B K Birla Institute of Engineering & Technology, Pilani

Laboratory Manual

7CS4-01 Internet of Things
(IoT)

For

Final year Students (CSE)

FOREWORD

It is my great pleasure to present this laboratory manual for Final year engineering students for the subject of 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.

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.

B K Birla Institute of Engineering & Technology, Pilani

Department of Computer Science and Engineering

Vision of CSE Department:

To produce excellent, innovative, self-motivated engineering entrepreneurs who can sustain in rapidly changing field of applied computational technology in various industries and be among the most highly reputed technology-focused learning institutions in the world.

Mission of the CSE Department:

In order to realize its vision, the Department of Computer Science and Engineering, BKBIET, Pilani will undertake the following mission:

1. Create conducive environment for learning and research
2. Establish industry and academia collaborations
3. Ensure professional and ethical values in all institutional endeavors.

Programme Outcomes (POs):

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

LAB INDEX

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

1. Study and Install IDE of Arduino and different types of Arduino.
2. Write program using Arduino IDE for Blink LED.
3. Write Program for RGB LED using Arduino.
4. Study the Temperature sensor and Write Program foe monitor temperature using Arduino.
5. Study and Implement RFID, NFC using Arduino.
6. Study and implement MQTT protocol using Arduino.
7. Study and Configure Raspberry Pi.
8. WAP for LED blink using Raspberry Pi.

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. Do not change the terminal on which you are working.
4. All the students are expected to get at least the algorithm of the program/concept to be implement.
5. Strictly follow the instructions given by the teacher/Lab Instructor.

Instruction for Laboratory Teachers

1. Submission related to whatever lab work has been completed should be done during the next lab session.
2. The immediate arrangements for printouts related to submission on the day of practical assignments.
3. Students should be taught for taking the printouts under the observation of lab teacher.
4. The promptness of submission should be encouraged by way of marking and evaluation patterns that will benefit the sincere students.

Experiment No 1

Aim: Study and Install IDE of Arduino and different types of Arduino

Objectives: Student should get the knowledge of Arduino IDE and different types of Arduino Board

Outcomes: Student will be get knowledge of Arduino IDE and different types of Arduino Board

Arduino:

Arduino is a prototype platform (open-source) based on an easy-to-use hardware and software. It consists of a circuit board, which can be programed (referred to as a microcontroller) and a ready-made software called Arduino IDE (Integrated Development Environment), which is used to write and upload the computer code to the physical board.

Arduino provides a standard form factor that breaks the functions of the micro- controller into a more accessible package.

Arduino is a prototype platform (open-source) based on an easy-to-use hardware and software. It consists of a circuit board, which can be programed (referred to as a microcontroller) and a ready-made software called Arduino IDE (Integrated Development Environment), which is used to write and upload the computer code to the physical board.

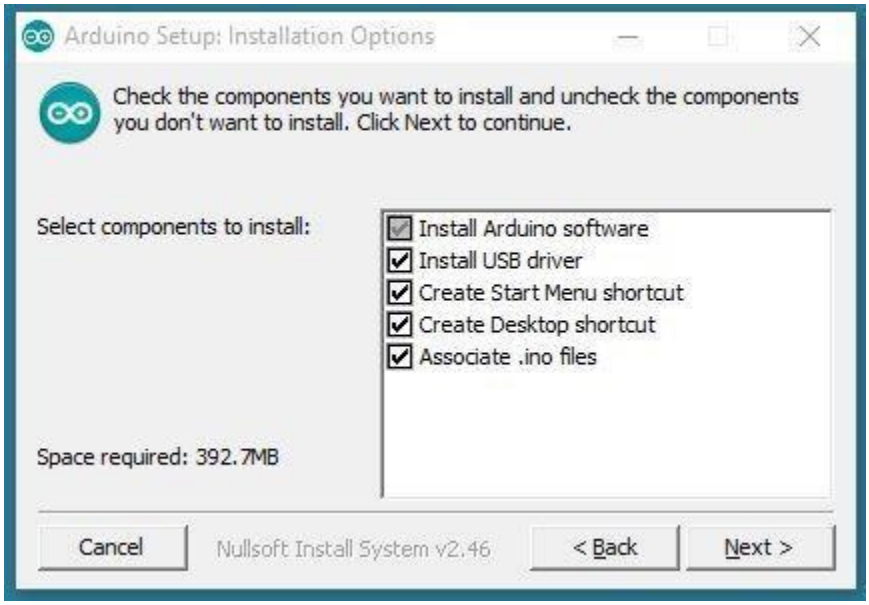
The key features are –

- 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.
- 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).
- 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.

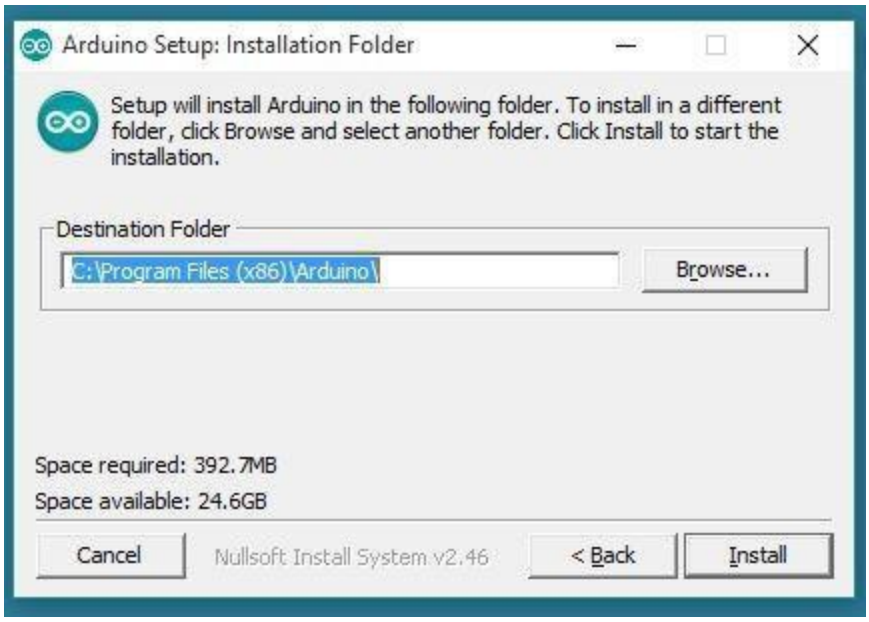
- Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program.
- Finally, Arduino provides a standard form factor that breaks the functions of the micro-controller into a more accessible package.

Download the Arduino Software (IDE)

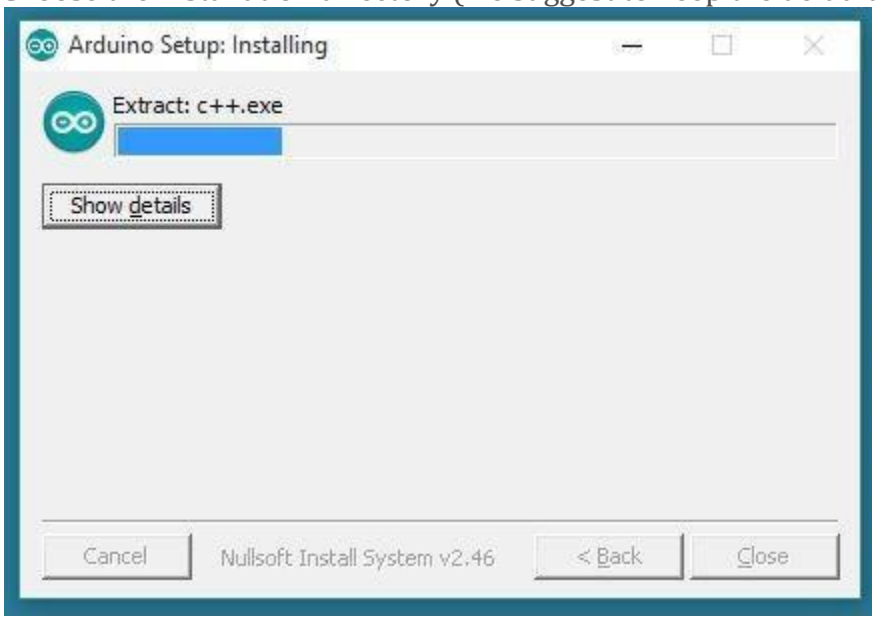
- Get the latest version from the arduino.cc web site. You can choose between the Installer (.exe) and the Zip packages. We suggest you use the first one that installs directly everything you need to use the Arduino Software (IDE), including the drivers. With the Zip package you need to install the drivers manually. The Zip file is also useful if you want to create a portable installation.
- When the download finishes, proceed with the installation and please allow the driver installation process when you get a warning from the operating system.



- Choose the components to install



- Choose the installation directory (we suggest to keep the default one)



- The process will extract and install all the required files to execute properly the Arduino Software
  (IDE)
- Proceed with board specific instructions
- When the Arduino Software (IDE) is properly installed you can go back to the

Different Arduino Boards:

Arduino USB

1.Arduino uno

This is the latest revision of the basic Arduino USB board. It connects to the computer with a standard USB cable and contains everything else you need to program and use the board.

2.Arduino NG REV-C

Revision C of the Arduino NG does not have a built-in LED on pin 13 - instead you'll see two small unused solder pads near the labels "GND" and "13".

Arduino Bluetooth

The Arduino BT is a microcontroller board originally was based on the ATmega168, but now is supplied with the 328, and the Bluegiga WT11 bluetooth module. It supports wireless serial communication over bluetooth.

Arduino Mega

The original Arduino Mega has an ATmega1280 and an FTDI USB-to- serial chip.

Arduino NANO

The Arduino Nano 3.0 has an ATmega328 and a two-layer PCB. The power LED moved to the top of the board.

Experiment No 2

Aim: Write program using Arduino IDE for Blink LED

Objectives: Student should get the knowledge of Arduino Board and different types of LED

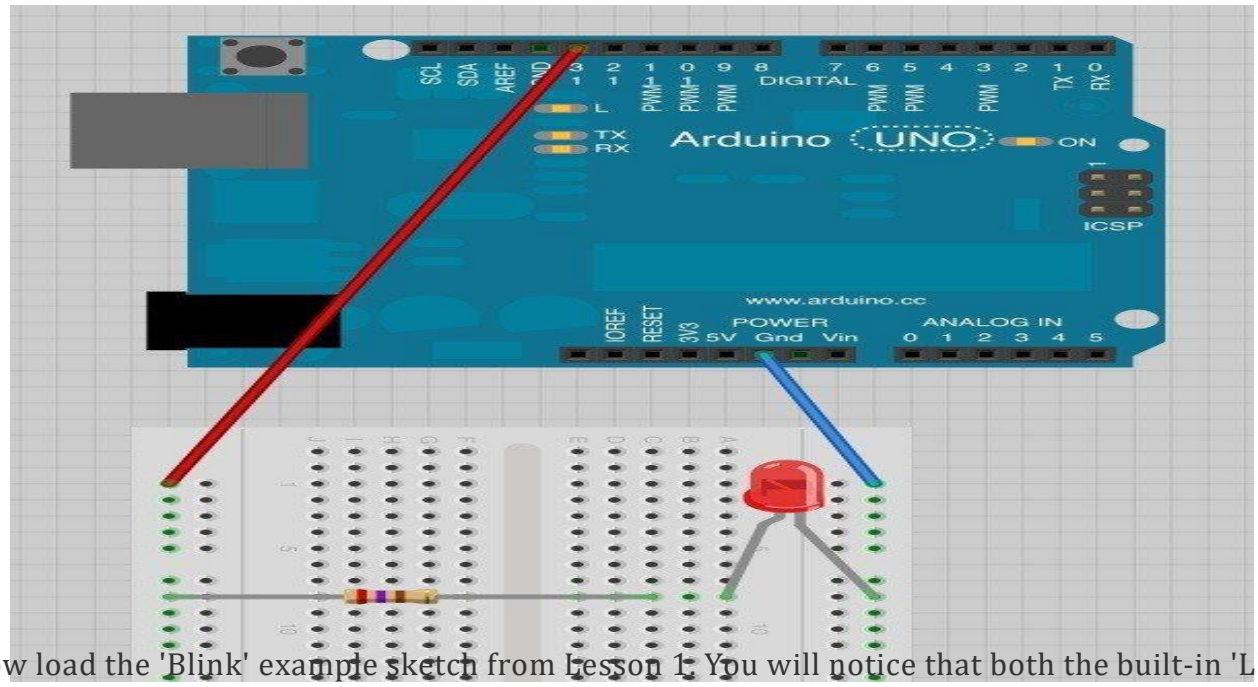
Outcomes: Student will be Write program using Arduino IDE for Blink LED

Hardware Requirements:

- 1x Breadboard
- 1x Arduino Uno R3
- 1x RGB LED
- 1x 330Ω Resistor
- 2x Jumper Wires

Blinking the RGB LED

With a simple modification of the breadboard, we could attach the LED to an output pin of the Arduino. Move the red jumper wire from the Arduino 5V connector to D13, as shown below:



Now load the 'Blink' example sketch from Lesson 1. You will notice that both the built-in 'L' LED and the external LED should now blink.

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
*/

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

Lets try using a different pin of the Arduino – say D7. Move the red jumper lead from pin D13 to pin D7 and modify the following line near the top of the sketch:

```
int led = 13;
```

so that it reads:

```
int led = 7;
```

Upload the modified sketch to your Arduino board and the LED should still be blinking, but this time using pin D7.

Experiment No 3

Aim: Write Program for RGB LED using Arduino.

Objectives: Student should get the knowledge of Arduino IDE and RGB Led

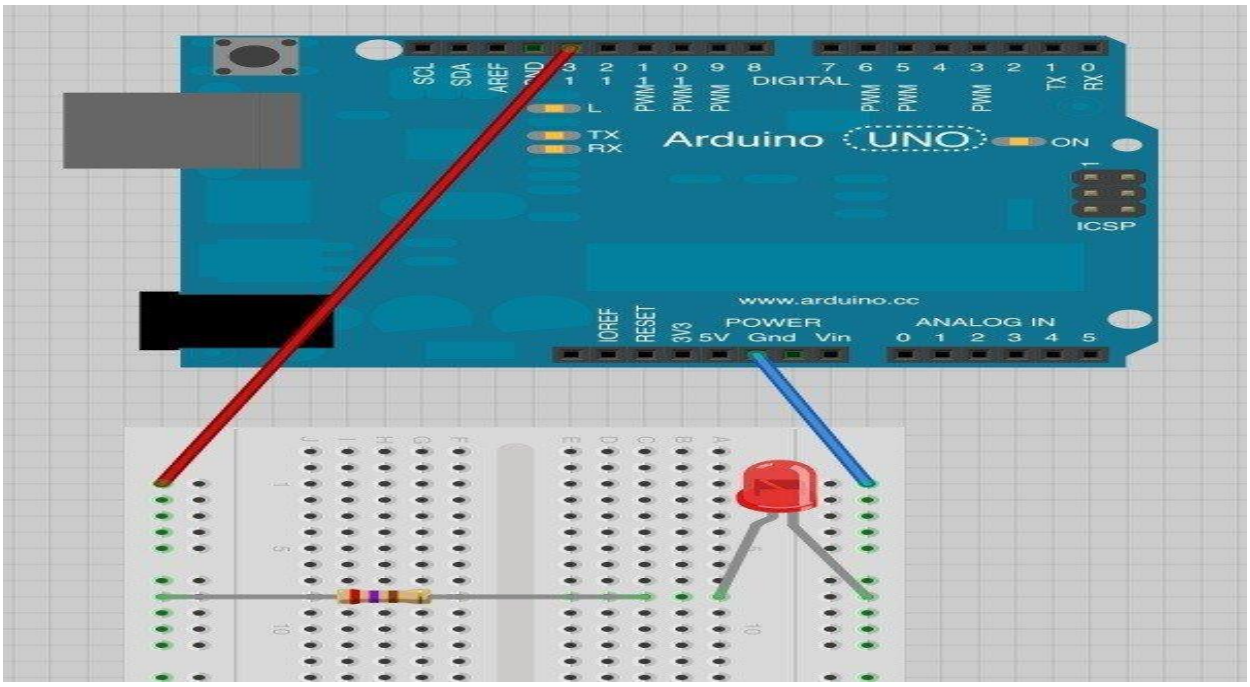
Outcomes: Student will be developed programs using Arduino IDE and Arduino Board for RGB Led

Hardware Requirements:

- 1x Breadboard
- 1x Arduino Uno R3
- 1x LED
- 1x 330Ω Resistor
- 2x Jumper Wires

Blinking the LED

With a simple modification of the breadboard, we could attach the LED to an output pin of the Arduino. Move the red jumper wire from the Arduino 5V connector to D13, as shown below:



Now load the 'Blink' example sketch from Lesson 1. You will notice that both the built-in 'L' LED and the external LED should now blink.

The following test sketch will cycle through the colors red, green, blue, yellow, purple, and aqua. These colors being some of the standard Internet colors.

```
/*
Adafruit Arduino - Lesson 3. RGB LED
*/

int redPin = 11;
int greenPin = 10;
int bluePin = 9;

//uncomment this line if using a Common Anode LED
//#define COMMON_ANODE

void setup()
{
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
}

void loop()
{
  setColor(255, 0, 0);  // red
  delay(1000);
  setColor(0, 255, 0);  // green
  delay(1000);
  setColor(0, 0, 255);  // blue
  delay(1000);
  setColor(255, 255, 0);  // yellow
  delay(1000);
  setColor(80, 0, 80);  // purple
  delay(1000);
  setColor(0, 255, 255);  // aqua
  delay(1000);
}

void setColor(int red, int green, int blue)
{
  #ifdef COMMON_ANODE
    red = 255 - red;
    green = 255 - green;
    blue = 255 - blue;
  #endif
  analogWrite(redPin, red);
  analogWrite(greenPin, green);
  analogWrite(bluePin, blue);
}
```

The sketch starts by specifying which pins are going to be used for each of the colors:

Download file

Copy Code

```
int redPin = 11;
int greenPin = 10;
int bluePin = 9;
```

The next step is to write the 'setup' function. As we have learnt in earlier lessons, the setup function runs just once after the Arduino has reset. In this case, all it has to do is define the three pins we are using as being outputs.

```
void setup()
{
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
}
```

Before we take a look at the 'loop' function, lets look at the last function in the sketch.

Download file

Copy Code

```
void setColor(int red, int green, int blue)
{
  analogWrite(redPin, red);
  analogWrite(greenPin, green);
  analogWrite(bluePin, blue);
}
```

This function takes three arguments, one for the brightness of the red, green and blue LEDs. In each case the number will be in the range 0 to 255, where 0 means off and 255 means maximum brightness. The function then calls 'analogWrite' to set the brightness of each LED.

If you look at the 'loop' function you can see that we are setting the amount of red, green and blue light that we want to display and then pausing for a second before moving on to the next color.

```
void loop()
{
  setColor(255, 0, 0);  // red
  delay(1000);
  setColor(0, 255, 0);  // green
  delay(1000);
  setColor(0, 0, 255);  // blue
  delay(1000);
  setColor(255, 255, 0);// yellow
  delay(1000);
```

```
11.  setColor(80, 0, 80);  // purple
12.  delay(1000);
13.  setColor(0, 255, 255);// aqua
14.  delay(1000);
15.}
```

Try adding a few colors of your own to the sketch and watch the effect on your LED.

Experiment No 4

Aim: Study the Temperature sensor and Write Program foe monitor temperature using Arduino.

Objectives: Student should get the knowledge of Temperature Sensor

Outcomes: Student will be developed programs using Arduino IDE and Arduino Board for Temperature Sensor

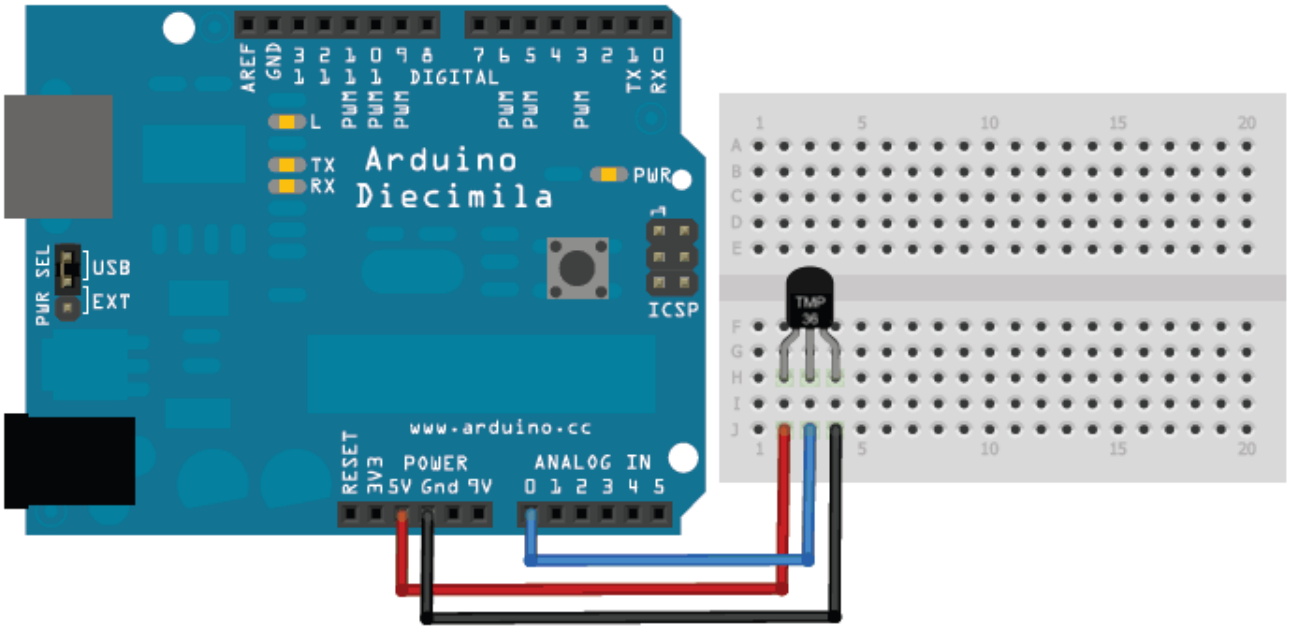
Connecting to a Temperature Sensor

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!

They come in a "TO-92" package which means the chip is housed in a plastic hemi-cylinder with three legs. The legs can be bent easily to allow the sensor to be plugged into a breadboard. You can also solder to the pins to connect long wires. If you need to waterproof the sensor, you can see below for an Instructable for how to make an excellent case.

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.



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

Simple Thermometer

This example code for Arduino shows a quick way to create a temperature sensor, it simply prints to the serial port what the current temperature is in both Celsius and Fahrenheit.

```
1. //TMP36 Pin Variables
2. int sensorPin = 0; //the analog pin the TMP36's Vout (sense) pin is connected to
3.                    //the resolution is 10 mV / degree centigrade with a
4.                    //500 mV offset to allow for negative temperatures
5.
6. /*
7.  * setup() - this function runs once when you turn your Arduino on
8.  * We initialize the serial connection with the computer
9.  */
10.void setup()
11.{
12.  Serial.begin(9600);   //Start the serial connection with the computer
13.                    //to view the result open the serial monitor
14.}
15.
16.void loop()                     // run over and over again
17.{
18. //getting the voltage reading from the temperature sensor
19. int reading = analogRead(sensorPin);
20.
21. // converting that reading to voltage, for 3.3v arduino use 3.3
22. float voltage = reading * 5.0;
23. voltage /= 1024.0;
24.
25. // print out the voltage
26. Serial.print(voltage); Serial.println(" volts");
27.
28. // now print out the temperature
29. float temperatureC = (voltage - 0.5) * 100 ;  //converting from 10 mv per degree
    wit 500 mV offset
30.                                   //to degrees ((voltage - 500mV)
    times 100)
31. Serial.print(temperatureC); Serial.println(" degrees C");
32.
33. // now convert to Fahrenheit
34. float temperatureF = (temperatureC * 9.0 / 5.0) + 32.0;
35. Serial.print(temperatureF); Serial.println(" degrees F");
36.
37. delay(1000);
38.}
```

Experiment No 5

Aim: Study and Implement RFID, NFC using Arduino.

Objectives: Student should get the knowledge of RFID, NFC using Arduino.

Outcomes: Student will be developed programs using Arduino IDE and Arduino Board for RFID, NFC.

Hardware Requirements:

- 1 x Arduino UNO or 1 x Starter Kit for Raspberry Pi + Raspberry Pi
- 1 x Communication Shield
- 1 x RFID 13.56 MHz / NFC Module for Arduino and Raspberry Pi
- 1 x Mifare tag (card/keyring/sticker)
- 1 x PC

RFID:

RFID system is made up of two parts: a tag or label and a reader. RFID tags or labels are embedded with a transmitter and a receiver. The RFID component on the tags have two parts: a microchip that stores and processes information, and an antenna to receive and transmit a signal. The tag contains the specific serial number for one specific object.

To read the information encoded on a tag, a two-way radio transmitter-receiver called an interrogator or reader emits a signal to the tag using an antenna. The tag responds with the information written in its memory bank. The interrogator will then transmit the read results to an RFID computer program.

How to Interface RFID Reader to Arduino

Lets first wire the whole thing up. You may observe the circuit diagram given below. Take note of the following stuffs.

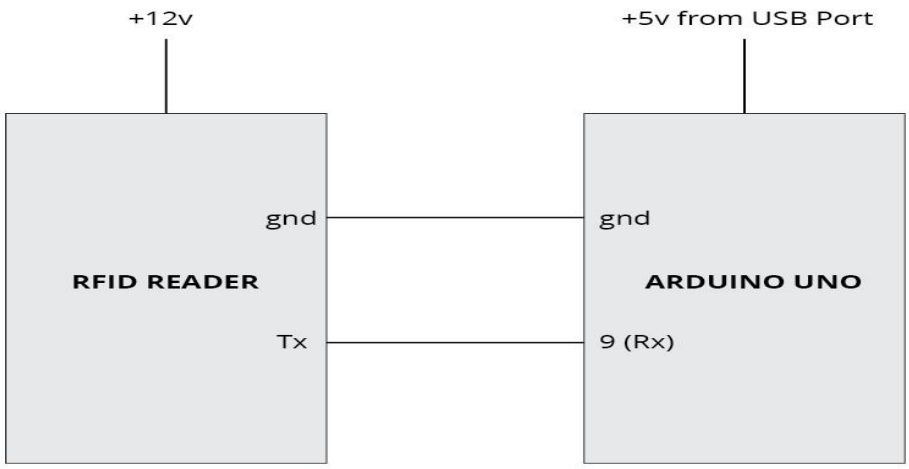
Note 1:- Power supply requirement of RFID Readers vary from product to product. The RFID reader I used in this tutorial is a 12 Volts one. There are 5 Volts and 9 Volts versions available in the market.

Note 2:- You may ensure the RFID Reader and RFID Tags are frequency compatible. Generally they are supposed to be 125Khz. You may ensure this before purchasing them.

Note 3:- There are two possible outputs from an RFID Reader. One is RS232 compatible output and other one is TTL compatible output. A TTL compatible output pin can be connected directly to Arduino. Whereas an RS232 compatible output must be converted

to TTL using an RS232 to TTL converter (You can design this yourself using MAX232 IC)

So that's all! Lets get to circuit diagram!



Make connections as shown. Make sure you connect Ground Pin of RFID reader to Ground Pin of Arduino. I am using the SoftwareSerial Library of Arduino which enables digital pins to be used in serial communication. I have used pin 9 as the Rx of Arduino. (You can also use the hardware Rx pin of Arduino uno – that's pin 0). If you are new to SoftwareSerial Library, you may read my previous tutorial on**interfacing GSM module to Arduino** (this article clearly explains how to use Software Serial Library).

Lets get to the programming side!

```
#include <SoftwareSerial.h>



SoftwareSerial mySerial(9, 10);

void setup()

{
```

```
    mySerial.begin(9600); // Setting the baud rate of Software Se rial Library


    Serial.begin(9600);    //Setting the baud rate of Serial Monito
r

  }void loop()

{

  if(mySerial.available()>0)

    {

    Serial.write(mySerial.read());

    }

}
```

mySerial.available() – checks for any data coming from RFID reader module through the SoftwareSerial pin 9. Returns the number of bytes available to read from software serial port. Returns a -1 if no data is available to read.

mySerial.read() – Reads the incoming data through software serial port.

Serial.write() – Prints data to serial monitor of arduino. So the function Serial.write(mySerial.read()) – prints the data collected from software serial port to serial monitor of arduino.

Experiment No 6

Aim: Study and Implement MQTT Protocol using Arduino.

Objectives: Student should get the knowledge of MQTT Protocol using Arduino.

Outcomes: Student will be developed programs using Arduino IDE and Arduino Board for MQTT Protocol

MQTT:

MQ Telemetry Transport (MQTT) is an open source protocol for constrained devices and low-bandwidth, high-latency networks. It is a publish/subscribe messaging transport that is extremely lightweight and ideal for connecting small devices to constrained networks.

MQTT is bandwidth efficient, data agnostic, and has continuous session awareness. It helps minimize the resource requirements for your IoT device, while also attempting to ensure reliability and some degree of assurance of delivery with grades of service. MQTT targets large networks of small devices that need to be monitored or controlled from a back-end server on the Internet. It is not designed for device-to-device transfer. Nor is it designed to "multicast" data to many receivers. MQTT is extremely simple, offering few control options.

MQTT methods

MQTT defines methods (sometimes referred to as *verbs*) to indicate the desired action to be performed on the identified resource. What this resource represents, whether pre-existing data or data that is generated dynamically, depends on the implementation of the server. Often, the resource corresponds to a file or the output of an executable residing on the server.

Connect

Waits for a connection to be established with the server.

Disconnect

Waits for the MQTT client to finish any work it must do, and for the TCP/IP session to disconnect.

Subscribe

Waits for completion of the Subscribe or UnSubscribe method.

UnSubscribe

Requests the server unsubscribe the client from one or more topics.

Publish

Returns immediately to the application thread after passing the request to the MQTT client.

Experiment No 7

Aim: Study and Configure Raspberry Pi.

Objectives: Student should get the knowledge of Raspberry Pi.

Outcomes: Student will be get knowledge of Raspberry Pi

Raspberry Pi

The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools and in developing countries. The original model became far more popular than anticipated, selling outside of its target market for uses such as robotics. Peripherals (including keyboards, mice and cases) are not included with the Raspberry Pi. Some accessories however have been included in several official and unofficial bundles.

According to the Raspberry Pi Foundation, over 5 million Raspberry Pis have been sold before February 2015, making it the best-selling British computer. By November 2016 they had sold 11 million units, reaching 12.5m in March 2017, making it the third best-selling "general purpose computer" ever.

To get started with Raspberry Pi, you need an operating system. NOOBS (New Out Of Box Software) is an easy operating system install manager for the Raspberry Pi.

How to get and install NOOBS

DOWNLOAD NOOBS OS FROM

We recommend using an SD card with a minimum capacity of 8GB.

1. GO to the https://www.raspberrypi.org/downloads/
2. Click on NOOBS, then click on the Download ZIP button under 'NOOBS (offline and network install)', and select a folder to save it to.
3. Extract the files from the zip.

FORMAT YOUR SD CARD

It is best to format your SD card before copying the NOOBS files onto it. To do this:

1. Download SD Formatter 4.0 for either Windows or Mac.

2. Follow the instructions to install the software.
3. Insert your SD card into the computer or laptop's SD card reader and make a note of the drive letter allocated to it, e.g. G:/
4. In SD Formatter, select the drive letter for your SD card and format it.

DRAG AND DROP NOOBS FILES

1. Once your SD card has been formatted, drag all the files in the extracted NOOBS folder and drop them onto the SD card drive.
2. The necessary files will then be transferred to your SD card.
3. When this process has finished, safely remove the SD card and insert it into your Raspberry Pi.

FIRST BOOT

1. Plug in your keyboard, mouse, and monitor cables.
2. Now plug the USB power cable into your Pi.
3. Your Raspberry Pi will boot, and a window will appear with a list of different operating systems that you can install. We recommend that you use Raspbian – tick the box next to Raspbian and click on Install.
4. Raspbian will then run through its installation process. Note that this can take a while.
5. When the install process has completed, the Raspberry Pi configuration menu (raspi- config) will load. Here you are able to set the time and date for your region, enable a Raspberry Pi camera board, or even create users. You can exit this menu by using Tab on your keyboard to move to Finish.

LOGGING IN AND ACCESSING THE GRAPHICAL USER INTERFACE

The default login for Raspbian is username pi with the password raspberry. Note that you will not see any writing appear when you type the password. This is a security feature in Linux.

To load the graphical user interface, type startx and press Enter.

Experiment No 8

Aim: WAP for LED blink using Raspberry Pi.

Objectives: Student should get the knowledge of LED blinking using Raspberry Pi. Outcomes:

Student will be developed program of LED bilking using Raspberry Pi. Hardware Requirements:

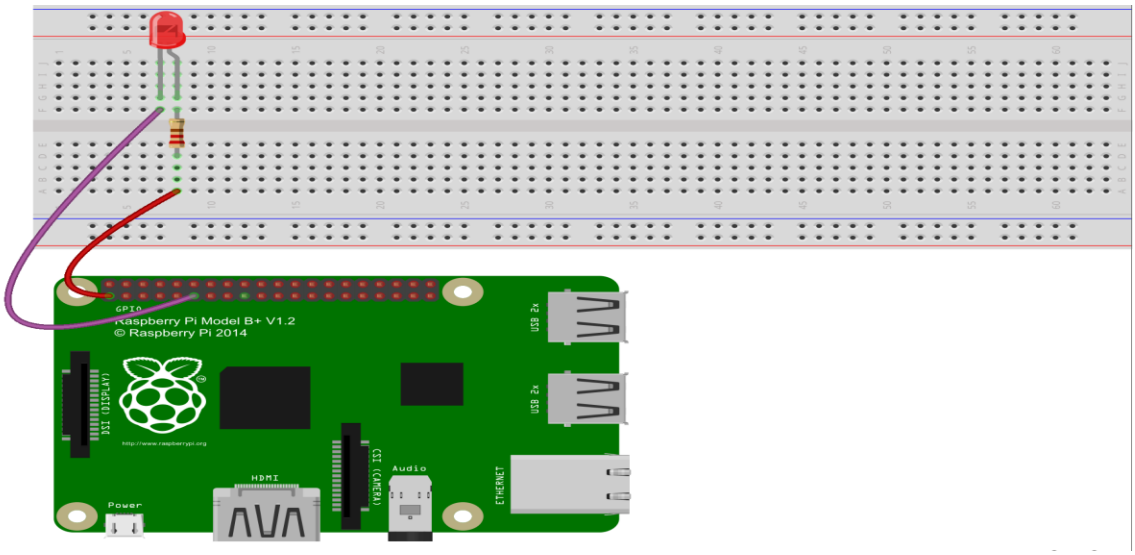
- 1x Breadboard
- 1x Raspberry Pi
- 1x RGB LED
- 1x 330Ω Resistor
- 2x Jumper Wires

Semiconductor light-emitting diode is a type of component which can turn electric energy into light energy via PN junctions. By wavelength, it can be categorized into laser diode, infrared light-emitting diode and visible light-emitting diode which is usually known as light-emitting diode (LED).

When 2V-3V forward voltage is supplied to an LED, it will blink only if forward currents flow through the LED. Usually there are red, yellow, green, blue and color-changing LEDs which change color with different voltages. LEDs are widely used due to their low operating voltage, low current, luminescent stability and small size.

LEDs are diodes too. Hence they have a voltage drop which usually varies from 1V to 3V depending on their types. Generally they brighten if supplied with a 5mA-30mA current and we usually use 10mA-20mA.Thus when an LED is used ,it is necessary to connect a current-limiting resistor to protect it from being burnt.

In this experiment, connect a 220Ω resistor to the anode of the LED, then the resistor to 3.3 V and connect the cathode of the LED to GPIO0 (See Raspberry Pi Pin Number Introduction). Write 1 to GPIO0, and the LED will stay off; write 0 to GPIO0, and then the LED will blink, just as indicated by the principle above



Step 1: Build the circuit given above

Step 2: Change directory
cd /home/pi/Sunfounder_SuperKit_Python_code_for_RaspberryPi/

Step 3: Run
sudo python 01_led.py
Now, you should see the LED blink.

Python Code

```
#!/usr/bin/env python import
RPi.GPIO as GPIO import time

LedPin = 11    # pin11

def setup():
        GPIO.setmode(GPIO.BOARD)       # Numbers GPIOs by physical location
        GPIO.setup(LedPin, GPIO.OUT)   # Set LedPin's mode is output
        GPIO.output(LedPin, GPIO.HIGH) # Set LedPin high(+3.3V) to off led
def loop():
        while True:
                print '...led on'
                GPIO.output(LedPin, GPIO.LOW)   # led on
                time.sleep(0.5)
                print 'led off...'
                GPIO.output(LedPin, GPIO.HIGH) # led off
                time.sleep(0.5)
```

```
def destroy():
    GPIO.output(LedPin, GPIO.HIGH)      # led off
    GPIO.cleanup()                      # Release resource


if __name__ == '__main__':     # Program start from here
    setup()
    try:
        loop()
    except KeyboardInterrupt:   # When 'Ctrl+C' is pressed, the child program destroy() will be executed.
        destroy()
```