}
```

How To Program Arduino

Once the circuit has been created on the breadboard, you'll need to upload the program (known as a sketch) to the Arduino.

The sketch is a set of instructions that tells the board what functions it needs to perform.

An Arduino board can only hold and perform one sketch at a time.

The software used to create Arduino sketches is called the IDE which stands for Integrated Development Environment.

Arduino Programming Language

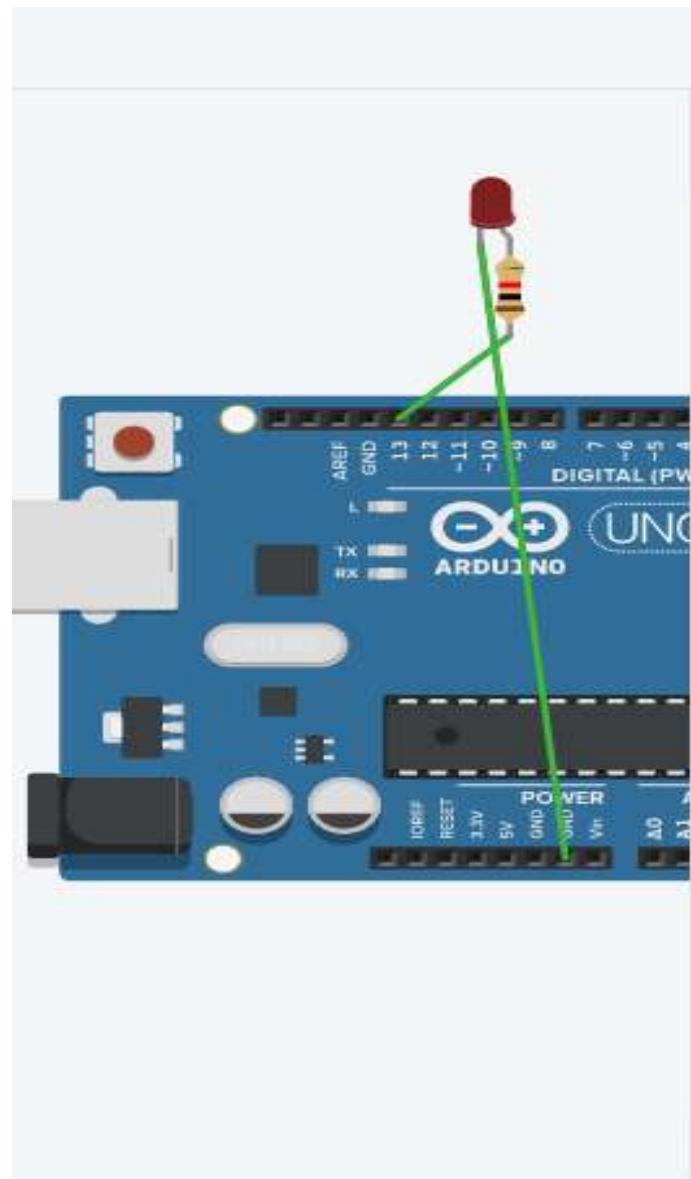
Arduino programming language can be divided in three main parts:

- FUNCTIONS - For controlling the Arduino board and performing computations.
- VARIABLES - Arduino data types and constants.
- STRUCTURE - The elements of Arduino (C++) code.

Every Arduino sketch has two main parts to the program:

void setup() – Sets things up that have to be done once and then don't happen again.

void loop() – Contains the instructions that get repeated over and over until the board is turned off.



The image shows an Arduino Uno microcontroller board. A red LED is connected to digital pin 13. The LED is connected in series with a 220 ohm resistor. One end of the resistor is connected to pin 13, and the other end is connected to ground. The Arduino Uno board has a blue PCB with various components like the ATmega328P microcontroller, crystal oscillator, and power regulation circuitry. The board is labeled "ARDUINO" and "UNO".

Code Editor

Text

Code

Start Simulation

Export

1 (Ardu)

```
1 void setup()
2 {
3     pinMode(13, OUTPUT);
4 }
5
6 void loop()
7 {
8     digitalWrite(13, HIGH);
9     delay(1000); // Wait for 1000 millisecond(s)
10    digitalWrite(13, LOW);
11    delay(1000); // Wait for 1000 millisecond(s)
12 }
```

Serial Monitor

Serial Library

Used for communication between the Arduino board and a computer or other devices. All Arduino boards have at least one serial port (also known as a UART or USART), and some have several.

As the name implies serial communication means sending and receiving data bit by bit over a single line. Arduino uno board has one serial port at digital pins 0(RX) and 1(TX) to communicate with other external serial devices or with computer through USB cable.

The process of sending and receiving data can be observed by flashing of TX and RX LED's on the arduino board.

The baud rate specifies how fast the data is sent over the serial line or in simple terms, the speed of serial communication. Some common rates for UART are 9600 baud, 11520 baud etc.

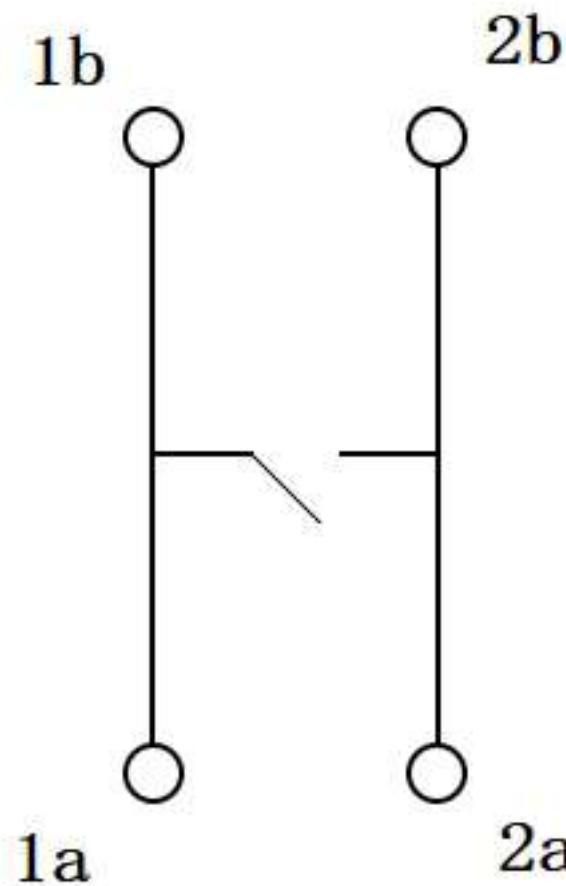
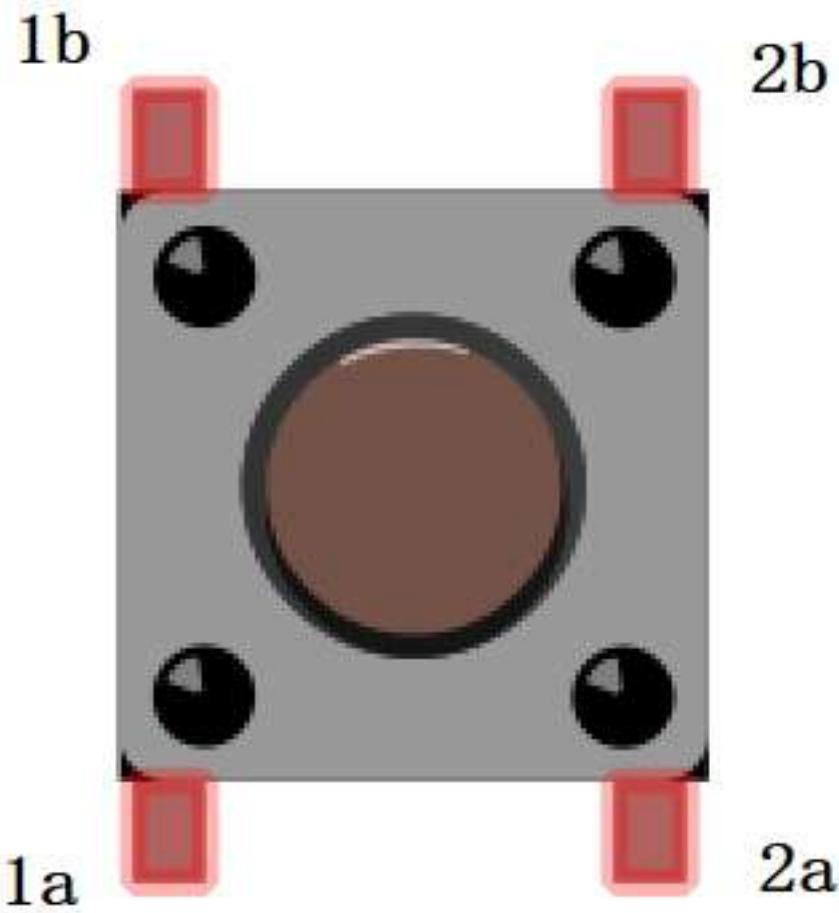
`Serial.begin(9600)` - opens the serial communication between the arduino and the computer at baud rate 9600bps.

`Serial.available()` - return the number of bytes that are currently present in the arduino serial buffer.

`Serial.println()` - this function prints data to the serial port to which arduino is connected.

`Serial.read()` – will read the serial data(ASCII value) from the key board

push button



Simulator time: 00:00:12

Code Stop Simulation Export Share

Text

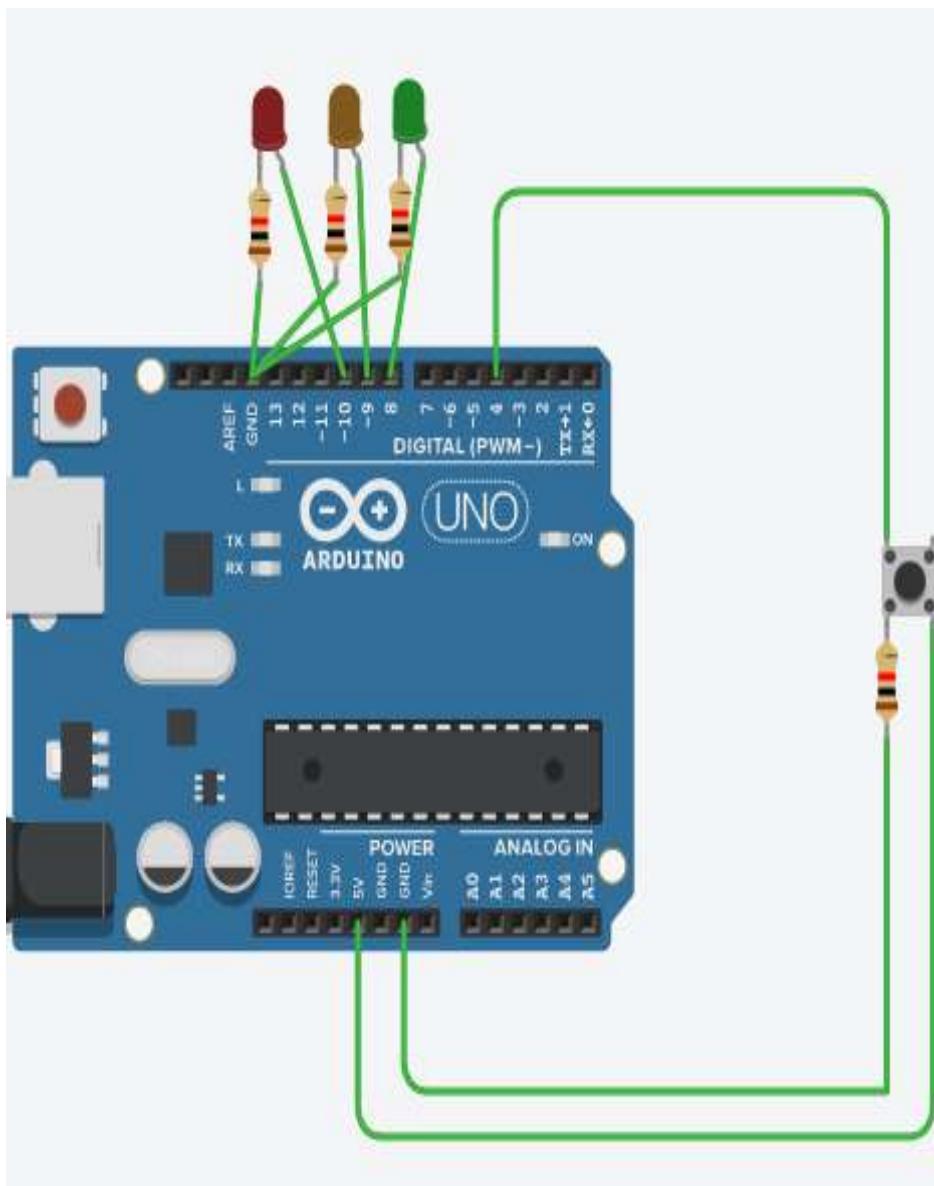
1 void setup()
2 {
3 pinMode(13, OUTPUT);
4
5 pinMode(12, INPUT);
6 Serial.begin(9600);
7 }
8
9 void loop()
10 {
11 if(digitalRead(12)==1){
12
13 digitalWrite(13, HIGH);
14 Serial.println("ON");
15 delay(1000); // Wait for 1000 millisecond(s)
16 }
17 digitalWrite(13, LOW);
18 delay(1000); // Wait for 1000 millisecond(s)
19 Serial.println("OFF");
20 }
21 }

Serial Monitor

OFF
ON
OFF
OFF
ON

Send Clear

LED Rotation



Text



1 (Arduino Uno R3)

```
1 int count=1;
2 void setup()
3 {
4     pinMode(4, INPUT);      pinMode(8, OUTPUT);
5     pinMode(9, OUTPUT);      pinMode(10, OUTPUT);
6 }
7
8 void loop()
9 {
10    int d=digitalRead(4);
11
12    if(d==HIGH && count==1){
13        digitalWrite(10,HIGH);  count=2;
14    }else if(d==HIGH && count==2){
15        digitalWrite(10,LOW);   digitalWrite(9,HIGH);
16        count=3;
17    }else if(d==HIGH && count==3){
18        digitalWrite(9,LOW);   digitalWrite(8,HIGH);
19        count=4;
20    }else if(d==HIGH && count==4){
21        digitalWrite(10,LOW);  digitalWrite(9,LOW);  digitalWrite(8,LOW);
22        count=1;
23    }
24    delay(200);
25
26 }
```



A **potentiometer** is a three-terminal resistor with a sliding or rotating contact that forms an adjustable voltage divider.

A variable resistor with a third adjustable terminal. The potential at the third terminal can be adjusted to give any fraction of the potential across the ends of the resistor.

An **LDR (Light Dependent Resistor)** is a component that has a (variable) resistance that changes with the light intensity that falls upon it. This allows them to be used in light sensing circuits.



map()

Re-maps a number from one range to another.

Arduino has an analogRead range from 0 to 1023, and an analogWrite range only from 0 to 255, therefore the data from the potentiometer needs to be converted to fit into the smaller range before using it to dim the LED.

Example:

```
int var=analogRead(A0);
int val=map(var,0,1024,0,256);
```

<https://www.tinkercad.com/things/2OCPKLPhcOb-incredible-crift-densor/editel?tenant=circuits>

Incredible Crift-Densor

All changes saved

Code Stop Simulation Export Share

Simulator time: 00:00:32

Text

1 (Arduino Uno R3)

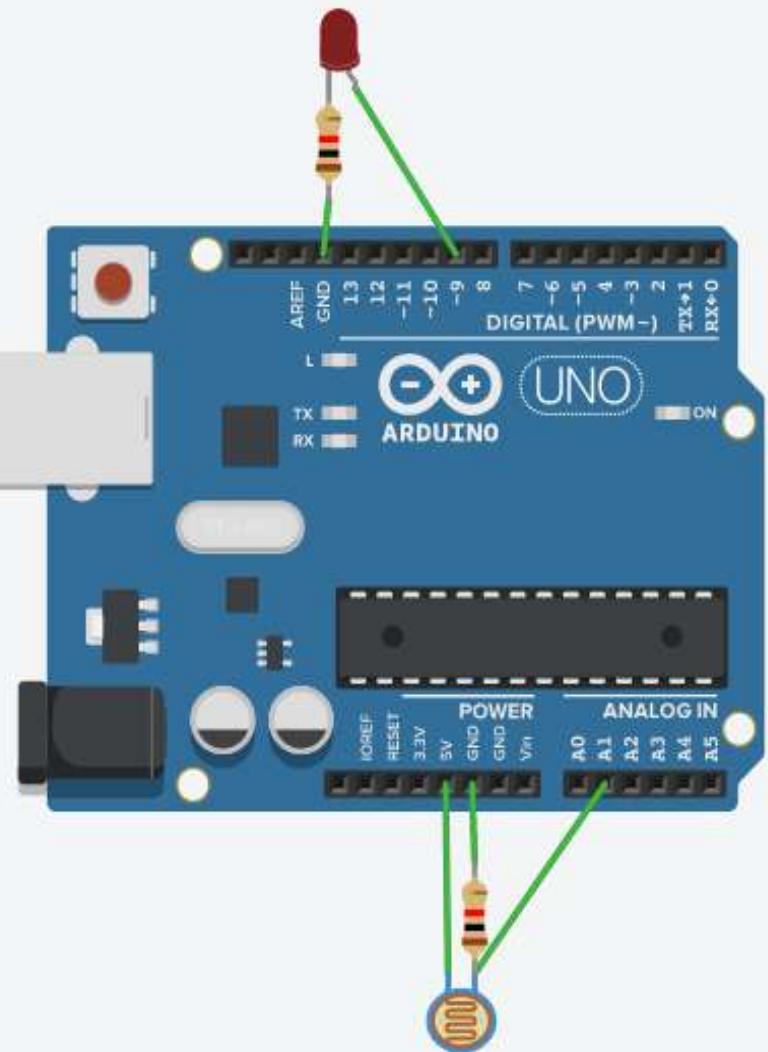
The circuit diagram shows an Arduino Uno board with a blue background. A red LED is connected to digital pin 13 through a green wire. The LED's ground lead is connected to pin 8 (GND) and also to the wiper of a blue potentiometer. The potentiometer's center lead is connected to analog pin A1, and its ground lead is connected to pin 8 (GND). The Arduino Uno has a red USB cable connected to its left side. The board features various pins labeled: AREF, GND, 13, 12, -11, -10, -9, 8, 7, 6, -5, -4, -3, 2, TX+1, RX+0, TX, RX, ARDUINO, IOREF, RESET, 3.3V, 5V, GND, V_{in}, ANALOG IN, A0, A1, A2, A3, A4, A5.

```
1 void setup()
2 {
3     pinMode(13, OUTPUT);
4     pinMode(A1, INPUT);
5     Serial.begin(9600);
6 }
7
8 void loop()
9 {
10    int var=0, y=0;
11    var=analogRead(A1);
12    y = map(var, 1, 1023, 1,255);
13    analogWrite(13, y);
14    delay(1000); // Wait for 1000 millisecond(s)
15    Serial.print(var);
16    Serial.print(":");
17    Serial.println(y);
18 }
```

Serial Monitor

1022:254
328:82
1:1
1:1
62:16
409:102
675:168
675:168

Send Clear



The image shows an Arduino Uno breadboard setup. A breadboard is populated with various components: a red LED connected to digital pin 9 through a resistor; a potentiometer connected to analog pin A1; a 74HC14 inverter IC with its output connected to digital pin 13; and a 74HC04 hex inverter IC with its outputs connected to pins 12, 11, 10, and 9. Pin 9 is also connected to ground. The Arduino Uno is connected to the breadboard via a USB cable. The code window shows the following sketch:

```
1 void setup()
2 {
3     pinMode(9, OUTPUT);
4     pinMode(A1, INPUT);
5     Serial.begin(9600);
6 }
7
8 void loop()
9 {
10    int var; int y;
11    var=analogRead(A1);
12    Serial.print(var);
13    Serial.print(":");
14    y=map(var,1,700,1,256);
15    Serial.println(y);
16    digitalWrite(9,y);
17 }
```

The Serial Monitor window displays the following output:

```
6:2
6:2
6:2
6:2
6:2
6:2
6:2
6:2
```

Servo Motor

- A Servo Motor is a small device that has an output shaft. This shaft can be positioned to specific angular positions by sending the servo a coded signal. Usually, shaft can turn upto 180 degrees.
- Servo motors were first used in the Remote Control (RC) world, usually to control the steering of RC cars or the flaps on a RC plane. With time, they found their uses in robotics, automation, and of course, the Arduino world.



Simulator time: 00:00:11

Code Stop Simulation Export Share

Text

1 (Arduino Uno R3)

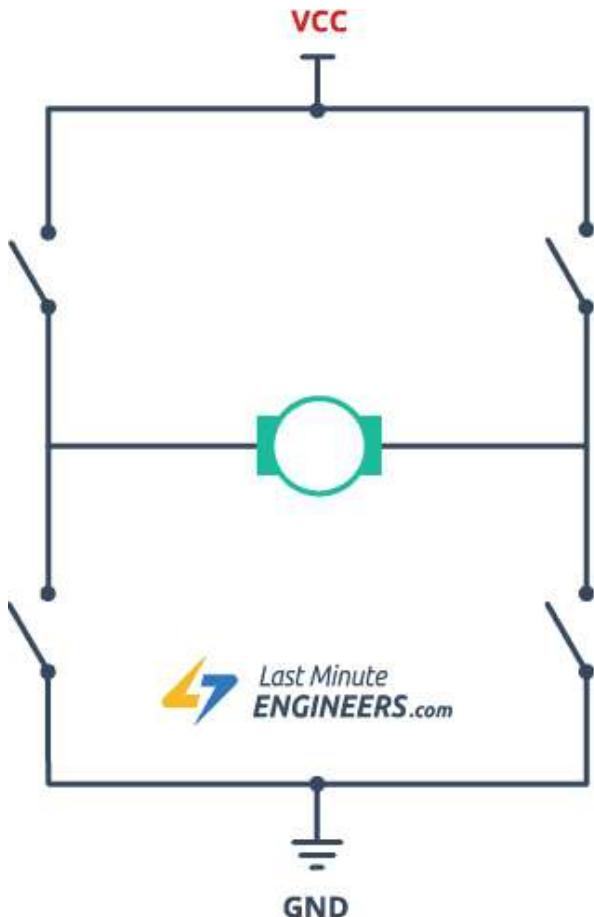
```
#include<Servo.h>
Servo myservo;
void setup()
{
  myservo.attach(9);
}

void loop()
{
  delay(1000);
  myservo.write(0);    delay(1000);
  myservo.write(45);  delay(1000);
  myservo.write(90);  delay(1000);
  myservo.write(135); delay(1000);
  myservo.write(180); delay(1000);
}
```

Serial Monitor

DC motors

- DC motors are interfaced to Arduino using L293D Motor Driver IC. It can control both speed and spinning direction of two DC motors.
- The L293D is a dual-channel H-Bridge motor driver capable of driving a pair of DC motors or one stepper motor.



IN1	IN2	Spinning Direction
Low(0)	Low(0)	Motor OFF
High(1)	Low(0)	Forward
Low(0)	High(1)	Backward
High(1)	High(1)	Motor OFF

<https://lastminuteengineers.com/l293d-dc-motor-arduino-tutorial/>

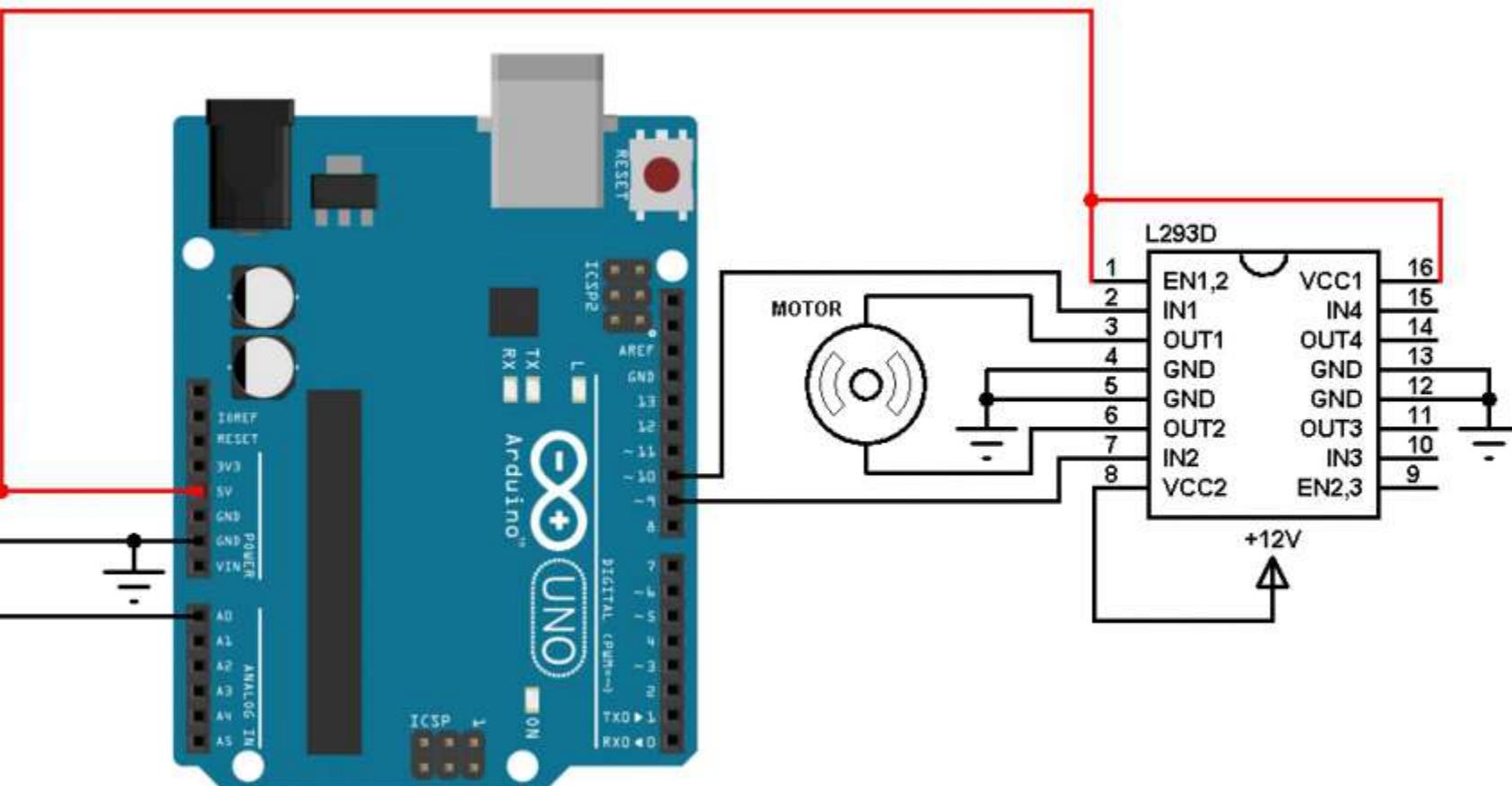


L293D Pinout

The L293D motor driver's output channels for the motor A and B are brought out to pins OUT1, OUT2 and OUT3, OUT4 respectively.

For each of the L293D's channels, there are two types of control pins which allow us to control speed and spinning direction of the DC motors at the same time viz. Direction control pins & Speed control pins. **IN1**, **IN2** & **IN3**, **IN4**

Joystick



Incredible Trug

All changes saved

Code Start Simulation Export Share

Text

1 void setup()
2 {
3 pinMode(2,OUTPUT);
4 pinMode(3,OUTPUT);
5 pinMode(12,OUTPUT);
6 pinMode(13,OUTPUT);}
7
8 void loop()
9 {
10 digitalWrite(2,LOW);
11 digitalWrite(3,HIGH);
12 digitalWrite(12,LOW);
13 digitalWrite(13,HIGH);
14 }
15
16 // IC - L293D

Serial Monitor

The circuit diagram shows an Arduino Uno connected to a breadboard. The Arduino pins are connected to a 9V battery and an L293D motor driver IC. The breadboard has two rows of 14 columns each, labeled 'a' through 'j' on both sides. The connections are as follows: Pin 13 to 'a', Pin 12 to 'b', Pin 3 to 'c', Pin 2 to 'd', and GND to 'e'. The L293D IC is connected to the breadboard with its pins labeled 1 through 16. The power supply is a 9V battery.

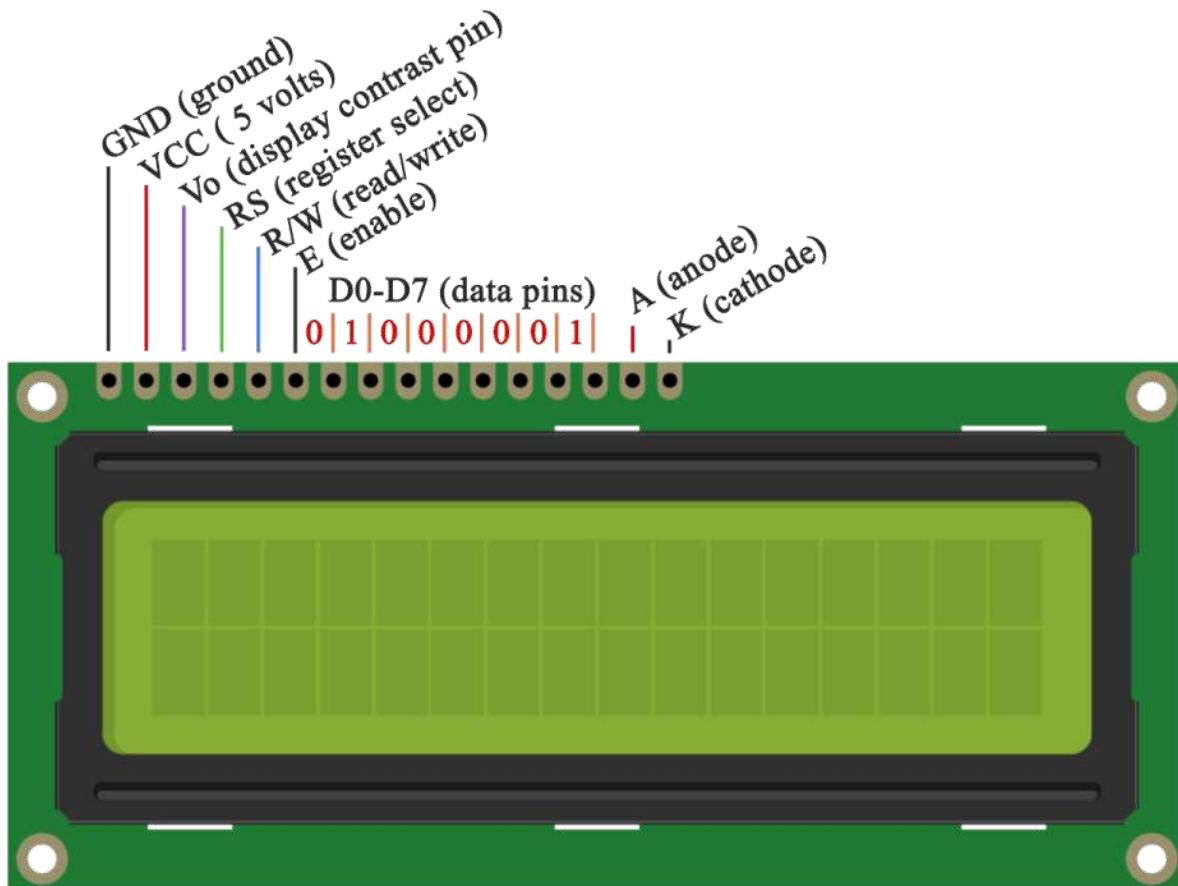
LCD – Liquid Crystal Display

The library works with in either 4- or 8-bit mode (i.e. using 4 or 8 data lines).

Vo pin on which we can attach a potentiometer for controlling the contrast of the display.

The **RS pin** is set on low state or zero volts, then we are sending commands to the LCD like: set the cursor to a specific location, clear the display, turn off the display and so on.

R/W pin - read/write mode



```
Liquid Crystal Library - #include <LiquidCrystal.h>
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
```

- `lcd.begin()`
- `lcd.print()`
- `lcd.setCursor()`
- `lcd.noDisplay();`
- `lcd.display();`

<https://www.tinkercad.com/things/7zJS2hA73Gi-glorious-gaaris-borwo/editel>

All changes saved

TIN KER CAD Glorious Gaaris-Borwo

Code Start Simulation Export Share

Text 1 (Arduino Uno R3)

```
#include<LiquidCrystal.h>
LiquidCrystal lcd(13,12,4,5,6,7);
void setup()
{
lcd.begin(16,2);
}
void loop()
{
lcd.clear();
lcd.write("Mohan Rhitesh");
delay(1000);
}
```

Toggle serial monitor

Serial Monitor

The circuit diagram shows an Arduino Uno connected to a breadboard. The Arduino's 5V pin is connected to the breadboard power rail. The GND pin is connected to the breadboard ground rail. Pin 13 is connected to the LCD's RS pin. Pin 12 is connected to the LCD's E pin. Pin 4 is connected to the LCD's D4 pin. Pin 5 is connected to the LCD's D5 pin. Pin 6 is connected to the LCD's D6 pin. Pin 7 is connected to the LCD's D7 pin. Pin 11 is connected to the LCD's A0 pin. Pin 10 is connected to the LCD's A1 pin. Pin 9 is connected to the LCD's A2 pin. Pin 8 is connected to the LCD's A3 pin. Pin 14 is connected to the LCD's A4 pin. Pin 15 is connected to the LCD's A5 pin. Pin 1 is connected to the LCD's VSS pin. Pin 3 is connected to the LCD's VDD pin. Pin 16 is connected to the LCD's D0 pin. Pin 17 is connected to the LCD's D1 pin. Pin 18 is connected to the LCD's D2 pin. Pin 19 is connected to the LCD's D3 pin. Pin 20 is connected to the LCD's D4 pin. Pin 21 is connected to the LCD's D5 pin. Pin 22 is connected to the LCD's D6 pin. Pin 23 is connected to the LCD's D7 pin. Pin 24 is connected to the LCD's A0 pin. Pin 25 is connected to the LCD's A1 pin. Pin 26 is connected to the LCD's A2 pin. Pin 27 is connected to the LCD's A3 pin. Pin 28 is connected to the LCD's A4 pin. Pin 29 is connected to the LCD's A5 pin. Pin 2 is connected to the LCD's VSS pin. Pin 4 is connected to the LCD's VDD pin. Pin 14 is connected to the LCD's D0 pin. Pin 15 is connected to the LCD's D1 pin. Pin 16 is connected to the LCD's D2 pin. Pin 17 is connected to the LCD's D3 pin. Pin 18 is connected to the LCD's D4 pin. Pin 19 is connected to the LCD's D5 pin. Pin 20 is connected to the LCD's D6 pin. Pin 21 is connected to the LCD's D7 pin. Pin 22 is connected to the LCD's A0 pin. Pin 23 is connected to the LCD's A1 pin. Pin 24 is connected to the LCD's A2 pin. Pin 25 is connected to the LCD's A3 pin. Pin 26 is connected to the LCD's A4 pin. Pin 27 is connected to the LCD's A5 pin. Pin 28 is connected to the LCD's VSS pin. Pin 29 is connected to the LCD's VDD pin.

(8) WhatsApp | (55) save dat | Maths sub li | IJANA — Int | Arduino - Po | Read PWM, | Three Ways | Circuit design | thinkercad- | Circuit de X +

https://www.tinkercad.com/things/alf4pQt7jDx-stunning-fyran-amberis/editel

All changes saved

TIN KER CAD Stunning Fyran-Amberis

Code Start Simulation Export Share

Text 1 (Arduino Uno R3)

```
void setup()
{
  pinMode(13, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  digitalWrite(13, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)

  digitalWrite(13, LOW);
  delay(1000); // Wait for 1000 millisecond(s)
}
```

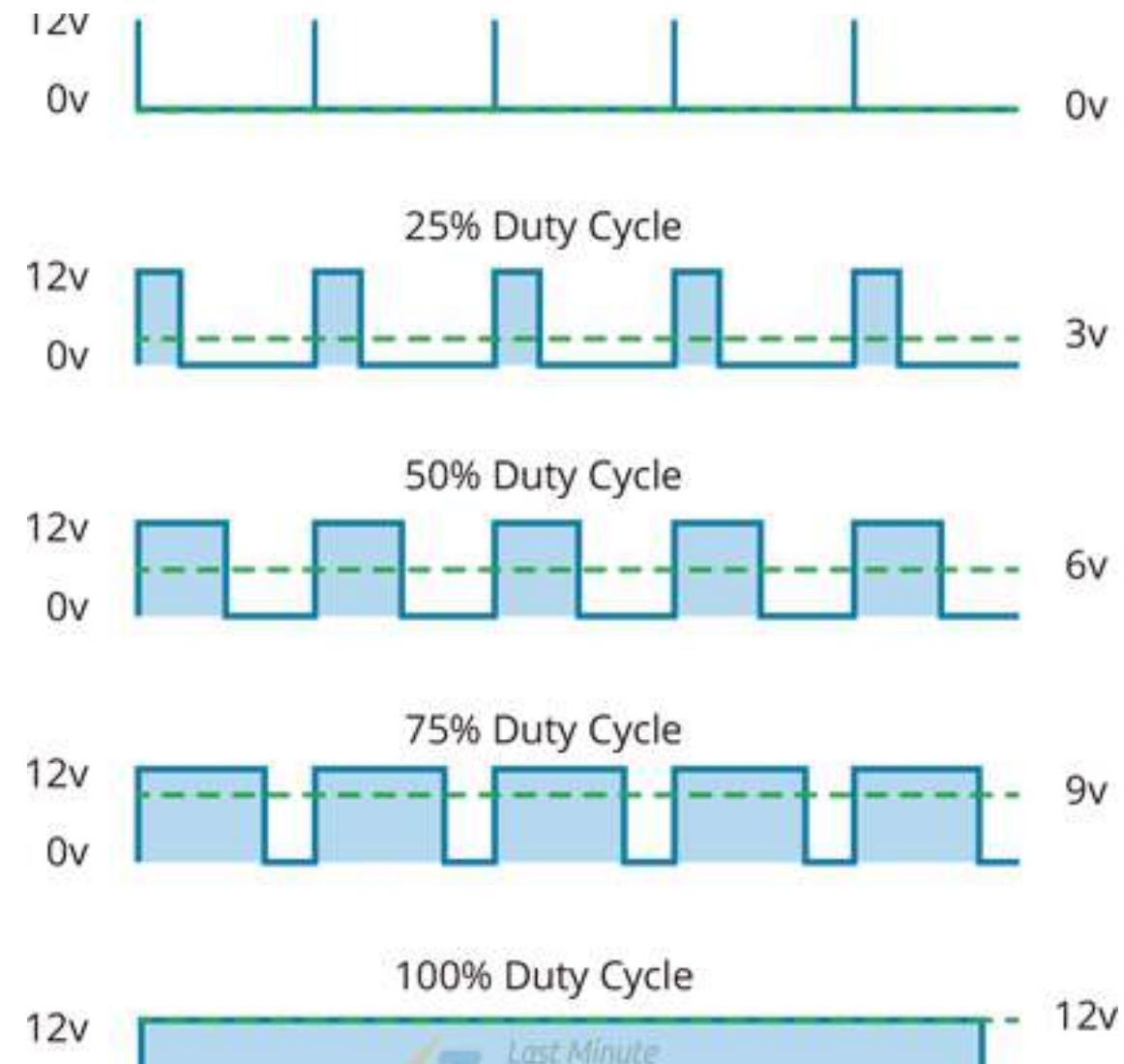
Toggle serial monitor

Serial Monitor

The screenshot shows a Tinkercad project titled "Stunning Fyran-Amberis". On the left, an Arduino Uno board is connected to a breadboard. A digital potentiometer is connected between pins 13 and 5. Pin 13 is also connected to a 10k pull-down resistor and ground. Pin 5 is connected to a 10k pull-up resistor and VCC. Two light bulbs are connected in parallel across the breadboard power rails. The breadboard has a central vertical connection point. The Arduino's 5V and GND pins are connected to the breadboard's power rails. The code in the editor uses digital pin 13 to toggle between HIGH and LOW states, with a 1-second delay for each state.

PWM - Pulse Width Modulation

- PWM is a technique where average value of the input voltage is adjusted by sending a series of ON-OFF pulses.
- The average voltage is proportional to the width of the pulses known as Duty Cycle.
- The higher the duty cycle, the greater the average voltage being applied to the dc motor(High Speed) and the lower the duty cycle, the less the average voltage being applied to the dc motor(Low Speed).
- Below image illustrates PWM technique with various duty cycles and average voltages.



Button Input:
On/off state change

User input features of the fan

- Potentiometer for speed control
 - Continually variable input makes sense for speed control
 - Previously discussed
- Start/stop
 - Could use a conventional power switch
 - Push button (momentary) switch
- Lock or limit rotation angle
 - Button click to hold/release fan in one position
 - Potentiometer to set range limit

Conventional on/off switch

- Basic light switch or rocker switch
 - Makes or breaks connection to power
 - Switch stays in position: On or Off
 - Toggle position indicates the state
 - NOT in the Arduino Inventors Kit



Image from sparkfun.com

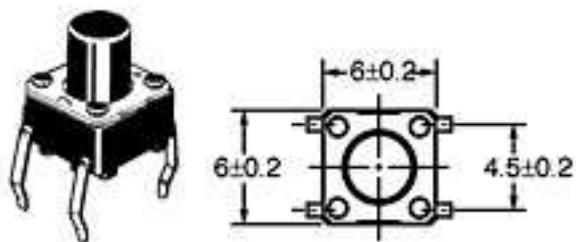
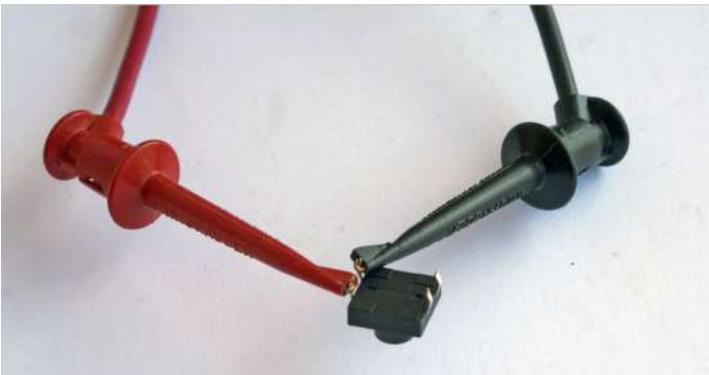


Image from lowes.com

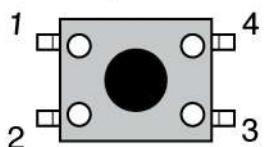
How does a button work?

- Simple switch schematic
- Use DMM to measure open/closed circuit
- Map the pin states

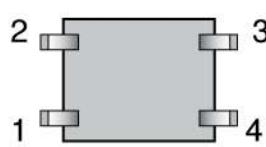
Measure Open and Closed Circuits



Top View



Bottom View



Connect Pins	Measured Resistance (Ω)	
	When not pressed	When pressed
1 and 2		
1 and 3		
1 and 4		
2 and 3		

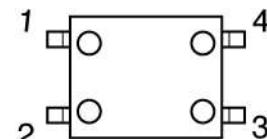
Measure Open and Closed Circuits

Data from Measurements:

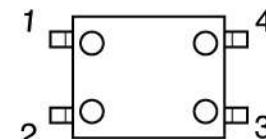
Connect Pins	Measured Resistance (Ω)	
	When not pressed	When pressed
1 and 2		
1 and 3		
1 and 4		
2 and 3		

Sketch Connections:

Draw lines between connectors

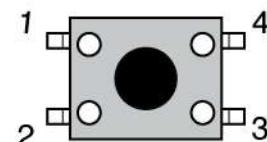


When not pressed

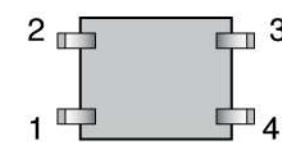


When pressed

Top View



Bottom View



Push Button Switches

- A momentary button is a “Biased Switch”
- Pushing the button changes state
- State is reversed (return to biased position) when button is released
- Two types
 - NO: normally open
 - NC: normally closed

Normally Open

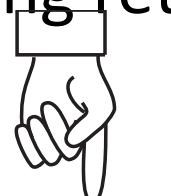


Normally Closed

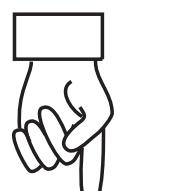


Momentary or push-button switches

- Normally open
 - electrical *contact is made* when button is pressed
- Normally closed
 - electrical *contact is broken* when button is pressed
- Internal spring returns button to its un-p



Open



Closed

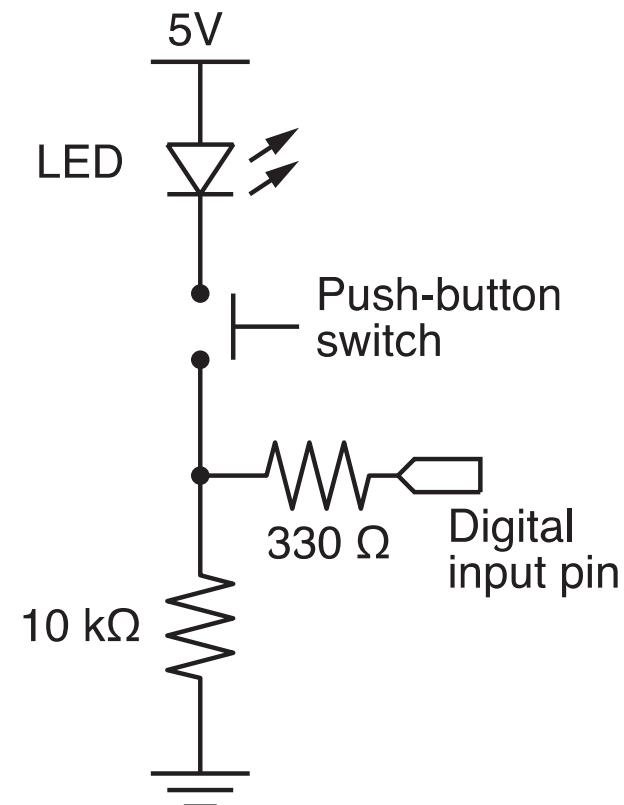


Putting buttons into action

- Build the circuit: same one is used for all examples
 - Test with LED on/off
 - LED is only controlled by the button, not by Arduino code
- Create a “wait to start” button
 - Simplest button implementation
 - Execution is blocked while waiting for a button click
- Use an interrupt handler
 - Most sophisticated: Don’t block execution while waiting for button input
 - Most sophisticated: Requires good understanding of coding
 - Requires “de-bouncing”
 - Not too hard to use as a black box

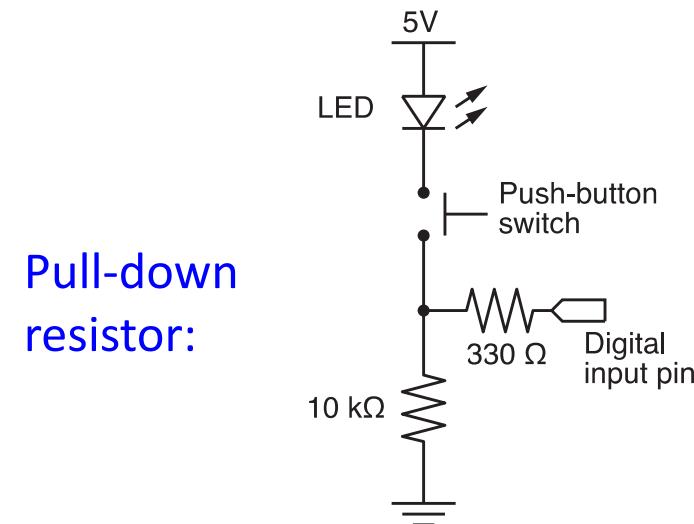
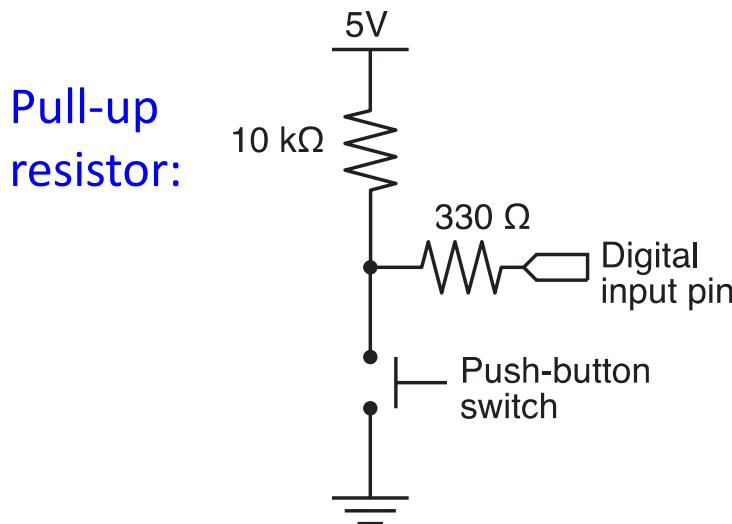
Momentary Button and LED Circuit

- Digital input with a *pull-down resistor*
 - When switch is open (button not pressed):
 - Digital input pin is tied to ground
 - No current flows, so there is no voltage difference from input pin to ground
 - Reading on digital input is LOW
 - When switch is closed (button is pressed):
 - Current flows from 5V to ground, causing LED to light up.
 - The 10k resistor limits the current draw by the input pin.
 - The 330Ω resistor causes a large voltage drop between 5V and ground, which causes the digital input pin to be closer to 5V.
 - Reading on digital input is HIGH



Technical Note

- Usually we do not include an LED directly in the button circuit. The following diagrams show plan button circuits with pull-up and pull-down resistors. In these applications, the pull-up or pull-down resistors should be 10k. Refer to Lady Ada Tutorial #5:
 - <http://www.ladyada.net/learn/arduino/lesson5.html>



Programs for the LED/Button Circuit

- Continuous monitor of button state
 - Program is completely occupied by monitoring the button
 - Used as a demonstration — not practically useful
- Wait for button input
- Interrupt Handler
- All three programs use the same electrical circuit

Continuous monitor of button state

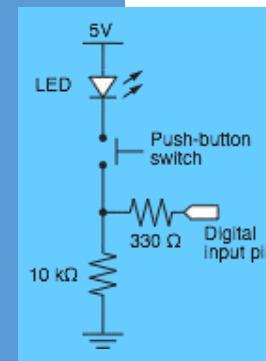
```
int button_pin = 4;           // pin used to read the button

void setup() {
    pinMode(button_pin, INPUT);
    Serial.begin(9600);        // Button state is sent to host
}

void loop() {
    int button;
    button = digitalRead(button_pin);

    if (button == HIGH) {
        Serial.println("on");
    } else {
        Serial.println("off");
    }
}
```

Serial monitor shows a continuous stream of “on” or “off”



This program *does not* control the LED

Programs for the LED/Button Circuit

- Continuous monitor of button state
 - Program is completely occupied by monitoring the button
 - Used as a demonstration — not practically useful
- Wait for button input
 - Blocks execution while waiting
 - May be useful as a start button
- Interrupt Handler
- All three programs use the same electrical circuit

Wait for button press

```
int button_pin = 4; // pin used to read the button

void setup() {
    int start_click = LOW; // Initial state: no click yet
    pinMode(button_pin, INPUT);
    Serial.begin(9600);

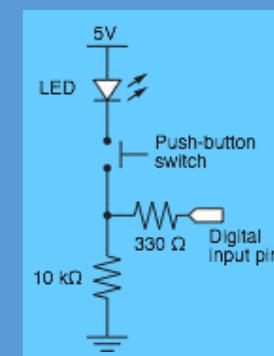
    while (!start_click) {
        start_click = digitalRead(button_pin);
        Serial.println("Waiting for button press");
    }
}

void loop() {
    int button;

    button = digitalRead(button_pin);
    if (button == HIGH) {
        Serial.println("on");
    } else {
        Serial.println("off");
    }
}
```

Same loop() function as
in the preceding sketch

while loop continues
as long as start_click
is FALSE



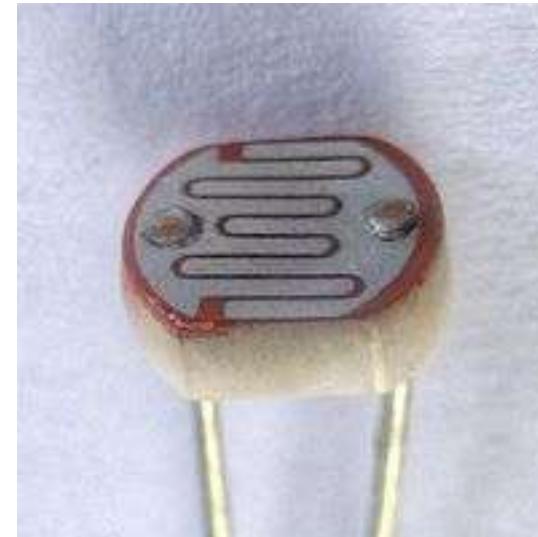
Controlling the intensity of LED using LDR

LDR- Light Dependent Resistor

A photoresistor or light dependent resistor is an electronic component that is sensitive to light.

When light falls upon it, then the resistance changes.

Values of the resistance of the **LDR** may change over many orders of magnitude the value of the resistance falling as the level of light increases.



How an LDR works

- It is relatively easy to understand the basics of how an LDR works without delving into complicated explanations. It is first necessary to understand that an electrical current consists of the movement of electrons within a material.
- Good conductors have a large number of free electrons that can drift in a given direction under the action of a potential difference. Insulators with a high resistance have very few free electrons, and therefore it is hard to make them move and hence a current to flow.

Photoresistor applications

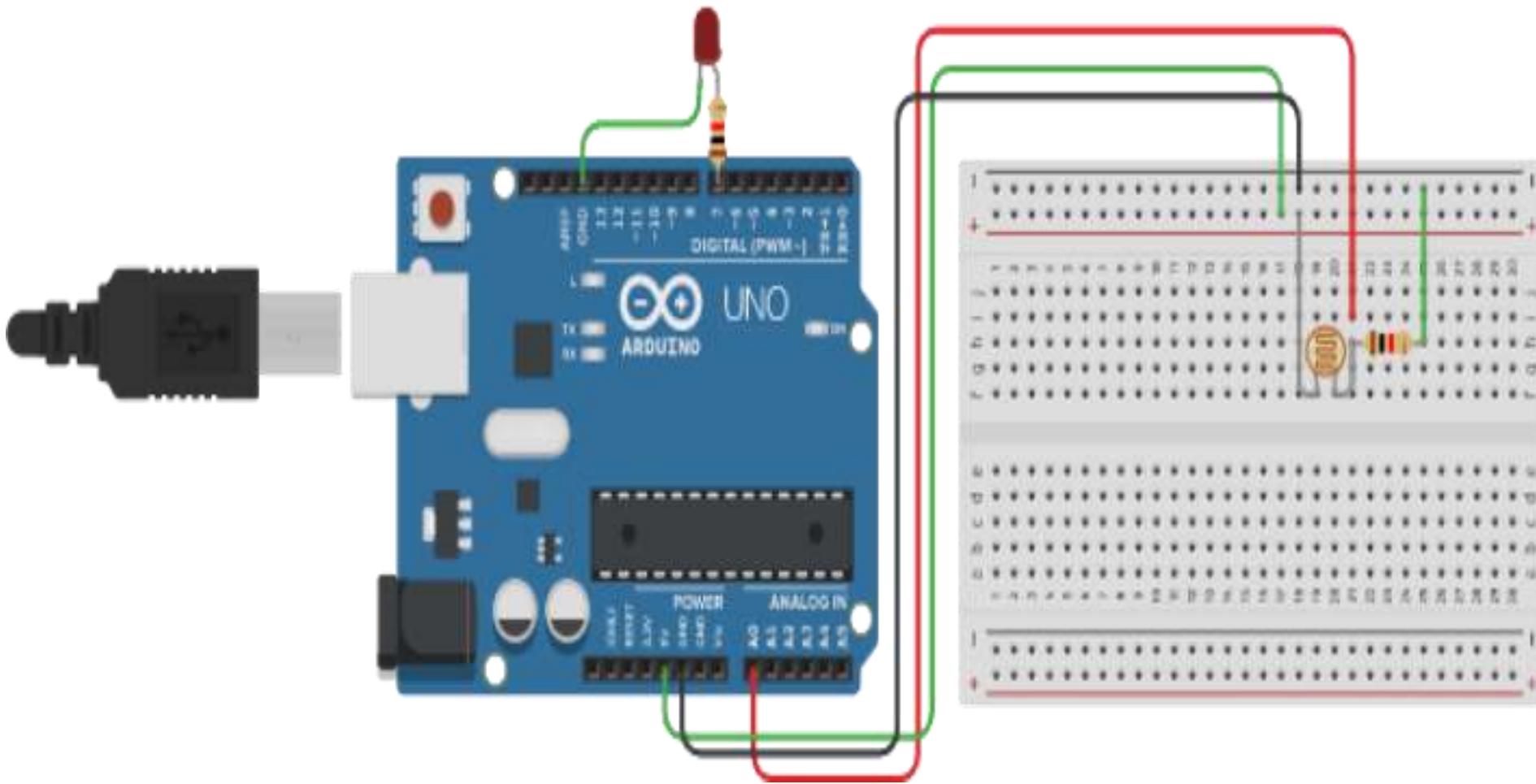
- They are widely used in many different items of electronic equipment and circuit designs including photographic light meters, fire or smoke alarms as well as burglar alarms, and they also find uses as lighting controls for street lamps.
- Photoresistors can also be used to detect nuclear radiation.

LDR on an Arduino Analog Pin

- This would be the most used and most obvious way to use an LDR, as it fluctuates resistance, producing many values.
- In this setup we will make it that the value read from the Analog pin will actually increase as light increases. For this we use a tiny circuit that pushes power through the LDR. Since the LDR decreases its resistance as light increases, more “power” will pass through it to the Analog pin, which results in the Arduino “reading” a higher value.
- The analog pin will read values between 0 and 1023, so it converts the analog signal to a digital representation – also called Analog Digital Converter (AD or ADC), which is build into the Arduino

Arduino LDR Sensor working





Connect **+5V** of the Arduino to one pin of the LDR (LDR pins can be swapped, so no worries about polarity here).

Connect the other pin of the LDR to AO (analog pin of the Arduino) and one pin of the $100\text{K}\Omega$ resistor.

Connect the other pin of the $100\text{K}\Omega$ resistor to GND of the Arduino.

```
const int ledPin = 7;
const int ldrPin = A0;
void setup()
{
Serial.begin(9600);
pinMode(ledPin, OUTPUT);
pinMode(ldrPin, INPUT);
}
void loop()
{
int ldrStatus = analogRead(ldrPin);
    if (ldrStatus <= 200)
    {
        digitalWrite(ledPin, HIGH);
        Serial.print("Its DARK, Turn on the LED : ");
        Serial.println(ldrStatus);
    }
    else
    {
        digitalWrite(ledPin, LOW);
        Serial.print("Its BRIGHT, Turn off the LED : ");
        Serial.println(ldrStatus);
    }
}
```

THANK YOU

What is a Sensor?

A sensor is a device which is used to sense the surroundings of it & gives some useful information about it.

This information is used based on our requirement.

An electronic sensor gives the information by giving some electronic signal.

This information is used to perform a given task which needs interaction with the surroundings.

WHAT IS AN IR SENSOR?

- An IR sensor is a device which detects IR radiation falling on it.
- The band of light which is a thousand times wider than that of a visible light is defined as Infrared light.
- Physically, light with wavelength from 0.7 m to 0.1 mm is called Infrared μ Light. As our eyes cannot see the light with wavelength longer than 7×10^{-5} m, thus IR light is invisible to then human.
- These types of radiations are invisible to our eyes, that can be detected by an infrared sensor.

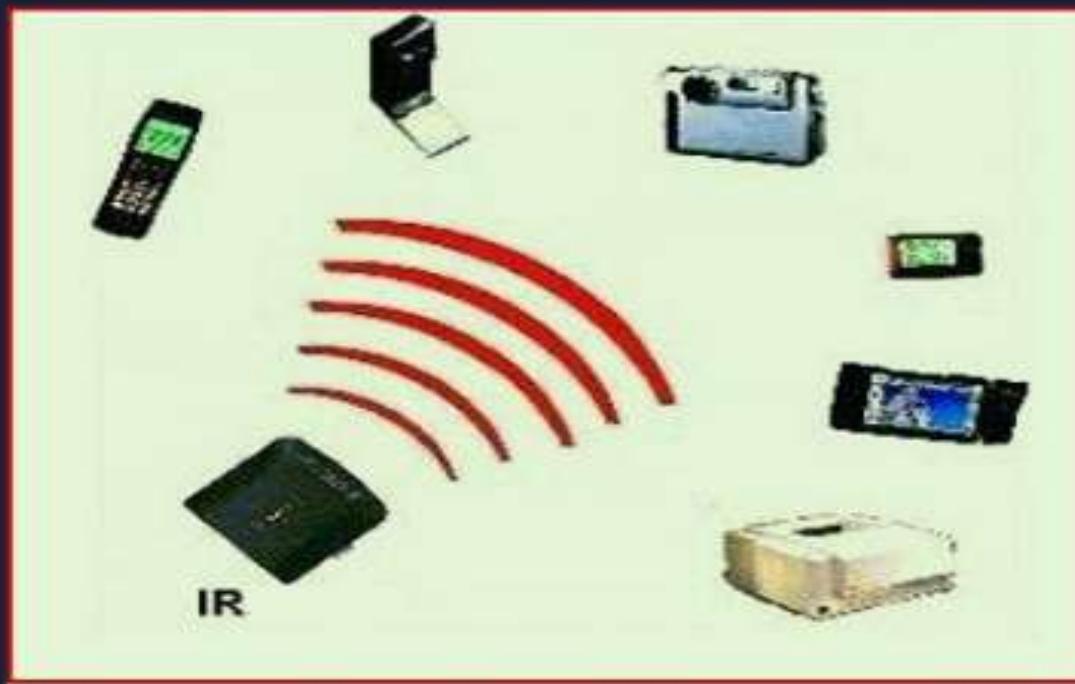
- IR detectors are little microchips with a photocell that are tuned to listen to infrared light.
- They are almost always used for remote control detection - every TV and DVD player has one of these in the front to listen for the IR signal from the clicker.
- Inside the remote control is a matching IR LED, which emits IR pulses to tell the TV to turn on, off or change channels.
- IR light is not visible to the human eye, which means it takes a little more work to test a setup

Different Types of IR Sensors and Their Applications

- IR sensors are classified into different types depending on the applications.
- **Radiation Thermometers:** IR sensors are used in radiation thermometers to measure the temperature depend upon the temperature
- **Flame Monitors:** These types of devices are used for detecting the light emitted from the flames and to monitor how the flames are burning. The Light emitted from flames extend from UV to IR region types.
- **Proximity sensors** (Used in Touch Screen phones and Edge Avoiding Robots), **Contrast sensors** (Used in Line Following Robots)
- **Gas Analyzers:** IR sensors are used in gas analyzers which use absorption characteristics of gases in the IR region. The temperature sensor is used for industrial temperature control.
- **PIR sensor** is used for automatic door opening system.

IR Imaging Devices

IR image device is one of the major applications of IR waves, primarily by virtue of its property that is not visible. It is used for thermal imagers, night vision devices, etc.



For examples Water, rocks, soil, vegetation, an atmosphere, and human tissue all features emit IR radiation.

Infrared IR Sensor Circuit Diagram and Working Principle

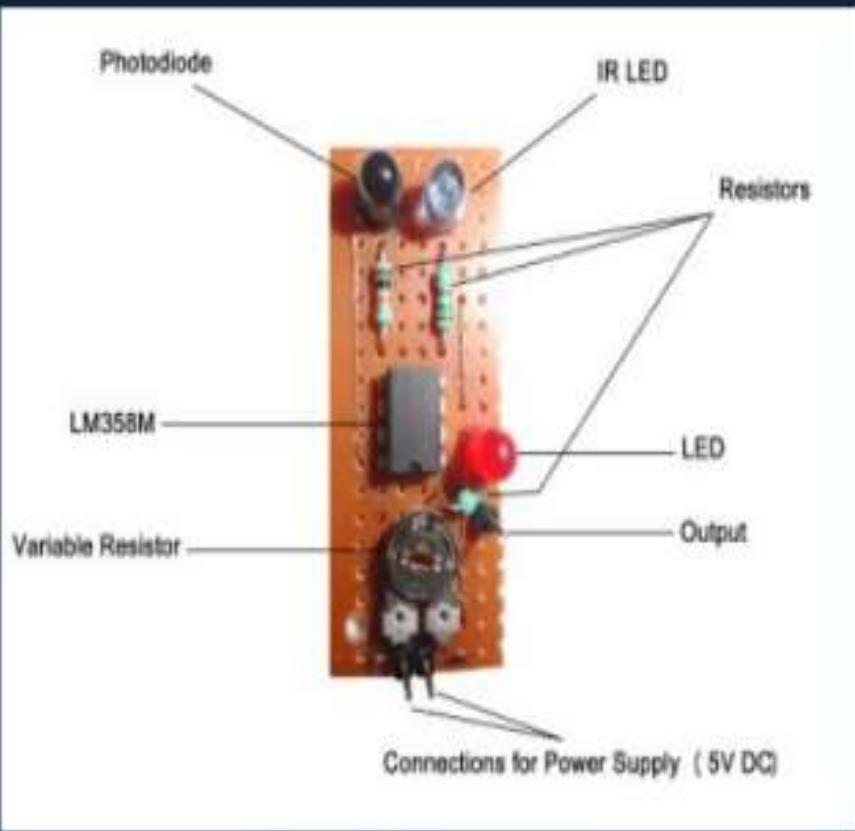
IR Sensor Circuit

- An infrared sensor circuit is one of the basic and popular sensor module in an electronic device. This sensor is analogous to human's visionary senses, which can be used to detect obstacles and it is one of the common applications in real time. This circuit comprises of the following components
 - LM358 IC 2 IR transmitter and receiver pair.
 - Resistors of the range of kilo ohms.
 - Variable resistors.
 - LED (Light Emitting Diode).
 - Photodiode



Infrared IR Sensor Circuit Diagram and Working Principle

IR Sensor Board



Working Principle of IR Sensor

- When the IR receiver does not receive a signal
- The potential at the inverting input goes higher than that non-inverting input of the comparator IC (LM339).
- Thus the output of the comparator goes low, but the LED does not glow.
- When the IR receiver module receives signal to the potential at the inverting input goes low.

IR Sensor Applications

1. Radiation Thermometers

IR sensors are used in radiation thermometers to measure the temperature depend upon the temperature and the material of the object and these thermometers have some of the following features:

- Measurement without direct contact with the object
- Faster response
- Easy pattern measurements

2. Flame Monitors

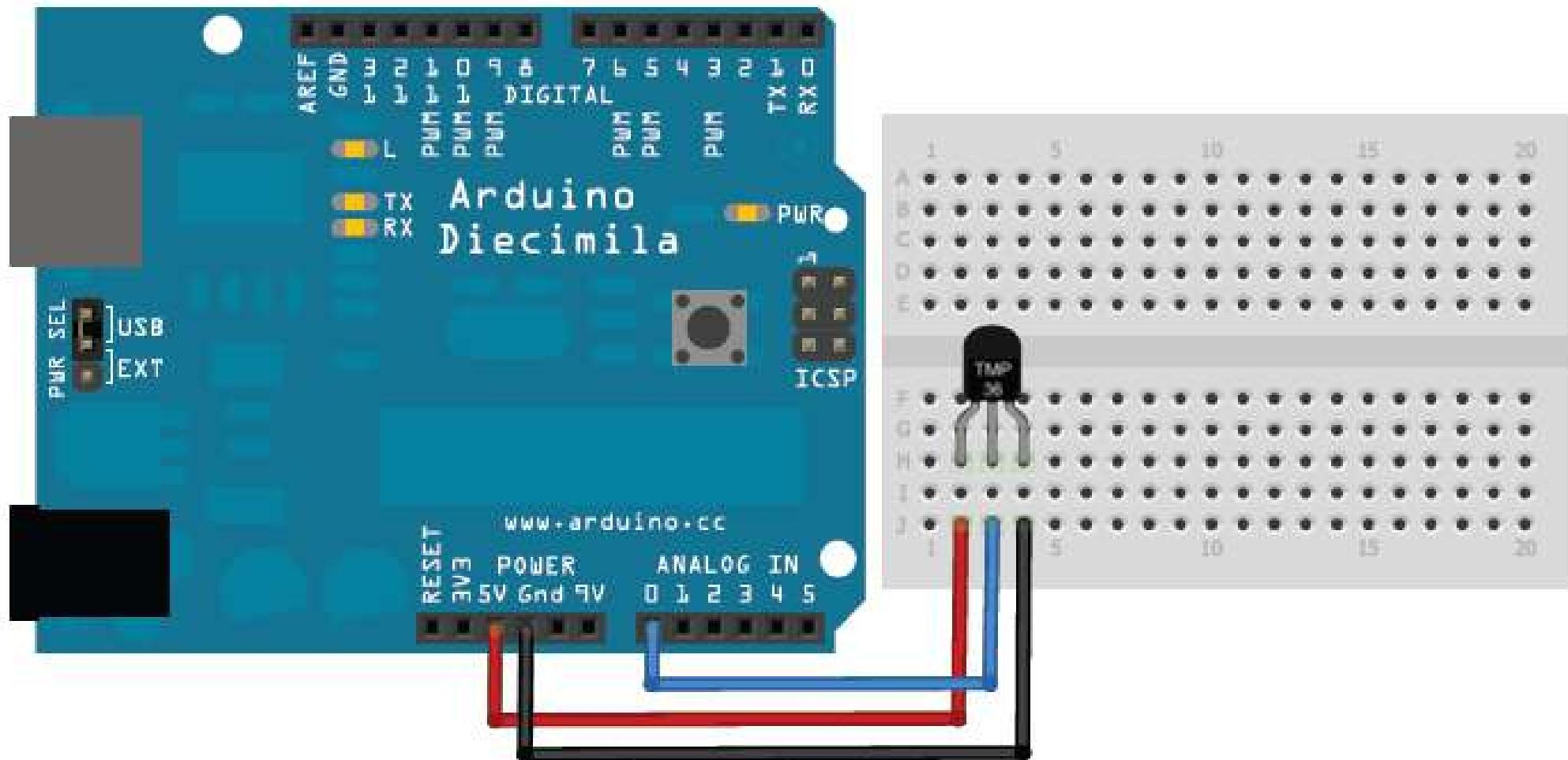
- These types of devices are used for detecting the light emitted from the flames and to monitor how the flames are burning. The Light emitted from flames extend from UV to IR region types. PbS, PbSe, Two-color detector, pyro electric detector are some of the commonly employed detector used in flame monitors.

3. Moisture Analyzers

- Moisture analyzers use wavelengths which are absorbed by the moisture in the IR region. Objects are irradiated with light having these wavelengths($1.1\text{ }\mu\text{m}$, $1.4\text{ }\mu\text{m}$, $1.9\text{ }\mu\text{m}$, and $2.7\mu\text{m}$) and also with reference wavelengths. The Lights reflected from the objects depend upon the moisture content and is detected by analyzer to measure moisture (ratio of reflected light at these wavelengths to the reflected light at reference wavelength). In GaAs PIN photodiodes, Pbs photoconductive detectors are employed in moisture analyzer circuits.

4. Gas Analyzers

- IR sensors are used in gas analyzers which use absorption characteristics of gases in the IR region. Two types of methods are used to measure the density of gas such as dispersive and non dispersive.



Connecting to a Temperature Sensor

- These sensors have little chips in them and while they're not that delicate, they do need to be handled properly.
- Be careful of static electricity when handling them and make sure the power supply is connected up correctly and is between 2.7 and 5.5V DC - so don't try to use a 9V battery!

Reading the Analog Temperature Data

- Unlike the FSR or photocell sensors we have looked at, the TMP36 and friends doesn't act like a resistor.
- Because of that, there is really only one way to read the temperature value from the sensor, and that is plugging the output pin directly into an Analog (ADC) input.

Applications

- DHT11 Relative Humidity and Temperature Sensor can be used in many applications like:
- HVAC (Heating, Ventilation and Air Conditioning) Systems
- Weather Stations
- Medical Equipment for measuring humidity
- Home Automation Systems
- Automotive and other weather control applications

- Remember that you can use anywhere between 2.7V and 5.5V as the power supply. For this example I'm showing it with a 5V supply but note that you can use this with a 3.3v supply just as easily. No matter what supply you use, the analog voltage reading will range from about 0V (ground) to about 1.75V.
- If you're using a 5V Arduino, and connecting the sensor directly into an Analog pin, you can use these formulas to turn the 10-bit analog reading into a temperature:
 - **Voltage at pin in milliVolts = (*reading from ADC*) * (5000/1024)**
This formula converts the number 0-1023 from the ADC into 0-5000mV (= 5V)
 - If you're using a 3.3V Arduino, you'll want to use this:
 - **Voltage at pin in milliVolts = (*reading from ADC*) * (3300/1024)**
This formula converts the number 0-1023 from the ADC into 0-3300mV (= 3.3V)
- Then, to convert millivolts into temperature, use this formula:
- **Centigrade temperature = [(analog voltage in mV) - 500] / 10**

LiquidCrystal Library

This library allows an Arduino board to control LiquidCrystal displays (LCDs) based on the Hitachi HD44780 (or a compatible) chipset, which is found on most text-based LCDs. The library works with either 4- or 8-bit mode (i.e. using 4 or 8 data lines in addition to the rs, enable, and, optionally, the rw control lines).

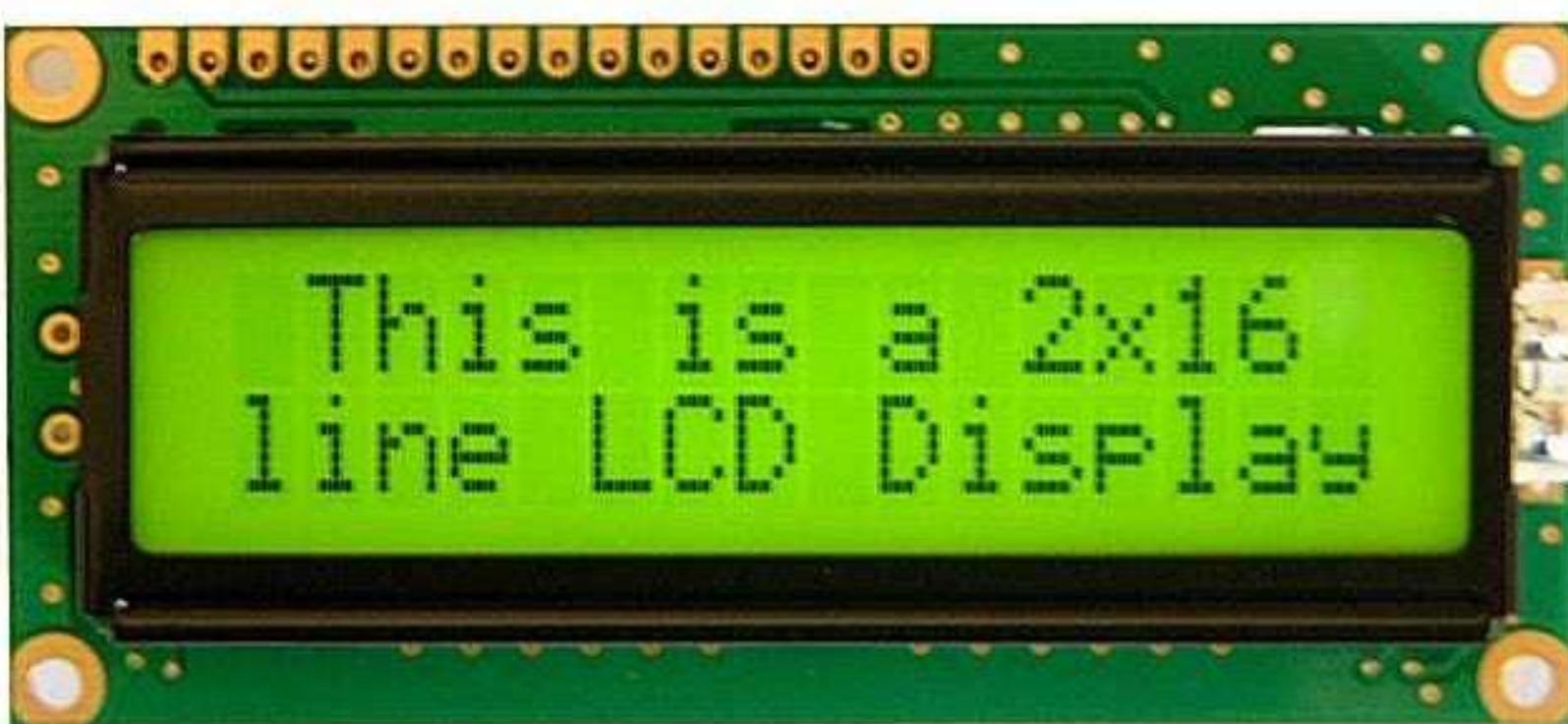
To use this library

```
#include <LiquidCrystal.h>
```

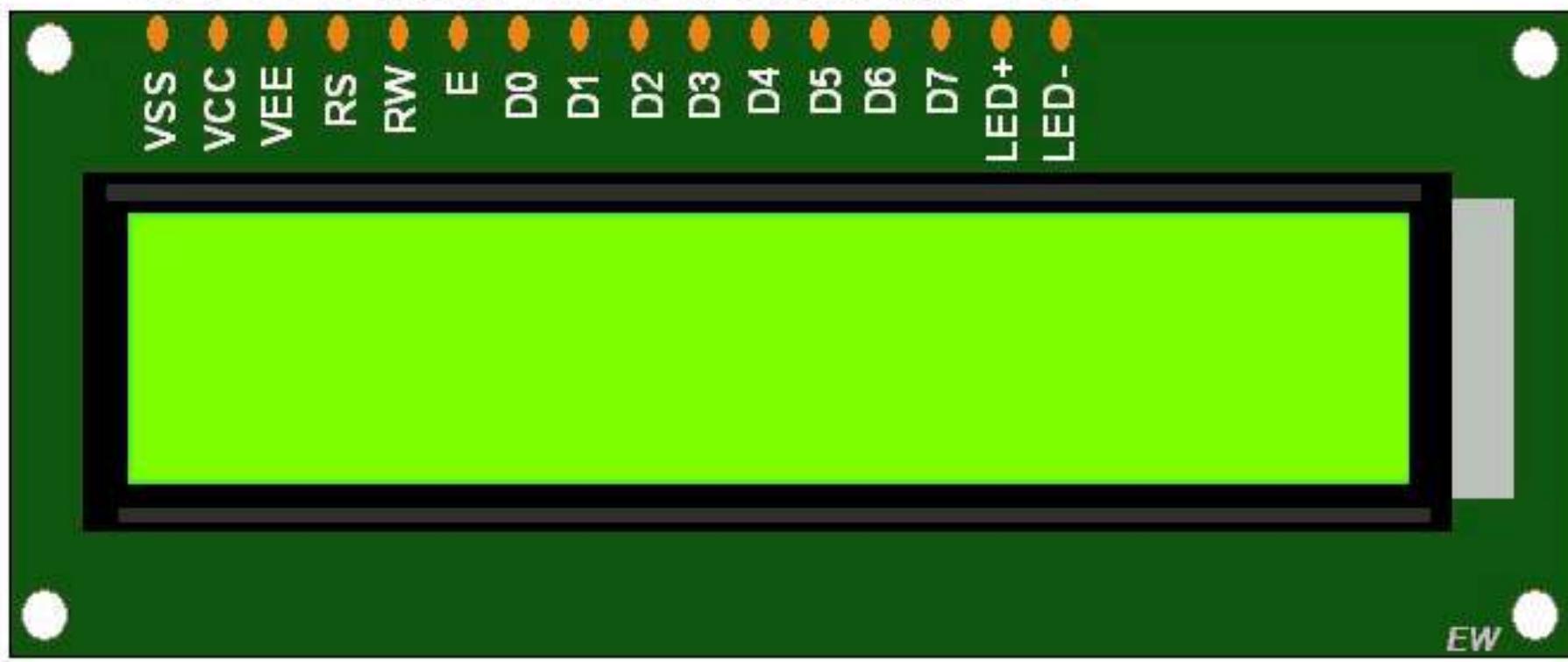
Examples

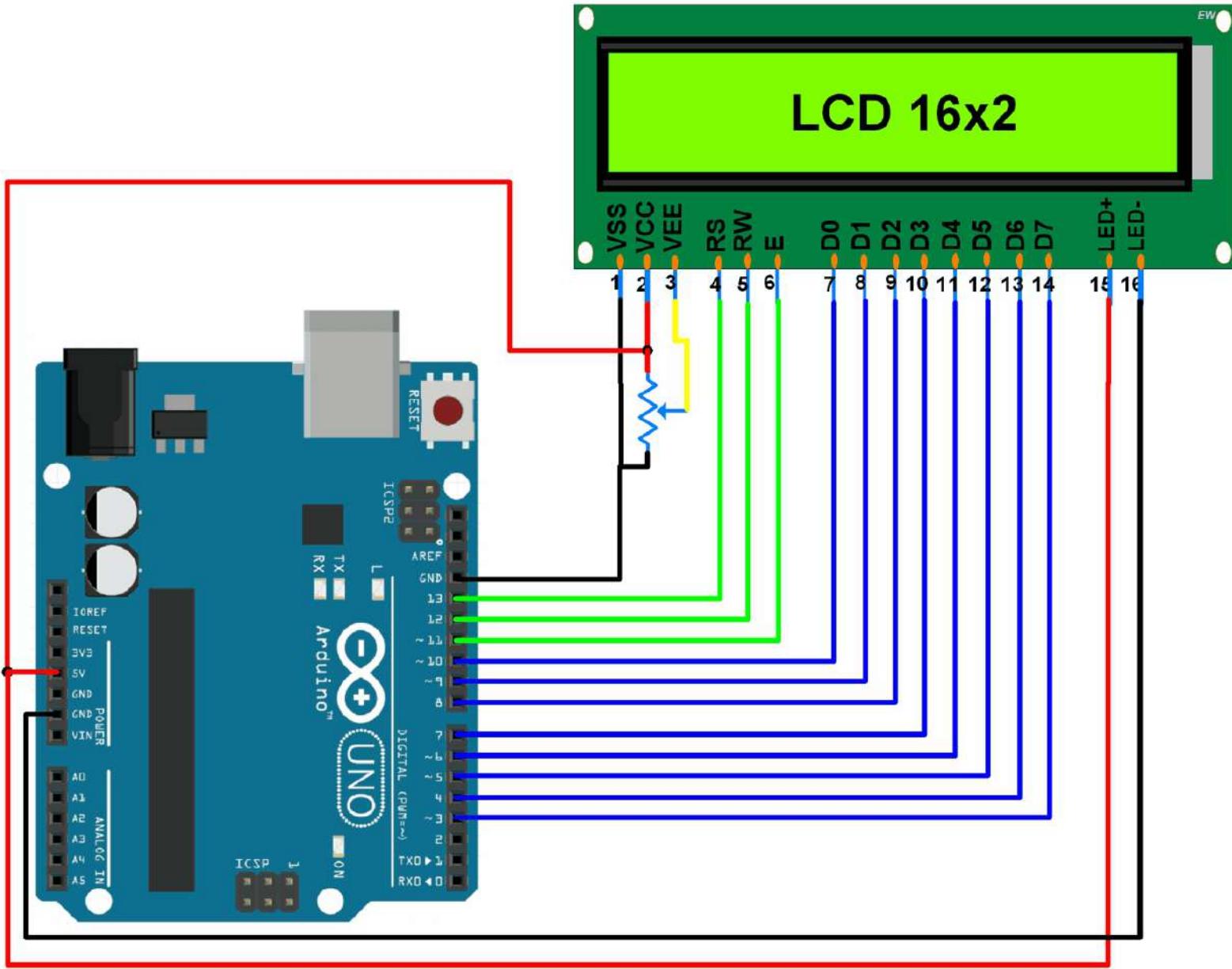
- Autoscroll: Shift text right and left.
- Blink: Control of the block-style cursor.
- Cursor: Control of the underscore-style cursor.
- Display: Quickly blank the display without losing what's on it.
- Hello World: Displays "hello world!" and the seconds since reset.
- Scroll: Scroll text left and right.
- Serial Display: Accepts serial input, displays it.
- Set Cursor: Set the cursor position.
- Text Direction: Control which way text flows from the cursor.

How to use an LCD display



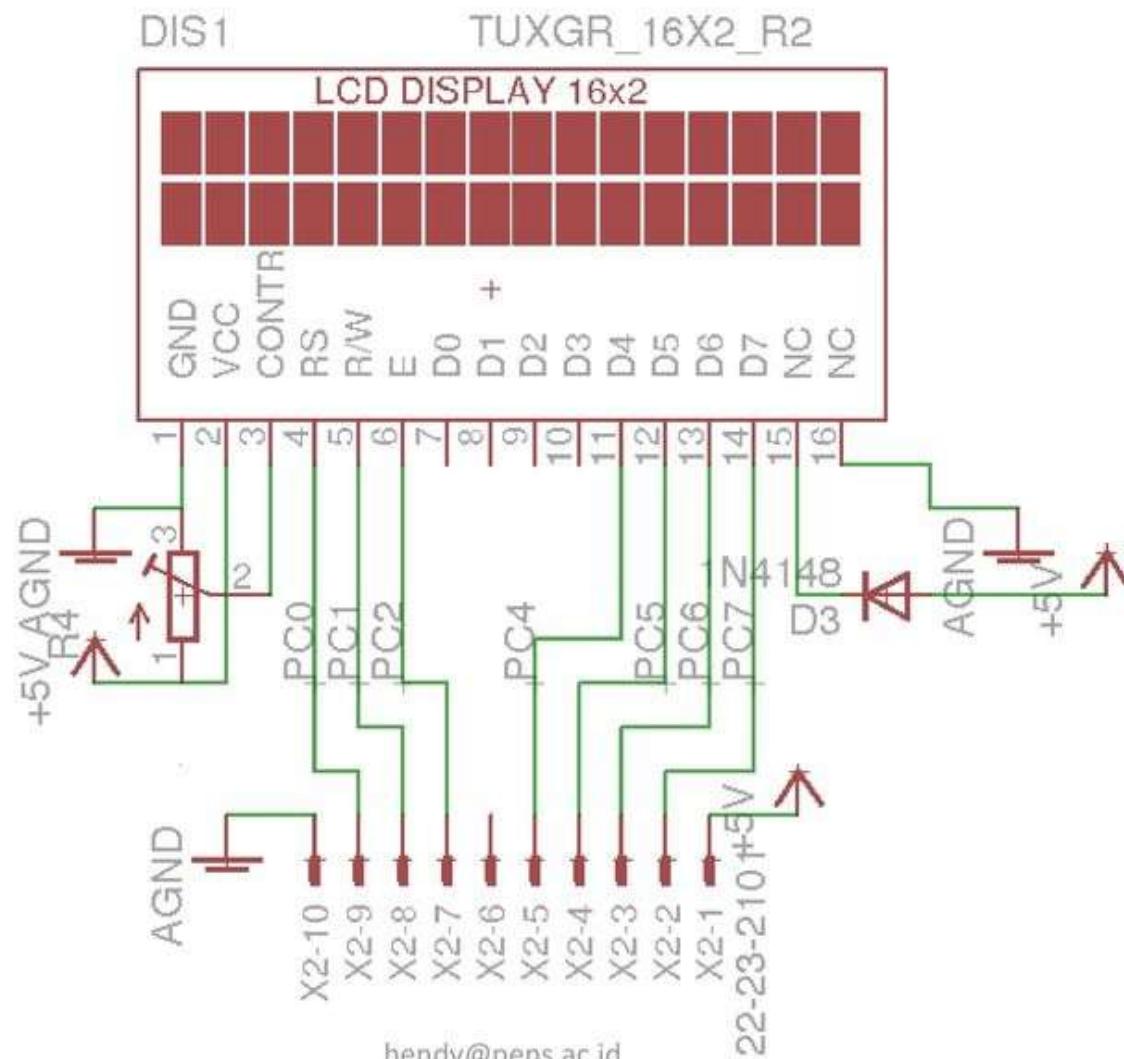
Arduino Tutorial





- The parallel interface consists of the following pins:
- Power Supply pins (Vss/Vcc): Power the LCD
- Contrast pin (Vo): Control the display contrast
- Register Select (RS) pin: Controls where in the LCD's memory you're writing data to
- Read/Write (R/W): Selects reading mode or writing mode
- Enable pin: Enables writing to the registers
- 8 data pins (D0 -D7): The states of these pins (high or low) are the bits that you're writing to a register when you write, or the values you're reading when you read.
- Backlight (Bklt+ and BKlt-) pins: Turn on/off the LED backlight

- The **Lcd.begin(16,2)** command set up the LCD number of columns and rows. For example, if you have an LCD with 20 columns and 4 rows (20x4) you will have to change this to `Lcd.begin(20x4)`.
- The **Lcd.print("--message--")** command print a message to first column and row of lcd display. The "message" must have maximum length equal to lcd columns number. For example, for 16 columns display max length is equal with 16 and for 20 columns display max length is equal with 20.
- The **Lcd.setCursor(0,1)** command will set cursor to first column of second row. If you have an LCD 20x4 and you want to print a message to column five and third row you have to use: `Lcd.setCursor(4,2)`.
-



- The circuit:
 - * LCD RS pin to digital pin 12
 - * LCD Enable pin to digital pin 11
 - * LCD D4 pin to digital pin 5
 - * LCD D5 pin to digital pin 4
 - * LCD D6 pin to digital pin 3
 - * LCD D7 pin to digital pin 2
 - * LCD R/W pin to ground
 - * LCD VSS pin to ground
 - * LCD VCC pin to 5V
 - * 10K resistor:
 - * ends to +5V and ground
 - * wiper to LCD VO pin (pin 3)

Program “Hello Word”

```
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
    // set up the LCD's number of columns and rows:
    lcd.begin(16, 2);
    // Print a message to the LCD.
    lcd.print("hello, world!");
}

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);
}
```

Program Display LCD dan Serial

```
void setup() {
    // set up the LCD's number of columns and rows:
    lcd.begin(16, 2);
    // initialize the serial communications:
    Serial.begin(9600);
}

void loop() {
    // when characters arrive over the serial port...
    if (Serial.available()) {
        // wait a bit for the entire message to arrive
        delay(100);
        // clear the screen
        lcd.clear();
        // read all the available characters
        while (Serial.available() > 0) {
            // display each character to the LCD
            lcd.write(Serial.read());
        }
    }
}
```

Program Display on/off

```
void setup() {  
    // set up the LCD's number of columns and rows:  
    lcd.begin(16, 2);  
    // Print a message to the LCD.  
    lcd.print("hello, world!");  
}  
  
void loop() {  
    // Turn off the display:  
    lcd.noDisplay();  
    delay(500);  
    // Turn on the display:  
    lcd.display();  
    delay(500);  
}
```

- How connect Arduino with LCD to potentiometer?

Take the 10K **potentiometer** and **connect** the first terminal to the **Arduino's** 5V pin and the second terminal (middle pin) to the **LCD's** pin 3 and the third terminal to the **Arduino's** GND pin. Next, power up the **Arduino**. You will notice that the backlight on the **LCD** turns ON.

Module 1

Introduction to IoT

INTERNET OF THINGS

A Hands-On Approach

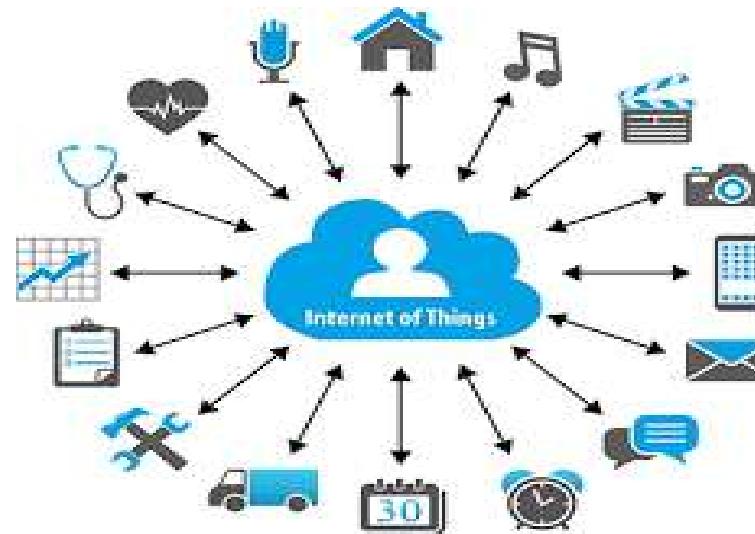


Arshdeep Bahga • Vijay Madisetti

Outline

- Definition of IoT
- Characteristics of IoT
- Physical design of IoT
- Logical design of IoT
- IoT protocols

IoT



IoT

- Internet Of Things is Fully Networked and Connected Devices sending analytics data back to cloud or data center.
- The definition of Internet of things is that it is the network in which every object or thing is provided unique identifier and data is transferred through a network without any verbal communication.
- Scope of IoT is not just limited to just connecting things to the internet, but it allows these things to communicate and exchange data, process them as well as control them while executing applications.

Formal Definition of IoT

- A **dynamic global network** infrastructure with **self- configuring capabilities** based on standard and **interoperable communication protocols**, where physical and virtual “things” have **identities**, physical attributes, and use intelligent interfaces, and are seamlessly **integrated into information network** that communicate data with users and environments.

Characteristics of IoT

- **Dynamic Global network & Self-Adapting** : Adapt the changes w.r.t changing contexts
- **Self Configuring** : Eg. Fetching latest s/w updates without manual intervention.
- **Interoperable Communication Protocols** : Communicate through various protocols
- **Unique Identity** : Such as Unique IP Address or a URI
- **Integrated into Information Network** : This allows to communicate and exchange data with other devices to perform certain analysis.

Physical Design of IoT

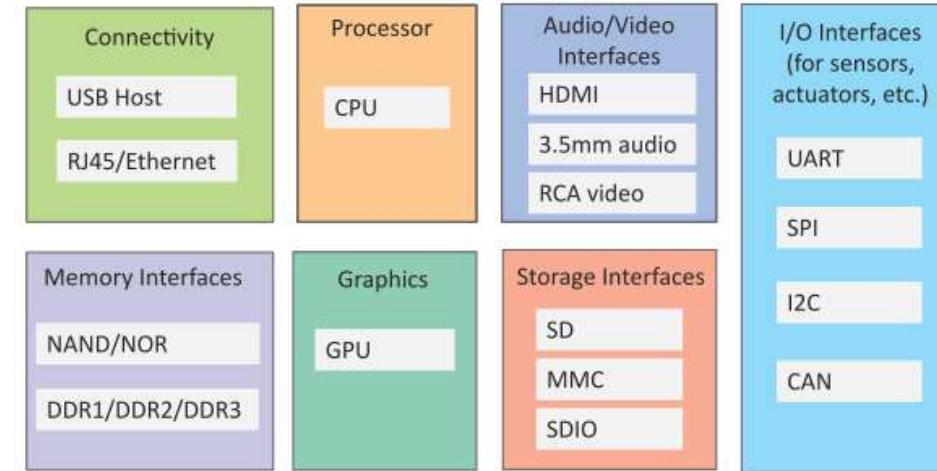
- Things in IoT
- IoT Protocols

Things in IoT

- Refers to IoT devices which have unique identities that can perform sensing, actuating and monitoring capabilities.
- IoT devices can exchange data with other connected devices or collect data from other devices and process the data either locally or send the data to centralized servers or cloud - based application back-ends for processing the data.

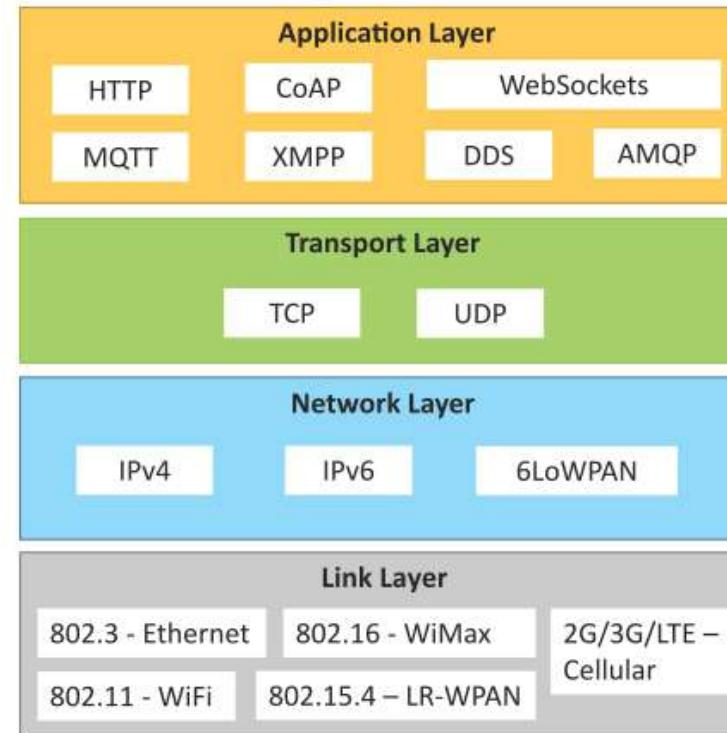
Generic Block Diagram of an IoT Device

- An IoT device may consist of several interfaces for connections to other devices, both wired and wireless.
 - I/O interfaces for sensors
 - Interfaces for internet connectivity
 - Memory and storage interfaces
 - Audio/video interfaces



IoT Protocols

- **Link Layer**
 - 802.3 – Ethernet
 - 802.11 – WiFi
 - 802.16 – WiMax
 - 802.15.4 – LR-WPAN
 - 2G/3G/4G
- **Network/Internet Layer**
 - IPv4
 - IPv6
 - 6LoWPAN
- **Transport Layer**
 - TCP
 - UDP
- **Application Layer**
 - HTTP
 - CoAP
 - WebSocket
 - MQTT
 - XMPP
 - DDS
 - AMQP



IoT Protocols...Link Layer...Ethernet

Sr.No	Standard	Shared medium
1	802.3	Coaxial Cable...10BASE5
2	802.3.i	Copper Twisted pair10BASE-T
3	802.3.j	Fiber Optic.....10BASE-F
4	802.3.ae	Fiber.....10Gbits/s

Data Rates are provided from 10Gbit/s to 40Gb/s and higher

IoT Protocols...Link Layer...WiFi

Sr.No	Standard	Operates in
1	802.11a	5 GHz band
2	802.11b and 802.11g	2.4GHz band
3	802.11.n	2.4/5 GHz bands
4	802.11.ac	5GHz band
5	802.11.ad	60Hz band

- Collection of Wireless LAN
- Data Rates from 1Mb/s to 6.75 Gb/s

IoT Protocols...Link Layer...WiMax

Sr.No	Standard	Data Rate
1	802.16m	100Mb/s for mobile stations 1Gb/s for fixed stations

- Collection of Wireless Broadband standards
- Data Rates from 1.5Mb/s to 1 Gb/s

IoT Protocols...Link Layer...LR-WPAN

- Collection of standards for low-rate wireless personal area networks
- Basis for high level communication protocols such as Zigbee
- Data Rates from 40Kb/s to 250Kb/s
- Provide low-cost and low-speed communication for power constrained devices
- 802.15.4-standard

IoT Protocols...Link Layer...2G/3G/4G –Mobile Communication

Sr.No	Standard	Operates in
1	2G	GSM-CDMA
2	3G	UMTS and CDMA 2000
3	4G	LTE

- Data Rates from 9.6Kb/s (for 2G) to up to 100Mb/s (for 4G)

IoT Protocols...Network/Internet Layer

- Responsible for sending of IP datagrams from source to destination network
- Performs the host addressing and packet routing
- Host identification is done using hierarchical IP addressing schemes such as IPV4 or IPV6

IoT Protocols...Network Layer

- IPV4
 - Used to identify the devices on a network using hierarchical addressing scheme
 - Uses 32-bit address scheme
- IPV6
 - Uses 128-bit address scheme
- 6LoWPAN (IPV6 over Low power Wireless Personal Area Network)
 - Used for devices with limited processing capacity
 - Operates in 2.4 Ghz
 - Data Rates of 250Kb/s

IoT Protocols...Network Layer

IP Routing on Subnets

Basic Subnet Masks

Class	Subnet Mask
A	255.0.0.0
B	255.255.0.0
C	255.255.255.0

IoT Protocols...Network Layer

Subnetting IP Addresses

Creating Subnets on a Class A Network

Network Address: 10.0.0.0
Number of Subnets: 30
Binary Equivalent: 00001010 00000000 00000000 00000000

First Node Octet
↓
Decimal Values of Octet: 128 64 32 16 8 4 2 1
↓

5 bits are required to create 30 subnets
 $1 + 2 + 4 + 8 + 16 - 1 = 30$ Subnets

Add 5 high-order bits to create the subnet mask
 $128 + 64 + 32 + 16 + 8 = 248$
New Subnet Mask: 255.248.0.0

IoT Protocols...Network Layer

Subnetting IP Addresses

Calculating IP Subnet Ranges on Class A Network

Subnet#	Start Address	End Address
1	10.8.0.1	10.15.255.254
2	10.16.0.1	10.23.255.254
3	10.24.0.1	10.31.255.254
4	10.32.0.1	10.39.255.254
5	10.40.0.1	10.47.255.254
6	10.48.0.1	10.55.255.254
7	10.56.0.1	10.63.255.254
8	10.64.0.1	10.71.255.254
9	10.72.0.1	10.79.255.254
10	10.80.0.1	10.87.255.254

IoT Protocols...Network Layer

Subnetting IP Addresses

Creating Subnets on a Class B Network

Network Address: 180.10.0.0

Number of Subnets: 6

Binary Equivalent 10110100 00001000 00000000 00000000

First Node Octet

Decimal Values of Octet: 128 64 32 16 8 4 2 1

3 bits are required to create 6 subnets
 $1 + 2 + 4 - 1 = 6$ Subnets

Add 3 high-order bits to create the subnet mask
 $128 + 64 + 32 = 224$
New Subnet Mask: 255.255.224.0

IoT Protocols...Network Layer

Subnetting IP Addresses

Calculating IP Subnet Ranges on Class B Network

Subnet #	Start Address	End Address
1	180.10.32.1	180.10.63.254
2	180.10.64.1	180.10.95.254
3	180.10.96.1	180.10.127.254
4	180.10.128.1	180.10.159.254
5	180.10.160.1	180.10.191.254
6		180.10.192.1
	180.10.223.254	

IoT Protocols...Network Layer

Subnetting IP Addresses

Creating Subnets on a Class C Network

Network Address: 200.10.44.0
Number of Subnets: 2
Binary Equivalent 11001000 00001010 10110000 00000000

First Node Octet

Decimal Values of Octet 128 64 32 16 8 4 2 1

2 bits are required to create 2 subnets
 $1 + 2 - 1 = 2$ Subnets

Add 2 high-order bits to create the subnet mask
 $128 + 64 = 192$
New Subnet Mask: 255.255.225.192

IoT Protocols...Network Layer

Subnetting IP Addresses

Calculating IP Subnet Ranges on Class C Network

Subnet#	Subnetwork Address	Start Address	End Address	Broadcast Address
<hr/>				
1	200.10.44.64	200.10.44.65	200.10.44.126	
2	0 . 1 0	. 4 4	. 1 2	7
2	200.10.44.128	200.10.44.129	200.10.44.190	
	200.10.44.191			

IoT Protocols...Transport Layer

- Provide end-to-end message transfer capability independent of the underlying network
- It provides functions such as error control, segmentation, flow-control and congestion control

IoT Protocols...TCP

- Transmission Control Protocol
- Connection Oriented
- Ensures Reliable transmission
- Provides Error Detection Capability to ensure no duplicacy of packets and retransmit lost packets
- Flow Control capability to ensure the sending data rate is not too high for the receiver process
- Congestion control capability helps in avoiding congestion which leads to degradation of n/w performance



IoT Protocols...UDP

- User Datagram Protocol
- Connectionless
- Does not ensure Reliable transmission
- Does not do connection before transmitting
- Does not provide proper ordering of messages
- Transaction oriented and stateless



IoT Protocols...Application Layer...Hyper Text Transfer Protocol

- Forms foundation of World Wide Web(WWW)
- Includes commands such as GET,PUT, POST, HEAD, OPTIONS, TRACE..etc
- Follows a request-response model
- Uses Universal Resource Identifiers(URIs) to identify HTTP resources



IoT Protocols...Application Layer...CoAP

- Constrained Application Protocol
- Used for Machine to machine (M2M) applications meant for constrained devices and n/w's
- Web transfer protocol for IoT and uses request-response model
- Uses client –server architecture
- Supports methods such as GET,POST, PUT and DELETE



IoT Protocols...Application Layer...WebSocket

- Allows full-duplex communication over single socket
- Based on TCP
- Client can be a browser, IoT device or mobile application

IoT Protocols...Application Layer...MQTT

- **Message Queue Telemetry Transport**, light-weight messaging protocol
- Based on publish-subscribe model
- Well suited for constrained environments where devices have limited processing, low memory and n/w bandwith requirement

IoT Protocols...Application Layer...XMPP

- Extensible messaging and presence protocol
- For Real time communication and streaming XML data between n/w entities
- Used for Applications such as Multi-party chat and voice/video calls.
- Decentralized protocol and uses client server architecture.

IoT Protocols...Application Layer...DDS

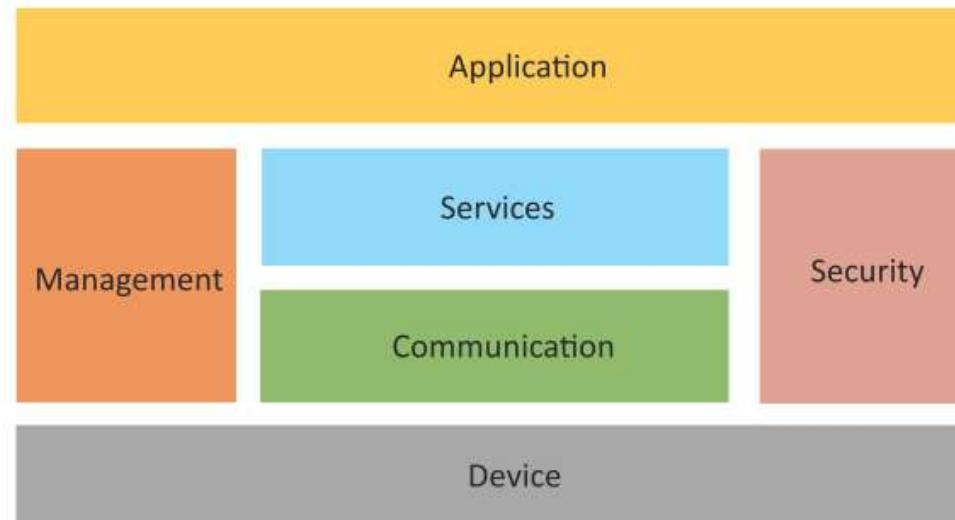
- Data Distribution service is a data-centric middleware standard for device-to-device or machine-to-machine communication.
- Publish subscribe model where publishers create topics to which subscribers can use.
- Provides Quality-of-service control and configurable reliability.

IoT Protocols...Application Layer...AMQP

- Advanced Messaging Queuing Protocol used for business messaging.
- Supports both point-to-point and publisher/subscriber models, routing and queuing
- Broker here receives messages from publishers and route them over connections to consumers through messaging queues.

Logical Design of IoT

- Logical design of an IoT system refers to an abstract representation of the entities and processes without going into the low-level specifics of the implementation.
- An IoT system comprises a number of functional blocks that provide the system the capabilities for identification, sensing, actuation, communication and management.

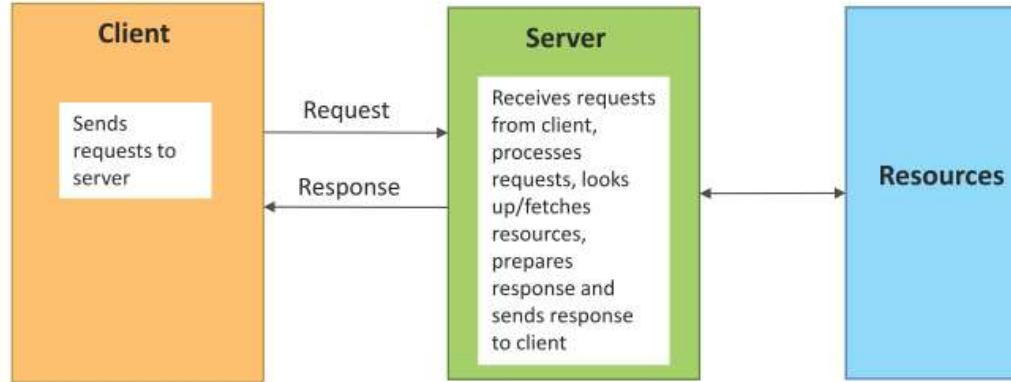


Logical Design of IoT

- Device : Devices such as sensing, actuation, monitoring and control functions.
- Communication : IoT Protocols
- Services like device monitoring, device control services, data publishing services and device discovery
- Management : Functions to govern the system
- Security : Functions as authentication, authorization, message and content integrity, and data security
- Applications

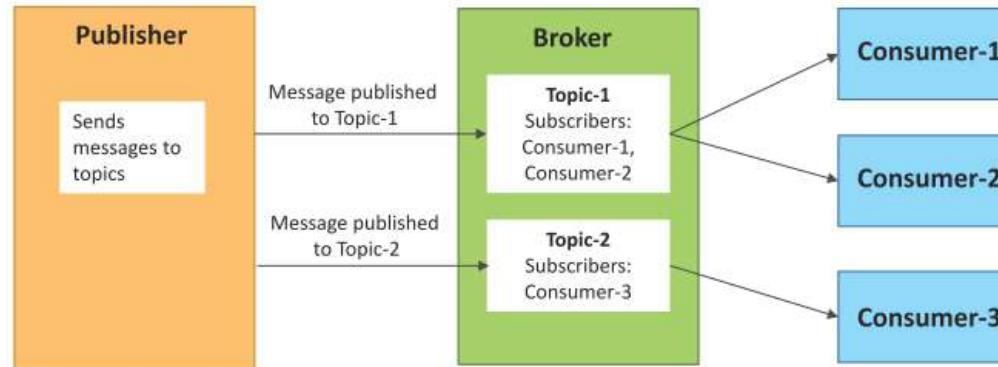
Request–Response Communication Model

- Request–Response is a communication model in which the client sends requests to the server and the server responds to the requests.
- When the server receives a request, it decides how to respond, fetches the data, retrieves resource representations, prepares the response and then sends the response to the client.
- Stateless communication model and each request-response pair is independent of others



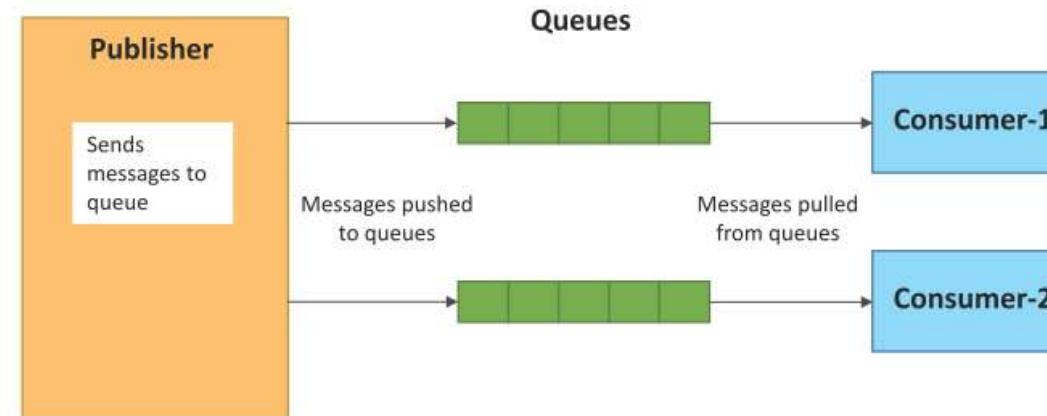
Publish–Subscribe Communication Model

- Publish–Subscribe is a communication model that involves publishers, brokers and consumers.
- Publishers are the source of data. Publishers send the data to the topics which are managed by the broker. Publishers are not aware of the consumers.
- Consumers subscribe to the topics which are managed by the broker.
- When the broker receives data for a topic from the publisher, it sends the data to all the subscribed consumers.



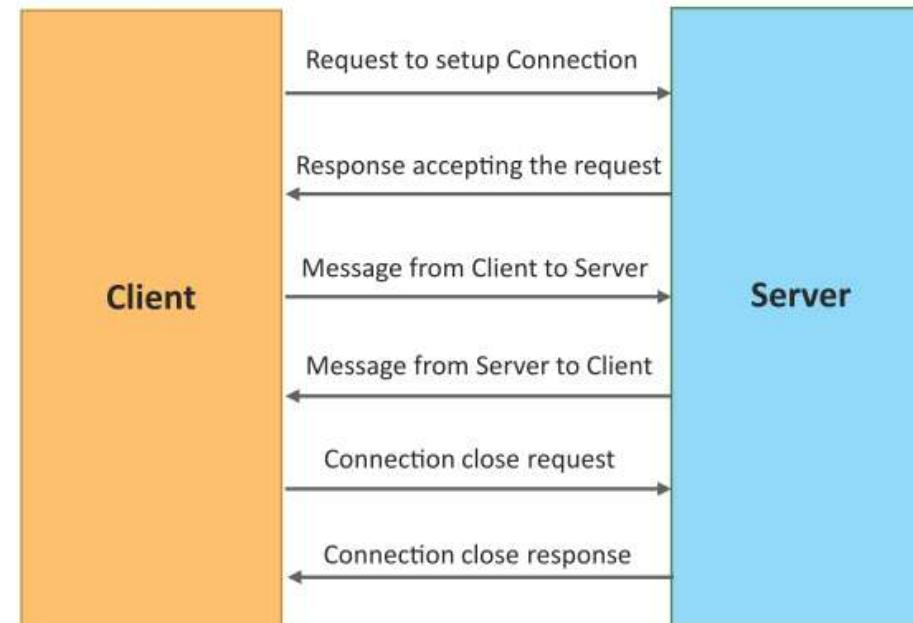
Push–Pull Communication Model

- Push–Pull is a communication model in which the data producers push the data to queues and the consumers pull the data from the queues. Producers do not need to be aware of the consumers.
- Queues help in decoupling the messaging between the producers and consumers.
- Queues also act as a buffer which helps in situations when there is a mismatch between the rate at which the producers push data and the rate at which the consumers pull data.



Exclusive Pair Communication Model

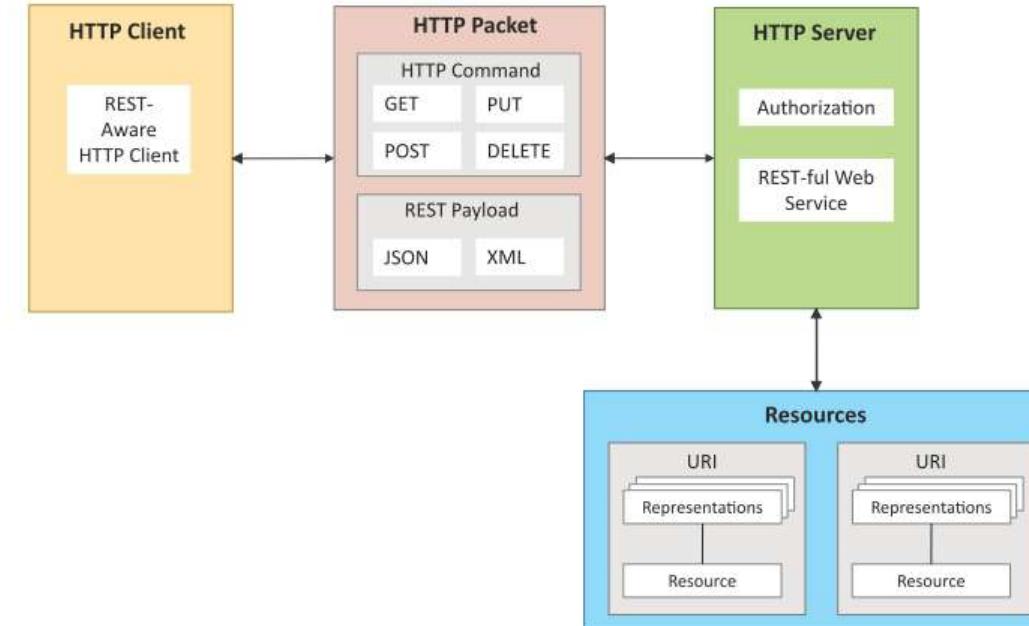
- Exclusive Pair is a bidirectional, fully duplex communication model that uses a persistent connection between the client and the server.
- Once the connection is set up it, remains open until the client sends a request to close the connection.
- Client and server can send messages to each other after connection setup.



IOT Communication APIs

REST-based Communication APIs

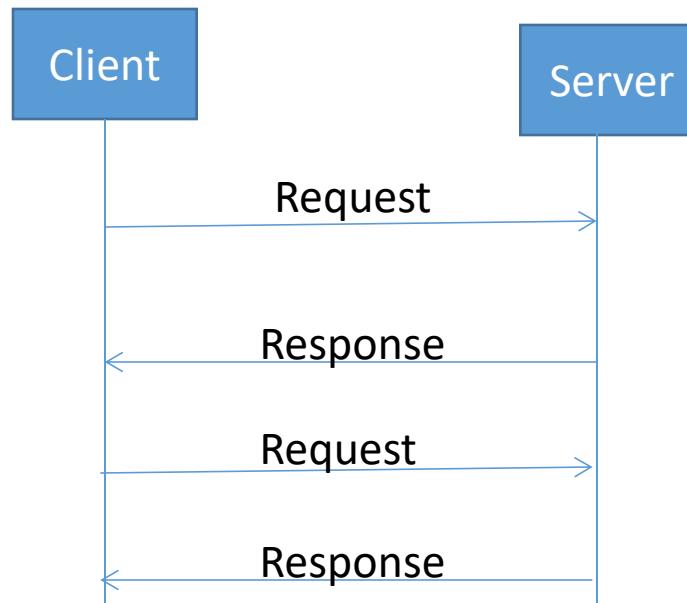
- Representational State Transfer (REST) is a set of architectural principles by which you can **design web services and web APIs** that focus on a system's resources and how resource states are addressed and transferred.
- REST APIs follow the **request–response communication model**.
- REST architectural constraints apply to the components, connectors and data elements within a distributed hypermedia system.



REST-based Communication APIs

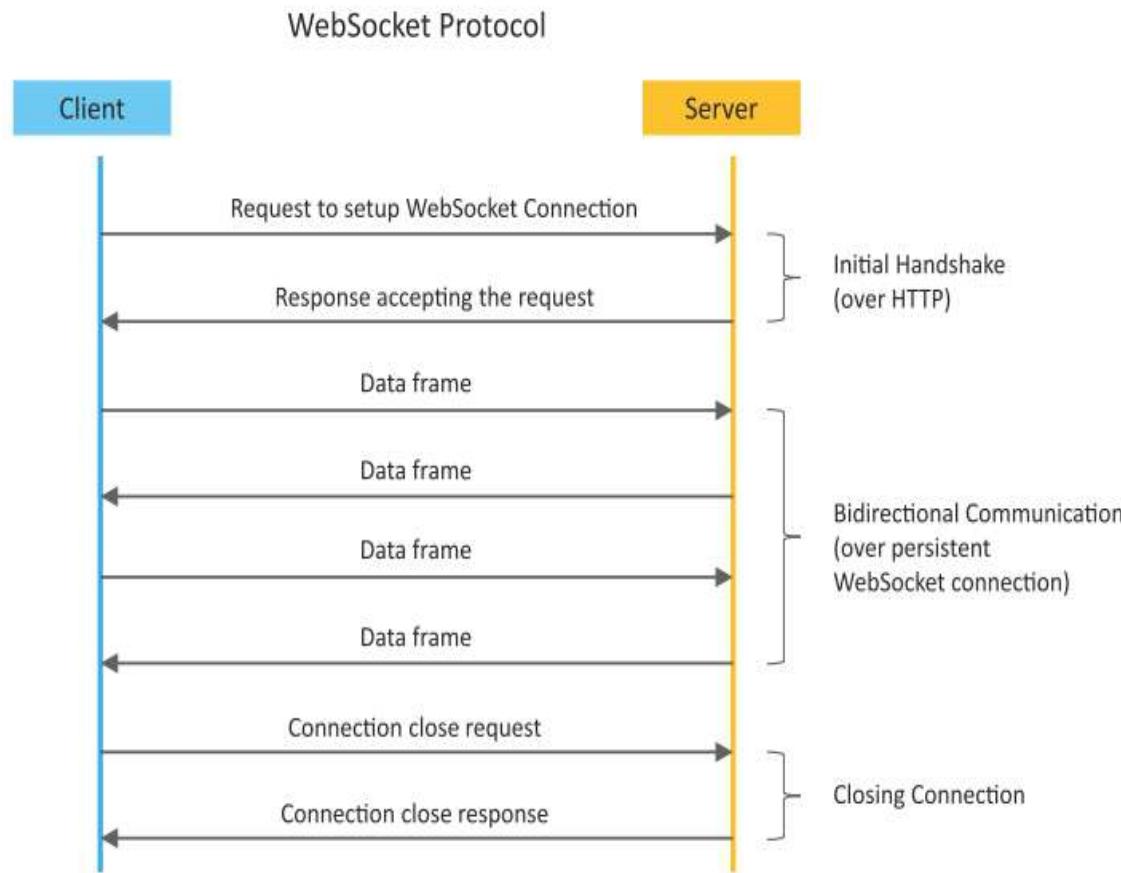
Constraints

- Client – Server
- Stateless
- Cacheable
- Layered System
- Uniform Interface
- Code on demand



WebSocket-based Communication APIs

- WebSocket APIs allow **bi-directional, full duplex communication** between clients and servers.
- WebSocket APIs follow the **exclusive pair communication model**.



Difference between REST and WebSocket-based Communication APIs

Comparison Based on	REST	Websocket
State	Stateless	Stateful
Directional	Unidirectional	Bidirectional
Req-Res/Full Duplex	Follow Request Response Model	Exclusive Pair Model
TCP Connections	Each HTTP request involves setting up a new TCP Connection	Involves a single TCP Connection for all requests
Header Overhead	Each request carries HTTP Headers, hence not suitable for real-time	Does not involve overhead of headers.
Scalability	Both horizontal and vertical are easier	Only Vertical is easier

IoT Enabling Technologies

- Wireless Sensor Network
- Cloud Computing
- Big Data Analytics
- Embedded Systems



WSN

- **Distributed Devices with sensors** used to monitor the environmental and physical conditions
- Consists of several **end-nodes acting as routers or coordinators** too
- **Coordinators collects data** from all nodes / **acts as gateway** that connects WSN to internet
- **Routers route the data packets** from end nodes to coordinators.

Example of WSNs in IoT & Protocols used

Example

- Weather monitoring system
- Indoor Air quality monitoring system
- Soil moisture monitoring system
- Surveillance systems
- Health monitoring systems

Protocols

- Zigbee

- Wsn ENABLESD BY 802.15.4

Operates at 2.4Ghz,250KB/s range 10 to 100 mts

WSN is selforganized

Cloud Computing

- **Deliver applications and services over internet**
- Provides computing, networking and storage resources on demand
- Cloud computing performs services such as IaaS, PaaS and SaaS
- IaaS : Rent Infrastructure
- PaaS : supply an on-demand environment for developing, testing, delivering and managing software applications.
- SaaS : method for delivering software applications over the Internet, on demand and typically on a subscription basis.

Big Data Analytics

- Collection of data whose volume, velocity or variety is too large and difficult to store, manage, process and analyze the data using traditional databases.
- It involves data cleansing, processing and visualization
- Lots of data is being collected and warehoused
 - Web data, e-commerce
 - purchases at department/ grocery stores
 - Bank/Credit Card transactions
 - Social Network



Big Data Analytics

Variety Includes different types of data

- Structured
- Unstructured
- SemiStructured
- All of above

Big Data Analytics

Velocity Refers to speed at which data is processed

- Batch
- Real-time
- Streams

Big Data Analytics

Volume refers to the amount of data

- Terabyte
- Records
- Transactions
- Files
- Tables

Thank You

Module 2

IOT COMMUNICATION MODEL AND PROTOCOLS



Outline

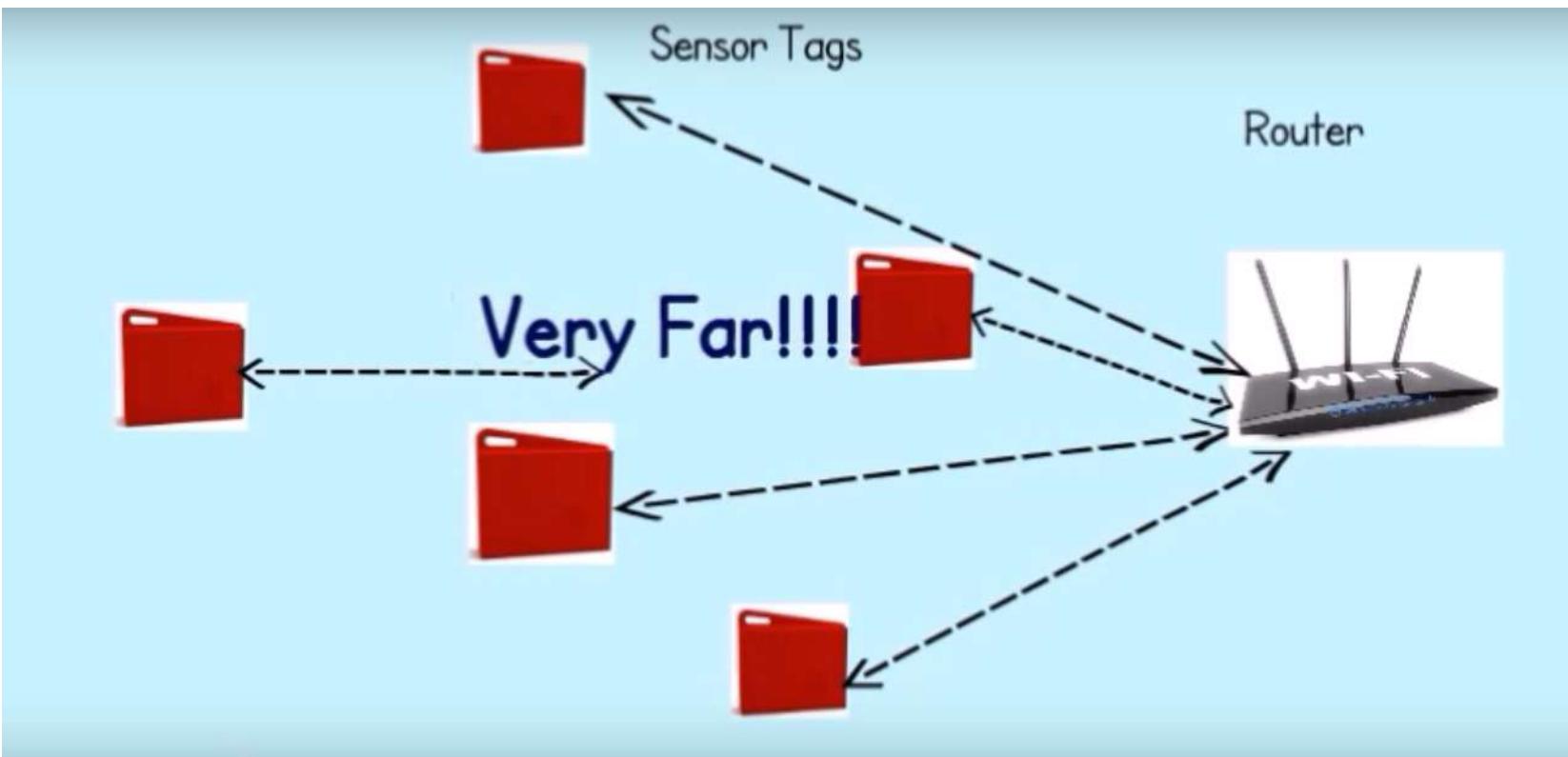
Connectivity Protocols

- 6LoWPAN
- IEEE 802.15.4
- Zigbee
- Wireless HART
- Z-Wave
- ISA 100
- NFC

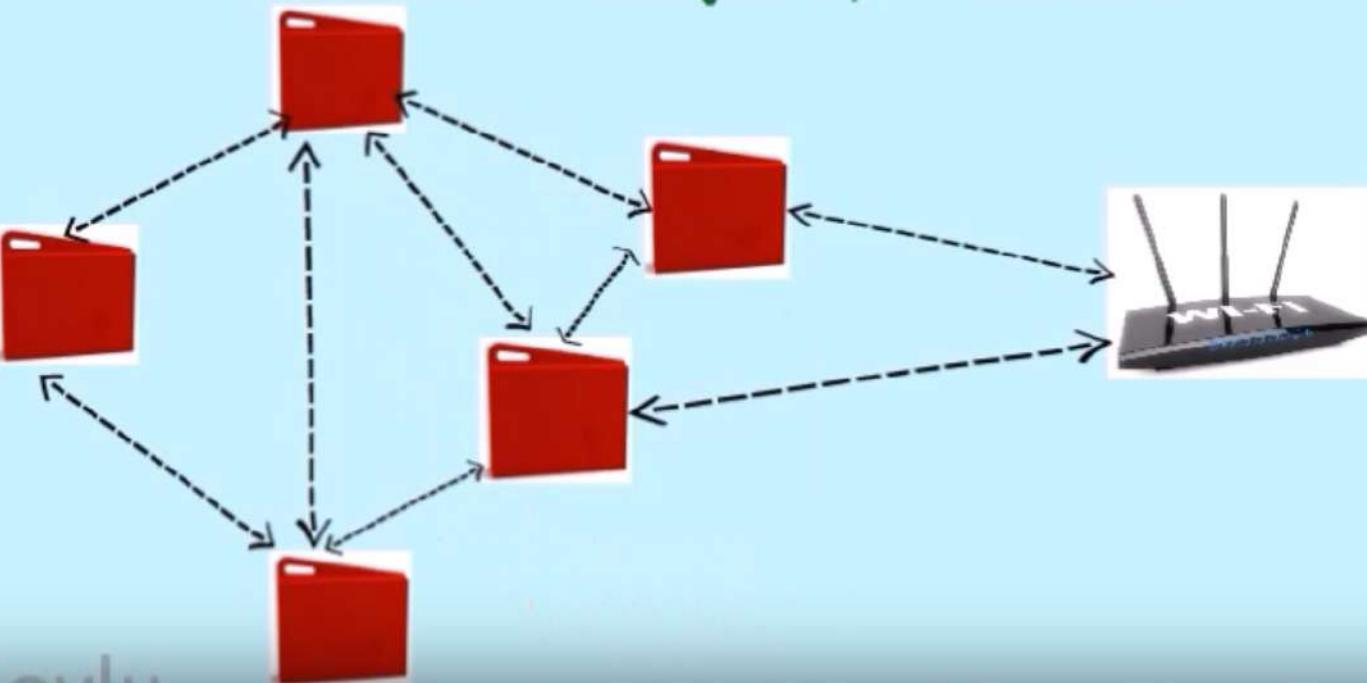
Outline

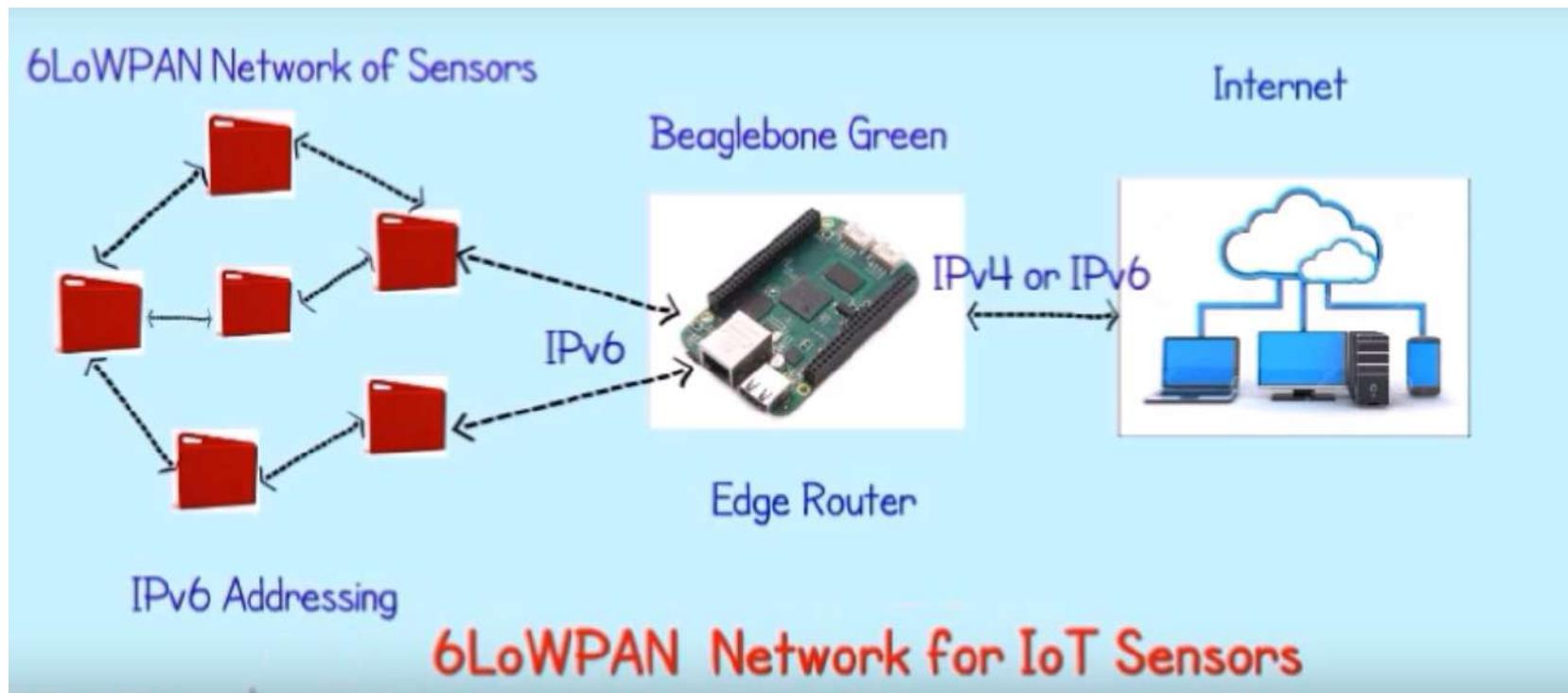
Data Protocols:

- Message Queue Telemetry Transport (MQTT)
- Constrained Application Protocol (CoAP)
- Advanced Message Queuing Protocol (AMQP)
- XMPP – Extensible Messaging and Presence Protocol.



Alternate Paths - Forming a complete Mesh Network





6LoWPAN

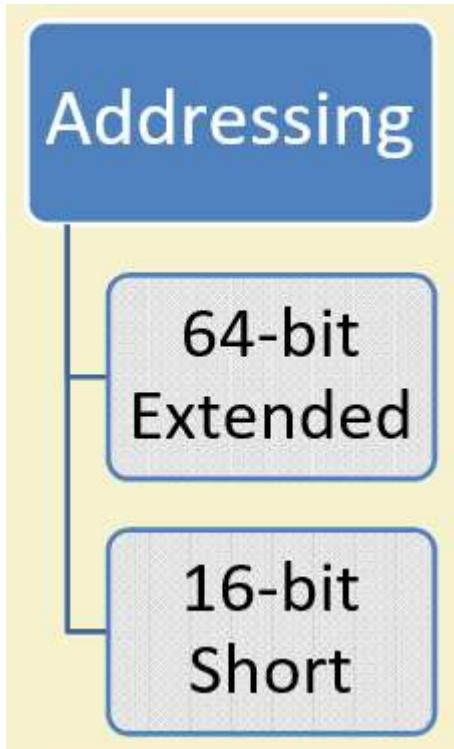
Introduction

- Low-power Wireless Personal Area Networks over IPv6.
- Allows for the smallest devices with limited processing ability to transmit information wirelessly using an Internet protocol.
- Allows low-power devices to connect to the Internet.
- Created by the Internet Engineering Task Force (IETF) - RFC 5933 and RFC 4919.

Features of 6LoWPANs

- Allows IEEE 802.15.4 radios to carry 128-bit addresses of Internet Protocol version 6 (IPv6).
- Header compression and address translation techniques allow the IEEE 802.15.4 radios to access the Internet.
- IPv6 packets compressed and reformatted to fit the IEEE 802.15.4 packet format.
- Uses include IoT, Smart grid, and M2M applications.

Addressing in 6LoWPAN



64-bit addresses: globally unique

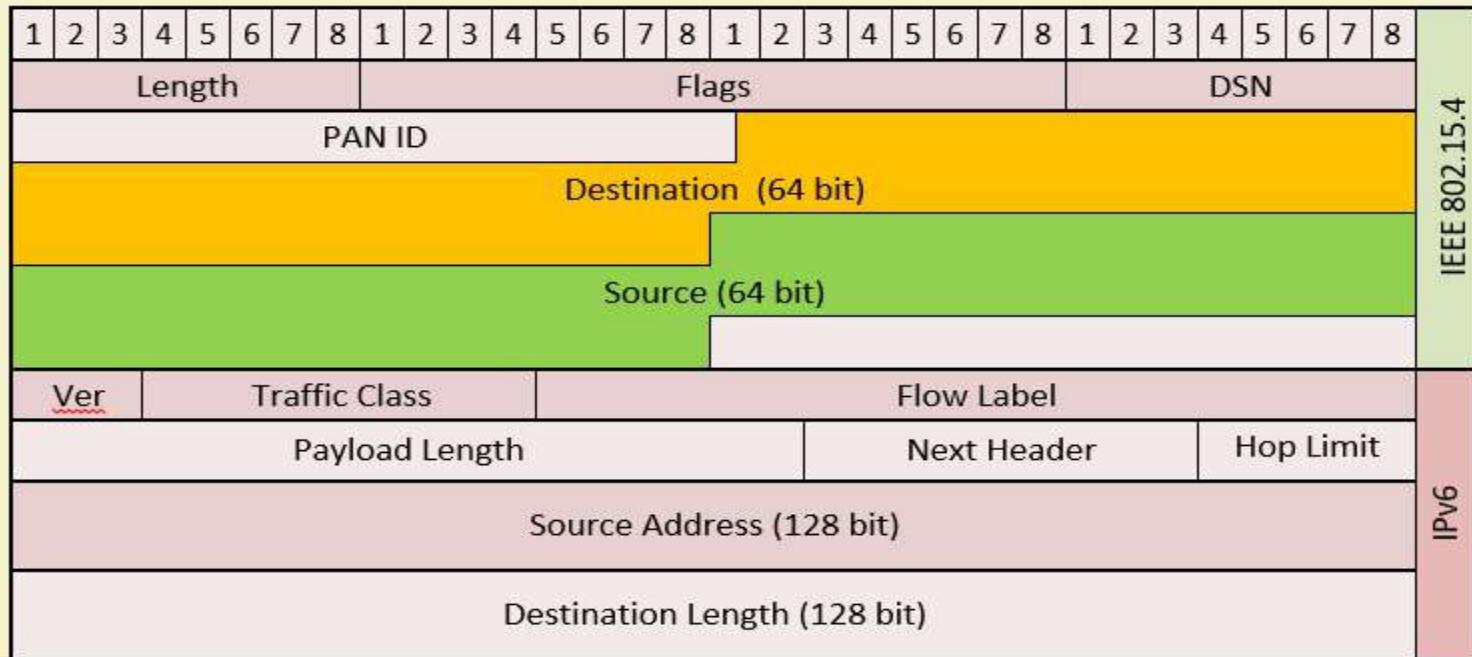
16 bit addresses: PAN specific; assigned by PAN coordinator

IPv6 multicast not supported by 802.15.4

IPv6 packets carried as link layer broadcast frames

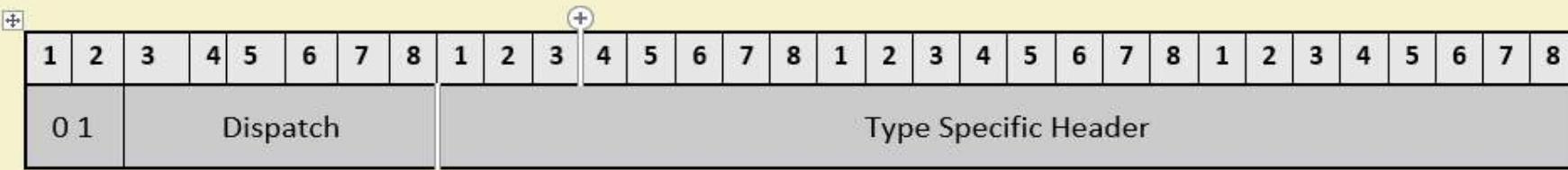
6LoWPAN

6LowPAN Packet Format



6LoWPAN

Header Type: Dispatch Header



- **Dispatch:** Initiates communication
- **0,1:** Identifier for Dispatch Type
- **Dispatch:**
 - 6 bits
 - Identifies the next header type
- **Type Specific Header:**
 - Determined by Dispatch header

6LoWPAN

Header Type: Mesh Addressing Header

1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
1	0	V	F	Hops Left				Originator Address								Final Address							

- **1,0:** ID for Mesh Addressing Header
- **V:** ‘0’ if originator is 64-bit extended address, ‘1’ if 16-bit address
- **F:** ‘0’ if destination is 64-bit addr., ‘1’ if 16-bit addr..
- **Hops Left:** decremented by each node before sending to next hop

6LoWPAN

Header Type: Fragmentation Header

1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
1	1	0	0	Datagram Size								Datagram Tag											

(a) First Fragment

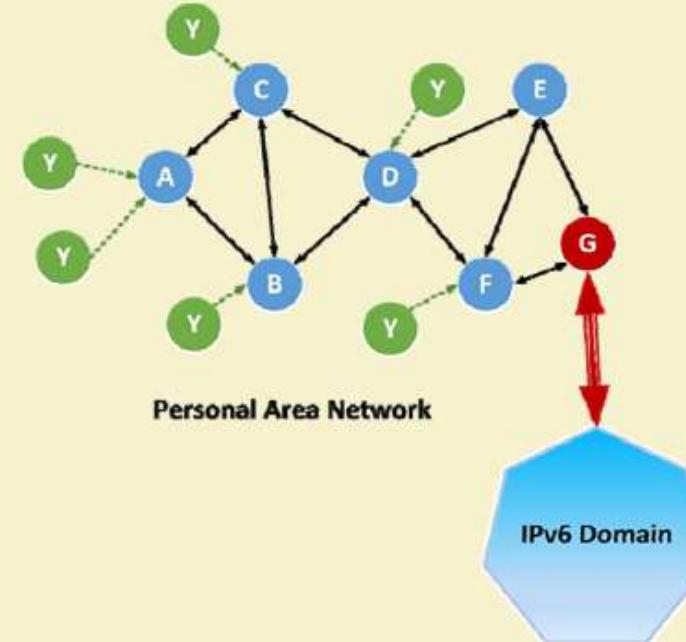
1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
1	1	0	0	Datagram Size								Datagram Tag											
Datagram Offset																							

(b) Subsequent Fragment

6LoWPAN

6LoWPAN Routing Considerations

- ✓ Mesh routing within the PAN space.
- ✓ Routing between IPv6 and the PAN domain
- ✓ Routing protocols in use:
 - LOADng
 - RPL



6LoWPAN

- LOADng Routing
- Derived from AODV and extended for use in IoT.
- Basic operations of LOADng include:
 1. Generation of **Route Requests (RREQs)** by a LOADng Router (originator) for discovering a route to a destination,
 2. **Forwarding of such RREQs** until they reach the destination LOADng Router,
 3. Generation of **Route Replies (RREPs)** upon receipt of an RREQ by the indicated destination, and unicast hop-by-hop forwarding of these RREPs towards the originator.

LOADng: Light weight On demand Adhoc Distance vector routing protocol, next generation

6LoWPAN

- If a route is detected to be broken, a **Route Error (RERR)** message is returned to the originator of that data packet to inform the originator about the route breakage.
- **Optimized flooding** is supported, reducing the overhead incurred by RREQ generation and flooding.
- Only the destination is permitted to respond to an RREQ.
- Intermediate LOADng Routers are explicitly prohibited from responding to RREQs, even if they may have active routes to the sought destination.
- RREQ/RREP messages generated by a given LOADng Router share a single unique, monotonically increasing sequence number.

6LoWPAN

- RPL Routing
- Distance Vector IPv6 **routing protocol for lossy and low power networks.**
- Maintains routing topology using low rate beaconing.
- Beaconing rate increases on detecting inconsistencies (e.g. node/link in a route is down).
- Routing information included in the datagram itself.
- **Proactive:** Maintaining routing topology.
- **Reactive:** Resolving routing inconsistencies.

6LoWPAN

- RPL separates packet processing and forwarding from the routing optimization objective, which helps in Low power Lossy Networks (LLN)
- RPL supports message confidentiality and integrity.
- Supports Data-Path Validation and Loop Detection
- Routing optimization objectives include
 - minimizing energy
 - minimizing latency
 - satisfying constraints (w.r.t node power, bandwidth, etc.)

6LoWPAN

- RPL operations require bidirectional links.
- In some LLN scenarios, those links may exhibit asymmetric properties.
- It is required that the reachability of a router be verified before the router can be used as a parent.

Functionality-based IoT Protocol Organization

- **Connectivity** (6LowPAN, RPL)
- **Identification** (EPC, uCode, IPv6, URIs)
- **Communication / Transport** (WiFi, Bluetooth, LPWAN)
- **Discovery** (Physical Web, mDNS, DNS-SD)
- **Data Protocols** (MQTT, CoAP, AMQP, Websocket, Node)
- **Device Management** (TR-069, OMA-DM)
- **Semantic** (JSON-LD, Web Thing Model)
- **Multi-layer Frameworks** (Alljoyn, IoTivity, Weave, Homekit)

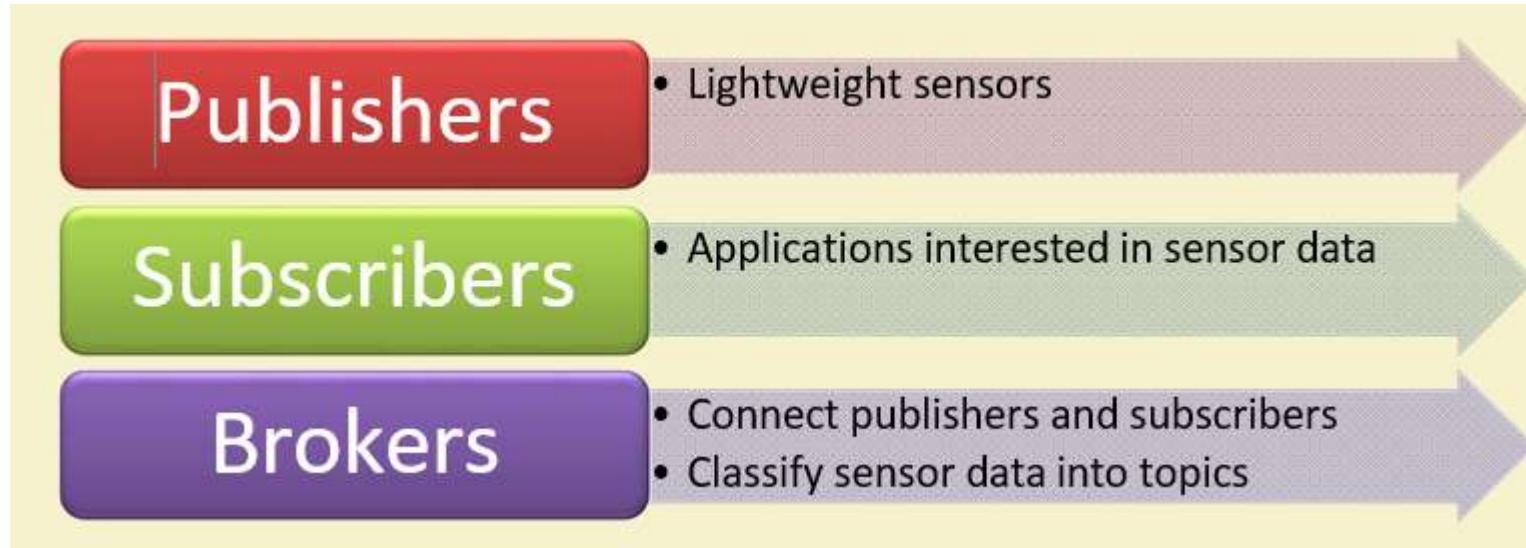
MQTT

- Introduction
- **Message Queue Telemetry Transport.**
- ISO standard (ISO/IEC PRF 20922).
- It is a publish-subscribe-based lightweight messaging protocol for use in conjunction with the TCP/IP protocol.
- MQTT was introduced by IBM in 1999 and standardized by OASIS in 2013.
- Designed to provide connectivity (mostly embedded) between applications and middle-wares on one side and networks and communications on the other side.

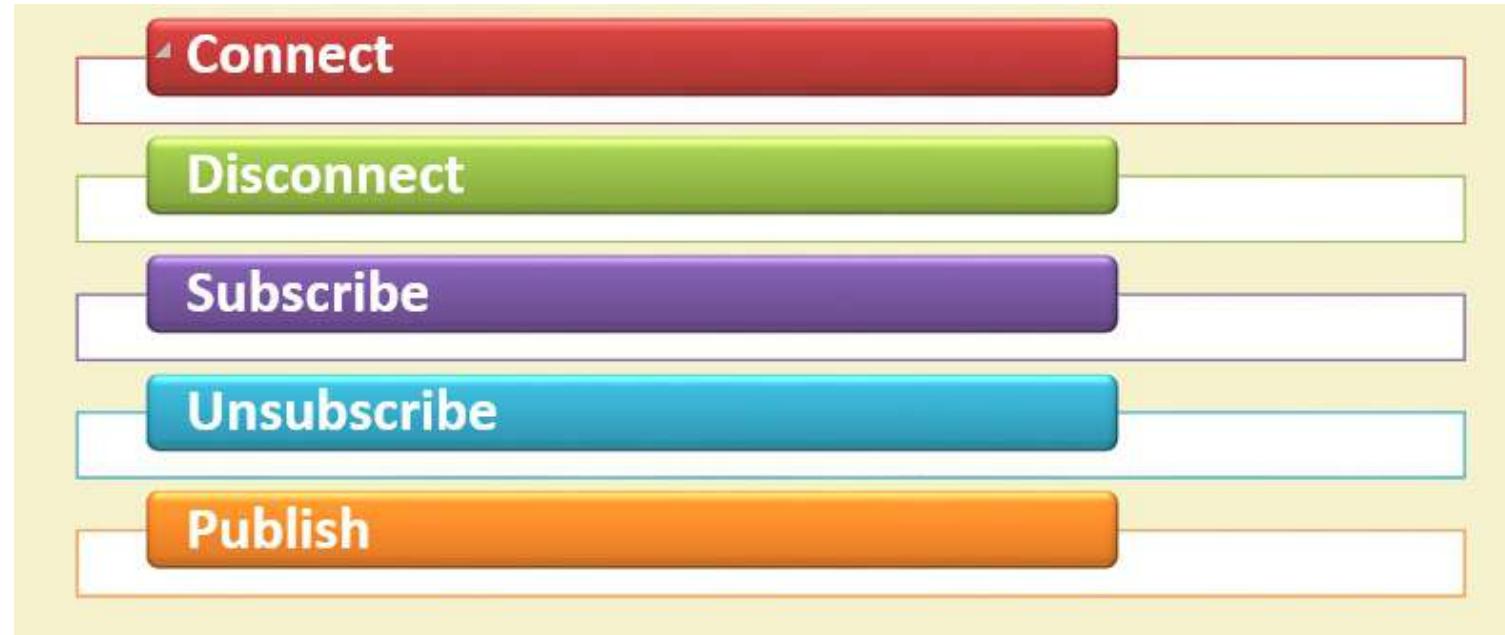
MQTT

- A message broker controls the publish-subscribe messaging pattern.
- A topic to which a client is subscribed is updated in the form of messages and distributed by the message broker.
- Designed for:
- Remote connections
- Limited bandwidth
- Small-code footprint

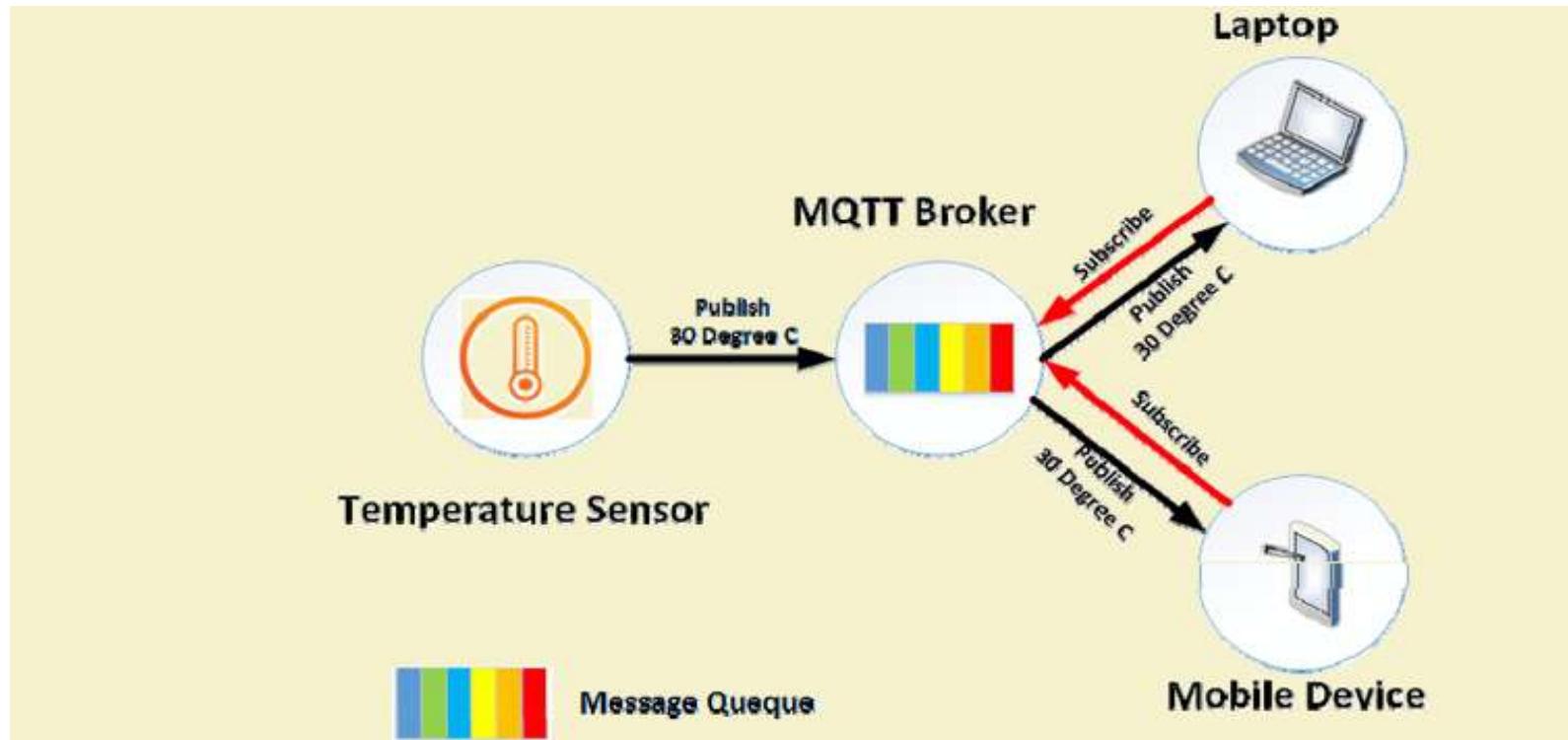
MQTT Components



MQTT Methods



MQTT



MQTT Communication

- The protocol uses a **publish/subscribe** architecture (HTTP uses a request/response paradigm).
- Publish/subscribe is **event-driven** and enables messages to be pushed to clients.
- The central **communication point is the MQTT broker**, which is in charge of dispatching all messages between the senders and the rightful receivers.
- Each client that publishes a message to the broker, includes a **topic** into the message. The **topic is the routing information for the broker**.

MQTT Communication

- Each client that wants to receive messages subscribes to a certain topic and the broker delivers all messages with the matching topic to the client.
- Therefore the clients don't have to know each other. They only communicate over the topic.
- This architecture enables highly scalable solutions without dependencies between the data producers and the data consumers.

MQTT Topics

- A topic is a **simple string** that can have more hierarchy levels, which are separated by a slash.
- A sample topic for sending temperature data of the living room could be *house/living-room/temperature*.
- On one hand the client (e.g. mobile device) can subscribe to the exact topic or on the other hand, it can use a **wildcard**.

MQTT Topics

- The subscription to *house/+/temperature* would result in all messages sent to the previously mentioned topic *house/living-room/temperature*, as well as any topic with an arbitrary value in the place of living room, such as *house/kitchen/temperature*.
- The plus sign is a **single level wild card** and only allows arbitrary values for one hierarchy.
- If more than one level needs to be subscribed, such as, the entire sub-tree, there is also a **multilevel wildcard (#)**.
- It allows to subscribe to all underlying hierarchy levels.
- For example *house/#* is subscribing to all topics beginning with *house*.

MQTT Applications

- **Facebook Messenger** uses MQTT for online chat.
- **Amazon Web Services** use Amazon IoT with MQTT.
- **Microsoft Azure IoT Hub** uses MQTT as its main protocol for telemetry messages.
- The **EVRYTHNG IoT platform** uses MQTT as an M2M protocol for millions of connected products.
- **Adafruit** launched a free MQTT cloud service for IoT experimenters called Adafruit IO.

Constrained Application Protocol

- Introduction

- CoAP – **Constrained Application Protocol**.
- **Web transfer protocol** for use with constrained nodes and networks.
- **Designed for Machine to Machine (M2M)** applications such as smart energy and building automation.
- Based on **Request-Response model** between end-points
- Client-Server interaction is **asynchronous over a datagram oriented transport protocol** such as UDP

Constrained Application Protocol

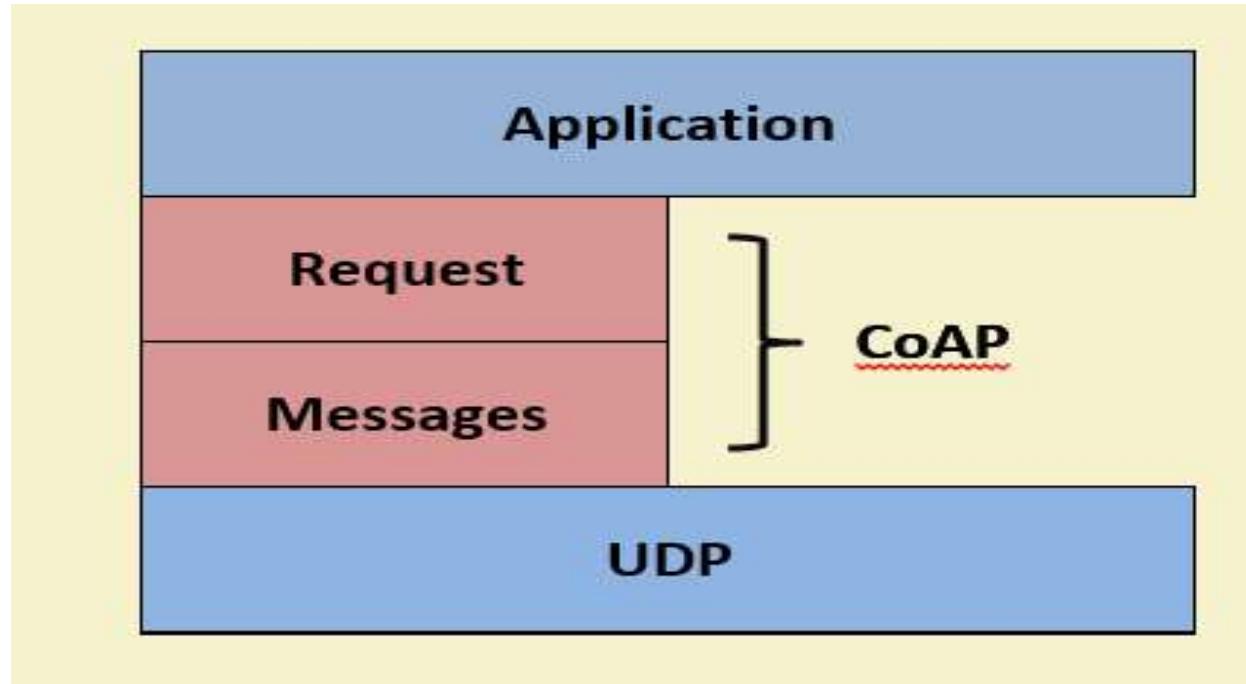
- The Constrained Application Protocol (CoAP) is a session layer protocol designed by IETF Constrained RESTful Environment (CoRE) working group to provide lightweight RESTful (HTTP) interface.
- Representational State Transfer (REST) is the standard interface between HTTP client and servers.
- Lightweight applications such as those in IoT, could result in significant overhead and power consumption by REST.
- CoAP is designed to enable low-power sensors to use RESTful services while meeting their power constraintshtweight RESTful (HTTP) interface.

CoAP

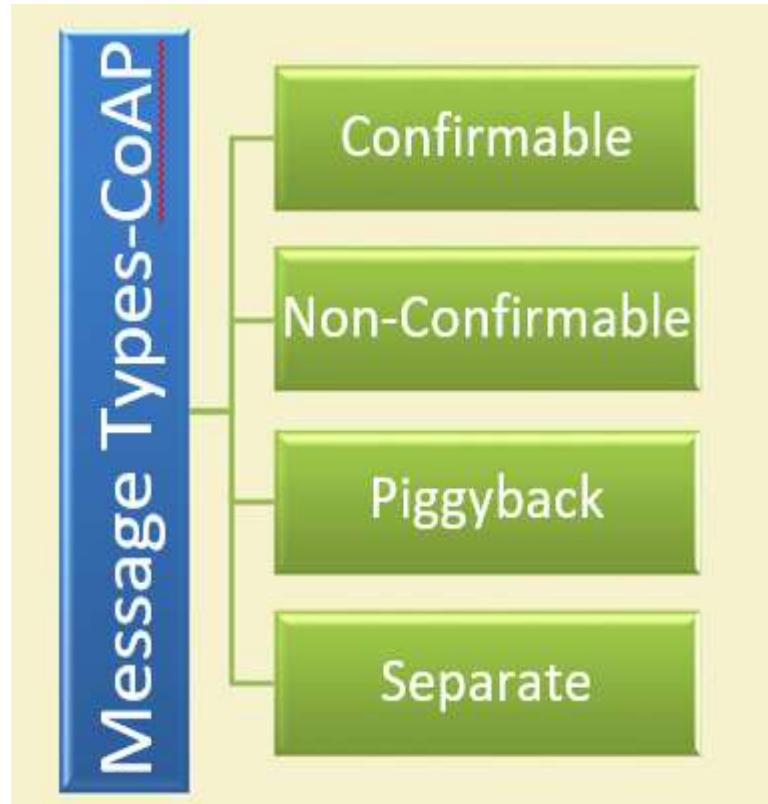
Constrained Application Protocol

- Built over UDP, instead of TCP (which is commonly used with HTTP) and has a light mechanism to provide reliability
- CoAP architecture is divided into two main sub-layers:
 - Messaging
 - Request/response
 - The messaging sub-layer is responsible for reliability and duplication of messages, while the request/response sub-layer is responsible for communication.
 - CoAP has four messaging modes:
 - Confirmable
 - Non-confirmable
 - Piggyback
 - Separate

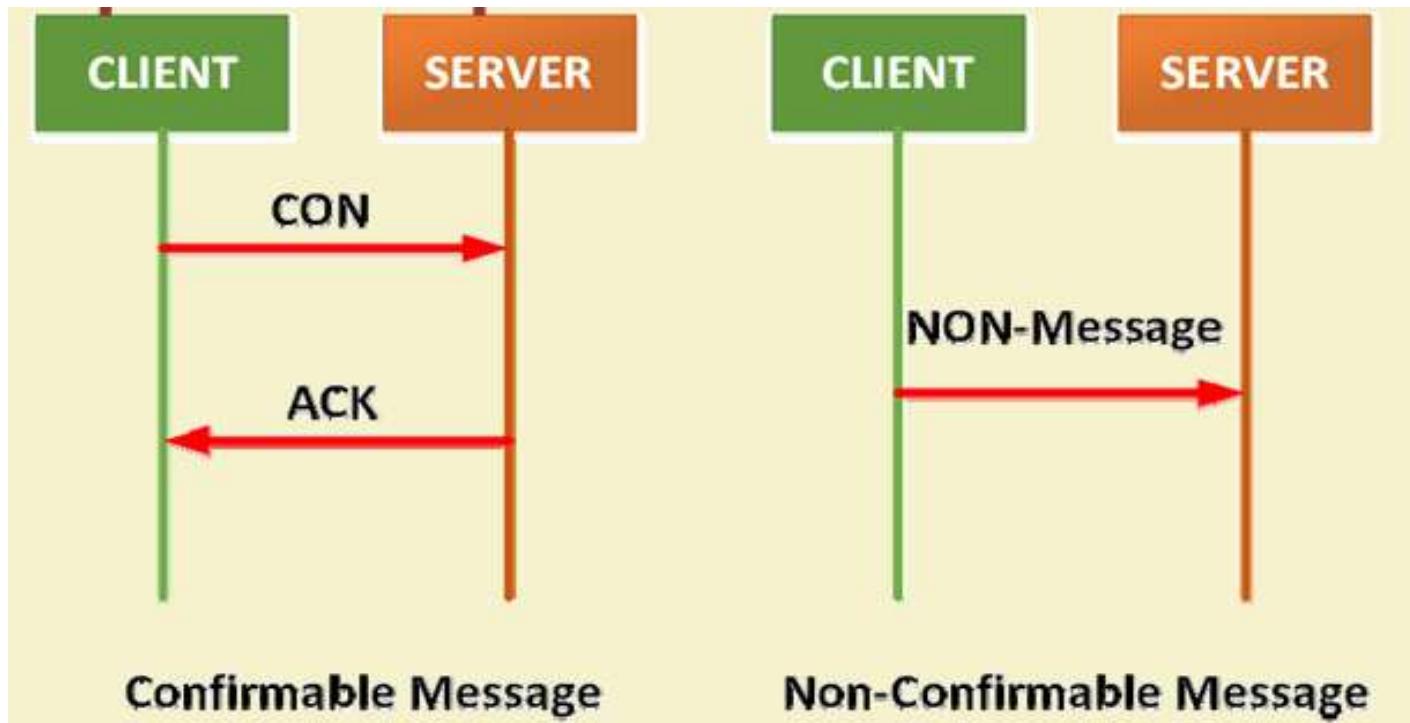
CoAP Position



CoAP Message Type



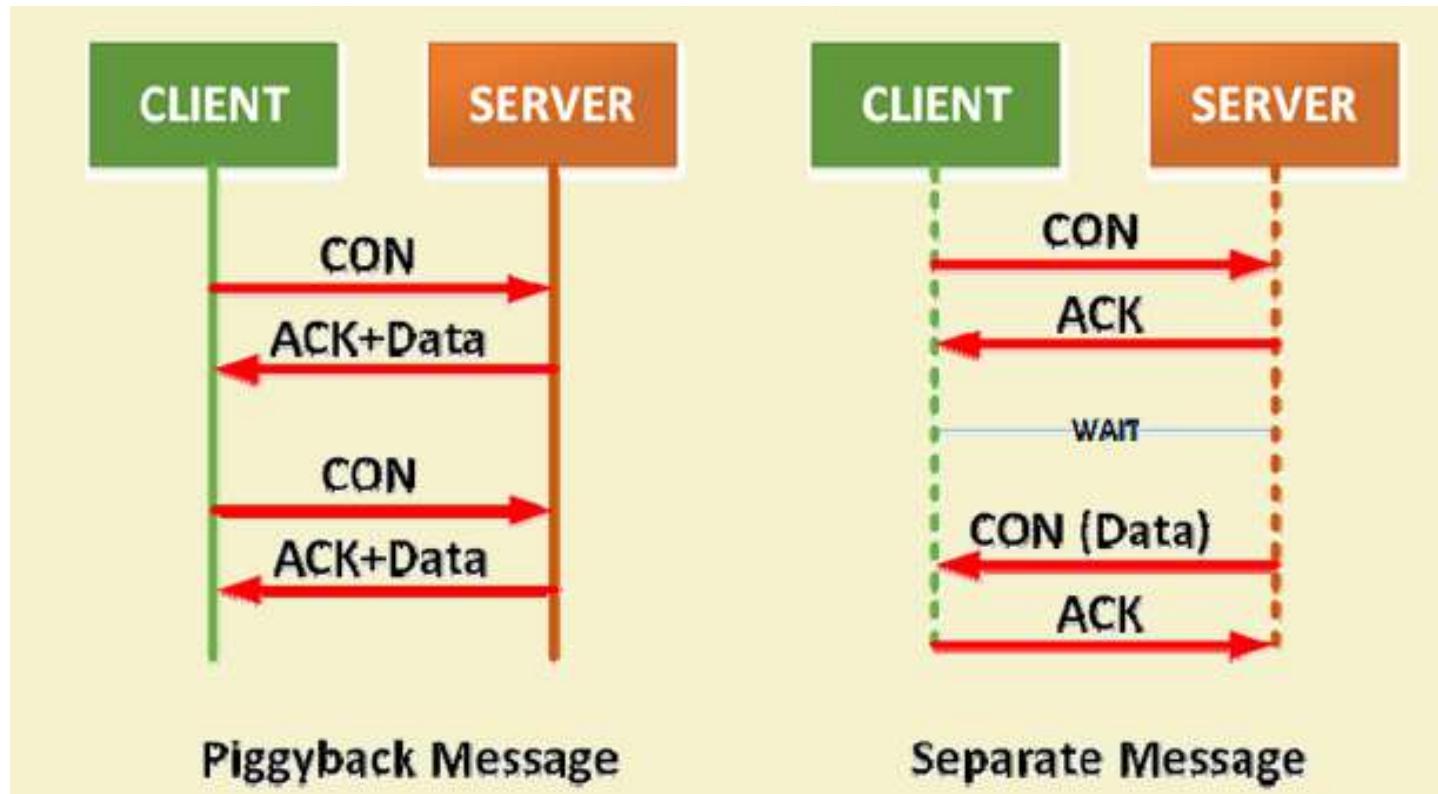
CoAP Request-Response Model



CoAP Request-Response Model

- Confirmable and non-confirmable modes represent the reliable and unreliable transmissions, respectively, while the other modes are used for request/response.
- Piggyback is used for client/server direct communication where the server sends its response directly after receiving the message, i.e., within the acknowledgment message.
- On the other hand, the separate mode is used when the server response comes in a message separate from the acknowledgment, and may take some time to be sent by the server.
- Similar to HTTP, CoAP utilizes GET, PUT, PUSH, DELETE messages requests to retrieve, create, update, and delete, respectively

CoAP Request-Response Model



CoAP Request-Response Model

Features

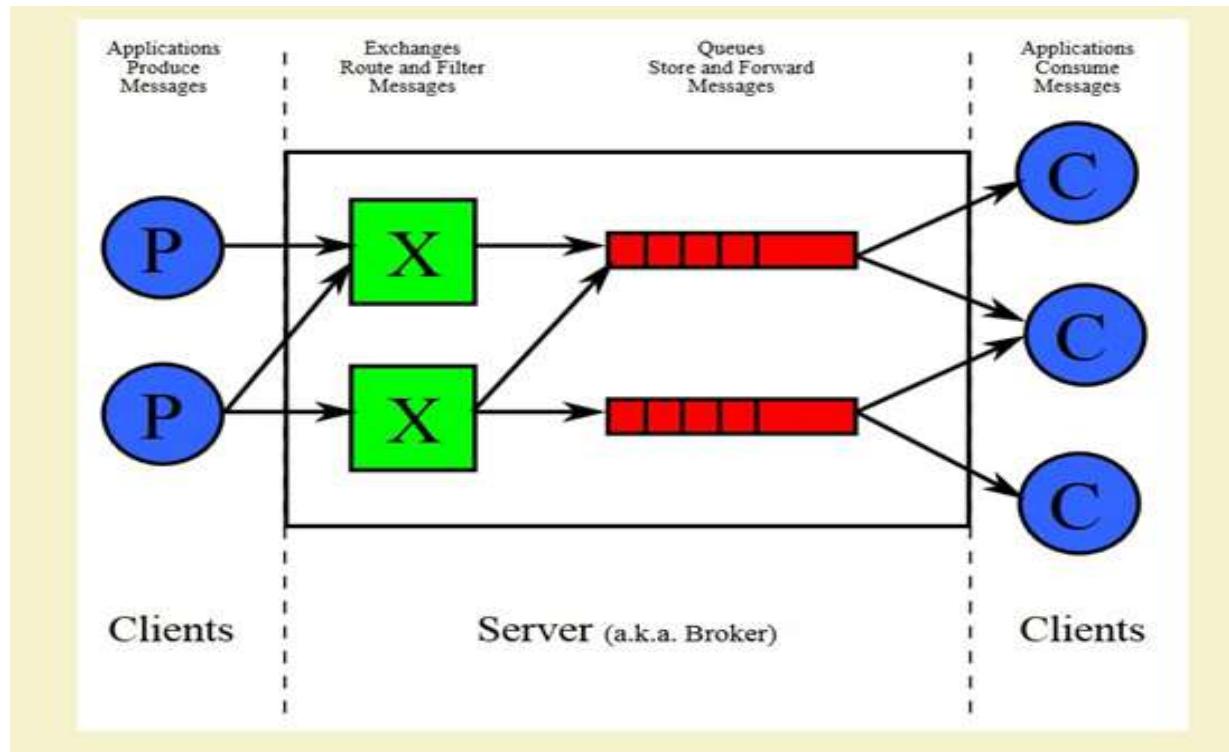
- Reduced overheads and parsing complexity.
- URL and content-type support.
- Support for the discovery of resources provided by known CoAP services.
- Simple subscription for a resource, and resulting push notifications.
- Simple caching based on maximum message age

AMQP

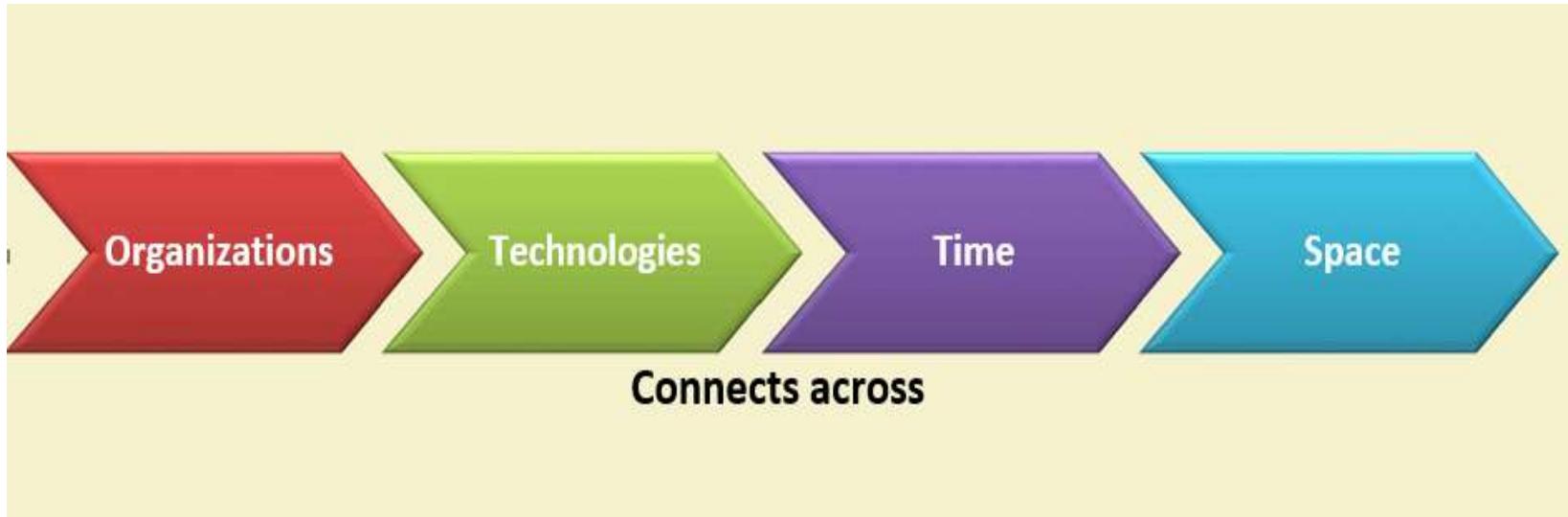
Introduction

- Advanced Message Queuing Protocol.
- Open standard for passing business messages between applications or organizations.
- Connects between systems and business processes.
- It is a binary application layer protocol.
- Basic unit of data is a *frame*.
- ISO standard: ISO/IEC 19464

AMQP



AMQP Features



AMQP Features



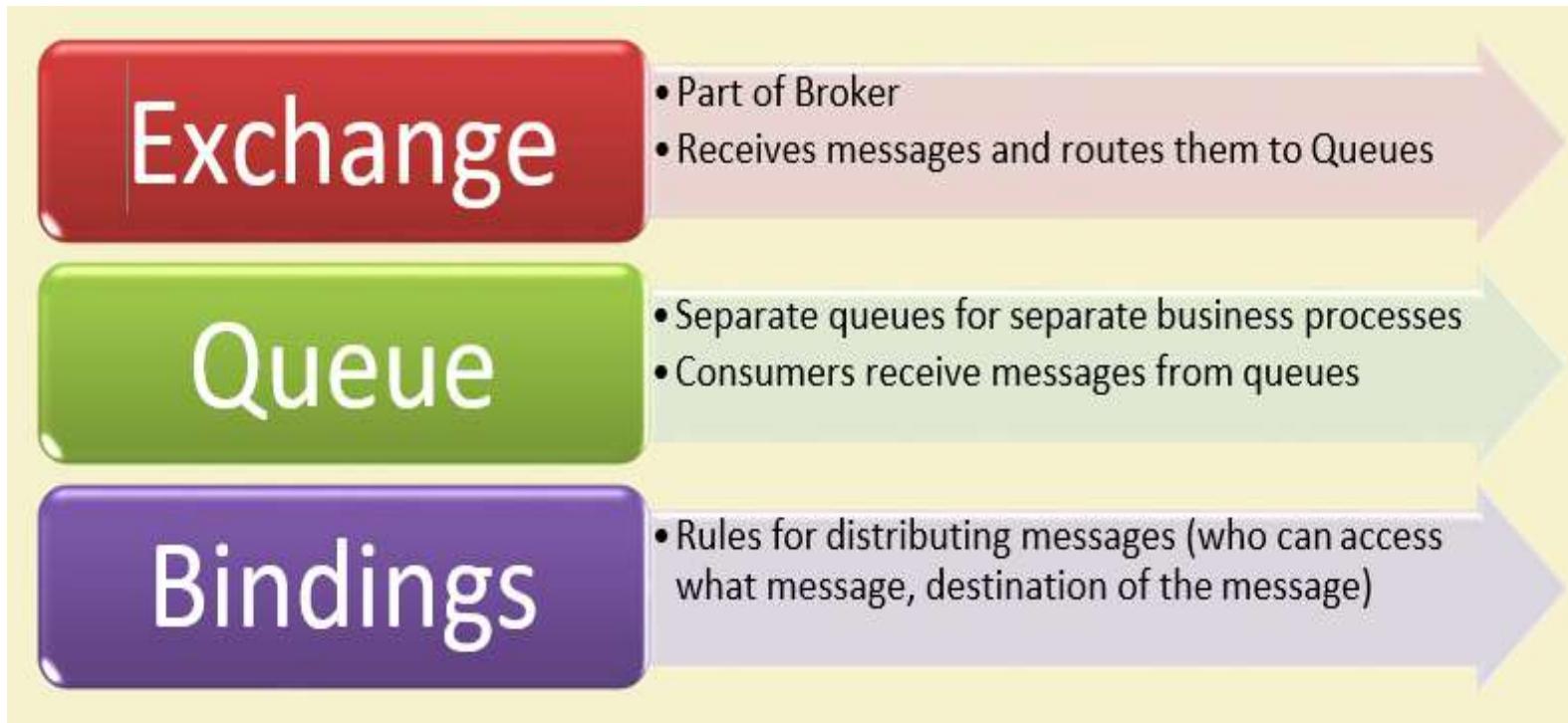
Message Delivery Guarantees

- *At-most-once*
 - each message is delivered once or never
- *At-least-once*
 - each message is certain to be delivered, but may do so multiple times
- *Exactly-once*
 - message will always certainly arrive and do so only once

AMQP Frame Types

- Nine AMQP frame types are defined that are used to initiate, control and tear down the transfer of messages between two peers:
 - Open (connection open)
 - Begin (session open)
 - Attach (initiate new link)
 - Transfer (for sending actual messages)
 - Flow (controls message flow rate)
 - Disposition (Informs the changes in state of transfer)
 - Detach (terminate the link)
 - End (session close)
 - Close (connection close)

AMQP Components



Module 3

RFID

INTERNET OF THINGS

A Hands-On Approach



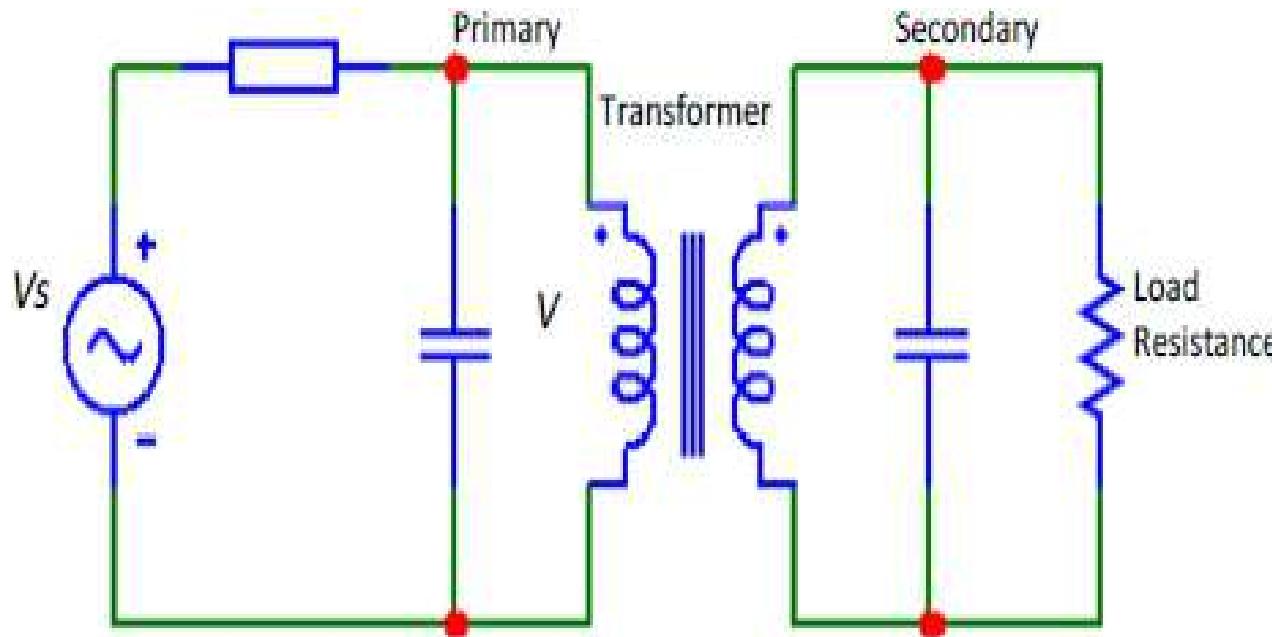
Arshdeep Bahga • Vijay Madisetti

RFID

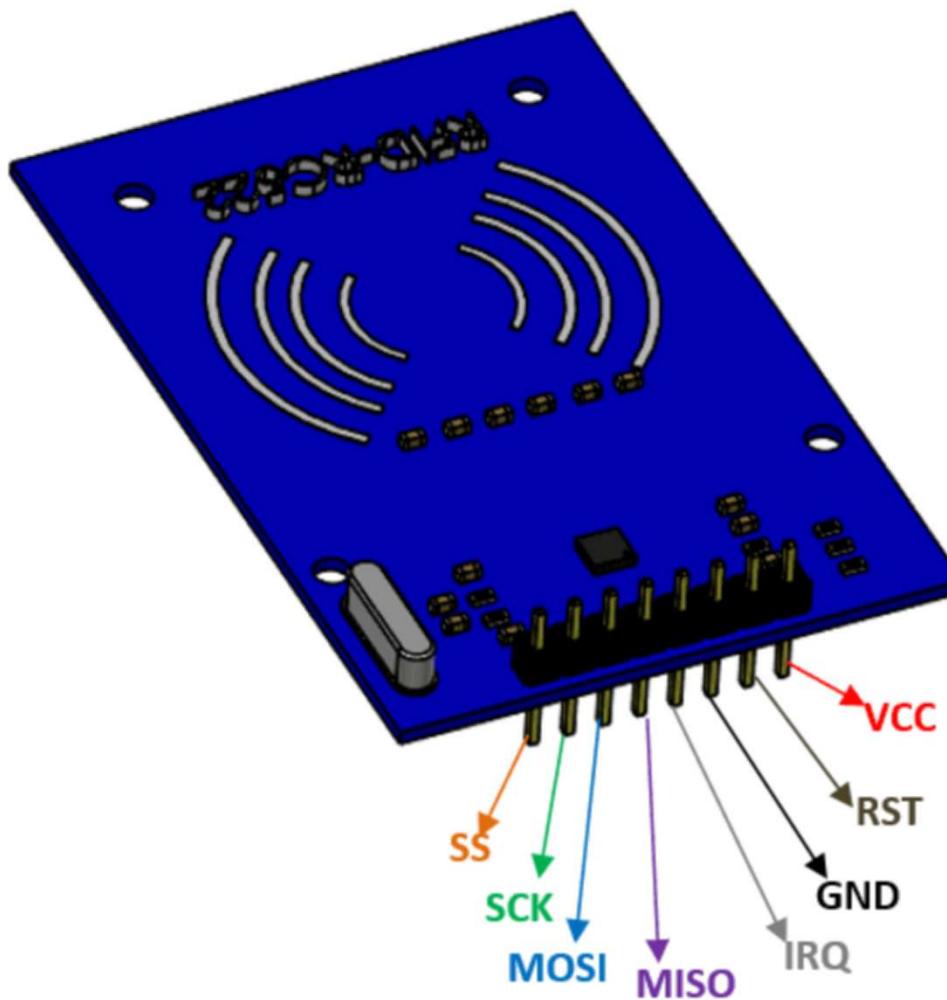
- The RFID systems operate in various frequency bands. Some of the most popular frequencies are:
 - – 125 kHz to 134.2 kHz (LF: low frequency);
 - – 13.56 MHz (HF: high frequency);
 - – 860 to 915 MHz (UHF: ultra-high frequency); and
 - – 2.45 GHz to 5.8 GHz (microwave frequency).

Principle of RFID

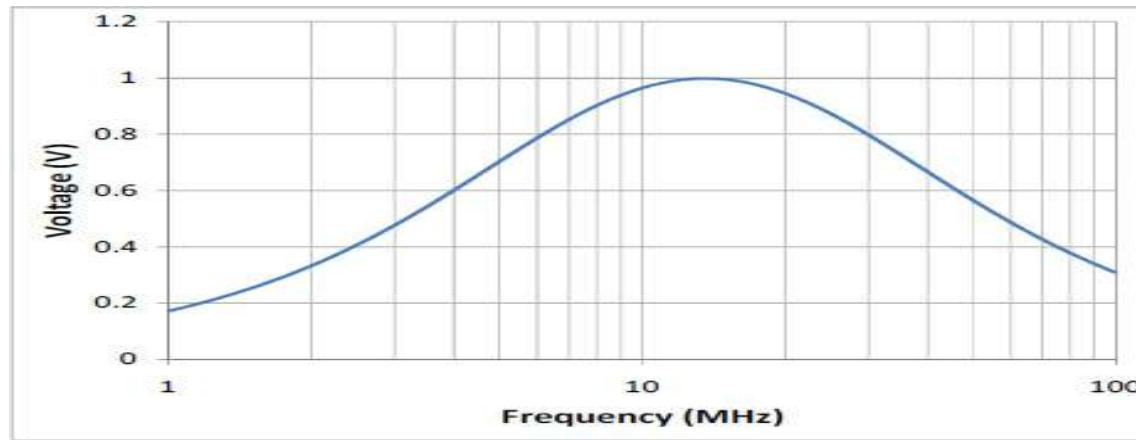
Circuit schematic of two coils electromagnetically interacting with each other



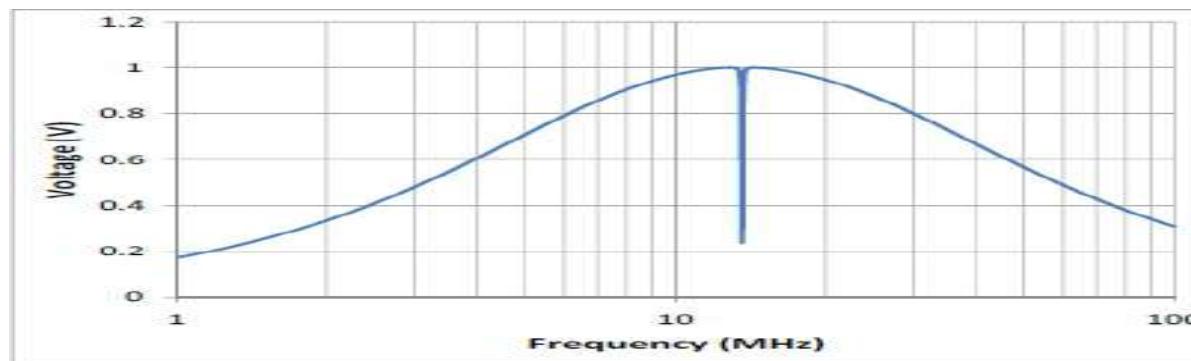




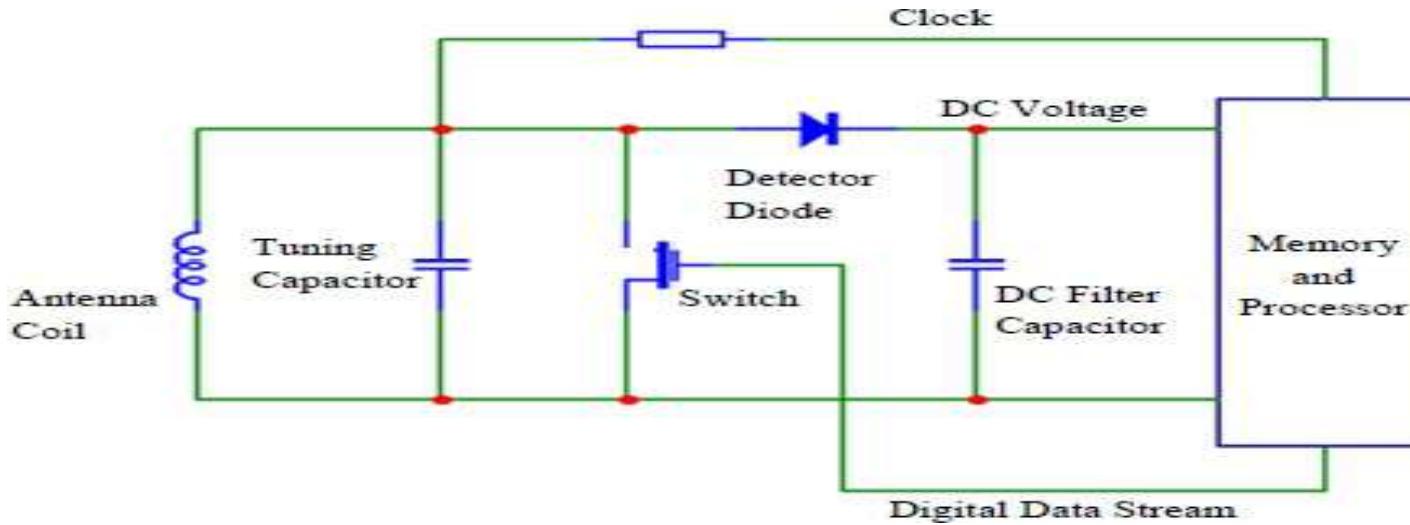
Voltage across the primary coil as a function of frequency



Voltage across the primary coil as a function of frequency for a different value of load resistance

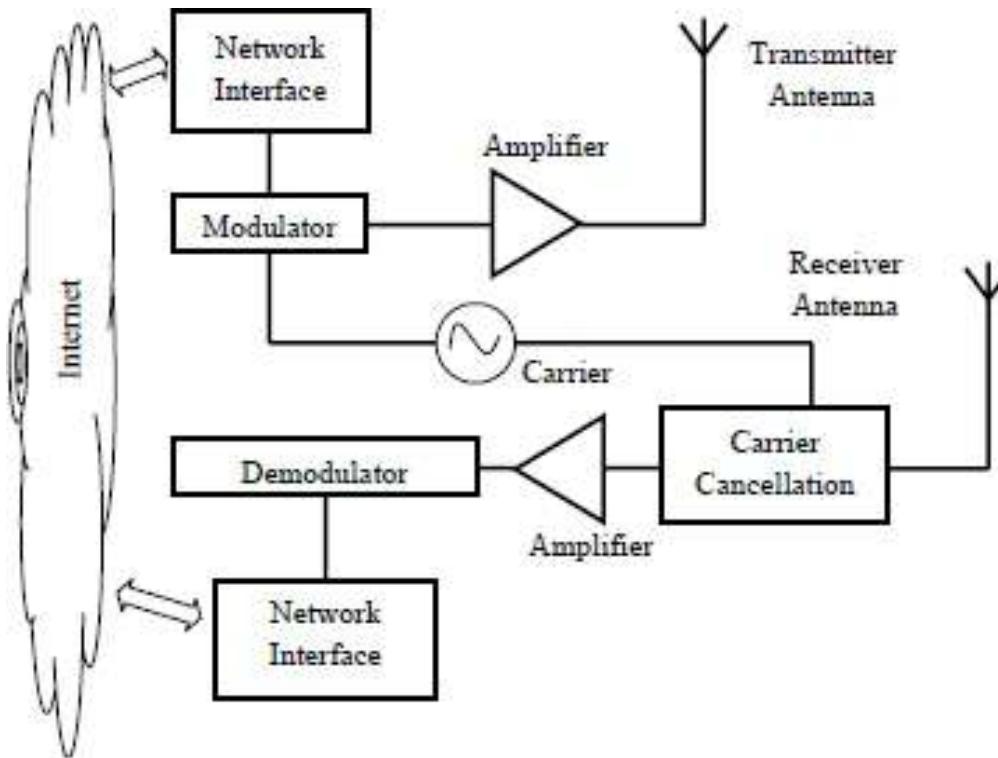


Schematic of an RFID tag

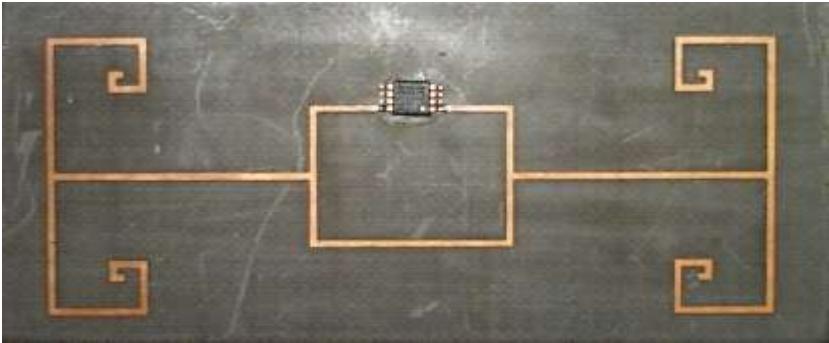


Components of RFID

- Reader



Components of RFID



- *RFID tag*

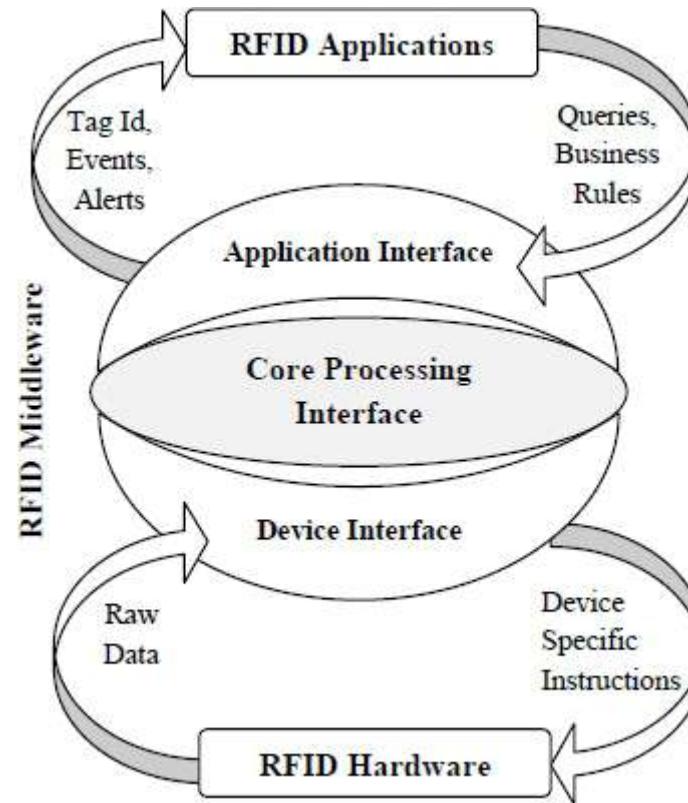
Photograph of an UHF RFID tag showing the antenna and the integrated circuit

RFID Middle ware

General middleware architecture is comprised of three components:

- – device interface;
- – core processing interface;
- – application interface.

RFID Middle ware



Input 1 = HIGH(5v)	Output 1 = HIGH	Motor 1 rotates in Clock wise Direction
Input 2 = LOW(0v)	Output 2 = LOW	
Input 3 = HIGH(5v)	Output 1 = HIGH	Motor 2 rotates in Clock wise Direction
Input 4 = LOW(0v)	Output 2 = LOW	

Input 1 = LOW(0v)	Output 1 = LOW	Motor 1 rotates in Anti-Clock wise Direction
Input 2 = HIGH(5v)	Output 2 = HIGH	
Input 3 = LOW(0v)	Output 1 = LOW	Motor 2 rotates in Anti -Clock wise Direction
Input 4 = HIGH(5v)	Output 2 = HIGH	

Input 1 = HIGH(5v)	Output 1 = HIGH	Motor 1 stays still
Input 2 = HIGH(5v)	Output 2 = HIGH	
Input 3 = HIGH(5v)	Output 1 = LOW	Motor 2 stays still
Input 4 = HIGH(5v)	Output 2 = HIGH	

S. No.	Lab Experiment on Arduino Uno using Autodesk TinkerCad Simulation.
1	<p>a) Write the code to blink an LED on Arduino Uno. Compile and verify the result on the serial monitor of Arduino IDE.</p> <p>Additional Programs:</p> <ul style="list-style-type: none"> i) To blink two LED's alternatively ii) To blink odd and even leds iii) To scroll an LED's
2	<p>Interfacing of Arduino Uno with LED and switch. Write a program to control LED using Switch.</p> <p>Additional Programs:</p> <ul style="list-style-type: none"> i) Single switch to control multiple LED's ii) Multi switches to control multiple LED's
3	<p>Interfacing of Arduino Uno with potentiometer and LED. Write a program to vary the intensity of LED using a potentiometer.</p> <p>Additional Programs:</p> <ul style="list-style-type: none"> i) Adjust the brightness of LED without potentiometer.
4	<p>(a) Interfacing of Ultrasonic sensor with Arduino Uno. Write a program to measure the distance from obstacle and display on the serial monitor.</p> <p>(b) Interface an IR sensor with Arduino Uno. Write a program to detect obstacle and display on the serial monitor.</p>
5	<p>(a) Interfacing of Temperature sensor with Arduino Uno. Write a program to read the specific temperature of a room and display on the serial monitor.</p> <p>(b) Interfacing of LDR with Arduino Uno. Write a program to control the intensity of LED using LDR.</p>
6	<p>(a) Interfacing of DC motor with Arduino Uno. Write a program to rotate the motor in clockwise and anticlockwise direction with using a delay of 2 sec.</p> <p>(b) Familiarize the concept of pulse width modulation. Write a program to control the speed of DC motor using PWM.</p>

7

- (a) **Interfacing of a display device, i.e., LCD x2 with Arduino Uno. Write a program to display “HELLO IOT” on LCD.**
- (b) **Write a program to scroll the message “Welcome to IoT Lab.”**
- (c) **Write a program to blink the message “Hello IoT.”**

EXPERIMENT 1

Aim of the Experiment: To Blink LED

Components Used: Arduino UNO, LED, Resistor, Tinkercad Simulator.

Circuit Diagram:

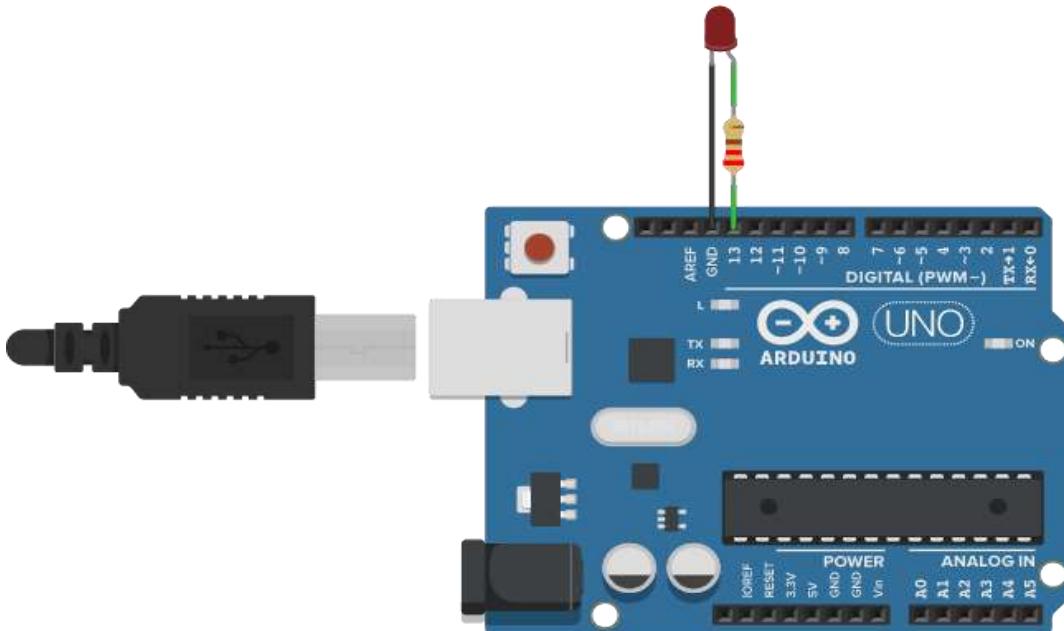


Figure : Blinking of LED

Software program:

```
void setup()
{
    pinMode(13, OUTPUT);
    Serial.begin(9600);
}
void loop()
{
    digitalWrite(13, HIGH);
    Serial.println("LED : ON");
    delay(1000); // Wait for 1000 millisecond(s)
    digitalWrite(13, LOW);
    Serial.println("LED : OFF");
    delay(1000); // Wait for 1000 millisecond(s)
}
```

Screenshot of Serial monitor:



The screenshot shows a "Serial Monitor" window with a light gray background. At the top left is a small icon of a computer monitor with a bar chart. To its right is the title "Serial Monitor". On the far right edge of the window frame, there is a small downward-pointing arrow. The main area of the window contains eight lines of text, each consisting of the word "LED" followed by a colon and either "ON" or "OFF", indicating the state of an LED. The text is arranged vertically from top to bottom.

```
LED : ON
LED : OFF
```

EXPERIMENT 2

Aim of the Experiment: Controlling LED with Switch.

Components Used: Arduino UNO, LED, Resistor, Tinkercad Simulator, Switch.

Circuit Diagram:

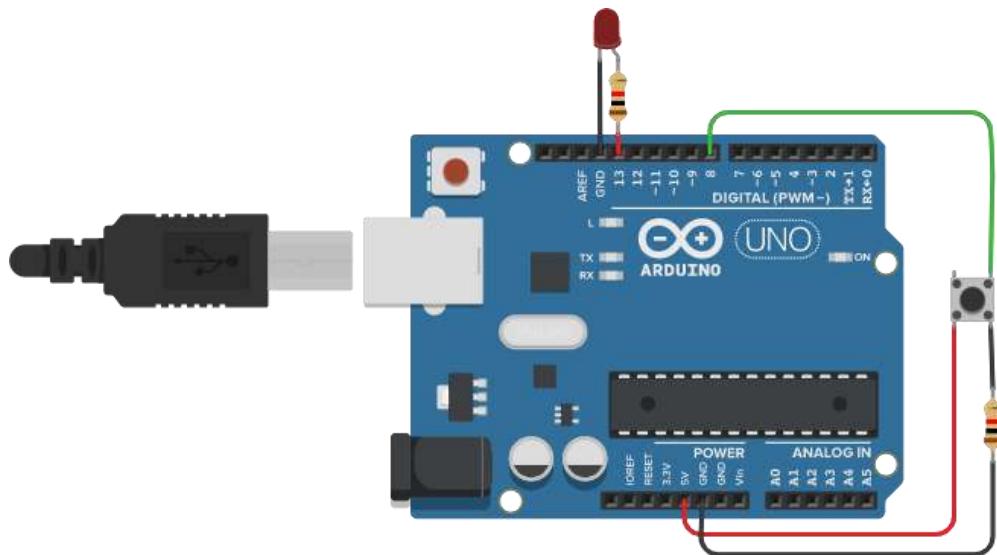


Figure : Control of LED using Switch

Software program:

```
void setup()
{
    pinMode(13, OUTPUT);
    Serial.begin(9600);
}

void loop()
{
    digitalWrite(13, HIGH);
    Serial.println("LED : ON");
    delay(1000); // Wait for 1000 millisecond(s)
    digitalWrite(13, LOW);
    Serial.println("LED : OFF");
    delay(1000); // Wait for 1000 millisecond(s)
}
```

EXPERIMENT 3

Aim of the Experiment : Intensity of Led using Potentiometer .

Components Used : Arduino UNO,Led,Potentiometer,Resistor,Tinckercad simulator .

Circuit Diagram:

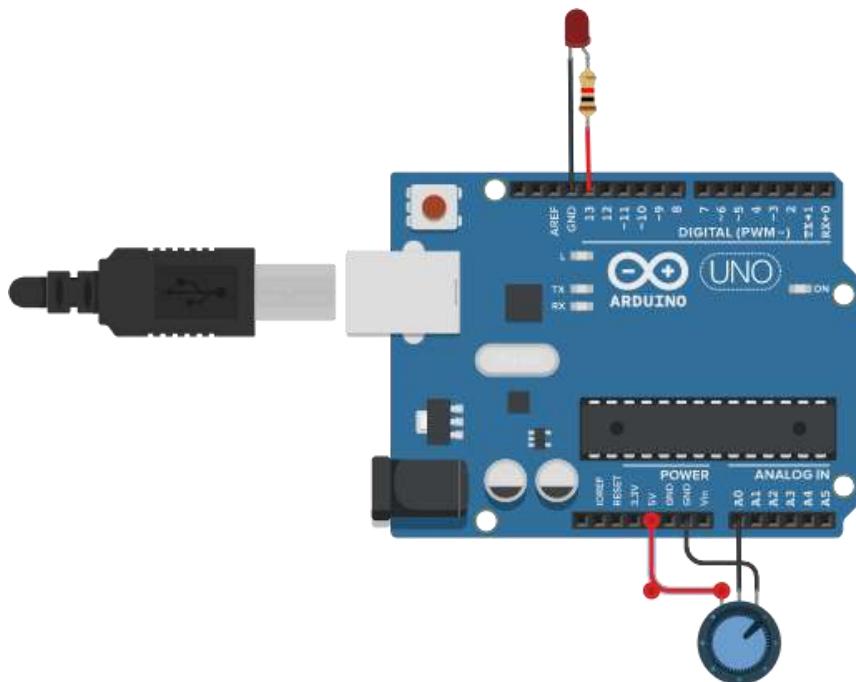


Figure : Intensity of Led using Potentiometer

Software program:

```
int value=0;
void setup()
{
  pinMode(A0, INPUT);
  pinMode(11, OUTPUT);
}
void loop()
{
  value = analogRead(A0);
  digitalWrite(11,HIGH);
  delay(value);

  digitalWrite(11, LOW);
  delay(value);
}
```

EXPERIMENT 4(a)

Aim of the Experiment: Measure the distance from obstacle and display on the serial monitor using ultrasonic sensor

Components Used: Arduino UNO, Ultrasonic sensor, Tinkercad Simulator.

Circuit Diagram:

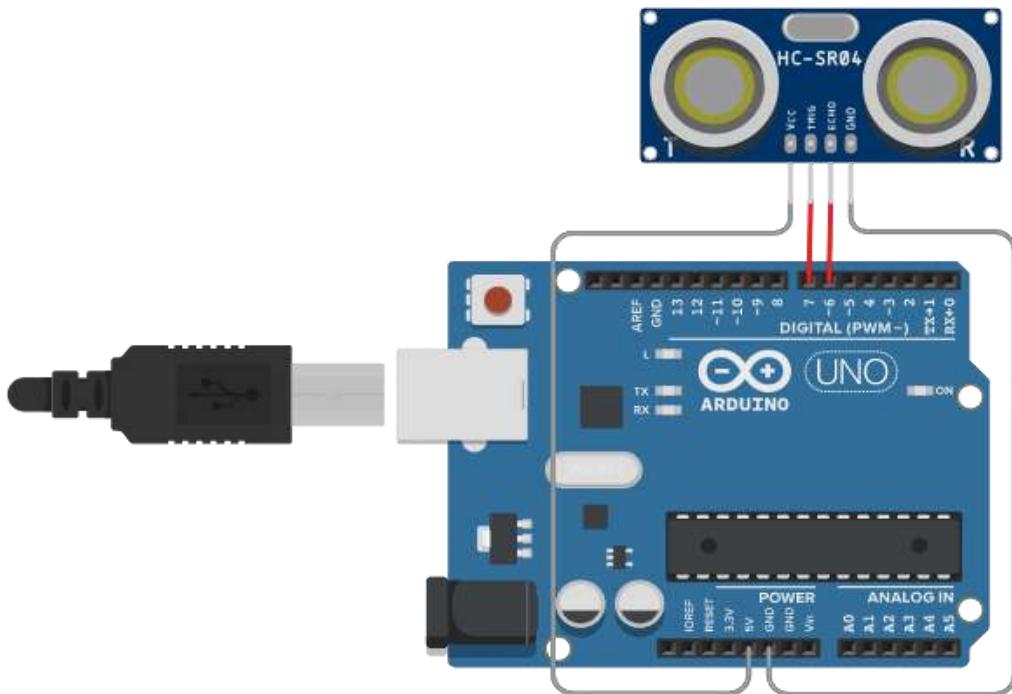


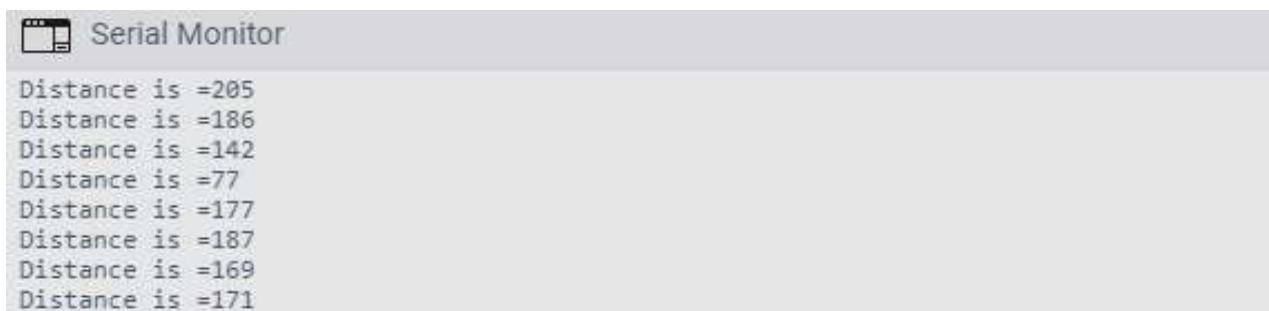
Figure : Ultrasonic Sensor

Software program:

```
const int trigpin =6;
const int echopin =7;
int distance;
int duration;
void setup()
{
pinMode(trigpin, OUTPUT);
pinMode(echopin, INPUT);
Serial.begin(9600);
}
```

```
void loop()
{
digitalWrite(trigpin, LOW);
delayMicroseconds(2);
digitalWrite(trigpin, HIGH);
delayMicroseconds(10);
digitalWrite(trigpin, LOW);
duration = pulseIn(echopin, HIGH);
distance = (0.034*duration)/2;
Serial.print("Distance is =");
Serial.println(distance);
}
```

Screenshot of Serial monitor:



A screenshot of the Arduino Serial Monitor window. The title bar says "Serial Monitor". The main area displays a series of text entries, each consisting of the string "Distance is =" followed by a numerical value representing distance in centimeters. The values listed are 205, 186, 142, 77, 177, 187, 169, and 171.

```
Distance is =205
Distance is =186
Distance is =142
Distance is =77
Distance is =177
Distance is =187
Distance is =169
Distance is =171
```

EXPERIMENT 4(b)

Aim of the Experiment: To Detect Obstacle.

Components Used: Arduino UNO, IR Sensor, IR Remote, Tinkercad Simulator.

Circuit Diagram :

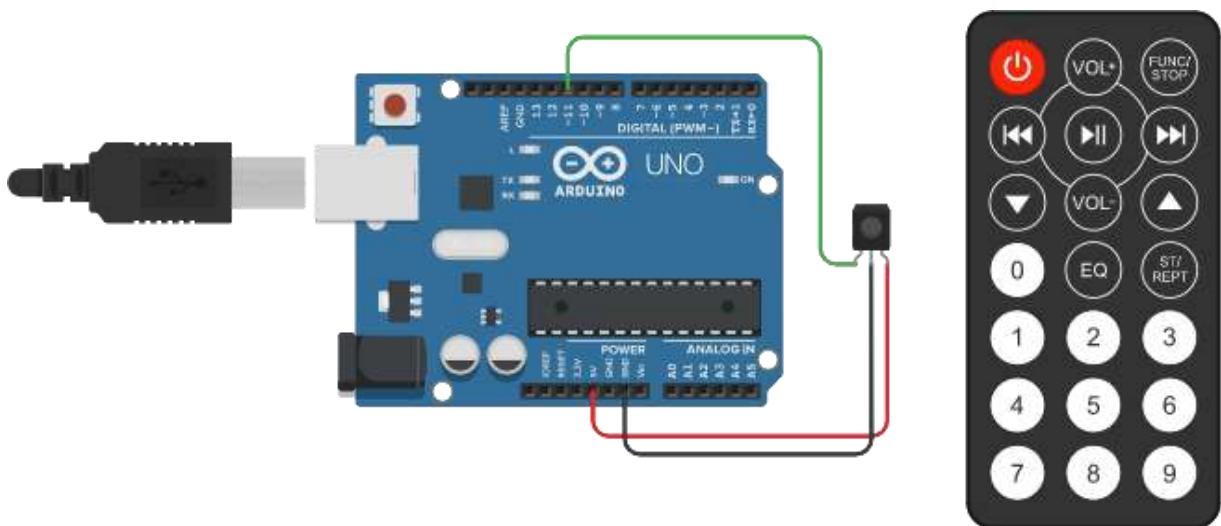
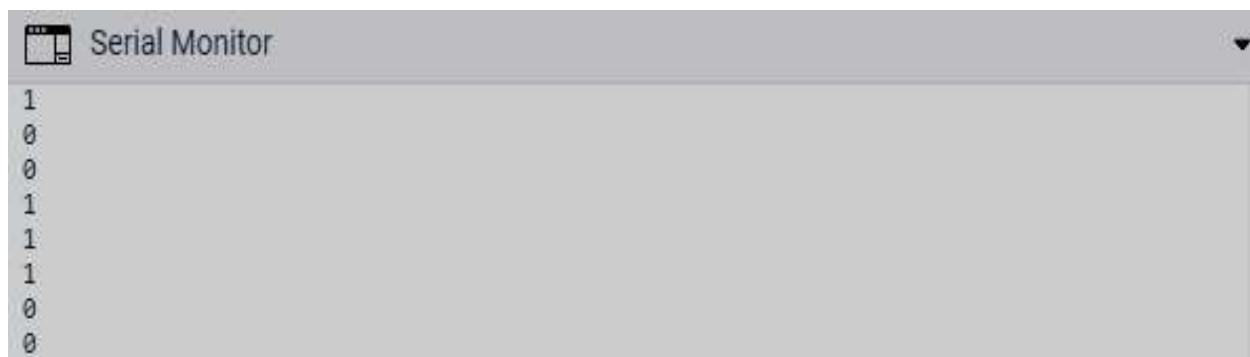


Figure : Setup of IR Sensor

Software program:

```
int ir;  
void setup()  
{  
    pinMode(11,INPUT);  
    Serial.begin(9600);  
}  
void loop()  
{  
    ir=digitalRead(11);  
    Serial.println(ir);  
}
```

Screenshot of Serial monitor:



A screenshot of a "Serial Monitor" window. The title bar reads "Serial Monitor". The main area contains the following binary data:
1
0
0
1
1
1
0
0

EXPERIMENT 5(a)

Aim of the Experiment: To read the specific temperature of a room.

Components Used: Arduino UNO, Temperature Sensor, Resistor, LCD, Potentiometre, Tinkercad Simulator.

Circuit Diagram:

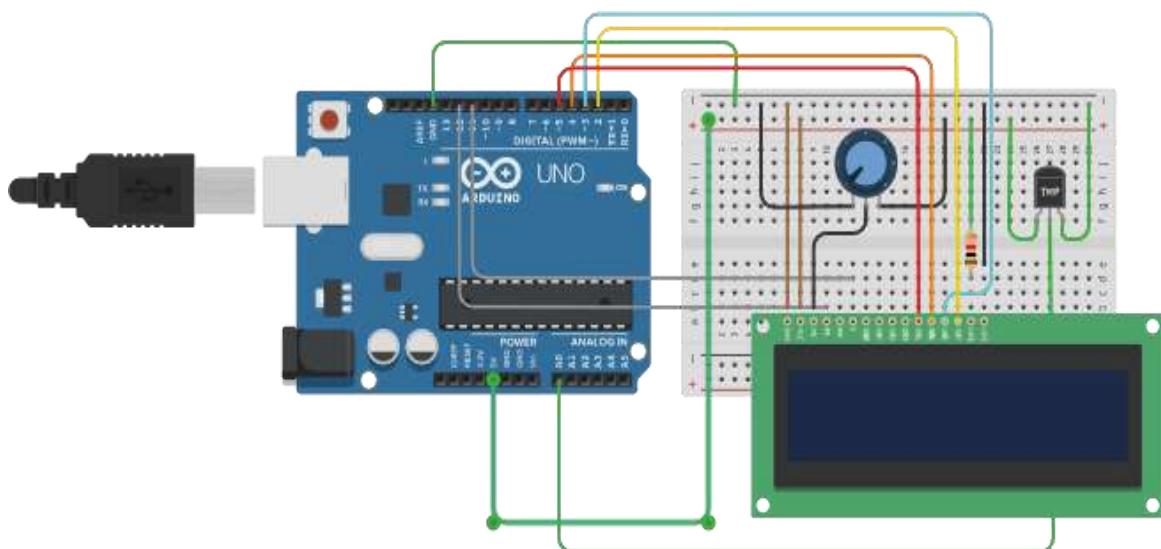


Figure : Temprature Sensor

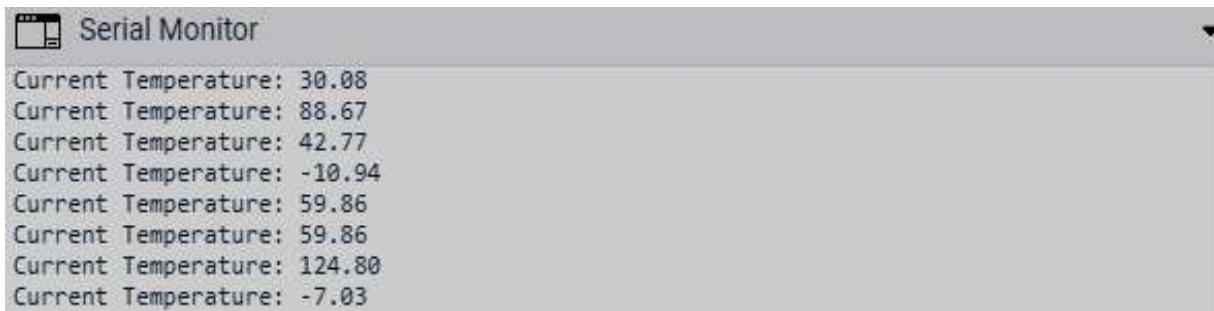
Software program:

```
#include <LiquidCrystal.h>
```

```
int sensePin = A0;  
int sensorInput;  
double temp;  
LiquidCrystal lcd(12,11,5,4,3,2);
```

```
void setup()
{
    Serial.begin(9600);
    lcd.begin(16,2);
}
void loop()
{
    sensorInput = analogRead(A0);
    temp = (double)sensorInput / 1024;
    temp = temp * 5;
    temp = temp - 0.5;
    temp = temp * 100;
    lcd.setCursor(0,0);
    lcd.print("Temp:");
    lcd.setCursor(6,0);
    lcd.print(temp);
    Serial.print("Current Temperature: ");
    Serial.println(temp);
}
```

Screenshot of Serial monitor:



The screenshot shows the Arduino Serial Monitor window titled "Serial Monitor". The window displays a series of temperature readings printed by the sketch. The text in the window is as follows:

```
Current Temperature: 30.08
Current Temperature: 88.67
Current Temperature: 42.77
Current Temperature: -10.94
Current Temperature: 59.86
Current Temperature: 59.86
Current Temperature: 124.80
Current Temperature: -7.03
```

EXPERIMENT 5(b)

Aim of the Experiment: Controlling the intensity of LED using LDR.

Components Used: Arduino UNO, LED, Photoresistor, Resistor,Tinkercad Simulation.

Circuit Diagram:

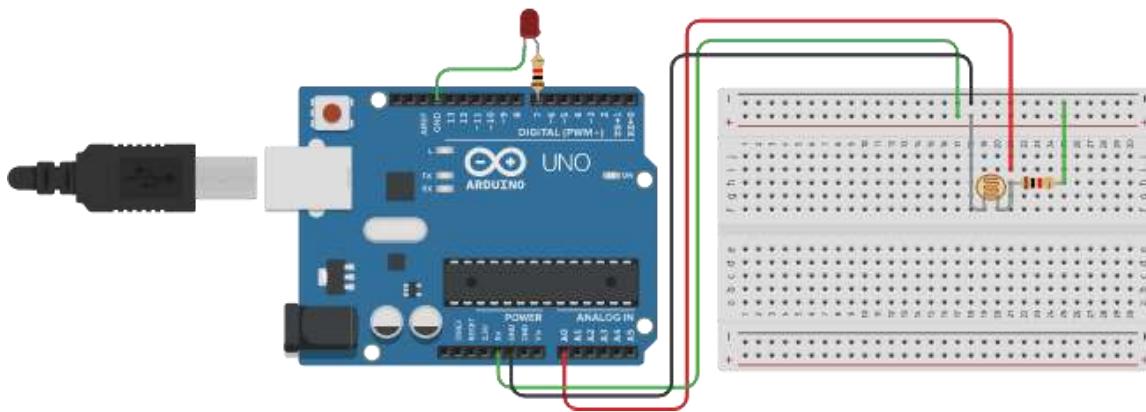


Figure Controlling intensity of LED using LDR

Software program:

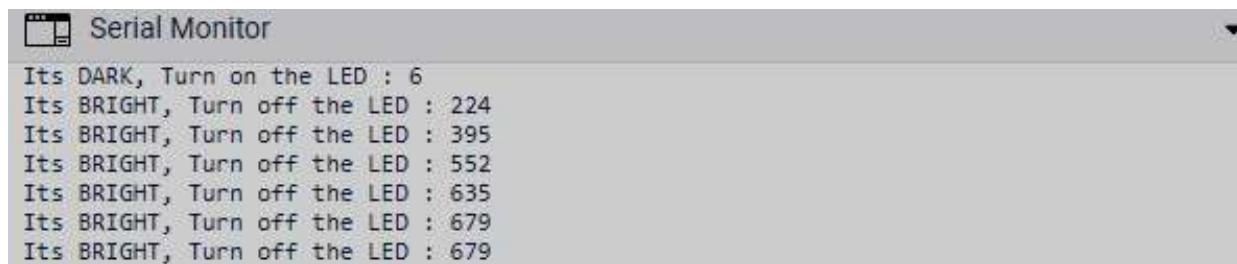
```
const int ledPin = 7;  
const int ldrPin = A0;  
void setup()  
{  
  
Serial.begin(9600);  
pinMode(ledPin, OUTPUT);  
pinMode(ldrPin, INPUT);  
  
}  
  
void loop()  
{
```

```
int ldrStatus = analogRead(ldrPin);
if (ldrStatus <= 200)
{
    digitalWrite(ledPin, HIGH);
    Serial.print("Its DARK, Turn on the LED : ");
    Serial.println(ldrStatus);

}
else
{
    digitalWrite(ledPin, LOW);
    Serial.print("Its BRIGHT, Turn off the LED : ");
    Serial.println(ldrStatus);

}
```

Screenshot of Serial monitor:



The screenshot shows the Arduino Serial Monitor window titled "Serial Monitor". The window displays a series of messages printed by the sketch. The messages alternate between "Its DARK, Turn on the LED : <value>" and "Its BRIGHT, Turn off the LED : <value>". The values listed are 6, 224, 395, 552, 635, 679, and 679. The background of the monitor is light gray, and the text is black.

```
Its DARK, Turn on the LED : 6
Its BRIGHT, Turn off the LED : 224
Its BRIGHT, Turn off the LED : 395
Its BRIGHT, Turn off the LED : 552
Its BRIGHT, Turn off the LED : 635
Its BRIGHT, Turn off the LED : 679
Its BRIGHT, Turn off the LED : 679
```

EXPERIMENT 6(a)

Aim of the Experiment: Rotating a DC Motor in clockwise & anticlockwise direction using delay.

Components Used: Arduino UNO, DC Motor, H- Bridge Motor Driver, Tinkercad Stimulation.

Circuit Diagram:

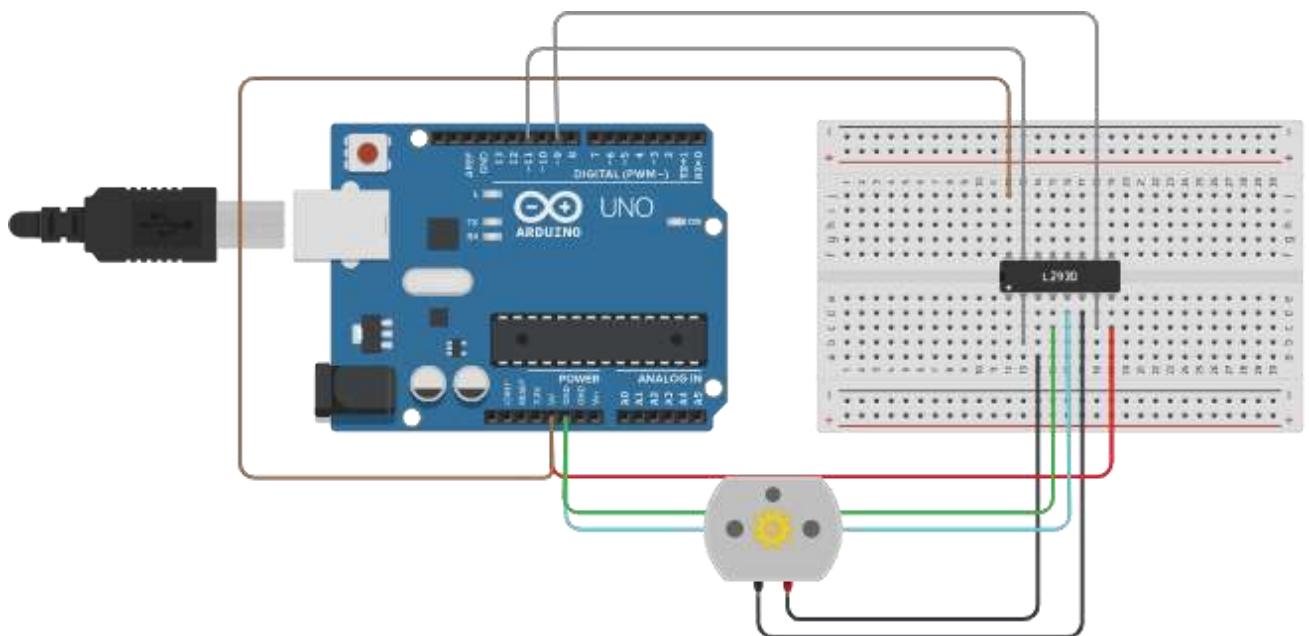


Figure Rotating DC motor

Software program:

```
int In1=12;  
int In2=9;  
void setup()  
{  
  
pinMode(In1,OUTPUT);  
pinMode(In2,OUTPUT);  
Serial.begin(9600);  
  
}
```

```
void loop()
{
    digitalWrite(In1,HIGH);
    digitalWrite(In2,LOW);
    Serial.println("Clockwise");
    delay(2000);
    digitalWrite(In1,LOW);
    digitalWrite(In2,HIGH);
    Serial.println("Anti Clockwise");
    delay(2000);
    digitalWrite(In1,LOW);
    digitalWrite(In2,LOW);
    Serial.println("Stop");
    delay(2000);
}
```

Screenshot of Serial Monitor:



The screenshot shows the Arduino Serial Monitor window. The title bar reads "Serial Monitor". The main area displays the following text, which corresponds to the output of the provided code:

```
Stop
Clockwise
Anti Clockwise
Stop
Clockwise
Anti Clockwise
Stop
Clockwise
```

EXPERIMENT 6(b)

Aim of the Experiment: Control the speed of DC motor using PWM.

Components Used: Arduino UNO, DC Motor, Tinkercad Simulator.

Circuit Diagram:

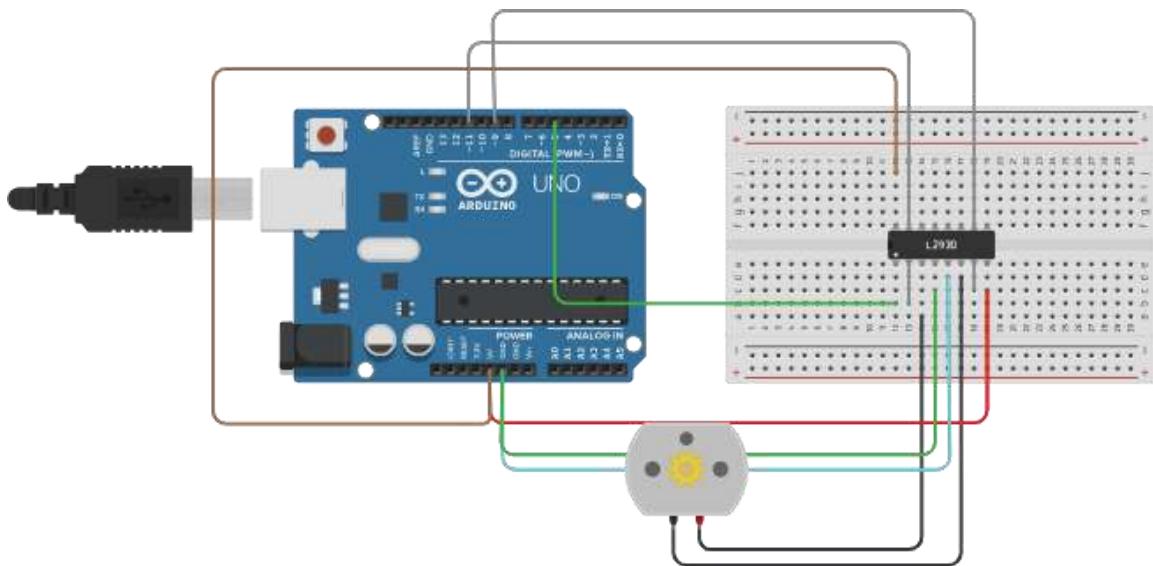


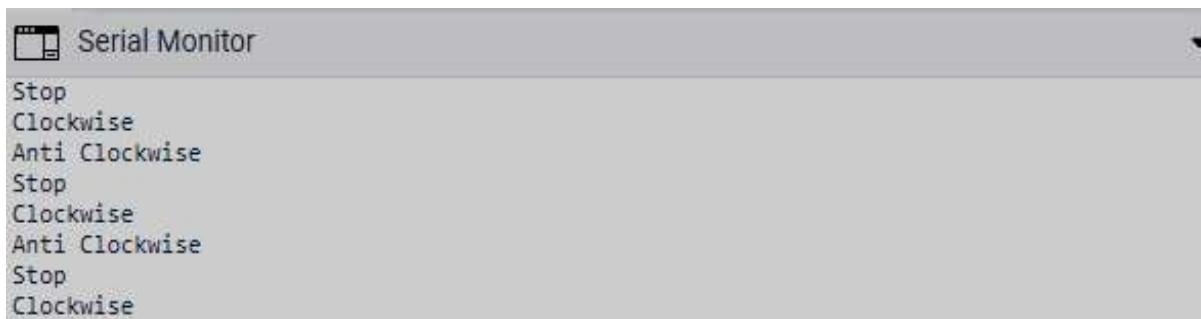
Figure : Rotating DC Motor

Software program:

```
int en=5;
int In1=12;
int In2=9;
void setup()
{
    pinMode(en,OUTPUT);
    pinMode(In1,OUTPUT);
    pinMode(In2,OUTPUT);
    Serial.begin(9600);
}
```

```
void loop()
{
analogWrite(en,255);
digitalWrite(In1,HIGH);
digitalWrite(In2,LOW);
Serial.println("Clockwise");
delay(2000);
analogWrite(en,255);
digitalWrite(In1,LOW);
digitalWrite(In2,HIGH);
Serial.println("Anti Clockwise");
delay(2000);
analogWrite(en,255);
digitalWrite(In1,LOW);
digitalWrite(In2,LOW);
Serial.println("Stop");
delay(2000);
}
```

Screenshot of Serial monitor:



The screenshot shows the Arduino Serial Monitor window. The title bar reads "Serial Monitor". The main area displays the following text:
Stop
Clockwise
Anti Clockwise
Stop
Clockwise
Anti Clockwise
Stop
Clockwise

EXPERIMENT 7(a)

Aim of the Experiment: Display “HELLO IOT” on LCD.

Components Used: Arduino UNO, LCD, Potentiometer, Resistor,Tinkercad Simulator.

Circuit Diagram:

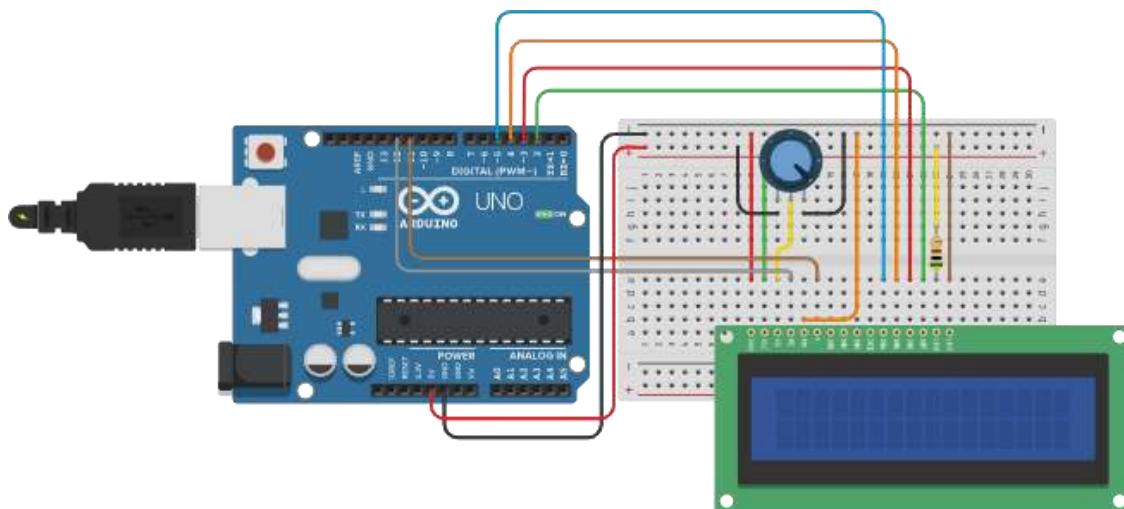


Figure : Setup for Print of Hello Iot

Software program:

cc

EXPERIMENT 7(b)

Aim of the Experiment: Scroll the message “Welcome to IoT Lab”.

Components Used: Arduino UNO, Tinkercad Simulation, LCD, Potentiometer, Resistor.

Circuit Diagram:

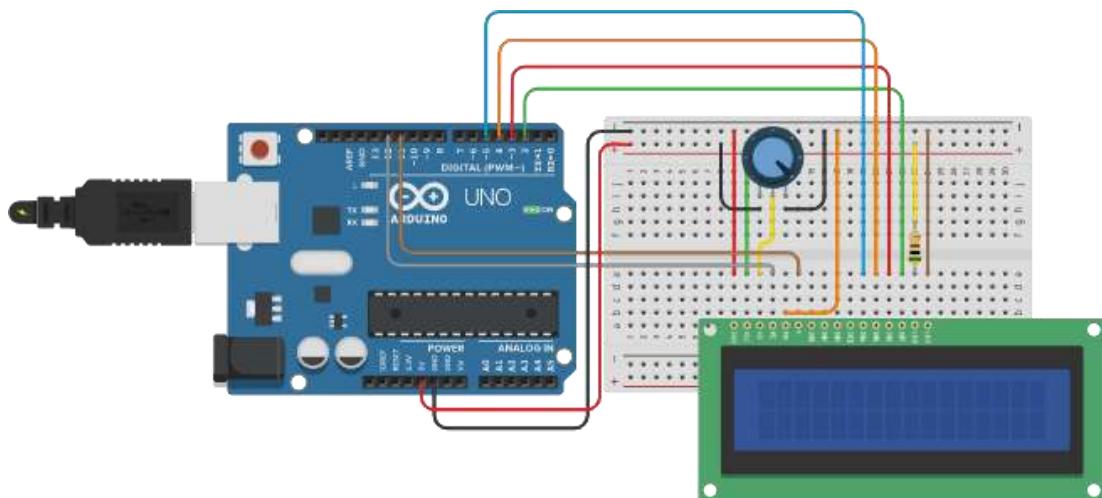


Figure Setup for Scrolling.

Software program:

```
#include <LiquidCrystal.h>
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
void setup()
{
    lcd.begin(16, 2);
    lcd.print("hello, world!");
    delay(1000);
}
void loop()
{
    for (int positionCounter = 0; positionCounter < 13; positionCounter++)
    {
        lcd.scrollDisplayLeft();
        delay(150);
    }
    for (int positionCounter = 0; positionCounter < 16; positionCounter++)
    {
        lcd.scrollDisplayLeft();
        delay(150);
    }
}
```

EXPERIMENT 7(c)

Aim of the Experiment: Blinking the message “Hello IoT”.

Components Used: Arduino UNO, Tinkercad Simulation, LCD, Resistor, Potentiometer.

Circuit Diagram:

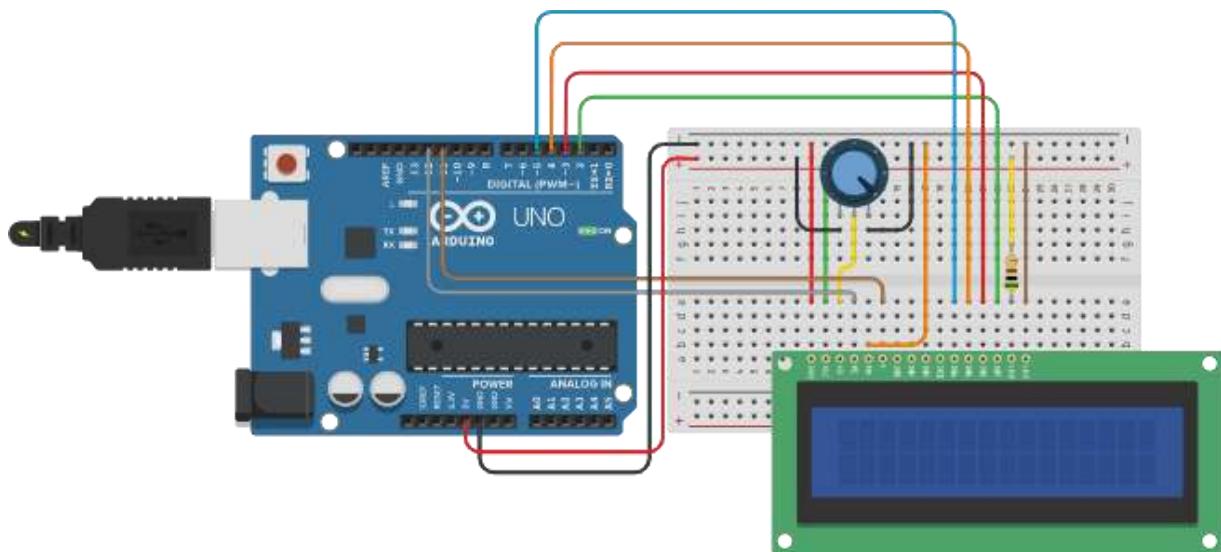


Figure Setup for Blinking

Software program:

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12,11,5,4,3,2);
void setup()
{ lcd.begin(16, 2); }
void loop()
{
  lcd.setCursor(4,0);
  lcd.print("Hello IoT");
  delay(1000);
  lcd.clear(); delay(500);
}
```



**PRESIDENCY
UNIVERSITY**

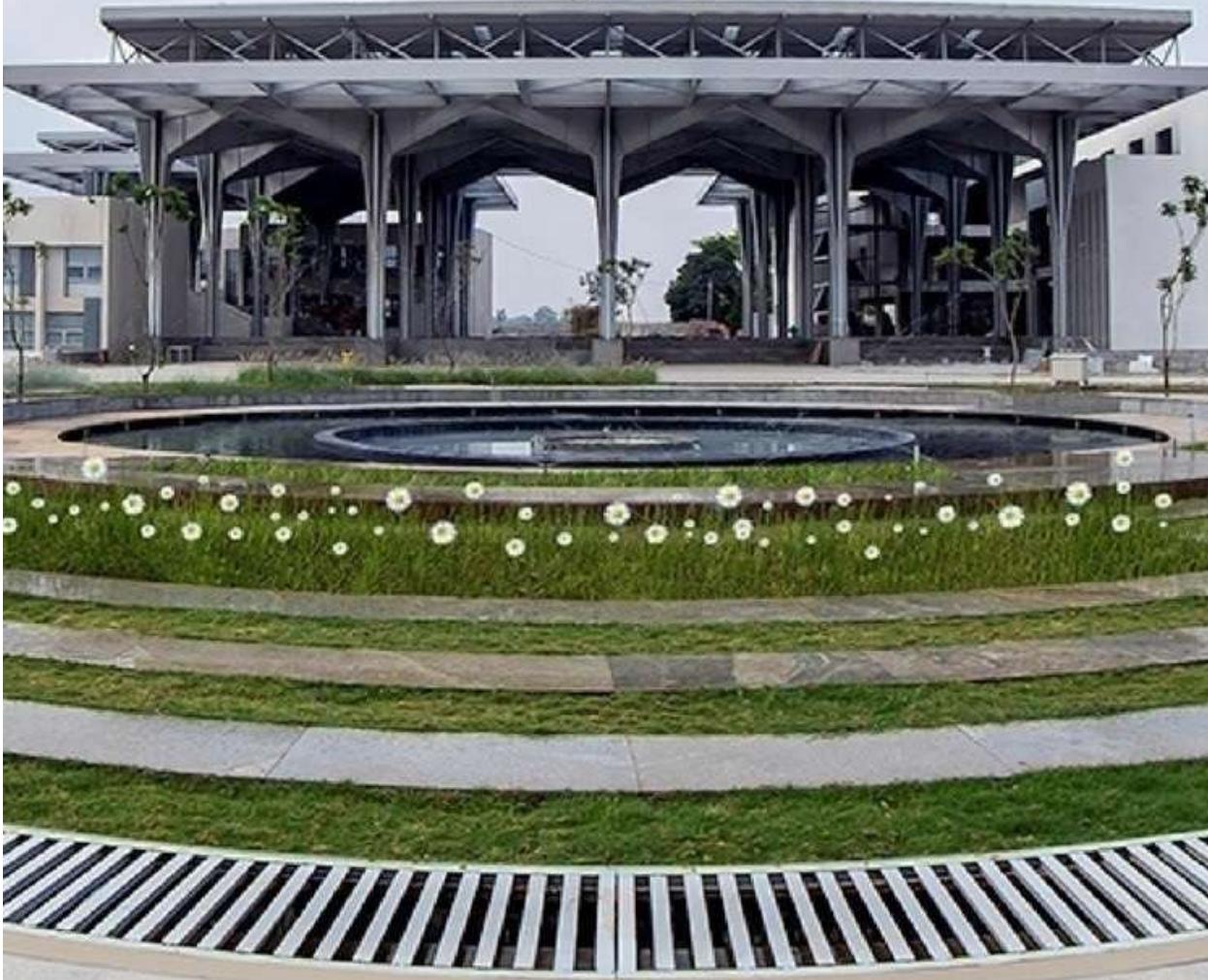
Private University Estd. in Karnataka State by Act No. 41 of 2013



Department of Computer Science and Engineering

Internet of Things -IoT CSE 202

Lab Manual 2018-19



www.presidencyuniversity.in

Laboratory Manual



Internet of Things (IoT)

For

Final year Students (CSE)

Itgalpur Rajanakunte, Yelahanka, Bengaluru, Karnataka 560064

FOREWORD

It is my great pleasure to present this laboratory manual for Final year engineering students for the subject Internet of Things (IoT).

As a student, many of you may be wondering with some of the questions in your mind regarding the subject and exactly what has been tried is to answer through this manual.

As you may be aware that Presidency University has already been awarded with Best Emerging University in south India by ASSOCHAM, INDIA and it's our endure to technically equip our students take the advantage of the procedural aspects of this certification.

Faculty members are also advised that covering these aspects in initial stage itself, will greatly relieve them in future as much of the load will be taken care by the enthusiasm energies of the students once they are conceptually clear.

Mohammed Mujeer
Assistant Professor,
Department of CSE,
Presidency University

LABORATORY MANUAL CONTENTS

This manual is intended for the final year students of Computer Science and Engineering in the subject of Internet of Things(IoT). This manual typically contains practical/lab sessions related Raspberry pi and Android implemented in Python programming covering various aspects related the subject to enhanced understanding.

Students are advised to thoroughly go through this manual rather than only topics mentioned in the syllabus as practical aspects are the key to understanding and conceptual visualization of theoretical aspects covered in the books.

Good Luck for your Enjoyable Laboratory Sessions

Dr. Mohan K.G

HOD,CSE

Mr. Mohammed Mujeer

PROGRAMME OBJECTIVES

B.Tech. Computer Science Engineering graduates of Presidency University, will be able to:

1. Direct and coordinate activities concerned with software/hardware design, development, and testing.
2. Design and develop products and systems for various applications.
3. Monitor and analyze computing system performance for network traffic, security, and capacity.
4. Order and maintain inventory of computing equipment for customer premises equipment, facilities, access networks and backbone network
5. Operate and trouble shoot computer-assisted engineering, design software and equipment to perform various engineering tasks.

PROGRAMME OUTCOMES

On successful completion of B.Tech. Computer Science Engineering programme from Presidency University a student will be able to

1. Apply mathematics, science and engineering fundamentals to computer science engineering.
2. Solve basic computer science engineering problems using first principles of mathematics and engineering sciences.
3. Design computing systems for domestic, commercial and industrial applications.
4. Investigate computer engineering problems including design of experiments, analysis and interpretation of data and synthesis of information to provide valid conclusions.
5. Select appropriate techniques, resources, and modern computer science engineering tools, including prediction and modelling, to complex engineering activities, with an understanding of the limitations.
6. Identify societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to computer science engineering practice.
7. Appraise professional ethics and responsibilities and norms of computer science engineering practice.
8. Relate the impact of computer science engineering solutions in a societal context and demonstrating knowledge of and need for sustainable development.
9. Assess management and business practices, such as risk and change management, and understanding their limitations in the context of computer science engineering.
10. Demonstrate the ability to work as an individual, and as a member or leader in diverse teams and in multi-disciplinary settings.
11. Write effective reports and design documentation, making effective presentations, and giving and receiving clear instructions.
12. Justify the need for engaging in independent and life-long learning.

DOs and DON'Ts in Laboratory:

1. Make entry in the Log Book as soon as you enter the Laboratory.
2. All the students should sit according to their roll numbers starting from their left to right.
3. All the students are supposed to enter the terminal number in the log book.
4. Do not change the terminal on which you are working.
5. All the students are expected to get at least the algorithm of the program/concept to be implement.
6. Strictly follow the instructions given by the teacher/Lab Instructor.

LAB INDEX

Design, Develop and implement following using Arduino, Raspberry Pi compiler and Python language in Linux/Windows environment.

Arduino Experiments	
1	Installation of arduino IDE & Arduino program to implement scrolling LED, to glow even/odd LED and to simulate using tinkerCAD.
2	Arduino program to demonstrate usage of push button to control the LED and to simulate using tinkerCAD.
3	Arduino program to demonstrate traffic control system and to simulate using tinkerCAD.
4	Arduino program to demonstrate usage of servo motor with potentio meter and to simulate using tinkerCAD.
5	Arduino program to implement soil moisture detector and to simulate using tinkerCAD.
6	Arduino program to implement smoke detector to detect fire and to simulate using tinkerCAD.
7	Arduino program to implement ultra sonic sensor to measure distance and to simulate using tinkerCAD.
8	Arduino program to implement an LCD to display text and to simulate using tinkerCAD.
9	Arduino program to implement and control DC motor and to simulate using tinkerCA.
Raspberry pi Experiments	
5	Installation of Raspberry pi software
6	Working basic commands on Raspberry pi & to demonstrate remote logging in raspberry pi
7	Raspberry pi program to implement blinking LED
8	Raspberry pi program to implement camera module for video
9	Raspberry pi program to obtain the temperature using DHT sensors

10	Using a Raspberry Pi with distance sensor (ultrasonic sensor HC-SR04)
11	Raspberry pi program to implement Garage spot light

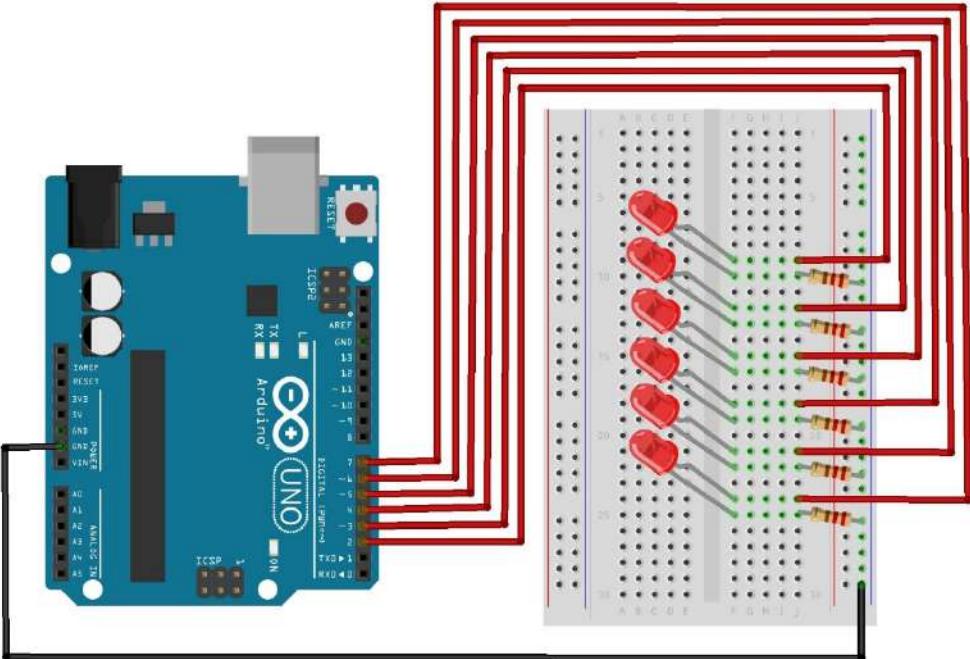
PRESIDENCY UNIVERSITY

Experiment No: 1

Arduino program to implement scrolling LED

This project will blink 6 LEDs, one at a time, in a back and forth formation. This type of circuit was made famous by the show Knight Rider which featured a car with looping LEDs.

Sl no.	Equipment's Needed
1	Arduino board(1)
2	Power cable(1)
3	Breadboard(1)
4	LED(6)
5	220 Ω Resistor(6)
6	Jumper Wires(7)

Step 1:	Connect the Arduino board to your computer using the USB cable.
Step 2:	Select the board and serial port as outlined in earlier section.
Step 3:	Make the Connection as follows:  <p>Made with fritzing</p>
Step 4:	Open project code to type following program /* For Loop Iteration Demonstrates the use of a for() loop.

Lights multiple LEDs in sequence, then in reverse.

The circuit:

* LEDs from pins 2 through 7 to ground

created 2006
by David A. Mellis
modified 30 Aug 2011
by Tom Igoe

This example code is in the public domain.

<http://www.arduino.cc/en/Tutorial/ForLoop>
*/

```
int timer = 100;      // The higher the number, the slower the timing.
```

```
void setup() {  
  // use a for loop to initialize each pin as an output:  
  for (int thisPin = 2; thisPin < 8; thisPin++) {  
    pinMode(thisPin, OUTPUT);  
  }  
}
```

```
void loop() {  
  // loop from the lowest pin to the highest:  
  for (int thisPin = 2; thisPin < 8; thisPin++) {  
    // turn the pin on:  
    digitalWrite(thisPin, HIGH);  
    delay(timer);  
    // turn the pin off:  
    digitalWrite(thisPin, LOW);  
  }
```

```
  // loop from the highest pin to the lowest:  
  for (int thisPin = 7; thisPin >= 2; thisPin--) {  
    // turn the pin on:  
    digitalWrite(thisPin, HIGH);  
    delay(timer);  
    // turn the pin off:  
    digitalWrite(thisPin, LOW);  
  }
```

```
}import RPi.GPIO as GPIO  
import time
```

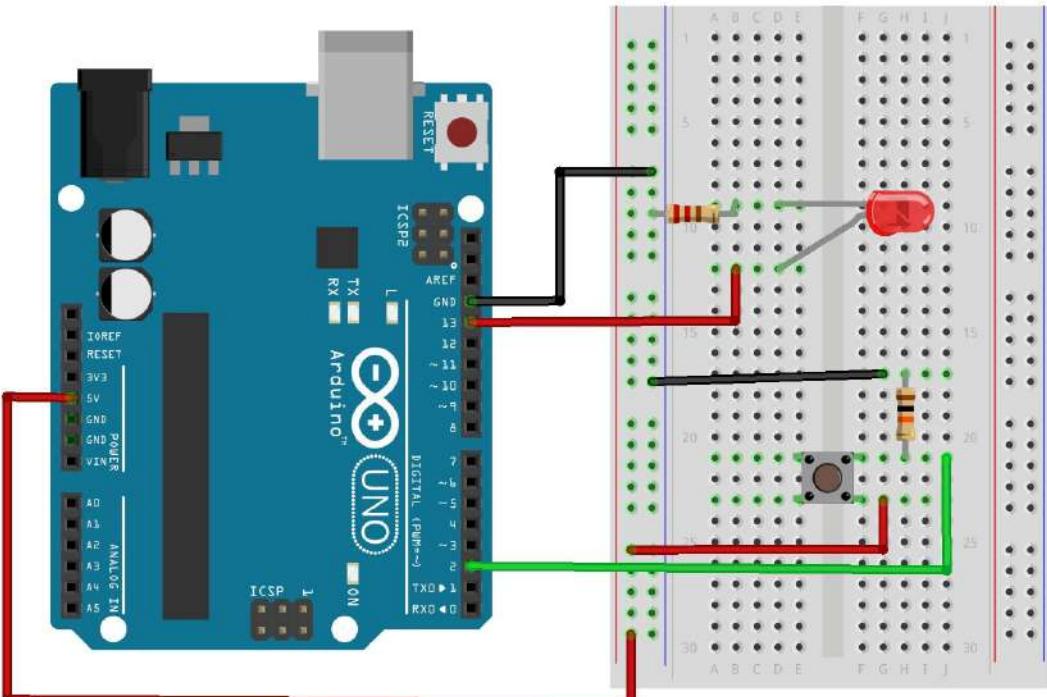
```
i=0  
while(i<4):  
    if(i%2==0):
```

```
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(18,GPIO.OUT)
    GPIO.output(18,GPIO.HIGH)
    time.sleep(3)
    GPIO.output(18,GPIO.LOW)
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(23,GPIO.OUT)
    GPIO.output(23,GPIO.HIGH)
    time.sleep(3)
    GPIO.output(23,GPIO.LOW)
    GPIO.cleanup()
else:
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(22,GPIO.OUT)
    GPIO.output(22,GPIO.HIGH)
    time.sleep(3)
    GPIO.output(22,GPIO.LOW)
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(25,GPIO.OUT)
    GPIO.output(25,GPIO.HIGH)
    time.sleep(3)
    GPIO.output(25,GPIO.LOW)
    GPIO.cleanup()
i+=1
```

Experiment No: 2

Arduino program to demonstrate usage of push button to control the LED

Sl no.	Equipment's Needed
1	Arduino board
2	Power cable
3	Push button Switch
4	Breadboard
5	LED
6	220 Ω Resistor
7	Jumper Wires

Step 1:	Connect the Arduino board to your computer using the USB cable.
Step 2:	Make the Connection as follows:  <p>Made with fritzing</p>
Step 3:	Open project code – and write the below program <pre>/* Button Turns on and off a light emitting diode(LED) connected to digital pin 13, when pressing a pushbutton attached to pin 2.</pre>

The circuit:

- * LED attached from pin 13 to ground
- * pushbutton attached to pin 2 from +5V
- * 10K resistor attached to pin 2 from ground

* Note: on most Arduinos there is already an LED on the board attached to pin 13.

created 2005
by DojoDave <<http://www.0j0.org>>
modified 30 Aug 2011
by Tom Igoe

This example code is in the public domain.

<http://www.arduino.cc/en/Tutorial/Button>
*/

```
// constants won't change. They're used here to
// set pin numbers:
const int buttonPin = 2;    // the number of the pushbutton pin
const int ledPin = 13;      // the number of the LED pin

// variables will change:
int buttonState = 0;        // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

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

	}
Step 4:	Select the board and serial port as outlined in earlier section.
Step 5:	Click upload button to send sketch to the Arduino

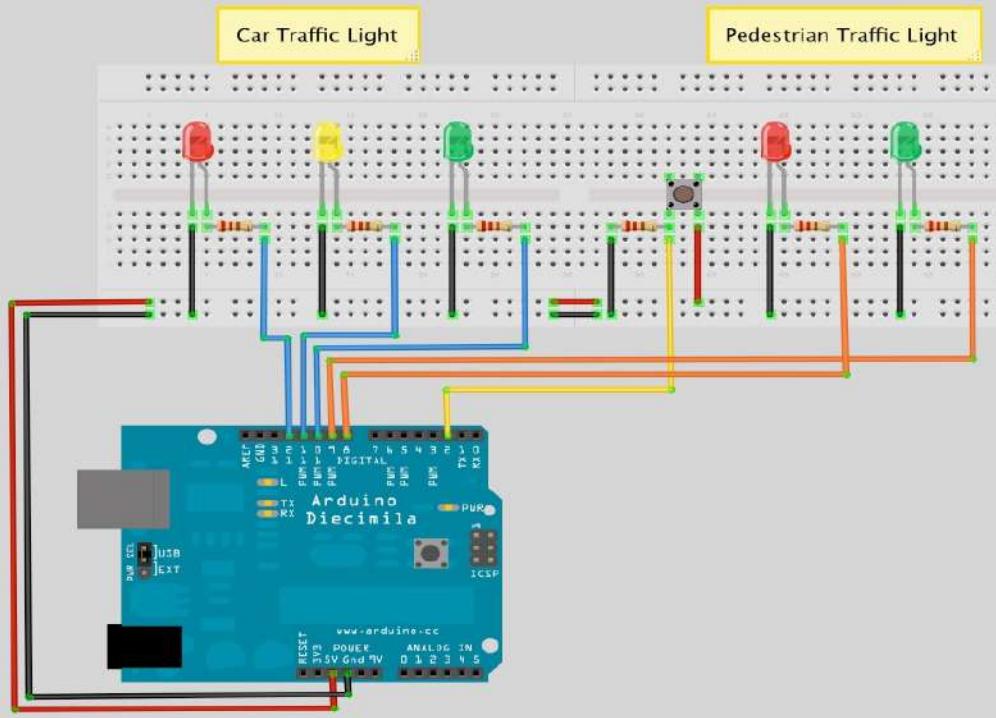
PRESIDENCY UNIVERSITY

Experiment No: 3

Arduino program to implement traffic control system

In this experiment we are going to make a traffic light with pedestrian light along with button, to request to cross the road. The Arduino will execute when the button is pressed by changing the state of light to make the cars stop and allow the pedestrian to cross safely.

Sl no.	Equipment's Needed
1	Arduino board(1)
2	Power cable(1)
3	Breadboard(1)
4	LED(6) 2x Red LED's 1x Yellow LED 2x Green LED's
5	220 Ω Resistor(6)
6	Jumper Wires(7)
7	Tactile Switch

Step 1:	Connect the Arduino board to your computer using the USB cable.
Step 2:	Select the board and serial port as outlined in earlier section.
Step 3:	Make the Connection as follows: 
Step 4:	Open project code to type following program

```
// Interactive Traffic Lights

int carRed = 12; // To assign the car lights
int carYellow = 11;
int carGreen = 10;
int pedRed = 9; // To assign the pedestrian lights
int pedGreen = 8;
int button = 2; // Tactile button pin
int crossTime = 5000; // time allowed to cross

unsigned long changeTime; // time since button pressed

void setup()
{
    pinMode(carRed, OUTPUT);
    pinMode(carYellow, OUTPUT);
    pinMode(carGreen, OUTPUT);
    pinMode(pedRed, OUTPUT);
    pinMode(pedGreen, OUTPUT);
    pinMode(button, INPUT); // button on pin 2
    // turn on the green light
    digitalWrite(carGreen, HIGH);
    digitalWrite(pedRed, HIGH);
}

void loop()
{
    int state = digitalRead(button);
    /* check if button is pressed and it is
       over 5 seconds since last button press */
    if (state == HIGH && (millis() - changeTime) > 5000)
    {
```

```
// Call the function to change the lights
changeLights();
}

}

void changeLights() {
    digitalWrite(carGreen, LOW); // green off
    digitalWrite(carYellow, HIGH); // yellow on
    delay(2000); // wait 2 seconds
    digitalWrite(carYellow, LOW); // yellow off
    digitalWrite(carRed, HIGH); // red on
    delay(1000); // wait 1 second till its safe
    digitalWrite(pedRed, LOW); // ped red off
    digitalWrite(pedGreen, HIGH); // ped green on
    delay(crossTime); // wait for preset time period
    // flash the ped green
    for (int x=0; x<10; x++) {
        digitalWrite(pedGreen, HIGH);
        delay(250);
        digitalWrite(pedGreen, LOW);
        delay(250);
    }
    // turn ped red on
    digitalWrite(pedRed, HIGH);
    delay(500);
    digitalWrite(carYellow, HIGH); // Yellow will switch on
    digitalWrite(carRed, LOW); // red will switch off
    delay(1000);
    digitalWrite(carGreen, HIGH);
    digitalWrite(carYellow, LOW); // Yellow will switch off

    // record the time since last change of lights
    changeTime = millis();
```

	// Retun / Loop }
--	----------------------

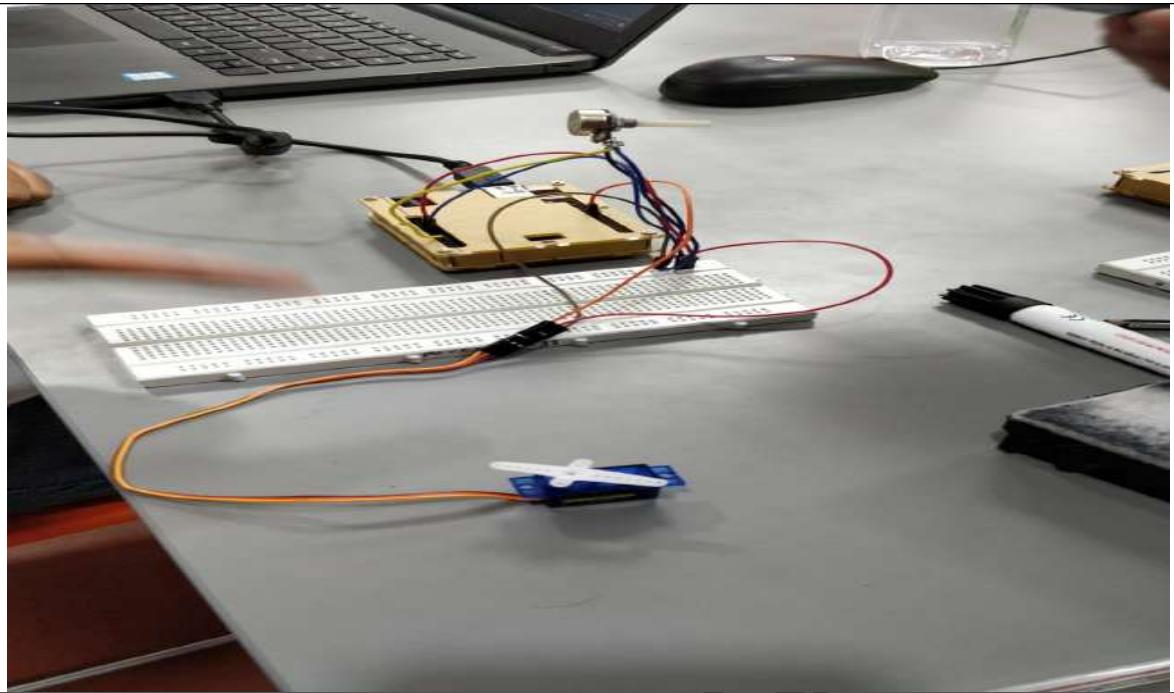
PRESIDENCY UNIVERSITY

Experiment No: 4

Arduino program to demonstrate control of servo motor using potentio meter

Sl no.	Equipment's Needed
1	Arduino board
2	Power cable
3	Servo Motor
4	Potentio meter

Step 1:	Connect the Arduino board to your computer using the USB cable.
Step 2:	Select the board and serial port as outlined in earlier section.
Step 3:	<p>Lots of IoT applications with respect to robotics use servo motor. Servo motor comes with 3 different arms with 2 variants full arm and half arm and runs 1800. Black pin->Gnd pin of arduino Red Pin-> +5V of arduino Orange ->Used to send pulse width modulation, and is connected to pin no 09 from arduino</p> <p>Potentio meter- is a three-terminal resistor with a sliding or rotating contact that forms an adjustable voltage divider. If only two terminals are used, one end and the wiper, it acts as a variable resistor or rheostat. 1st Nob->Gnd pin of arduino 2nd Nob->A0 pin of arduino 3rd Nob->+5v pin of arduino</p>
Step 4:	Make the Connection as follows:



Step 5:	<p>Open project code to type following program</p> <pre>/* Controlling a servo position using a potentiometer (variable resistor) by Michal Rinott <http://people.interaction-ivrea.it/m.rinott> modified on 8 Nov 2013 by Scott Fitzgerald http://www.arduino.cc/en/Tutorial/Knob */ #include <Servo.h> Servo myservo; // create servo object to control a servo int potpin = 0; // analog pin used to connect the potentiometer int val; // variable to read the value from the analog pin void setup() { myservo.attach(9); // attaches the servo on pin 9 to the servo object } void loop() { val = analogRead(potpin); // reads the value of the potentiometer (value between 0 and 1023) val = map(val, 0, 1023, 0, 180); // scales it to use it with the servo (value between 0 and 180) myservo.write(val); // sets the servo position according to the scaled value delay(15); } // waits for the servo to get there</pre>
---------	---

PRESIDENCY UNIVERSITY

Experiment No: 5

Installation of Raspberry pi software

Sl no.	Equipment's Needed
1	Raspberry Pi
2	VGA cable
3	Power Cable

Introduction: The Raspberry Pi is a fully-fledged mini computer, capable of doing whatever you might do with a computer. It comes with 4x USB, HDMI, LAN, built-in Bluetooth/WiFi support, 1GB RAM, 1.2GHz quad-core ARM CPU, 40 GPIO (General Purpose Input Output) pins, audio and composite video output, and more.



One can use Raspberry Pis as home security cameras, server monitoring devices, cheap headless machines (basically running low-weight scripts 24/7 with a low cost-to-me)... others have used them for media centers and even for voice-enabled IoT devices. The possibilities are endless, but first we need to get acquainted!

Make sure that, if you do get a case, it has openings for the GPIO pins to be connected, you will also need a [1000mA+ mini usb power supply](#) and at least an 8GB micro SD card, but suggested is [16 GB micro SD card](#) or greater.

You will also want to have a spare monitor (HDMI), keyboard, and mouse handy to make things easier when first setting up. You wont will eventually be able to control your Pi remotely, so you wont always need a separate keyboard, mouse, and monitor. If you don't have a monitor with HDMI

input, you can buy something like an HDMI to DVI converter.

If you're using an older version board, please see what you might need to change, for example, the older Raspberry Pis take a full-sized SD card, but the latest model requires a micro SD card. Also, the Raspberry Pi 3 Model B has built-in wifi, where the older models will require a wifi dongle.

A typical Raspberry Pi shopping list, assuming you have a mouse, keyboard, and HDMI monitor that you can use temporarily while setting up is:

1. [Raspberry Pi](#) -
2. [1000mA+ mini usb power supply](#) -
3. [16 GB micro SD card](#) –

Additionally, if you plan to join us on the initial GPIO (General Purpose Input Output pins) tutorials, you will also want to pick up:

1. 10 x [Male-to-Female jumper wires](#) (you should consider just buying a bunch of these so you have plenty in the future).
2. 1 x [Breadboard](#) (You may also want multiples of these)
3. 3 x [LED light](#) (...more wouldn't hurt)
4. ~6 x Resistors (between 300 and 1K Ohm). You will need at least 1K and 2K ohms for the distance sensor, then ~300-1K resistance per LED bulb. You probably should just buy a kit, they're super cheap.
5. 1 x [HC-SR04 Ultrasonic Distance Sensor](#) (...you know what I'm going to say...think about maybe a few.)
6. 1 x [Raspberry Pi](#) camera module. You only need one of these!

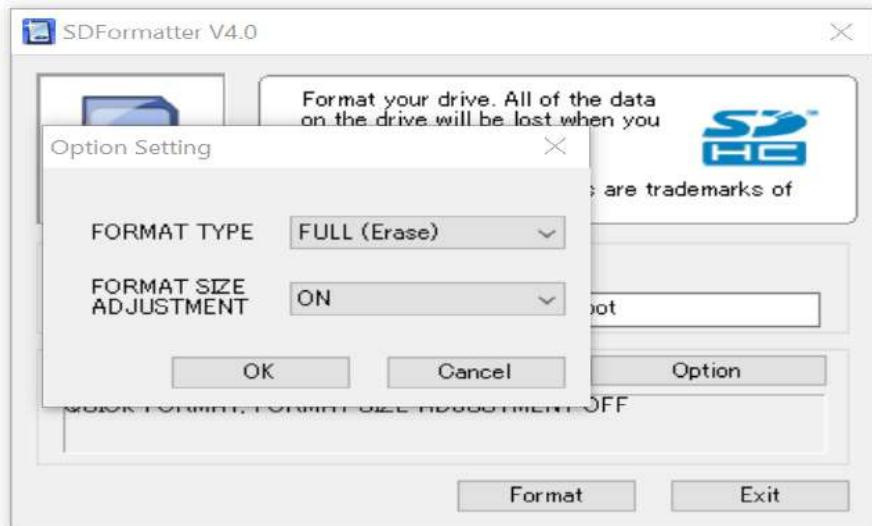
For the jumpers, breadboard, and leds, you could also just buy a kit, something like: [this GPIO starter kit](#).

There are also a few ways to install and use an operating system on the Raspberry Pi. The most user-friendly method is to use the NOOBS (New Out of Box Software) installer. If you're comfortable enough, you can just simply download the operating system ISO, format the SD card, mount the ISO, and boot the Pi. If that sounds like gibberish to you, then follow along with the NOOBS installation option.

While we're working with the SD card, let's go ahead and [Download NOOBS](#), which is just over 1GB.

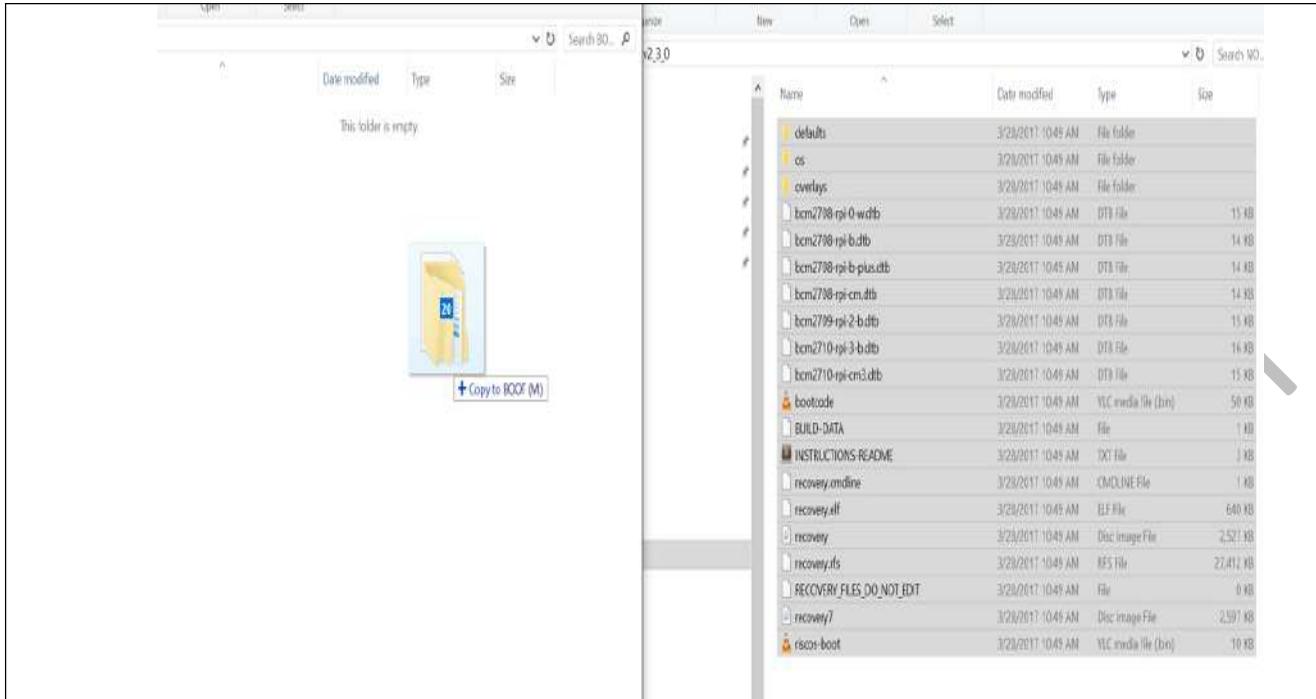
First, we must format the SD card. If you are on Windows, you can use [SD Formatter](#). Mac users can also use SD Formatter, but they have a built in formatter, and Linux users can use GParted. Whatever the case, you need to format the SD card, do not do a "quick format" and do make sure

you have the "resize" option on. Using SDFormatter on Windows, and choosing options:



This should go without saying, but do make sure you're formatting the right drive. This will format any flash drive, in alphabetical order. If you had something plugged in already, like your favorite USB drive, and forgot about it, that'd likely be the default choice to format, and then you'd spend all afternoon trying to recover your data rather than enjoying playing with your Raspberry Pi.

Now, assuming you've downloaded the NOOBS package, let's go ahead and extract that. Now, we want to copy all these NOOBS contents to our SD Card. Do not drag the directory, but rather the contents:



While that's transferring, let's talk about a few things on the actual Raspberry Pi board:



The GPIO (General Purpose Input/Output) pins are underlined in blue. We can use these to control peripheral devices like motors, servos, and more. Circled in red is the micro USB power input for the board. In orange, the HDMI output port. The yellow is where you can plug in the Raspberry Pi camera module. The grey circle has the USB ports. This is obviously not everything, but these are the main things to note.

Once everything is transferred to the micro SD card, you can put it in the Raspberry Pi. The slot is

on the bottom side of the board, circled in

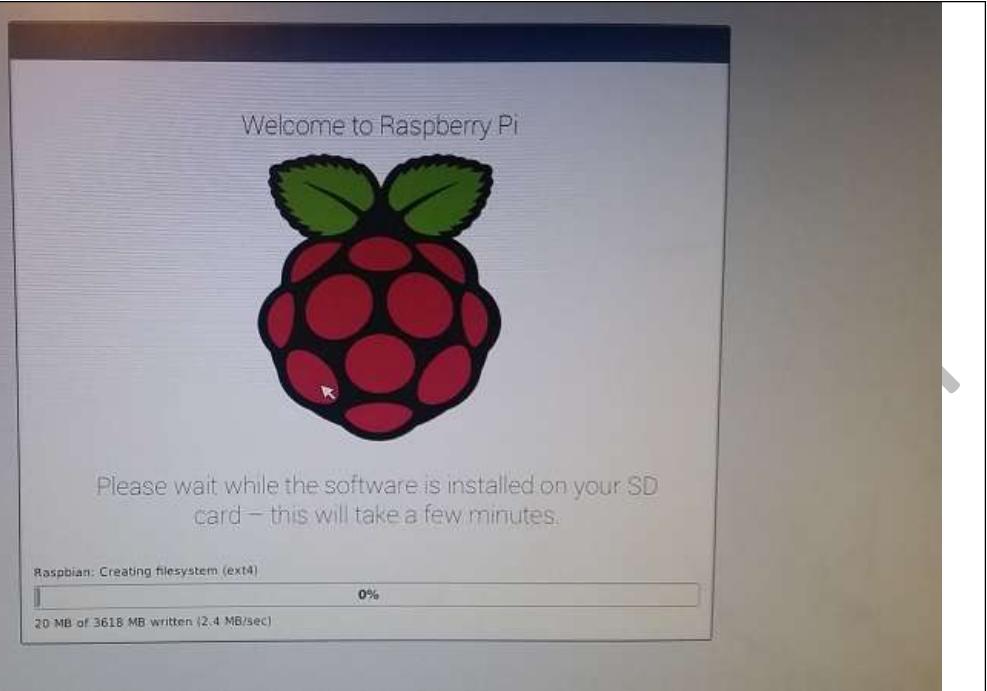


J now here: yell

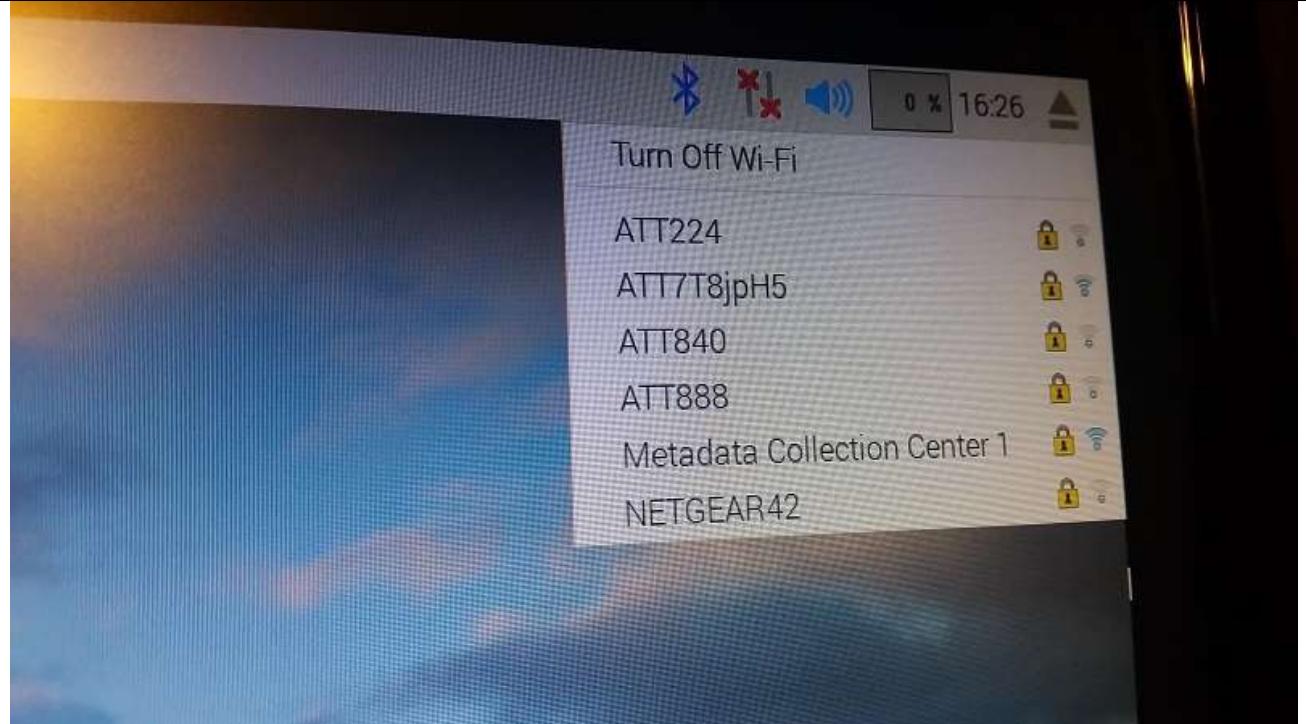
Let the process go, this will take a while, something like 20-30 minutes or so.



sa



Once that's done, hit okay and the device should reboot to desktop. While on the desktop, wait for a moment for wifi to start up and find available connections. Connect to your wifi network if possible. You can also plug directly in with an ethernet cable if you don't have wifi. You can also just continue interacting directly with the Raspberry Pi with the mouse and keyboard connected to it if you like, but I prefer to access it remotely.



Once we've connected to our network, we'd like to actually interface with the Pi. First, we want to update. Open a terminal by either right clicking on the desktop and opening terminal that way, or by doing control+alt+t. Now, in the terminal, do:

```
$sudo apt-get update
```

and then

```
$ sudo apt-get upgrade
```

You do not type the \$ sign, it's there to denote when you're typing something in the command line.

The upgrade might take a minute. While we wait, your Raspberry Pi's default credentials

are: username: pi password: raspberry. For some reason, the apt-get upgrade for me was taking absurdly long. You need to be connected to your network, and have internet access, so make sure you have those things first before doing this, but still was having trouble. One can solve this by doing:

```
$ sudo nano /etc/apt/sources.list
```

Then replace everything here with:

```
deb http://archive.raspbian.org/raspbian jessie main contrib non-free  
deb-src http://archive.raspbian.org/raspbian jessie main contrib non-free
```

control+x, y, enter

```
$ sudo apt-get dist-upgrade
```

```
$ sudo apt-get update
```

```
$ sudo apt-get upgrade
```

To save some space you can also do: \$ sudo apt-get purge wolfram-engine and then \$ sudo apt-get autoremove. This alone freed up almost 700mb of space for me.

This is going to conclude the first part of this tutorial series. In the next tutorial, we're going to cover how we can remotely access our Raspberry Pi.

Experiment No: 6

Working basic commands on Raspberry pi

Sl no.	Equipment's Needed
1	Raspberry Pi
2	VGA cable

Introduction:

In this Experiment, we're going to have a quick crash course for using the terminal. Most of our interactions with the Raspberry Pi will be via shell, since this is the simplest and most lightweight to keep your Pi accessible.

To begin, let's assume you've just logged in. The location that you will usually log in to will be the root directory for your user. In this case, our user is "pi," so we log in to /home/pi. We can confirm this by using the command: pwd (print working directory)

```
pi@raspberrypi:~ $ pwd  
/home/pi
```

Often times, to denote the terminal, you will see people use the \$ sign. Since, most of the time, people won't be using the same usernames and hostnames, this is the standard to denote that we're operating in the terminal, so, instead, if you googled how you get your current directory in linux, you'd probably see: \$ pwd. This doesn't mean that you should actually type the \$ sign, it's just meant to denote that you're in the terminal.

Next, many times you might see a "permission denied" error when trying to do something. For many tasks, you need to act as the super user, like the administrator account. The command to act as the super user is sudo, which is short for "super user do." An example of this was when we wanted to update and upgrade, we used sudo. Be careful when using sudo to create files...etc. The creator is the owner, so if you use sudo to create the file, it can only be further edited by the super user. Many files will require sudo to edit, however.

To move around the system, you can use cd, which stands for "change directory." At any time, we can use ~ to reference the current user's home. In our case ~ is short for /home/pi. We can

```
change directories into there with $ cd ~.
```

We were already there, but, just in case you weren't, you are now!

Next, to discover the contents of the directory you are in, you can use ls to list directory contents:

```
$ ls
```

```
pi@raspberrypi:~ $ ls
```

```
Desktop Documents Downloads Music Pictures Public python_games Templa
```

It won't always be the case, but often you will see different colors for different types of files. In our case, we just have directories, so they are all the same color.

Let's assume we want to make our own directory, to do this, we can use the mkdir command, short for make dir.

```
$ mkdir example
```

We can list the contents again:

```
$ ls
```

Seeing our new directory:

```
pi@raspberrypi:~ $ ls
```

```
Desktop Documents Downloads example Music Pictures Public python_games
```

Then we can change directories into our new directory:

```
$ cd example
```

Maybe we don't want to be here. We can move backwards with: \$ cd ..

```
pi@raspberrypi:~/example $ cd ..
```

```
pi@raspberrypi:~ $
```

We can also move back a few directories at a time: \$cd ../../

```
pi@raspberrypi:~ $ cd ../../  
pi@raspberrypi:/ $ ls  
bin boot debian-binary dev etc home lib lost+found man media mnt opt proc root run  
sbin srv sys tmp usr var
```

Let's go into our example dir now:

```
$ cd example
```

Next, we can create a simple file with any of the built in editors. There are many to choose from, I think nano is the easiest, so I will use that:

```
$ nano test.py
```

Nano can be used to both create or edit files. If the file doesn't yet exist, it will be created. If it does exist, it will allow you to save a new one.

Let's just add: print("hi") to this file. When done, we can exit with control+x, then press y to save, and then enter to keep the name.

To run this file, we can do: \$ python test.py

```
pi@raspberrypi:~/example $ python test.py  
hi
```

Next, let's make another directory:

```
$ mkdir downloads
```

```
$ cd downloads
```

To demonstrate remote logging in raspberry pi

Sl no.	Equipment's Needed
1	Raspberry Pi
2	VGA cable

Introduction :

In this Experiment, we're going to cover how we can remotely access our Raspberry Pi, both with SSH and with a remote desktop client. We want to eventually be able to remotely access our Raspberry Pi because much of the "value" of the Raspberry Pi is its size, and that it can be put in a variety of places that we might not want to have a keyboard, mouse, and monitor attached to it at all times and we probably don't want to have to carry over all of this stuff when we do want to access it.

First, let's connect via shell (SSH). Open the terminal on the Raspberry Pi (control+alt+t), and type ifconfig. If you're connected via wifi, then go under the wlan section, and look for your inet address. This will be your local ip, something like 192.168.XX.XXX. We can use this to connect via SSH (user: pi, pass: raspberry), BUT we first have to enable the SSH server. To do this, type sudo raspi-config. Here, we can do quite a few things, but let's head into option #5 interfacing options, next choose the 2nd option for SSH and enable the server. Once this is done, you can shell into the Raspberry Pi.

On Windows, you will need to use an SSH client. But the recommended is [PuTTY](#). Once downloaded, you can open PuTTy, fill in "host name" field with your Pi's local ip, hit enter, and then you will be asked for a username and password.

When successful, you should have something like:

```
pi@raspberrypi: ~
Session Special Command Window Logging Files Transfer Hangup ?
login as: pi
pi@192.168.0.100's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Mar 28 21:25:26 2017

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
a new password.

pi@raspberrypi:~ $
```

00:00:40 Conn: SSH/22

This is identical to the terminal you accessed earlier from the Pi's desktop.

Now, sometimes, you might really want to get access to the desktop instead of just the terminal. To do this, you need the "host" to have remote desktop capabilities, as well as whatever remote PC you attempt to use to access it. First, let's deal with the Raspberry Pi. We're going to install xrdp.

```
sudo apt-get remove xrdp vnc4server tightvncserver
```

```
seudo apt-get update
```

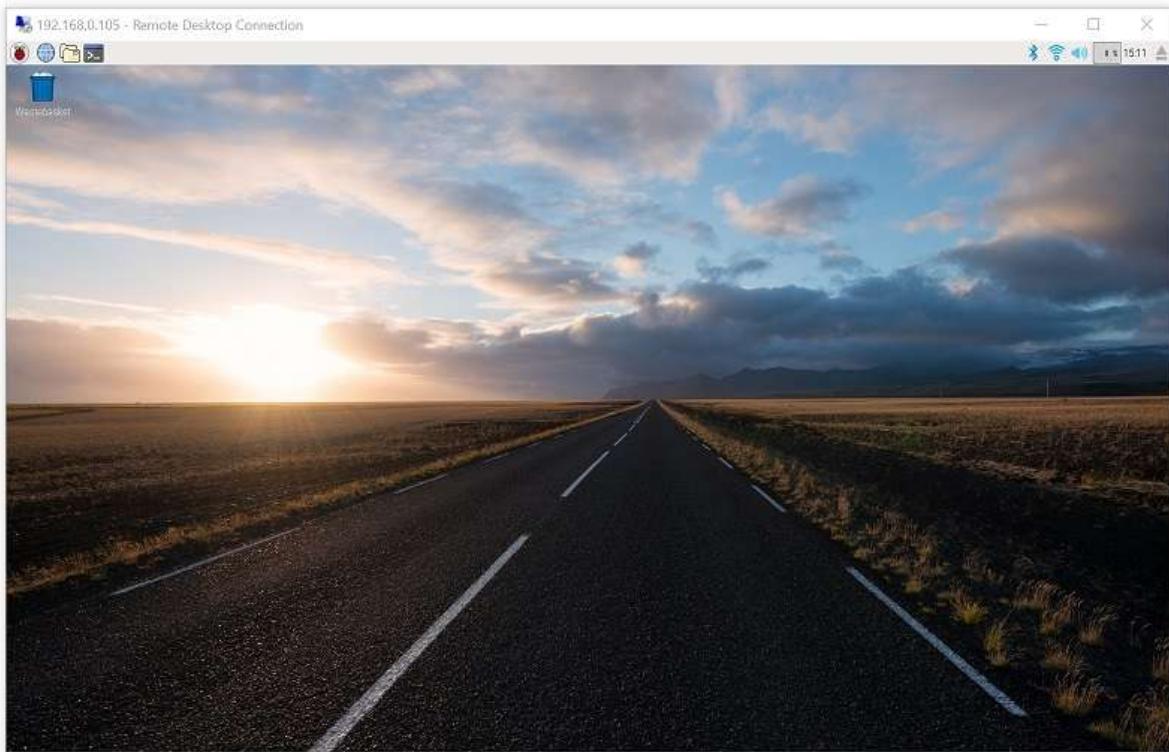
```
sudo apt-get install xrdp
```

```
sudo apt-get install tightvncserver
```

Now let's deal with the computer from which you plan to connect your Raspberry Pi.

For Mac and Windows, you can use the Microsoft application called Remote Desktop. On linux, you can use grdesktop (sudo apt-get install grdesktop). All three of these examples are going to act the same way. Run them, fill in the IP address of the Raspberry pi, the username, the password, and connect!

When done, you should have something like:



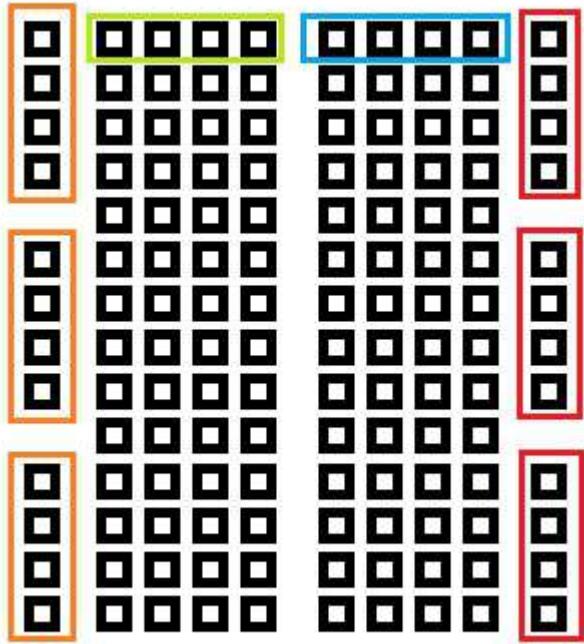
With the SSH server turned on, you will likely be seeing warnings every time you log in that your password is still the default password. The SSH server is turned off by default because, as the Pi has gained popularity, people who might not realize the security risk are using it in places like their homes and businesses. All it takes is someone to connect to your wifi, scan for other local ips, and try them all with default usernames and passwords for various devices, like the Raspberry Pi, and boom they're in. If you want to change your password, you can do sudo raspi-config, and it's the first option.

Experiment No: 7

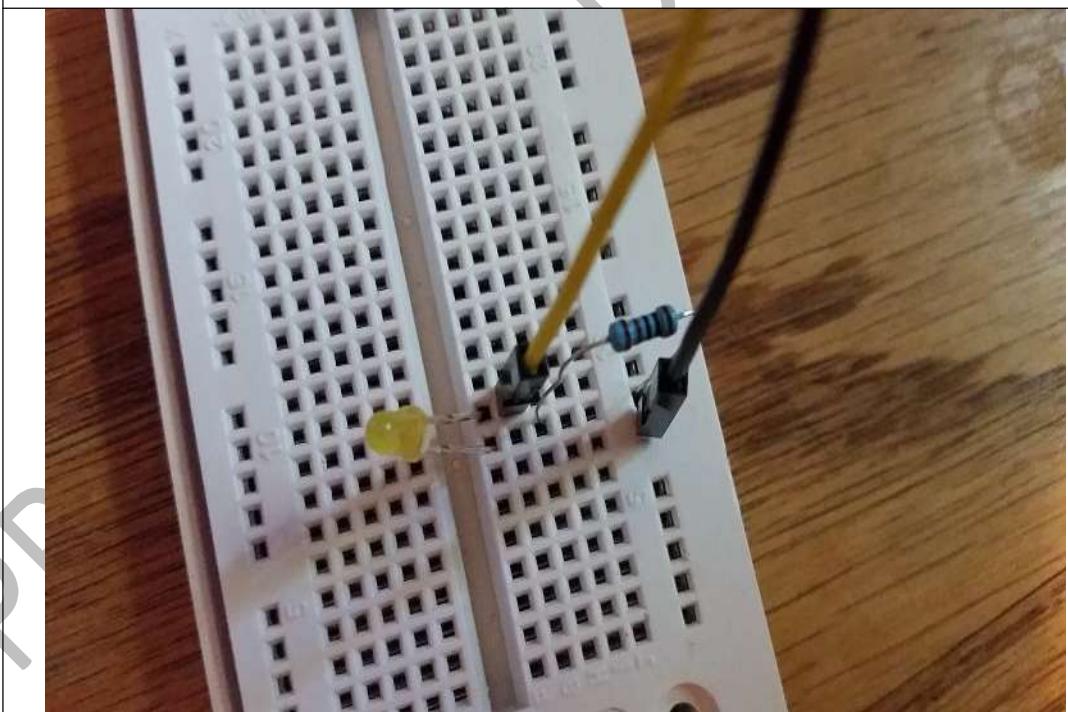
Raspberry pi program to implement blinking LED

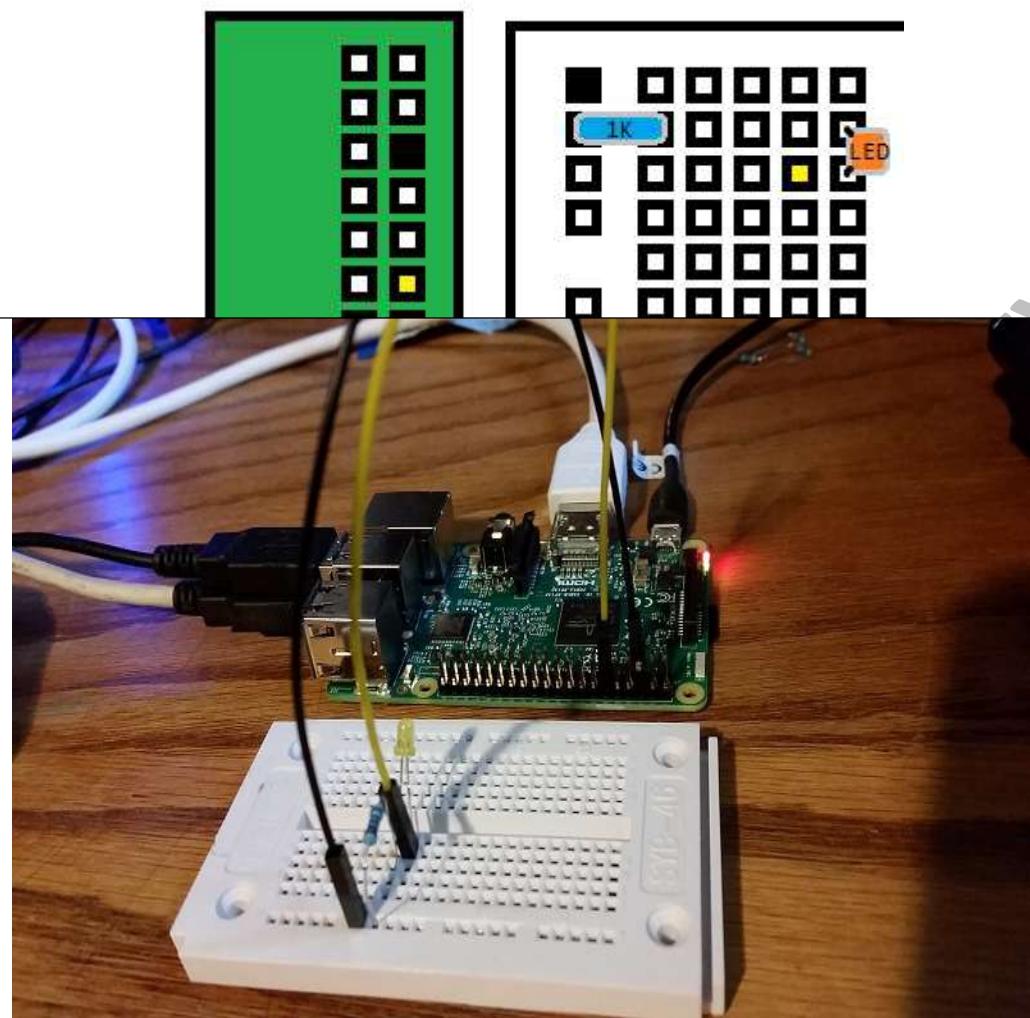
Sl no.	Equipment's Needed
1	Raspberry Pi
2	VGA cable
3	Power Cable
4	Jumper Cables

Step 1:	Understanding of Raspberry pi Pin out diagram and Bread board connections																																																												
	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>3.3V</td><td>GPIO2</td><td>5V</td></tr> <tr><td></td><td>GPIO3</td><td>GND</td></tr> <tr><td></td><td>GPIO4</td><td>GPIO14</td></tr> <tr><td></td><td>GND</td><td>GPIO15</td></tr> <tr><td></td><td>GPIO17</td><td>GPIO18</td></tr> <tr><td></td><td>GPIO27</td><td>GND</td></tr> <tr><td></td><td>GPIO22</td><td>GPIO23</td></tr> <tr><td></td><td>3.3V</td><td>GPIO24</td></tr> <tr><td></td><td>GPIO10</td><td>GND</td></tr> <tr><td>Used by DotStars GPIO unavailable</td><td>GPIO9</td><td>GPIO25</td></tr> <tr><td></td><td>GPIO11</td><td>GPIO8</td></tr> <tr><td></td><td>GND</td><td>GPIO7</td></tr> <tr><td></td><td>DNC</td><td>DNC</td></tr> <tr><td></td><td>GPIO5</td><td>GND</td></tr> <tr><td></td><td>GPIO6</td><td>GPIO12</td></tr> <tr><td></td><td>GPIO13</td><td>GND</td></tr> <tr><td></td><td>GPIO19</td><td>GPIO16</td></tr> <tr><td></td><td>GPIO26</td><td>GPIO20</td></tr> <tr><td></td><td>GND</td><td>GPIO21</td></tr> </table> <p style="text-align: right;">Pins above this line are present on all Raspberry Pi boards</p> <p style="text-align: right;">Pins below this line on Models A+, B+ and Pi 2</p> <p style="text-align: center;">GPIO Availability:</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>No</td></tr> <tr><td>Maybe</td></tr> <tr><td>Yes</td></tr> </table>	3.3V	GPIO2	5V		GPIO3	GND		GPIO4	GPIO14		GND	GPIO15		GPIO17	GPIO18		GPIO27	GND		GPIO22	GPIO23		3.3V	GPIO24		GPIO10	GND	Used by DotStars GPIO unavailable	GPIO9	GPIO25		GPIO11	GPIO8		GND	GPIO7		DNC	DNC		GPIO5	GND		GPIO6	GPIO12		GPIO13	GND		GPIO19	GPIO16		GPIO26	GPIO20		GND	GPIO21	No	Maybe	Yes
3.3V	GPIO2	5V																																																											
	GPIO3	GND																																																											
	GPIO4	GPIO14																																																											
	GND	GPIO15																																																											
	GPIO17	GPIO18																																																											
	GPIO27	GND																																																											
	GPIO22	GPIO23																																																											
	3.3V	GPIO24																																																											
	GPIO10	GND																																																											
Used by DotStars GPIO unavailable	GPIO9	GPIO25																																																											
	GPIO11	GPIO8																																																											
	GND	GPIO7																																																											
	DNC	DNC																																																											
	GPIO5	GND																																																											
	GPIO6	GPIO12																																																											
	GPIO13	GND																																																											
	GPIO19	GPIO16																																																											
	GPIO26	GPIO20																																																											
	GND	GPIO21																																																											
No																																																													
Maybe																																																													
Yes																																																													



Step 2: Make the connections as follows





Step 3: Write the python program to implement blinking LED

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

GPIO.setup(18, GPIO.OUT)
GPIO.output(18, GPIO.HIGH)

time.sleep(3)

GPIO.output(18, GPIO.LOW)
GPIO.cleanup()
```

Experiment No: 8

Raspberry pi-camera module

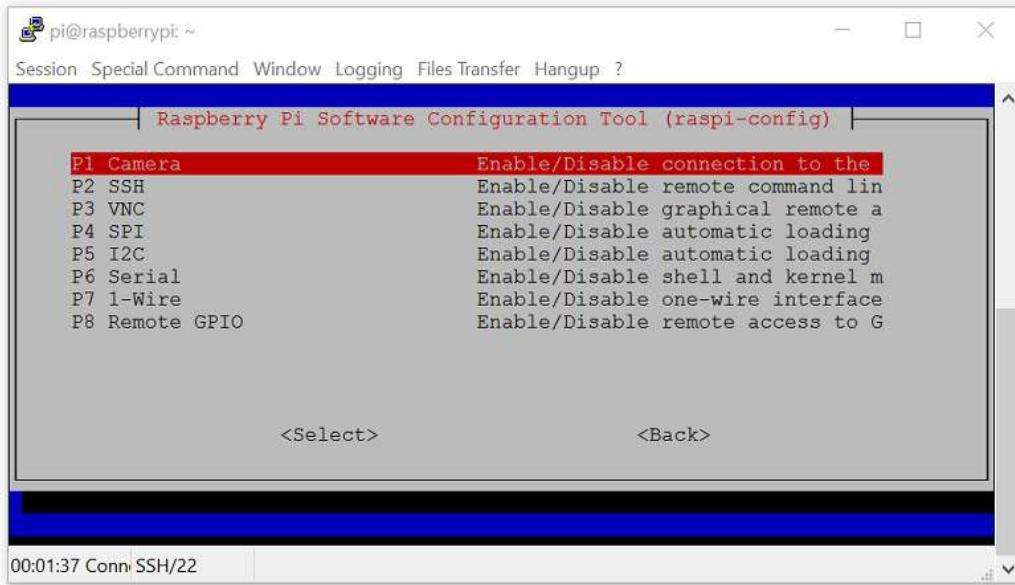
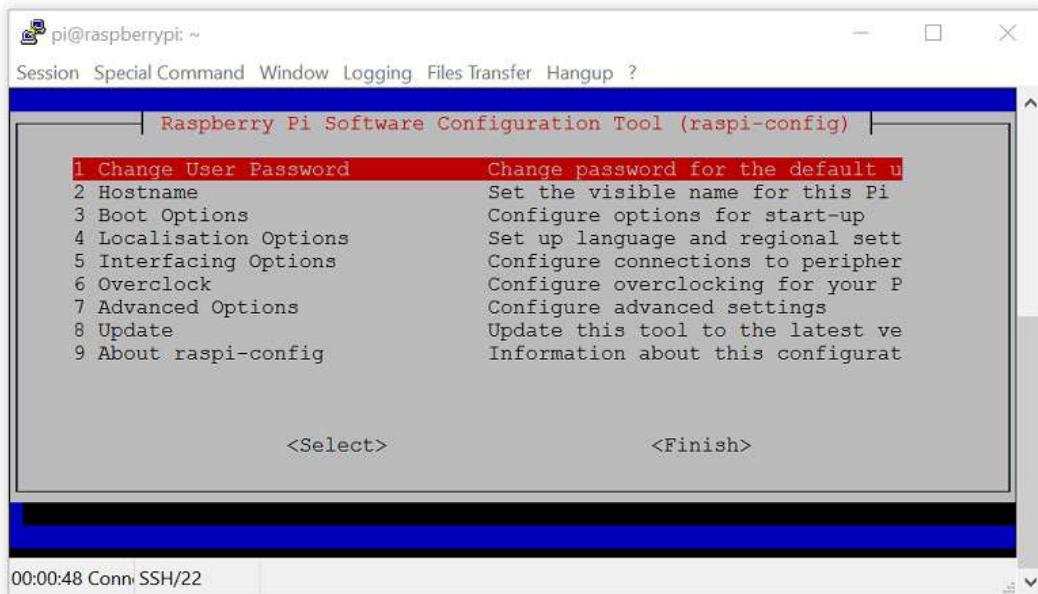
Sl no.	Equipment's Needed
1	Raspberry Pi
2	Raspberry Pi Camera Module
3	VGA cable
4	Power Cable

Step 1: Open the notch “Connect camera” module to raspberry pi Blue should be facing towards Ethernet port.



Step 2: `pi@raspberrypi:~ $cd Desktop/`

Step 3: Enable Camera
`pi@raspberrypi:~/Desktop$ sudo raspi-config`
Interfacing option → Enable camera



Yes to enable, and then go ahead and reboot:

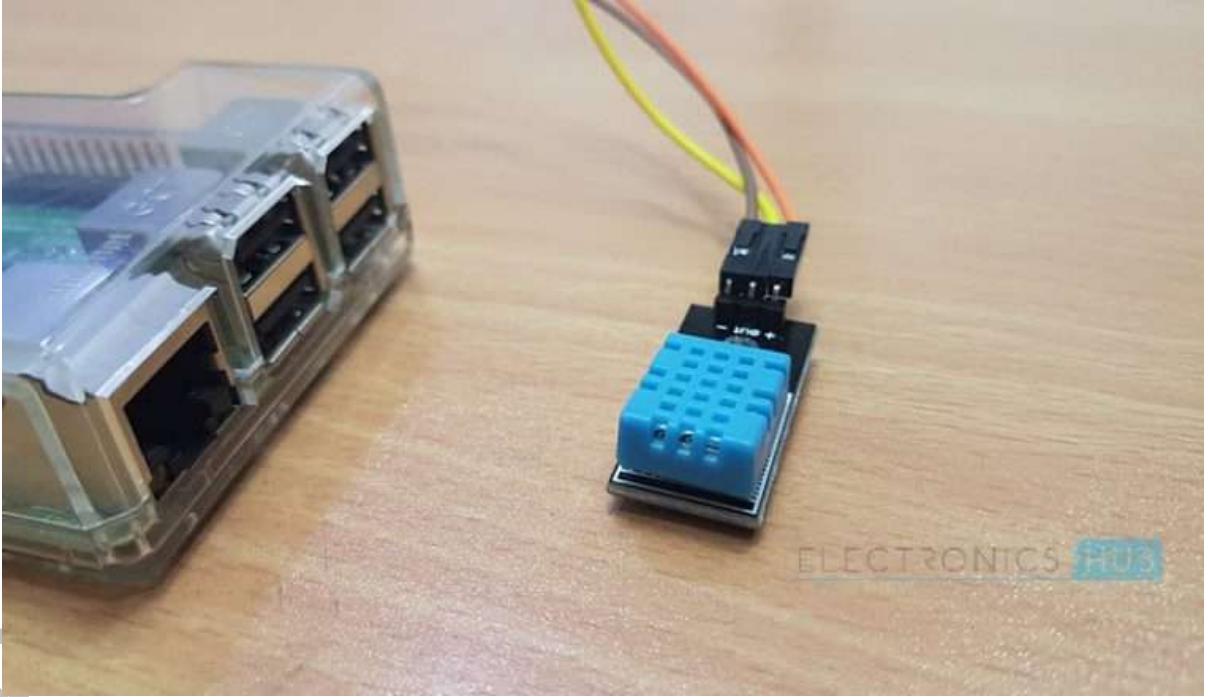


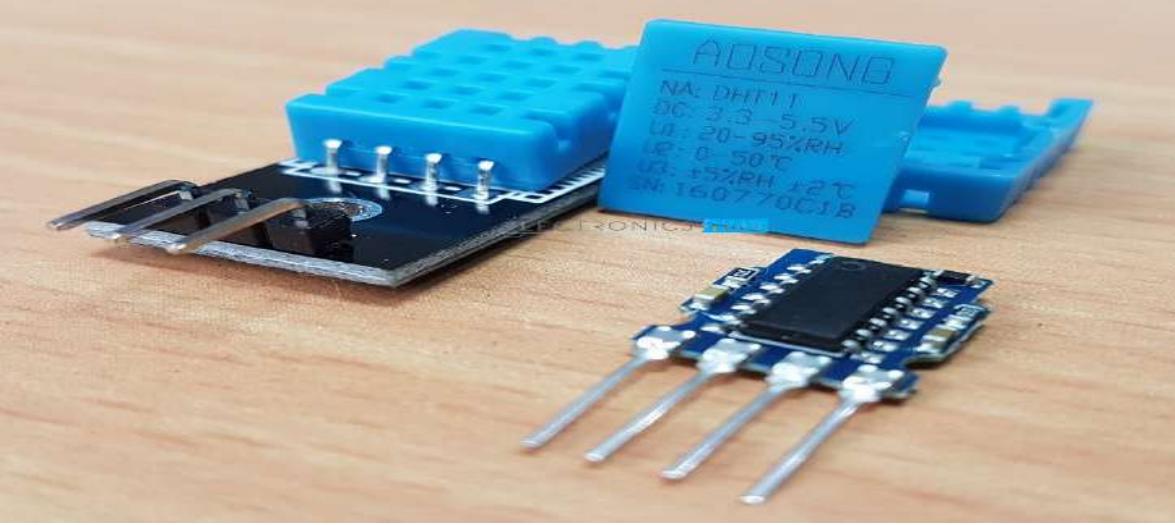
Step4:	Let's open the terminal next (control+alt+t) and do a \$ sudo apt-get install python3-picamera. You will likely find that you already have it, but we want to be sure. Now, in our cameraexample.py file:
Step 5:	Write the Python program <pre>pi@raspberrypi:~/Desktop\$ nano cameraexample.py</pre> <pre>import picamera import time camera = picamera.PiCamera() camera.capture('example.jpg') camera.vflip = True camera.capture('example2.jpg') camera.start_recording('examplevid.h264') time.sleep(5) camera.stop_recording()</pre>
Step 6:	Command to view recorded video \$ omxplayer examplevid.h264.

Experiment No: 09

Raspberry pi program to implement temperature sensor

Sl no.	Equipment's Needed
1	Raspberry Pi Model 3
2	Raspberry Pi Camera Module
3	Digital humidity and temperature sensor-(DHT 22)
4	VGA cable
5	Power Cable

Step No.	Raspberry Pi DHT11 Humidity and Temperature Sensor Interface
1	<p>Raspberry Pi DHT11 Humidity and Temperature Sensor Interface</p> <p>In this project, we will learn about DHT11 Humidity and Temperature Sensor and how the Raspberry Pi DHT11 Humidity Sensor interface works. By Interfacing DHT11 Temperature and Humidity Sensor with Raspberry Pi, you can implement a basic IoT Project like a simple Weather Station.</p>  <p>Overview</p> <p>DHT11 is a Digital Sensor consisting of two different sensors in a single package. The sensor contains an NTC (Negative Temperature Coefficient) Temperature Sensor, a Resistive-type Humidity Sensor and an 8-bit Microcontroller to convert the analog signals from these sensors and produce a Digital Output.</p>



We know that the output from the DHT11 Sensor is Digital. But how exactly we can read this digital data?

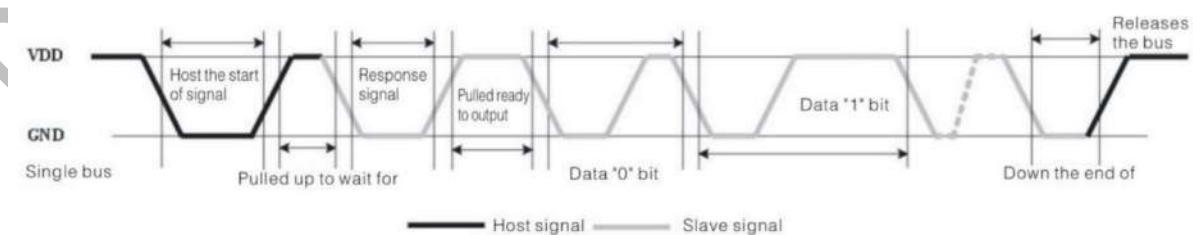
Reading Digital Output from DHT11

DHT11 uses a Single bus data format for communication. Only a single data line between an MCU like Arduino or Raspberry Pi and the DHT11 Sensor is sufficient for exchange of information.

In this setup, the Microcontroller acts as a Master and the DHT11 Sensor acts as a Slave. The Data OUT of the DHT11 Sensor is in open-drain configuration and hence it must always be pulled HIGH with the help of a $5.1\text{K}\Omega$ Resistor.

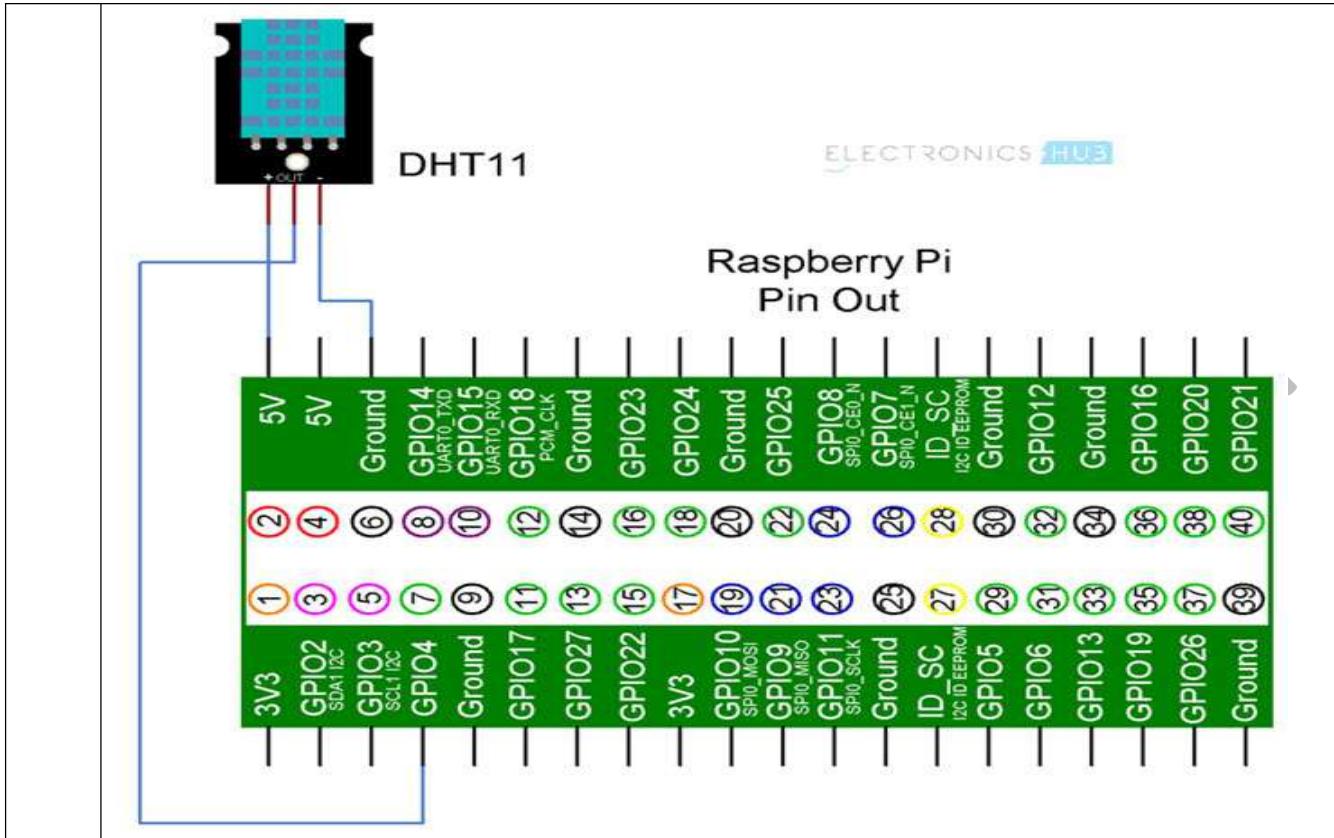
This pull-up will ensure that the status of the Data is HIGH when the Master doesn't request the data (DHT11 will not send the data unless requested by the Master).

Now, we will see how the data is transmitted and the data format of the DHT11 Sensor. Whenever the Microcontroller wants to acquire information from DHT11 Sensor, the pin of the



Microcontroller is configured as OUTPUT and it will make the Data Line low for a minimum

	<p>time of 18ms and releases the line. After this, the Microcontroller pin is made as INPUT.</p> <p>The data pin of the DHT11 Sensor, which is an INPUT pin, reads the LOW made by the Microcontroller and acts as an OUTPUT pin and sends a response of LOW signal on the data line for about 80µs and then pulls-up the line for another 80µs.</p> <p>After this, the DHT11 Sensor sends a 40 bit data with Logic ‘0’ being a combination of 50µs of LOW and 26 to 28µs of HIGH and Logic ‘1’ being 50µs of LOW and 70 to 80µs of HIGH.</p> <p>After transmitting 40 bits of data, the DHT11 Data Pin stays LOW for another 50µs and finally changes its state to input to accept the request from the Microcontroller.</p> <p>NOTE: We have implemented this logic while programming the Arduino. But for Raspberry Pi, we used a library that takes care of all these things.</p> <p>Raspberry Pi DTH11 Humidity and Temperature Sensor Interface</p> <p>By interfacing the DHT11 Sensor with Raspberry Pi, you can build your own IoT Weather Station. All you need to implement such IoT Weather is a Raspberry Pi, a DHT11 Humidity and Temperature Sensor and a Computer with Internet Connectivity.</p>
2	<p>Circuit Diagram</p> <p>The following is the circuit diagram of the DHT11 and Raspberry Pi Interface.</p>

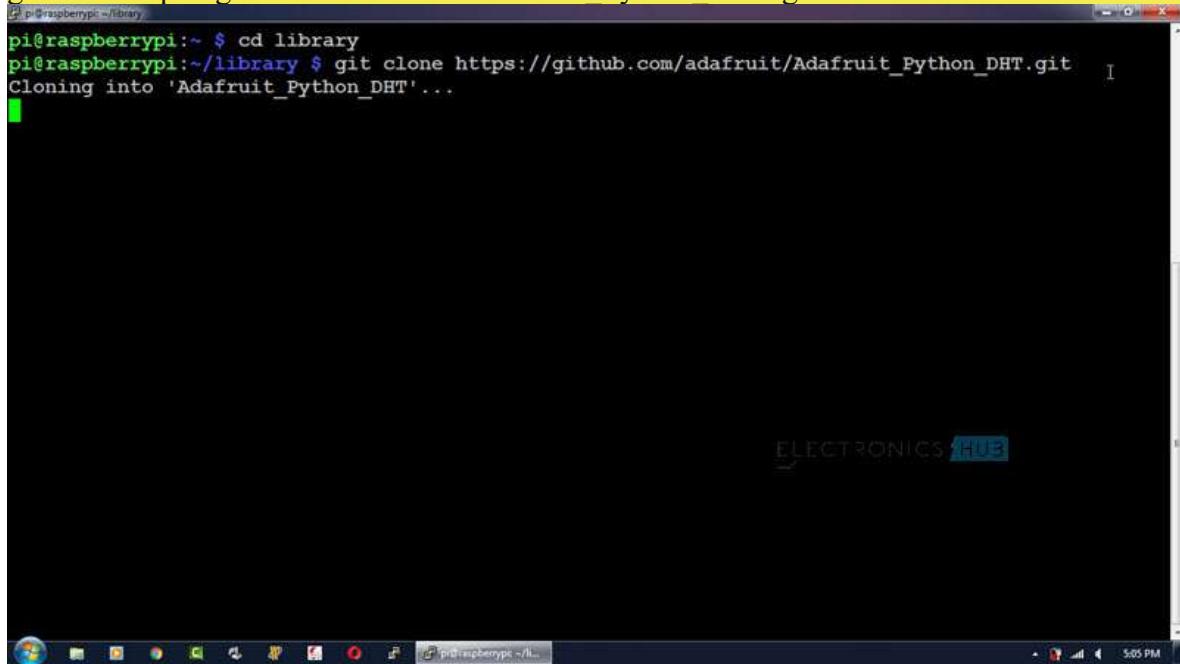


3	<p>Components Required</p> <ul style="list-style-type: none"> Raspberry Pi 3 Model B DHT11 Temperature and Humidity Sensor Connecting Wires Power Supply Computer
4	<p>Circuit Design</p> <p>If you observe the circuit diagram, there is not a lot of stuff going on with respect to the connections. All you need to do is to connect the VCC and GND pins of the DHT11 Sensor to +5V and GND of Raspberry Pi and then connect the Data OUT of the Sensor to the GPIO4 i.e. Physical Pin 7 of the Raspberry Pi.</p>
5	<p>Installing DTH11 Library</p> <p>Since we are using a library called Adafruit_DHT provided by Adafruit for this project, we need to first install this library into Raspberry Pi.</p> <p>First step is to download the library from GitHub. But before this, I have created a folder called 'library' on the desktop of the Raspberry Pi to place the downloaded files. You don't</p>

have to do that.

Now, enter the following command to download the files related to the Adafruit_DHT library.

```
git clone https://github.com/adafruit/Adafruit_Python_DHT.git
```



```
pi@raspberrypi:~ $ cd library
pi@raspberrypi:~/library $ git clone https://github.com/adafruit/Adafruit_Python_DHT.git
Cloning into 'Adafruit_Python_DHT'...
```

ELECTRONICS HUB

All the contents will be downloaded to a folder called ‘Adafruit_Python_DHT’. Open this directory using `cd Adafruit_Python_DHT`. To see the contents of this folder, use ‘ls’ command.

In that folder, there is file called ‘setup.py’. We need to install this file using the following command.

```
sudo python setup.py install
```

Code

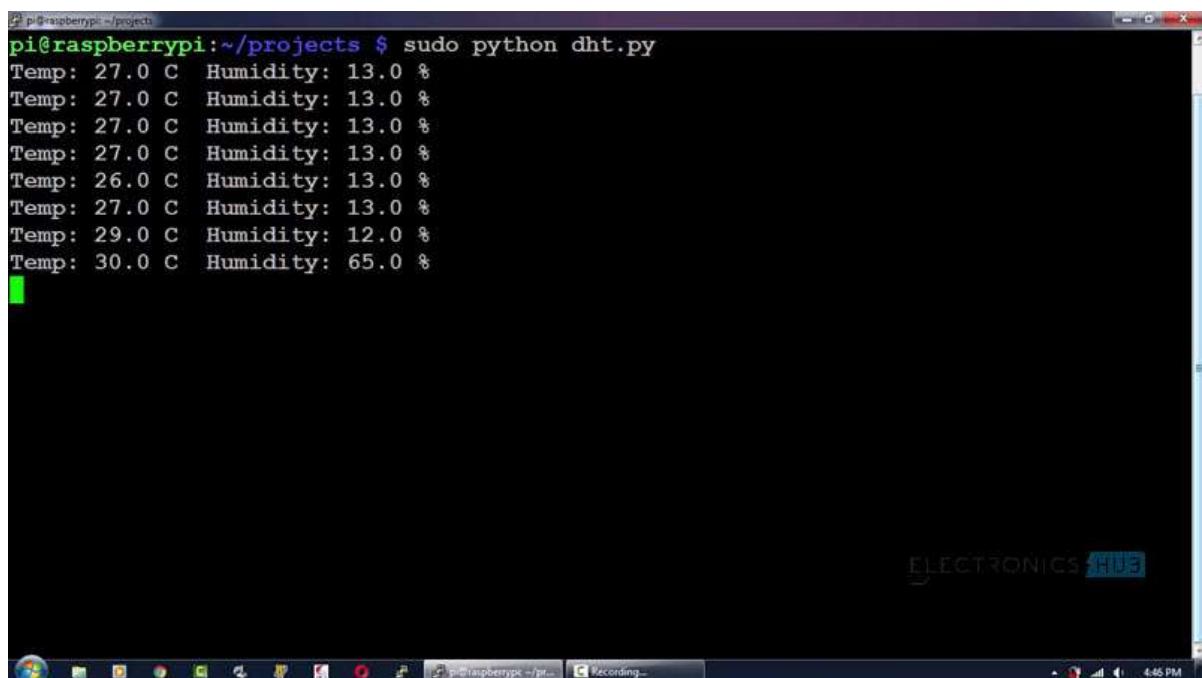
As we are using the library Adafruit_DHT for this project, there is nothing much to do in the Python Programming part. All you need to do is to invoke the library with the Sensor and GPIO Pin and print the values of Temperature and Humidity.

	<code>import sys</code>
	<code>import Adafruit_DHT</code>
	<code>import time</code>

	<code>while True:</code>
	<code>humidity, temperature = Adafruit_DHT.read_retry(22, 4)</code>
	<code>print 'Temp: {0:0.1f} C Humidity: {1:0.1f} %'.format(temperature, humidity)</code>
	<code>time.sleep(1)</code>

[view rawRaspberry Pi DHT 11.py](#) hosted with by [GitHub](#)
Working

Make the connections as per the circuit diagram and install the library. Use the above python program to see the results.



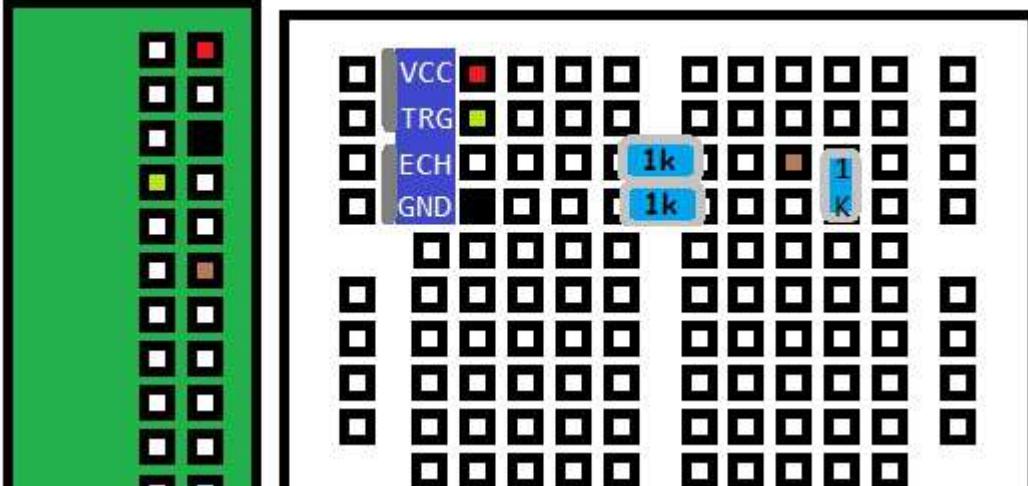
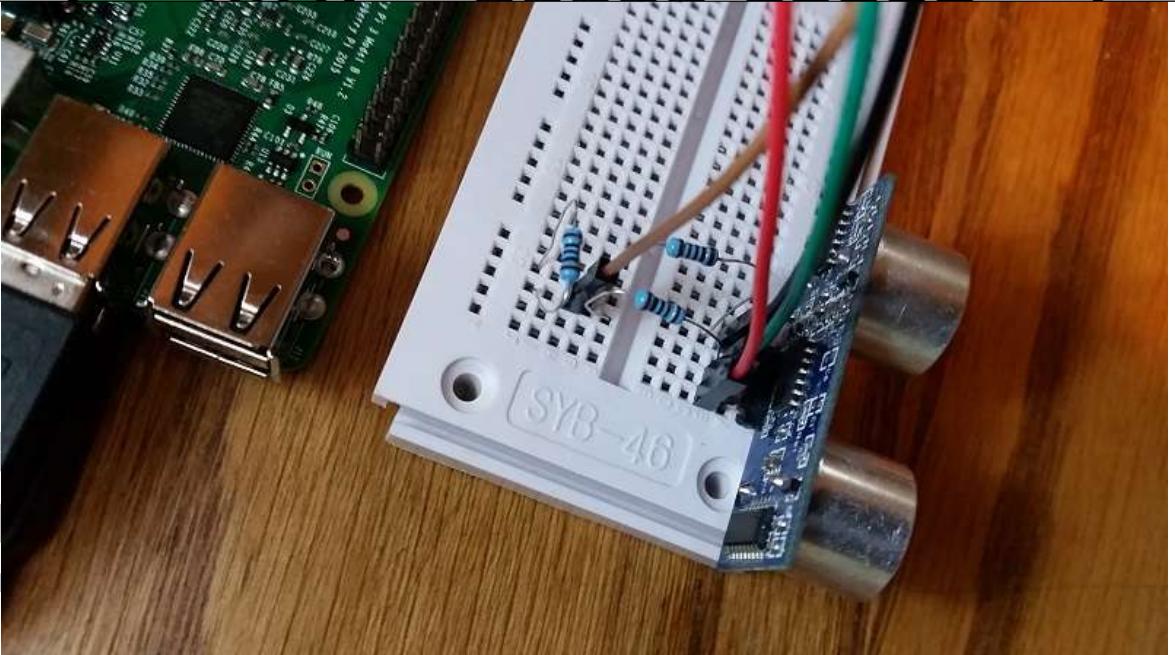
```
pi@raspberrypi:~/projects $ sudo python dht.py
Temp: 27.0 C  Humidity: 13.0 %
Temp: 26.0 C  Humidity: 13.0 %
Temp: 27.0 C  Humidity: 13.0 %
Temp: 29.0 C  Humidity: 12.0 %
Temp: 30.0 C  Humidity: 65.0 %
```

The screenshot shows a terminal window on a Raspberry Pi desktop environment. The terminal output displays the temperature and humidity readings from a DHT11 sensor, which fluctuate between 26.0°C and 30.0°C and 12.0% and 65.0% humidity. The desktop background features a blue and white abstract design. The taskbar at the bottom includes icons for the terminal, file manager, and system status. A watermark 'ELECTRONICS HUB' is visible in the bottom right corner of the desktop screen.

Experiment No: 10

Demonstration of Raspberry Pi with distance sensor (ultrasonic sensor HC-SR04)

Sl no.	Equipment's Needed
1	Raspberry Pi Model 3
2	Raspberry Pi ultrasonic sensor HC-SR04
3	VGA cable
4	Power Cable

Step 1:	Make the connections as follows 
	
Step 2:	Open the terminal CTRL+ALT+T and type \$ Sudo nano Filename.py to open empty notepad
Step 3:	Write the python program to implement use of raspberry pi with temperature sensor
	<pre>import RPi.GPIO as GPIO</pre>

```

import time

GPIO.setmode(GPIO.BCM)

TRIG = 4
ECHO = 18

GPIO.setup(TRIG,GPIO.OUT)
GPIO.setup(ECHO,GPIO.IN)

GPIO.output(TRIG, True)
time.sleep(0.00001)
GPIO.output(TRIG, False)

while GPIO.input(ECHO) == False:
    start = time.time()

while GPIO.input(ECHO) == True:
    end = time.time()

sig_time = end-start

#CM:
distance = sig_time / 0.000058

#inches:
#distance = sig_time / 0.000148

print('Distance: {} centimeters'.format(distance))

GPIO.cleanup()

```

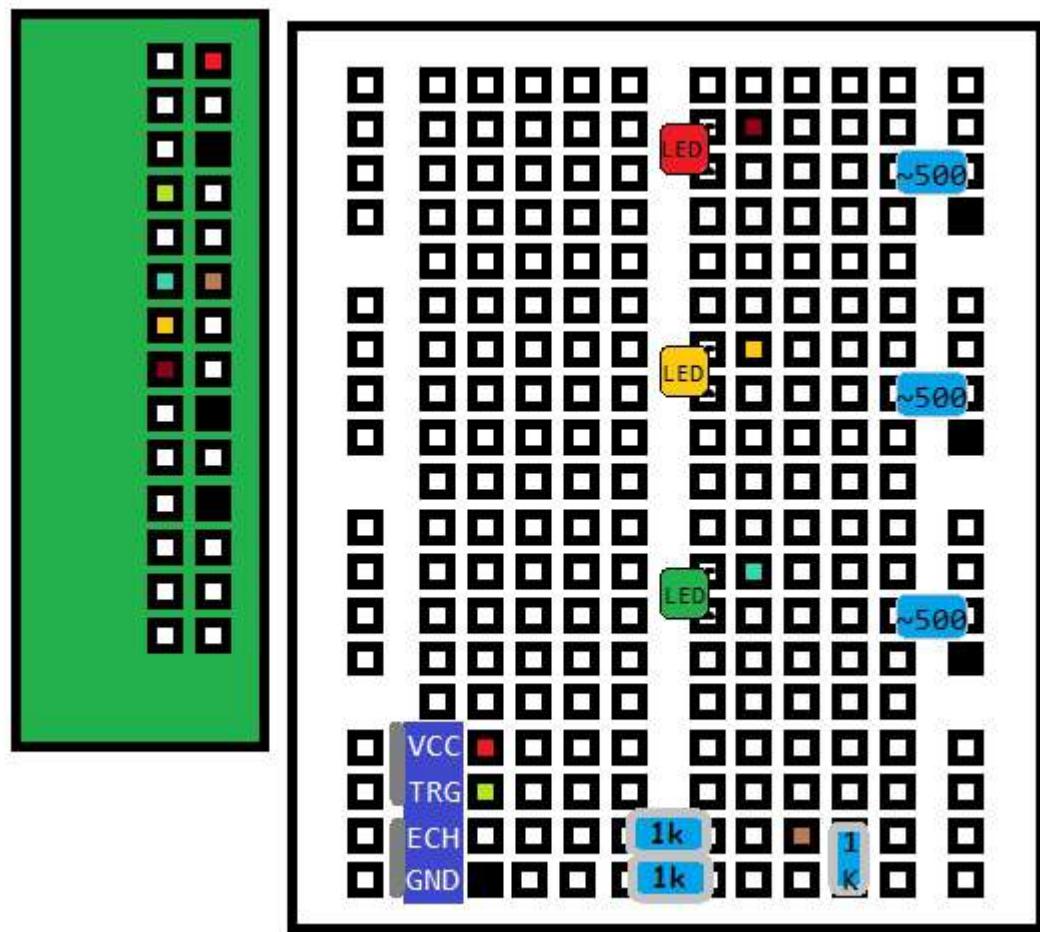
Step 3: Compile the program using command \$ Python Filename.py
And verify the output

Experiment No: 11

Raspberry pi program to implement Garage spot light.

Sl no.	Equipment's Needed
1	Raspberry Pi Model 3
2	Raspberry Pi ultrasonic sensor HC-SR04
3	VGA cable
4	Power Cable
5	Jumper cables- Male to Female
6	3 different colored LEDs-RGB

Step 1:	The idea of a garage stop-light is to show green when you have plenty of room to pull your car forward in your garage, and then turn yellow as you approach the fully forward position, and then red when you should stop. We're going to build this system with our Raspberry Pi, and use some distances that we can easily test.
	
Step 2:	To start, we'll just leave the distance sensor hooked up as is, but now we're going to add the light circuits, only this time we have three. Here's how I set mine up:

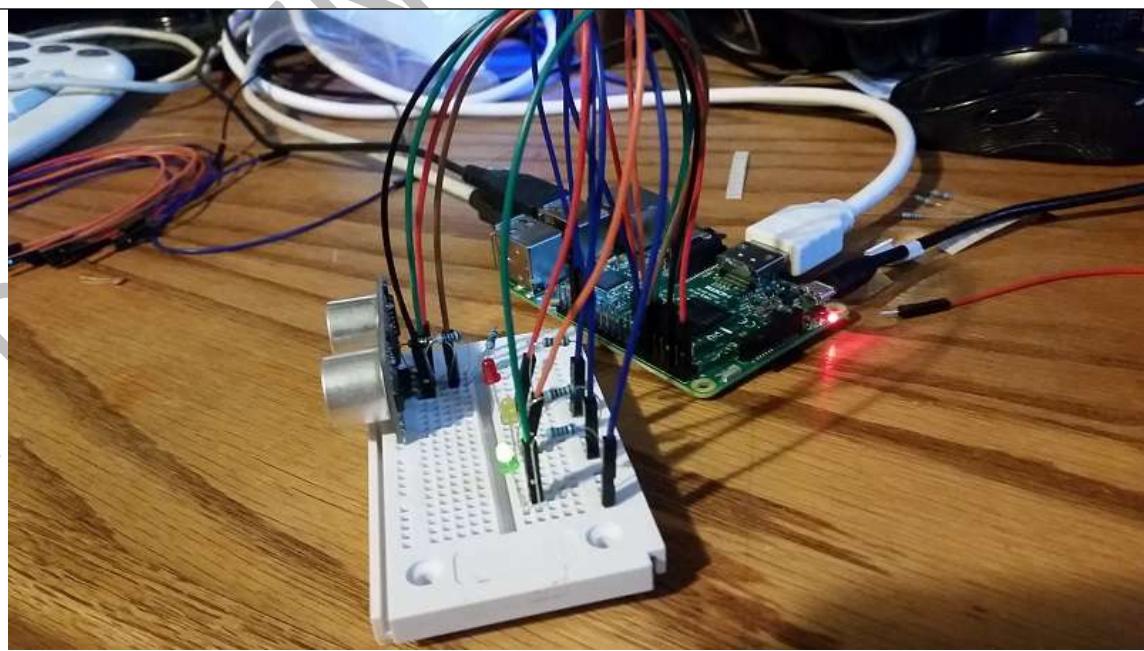


Step 3: Once you have everything hooked up and you're confident you've not messed anything up, you can power on your Raspberry Pi. You may want to re-reference the Adafruit image shared earlier:

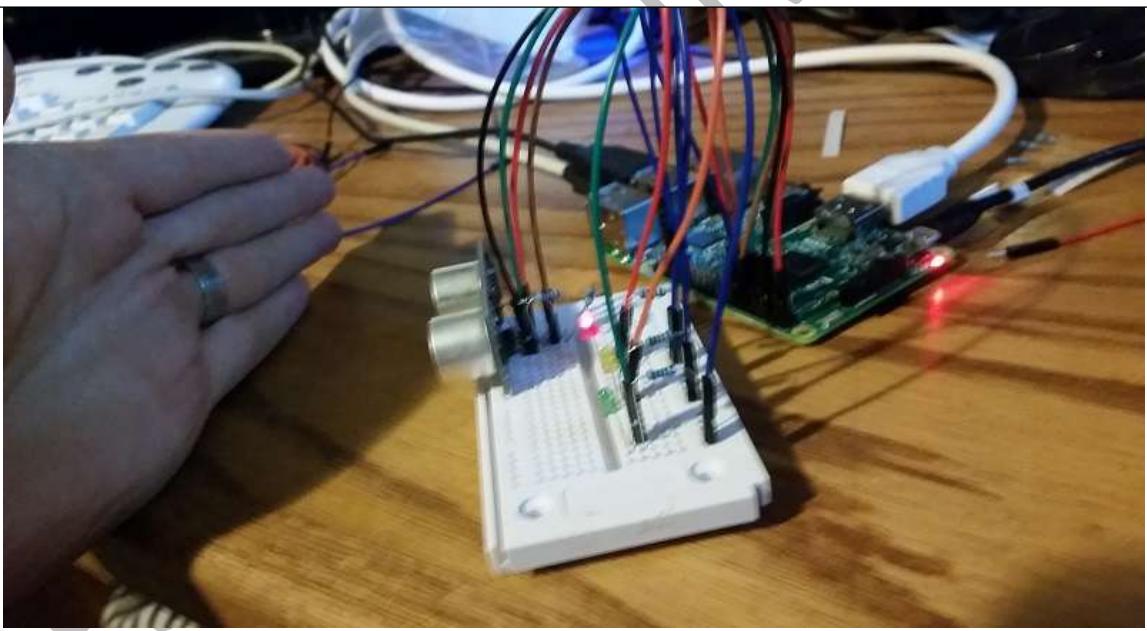
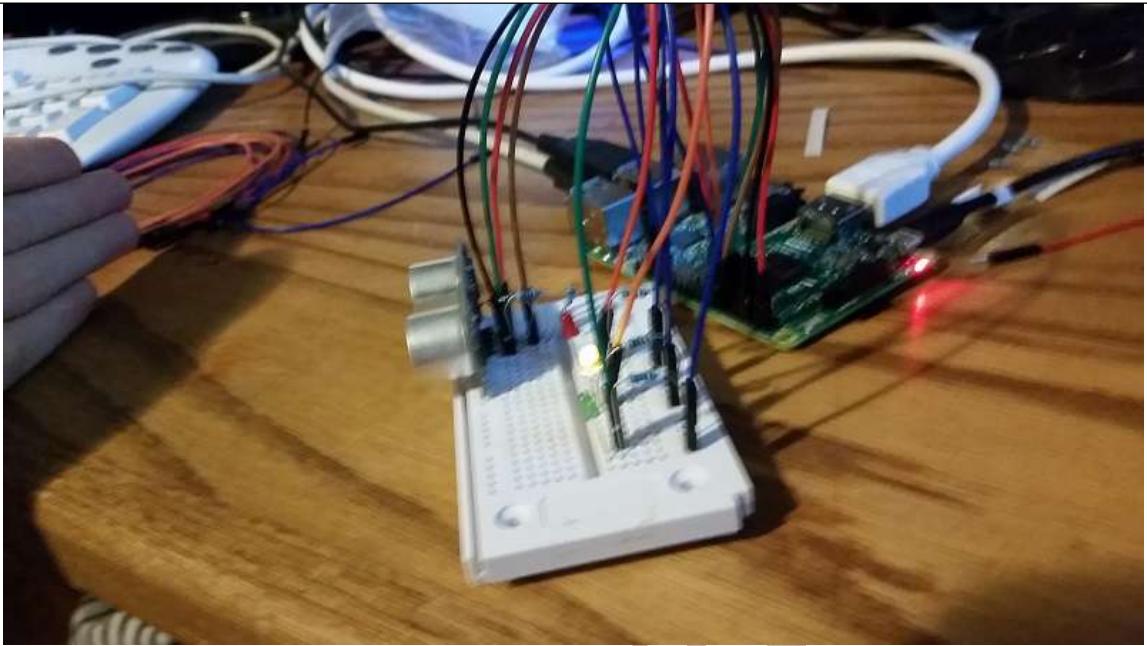
	<p>Available for GPIO if I2C is disabled using raspi-config</p> <table border="1"> <tbody> <tr><td>3.3V</td><td>GPIO2</td><td>5V</td></tr> <tr><td>GPIO3</td><td>GPIO4</td><td>GND</td></tr> <tr><td>GND</td><td>GPIO17</td><td>GPIO14</td></tr> <tr><td>GPIO17</td><td>GPIO27</td><td>GPIO15</td></tr> <tr><td>GPIO27</td><td>GPIO22</td><td>GND</td></tr> <tr><td>3.3V</td><td>GPIO10</td><td>GPIO18</td></tr> <tr><td>GPIO10</td><td>GPIO9</td><td>GPIO23</td></tr> <tr><td>GPIO9</td><td>GPIO11</td><td>GPIO24</td></tr> <tr><td>GND</td><td>DNC</td><td>GPIO25</td></tr> <tr><td>DNC</td><td>GPIO5</td><td>GPIO8</td></tr> <tr><td>GPIO5</td><td>GPIO6</td><td>GPIO7</td></tr> <tr><td>GPIO6</td><td>GPIO13</td><td>GND</td></tr> <tr><td>GPIO13</td><td>GPIO19</td><td>GPIO12</td></tr> <tr><td>GPIO19</td><td>GPIO26</td><td>GPIO16</td></tr> <tr><td>GPIO26</td><td>GND</td><td>GPIO20</td></tr> <tr><td>GND</td><td>DNC</td><td>GPIO21</td></tr> </tbody> </table> <p>Used by DotStars GPIO unavailable</p> <p>GPIO Availability: No Maybe Yes</p> <p>Available for GPIO if serial is disabled using raspi-config</p>	3.3V	GPIO2	5V	GPIO3	GPIO4	GND	GND	GPIO17	GPIO14	GPIO17	GPIO27	GPIO15	GPIO27	GPIO22	GND	3.3V	GPIO10	GPIO18	GPIO10	GPIO9	GPIO23	GPIO9	GPIO11	GPIO24	GND	DNC	GPIO25	DNC	GPIO5	GPIO8	GPIO5	GPIO6	GPIO7	GPIO6	GPIO13	GND	GPIO13	GPIO19	GPIO12	GPIO19	GPIO26	GPIO16	GPIO26	GND	GPIO20	GND	DNC	GPIO21	Pins above this line are present on all Raspberry Pi boards
3.3V	GPIO2	5V																																																
GPIO3	GPIO4	GND																																																
GND	GPIO17	GPIO14																																																
GPIO17	GPIO27	GPIO15																																																
GPIO27	GPIO22	GND																																																
3.3V	GPIO10	GPIO18																																																
GPIO10	GPIO9	GPIO23																																																
GPIO9	GPIO11	GPIO24																																																
GND	DNC	GPIO25																																																
DNC	GPIO5	GPIO8																																																
GPIO5	GPIO6	GPIO7																																																
GPIO6	GPIO13	GND																																																
GPIO13	GPIO19	GPIO12																																																
GPIO19	GPIO26	GPIO16																																																
GPIO26	GND	GPIO20																																																
GND	DNC	GPIO21																																																

Step 4:	Garage spot light script
---------	--------------------------

	Output
--	--------



Hand getting close:



"Hello World!"

The LiquidCrystal library allows you to control LCD displays that are compatible with the Hitachi HD44780 driver. There are many of them out there, and you can usually tell them by the 16-pin interface.

This example sketch prints "Hello World!" to the LCD and shows the time in seconds since the Arduino was reset.



output of the sketch on a 16x2 LCD

The LCDs have a parallel interface, meaning that the microcontroller has to manipulate several interface pins at once to control the display. The interface consists of the following pins:

A **register select (RS) pin** that controls where in the LCD's memory you're writing data to. You can select either the data register, which holds what goes on the screen, or an instruction register, which is where the LCD's controller looks for instructions on what to do next.

A **Read/Write (R/W) pin** that selects reading mode or writing mode

An **Enable pin** that enables writing to the registers

8 data pins (D0 -D7). The states of these pins (high or low) are the bits that you're writing to a register when you write, or the values you're reading when you read.

There's also a **display contrast pin (Vo)**, **power supply pins (+5V and Gnd)** and **LED Backlight (Bklt+ and BKlt-)** pins that you can use to power the LCD, control the display contrast, and turn on and off the LED backlight, respectively.

The process of controlling the display involves putting the data that form the image of what you want to display into the data registers, then putting instructions in the instruction register. The [LiquidCrystal Library](#) simplifies this for you so you don't need to know the low-level instructions.

The Hitachi-compatible LCDs can be controlled in two modes: 4-bit or 8-bit. The 4-bit mode requires seven I/O pins from the Arduino, while the 8-bit mode requires 11 pins. For displaying text on the screen, you can do most everything in 4-bit mode, so example shows how to control a 16x2 LCD in 4-bit mode.

Hardware Required

- Arduino Board
- LCD Screen (compatible with Hitachi HD44780 driver)
- pin headers to solder to the LCD display pins
- 10k ohm potentiometer
- 220 ohm resistor
- hook-up wires
- breadboard

Circuit

Before wiring the LCD screen to your Arduino board we suggest to solder a pin header strip to the 14 (or 16) pin count connector of the LCD screen, as you can see in the image above. To wire your LCD screen to your board, connect the following pins:

- LCD RS pin to digital pin 12
- LCD Enable pin to digital pin 11
- LCD D4 pin to digital pin 5
- LCD D5 pin to digital pin 4

- LCD D6 pin to digital pin 3
- LCD D7 pin to digital pin 2
- LCD R/W pin to GND
- LCD VSS pin to GND
- LCD VCC pin to 5V
- LCD LED+ to 5V through a 220 ohm resistor
- LCD LED- to GND

Additionally, wire a 10k pot to +5V and GND, with it's wiper (output) to LCD screens VO pin (pin3).

click the images to enlarge

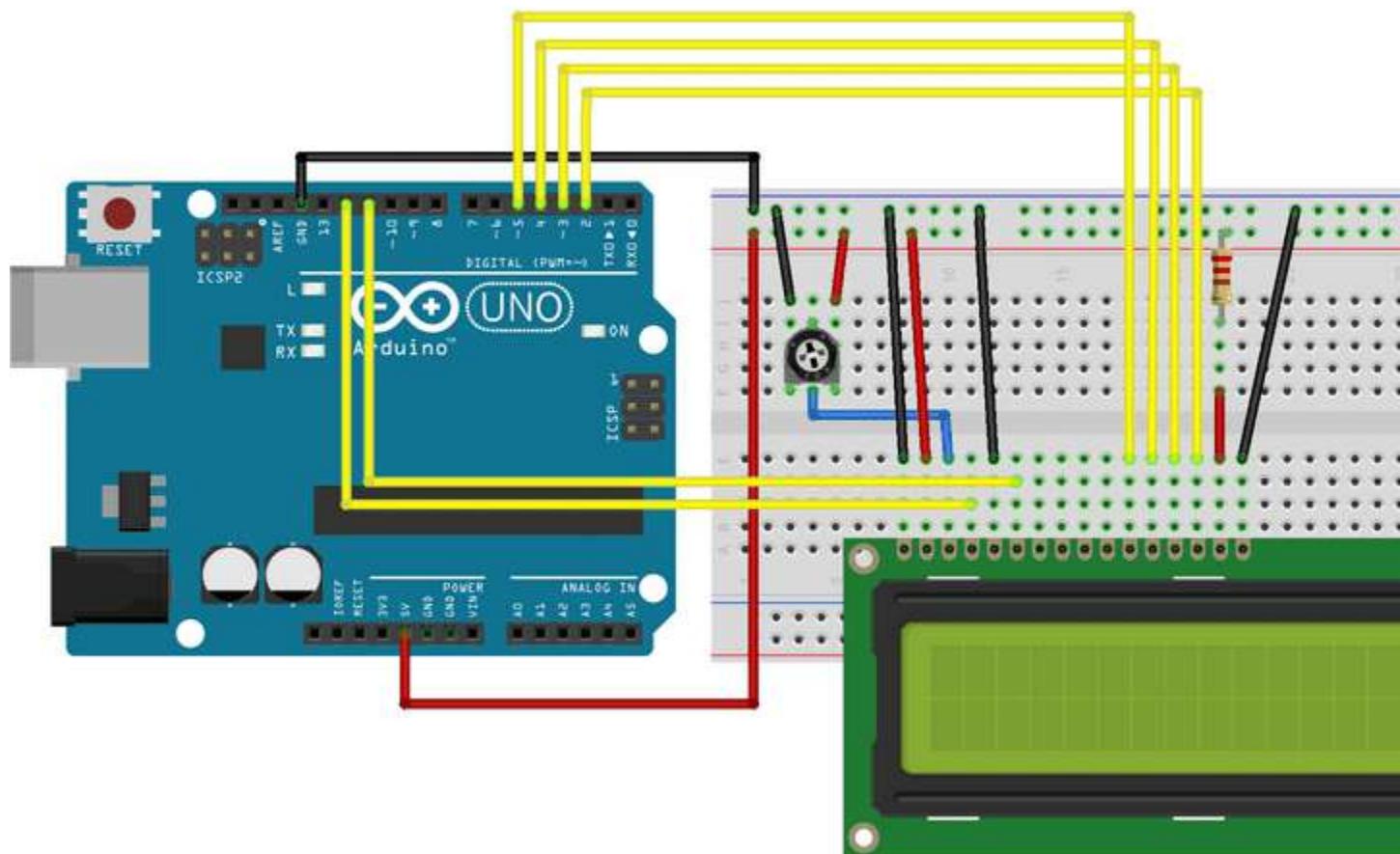


image developed using [Fritzing](#). For more circuit examples, see the [Fritzing project page](#)

Schematic

click the images to enlarge

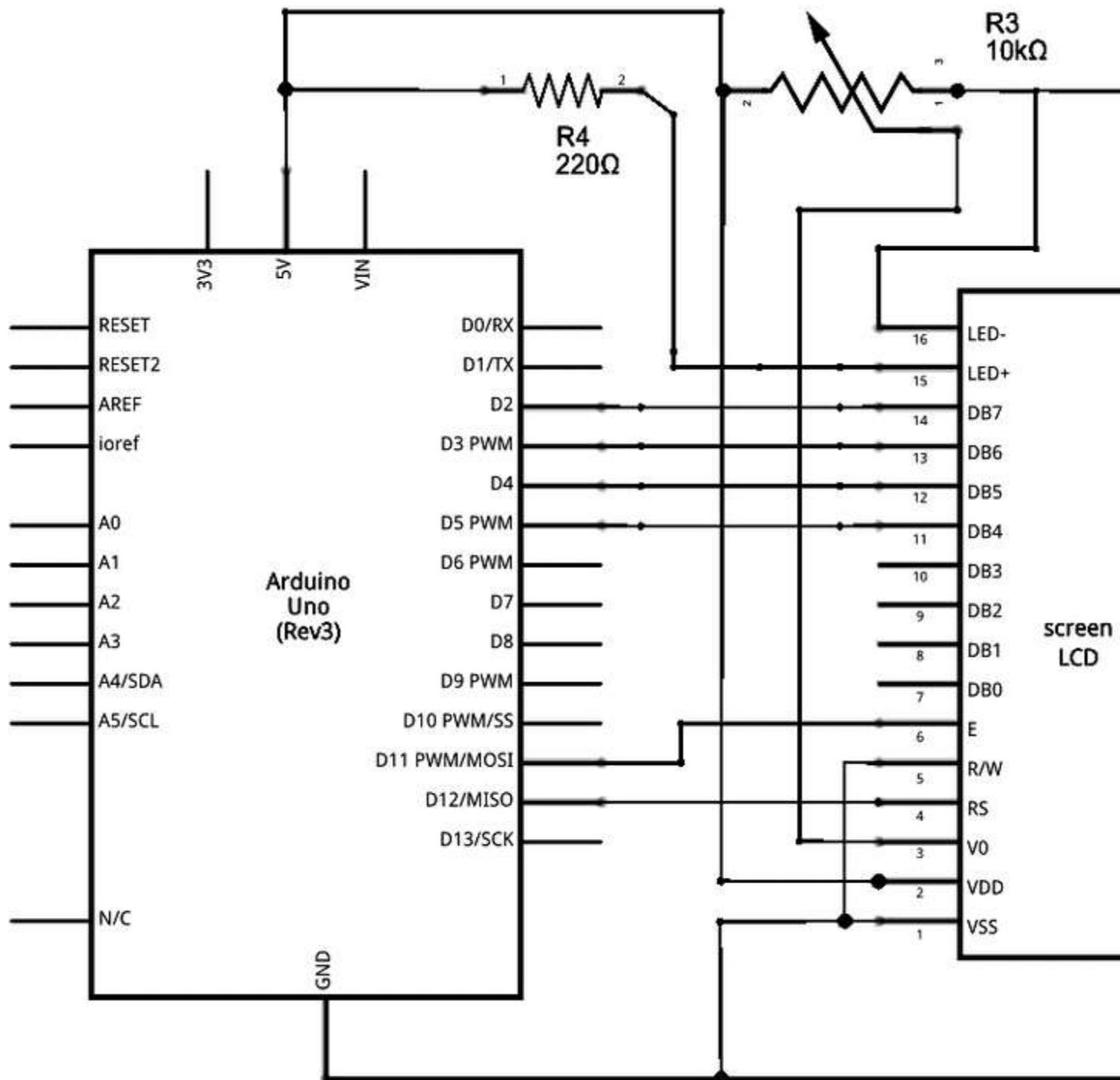


image developed using [Fritzing](#). For more circuit examples, see the [Fritzing project page](#)

Code

See Also

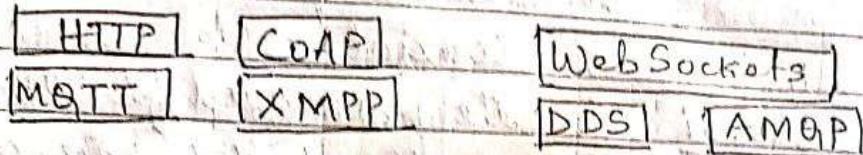
- [Liquid Crystal Library](#) - Your reference for the Liquid Crystal library.
- [lcd.begin\(\)](#)
- [lcd.print\(\)](#)
- [lcd.setCursor\(\)](#)
- [Blink](#) - Control of the block-style cursor.
- [Cursor](#) - Control of the underscore-style cursor.
- [Display](#) - Quickly blank the display without losing what's on it.
- [TextDirection](#) - Control which way text flows from the cursor.
- [Scroll](#) - Scroll text left and right.
- [Serial display](#) - Accepts serial input, displays it.
- [SetCursor](#) - Set the cursor position.
- [Autoscroll](#) - Shift text right and left.

IOT Protocols :

Date

Page

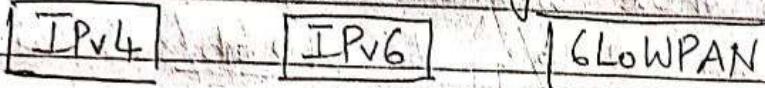
Application Layer



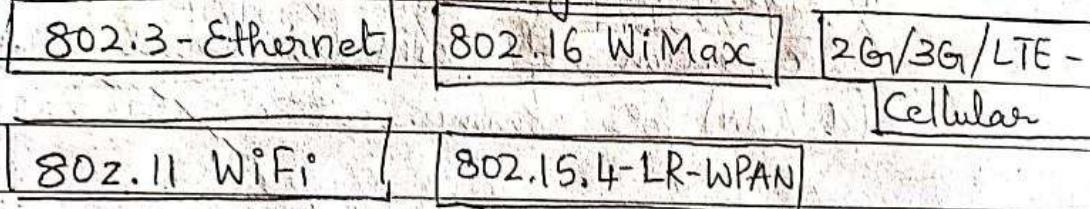
Transport Layer



Network Layer



Link Layer



Link Layer : determines how the data is sent over the network's physical layer or medium (eg: copper wire, coaxial cable, radio wave).

→ The scope of the Link Layer is the local network connection to which the host is connected/attached.

→ Link layer determines how the packets are coded and signaled by the hardware device over the medium to which the host is connected/attached.

WiMax: (Worldwide Interoperability for Microwave Access)

Some of the link layer protocols in To

i) Ethernet 802.3

is the standard for 10BASE5 Ethernet that uses coaxial cable.

802.3i is the standard for 10BASE-T Ethernet over copper twisted-pair connection.

802.3j is the standard for 10BASE-F for Ethernet over fibre optic connections.

802.3ae is the standard for 10G bits/s Ethernet over fibre.

These standards provide data rates from 10 Mb/s to 10 Gb/s or higher.

ii) 802.11 WiFi : IEEE 802.11 is a collection of wireless local area network (WLAN) communication standards.

Eg: 802.11a operates in the 5 GHz band, 802.11b and 802.11g operate in 2.4 GHz band. 802.11n operates in 2.4/5 GHz band. These standards provide data rates from 1 Mb/s to 6.75 Gb/s.

iii) 802.16 WiMax (Worldwide Interoperability for Microwave Access), is a collection of wireless broadband standards. WiMax provides data rates from 1.5 Mb/s to 1 Gb/s.

→ The recent update (802.16m) provides data rates of 100 Mb/s for mobile stations and 1 Gb/s for fixed stations.

iv) 802.15.4 - LR-WPAN (Low-rate wireless personal area networks (LR-WPANS)). These standards form the bases of specifications for high level communication protocol such as Zigbee. IEEE WPAN provides data rates from 40 kb/s to 230 kb/s.

These standards provide low-cost and low speed communication for power constrained devices.

v) 2G/3G/4G - mobile communications:

2G (2G including GSM and CDMA)

Network - Layer

IP Subnetting

The subnet mask for a particular IP address is actually used by the router to resolve which part of IP address is providing the network address and which part of the address is providing host address.

Subnet mask also consists of four octets of information.

The router matches up the information in the subnet mask with the actual IP address and determines the network address and the node address.

Class Subnet Mask

A 255.0.0.0

B 255.255.0.0

C 255.255.255.0

Subnet Mask:

How does a router use the subnet mask to determine which part of IP address refers to the network address.

It uses the process of anding

128 64 32 16 8 4 2 1

Date _____

Page _____

Subnet-Mask: 255.255.0.0

IP Address: 180.20.5.9

Network Address:

1111111 1111111 00000000 0.00000000

10110100 00010000 00000101 00001001

10110100 00010100 00000000 00000000

Network address: 180.20

Subnetting provides the capability to break a large network into subnets that are connected with routers.

Consider the network address 10.0.0.0 and need of 30 subnets.

In class A (255.0.0.0), the first octet defines the network address, 24 bits define host address.

Network address: 10.0.0.0

Number of subnets: 30

00001010 00000000 00000000 00000000

00000000 00000000 11111111 11111111

Decimal values of octet 00111111.

255 254 128 64 32 16 8 4 2 1

$(2^7 - 2)$ 5 bits are required to create 30 subnets
 $= 16,777,214$ $1 + 2 + 4 + 8 + 16 - 1 = 30$

1's for broadcasting Add 5 high-order bits to create the subnet masks

$$128 + 64 + 32 + 16 + 8 = 248$$

New Subnet Mask 255.248.0.0

Calculating IP subnet ranges:

Class A network has been subnetted.

so the new subnet mask is 255.248.0.0

This subnet mask tells routers and other devices that the class A network contains 30 subnets.

5 bits higher order

Date _____

Page _____

The higher-order decimal values that you used for the subnet mask:

128, 64, 32, 16 and 8.

Take the lowest of the higher-order bits that you used to calculate the new subnet mask, i.e. 8. This number becomes the increment used to create the IP address range.

10.8.0.1

Subnet 1	Start Address	End Address
1	10.8.0.1	10.15.255.254
2	10.16.0.1	10.23.255.254
3	10.24.0.1	10.31.255.254
4	10.32.0.1	10.39.255.254
5	10.40.0.1	10.47.255.254
6	10.48.0.1	10.55.255.254
7	10.56.0.1	10.63.255.254
8	10.64.0.1	10.71.255.254
9	10.72.0.1	10.79.255.254
10	10.80.0.1	10.87.255.254

128 64 32 16 8 4 2 1
1 0 1 0 1 0 1 0 0 0 0

Class B example:

Network address: 180.10.0.0

Number of subnets: 6

Binary equivalent:

10110100 (180)

10110100 00001010 00000000 000000

Decimal values of octet:

128 64 32 16 8 4 2 1

3 bits are required to create 6 subnets.

$$1+2+4=6 \text{ Subnets}$$

Add three higher order bits to create the subnet mask

$$128+64+32=224$$

New Subnet Mask 255.255.224

Subnet# Start address

1	180.10.32.1	180.10.63.254
2	180.10.64.1	180.10.95.254
3	180.10.96.1	180.10.127.254
4	180.10.128.1	180.10.159.254
5	180.10.160.1	180.10.191.254
6	180.10.192.1	180.10.223.254

Class C example

Network Address: 200.10.44.0

Number of subnets: 2

11001000 00001010 00101000 00000000

First Node
Octet

Decimal values of octets

(128) 64 32 16 8 4 2 1 (111)

2 bits are required to create 2 subnets

$$1+2 = 1+2 \text{ subnets}$$

Add 2 higher-order bits to create subnet mask

$$128+64 = 192$$

The new subnet mask is 255.255.255.192

Subnet 1 start address end address

1 200.10.44.65 200.10.44.126

64 2 200.10.44.127 200.10.44.190

127

6LoWPAN: (IPV6 over Low power Wireless Personal Area Networks) brings IP protocol to the low-power devices which have limited processing capability.

→ 6LoWPAN: operates in the 2.4GHz frequency range and provides data transfer rates of 250 Kbps.

→ 6LoWPAN works with 802.15.4 link layer protocol & provides header compression for IPv6 datagrams.

IOT protocols... Transport Layer

- Provide end-to-end message transfer capability independent of the underlying network.
- It provides functions such as error control, segmentation, flow control & congestion control.

→ TCP UDP

TCP is used by web browsers, email pgm (SMTP) and file transfer (FTP). TCP/IP is a connection and stateful. While IP protocol deals with sending packets, TCP ensures reliable transmission of packets in-order.

- TCP provides error detection capability so that duplicate^{packets} can be discarded and lost^{packets} are retransmitted.
- The flow control capability of TCP ensures that rate at which the sender sends the data is not too high for the receiver to process.

UDP: is a connectionless protocol. UDP is useful for time-sensitive applications that have very small data units to exchange and do not want the overhead of connection setup. It is a stateless prot.

- UDP does not provide guaranteed delivery.

ordering of messages and duplicate elimination

NOTE:

→ publish/subscribe

Application Layer:

Application layer protocol defines how the Application interface (with the lower layer protocols to send the data over the n/w.

Application data i.e files, is encoded by the application layer protocol and encapsulated in the transport layer protocol which provided connection or connectionless communication over the n/w.

Eg:- Ports no.s are used for application addressing (80 for HTTP, 22 for SSH, etc)

i) HTTP (HyperText Transfer Protocol) is the application layer protocol that forms the foundation of WWW.

It includes commands such as GET, PUT, POST, DELETE, HEAD, TRACE, OPTIONS etc.

The protocol follows a request-response model.

HTTP is a stateless protocol and each HTTP request is independent of other requests.

→ HTTP client can be a browser or the applets running on IoT device or SW. HTTP uses URI Uniform Resource Identifiers to identify HTTP resources.

- ii) COAP: Constrained Application Protocol (CoAP)
is ALP for machine-to-machine (M2M) applications, meant for constrained environments with constrained devices & constrained networks.
- COAP is a web transfer protocol and uses a request-response model, however it runs on top of UDP instead of TCP.
 - COAP uses a client-server architecture where clients communicate with servers using connectionless datagrams.

- iii) WebSockets: Websockets allows a full duplex communication over a single socket connection for sending messages between client and servers.
- Websocket is based on TCP and allows streams of messages to be sent back and forth b/w the client and server while keeping the TCP connection open.
- The client can be browser, an IoT device or a mobile application.

- iii) MQTT (Message Queue Telemetry Transport) is a light weight messaging protocol based on the publish-subscribe model. MQTT uses client-server architecture where the client (an IoT device) connects to the server (MQTT Broker) and publish messages to topics on the server. The broker forwards the messages to the clients.

- Subscribed to topics.
- MQTT is well suited for constrained environments where the devices have limited processing and memory resources and the n/w bandwidth is low.
- i) XMPP (Extensible Messaging and Presence Protocol)
- It is a protocol for real time communication and streaming XML data b/w n/w entities.
 - XMPP powers wide range of applications including messaging, presence, data synchronizations, gaming, multi-party chat and voice/video calls.

XMPP allows sending small chunks of XML data from one n/w entity to another in real-time.

XMPP is a decentralized protocol and uses a client-server architecture. It supports both client to server & service-to-service ~~multicast~~ communication paths.

XMPP allows real-time communication b/w IoT device

- ii) DDS: Data Distribution Service is a data centric middleware standard for device-to-device or machine-to-machine communication. DDS uses a publish-subscribe model where publishers (devices that generate data) create topics, to which subscribers can subscribe.

DPS provides QoS control & configurable reliability.

AMQP: Advanced Message Queuing Protocol is an open application Layer protocol for business messaging. AMQP supports both point-to-point and publisher/subscriber routing and queuing.

AMQP supports both point brokers receive messages from publishers (IoT devices & applications that generate data) and route them over connections to consumers.

Publisher publish the messages to exchanges which then distribute message copies to queues. Messages are either delivered by the broker to the consumers who have subscribed to queues or the consumers can pull the messages from the queues.

Logic Design of IoT

It refers to an abstract representation of entities and processes without going into the low-level specifics of the implementation.

IoT Functional Blocks

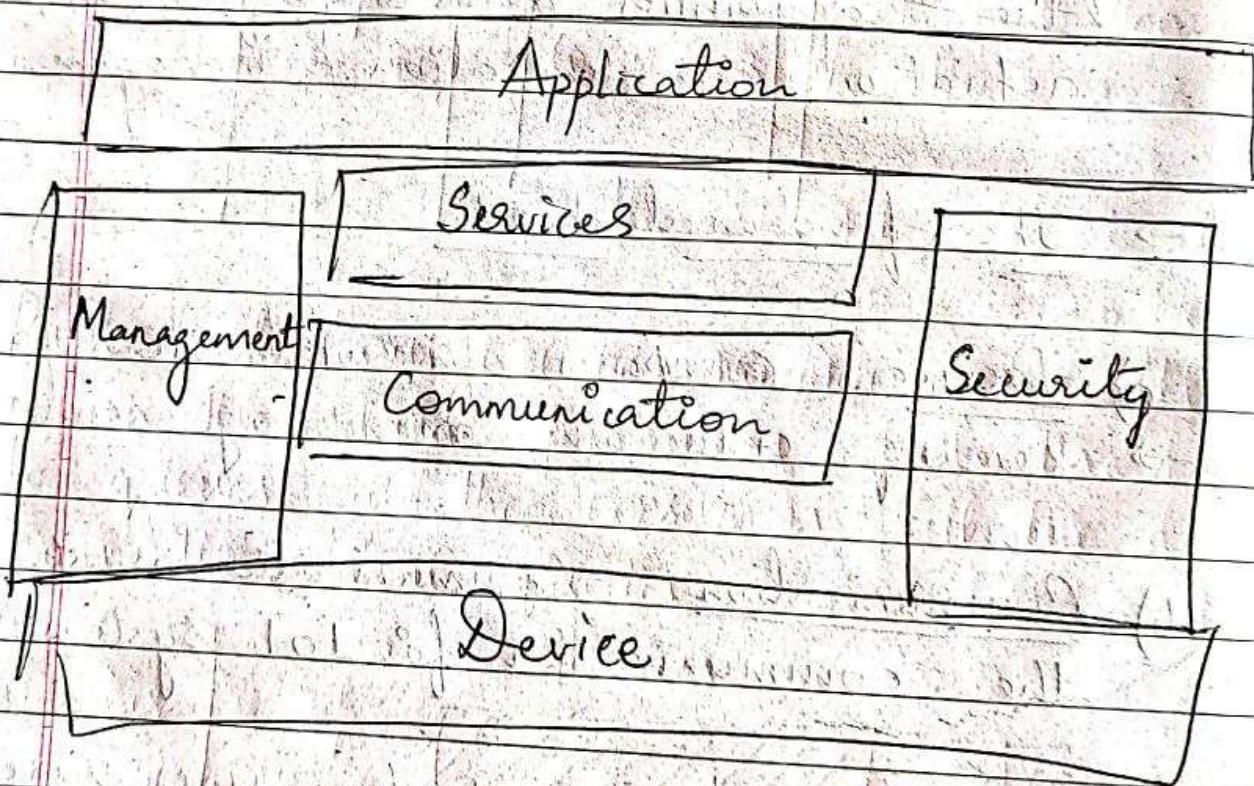
IoT system comprises of a number of functional blocks that provide the system the capabilities of identification, sensing, actuation, communication & management.

→ The functional blocks

- a) Device: consists of devices that provides sensing, actuation, monitoring & control functions.
- b) Communication: Communication block handles the communication for IoT system.
- c) Services: An IoT system uses various IoT services such as services for device monitoring, device control services, data publishing services and services for device discovery.
- d) Management: To govern IoT system.
- e) Security: IoT System of Security functional

block ~~enters~~ secures the IoT system and by providing functions such as authentication, authorization, message and content integrity & data security.

Application: provide an interface that the users can use to control and monitor various aspects of IoT system. Application also allows users to view the system status and view or analyze the processed data.

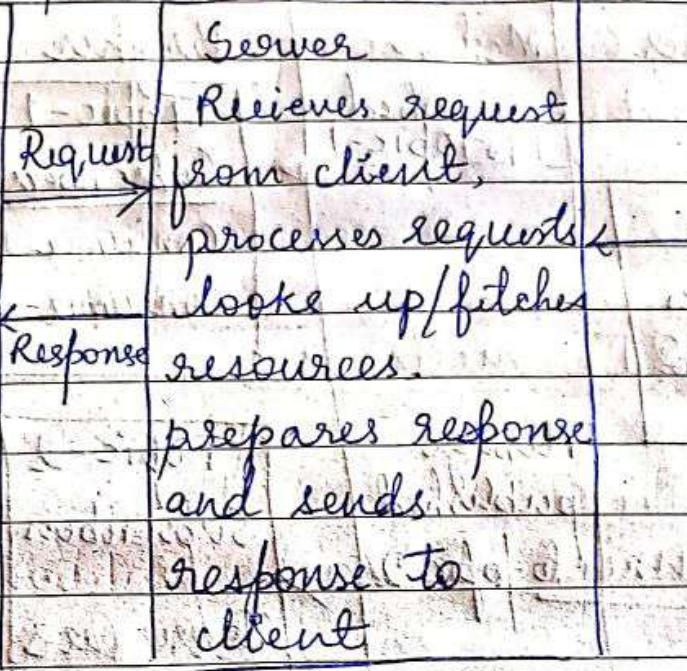


→ IoT Communication Model (ii)

1) Request - Response Model.

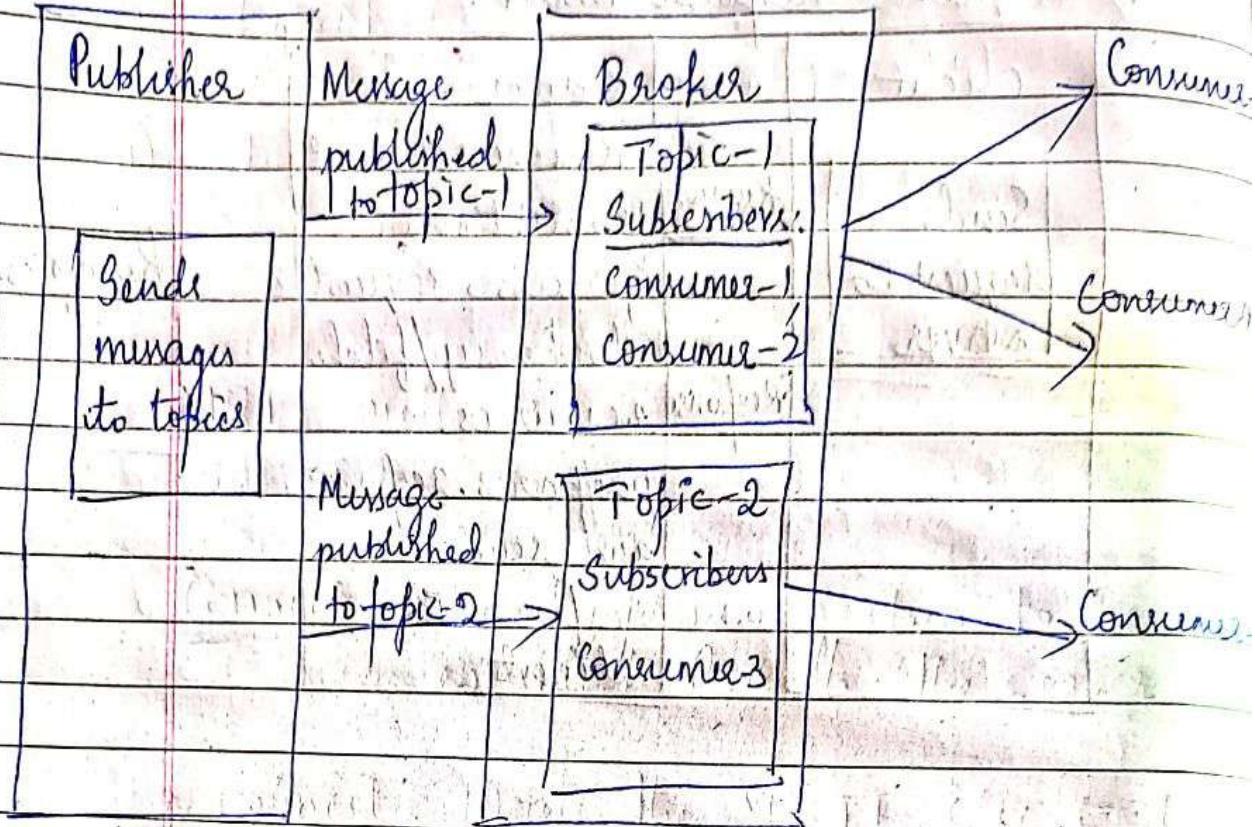
→ Client

Sends
request to
server

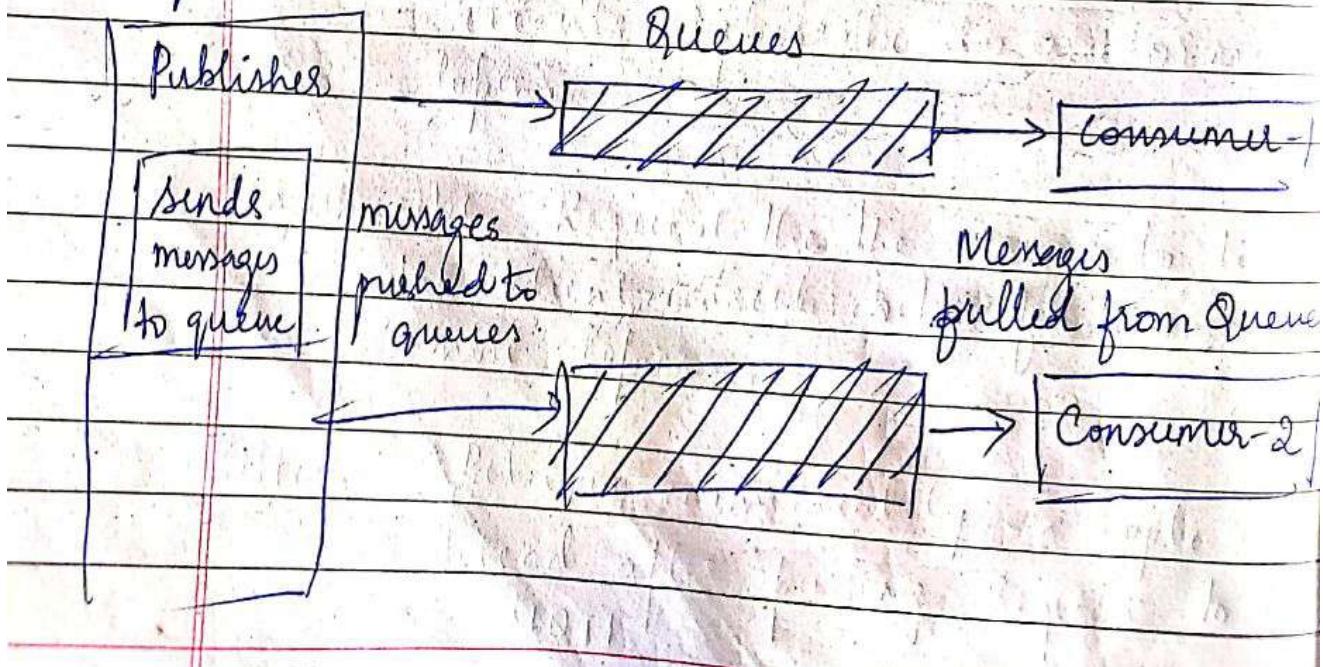


Resources

ii) Publisher + Subscribe model:



Push-Pull: is a communication model in which the data producers push the data to queues and the consumers pull the data from the ~~client~~ queues.



Exclusive Pair is a bidirectional, full duplex communication model that uses persistent connection b/w the client and server.

