Government Polytechnic Jalgaon

Academic Year 2020-21

Online Industrial Training

EJ 5 I

Name of student -: Prathamesh ganesh saraf
Training subject -: internet of things (IOT)
Training provider-:enovote skils ,(ntttr chd. Start up).
Duration -: 4 to 6 weeks
Mode of training -: online (at zoom)

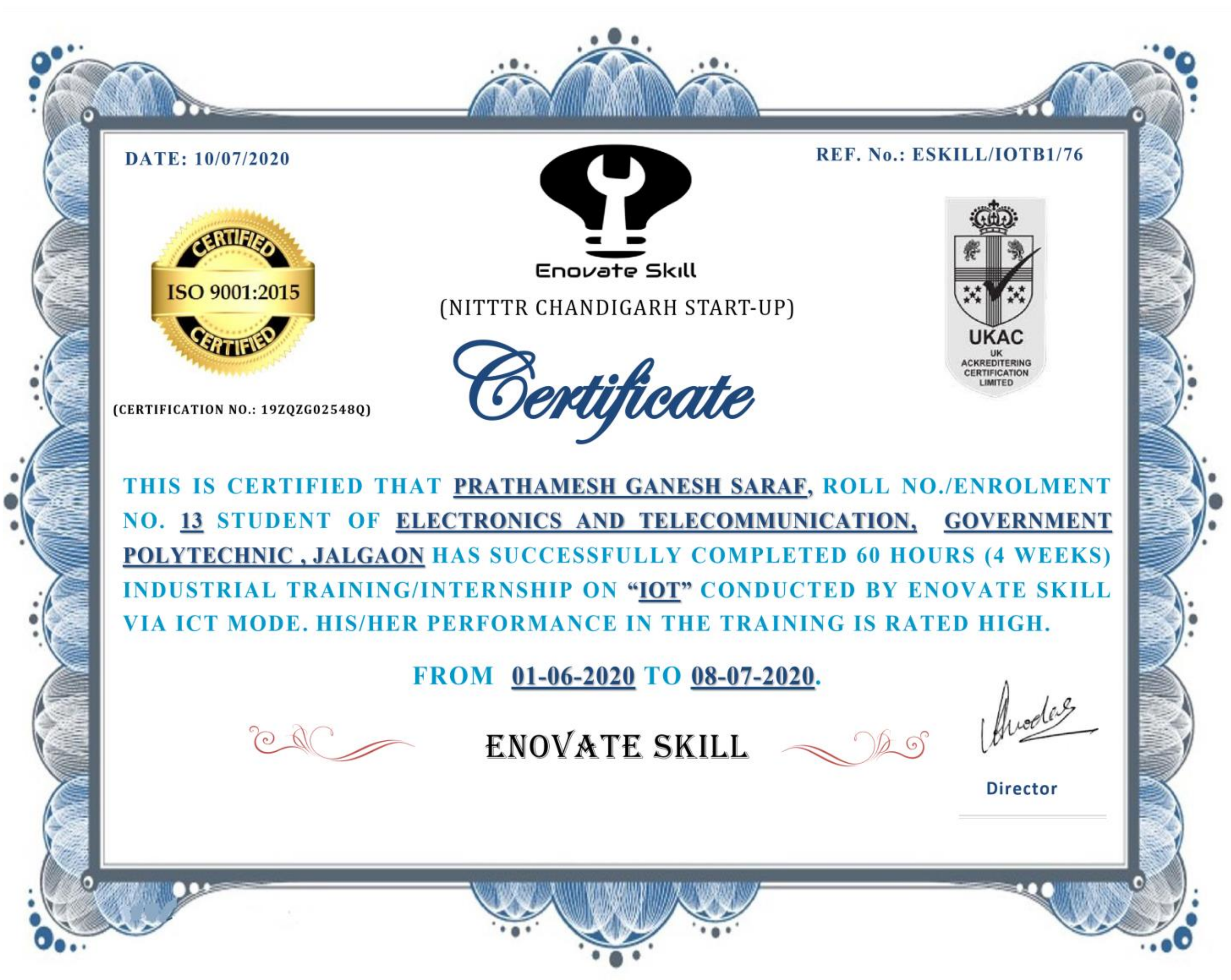
Index

Acknowledgement

Firstly, I would like to express my appreciation to my Head Of departmental Prof. K.P.Akole. His constant guidance and advice played a vital role in making the execution of the report. He always gave me his suggestions that were crucial in making this report as possible. .

Successfully completion of any type of project requires helps from a number of persons. I have also taken help from different people for the preparation of this report. Now, there is a little effort to show my deep gratitude to that helpful person. .

I convey my sincere gratitude to my mentor and also Head of Department Prof. K.P.Akole Without his kind direction and proper guidance, this study would have been a little success. In every phase of the training, his supervision and guidance shaped this report to be completed perfectly. I would also like to appreciate the Prof Ajay Godara founder and CEO of Inovate chindargh pvt lmt who conduct this 4 week training.



DATE: 10/07/2020

REF. No.: ESKILL/IOTB1/76

CERTIFIED ISO 9001:2015 CERTIFIED

(CERTIFICATION NO.: 19ZQZG02548Q)

Enovate Skill

(NITTTR CHANDIGARH START-UP)

UKAC UK ACKREDITERING CERTIFICATION LIMITED

Certificate

THIS IS CERTIFIED THAT **PRATHAMESH GANESH SARAF**, ROLL NO./ENROLMENT NO. **13** STUDENT OF **ELECTRONICS AND TELECOMMUNICATION**, **GOVERNMENT POLYTECHNIC , JALGAON** HAS SUCCESSFULLY COMPLETED 60 HOURS (4 WEEKS) INDUSTRIAL TRAINING/INTERNSHIP ON "**IOT**" CONDUCTED BY ENOVATE SKILL VIA ICT MODE. HIS/HER PERFORMANCE IN THE TRAINING IS RATED HIGH.

FROM **01-06-2020** TO **08-07-2020**.

ENOVATE SKILL

[illegible]

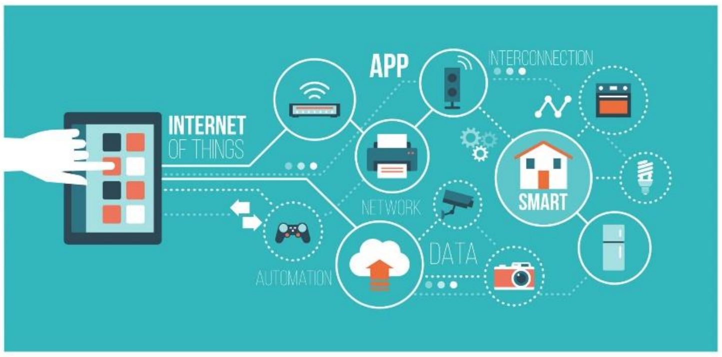
Director

Week 1
(1/6/2020 To 6/6/2020)

Topics of the week -:

1) Introduction to IOT
2) Application of IOT
3) Introduction of electronics components.
4) Introduction Iot Based Tools .
5) Introduction to Tinkercad .
6) Motor driver circuit using transistor as a switch
7) Analysis of voltage divider circuit

Introduction of IOT



The Internet of things (IoT) is a system of interrelated computing devices, mechanical and digital machines provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.
The definition of the Internet of things has evolved due to the convergence of multiple technologies, real-time analytics, machine learning, commodity sensors, and embedded systems. Traditional fields of embedded systems , wireless sensor networks , automation (including home and building automation), and others all contribute to enabling the Internet of things. In the consumer market, IoT technology is most synonymous with products pertaining to the concept of the "smart home", including devices and appliances (such as lighting fixtures, thermostats, home security systems and cameras, and other home appliances) that support one or more common ecosystems, and can be controlled via devices associated with that ecosystem, such as smartphones and smart speakers.

History of IOT

The main concept of a network of smart devices was discussed as early as 1982, with a modified Coca-Cola vending machine at Carnegie Mellon University becoming the first Internet-connected appliance, able to report its inventory and whether newly loaded drinks were cold or not. Mark Weiser's 1991 paper on ubiquitous computing,
The term "Internet of things" was likely coined by Kevin Ashton of Procter & Gamble, later MIT's Auto-ID Center, in 1999, though he prefers the phrase "Internet for things". At that point, he viewed radio-frequency identification (RFID) as essential to the Internet of things,which would allow computers to manage all individual things. Defining the Internet of things as "simply the point in time when more 'things or objects' were connected to the Internet than people", Cisco Systems estimated that the IoT was "born" between 2008 and 2009, with the things/people ratio growing from 0.08 in 2003 to 1.84 in 2010.

Applications of IOT

1. Consumer applications

A growing portion of IoT devices are created for consumer use, including connected vehicles, home automation, wearable technology, connected health, and appliances with remote monitoring capabilities.

2. Smart home

IoT devices are a part of the larger concept of home automation, which can include lighting, heating and air conditioning, media and security systems.Long-term benefits could include energy savings by automatically ensuring lights and electronics are turned off.

A smart home or automated home could be based on a platform or hubs that control smart devices and appliances.For instance, using Apple's HomeKit, manufacturers can have their home products and accessories controlled by an application in iOS devices such as the iPhone and the Apple Watch. This could be a dedicated app or iOS native applications such as Siri.This can be demonstrated in the case of Lenovo's Smart Home Essentials, which is a line of smart home devices that are controlled through Apple's Home app or Siri without the need for a Wi-Fi bridge.There are also dedicated smart home hubs that are offered as standalone platforms to connect different smart home products and these include the Amazon Echo, Google Home, Apple's HomePod, and Samsung's SmartThings Hub. In addition to the commercial systems, there are many non-proprietary, open source ecosystems; including Home Assistant, OpenHAB and Domoticz.

3. Elder care

One key application of a smart home is to provide assistance for those with disabilities and elderly individuals. These home systems use assistive technology to accommodate an owner's specific disabilities.Voice control can assist users with sight and mobility limitations while alert systems can be connected directly to cochlear implants worn by hearing-impaired users. They can also be equipped with additional safety features. These features can include sensors that monitor for medical emergencies such as falls or seizures. Smart home technology applied in this way can provide users with more freedom and a higher quality of life.

The term "Enterprise IoT" refers to devices used in business and corporate settings. By 2019, it is estimated that the EIoT will account for 9.1 billion devices.

4. V2X communications

In vehicular communication systems, vehicle-to-everything communication (V2X), consists of three main components: vehicle to vehicle communication (V2V), vehicle to infrastructure communication (V2I) and vehicle to pedestrian communications (V2P). V2X is the first step to autonomous driving and connected road infrastructure.

5. Building and home automation

IoT devices can be used to monitor and control the mechanical, electrical and electronic systems used in various types of buildings (e.g., public and private, industrial, institutions, or residential) in home automation and building automation systems

6. Industrial applications

Main article: Industrial internet of things

Also known as IIoT, industrial IoT devices acquire and analyze data from connected equipment, operational technology (OT), locations and people. Combined with operational technology (OT) monitoring devices, IIoT helps regulate and monitor industrial systems.Also, the same implementation can be carried out for automated record updates of asset placement in industrial storage units as the size of the assets can vary from a small screw till the whole motor spare part and misplacement of such assets can cause a percentile loss of manpower time and money.

Introduction of electronics components :-

An electronic circuit comprises of various types of components, which are classified into two types: active components like transistors, diodes, IC's; and passive components like capacitors, resistors, inductors, etc.

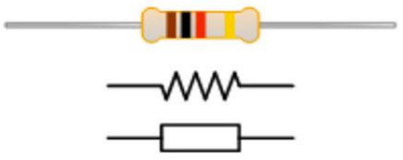
Passive Electronic Components

These components can store or maintains energy either in the form of current or voltage. Some of these components are discussed below.

1. Resistors

A resistor is a two-terminal passive electronics component, used to oppose or limit the current. Resistor works based on the principle of Ohm's law which states that "voltage applied across the terminals of a resistor is directly proportional to the current flowing through it".

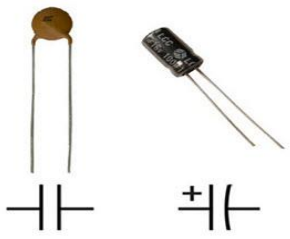
The units of the resistance is ohms. V=IR



2. Capacitors:

A capacitor made from two conductive plates with an insulator between them and it stores electrical energy in the form of an electric field. A capacitor blocks the DC signals and allows the AC signals and also used with a resistor in a timing circuit.

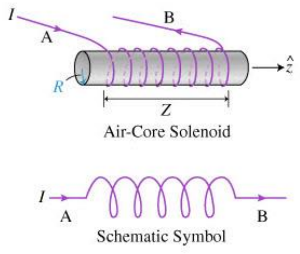
The stored charge is Q=CV.



3. Inductors

An inductor is also referred as AC resistor which stores electrical energy in the form of magnetic energy. It resists the changes in the current and the standard unit of inductance is Henry. Capability of producing magnetic lines is referred as inductance.

The inductance of the inductor is given as L= (μ.K.N2.S)/I.

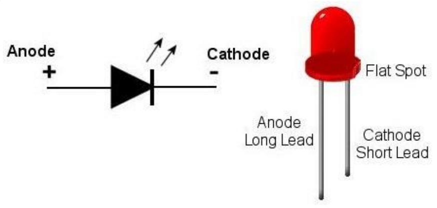


Active Electronic Components

These components rely on a source of energy and are able to control the electron flow through them. Some of these components are semiconductors like diodes, transistors, integrated circuits, various displays like LCD, LED, CRTs and power sources like batteries, PV cells and other AC and DC supply sources.

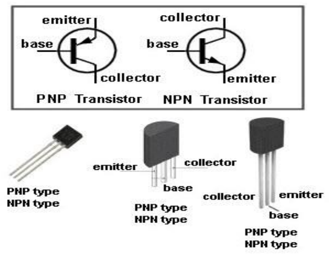
1. **Diodes**

A diode is a device that allows current to flow in one direction and usually made with semiconductor material. It has two terminals, anode and cathode terminals. These are mostly used in converting circuits like AC to DC circuits. These are are of different types like PN diodes, Zener diodes, LEDs, photo diodes, etc.



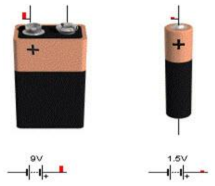
2. **Transistors**

A transistor is a three terminal semiconductor device. Mostly it is used as switching device and also as an amplifier. This switching device can be a voltage or current controlled.By controlling the voltage applied to the one terminal controls the current flow through the other two terminals. Transistors are of two types, namely bipolar junction transistor (BJT) and field effect transistors (FET). And further these can be PNP and NPN transistors.



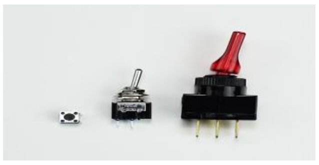
- **Batteries**

Batteries are most common power source for standalone industrial, domestic and handheld device applications. It converts chemical energy into electrical energy through electrochemical discharge reactions.



- **Switch**

Switches can come in many forms such as pushbutton, rocker, momentary and others. Their basic function is to interrupt electric current by turning a circuit on or off.

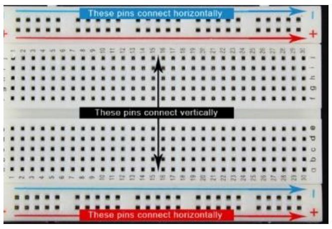


Introduction of IOT Tools

Now that you have a good workspace set up, it's time to stock it with the proper tools and equipment. This isn't a complete list but it does highlight the most common items used in electronics.

1. Breadboard

Breadboards are an essential tool for prototyping and building temporary circuits. These boards contain holes for inserting wire and components. Because of their temporary nature, they allow you to create circuits without soldering. The holes in a breadboard are connected in rows both horizontally and vertically as shown below.


These pins connect horizontally
These pins connect vertically
These pins connect horizontally

2. Digital Multimeter

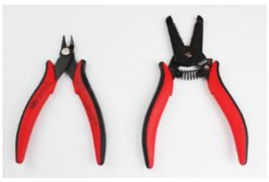
A multimeter is a device that's used to measure electric current (amps), voltage (volts) and resistance (ohms). It's a great for troubleshooting circuits and is capable of measuring both AC and DC voltage. Check out this post for more info on how to use a multimeter.

3. Test Leads (Alligator Clips)

Test leads are great for connecting components together to test a circuit without the need for soldering.

4. Wire Cutter

Wire cutters are essential for stripping stranded and solid copper wire.

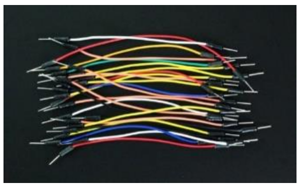


5. Heat Gun

A heat gun is used to shrink plastic tubing known as heat shrink to help protect exposed wire. Heat shrink has been called the duct tape of electronics and comes in handy in a wide variety of applications.

6. Jumper Wire

These wires are used with breadboard and development boards and are generally 22-28 AWG solid core wire. Jumper wires can have male or female ends depending on how they need to be used.



7. Soldering Iron

When it time to create a permanent circuit, you'll want to solder the parts together. To do this, a soldering iron is the tool you would use. Of course a soldering iron isn't any good unless you have solder to go with it. You can choose leaded or lead-free solder in a few diameters.

Introduction to Tinkercad

Tinkercad was founded as a company in 2010 in the European Union[4] by former Google engineer Kai Backman and his cofounder Mikko Mononen, with a goal to make 3D modeling, especially the design of physical items, accessible to the general public, and allow users to publish their designs under a Creative Commons license. In 2011, the tinkercad.com website was launched as a web-based 3D modeling tool for WebGL-enabled browsers, and in 2012 the company moved its headquarters to San Francisco. By 2012 over 100,000 3D designs had been published by users.

In May 2013, Autodesk announced at a Maker Faire that they would acquire Tinkercad. [n]

In May 2017, Autodesk discontinued its 123D Circuits (Circuits.io) "Electronics Lab" feature and merged it into Tinkercad.

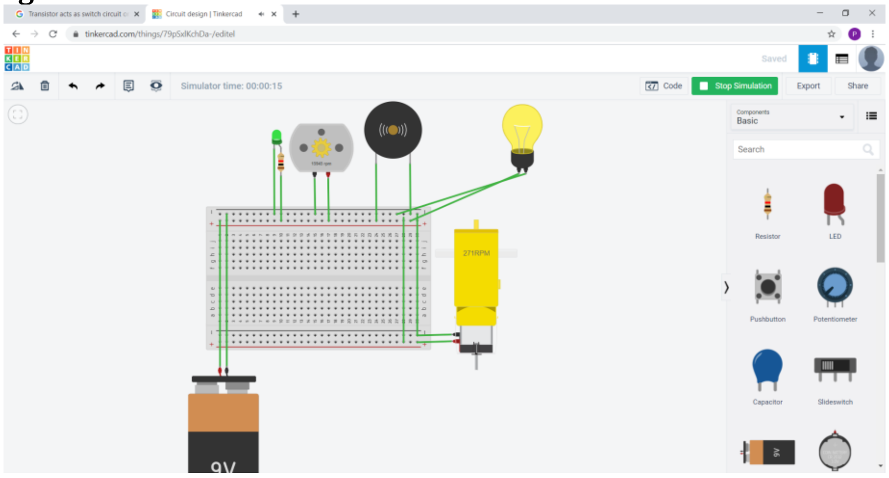
TIN KER CAD AUTODESK® TINKERCAD®

Tinkercad already has a lot to offer as a design program, but it also serves as a replacement for Autodesk's discontinued "123D Circuits" service, which was a free and easy to use breadboard simulator. This article will introduce you to the basics of Tinkercad Circuits which, like Fritzing, is a great design resource for makers. Learn how to use Tinkercad to design, build, and test simple circuits.

1. To get started, visit Tinkercad's website and create an account or log into an existing one. Then select "Circuits" on the left side of the screen.
2. Select "Create new Circuit" on the next page and you'll be greeted.
3. After showing left side various type components .as per your circuit select the component and build the circuit.

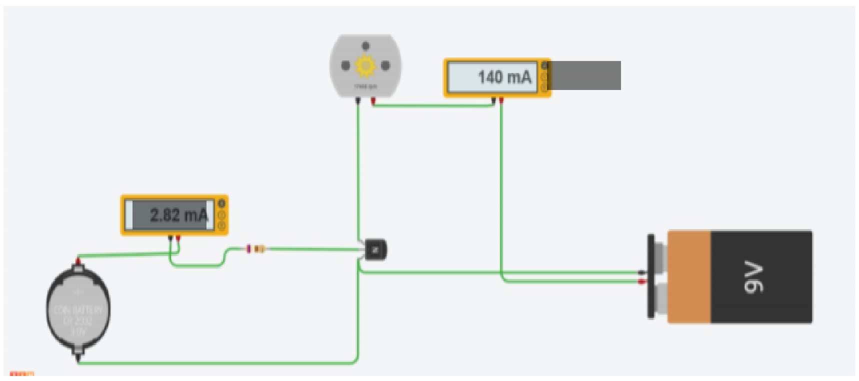
Eg:-Builded Circuit on tikercad website shown below.

1. **Using breadboard basic circuits.**



In this circuit use breadboard , LED with resister , battery 9v, DC moter , piezo , hobby gearmoter , bulib ect. Click the start simulation option and circuit is operated ..

2. Motor driver circuit using transistor as a switch

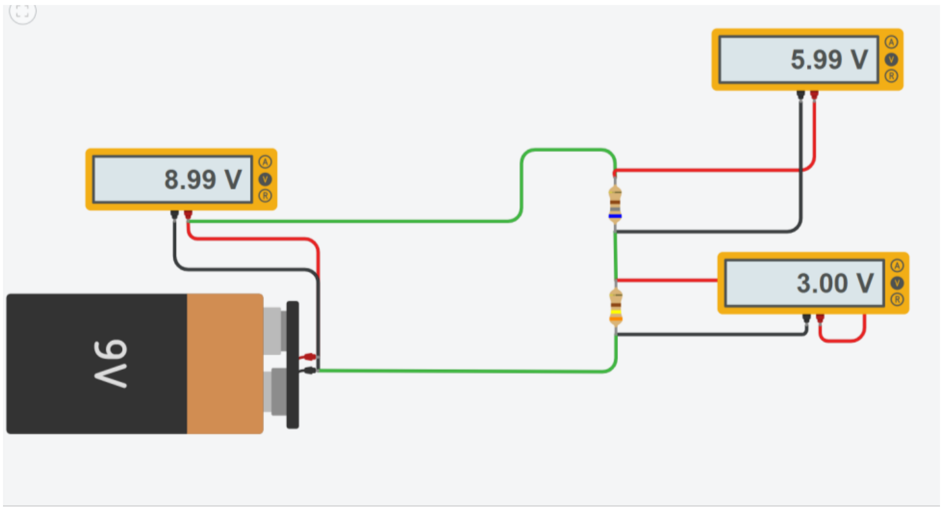


Circuit components list

Name	Quantity	Component
M1	1	DC Motor
T1	1	NPN Transistor (BJT)
BAT3	1	9V Battery
R2	1	785 â„¦ Resistor
Meterlc1, Meterlb1	2	Amperage Multimeter
BAT1	1	Coin Cell 3V Battery

Description -: Motors work through a process called induction. When you an put electric charge through wire, a magnetic field is created. A coiled wire will create a stronger field, as will increased current. In a DC motor, a coiled wire surrounds the motor's shaft. The generated magnetic field is pulled and repulsed by magnets inside the motor's body.
When a motor stops, there is the potential for a small amount of current to be generated as the shaft continues spinning. A diode placed in parallel with the motor leads will keep any generated electricity from damaging your circuit.
Motors will pull the most current when they start up, or have a load. The stall current is the amount of current a motor will pull when it is stopped by a force. When a motor is up and running, it will pull significantly less current.
The voltage rating describes the peak operating voltage for a motor, when it works at optimum efficiency. Going over or under the motor's rated voltage will, over time, shorten the motor's life. If you provide less than the rated voltage, the motor will spin more slowly. Typically, a motor needs about 1/2 its rated voltage to run. If you provide less than that when starting up, it probably won't begin to move.

3. Voltage divider circuit



Circuit components list

Name	Quantity	Component
R1	1	500 â„¦ Resistor
R2	1	200 â„¦ Resistor
BAT1	1	9V Battery
Meter2, Meter4, Meter5	3	Voltage Multimeter

Description-: In electronics, a **voltage divider** (also known as a **potential divider**) is a passive linear circuit that produces an output voltage (V_{out}) that is a fraction of its input voltage (V_{in}). **Voltage division** is the result of distributing the input voltage among the components of the divider. A simple example of a voltage divider is two resistors connected in series, with the input voltage applied across the resistor pair and the output voltage emerging from the connection between them.

Week 2

(8/6/2020 To 13/6/2020)

Topics of the week -:

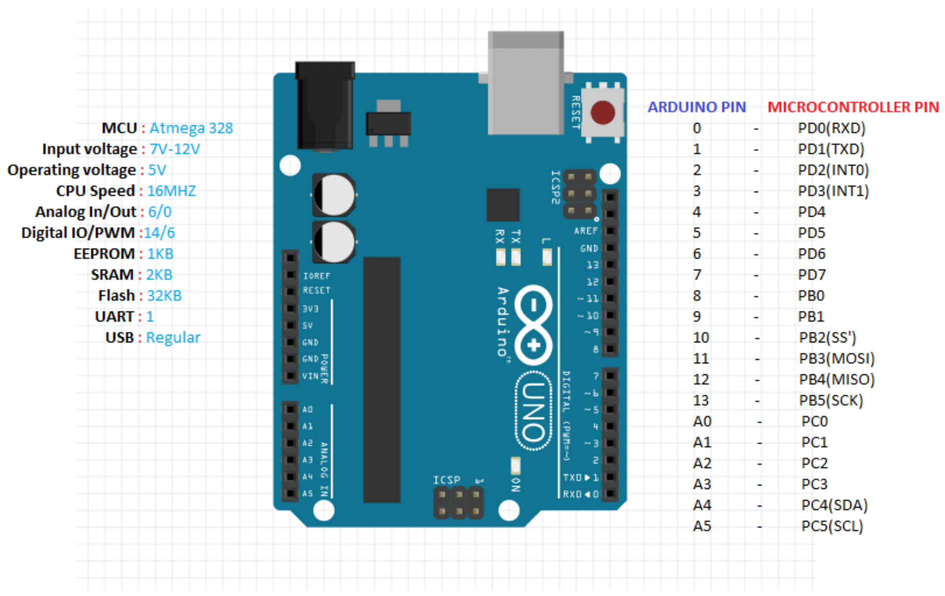
1. Introduction of Arduino
2. Introduction of Scratch block programming to arduino .
3. Introduction of Arduino programming
4. concept of coding for ardiuno. (in Embedded c)
    - digital write()
    - digital read()
    - if()statement/boolen
    - delay
    - analog read() and analog write
    - serial communication
5. Concept Pull up and pull down switch circuits.
6. Tinkercad circuit with Arduino program
    1) LED control control using Arduino programming
    2) Traffic light control circuit

Introduction of Arduino

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online.
Arduino board designs use a variety of microprocessors and controllers. The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards ('shields') or breadboards (for prototyping) and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs from personal computers. The microcontrollers can be programmed using the C and C++ programming languages, using a standard API which is also known as the "Arduino language". In addition to using traditional compiler toolchains, the Arduino project provides an integrated development environment (IDE) and a command line tool (arduino-cli) developed in Go. The Arduino project started in 2005 as a tool for students at the Interaction Design Institute Ivrea in Ivrea, Italy, aiming to provide a low-cost and easy way for novices and professionals to create devices that interact with their environment using sensors and actuators. Common examples of such devices intended for beginner hobbyists include simple robots, thermostats and motion detectors.

The key features are:

1. Arduino boards are able to read analog or digital input signals from different sensors and turn it into an output such as activating a motor, turning LED on/off, connect to the cloud and many other actions.
2. You can control your board functions by sending a set of instructions to the microcontroller on the board via Arduino IDE (referred to as uploading software).
3. Unlike most previous programmable circuit boards, Arduino does not need an extra piece of hardware (called a programmer) in order to load a new code onto the board. You can simply use a USB cable.



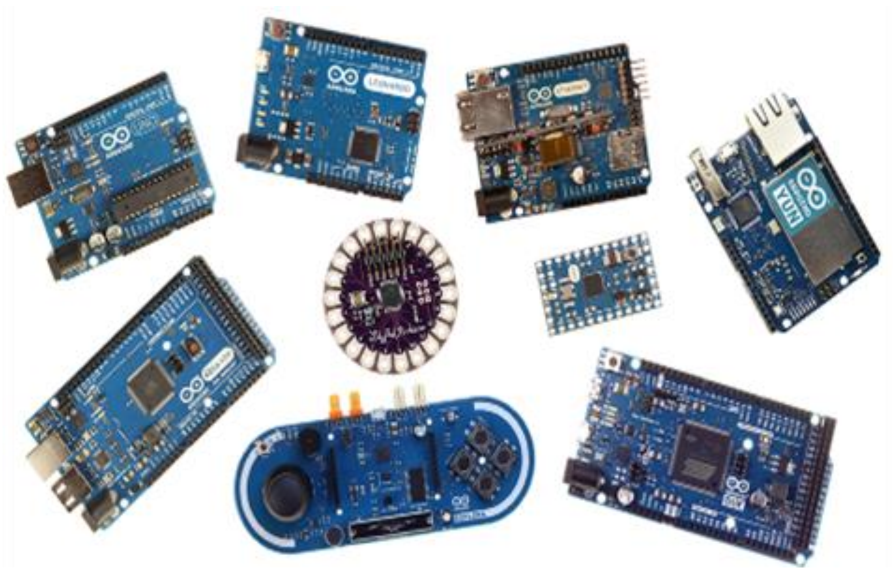
MCU :	Atmega 328
Input voltage :	7V-12V
Operating voltage :	5V
CPU Speed :	16MHZ
Analog In/Out :	6/0
Digital IO/PWM :	14/6
EEPROM :	1KB
SRAM :	2KB
Flash :	32KB
UART :	1
USB :	Regular

ARDUINO PIN		MICROCONTROLLER PIN
0	-	PD0(RXD)
1	-	PD1(TXD)
2	-	PD2(INT0)
3	-	PD3(INT1)
4	-	PD4
5	-	PD5
6	-	PD6
7	-	PD7
8	-	PB0
9	-	PB1
10	-	PB2(SS')
11	-	PB3(MOSI)
12	-	PB4(MISO)
13	-	PB5(SCK)
A0	-	PC0
A1	-	PC1
A2	-	PC2
A3	-	PC3
A4	-	PC4(SDA)
A5	-	PC5(SCL)

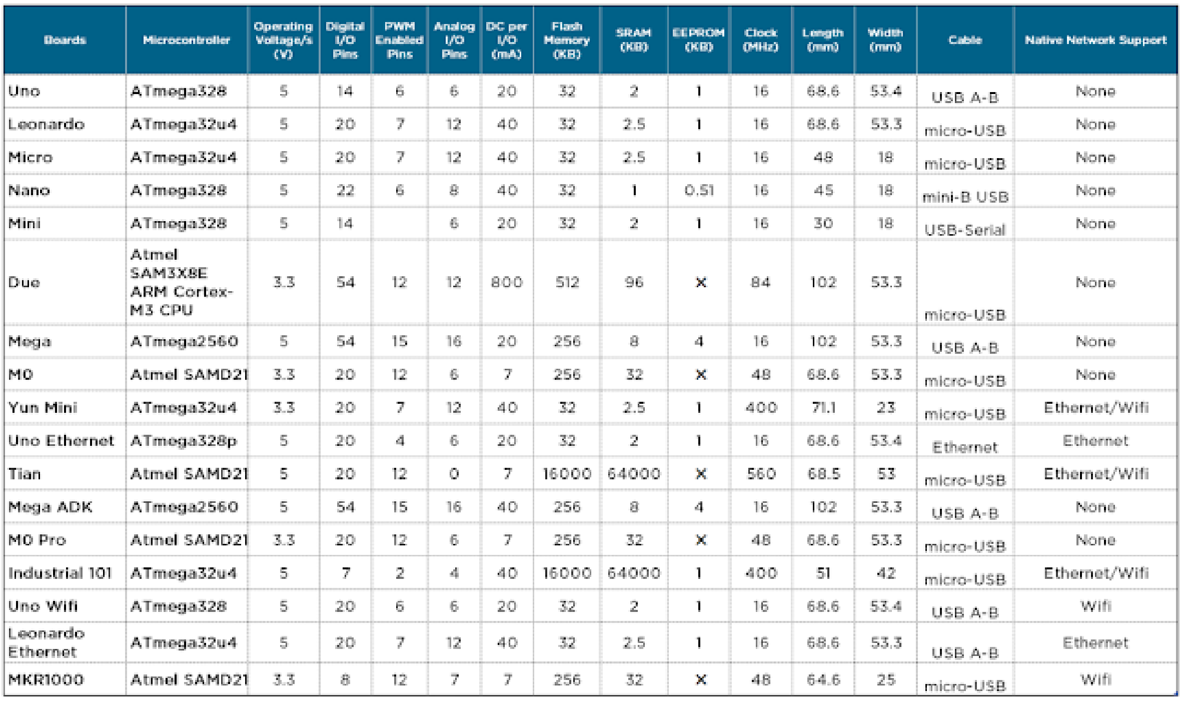
Fig-: pin diagram of arduino

Different Types Of Arduino Boards

Arduino board was designed in the Ivrea Interaction Design Institute intended for students without a background in electronics and programming concept. This board started altering to adapt to new requirements and challenges, separating its present from simple 8-bit boards to products for IoT (Internet of Things) applications, 3D printing, wearable, and embedded surroundings. All boards are entirely open-source, allowing users to build them separately and finally adapt them to their exact needs. Over the years the Arduino boards has been used to build thousands of projects, from daily objects to compound scientific instruments. An international community of designers, artists, students, programmers, hobbyists and experts has gotten together around this open source stage, their donations have added up to an unbelievable amount of available knowledge that can be of immense help to beginners and specialists alike.



Here is a list of different Arduino boards available.



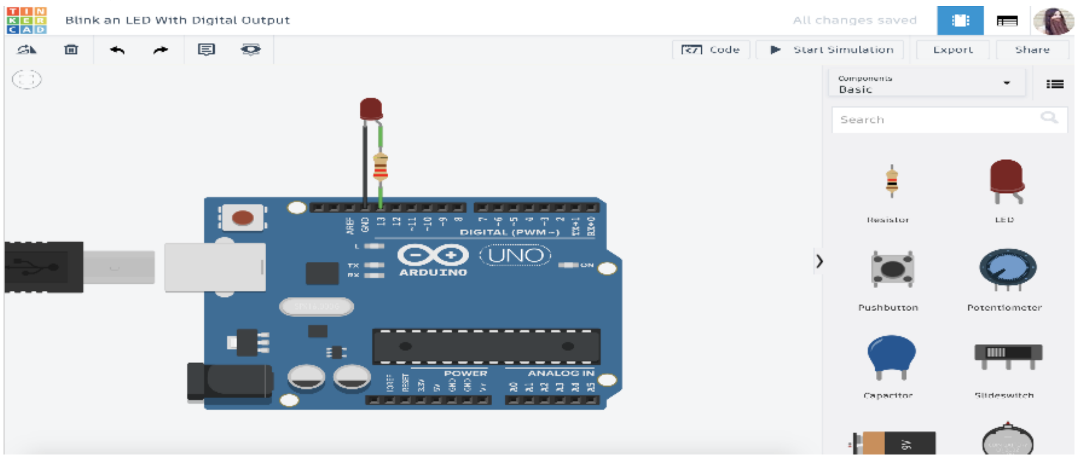
Boards	Microcontroller	Operating Voltage/s (V)	Digital I/O Pins	PWM Enabled Pins	Analog I/O Pins	DC per I/O (mA)	Flash Memory (KB)	SRAM (KB)	EEPROM (KB)	Clock (MHz)	Length (mm)	Width (mm)	Cable	Native Network Support
Uno	ATmega328	5	14	6	6	20	32	2	1	16	68.6	53.4	USB A-B	None
Leonardo	ATmega32u4	5	20	7	12	40	32	2.5	1	16	68.6	53.3	micro-USB	None
Micro	ATmega32u4	5	20	7	12	40	32	2.5	1	16	48	18	micro-USB	None
Nano	ATmega328	5	22	6	8	40	32	1	0.51	16	45	18	mini-B USB	None
Mini	ATmega328	5	14		6	20	32	2	1	16	30	18	USB-Serial	None
Due	Atmel SAM3X8E ARM Cortex-M3 CPU	3.3	54	12	12	800	512	96	X	84	102	53.3	micro-USB	None
Mega	ATmega2560	5	54	15	16	20	256	8	4	16	102	53.3	USB A-B	None
M0	Atmel SAMD21	3.3	20	12	6	7	256	32	X	48	68.6	53.3	micro-USB	None
Yun Mini	ATmega32u4	3.3	20	7	12	40	32	2.5	1	400	71.1	23	micro-USB	Ethernet/Wifi
Uno Ethernet	ATmega328p	5	20	4	6	20	32	2	1	16	68.6	53.4	Ethernet	Ethernet
Tian	Atmel SAMD21	5	20	12	0	7	16000	64000	X	560	68.5	53	micro-USB	Ethernet/Wifi
Mega ADK	ATmega2560	5	54	15	16	40	256	8	4	16	102	53.3	USB A-B	None
M0 Pro	Atmel SAMD21	3.3	20	12	6	7	256	32	X	48	68.6	53.3	micro-USB	None
Industrial 101	ATmega32u4	5	7	2	4	40	16000	64000	1	400	51	42	micro-USB	Ethernet/Wifi
Uno Wifi	ATmega328	5	20	6	6	20	32	2	1	16	68.6	53.4	USB A-B	Wifi
Leonardo Ethernet	ATmega32u4	5	20	7	12	40	32	2.5	1	16	68.6	53.3	USB A-B	Ethernet
MKR1000	Atmel SAMD21	3.3	8	12	7	7	256	32	X	48	64.6	25	micro-USB	Wifi

Application of Arduino :-

- Arduboy, a handheld game console based on Arduino
- Arduinome, a MIDI controller device that mimics the Monome
- Ardupilot, drone software and hardware
- ArduSat, a cubesat based on Arduino.
- C-STEM Studio, a platform for hands-on integrated learning of computing, science, technology, engineering, and mathematics (C-STEM) with robotics.
- Data loggers for scientific research.
- OBDuino, a trip computer that uses the on-board diagnostics interface found in most modern cars.

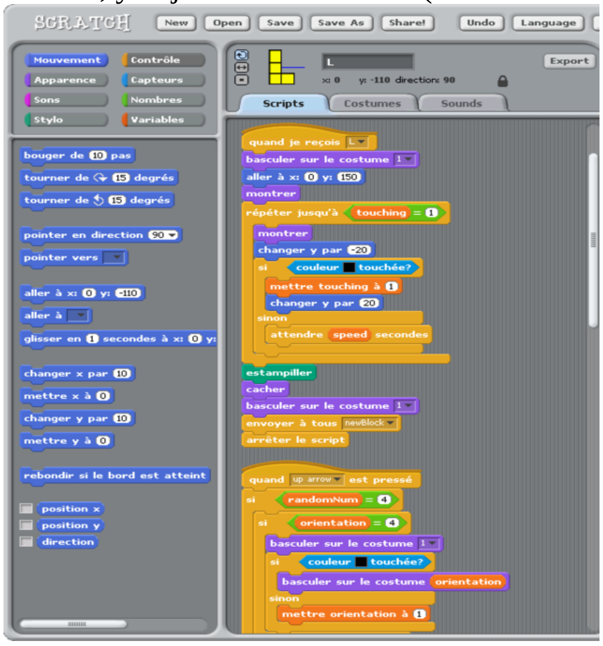
Using Arduino circuit on tinkercad websites

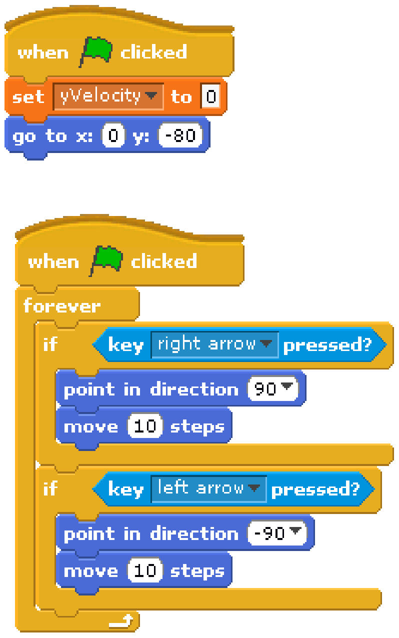
1. Basic ardiuno circuit using LED and resistor . click the simulate option then circuit is operated LED is start.



Introduction of Scratch block programming to arduino .

Scratch is a programming language developed for kids to learn programming in an interactive way. In Scratch, you join labeled blocks (which serve as code snippets) to write a full-fledged program or game.

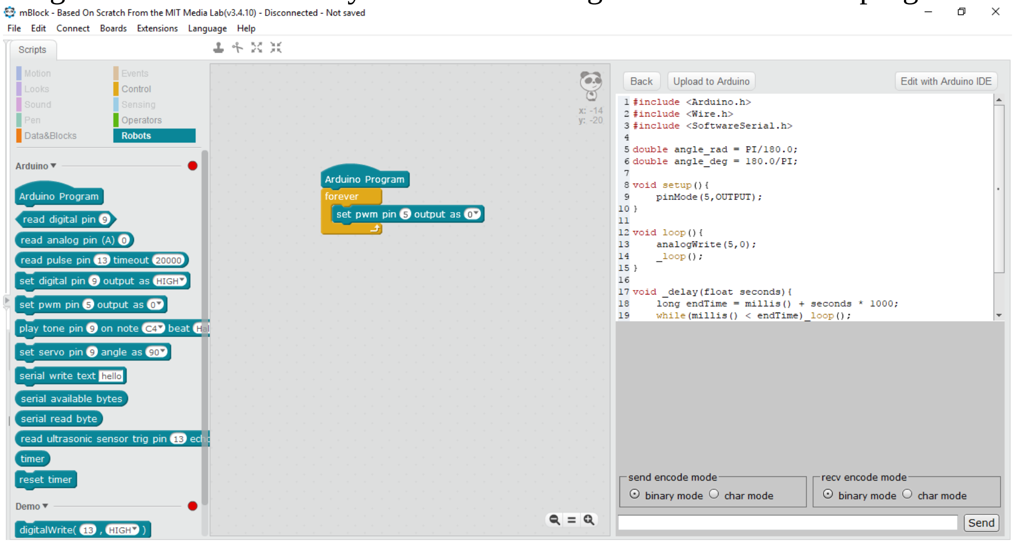




mBlock

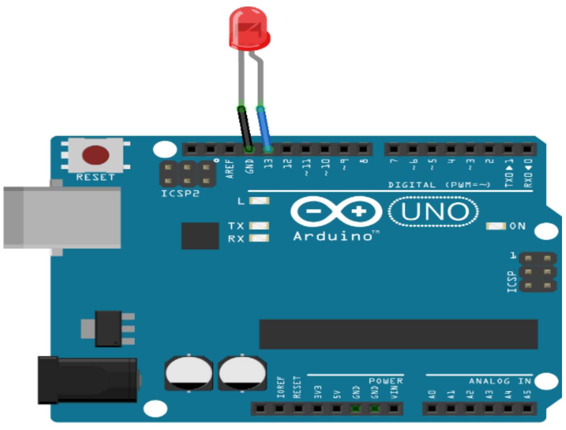
Using Scratch, some users have developed another mod of scratch called mBlock. The difference between mBlock and Scratch is that mBlock allows you to program the Arduino in an easy and interactive way.

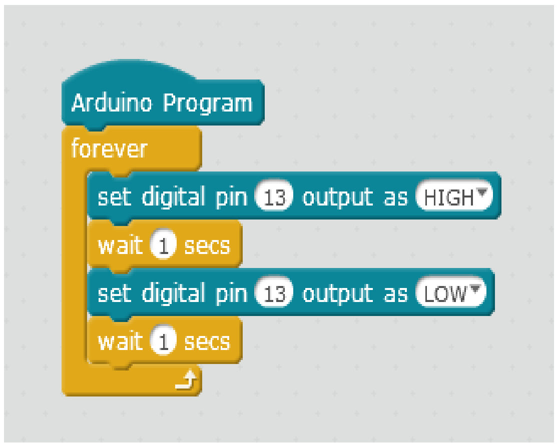
An interesting thing about mBlock is that you can see the original C++ code after programming Arduino.


mBlock - Based On Scratch From the MIT Media Lab(v3.4.10) - Disconnected - Not saved
File Edit Connect Boards Extensions Language Help
Scripts
Arduino Program
forever
set pwm pin 5 output as 0
Back
Upload to Arduino
Edit with Arduino IDE
send encode mode
recv encode mode
Send

```
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>

double angle_rad = PI/180.0;
double angle_deg = 180.0/PI;

void setup(){
    pinMode(5,OUTPUT);
}

void loop(){
    analogWrite(5,0);
    _loop();
}

void _delay(float seconds){
    long endTime = millis() + seconds * 1000;
    while(millis() < endTime)_loop();
```

Making an LED Blink Using mBlock





```
Arduino Program
forever
    set digital pin 13 output as HIGH
    wait 1 secs
    set digital pin 13 output as LOW
    wait 1 secs
```

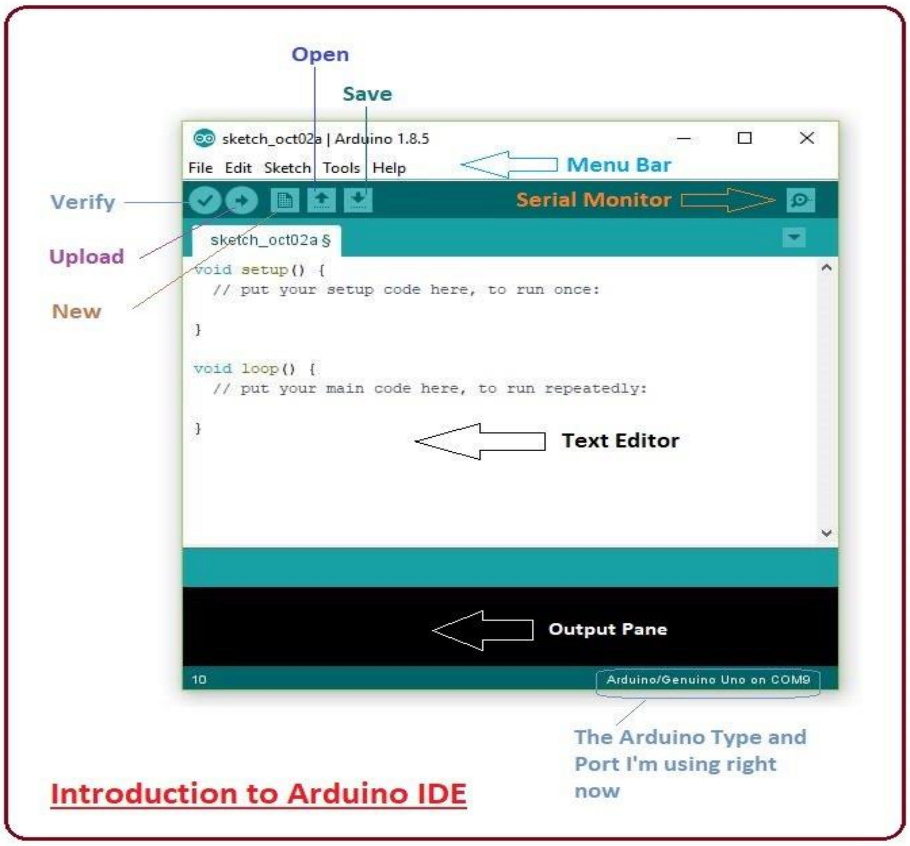
In the program above, you can see that we have placed an Arduino block and a forever block. These two blocks are musts for programming an Arduino.
The point of using a forever block is, in an Arduino program, the logic should be such that it would run in a loop indefinitely. In our case, we need to blink the LED again and again, so use of a forever block is a must in many cases, and it makes life a lot easier when programming Arduino.
Inside forever, set which digital pin block is used. This block can make a pin voltage high or low. So, if I have an LED connected to pin number 13 of the Arduino (below) and I want to turn it on, I will use “set digital pin 13 output as HIGH”, and my LED will light up.

Introduction of Arduino programing using Arduino software

the **Arduino Integrated Development Environment (IDE)** is a cross-platform application (for Windows, macOS, Linux) that is written in functions from C and C++. It is used to write and upload programs to Arduino compatible boards, but also, with the help of third-party cores, other vendor development boards. The Arduino IDE is very simple and this simplicity is probably one of the main reason Arduino became so popular. We can certainly state that being compatible with the Arduino IDE is now one of the main requirements for a new microcontroller board. Over the years, many useful features have been added to the Arduino IDE and you can now managed third-party libraries and boards from the IDE, and still keep the simplicity of programming the board.

How to run program-:

1. Once the software starts.
2. create a new project
3. open an existing project example.
4. To create a new project, select File → New. And then edit the file.
5. To open an existing project example, select File → Example → Basics → (select the example from the list).


Open
Save
sketch_oct02a | Arduino 1.8.5
File Edit Sketch Tools Help
Menu Bar
Verify
Upload
New
Serial Monitor
sketch_oct02a §
Text Editor
Output Pane
10
Arduino/Genuino Uno on COM9
The Arduino Type and Port I'm using right now
Introduction to Arduino IDE

```
void setup() {
  // put your setup code here, to run once:

}

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

}
```

Concept Of Coding For Ardiuno

1. Digital Write() :- Write a HIGH or a LOW value to a digital pin.
If the pin has been configured as an OUTPUT with pinMode(), its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW.
Syntax
digitalWrite(pin, value)
2. Digital Read() :- Reads the value from a specified digital pin, either HIGH or LOW.

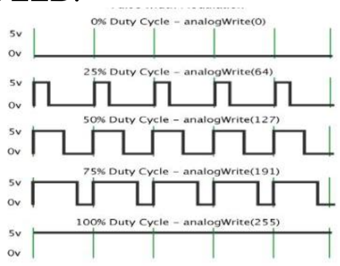
Syntax
digitalRead(pin)

3. **If( ) Statement:-** The if statement checks for a condition and executes the proceeding statement or set of statements if the condition is 'true'.

Syntax
```
if (condition) {
  //statement(s)
}
```
4. **Delay()** [Time]
Pauses the program for the amount of time (in milliseconds) specified as parameter. (There are 1000 milliseconds in a second.)

Syntax
delay(ms)

5. **Analog read and analog write (input and output) in Arduino-:**
Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means. Digital control is used to create a square wave, a signal switched between on and off. This on-off pattern can simulate voltages in between full on (5 Volts) and off (0 Volts) by changing the portion of the time the signal spends on versus the time that the signal spends off. The duration of "on time" is called the pulse width. To get varying analog values, you change or modulate, that pulse width. If you repeat this on-off pattern fast enough with an LED, for example, the result is as if the signal is a steady voltage between 0 and 5v controlling the brightness of the LED.



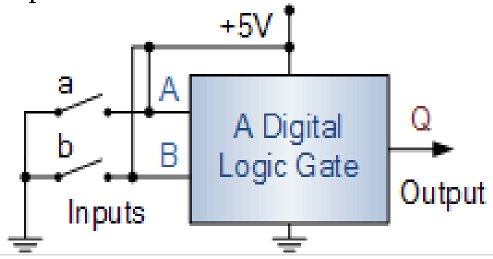
In the graphic below, the green lines represent a regular time period. This duration or period is the inverse of the PWM frequency. In other words, with Arduino's PWM frequency at about 500Hz, the green lines would measure 2 milliseconds each. A call to analogWrite() is on a scale of 0 – 255, such that analogWrite(255) requests a 100% duty cycle (always on), and analogWrite(127) is a 50% duty cycle (on half the time) .

6. **Serial Communication**
Serial communication on pins TX/RX uses TTL logic levels (5V or 3.3V depending on the board). Don't connect these pins directly to an RS232 serial port; they operate at +/- 12V and can damage your Arduino board.

Concept Pull up and pull down switch circuits.

Pull-up Resistors

The most common method of ensuring that the inputs of digital logic gates and circuits can not self-bias and float about is to either connect the unused pins directly to ground (0V) for a constant low "0" input, (OR and NOR gates) or directly to Vcc (+5V) for a constant high "1" input (AND and NAND gates).
lets look again at our two switched inputs.



This time, to stop the two inputs, A and B, from “floating” about when the corresponding switches, “a” and “b” are open (OFF), the two inputs are connected to +5V supply. when switch “a” is open (OFF), the input is connected to Vcc (+5V) and when the switch is closed (ON), the input is connected to ground as before, then inputs “A” or “B” always have a default state regardless of the position of the switch. However, this is a bad condition because when either of the switches are closed (ON), there will be a direct short circuit between the +5V supply and ground, resulting in excessive current flow either blowing a fuse or damaging the circuit which is not good news. One way to overcome this issue is to use a pull-up resistor connected between the input pin and the +5V supply rail as shown.

Calculating Pull-up Resistor Value

$$R_{MAX} = \frac{V_{CC} - V_{IH(MIN)}}{I_{IH}}$$

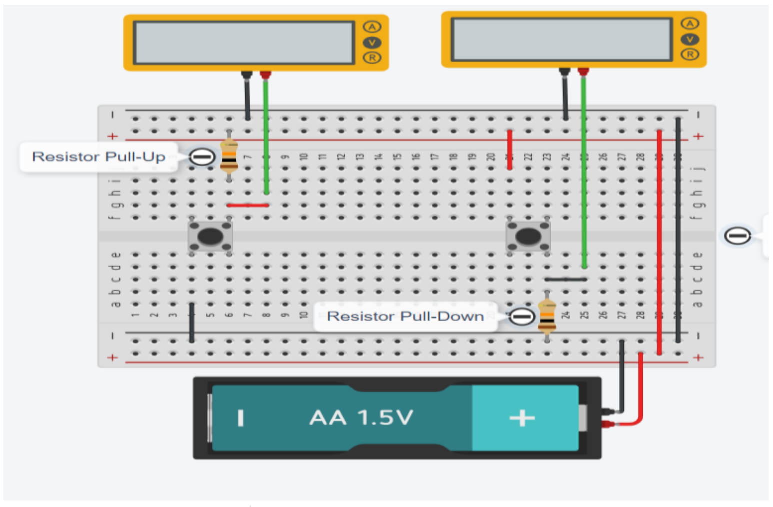
Pull-down Resistors

A Pull-down *resistor* works in the same way as the previous pull-up resistor, except this time the logic gates input is tied to ground, logic level “0” (LOW) or it may go HIGH by the operation of a mechanical switch. This pull-down resistor configuration is particularly useful for digital circuits like latches, counters and flip-flops that require a positive one-shot trigger when a switch is momentarily closed to cause a state change. While they may seem to operate in the same way as the pull-up resistor, the resistive value of a passive pull-down resistor is more critical with TTL logic gates than with similar CMOS gates. This is because a TTL input sources much more current out of its input in its LOW state.

Pull-down Resistor Value

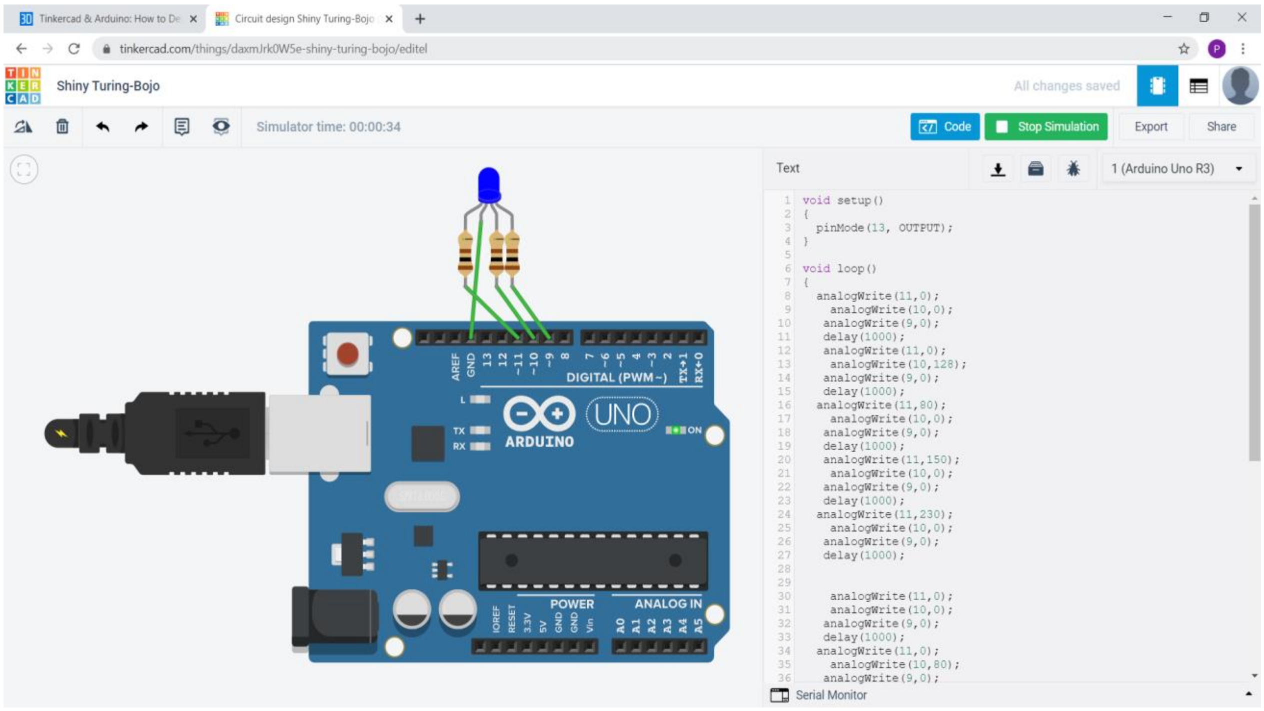
$$R_{MAX} = \frac{V_{IL(MAX)} - 0}{I_{IL}}$$

Pull up and pull down resistance circuit on tinkercad



Tinkercad circuit with Arduino program

1. LED control circuit



```
void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  analogWrite(11,0);
    analogWrite(10,0);
   analogWrite(9,0);
   delay(1000);
   analogWrite(11,0);
    analogWrite(10,128);
   analogWrite(9,0);
   delay(1000);
  analogWrite(11,80);
    analogWrite(10,0);
   analogWrite(9,0);
   delay(1000);
   analogWrite(11,150);
    analogWrite(10,0);
   analogWrite(9,0);
   delay(1000);
  analogWrite(11,230);
    analogWrite(10,0);
   analogWrite(9,0);
   delay(1000);


    analogWrite(11,0);
    analogWrite(10,0);
   analogWrite(9,0);
   delay(1000);
  analogWrite(11,0);
    analogWrite(10,80);
   analogWrite(9,0);
```

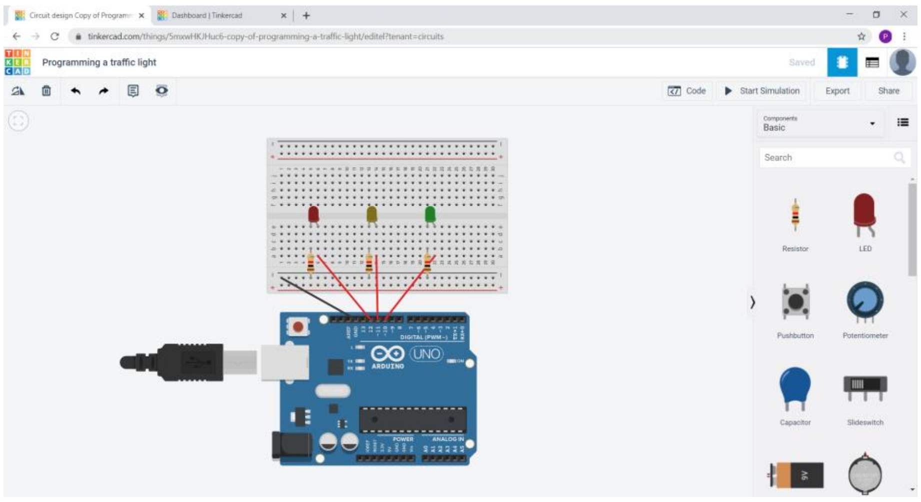
In this circuit to need 1 LED , 3 resistor (100 ohm), Arduino uno . all components selected and connected to input pins (R=11 pin ,B=10 pin,G=9 pin) with resister .and connect the cathod to ground .then program is success full run. After click simulate circuit and observed the automatic LED color change with delay time .

Program start -:

```
void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  analogWrite(11,0);
    analogWrite(10,0);
   analogWrite(9,0);
   delay(1000);
   analogWrite(11,0);
    analogWrite(10,128);
   analogWrite(9,0);
   delay(1000);
  analogWrite(11,80);
    analogWrite(10,0);
   analogWrite(9,0);
   delay(1000);
   analogWrite(11,150);
    analogWrite(10,0);
   analogWrite(9,0);
   delay(1000);
  analogWrite(11,230);
    analogWrite(10,0);
   analogWrite(9,0);
   delay(1000);


    analogWrite(11,0);
    analogWrite(10,0);
   analogWrite(9,0);
   delay(1000);
  analogWrite(11,0);
    analogWrite(10,80);
   analogWrite(9,0);
   delay(1000);
   analogWrite(11,0);
    analogWrite(10,150);
   analogWrite(9,0);
   delay(1000);
  analogWrite(11,0);
    analogWrite(10,230);
   analogWrite(9,0);
   delay(1000);

    analogWrite(11,0);
    analogWrite(10,0);
   analogWrite(9,0);
   delay(1000);
  analogWrite(11,0);
    analogWrite(10,0);
   analogWrite(9,80);
   delay(1000);
   analogWrite(11,0);
    analogWrite(10,0);
   analogWrite(9,150);
   delay(1000);
  analogWrite(11,0);
    analogWrite(10,0);
   analogWrite(9,230);
   delay(1000);

  delay(10);
   }
```

2. Traffic light control circuit

.In this circuit 3 LED different color (red, green, yellow) ,3 resistance (1K ohm), Arduino, breadboard .then set the programe for traffic control . set the 0.25 sec delay every light the program successfully run.then click the simulate.programing this circuit given below .


Circuit design Copy of Program
Dashboard | Tinkercad
tinkercad.com/things/5mxwHKJHuc6-copy-of-programming-a-traffic-light/editel?tenant=circuits
Programming a traffic light
Saved
Code
Start Simulation
Export
Share
Components
Basic
Search
Resistor
LED
Pushbutton
Potentiometer
Capacitor
Slideswitch
UNO
ARDUINO

Program:-

```
int rojo=12; //It defines the value of the pin for the red led
int amarillo=11; //It defines the value of the pin for the yellow led
int verde=10; //It defines the value of the pin for the green led

void setup() { //declarations
pinMode(verde,OUTPUT); //It declares the green pin as output
pinMode(amarillo,OUTPUT);//It declares the yellow pin as output
pinMode(rojo,OUTPUT); //It declares the red pin as output
}
void loop() { //repeat loop continuously
digitalWrite(verde,HIGH); //It turns on the green led
delay(15000); //wait 15 seconds
digitalWrite(verde,LOW); //It turns off the green led
delay(250); //wait 0.25 seconds

digitalWrite(amarillo,HIGH); //It turns on the yellow led
delay(3000); //wait 3 seconds
digitalWrite(amarillo,LOW); //It turns off the yellow led
delay(250); //wait 0.25 seconds

digitalWrite(rojo,HIGH); //It turns the red led
delay(15000); //wait 15 seconds
digitalWrite(rojo,LOW); //It turns off the red led
delay(250); //wait 0.25 seconds  }
```

WEEK 3

(15/6/2020 To 21/6/2020)

Topics of the week

1) Basic C programming loop control statement
    - for loop
    - while loop
    - do while loop
    - break statement
2) Buzzer working
    - Circuit of piano using Arduino and buzzer
3) Introduction to Mit app inventer
4) Introduction of Bluetooth module (HC 0 5)

Basic C programming loop control statement

A **Loop** executes the sequence of statements many times until the stated condition becomes false. A loop consists of two parts, a body of a loop and a control statement. The control statement is a combination of some conditions that direct the body of the loop to execute until the specified condition becomes false. The purpose of the loop is to repeat the same code a number of times.
'C' programming language provides us with three types of loop constructs:
1. The while loop
2. The do-while loop
3. The for loop

1. The while loop

A while loop is the most straightforward looping structure. The basic format of while loop is as follows:

```
while (condition) {
          statements;
}
```

It is an entry-controlled loop. In while loop, a condition is evaluated before processing a body of the loop. If a condition is true then and only then the body of a loop is executed. After the body of a loop is executed then control again goes back at the beginning, and the condition is checked if it is true, the same process is executed until the condition becomes false. Once the condition becomes false, the control goes out of the loop. After exiting the loop, the control goes to the statements which are immediately after the loop. The body of a loop can contain more than one statement. If it contains only one statement, then the curly braces are not compulsory. It is a good practice though to use the curly braces even we have a single statement in the body.

2. The do-while loop

A do-while loop is similar to the while loop except that the condition is always executed after the body of a loop. It is also called an exit-controlled loop.
The basic format of while loop is as follows:

```
 do {
  statements
} while (expression);
```

As we saw in a while loop, the body is executed if and only if the condition is true. In some cases, we have to execute a body of the loop at least once even if the condition is false. This type of operation can be achieved by using a do-while loop. In the do-while loop, the body of a loop is always executed at least once. After the body is executed, then it checks the condition. If the condition is true, then it will again execute the body of a loop otherwise control is transferred out of the loop. Similar to the while loop, once the control goes out of the loop the statements which are immediately after the loop is executed.

3. The for loop

A for loop is a more efficient loop structure in 'C' programming. The general structure of for loop is as follows:

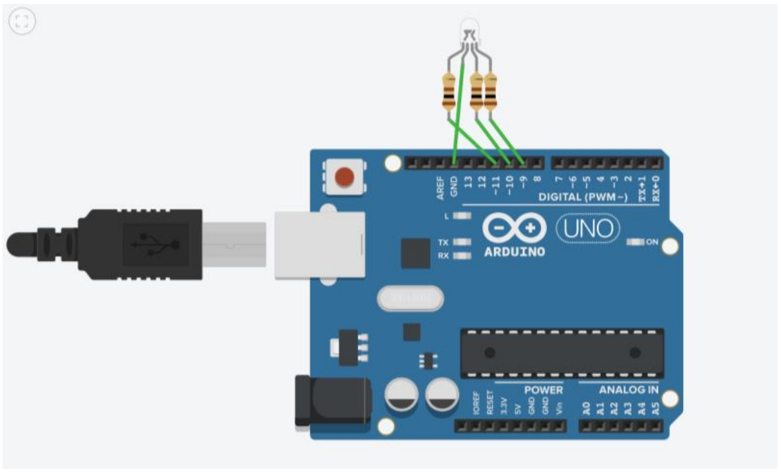
```
for (initial value; condition; incrementation or decrementation )
{
statements;
}
```

The initial value of the for loop is performed only once.

The condition is a Boolean expression that tests and compares the counter to a fixed value after each iteration, stopping the for loop when false is returned.

The incrementation/decrementation increases (or decreases) the counter by a set value.

Led blinking circuit using led by looping program



Program start

```
void setup()
{
  pinMode(13, OUTPUT);
}
void loop()
{
  analogWrite(11,0);
    analogWrite(10,0);
   analogWrite(9,0);
   delay(1000);
   analogWrite(11,0);
    analogWrite(10,128);
   analogWrite(9,0);
   delay(1000);
  analogWrite(11,80);
    analogWrite(10,0);
   analogWrite(9,0);
   delay(1000);
   analogWrite(11,150);
    analogWrite(10,0);
   analogWrite(9,0);
  delay(1000);
 analogWrite(11,230);
   analogWrite(10,0);
  analogWrite(9,0);
  delay(1000);


   analogWrite(11,0);
   analogWrite(10,0);
  analogWrite(9,0);
  delay(1000);
 analogWrite(11,0);
   analogWrite(10,80);
  analogWrite(9,0);
  delay(1000);
  analogWrite(11,0);
   analogWrite(10,150);
  analogWrite(9,0);
  delay(1000);
 analogWrite(11,0);
   analogWrite(10,230);
  analogWrite(9,0);
  delay(1000);

   analogWrite(11,0);
   analogWrite(10,0);
  analogWrite(9,0);
  delay(1000);
 analogWrite(11,0);
   analogWrite(10,0);
  analogWrite(9,80);
  delay(1000);
  analogWrite(11,0);
   analogWrite(10,0);
  analogWrite(9,150);
  delay(1000);
 analogWrite(11,0);
   analogWrite(10,0);
  analogWrite(9,230);
  delay(1000);

 delay(10);   }
```

Buzzer

A **buzzer** is a small yet efficient component to add sound features to our project/system. It is very small and compact 2-pin structure hence can be easily used on breadboard, Perf Board and even on PCBs which makes this a widely used component in most electronic applications.

There are two types are buzzers that are commonly available. The one shown here is a simple buzzer which when powered will make a Continuous Beeeeeeppp.... sound, the other type is called a readymade buzzer which will look bulkier than this and will produce a Beep. Beep. Beep. Sound due to the internal oscillating circuit present inside it. But, the one shown here is most widely used because it can be customised with help of other circuits to fit easily in our application.

This buzzer can be used by simply powering it using a DC power supply ranging from 4V to 9V. A simple 9V battery can also be used, but it is recommended to use a regulated +5V or +6V DC supply. The buzzer is normally associated with a switching circuit to turn ON or turn OFF the buzzer at required time and require interval.

Buzzer Pin Configuration

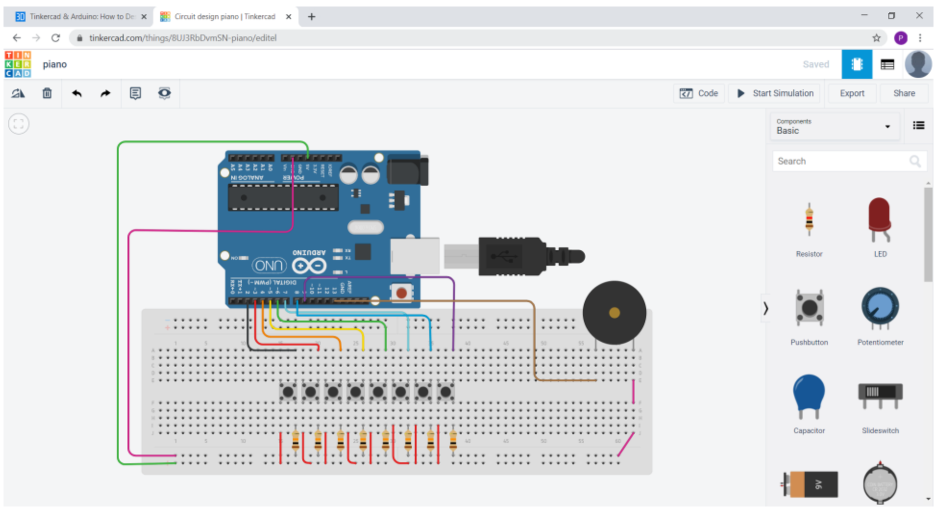
Pin Number	Pin Name	Description
1	Positive	Identified by (+) symbol or longer terminal lead. Can be powered by 6V DC
2	Negative	Identified by short terminal lead. Typically connected to the ground of the circuit

Buzzer Features and Specifications

- Rated Voltage: 6V DC
- Operating Voltage: 4-8V DC
- Rated current: <30mA
- Sound Type: Continuous Beep
- Resonant Frequency: ~2300 Hz
- Small and neat sealed package
- Breadboard and Perf board friendly

3. Build the circuit piano using switch button .

In this circuit to need 8 switch button ,8 resister (10K ohm),piezo ,Arduino , breadboard . the circuit connected piezo is output and switch is input to the Arduino. Then after connecting circuit to made program .in program to set the value tone to output the piezo . program run successfully then click simulate circuit. So one button press one tone is heared the anther button press then another tone heared and in this way so on the piano circuit builded



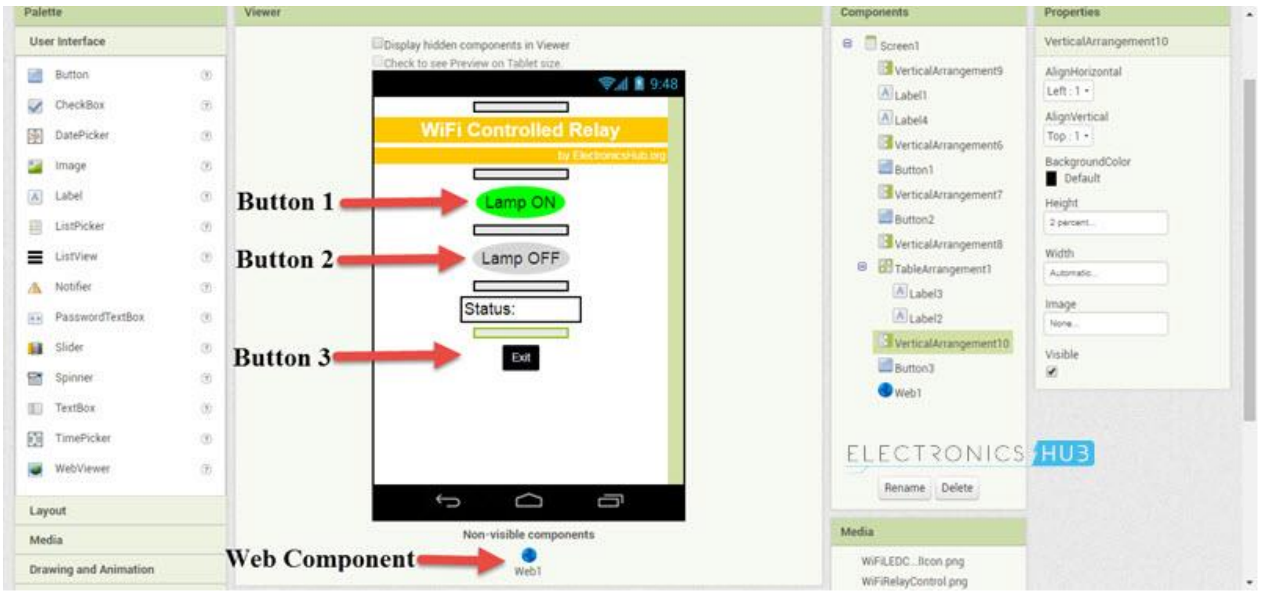
Program this circuit given below .

```
int but1 = 2;
int but2 = 3;
int but3 = 4;
int but4 = 5;
int but5 = 6;
int but6 = 7;
int but7 = 8;
int but8 = 9;
int buzzer = 13;

void setup()
{
  //let's declare the button pins as input
  pinMode(but1,INPUT);
  pinMode(but2,INPUT);
  pinMode(but3,INPUT);
  pinMode(but4,INPUT);
  pinMode(but5,INPUT);
  pinMode(but6,INPUT);
  pinMode(but7,INPUT);
  pinMode(but8,INPUT);
  //declare buzzer pin as output
  pinMode(buzzer,OUTPUT);

}

void loop()
{
  // read the value from buttons
  int b1 = digitalRead(but1);
  int b2 = digitalRead(but2);
  int b3 = digitalRead(but3);
  int b4 = digitalRead(but4);
  int b5 = digitalRead(but5);
  int b6 = digitalRead(but6);
  int b7 = digitalRead(but7);
  int b8 = digitalRead(but8);

  if( b1 == 1 ){
     tone(buzzer,300,100);
  }
    if( b2 == 1 ){
     tone(buzzer,400,100);
  }
    if( b3 == 1 ){
     tone(buzzer,500,100);
  }
    if( b4 == 1 ){
     tone(buzzer,600,100);
  }
    if( b5 == 1 ){

     tone(buzzer,700,100);
  }
    if( b6 == 1 ){
     tone(buzzer,800,100);
  }
    if( b7 == 1 ){
     tone(buzzer,900,100);
  }
    if( b8 == 1 ){
     tone(buzzer,1000,100);
  }

// now let's put a short delay for a nice pitch
  delay(10);
}
```

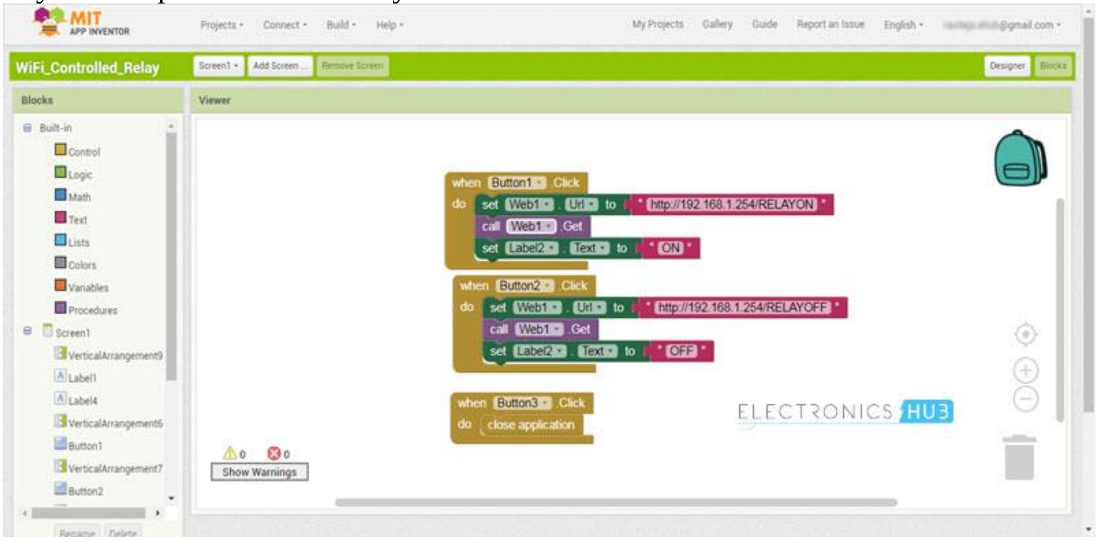
Introduction to MIT App Inventor:

MIT App Inventor is a web application integrated development environment originally provided by Google, and now maintained by the Massachusetts Institute of Technology (MIT). It allows newcomers to computer programming to create application software(apps) for two operating systems (OS): Android, and iOS, which, as of 8 July 2019, is in final beta testing. It is free and open-source software released under dual licensing: a Creative Commons Attribution ShareAlike 3.0 Unported license, and an Apache License 2.0 for the source code.
App Inventor is a cloud-based tool, by which you can build android applications using web browser. This website offers all the support you'll need to learn how to build your own apps. MIT App Inventor is an online platform designed to teach computational thinking concepts through development of mobile applications. Students create applications by dragging and dropping components into a design view and using a visual blocks language to program application behavior.

Eg:-It consists of three Buttons, few Labels and a Web Component.
In the Blocks section, create blocks like shown in the following image. The Static IP Address, which you have assigned in the code, must be entered here in the URL section of the block.



You do not have to create the exact same App. You can finish off with a minimalistic design and interface. After completing the Blocks section as well, you can debug the App directly from the Browser and an Android Phone without actually installing the App. For this, you have to download and install two Applications: one in your computer and other in your Android Phone.



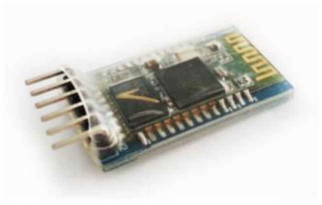
```
when Button1 .Click
do  set Web1 . Url to " http://192.168.1.254/RELAYON "
    call Web1 .Get
    set Label2 . Text to " ON "

when Button2 .Click
do  set Web1 . Url to " http://192.168.1.254/RELAYOFF "
    call Web1 .Get
    set Label2 . Text to " OFF "

when Button3 .Click
do  close application
```

The software for the Computer is called as MIT_Appinventor_Tools and the App for the Android phone is called MIT AI2 Companion.
I will not go into the details of this, but if you want further information, you can find it here.
Finally, after you are done with the interface, blocks and debugging (if any) of the App, you can download the .apk file to your computer and install it on to your device (Android Phone).

Introduction of Bluetooth module (HC 0 5)

The Bluetooth technology manages the **communication** channel of the wireless part. The Bluetooth modules can transmit and receives the data wirelessly by using two devices. The Bluetooth module can receive and transmits the data from a host system with the help of the host controller interface (HCI).

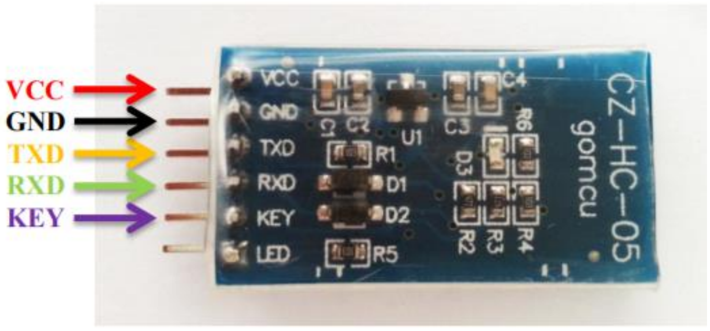
HC-05 Bluetooth Module



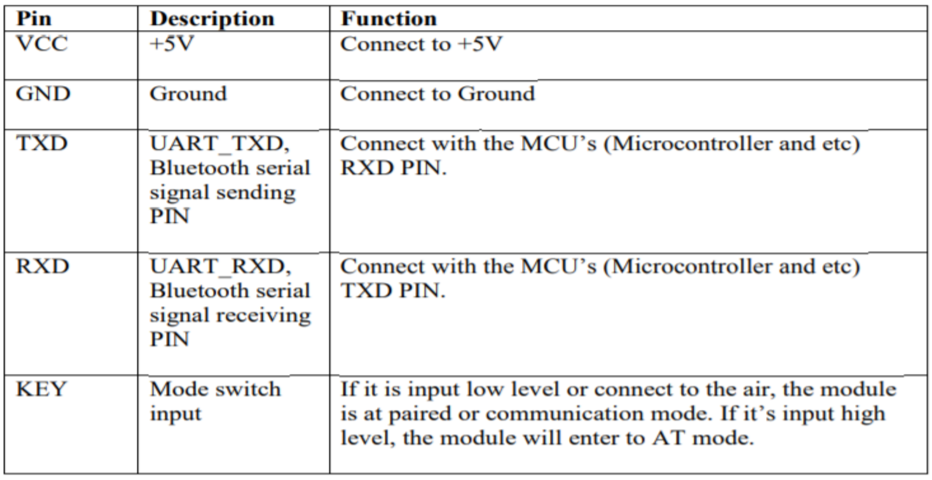
HC-05 Bluetooth Module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup. Its communication is via serial communication which makes an easy way to interface with controller or PC. HC-05 Bluetooth module provides switching mode between master and slave mode which means it able to use neither receiving nor transmitting data.

Specification:

- Model: HC-05
- Input Voltage: DC 5V
- Communication Method: Serial Communication
- Master and slave mode can be switched



Pin Definition



Pin	Description	Function
VCC	+5V	Connect to +5V
GND	Ground	Connect to Ground
TXD	UART_TXD, Bluetooth serial signal sending PIN	Connect with the MCU's (Microcontroller and etc) RXD PIN.
RXD	UART_RXD, Bluetooth serial signal receiving PIN	Connect with the MCU's (Microcontroller and etc) TXD PIN.
KEY	Mode switch input	If it is input low level or connect to the air, the module is at paired or communication mode. If it's input high level, the module will enter to AT mode.

WEEK 4

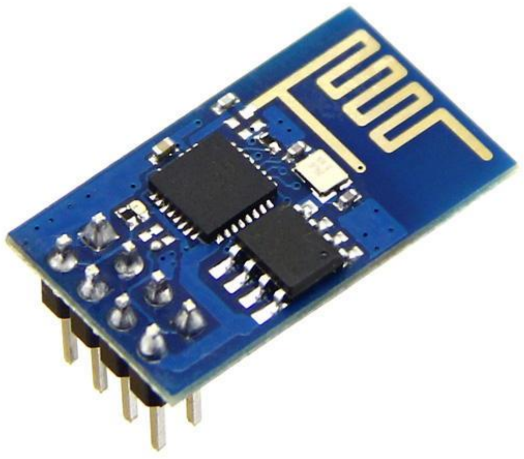
(22/6/2020 To 27/6/2020)

Topics of the week

1. Introduction to wifi module .
2. Light control to wifi module using MIT app inverter .
3. Introduction to cloud IOT.
4. Light control use cloud (**cayenne website.)**

Introduction to wifi module .

ESP8266 is Wi-Fi enabled system on chip (SoC) module developed by Espressif system. It is mostly used for development of IoT (Internet of Things) embedded applications.



ESP8266-01 WiFi Module

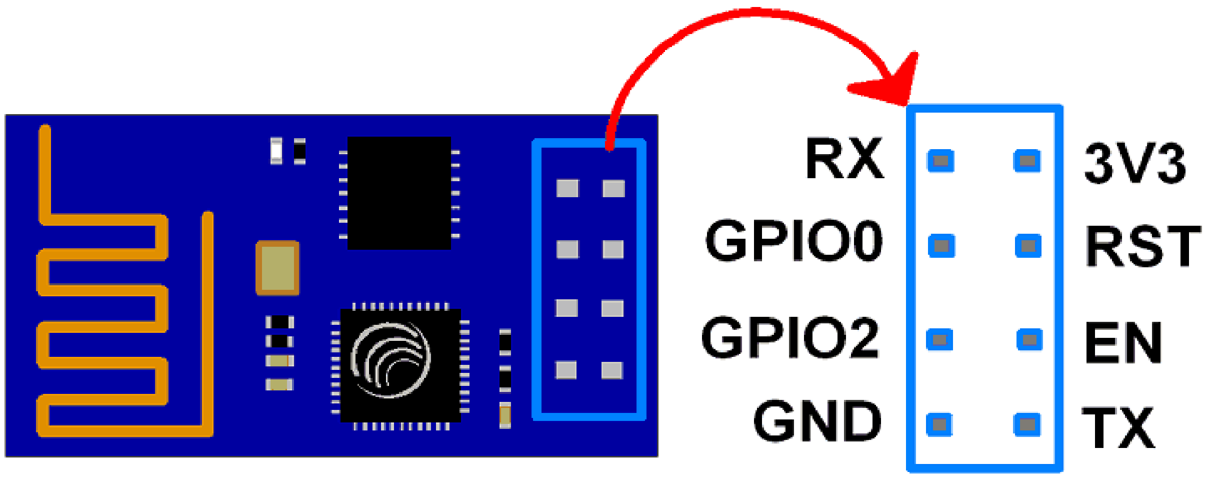
ESP8266 comes with capabilities of

- 2.4 GHz Wi-Fi (802.11 b/g/n, supporting WPA/WPA2),
- general-purpose input/output (16 GPIO),
- Inter-Integrated Circuit (I²C) serial communication protocol,
- analog-to-digital conversion (10-bit ADC)
- Serial Peripheral Interface (SPI) serial communication protocol,
- I²S (Inter-IC Sound) interfaces with DMA(Direct Memory Access) (sharing pins with GPIO),
- UART (on dedicated pins, plus a transmit-only UART can be enabled on GPIO2), and
- pulse-width modulation (PWM).

It employs a 32-bit RISC CPU based on the Tensilica Xtensa L106 running at 80 MHz (or overclocked to 160 MHz). It has a 64 KB boot ROM, 64 KB instruction RAM and 96 KB data RAM. External flash memory can be accessed through SPI.
ESP8266 module is low cost standalone wireless transceiver that can be used for end-point IoT developments.

To communicate with the ESP8266 module, microcontroller needs to use set of AT commands. Microcontroller communicates with ESP8266-01 module using UART having specified Baud rate.

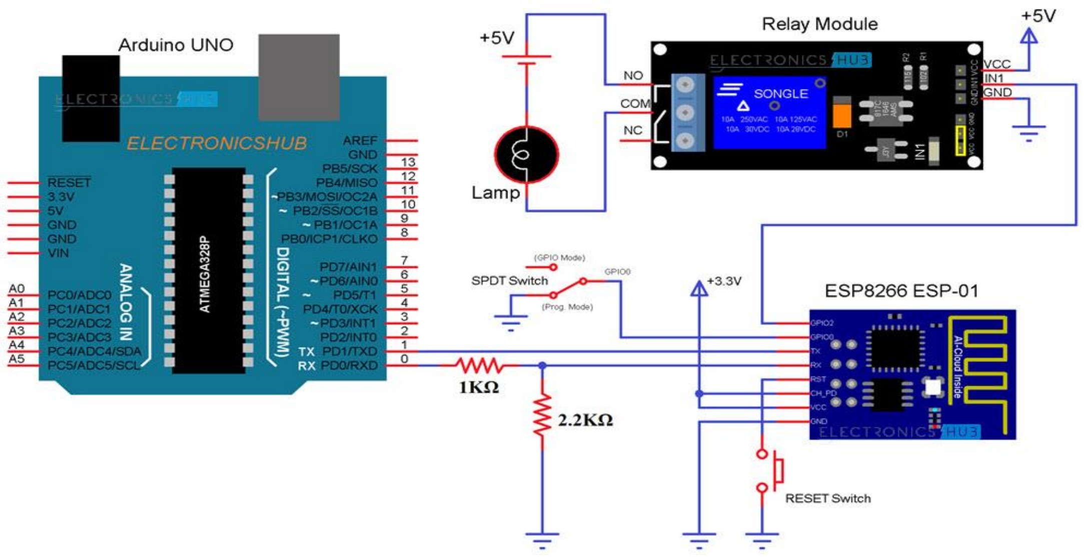
ESP8266-01 Module Pin Description



3V3: - 3.3 V Power Pin.
GND: - Ground Pin.
RST: - Active Low Reset Pin.
EN: - Active High Enable Pin.
TX: - Serial Transmit Pin of UART.
RX: - Serial Receive Pin of UART.
GPIO0 & GPIO2: - General Purpose I/O Pins. These pins decide what mode (boot or normal) the module starts up in. It also decides whether the TX/RX pins are used for Programming the module or for serial I/O purpose.
Applications of WiFi module:

- Serial port (RS232/RS485) to WiFi, TTL to WiFi;
- WiFi monitoring, TCP/IP and WIFI coprocessor;
- WiFi remote control aircraft, cars and other toys;
- WiFi network radio, camera, digital photo frame;
- Medical devices, data acquisition and handheld devices;
- WiFi fat scale, smart card terminal;Intelligent home;
- Instrument, equipment parameter monitoring, wireless POS machine;

Light operated to wifi modulator using MIT app inverter



Components Required

- ESP8266
- Arduino UNO
- Resistors (1KΩ and 2.2KΩ) – both are ¼ Watt Resistors
- Jumper Wires
- Relay Module
- Small 5V Bulb
- Push Button
- SPDT Switch
- Android App
- Android Phone
- Computer with Internet

Code

```
#include<ESP8266WiFi.h>

const char* ssid = "SSID";//type your ssid
const char* password = "PASSWORD";//type your password

int relayPin = 2; // GPIO2 of ESP8266
WiFiServer ESPserver(80);//Service Port

void setup()
{
Serial.begin(115200);
pinMode(relayPin, OUTPUT);
digitalWrite(relayPin, HIGH);

Serial.println();
Serial.println();
Serial.print("Connecting to: ");
Serial.println(ssid);

WiFi.begin(ssid, password);
delay(5000);


IPAddress ip(192,168,1,254);
IPAddress gateway(192,168,1,1);
IPAddress subnet(255,255,255,0);
```

```
WiFi.config(ip, gateway, subnet);
delay(5000);

while (WiFi.status() != WL_CONNECTED)
{
delay(100);
Serial.print("*");
}
Serial.println("");
Serial.println("WiFi connected");

// Start the server
ESPserver.begin();
Serial.println("Server started");

// Print the IP address
Serial.print("The URL to control ESP8266: ");
Serial.print("http://");
Serial.print(WiFi.localIP());
}

void loop()
{
// Check if a client has connected
WiFiClient client = ESPserver.available();
if (!client)
{
return;
}

// Wait until the client sends some data
Serial.println("New Client");
while(!client.available())
{
delay(1);
}

// Read the first line of the request
String request = client.readStringUntil('\r');
Serial.println(request);
client.flush();

// Match the request

int value = LOW;
if (request.indexOf("/RELAYON") != -1)
{
Serial.println("LAMP is ON");
digitalWrite(relayPin, LOW);
value = LOW;
}
if (request.indexOf("/RELAYOFF") != -1)
{
Serial.println("LAMP is OFF");
digitalWrite(relayPin, HIGH);
value = HIGH;
}

// Return the response
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
client.println(""); //  IMPORTANT
client.println("<!DOCTYPE HTML>");
client.println("<html>");

client.print("Status of the Lamp: ");

if(value == LOW)
{
client.print("ON");
}
else
{
client.print("OFF");
}

delay(1);
//client.stop();
Serial.println("Client disconnected");
Serial.println("");
}
```

Working

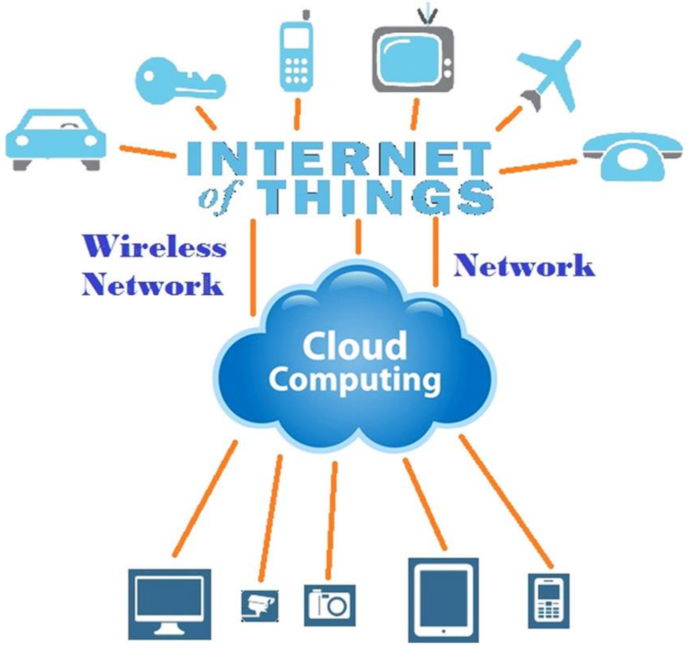
First, make all the necessary connections as per the circuit diagram and upload the program to the ESP8266 WiFi Module.
After the program is uploaded, you will get a confirmation message regarding WiFi connection and Static IP Address. Now open the Android App which we developed using MIT App Inventor 2 and installed it on your Android Phone.
If everything goes well, when you hit the “Lamp ON” button on the App, the Relay gets a Logic LOW signal and the Lamp is turned ON. Similarly, when the “Lamp OFF” button is pressed, the Lamp turn off.
The software for the Computer is called as MIT_Appinventor_Tools and the App for the Android phone is called MIT AI2 Companion.
I will not go into the details of this, but if you want further information, you can find it here.
Finally, after you are done with the interface, blocks and debugging (if any) of the App, you can download the .apk file to your computer and install it on to your device (Android Phone).

Applications

- In this project, we have seen how to control a relay using ESP8266 through WiFi with the help of an Android App developed using MIT App Inventor 2 Application.
- Such projects can be the stepping stone for complex home automation system, where the maker can not only assemble the circuit but also make his own Android Application.
- The next big step of this project would be controlling the relay from anywhere in the World i.e. a true Web Controlled Relay.

Introduction to cloud IOT.

Cloud IoT is a complete set of tools to connect, process, store, and analyze data both at the edge and in the cloud. The platform consists of scalable, fully-managed cloud services; an integrated software stack for edge/on-premises computing with machine learning capabilities for all your IoT needs.



Introduction of cayenne website.

Cayenne is the first of its kind drag and drop IoT project builder that empowers developers to quickly create and host their connected device projects. Cayenne was designed for the Internet of Things. It can control hardware remotely, it can display sensor data, it can store data, analyze and do many other cool things.

Features:

- Connection using Ethernet, Wi-Fi and cellular (mobile app only)
- Discover and setup Raspberry Pis on a network (Ethernet or Wi-Fi only)
- Customizable dashboard with drag and drop widgets
- Remotely access, reboot and shutdown a Pi
- Add and control sensors, actuators and extensions connected to Raspberry Pis
- Configure triggers for Pis, sensors and actuators
- Setup and receive threshold alerts via email and text messages
- Monitor device and sensor history data
- Remotely test and and configure hardware using GPIO
- Coming soon! Setup reoccurring actions and commands

This guide will help you get started using Cayenne in minutes. We will quickly cover the following:

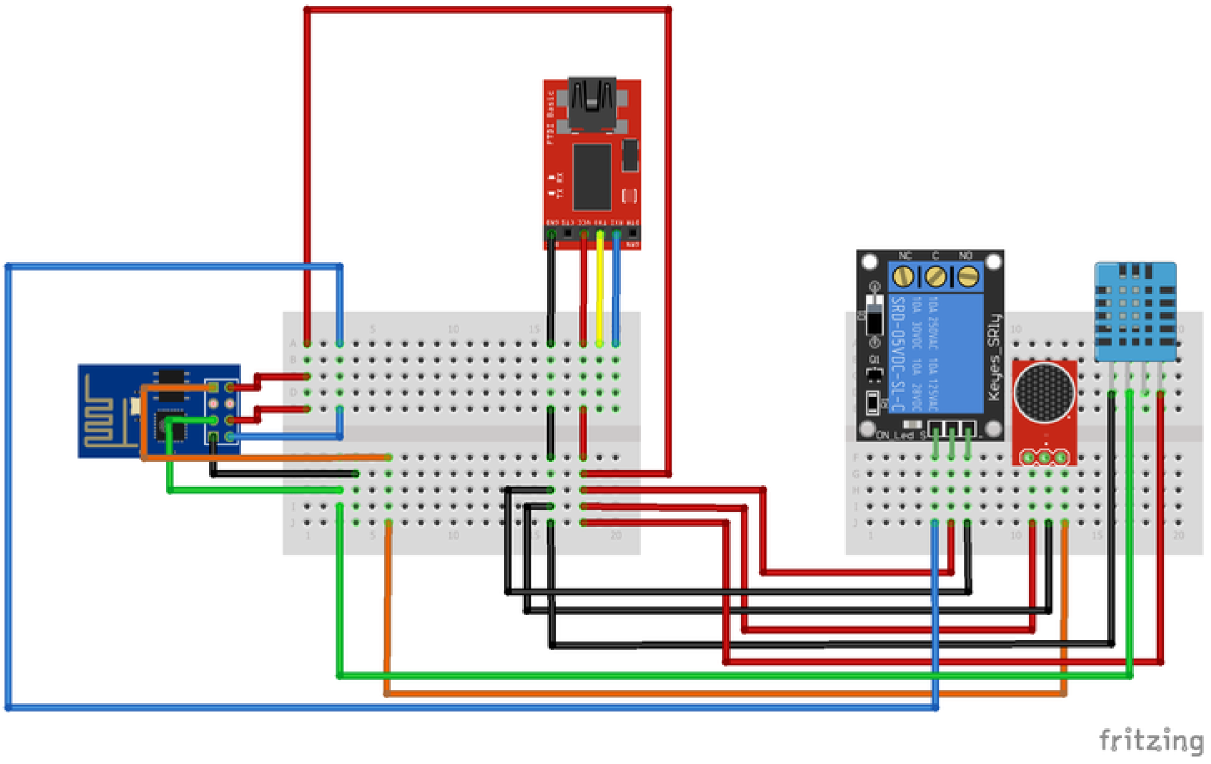
- Creating your account
- Installing and setting up Cayenne using a terminal
- Configuring your first sensor (Temperature)
- Configure your first actuator (LED Switch)

- Setting up a trigger

Light control use cloud (cayenne website.)

Components use:-

1. ESP8266 ESP-01 WiFi Module
2. FTDI Breakout Board + Mini USB cable
3. ESP8266 ESP-01 Breakout Board (Optional)
4. 1-Channel Relay Module
5. DHT11 Temperature and Humidity Sensor
6. Sound Sensor
7. Breadboards (I will be using 2 small breadboards)
8. 5V to 3.3V Stepdown Buck Module (Optional)
9. Plenty of jumper wires

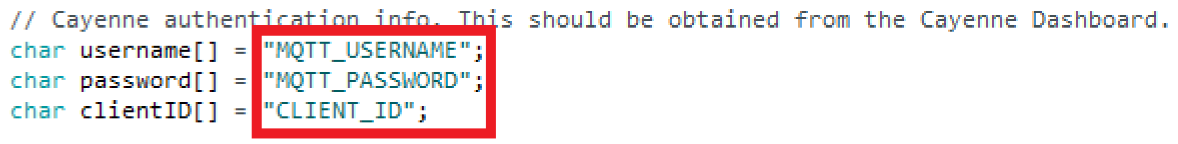


All working instruction to run this circuit:-

1. Log on to your Cayenne account (register for FREE if you haven't one yet at **Cayenne Signup Page**): https://accounts.mydevices.com/auth/realms/cayenne/protocol/openid-connect/auth?response_type=code&scope=email+profile&client_id=cayenne-web-app&state=kdHCakZa1p3nDxsps3bkMin5sbPypkxhCmGOM8jz&redirect_uri=https%3A%2F%2Fcayenne.mydevices.com%2Fauth%2Fcallback
2. Once logged in, the next step would be to choose the device we are going to use. If this is your first device in your dashboard, choose **All Devices**
    - choose Generic ESP8266 under Microcontrollers

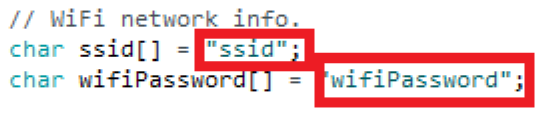
- already have existing devices added in previously, you need to add a device by clicking the Add Devices/Widget menu.
- Then choose ESP8266 since this is our development board

3. In the succeeding page, you are shown with the MQTT Username, MQTT Password, and Client ID. We need these details in the sketch later. You will also see at the bottom portion of the page that it is waiting for the device to connect.
4. Now, we proceed with the Sketch. Open the Arduino IDE. Go to File > Examples > Cayenne-MQTT-ESP8266 > **ESP8266**. This sketch will be our base sketch.
5. Go back to the Cayenne Portal. Take note of the following, because we are going to use it in our sketch:
   a. **MQTT USERNAME**
   b. **MQTT PASSWORD**
   c. **CLIENT ID**
6. Now, go back to the Arduino IDE. Paste the values in the section highlighted in the screenshot below:



```
// Cayenne authentication info. This should be obtained from the Cayenne Dashboard.
char username[] = "MQTT_USERNAME";
char password[] = "MQTT_PASSWORD";
char clientID[] = "CLIENT_ID";
```

7. Also, supply the WiFi name (SSID) and password in this section of the code:



```
// WiFi network info.
char ssid[] = "ssid";
char wifiPassword[] = "wifiPassword";
```

8. By the way, we need to include the needed libraries: **SimpleDHT** and **SimpleTimer**. The SimpleDHT will allow us to use the DHT Temperature. The SimpleTimer will allow us to execute methods outside the loop() method:

```
#include <SimpleTimer.h>          // Download from https://github.com/jfturcot/SimpleTimer
#include <SimpleDHT.h>            // Download from https://github.com/adafruit/DHT-sensor-library
```

9. We now need to declare the pins to be used by the sensors.
   a. DHT11 Pin = Pin 2 (GPIO2 of ESP8266 ESP-01)
   b. Sound Pin = Pin 3 (GPIO3 of ESP8266 ESP-01)
   c. Relay Pin = Pin 1 (GPIO1 of ESP8266 ESP-01)

```
// DHT11 Pin
int pinDHT11 = 2;
SimpleDHT11 dht11;

// Sound Pin
int soundPin = 3;

// Relay Pin
int relayPin = 1
```

10. Since we are going to send data to the Cayenne IoT platform using MQTT API, we need to pass the sensor value to the virtual pins. The virtual pins will be used by the widgets in our dashboard. We are

not going to use V1, because we assigned this as an output (to control the relay – used in the Cayenne dashboard widgets):

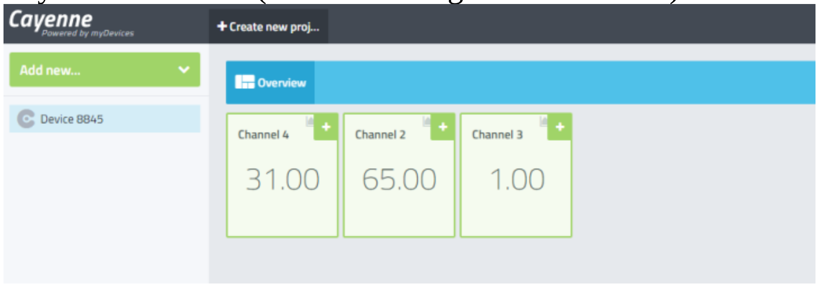
   a. V2 – Humidity (data from DHT11)
   b. V3 – Temperature (data from DHT11)
   c. V4 – Sound (data from Sound Sensor)

11. Upload the sketch to the ESP8266 ESP-01 Board. Be sure that the chosen board is **Generic ESP8266 Module**

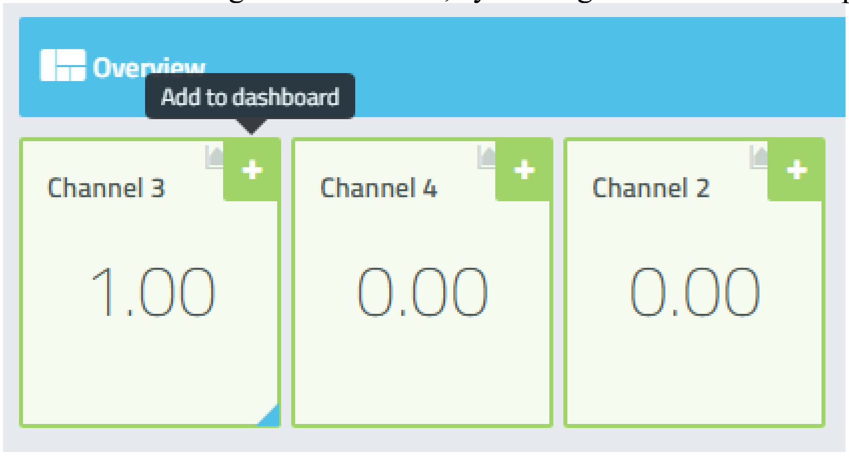
Complete Code

```
// This example shows how to connect to Cayenne using an ESP8266 and send/receive sample data.
// Make sure you install the ESP8266 Board Package via the Arduino IDE Board Manager and select the correct ESP8266
board before compiling.

//#define CAYENNE_DEBUG
#define CAYENNE_PRINT Serial
#include <CayenneMQTTESP8266.h>

#include <SimpleTimer.h> // Download from https://github.com/jfturcot/SimpleTimer
#include <SimpleDHT.h> // Download from https://github.com/adafruit/DHT-sensor-library

// WiFi network info.
char ssid[] = "<your ssid/wifi name>";
char wifiPassword[] = ""<your ssid/wifi password>"";

// Cayenne authentication info. This should be obtained from the Cayenne Dashboard.
char username[] = "<your MQTT Username>";
char password[] = "<your MQTT Password>";
char clientID[] = "<your client id>";

// DHT11 Pin
int pinDHT11 = 2;
SimpleDHT11 dht11;

// Sound Pin
int soundPin = 3;

// Relay Pin
int relayPin = 1;

// Timer
SimpleTimer timer;

void setup() {
 Serial.begin(9600);
 Cayenne.begin(username, password, clientID, ssid, wifiPassword);
 pinMode(relayPin, OUTPUT); // Relay
 digitalWrite(relayPin, HIGH);
 pinMode(soundPin, INPUT); // Sound
 timer.setInterval(200L, transmitData); // Method to execute every 200ms
}

void loop() {
 Cayenne.loop();
 timer.run();
}

CAYENNE_IN(relayPin) {
 if (getValue.asInt() == 1) { // NOTE: Channel = Virtual Pin
 digitalWrite(relayPin, LOW);
 }
 else {
 digitalWrite(relayPin, HIGH);
 }
}

void transmitData()
{
 byte temperature = 0;
 byte humidity = 0;
 int err = SimpleDHTErrSuccess;

if ((err = dht11.read(pinDHT11, &temperature, &humidity, NULL)) != SimpleDHTErrSuccess) {
 Cayenne.virtualWrite(V4, 0);
 Cayenne.virtualWrite(V2, 0);
 }
 else {
 Cayenne.virtualWrite(V4, (int)temperature);
 Cayenne.virtualWrite(V2, (int)humidity);
 }

if (digitalRead(soundPin) == HIGH) {
 Cayenne.virtualWrite(V3, HIGH);
 }
 else {
 Cayenne.virtualWrite(V3, LOW);
 }
}
```

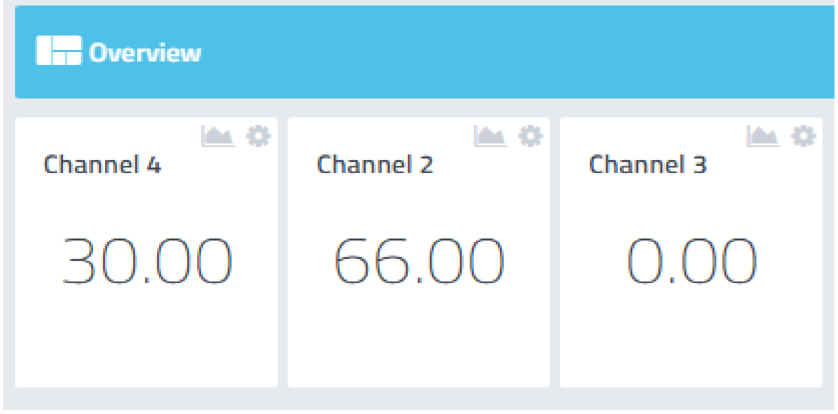
Initially, this is how the dashboard looks like when the ESP8266 ESP-01 WiFi module starts to send data to the Cayenne Dashboard (under the configured device there):



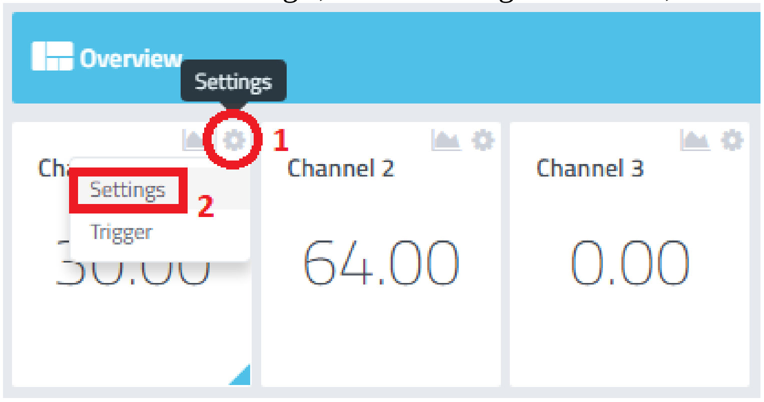
Add the initial widgets shown above, by clicking the "+" icon at the upper right hand corner of each initial widget:



Once added, we will now customize each widget. This is how it looks like when these initial widgets are added:

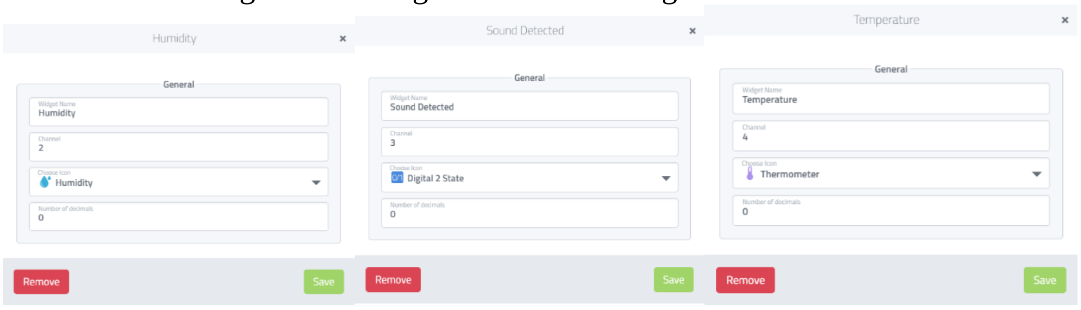


To customize each widget, click on the right-most icon, then select Settings. Refer to screenshot below:



A pop-up window appears and will allow us to change the Widget Name, Channel, Widget Icon, and Number of Decimal Places. We will leave **Channel** as it is. Don't change the value of Channel, since the value here is the **Virtual Pin (V0, V1, V2…)** used in the code for this widget. Virtual Pin 4 (V4) is the temperature, Virtual Pin 2 (V2) is the humidity, and Virtual Pin 3 (V3) is the value of the sound sensor (0/1).

Below are the configuration/setting for each initial widget:


Humidity

General

Widget Name: Humidity
Channel: 2
Choose Icon: Humidity
Number of decimals: 0

Remove Save

Sound Detected

General

Widget Name: Sound Detected
Channel: 3
Choose Icon: Digital 2 State
Number of decimals: 0

Remove Save

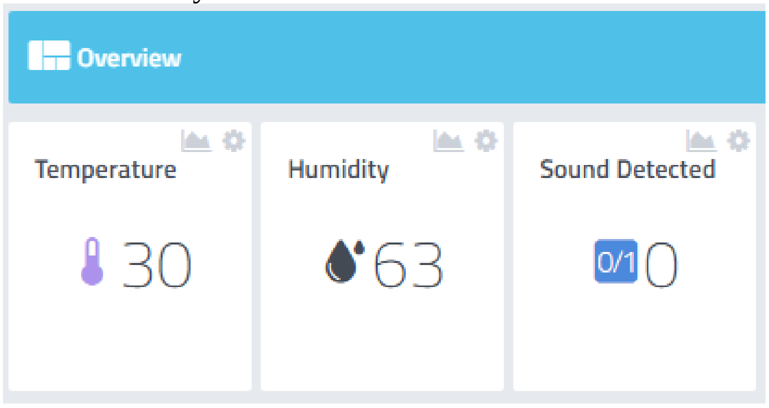
Temperature

General

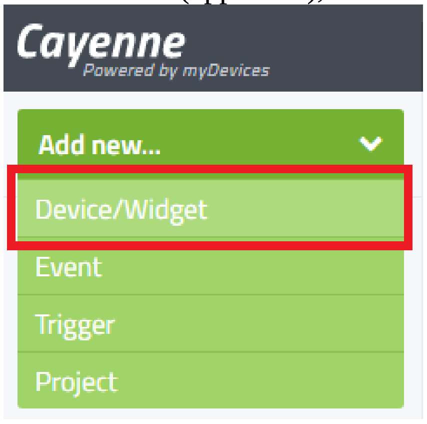
Widget Name: Temperature
Channel: 4
Choose Icon: Thermometer
Number of decimals: 0

Remove Save

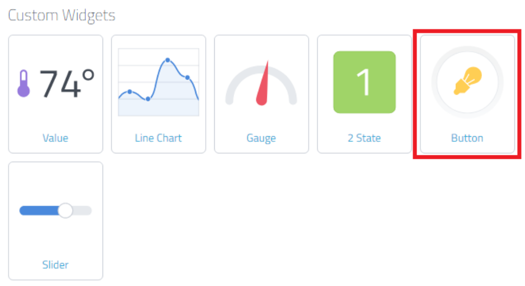
Here is how they look afterward:



Now, we still need to add one more widget. This widget is for the relay, to turn it on/off. On the left side portion of the dashboard (upper-left), click **Add New**, then click **Device/Widget**:



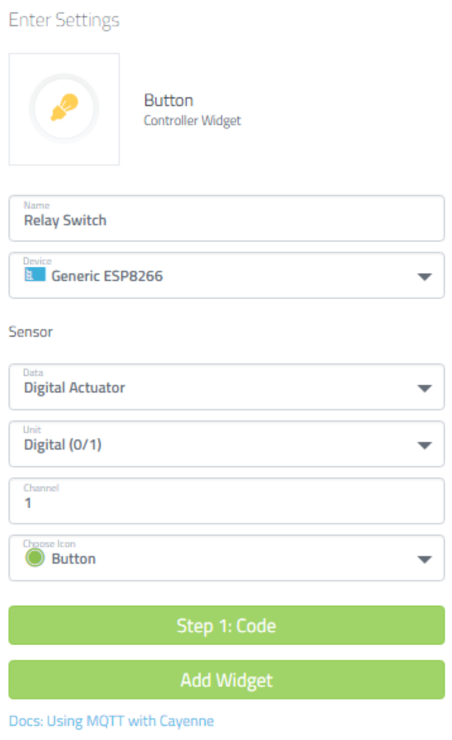
Scroll down further down, and look for **Custom Widgets**. Select **Button**:



We need to change the settings for the Button widget:

- Name = Relay Switch
- Device = Generic ESP8266 (or your named device if you have already)
- Data = Digital Actuator
- Unit = Digital (0/1)
- Channel = 1 (this is GPIO1 of the ESP8266 ESP-01 WiFi Module – review code)
- Choose Icon = Button (you may choose what icon you desire)

Below is the screenshot of the Button settings. Click Add Widget afterwards:



Enter Settings

Button
Controller Widget

Name: Relay Switch

Device: Generic ESP8266

Sensor

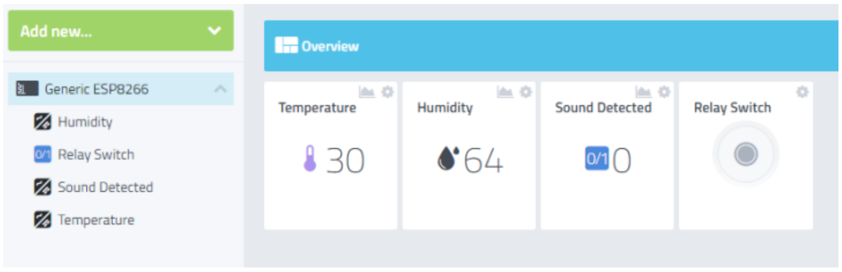
Data: Digital Actuator

Unit: Digital (0/1)

Channel: 1

Choose Icon: Button

The final dashboard now looks like this:



Temperature 30

Humidity 64

Sound Detected 0/1 0

Relay Switch

And we're DONE! Now, try to switch on/off the Relay Switch. The relay connected to the ESP8266 ESP-01 WiFi Module will turn on or off – you will hear clicking sounds. Try to make some noise. The sound widget will register 1 (if noise is detected, 0 if not). Try observing the Temperature and Humidity widgets – these values change according to what is "sensed" from where the DHT11 module is.

References-:

1. www.tinkercad.com
2. www.arduino.com
3. https://www.electronicwings.com/sensors-modules/esp8266-wifi-module
4. https://appinventor.mit.edu/
5. https://en.scratch-wiki.info/wiki/Scratch_Blocks#:~:text=Scratch%20Blocks%20is%20a%20library,wide%20range%20of%20other%20objects.
6. https://www.makerspaces.com/basic-electronics/