V.S.B. ENGINEERING COLLEGE

(AN AUTONOMOUS INSTITUTION)

KARUDAYAMPALAYAM, KARUR - 639 111.

_____ RECORD

NAME _____ CLASS _____

REGISTER No. _____ BRANCH _____

ROLL No. _____

Certified that this is a bonafide record of work done by the above student during the year 20 - 20

Date :

Lab In-charge

Head of the Department

Submitted for the "UNIVERSITY PRACTICAL EXAMINATION"

held on.....

Internal Examiner.

External Examiner.

CONTENTS

Sl.No.	Date	Name of Experiments	Page No.	Marks Award	Sign.

CONTENTS

Sl.No.	Date	Name of Experiment	Page No.	Marks Award	Sign.

EMBEDDED SYSTEMS AND IOT LAB

EXP 1 WRITE 8051 ASSEMBLY LANGUAGE EXPERIMENTS USING SIMULATOR.

1. A PROGRAM FOR BLINKING AN LED USING Ed8im51

AIM: To write an assembly level language program for Blinking an LED light using Ed8im51

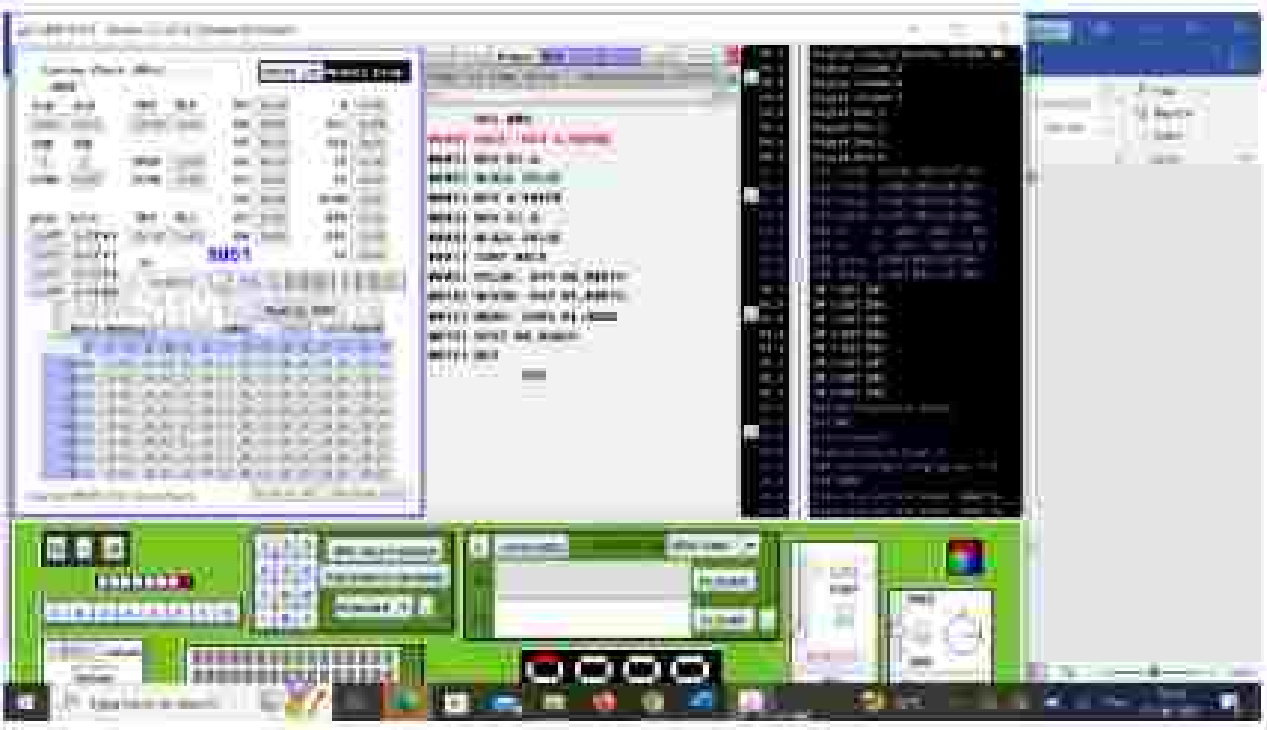
ALGORITHM

1. Start at address 0x0000 (ORG 0x0000).
2. Set up a continuous loop labeled as "BACK."
3. Move the value 0xFE (254 in decimal) to the accumulator register (A).
4. Move the value in the accumulator (A) to Port 1 (P1) to turn on the LED.
5. Call the delay subroutine by using the instruction "ACALL DELAY."
6. Move the value 0xFF (255 in decimal) to the accumulator register (A).
7. Move the value in the accumulator (A) to Port 1 (P1) to turn off the LED.
8. Call the delay subroutine by using the instruction "ACALL DELAY."
9. Jump back to the "BACK" loop using the instruction "SJMP BACK."
10. Define the delay subroutine labeled as "DELAY."
11. Move the value 0xFF (255 in decimal) to register R0.
12. Use a nested loop to create a delay:
    a. Move the value 0xFF (255 in decimal) to register R1.
    b. Create a loop labeled as "HERE" and decrement R1 using the instruction "DJNZ R1, HERE."
    c. Check if R1 is zero. If not zero, repeat the loop.
    d. Decrement R0 using the instruction "DJNZ R0, AGAIN."
    e. Check if R0 is zero. If not zero, repeat the nested loop.
    Return from the subroutine using the instruction "RET."

CODE:

```
ORG 00H
BACK: MOV A,#0FEH
MOV P1,A
ACALL DELAY
MOV A,#0FFH
MOV P1,A
ACALL DELAY
SJMP BACK
DELAY: MOV R0,#0FFH
AGAIN: MOV R1,#0FFH
HERE: DJNZ R1,HERE
DJNZ R0,AGAIN
RET
END
```

OUTPUT:



RESULT:

1.b LOGICAL AND USING EdSim51

AIM:

To write an assembly level language program Logical AND using EdSim51

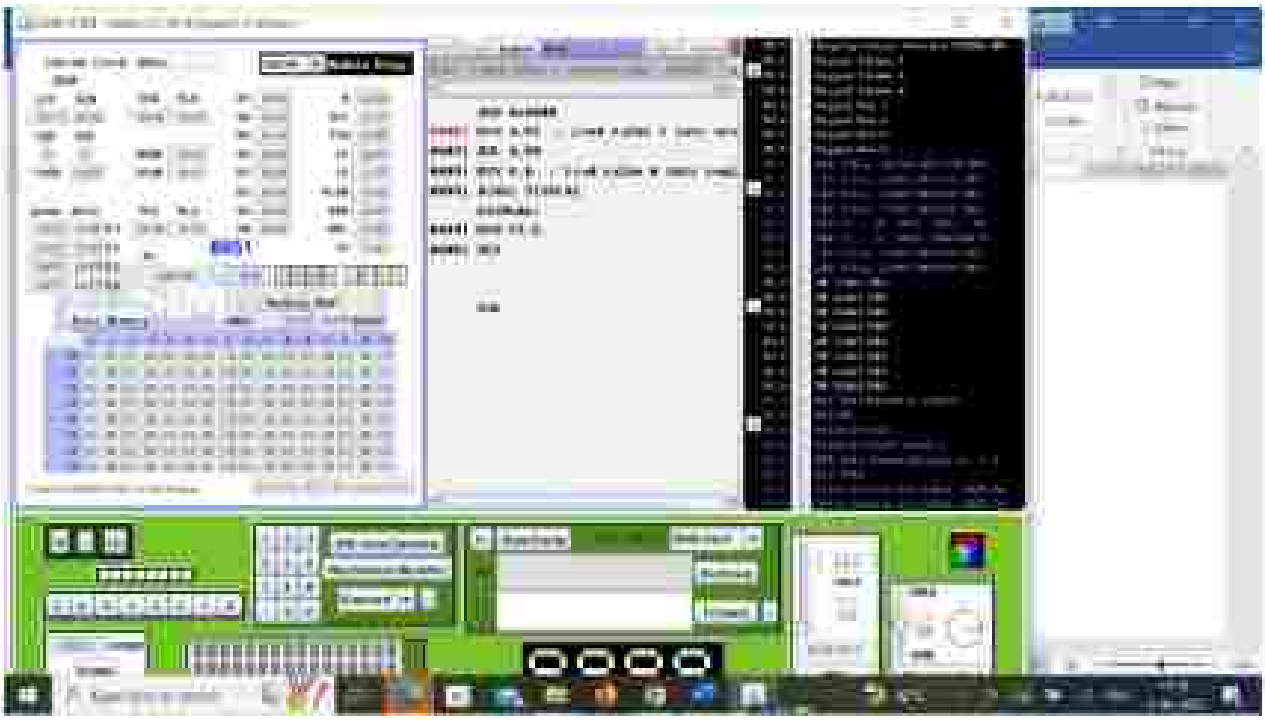
ALGORITHM:

1. Start at address 0x0000 (ORG 0x0000).
2. Move the value 1 into the accumulator register A using the instruction "MOV A, #1".
3. Perform a logical AND operation between the accumulator A and the value 0 using the instruction "ANL A, #0". This operation will result in 0, as any value ANDed with 0 gives 0.
4. Move the value in the accumulator A to register B using the instruction "MOV B, A". This stores the result of the logical AND operation in register B.
5. Call the DISPLAY subroutine using the instruction "ACALL DISPLAY".
6. Define the DISPLAY subroutine: a. Move the value in register B to Port 2 (P2) using the instruction "MOV P2, B". This will display the value stored in register B on Port 2. b. Return from the subroutine using the instruction "RET".
7. End the program using the "END" directive.

CODE:

```
ORG 0x0000

MOV A,#1  ; Load value 1 into accumulator A

ANL A,#0

MOV B,A ; Load value 0 into register B

ACALL DISPLAY

DISPLAY:

MOV P2,B

RET

END
```

OUTPUT:



PARTICULARS	MARKS ALLOTTED	MARKS OBTAINED
PERFORMANCE		
VIVA-VOCE		
RECORD		
TOTAL		

RESULT:

EXP 2 TEST DATA TRANSFER BETWEEN REGISTERS AND MEMORY USING EDSIM51

AIM:

To write a test data transfer between registers and memory using Ed8im51

ALGORITHM:

1. Start at address 0x0000 (ORG 0x0000).
2. Load the value 10 into register R0 using the instruction "MOV R0, #10".
3. Initialize register R1 with the value 0 using the instruction "MOV R1, #0".
4. Move the value from register R0 to the accumulator A using the instruction "MOV A, R0".
5. Move the value from the accumulator A to the memory location pointed to by R1 using the instruction "MOV @R1, A".
6. Increment the value in register R1 to access the next memory location using the instruction "INC R1".
7. Move the value from the memory location pointed to by R1 to the accumulator A using the instruction "MOV A, @R1".
8. Move the value from the accumulator A to register R2 using the instruction "MOV R2, A".
9. End the program using the "END" directive.

CODE:

```
ORG 0x0000  ; Starting address


MOV R0, #10 ; Load value 10 into register R0

MOV R1, #0  ; Initialize register R1 with 0


MOV A, R0   ; Move the value from register R0 to accumulator A

MOV @R1, A ; Move the value from accumulator A to the memory location pointed
             by R1


INC R1      ; Increment the value in register R1 to access the next memory location


MOV A, @R1 ; Move the value from the memory location pointed by R1 to
             accumulator A
```

```
MOV R2, A    ; Move the value from accumulator A to register R2

END
```

OUTPUT:

PARTICULARS	MARKS ALLOTTED	MARKS OBTAINED
PERFORMANCE		
VIVA-VOCE		
RECORD		
TOTAL		

RESULT:

EXP.3 **PERFORM ALU OPERATIONS**

AIM:

To perform ALU operations in Embedded Systems using Simulator.

ALGORITHM:

Step 1: Read input operands and control signal.

Step 2: Determine the type of operation based on the control signal.

Step 3: Execute the selected operation:

- For arithmetic operations, perform the calculation.
- For logical operations, perform the logical operation.
- For other operations, follow the respective algorithm.

Step 4: Store the result in the ALU output register.

Step 5: Display the result on the output device.

Step 6: End the ALU operation module.

Code:

1. **Addition:**

```
ORG 0x0000
MOV A, #10H
MOV B, #20H
ADD A, B
MOV R2,A
   JMP $
```

OUTPUT:

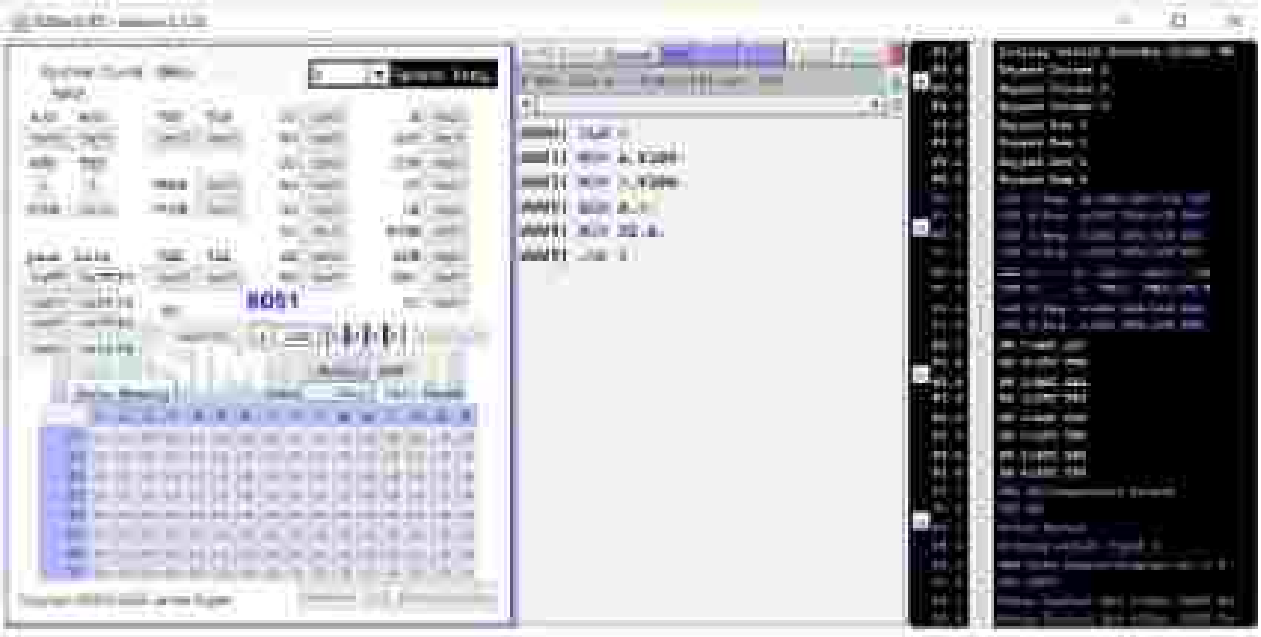


Fig addition

2.SUBTRACTION

```
ORG 0x0000
MOV A, #50H
MOV B, #30H
SUBB A, B
MOV R1,,A
JMP $
```

Output:

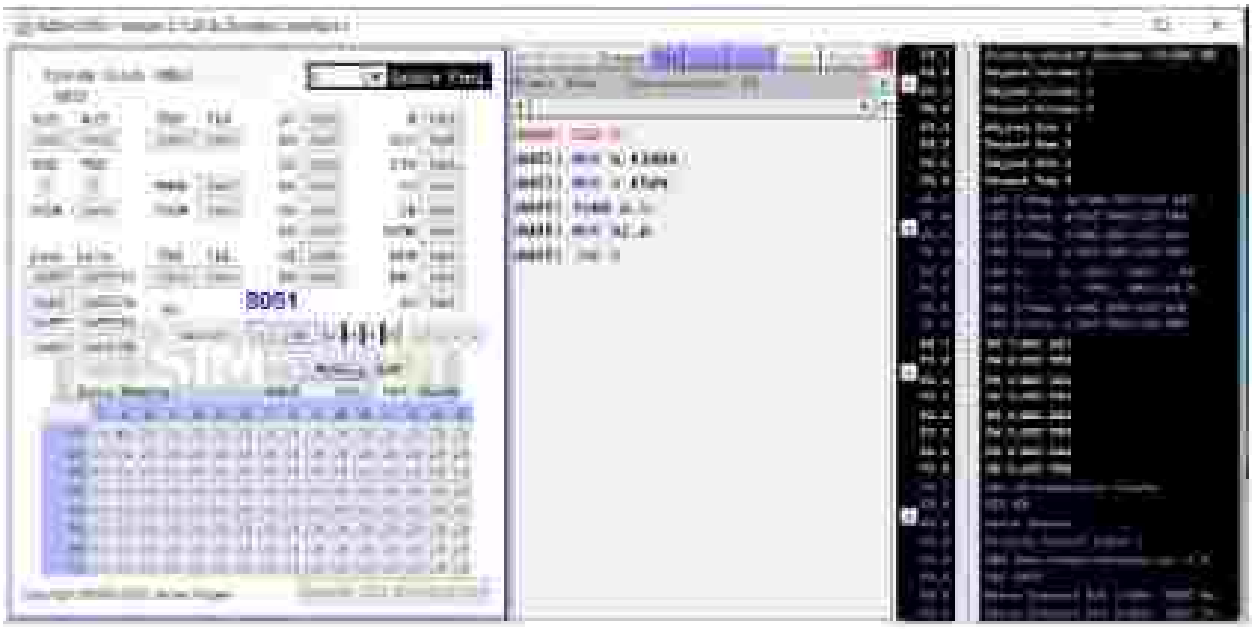


Fig: subtraction

3.Multiplication

```
ORG 0X000
        MOV A,#06H
        MOV B,#03H
        MUL AB
        MOV R3,A
        JMP $
```

Output:

Fig multiplication

4.Division

ORG 0X000

MOV A,#06H

MOV B,#03H

DIV AB

MOV R0,A

JMP $

OUTPUT:

Fig Division

PARTICULARS	MARKS ALLOTTED	MARKS OBTAINED
PERFORMANCE		
VIVA-VOCE		
RECORD		
TOTAL		

RESULT:

EXP 4 Write Basic and arithmetic Programs Using Embedded C in keil

AIM:

To write the Embedded C programs using in keil software

ALGORITHM:

1. Start the program.
2. Declare the necessary variables for the arithmetic operation (e.g., num1, num2, result).
3. Prompt the user to enter the input values (e.g., num1, num2).
4. Read the input values from the user.
5. Perform the desired arithmetic operation (e.g., addition, subtraction, multiplication, division) using the input values.
6. Store the result in the 'result' variable.
7. Display the result to the user.
8. End the program.

CODE 1 :Program to find the sum of two numbers:

```
#include <stdio.h>

int main() {
  int num1, num2, sum;

  printf("Enter two numbers: ");
  scanf("%d %d", &num1, &num2);

  sum = num1 + num2;

  printf("Sum: %d\n", sum);

  while (1) {} // To keep the program running in Keil

  return 0;
```

}

OUTPUT:

RESULT:

CODE 2:Program to check if a number is even or odd:

```
#include <stdio.h>

int main() {
  int num;

  printf("Enter a number: ");
  scanf("%d", &num);

  if (num % 2 == 0) {
    printf("%d is even\n", num);
  } else {
    printf("%d is odd\n", num);
  }

  while (1) {}  // To keep the program running in Keil

  return 0;
}
```

OUTPUT:

RESULT:

CODE 3:Program to find the factorial of a number:

```
#include <stdio.h>

int main() {
  int num, factorial = 1;
```

```
    printf("Enter a number: ");
    scanf("%d", &num);

    for (int i = 1; i <= num; i++) {
        factorial *= i;
    }

    printf("Factorial: %d\n", factorial);

    while (1) {} // To keep the program running in Keil

    return 0;
}
```

OUTPUT:

PARTICULARS	MARKS ALLOTTED	MARKS OBTAINED
PERFORMANCE		
VIVA-VOCE		
RECORD		
TOTAL		

RESULT:

EXP. 5 BLINKING OF LED USING ARDUINO PROGRAMMING

AIM:

To create a Circuit using Arduino Board to control the LED On/OFF Using Switch.

Components Required:

Name	Quantity	Component
U1	1	Arduino uno
S1	1	Pushbutton
R1	1	10 kΩ Resistor
D1	1	5MM LED

ALGORITHM:

Step 1: Set up the hardware

- Connect the longer leg (anode) of the LED to a digital pin on the Arduino board (e.g., pin 13).
- Connect the shorter leg (cathode) of the LED to the ground (GND) pin on the Arduino board using the current-limiting resistor.

Step 2: Set the pin mode

In the Arduino setup function, set the pin mode for the LED pin to OUTPUT. This will configure the pin to send output signals to the LED.

Step 3: Blink the LED

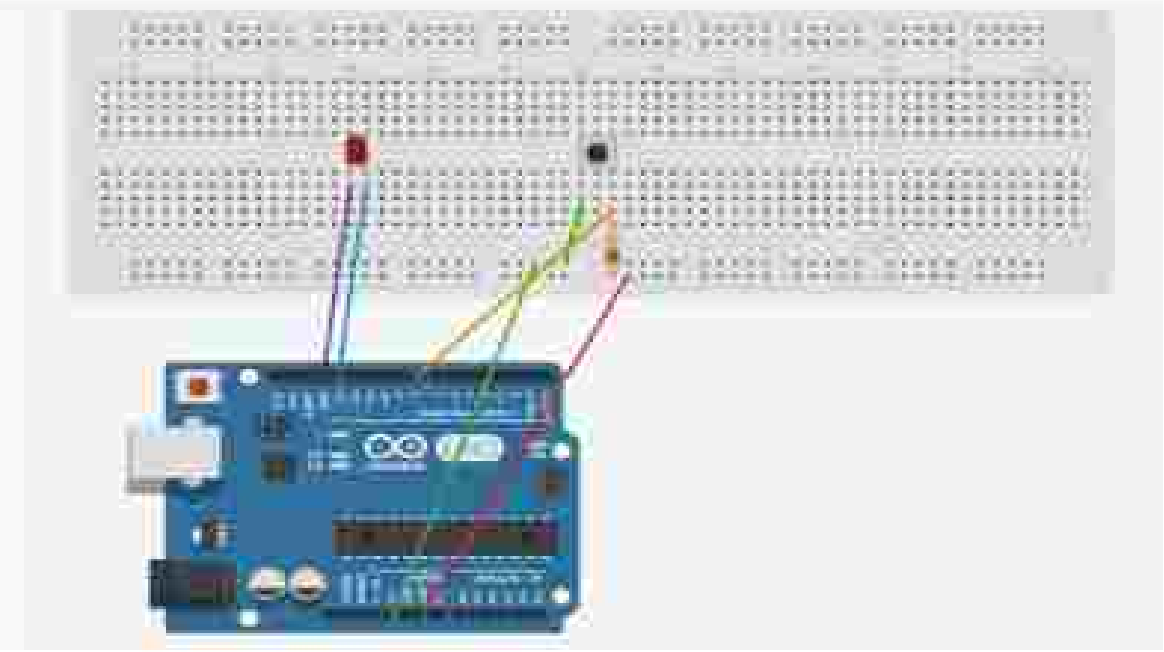
- In the Arduino loop function, turn the LED ON by setting the digital pin HIGH.
- Add a short delay using the delay() function to keep the LED on for a specific duration (e.g., 1000 milliseconds or 1 second).
- Turn the LED OFF by setting the digital pin LOW.
- Add another delay to keep the LED off for the same duration as before.

Step 4: Repeat

- The loop function runs continuously, so the LED will keep blinking ON and OFF in a loop.

CODE :

```
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int ledPin = 13;
int switchPin = 8;
boolean lastButton= LOW;
boolean currentButton= LOW;
boolean ledOn= false;
// the setup routine runs once when you press reset:
void setup()
{
// initialize the digital pin as an output.
pinMode(8, INPUT);
pinMode(13,OUTPUT);
}
boolean debounce(boolean last)
{
boolean current= digitalRead(switchPin);
if(last != current)
{
delay(15);
current=digitalRead(switchPin);
}
return current;
}

// the loop routine runs over and over again forever:
void loop() {
currentButton=debounce(lastButton);
if(lastButton==LOW &&currentButton==HIGH)
{
ledOn = !ledOn;
}
lastButton= currentButton;
digitalWrite(ledPin,ledOn);
}
```



PARTICULARS	MARKS ALLOTTED	MARKS OBTAINED
PERFORMANCE		
VIVA-VOCE		
RECORD		
TOTAL		

RESULT:

Thus the LED Blink control using Arduino is executed successfully.

EXPT 6: EXPLORE THE DIFFERENT COMMUNICATION METHODS WITH IOT (ZIGBEE, GSM, BLUETOOTH)

Aim:

To explore the different communication methods with IOT like zigbee, Gsm, Bluetooth.

Theory:

IoT Communication:

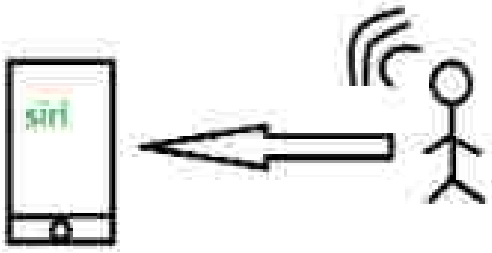
IoT is the connection of devices over the internet, where these smart devices communicate with each other , exchange data , perform some tasks without any human involvement. These devices are embedded with electronics, software, network and sensors which help in communication. Communication between smart devices is very important in IOT as it enables these devices to gather, exchange data which contribute in success of that IOT product/project.

Types of Communications in IOT:

The following are some communication types in IoT:-

1. Human to Machine (H2M):

In this human gives input to IOT device i.e as speech text image etc. IOT device (Machine) like sensors and actuators then understands input, analyses it and responds back to human by means of text or Visual Display. This is very useful as these machines assist humans in every everyday tasks. It is a combo of software and hardware that includes human interaction with a machine to perform a task.



H2M communication

Merits: This H2M has a user-friendly interface that can be quickly accessed by following the instructions. It responds more quickly to any fault or failure. Its features and functions can be customized.

Examples:

- Facial recognition.
- Bio-metric Attendance system.
- Speech or voicerecognition.

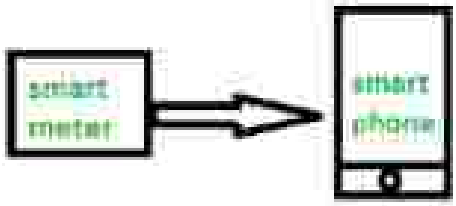
2. Machine to Machine (M2M):

The process of exchanging information or messages between two or more machines or devices is known as Machine to Machine (M2M) communication.

It is the communication among the physical things which do not need human intervention.

M2M communication is also named as Machine Type communication in 3GPP (3rd Generation Partnership Project).

In this the interaction or communication takes place between machines by automating data programs. In this machine level instructions are required for communication. Here communication takes place without human interaction. The machines may be either connected through wires or by wireless connection. An M2M connection is a point-to-point connection between two network devices that helps in transmitting information using public networking technologies like Ethernet and cellular networks. IoT uses the basic concepts of M2M and expands by creating large "cloud" networks of devices that communicate with one another through cloud networking platforms.

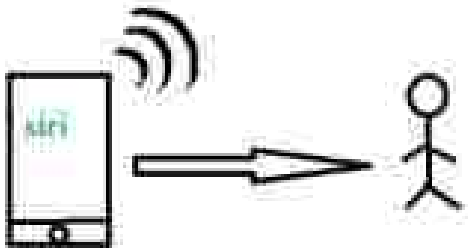


M2M communication

3. Machine to Human (M2H):

In this machine interacts with Humans. Machine triggers information(text messages images voice signals) respective / irrespective of any human presence. This type of communication is most commonly used

where machines guide humans in their daily life. It is way of interaction in which humans co-work with smart systems and other machines by using tools or devices to finish a task



M2H communication

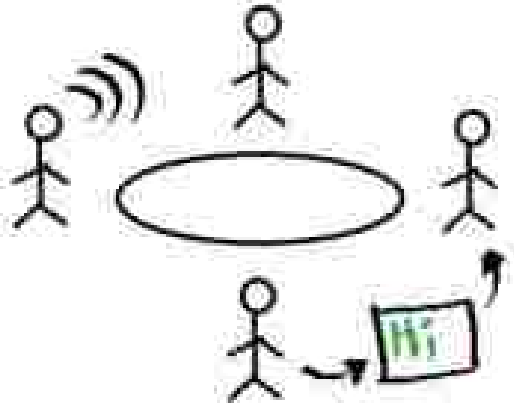
Examples:

- Fire Alarms
- Traffic Light
- Fitness bands
- Health monitoring devices

4. Human to Human (H2H) :

This is generally how humans communicate with each other to exchange information by speech, writing, drawing, facial expressions, body language etc. Without H2H, M2M applications cannot produce the expected benefits unless humans can immediately fix issues, solve challenges, and manage scenarios.

The process of exchanging information or messages between two or more people is known as human to human (H2H) communication. This can be done through various means such as verbal, non-verbal, or written communication.



H2H communication

For, communication of IoT devices many protocols are used. These IoT protocols are modes of communication which give security to the data being exchanged between IoT connected devices. Example bluetooth, wifi, zigbee etc.

BLUETOOTH:

Bluetooth is universal for short-range wireless voice and data communication. It is a Wireless Personal Area Network (WPAN) technology and is used for exchanging data over smaller distances. This technology was invented by Ericsson in 1994. It operates in the unlicensed, industrial, scientific, and medical (ISM) band from 2.4 GHz to 2.485 GHz. Maximum devices that can be connected at the same time are 7. Bluetooth ranges up to 10 meters. It provides data rates up to 1 Mbps or 3 Mbps depending upon the version. The spreading technique that it uses is FHSS (Frequency-hopping spread spectrum). A Bluetooth network is called a **piconet** and a collection of interconnected piconets is called **scatternet**.

What is Bluetooth?

Bluetooth simply follows the principle of transmitting and receiving data using radio waves. It can be paired with the other device which has also Bluetooth but it should be within the estimated communication range to connect. When two devices start to share data, they form a network called piconet which can further accommodate more than five devices.

Points to remember for Bluetooth:

- Bluetooth Transmission capacity 720 kbps.
- Bluetooth is Wireless.
- Bluetooth is a Low-cost short-distance radio communications standard.
- Bluetooth is robust and flexible.
- Bluetooth is cable replacement technology that can be used to connect almost any device to any other device.
- The basic architecture unit of Bluetooth is a piconet.

ZIGBEE:

ZigBee is a Personal Area Network task group with low rate task group 4. It is a technology of home networking. ZigBee is a technological standard created for controlling and sensing the network. As we know that ZigBee is

the Personal Area Network of task group 4 so it is based on IEEE 802.15.4 and is created by Zigbee Alliance.

ZigBee is an open, global, packet-based protocol designed to provide an easy-to-use architecture for secure, reliable, low power wireless networks. Flow or process control equipment can be place any here and still communicate with the rest of the system. It can also be moved, since the network doesn't care about the physical location of a sensor, pump or valve.

IEEE802.15.4 developed the PHY and MAC layer whereas, the ZigBee takes care of upper higher layers.

ZigBee is a standard that addresses the need for very low-cost implementation of Low power devices with Low data rates for short-range wireless communications.

IEEE 802.15.4 supports star and peer-to-peer topologies. The ZigBee specification supports star and two kinds of peer-to-peer topologies, mesh and cluster tree. ZigBee-compliant devices are sometimes specified as supporting point-to-point and point-to-multipoint topologies.

Why another short-range communication standard??

Features of Zigbee:

1. Stochastic addressing A device is assigned a random address and announced. Mechanism for address conflict resolution. Parents node don't need to maintain assigned address table.
2. Link Management: Each node maintains quality of links to neighbors. Link quality is used as link cost in routing.
3. Frequency Agility: Nodes experience interference report to channel manager, which then selects another channel
4. Asymmetric Link: Each node has different transmit power and sensitivity. Paths may be asymmetric.
5. Power Management: Routers and Coordinators use main power. End Devices use batteries.

GSM:

GSM stands for **Global System for Mobile Communication.** GSM is an open and digital cellular technology used for mobile communication. It uses 4 different frequency bands of 850 MHz, 900 MHz, 1800 MHz and 1900 MHz . It uses the combination of FDMA and TDMA. This article includes all the concepts of GSM architecture and how it works.

GSM is having 4 different sizes of cells are used in GSM :

1. Macro: In this size of cell, Base Station antenna is installed
2. Micro: In this size of cell, antenna height is less than the average roof level.
3. Pico: Small cells' diameter of few meters.
4. Umbrella: It covers the shadowed (Fill the gaps between cells) regions.

Features of GSM are:

1. Supports international roaming
2. Clear voice clarity
3. Ability to support multiple handheld devices.
4. Spectral / frequency efficiency.
5. Low powered handheld devices.
6. Ease of accessing network
7. International ISDN compatibility.
8. Low service cost.
9. New features and services.

GSM security:

- GSM offers several security using confidential information stored in the AUC and in the individual SIM.
- The SIM stores personal secret data and is protected with a pin against unauthorized use.

PARTICULARS	MARKS ALLOTTED	MARKS OBTAINED
PERFORMANCE		
VIVA-VOCE		
RECORD		
TOTAL		

RESULT:

EXPT. 7. INTRODUCTION TO RASPBERRY PI PLATFORM AAND PYTHON PROGRAMMING

Aim:

To study about Raspberry Pi platform and python programming

Description:

Raspberry Pi is a credit-card sized, single-board computer developed by the Raspberry Pi Foundation, a UK-based charity that works to ensure global access of computing and digital technology. It was initially developed as a low-cost microcomputer to impart digital skills to kids. It comes without any power supply or peripherals as it's meant to be programmed in Python (hence, the "Pi" in its name).

Raspberry Pi (RPi) is not the only single-board computer but it's currently the most popular. It offers several features, including

- A powerful ARM processor
- Ethernet
- Onboard Wi-Fi and Bluetooth
- HDMI ports
- DSI display port
- CSI camera port
- Stereo audio and composite video port
- USB ports
- A 40-pin GPIO header

RPi is an all-in-one package that can be used for desktop programming, networking, web development, embedded systems programming, the Internet of things (IoT), robotics, automation — and for anything that falls within a desktop, server, or embedded domain. It can run any Debian-based Linux distribution, as well as Windows and Android.

Additionally, RPi is not limited to Python but can also take advantage of C and Java programming languages. The availability of a 40-pin, general-purpose input-output header — which includes I2C and SPI interfaces — makes it an ideal machine for embedded programming, backed by Linux (or another operating system).

RPi has truly put the power of computing into the hands of the masses by offering a cross-platform computing machine that's extremely affordable.

What can RPi be used for?

Raspberry Pi is a low-power, general-purpose microcomputer built on a single board. The board includes its processor, GPU, RAM, and input output peripherals. It also offers several unique hardware features that can be used for multiple applications, which are independent of any computing domain (desktop, mobile, server, or embedded), operating system, or programming language.

Since it's possible to run any Debian-based Linux distribution on an RPi board, it can work with Linux and its internal systems, such as shell scripting or Linux system administration. The computing device can also host Windows and Android operating systems.

Essentially, RPi is a low-cost test machine for system programming and administration. This means any programming language can be used for application development, provided it's supported by the operating system that's running on it. Such freedom is significant as it opens near-limitless programming possibilities for different types of apps on Raspberry Pi. Any general-purpose application on any programming language can be developed on an RPi computer.

RPi can also be used for any desktop programming task because it's equipped with RAM, GPU, HDMI and USB ports, and a DSI display port with an onboard processor. Thanks to a 40-pin GPIO header, RPi can even connect with parallel computer bus interfaces. The additional stereo audio and composite video ports mean the board can be used to develop several multimedia applications, gaming, or a home theatre PC. The Ethernet, Bluetooth, BLE, and Wi-Fi capabilities allow for program network applications. In fact, the RPi can itself be used as a low-profile web server.

The 40-pin GPIO header (of which 28 are available) on the tiny RPi board can be used to interface sensors and actuators for embedded systems programming, robotics, and automation. The advantage of using Raspberry Pi for embedded systems: it will have an operating system running on it. This allows for complex embedded applications that may require sophisticated software. Another plus: embedded scripts or apps can be easily be upgraded or developed to upgrade when necessary via an Internet connection.

RPi can also be used for the development of Android-specific applications. Examples include cluster computing, artificial intelligence, scientific computing, supercomputing, web development, and the IoT. This is a small and powerful computer that can be used for a variety of multi-platform purposes — or until its processor hits its limit.

Operating systems

It's possible to run any Debian-based Linux distribution on Raspberry Pi as well as Windows or Android. However, Raspbian is the official Linux distribution that's developed and maintained by the Raspberry Pi Foundation.

Other popular Linux distributions that can be run on RPi include Ubuntu, Kali Linux, CentOS, Arch Linux ARM, Gentoo, openSUSE, OSMC, OpenMediaVault, Recalbox, RetroPie, Lakka, and many others.

Programming languages

Python is the primary programming language on Raspberry Pi. Scratch is another primary language available on RPi that can be used for basic computing. There is no limitation on the use of any programming language on the RPi board. Any programming language that is supported by the operating system running on it can be used for software development. Other popular programming languages that can be used on Raspberry Pi include C, C++, Java, HTML5, JavaScript, JQuery, Pearl, Erlang, etc.

PARTICULARS	MARKS ALLOTTED	MARKS OBTAINED
PERFORMANCE		
VIVA-VOCE		
RECORD		
TOTAL		

RESULT:

EXP NO 8: MEASURING DISTANCE USING ARDUINO AND ULTRASONIC SENSORS

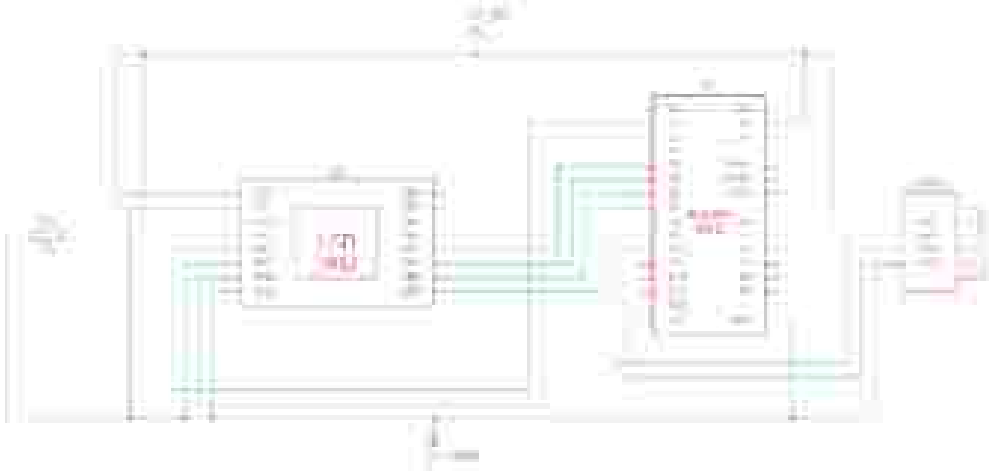
AIM:

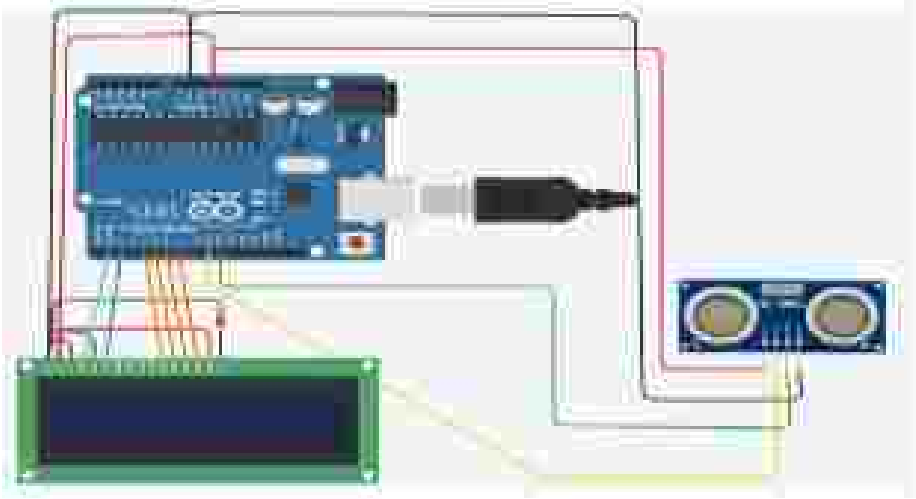
To measure the distance of an object using Ultrasonic Sensors and Arduino.

Components Required:

Name	Quantity	Component
U1	1	Arduino Uno R3
U2	1	LCD 16x2
R1	1	1 kΩ Resistor
ultra1	1	Ultrasonic Distance Sensor

Circuit Diagram:

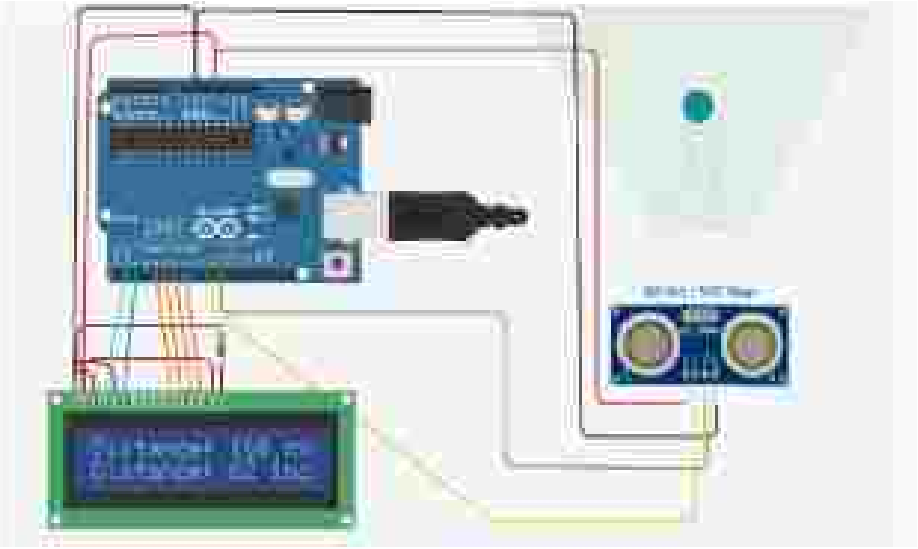




Code:

```
=include <LiquidCrystal.h> // includes the LiquidCrystal Library
LiquidCrystallcd(1, 2, 4, 5, 6, 7); // Creates an LCD object. Parameters: (rs,
enable, d4, d5, d6, d7)
const int trigPin = 9;
const int echoPin = 10;
long duration;
int distanceCm, distanceInch;
void setup() {
lcd.begin(16,2); // Initializes the interface to the LCD screen, and specifies the
dimensions (width and height) of the display
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
}
void loop() {
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distanceCm= duration*0.034/2;
distanceInch = duration*0.0133/2;
lcd.setCursor(0,0); // Sets the location at which subsequent text written to the
LCD will be displayed
lcd.print("Distance: "); // Prints string "Distance" on the LCD
lcd.print(distanceCm); // Prints the distance value from the sensor
lcd.print(" cm");
delay(10);
lcd.setCursor(0,1);
lcd.print("Distance: ");
lcd.print(distanceInch);
lcd.print(" inch");
delay(10);
}
```

OUTPUT :



PARTICULARS	MARKS ALLOTTED	MARKS OBTAINED
PERFORMANCE		
VIVA-VOCE		
RECORD		
TOTAL		

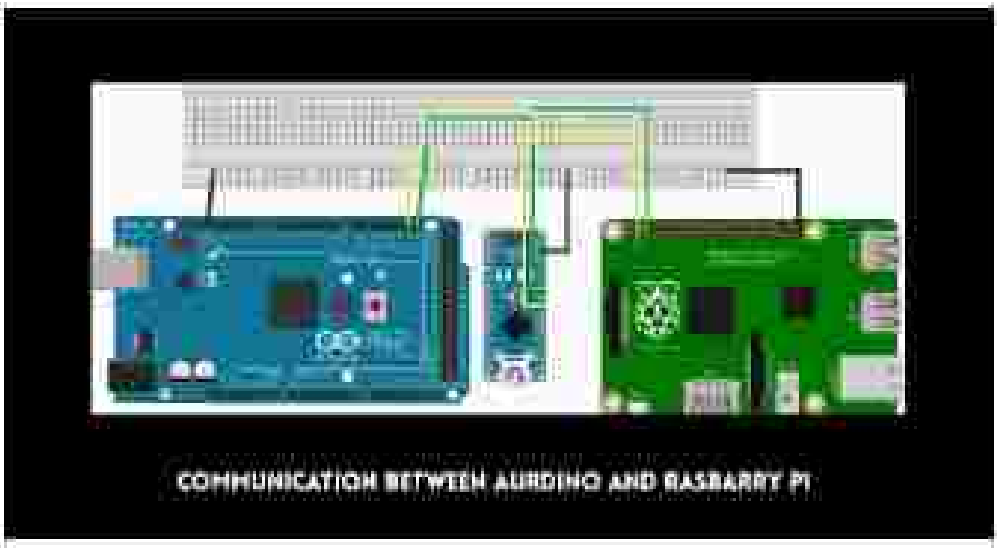
RESULT:

EXPT 9 **COMMUNICATE BETWEEN ARDUINO AND RASPBERRY PI USING ANY WIRELESS MEDIUM**

Aim:

To study about the communication between Arduino and Raspberry PI using any wireless medium

Description:



1. Summary:

- This blog walks you through the process of how to communicate an Arduino and a Raspberry Pi with nRF24L01 module. To Start off We will be learning Some basics of Raspberry Pi then we will have a look at Headless Raspberry Pi setup. For the Implementation firstly, we will understand the simple communication of Arduin to Arduino then we will learn Arduino to the raspberry pi.

- I found many guides to accomplish this around the Internet, but none of them was complete or fully worked for me. Here we will understand step by step guide of this process. And at the end will be showing a working example for the same

2. Required Components:

- Ultrasonic Sensor
- 2 Arduino Uno
- Raspberry Pi 3
- 2 nRF2401 transmitter and receiver
- Jump wires
- Arduino cable
- MINI USB 2.0 for Pi
- Breadboard

3. Required Software:

- Raspbian for pi
- Arduino IDE or Visual Studio
- Putty on a remote computer for SSH
- VNC viewer on a remote computer

4. Introduction to Raspberry Pi 3:

Raspberry Pi 3 On-chip Hardware

4.1. Components Specifications:

- 1.4 GHz 64-bit quad-core ARM Cortex-A53, 1GB RAM
- 2.4/5Ghz dual-band 802.11ac Wireless LAN, 10/100/1000Mbps Ethernet Bluetooth 4.2
- 4 USB ports, Full HDMI port, Combined 3.5mm audio jack and composite video port, 40 GPIO pins
- Micro SD card slot, Video Core IV 3D graphics core, Camera interface (CSI), Display interface (DSI)

4.2 Raspberry Pi Pinout:

4.3. Headless Raspberry Pi Setup:

- Say, I just bought a raspberry pi and wish to check out how it works. But all I have is my Laptop, the Pi, a micro SD card, and my Wi-Fi network. How do I connect and control the Pi?

1) Download Raspbian:

- Your Pi needs an OS. Download Raspbian from Raspberrypi.org 's download section:

2) Download SD Memory Card Formatter:

- It is used to format the SD card as it is needed that the SD card should be empty before the flashing image you downloaded. You can download it from https://www.sdcard.org/downloads/formatter/eula_windows/

3) Flash it onto an SD card:

- You need to flash this downloaded image to the micro SD card. Assuming your laptop has an SD card slot or a micro Sd card reader, you need a flashing software like etcher. Go ahead and download from https://etcher.io

4) Configure Wi-Fi:

- It's easier to make two devices talk to each other if they are in the same network. An ethernet cable can easily make your laptop's network available to the Pi. But we don't have one. So, we are going

to add a file to the SD card so that the Pi boots with a wifi pre-configured.

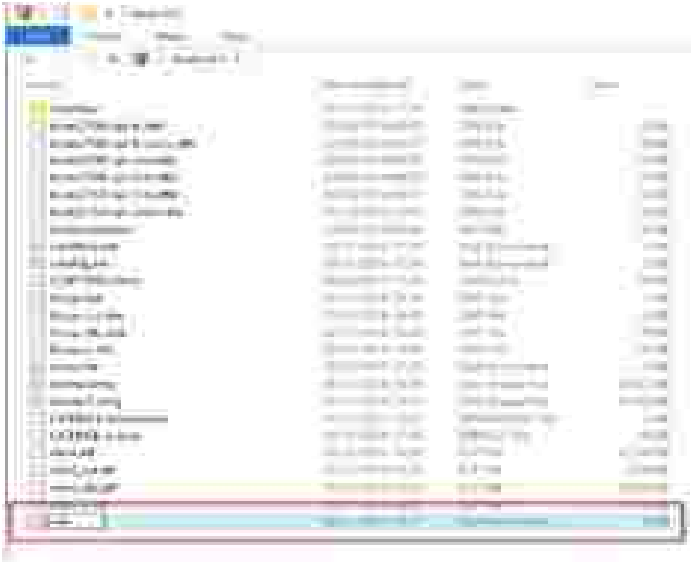
- The SD card mounts as two volumes boot and rootfs . Open the boot volume and create a file named wpa_supplicant.conf . n booting the RPi, this file will be copied to /etc/wpa_supplicant directory in /rootfs partition. The copied file tells the Pi the WIFI setup information. This would overwrite any existing WIFI configuration, so if you had already configured WIFI on the pi, then that will be overwritten.

- A typical wpa_supplicant.conf file is as follows:

```
ctrl_interface=DIR=/var/run/wpa_supplicant
GROUP=netdev update_c nfig=1country=US
network={ ssid="«your_SSID»"  psk="«your_PSK»"  key_mgmt=WPA-PSK}
```

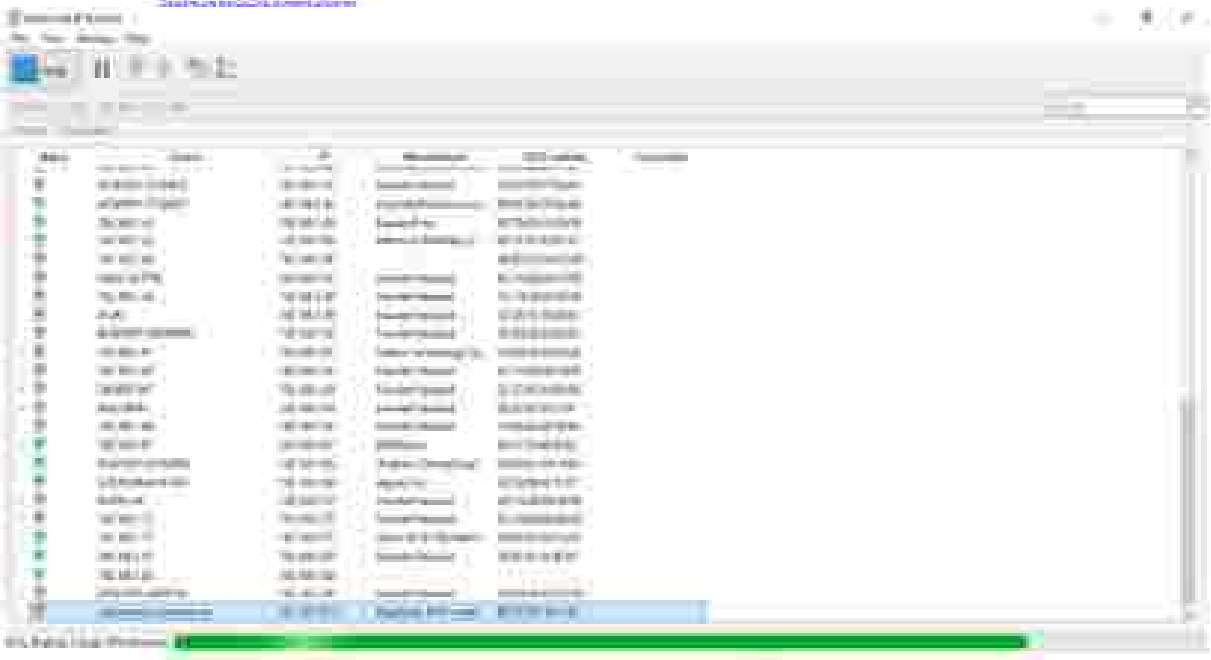
5) Enable SSH

- We will later access the Pi using a secured shell (SSH), SSH is disabled by default in Raspbian. To enable SSH, create a file named ssh in the boot partition. If you are on Linux, use the touch command to do that.



6) Find Pi's Ip address:

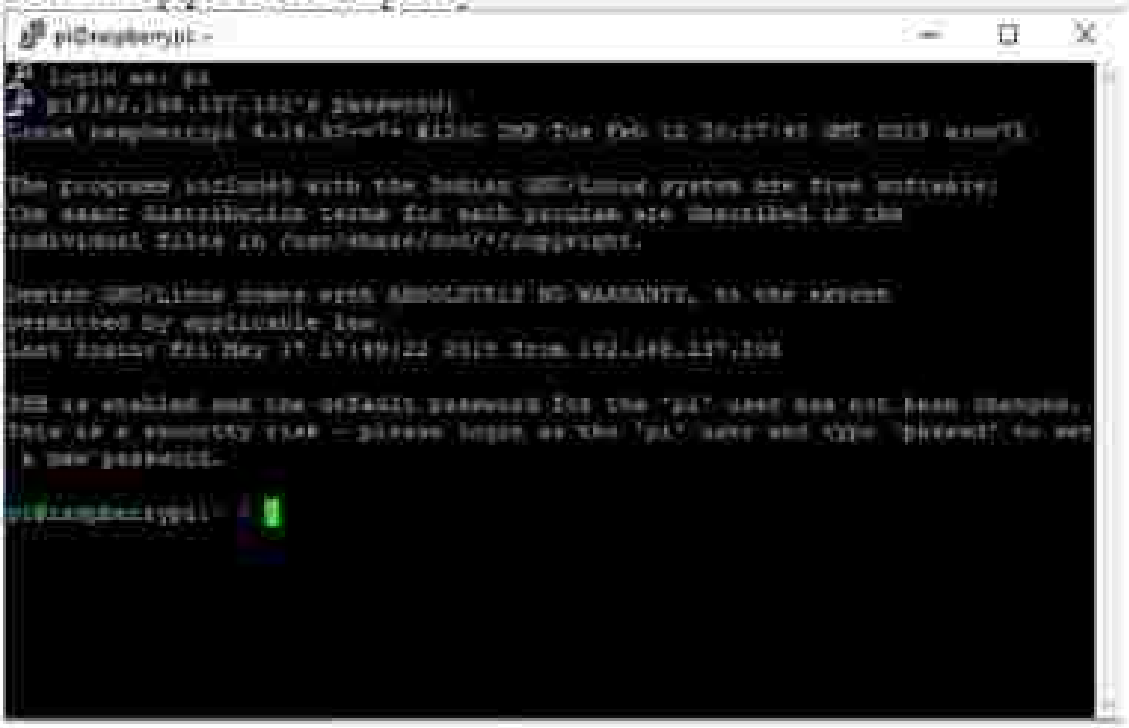
- Before switching on your raspberry pi, we need to find out the existing devices connected to the network. Make sure your laptop is connected to the same WIFI network as the one you configured on pi above.
- Download the Advanced IP Scanner to scan the IP of our raspberry pi. You can download it from here https://www.advanced-ip-scanner.com/



7) SSH into your Pi:

- To create a secured shell connection in Linux we can use the ssh command. If you are on windows, try downloading Pi tty from https://www.putty.org/

Default credentials are:
username: pipassword: raspberry



8) Access Pi remotely:

- Sometimes it doesn't feel right if we can't use the mouse. For that, we need to look into the Raspbian desktop

- We need to setup VNC (Virtual Network Connection) to see and control Pi graphically. Let's do that.

- To access the remote desktop, you need VNC-viewer (client) for your laptop. Fortunately, RealVNC is available for a lot of OSes, pick one for your OS
  from https://www.realvnc.com/en/connect/download/viewer/

9) Commands for vncserver:

10) Now open VNC Viewer on your remote computer:

5. Implementation and Working:

5.1. Wireless communication of Arduino to Arduino with nRF24L01:

- In this, we will learn how to make wireless communication between two Arduino boards using the NRF24L01. And measure distance with ultrasonic sensor and transmit it to another Arduino with transceiver module.

Wiring Instructions:

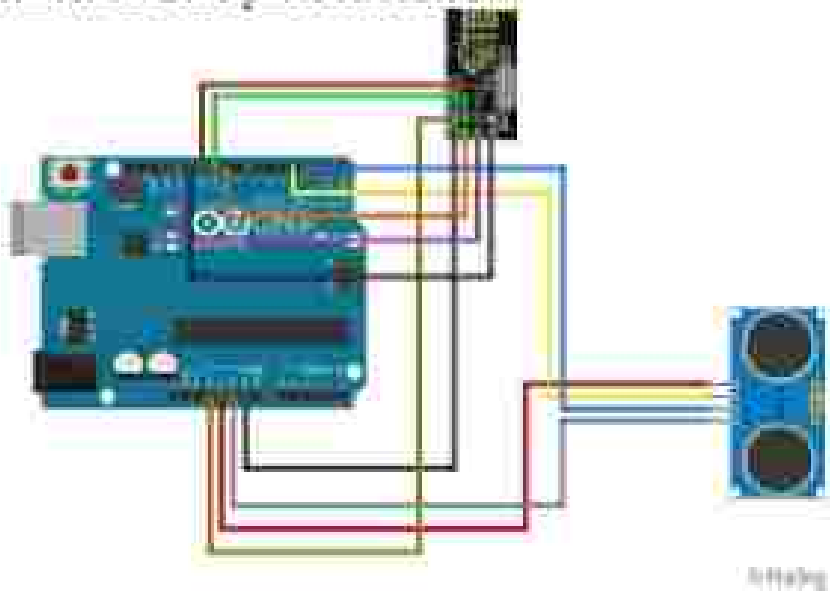
To wire your NRF24L01+ wireless Sender to your Arduino, connect the following pins:

- Connect the VCC pin to 3.3 Volts
- Connect the GND pin to ground (GND)

- Connect the CE pin to Arduino 9
- Connect the CSN pin to Arduino 10
- Connect the SCK pin to Arduino 13
- Connect the MOSI pin to Arduino 11
- Connect the MISO pin to Arduino 12

To wire your ultrasonic sensor to your Arduino, connect the following pins:

- Connect the VCC pin to Arduino 5Volts
- Connect the GND pin to ground (GND)
- Connect the Trig pin to Arduino 4
- Connect the Echo pin to Arduino 3



Schematic Diagram for wiring of Arduino Uno with ultrasonic sensor and NRF24L01

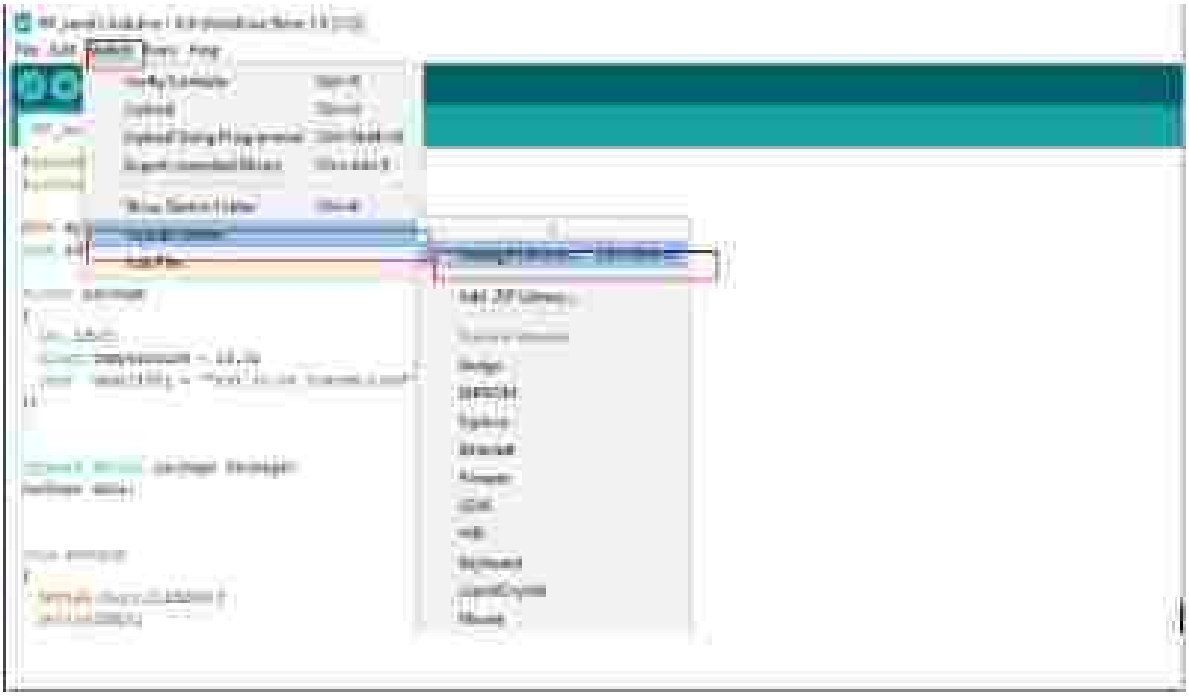
To wire your NRF24L01+ wireless sender to your Arduino, connect the following pins:

- Connect the VCC pin to 3.3 Volts
- Connect the GND pin to ground (GND)
- Connect the CE pin to Arduino 9
- Connect the CSN pin to Arduino 10
- Connect the SCK pin to Arduino 13
- Connect the MOSI pin to Arduino 11
- Connect the MISO pin to Arduino 12

Schematic Diagram for wiring of Arduino Uno NRF24L01

NOTE: RF24 module is mandatory for the code to run so you can add the library accordingly

- Start Arduino IDE then add the Downloaded Library from Here :



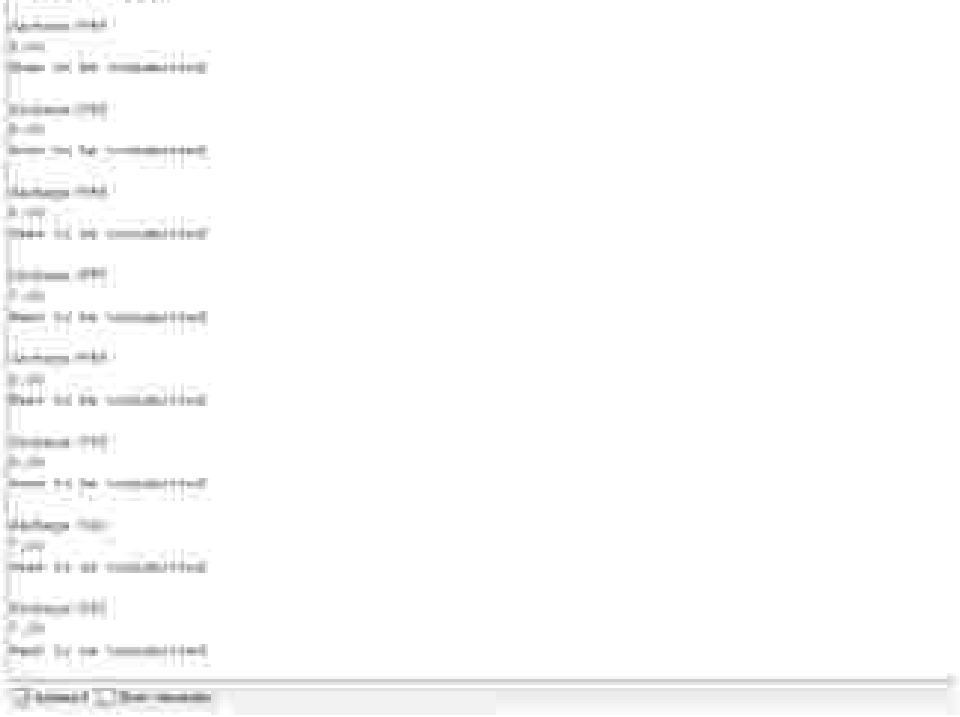
5.2. Code:

Sender Side code:

Receiver Side code:

Sending the Data

Receiving the data:



6. Wireless communication of Arduino to Raspberry Pi with nRF24L01:

6.1: Installation of RF24 Module on Raspberry Pi:

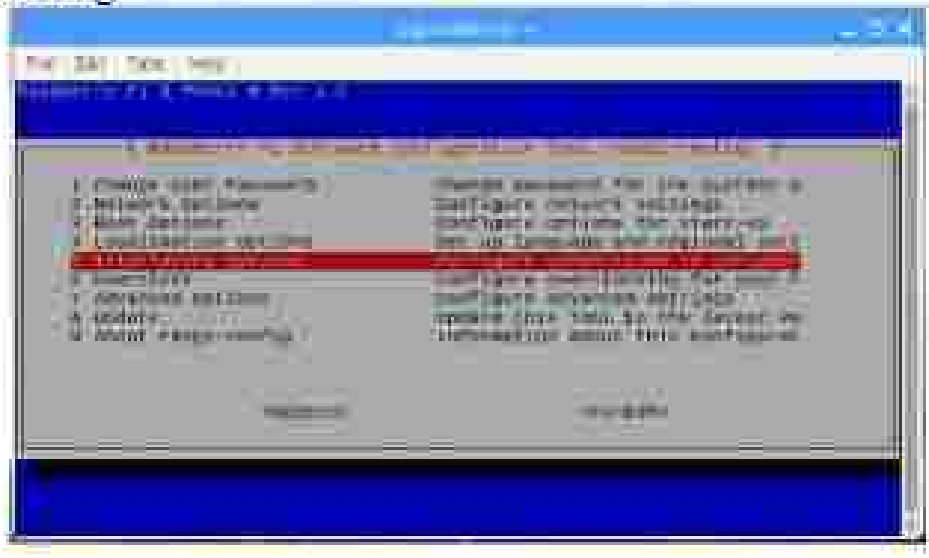
- It is the most important and foremost step for any Communication to work between Arduino and Raspberry Pi as we have used RF24 library in Arduino for communication so the same Library is needed on Pi.
- Further are the steps to which involve the installation of the Library. It took me almost one week to install it as no clear idea about it is present.

=> Way to go:

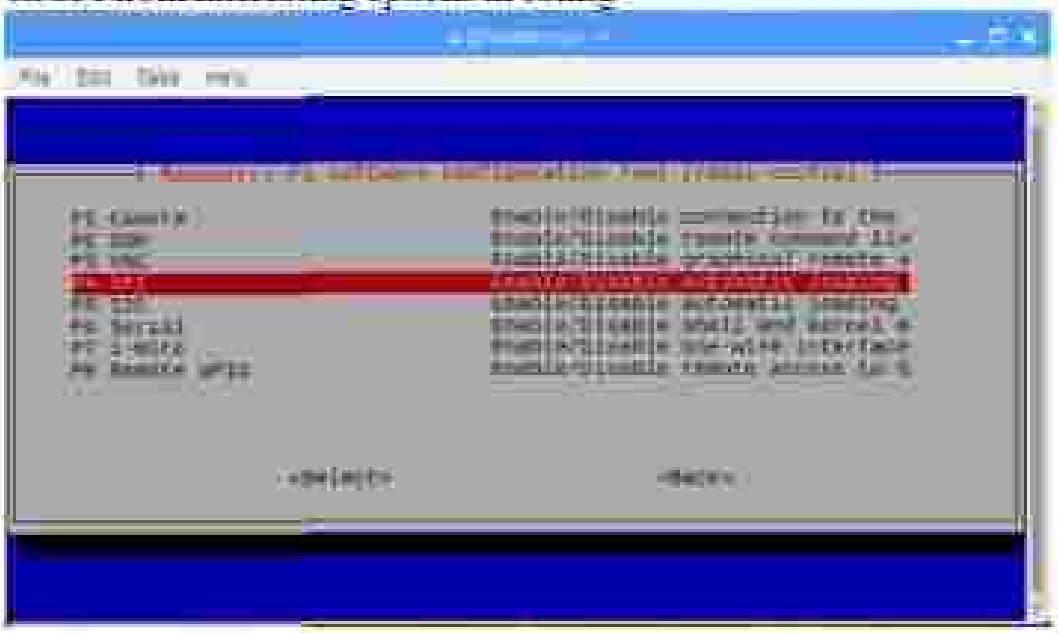
1. Login to Raspberry Pi using Putty.

2. Go to the VNC server for GUI.

3. In the terminal type:
sudoraspi-config



Turn on SPI from Interfacing options in config



4. Reboot the Pi. In the terminal, type:

```
sudo reboot
```

5. In the terminal type:

```
sudo apt-get update
```

6. Download the install.sh file

from http://tmrh20.github.io/RF24Installer/RPi/install.sh or Run this on terminal:

```
wgethttp://tmrh20.github.io/RF24Installer/RPi/install.sh
```

7. Make it executable:

```
chmod +x install.sh
```

8. Run it and choose your options:

9. Run an example from one of the libraries:

```
cd rf24libs/RF24/examples_linux
make
sudo ./gettingstarted
```

Run Following Commands to run the program.

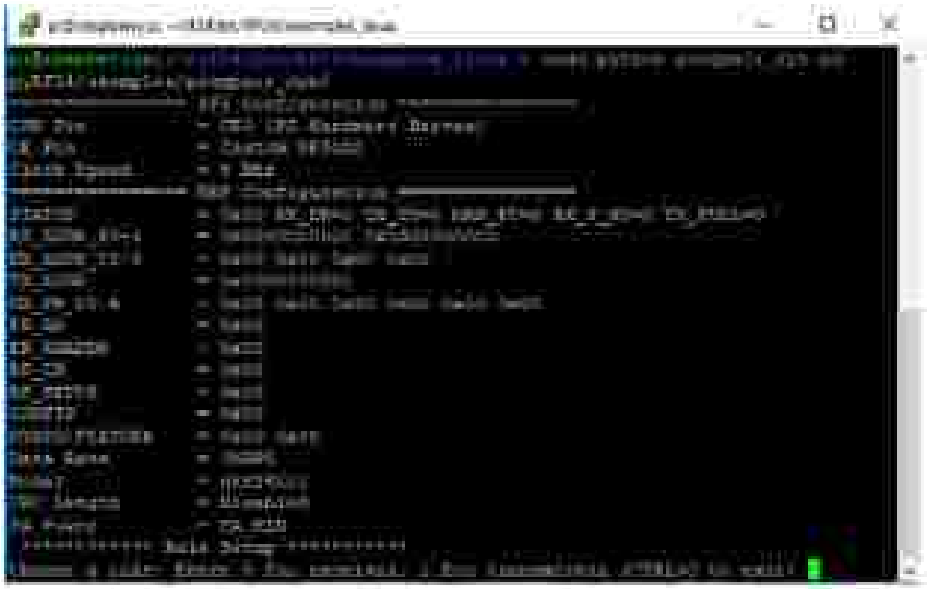
10. Further, if we want to run Python Programs for the same purpose, we can do this:

Running the Example Edit the pingpair_dyn.py example to configure the appropriate pins per the above documentation:

```
nano pingpair_dyn.py
```

- Configure another device, Arduino or RPi with the pingpair_dyn example
- Run the example

```
sudo python pingpair_dyn.py
```



6.2. Wireless communication of Arduino to Arduino with nRF24L01:

- In this, we will learn how to make wireless communication between Arduino and Raspberry Pi using the NRF24L01. And measure distance with an ultrasonic sensor with the help of Arduino Uno and transmit it to Raspberry Pi and Data is received.

Wiring Instructions:

To wire your NRF24L01+ wireless Sender to your Arduino, connect the following pins:

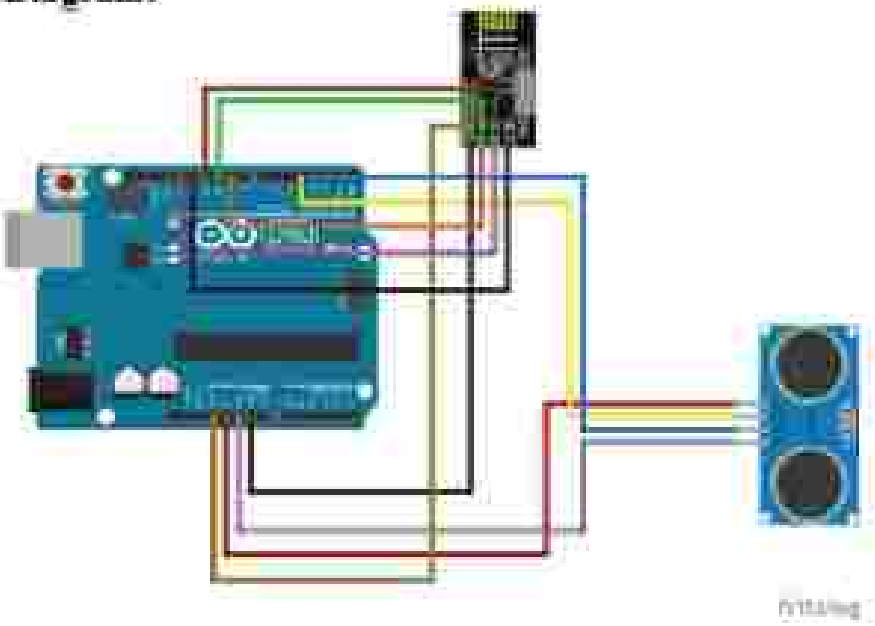
- Connect the VCC pin to 3.3 Volts
- Connect the GND pin to ground (GND)
- Connect the CSN pin to Arduino 10
- Connect the CE pin to Arduino 9

- Connect the SCK pin to Arduino 13
- Connect the MISO pin to Arduino 12
- Connect the MOSI pin to Arduino 11

To wire your ultrasonic sensor to your Arduino, connect the following pins:

- Connect the VCC pin to Arduino 5Volts
- Connect the GND pin to ground (GND)
- Connect the Trig pin to Arduino 4
- Connect the Echo pin to Arduino 3

Schematic Diagram:

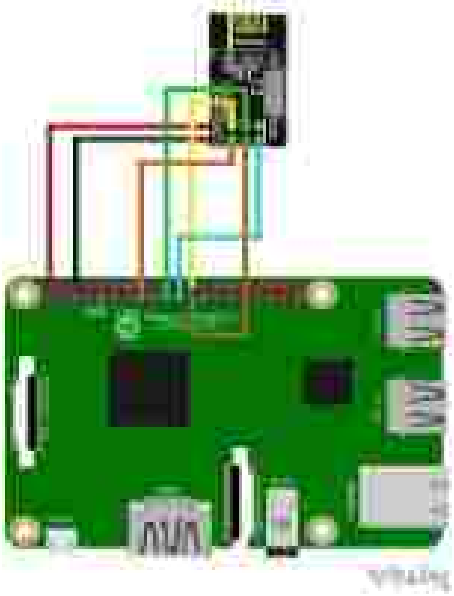


Schematic Diagram for wiring of Arduino Uno with ultrasonic sensor and NRF24L01

To wire your NRF24L01+ Wireless Receiver to your Raspberry Pi, connect the following pins:

- Connect the VCC pin to 3.3 Volts (Pin 1)
- Connect the GND pin to ground (GND) (Pin 6)
- Connect the CE pin to Raspberry GPIO 22
- Connect the CSN pin to Raspberry GPIO 8
- Connect the SCK pin to Raspberry GPIO 11
- Connect the MOSI pin to Raspberry GPIO 10
- Connect the MISO pin to Raspberry GPIO 09

Schematic Diagram:



Schematic Diagram for wiring of Raspberry Pi and NRF24L01

6.3. Code:

Sender Side Code:

Receiver Side Code:

- It's not mandatory to use this code as it is tweaked by me as per my requirement.

To check the proper functioning of your connection and code you can run the examples present in the library like pingpair_dyn.ino on your Arduino and pingpair_dyn.py on Raspberry Pi

- The snippet of my running communication:

Conclusion: It will be always fun experimenting and playing with the IoT Devices such as learning about Headless Raspberry Pi set-up, Arduino, and Raspberry pi by making them communicating with each other and sending data and to overcome the errors and challenges like one I faced while installing RF24 module. The purpose of this tutorial is to serve you with a step-by-step process and hope that it was easy to follow and learn as well.

PARTICULARS	MARKS ALLOTTED	MARKS OBTAINED
PERFORMANCE		
VIVA-VOCE		
RECORD		
TOTAL		

RESULT:

EXPT. 10 SETUP A CLOUD PLATFORM TO LOG THE DATA

Aim:

To setup a cloud platform to log the data

STEPS INVOLVED:

Step 1: Prepare Raspberry Pi

1. Download the operating system for RPi. I recommend a full version of Raspbian, like this here .
2. Assume that you have a working computer running on Windows, download Rufus software to write the image to the SD card. Scroll down and get rufus 3.0 (June 2018)
3. Unzip Raspbian file (4.6 GB), and carefully select your target SD card. It takes about 3 minutes to complete writing the image to the card.
4. Insert the SD card the slot, plug in power cable to the micro USB port, a Display with HDMI cable, a keyboard, a mouse is optional.
5. Wait until the RPi boot up and show the desktop, press Window key, go to Accessories Terminal and Enter.
6. Configure RPi by type:

```
sudoraspi-config
```

7. Change couple of thing in RPi configuration:

- In 1. Change password
- In 2. Network Option: Change hostname (something unique, mine is rpi1001), and wifi SSID and password.
- In 4. Localisation Options: Change Timezone, Keyboard, locale
- In 5. Interfacing Options: Enable SSH (to log in via command line), Enable VNC (for desktop remote)
- In 7: Advance Option: Expand Filesystems.
- Reboot
- After booting up: note IP address by running

```
sudoifconfig
```

If Ethernet is using, the IP should the be first block, if Wifi is used, the IP is on the third block, something like 192.168.1.40, run this command to update the Linux distro:

```
sudo apt-get update &&sudo apt-get upgrade –y &&sudopoweroff
```

The last one will power off the RPi after done with the updates

8. If you forgot the note the IP of RPi, or it is changed recently, use PowerShell (type PowerShell to search box in Windows)

In PowerShell to ping the Rpi: *ping rpi1001.local –4* get something like this 192.168.1.40. rpi1001 is my hostname for my Rpi.

9. Install VNCViewer , this software works like TeamViewer, or Desktop Remote on Windows (only Win 10 Pro has Desktop Remote feature).

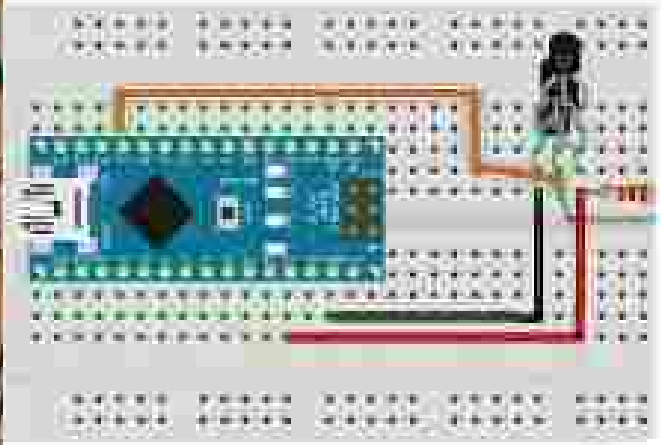
Install on your Windows machine, in the search box on the top of VNCViewer, type in the RPi's IP (**192.168.1.40**) or RPi's hostname (mine is **rpi1001.local**) and Enter. Enter your name RPi's password, select 'Remember password' if desired so. If everything went well, you should see the pop-up Rpi desktop

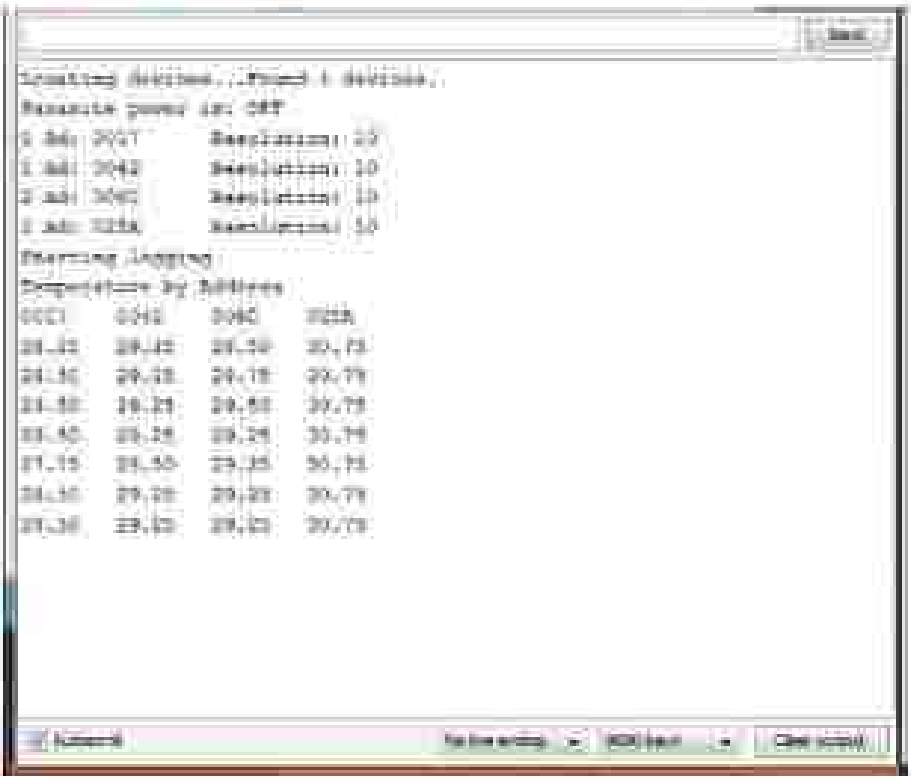
10. To transfer file from or to TeamView, the simplest way to use built-in file transfer by VNCView:

And that it, you can use a Raspberry to collect data for you, and log in to get data when needed.

Add TipAskQuestionCommentDownload

Step 2: Prepare Simple Example to Collect Data From Arduino





Let say you want to collect the temperature from 4 sensors. In this example, I used 18B20, a popular temperature sensor. Other options are the TMP35,36 family or a thermistor.

The wiring is included above. The 18B20s share the wire (or bus), and here is the Arduino code on Github. The attachment in below file contains the codes and wiring map as well.

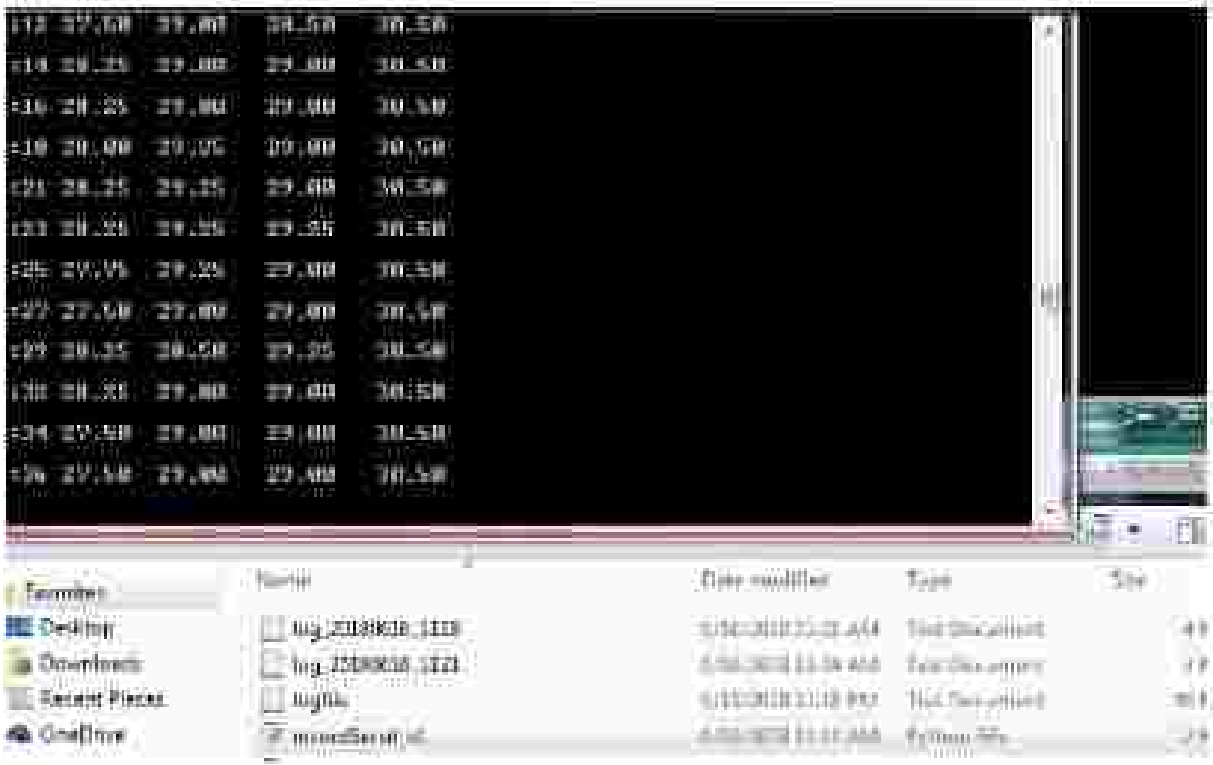
Also, install the USB driver for Arduino in Windows. Most 'clone' Arduino use CH341 USB driver. The driver is here.

Install driver, when plugging the Arduino to your Windows' USB, it should recognize the driver and assign a COM port (mine is COM4)

The Serial Monitor should output like the photo above.

Add TipAskQuestionCommentDownload

Step 3: Boot Up RPi by Unplug and Plug Power to RPi. Start VNCViewer, Log in Your RPi



Transfer the Python script, using VNCViewer tool. There is a banner on the near the top of the VNCViewer, look for two arrows button. To transfer data from the RPi to Windows, use the File Transfer in VNC symbol on the top-right (near the Wifi symbol) on the RPi's desktop.

Raspberry runs on Linux, and Python 2 and 3 are installed by default. You only need to install a Python library called pyserial to collect data print out from the serial line.

```
sudo apt-get install python3-serial
```

- in Windows: type this PowellShell:
- pip.exe install pyserial

Then download the script and save it to the RPi's desktop

To run the script, first make it executable by:

```
sudochownu+x recordSerial.py
```

Upload the Arduino code to the Arduino board, then plug your USB cable with USB into, check the port by:

- On Linux, type this to the terminal: *ls /dev/ttyUSB**
- On Windows: go to Device Manager, check COM #

If the script is modified on Windows, you may need to run the Python script to *dos2unix* to remove weird line ending character by Windows. Install it by

```
sudo apt-get install dos2unix
```

, and convert the script by running this in the terminal

```
dos2unix recordSerial.py
```

Modified the actual port in the script by a text editor:

```
sudo nano recordSerial.py
then run ./recordSerial.py yourfilename.txt
```

The Python script will save the data from the RAM to the disk for every 10 lines, which can be adjusted.

To stop recording, press Ctrl + C.

The script can be run on Windows (double click), the name of log data is the default which includes a timestamp

If you PowerShell, you could enter your customized filename,

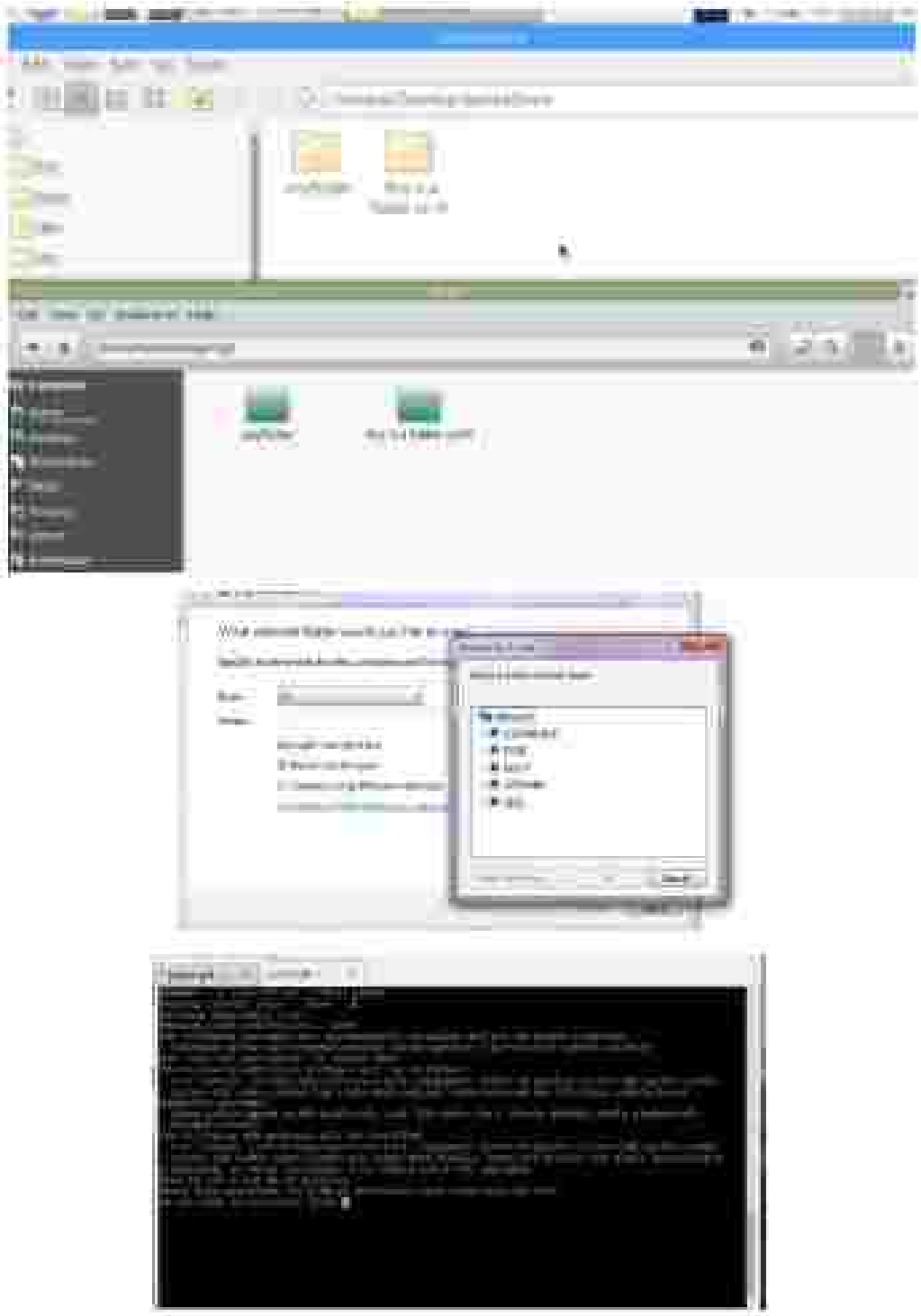
```
python.exe recordSerial.py awesome.txt
```

Not all 18B20s are the same. See the readout!

I pushed the code to GitHub as well.

Hope this tutorial is helpful!

Step 4: Install Samba to Share Folder



3 More Images

This step will walk you through some basic setup to have a shared folder hosted on Raspberry Pi that can be accessed to from other computers.

First, install samba, a program to share and manage sharing folder across the network:

```
sudo apt-get install samba
```

make a shared folder

```
mkdir ~/Desktop/sambaShare
```

modify the configuration file for the samba by:

```
sudo samba /etc/samba/smb.conf
```

add the following lines to the end of the file:

```
[sambaShare]<br> comment = share Folder on Research RPI
  path = /home/pi/Desktop/sambaShare
browseable = yes
  read only = no
  writable = yes
  public = yes
  create mask = 0777
  rectory mask = 0777
  guest ok = yes
```

If you have the problem with writing on windows, force it by adding this line to the end of the file: **force user = pi**

On Linux, you may need to use the root user (I will post the problem once I found out)

Next, add a user to samba and create a password as well:

```
sudosambpasswd -a pi
```

then enter a password for the samba (can be the same or different than the password for pi user on the system)

test if the config file is okay:

```
testparm
```

press Ctrl+X to save, and then restart samba service by:

```
sudosystemctl restart smbd
```

On the host computer, let say Linux:

if not yes install samba plus sambaclient, and cifs to support the share drive, please do so by running:

```
sudo apt-get installs samba smbclientcifs-utils
```

Check if the shared folder on RPI is ready by:

```
sudosmbclient -L yourRPI_IP
```

if you see the share drive, then create a mount point on Linux:

```
sudomkdir /mnt researchRPi
sudochownuser:usergroup -R /mnt researchRPi
```

user, usergroup is your Linux user and group name

then mount the share by:

```
sudo mount -f cifs -o username=pi //your_rpi_IP sambaShare /mnt researchRPi
```

enter your passwords, and make a soft link to your desktop:

```
sudo ln -s /mnt researchRPi ~Desktop researchRPi
```

if you have problem with read-write permission on the share folder, experiment with a weak permission:

on PI:

```
sudochmod -R 776 ~Desktop sambaShare
```

On Windows, it is even easier,
Go to My Computer, and map a folder, then enter the IP of RPI, the shared folder should appear. I believe there is something on Mac that you can browse the shared folder on the network.

PARTICULARS	MARKS ALLOTTED	MARKS OBTAINED
PERFORMANCE		
VIVA-VOCE		
RECORD		
TOTAL		

RESULT:

EXPT. 11. LOG DATA USING RASPBERRY PI AND UPLOAD TO THE CLOUD PLATFORM

AIM:

To Log Data using Raspberry PI and upload to the cloud platform

Descripton:

The software running on Raspberry Pi is like any (production) system which benefits from proper logs. Logs are crucial to debugging, monitoring, alerting, and just keeping the systems up and running without interruptions. Depending on your setup, you might have several Raspberry Pis (other any IoT devices) so the logs from all the devices need to be aggregated and accessible in a single place.

There are many alternatives for centralized logging and Google Cloud Logging is one of the options. But why choose it? Well, it's practically free at least on a small scale and it provides centralized logging with easy integration. You can also easily create log-based metrics and alerts based on these which provides visibility to your devices.

Create a service account

First, you need to create a service account to allow writing the logs from Raspberry Pi. You can create the service account from GCP console: https://cloud.google.com/docs/authentication/production The "Log Writer" permissions are needed at least.

Once the service account is ready, you need to create a new key and download it as a JSON file. This file contains the authentication credentials for the logging driver and you need to provide the JSON file to it by setting the

GOOGLE_APPLICATION_CREDENTIALS environment variable. I'm running the application using Docker, so the easiest way to configure Docker using systemd. Many Linux distributions use systemd to start the Docker daemon so you can override the environment variables by adding configuration file to /etc/systemd/system/docker.service.d/.

Example: /etc/systemd/system/docker.service.d/service-account-env-variables.conf:

```
[Service]
Environment="GOOGLE_APPLICATION_CREDENTIALS=/path/to/service-account.json"
```

This will pass the correct environment variable for all the Docker containers and to the logging driver. Remember to restart the Docker after this:

```
sudo systemctl daemon-reload
sudo systemctl restart docker
```

Using the Google Cloud Logging driver

There are different ways of uploading the logs to GCP. An easy way is to use the Google Cloud Logging driver which can set as a default logging driver for the Docker containers. You can do it with the --log-driver option:

```
docker run --log-driver=gcplogs ...
```

Or just by adding the following JSON to /etc/docker/daemon.json so this logging driver will be used by all the Docker containers.

```
{
  "log-driver": "gcplogs",
  "log-opts": {
    "gcp-meta-name": "10000000c8ac1791",
    "gcp-project": "my-gcp-project",
    "mode": "non-blocking",
    "max-buffer-size": "2m"
```

```
  }
}
```

gcp-meta-name specifies the name of the instance where the logs are coming from. If you have multiple Raspberry Pi devices sending logs to the same log bucket, you can use Raspberry Pi's serial number to differentiate the devices. Run the following command on Raspberry Pi and add the output as gcp-meta-name.

```
cat /proc/cpuinfo | grep Serial | cut -d' ' -f2
```

Using Fluentd

Another way is to use Fluentd, which can forward the logs to various places, including to Google Cloud Logging.

Setting up Fluentd requires a bit more work though. I decided to build a separate Docker image for Fluentd which contains the necessary configurations.

I'm using the ARMv7 specific image of Fluentd as the base image and install fluent-plugin-google-cloud output plugin to it. This plugin allows pushing the logs to Cloud Logging API.

I also got plenty of permission issues with the Fluentd plugins unless I gave updated permission to all of them. Didn't really understand why this happens but changing the file permissions to 644 fixed the issue.

```
FROM fluent/fluentd:v1.11.2-debian-armhf-1.0USER rootRUN apt-get update
RUN apt-get install -y make gcc g++ libc6-dev ruby-dev libffi-dev \
      ca-certificates \
      liblz4-1 \
ruby:RUN echo 'gem: --no-document' >> /etc/gemrc
RUN gem install fluent-plugin-google-cloud -v 0.6.25:1RUN chmod 644
/usr/local/bundle/gems/fluent-plugin-google-cloud-
```

```
0.6.25.1/lib/fluent/plugin/* /vCOPY ./config/fluent/fluent.conf
/fluentd/etc/fluent.confUSER fluent
```

The configuration file is quite simple. I'm also using the JSON filter plugin as the applications are logging everything as JSON:

```
<source>
  @type forward
  port 24224
  bind 0.0.0.0
</source><filter **>
@type parser
key_name log
reserve_data true
<parse>
    @type json
</parse>
</filter>
<filter **>
  @type add_insert_ids
</filter><match **>
  @type google_cloud
enable_metadata_agent false
vm_id "<GCP_PROJECT_ID>"
zone "<GCP_ZONE>"
split_logs_by_tag false
use_metadata_service false
detect_json true buffer_type file
buffer_path /fluentd/log/fluentd.buffer
buffer_queue_full_action block num_threads 2
use_grpc true
</match>
```

The docker-compose files starts the application container(s) and the Fluentd instance:

```
services:
fluentd:
    image: myfluentimage
container_name: fluentd
    ports:
      - "24224:24224" mycontainer:
```

```
    image: mycontainer
container_name: mycontainer
depends_on:
      - fluentd
    logging:
      driver: fluentd
      options:
fluentd-address: 0.0.0.0:24224
fluentd-async-connect: "tr e"
```

There you go! Logs are automatically uploaded from the Raspberry Pi and available on GCP:

PARTICULARS	MARKS ALLOTTED	MARKS OBTAINED
PERFORMANCE		
VIVA-VOCE		
RECORD		
TOTAL		

RESULT:

EXP.NO 12: DESIGN AN IOT BASED SYSTEM

AIM:

Task: A LED and a piezo speaker are supposed to blink or beep continuously

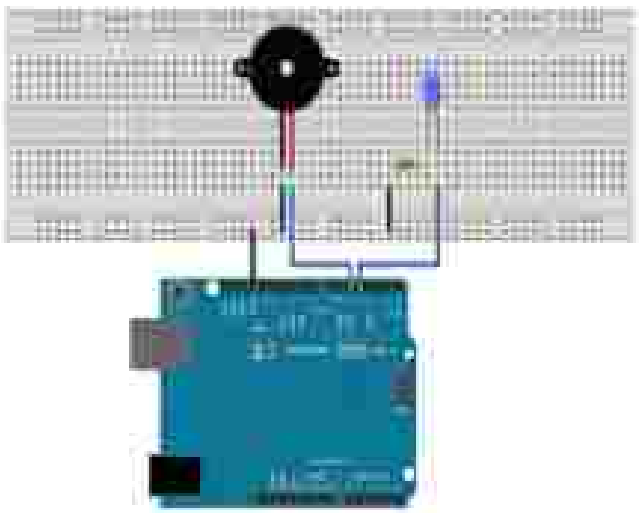
ALGORITHM:

1. Initialize the LED and piezo speaker pins as OUTPUT.
2. Enter a loop that runs indefinitely (the main loop).
3. Inside the main loop:
   a. Turn ON the LED.
   b. Generate a beep on the piezo speaker.
   c. Wait for a specific duration (blink and beep time).
   d. Turn OFF the LED.
   e. Stop the piezo speaker sound.
   f. Wait for a specific duration (the interval between blinks and beeps).
   g. Go back to step 3 (repeat the loop).

COMPONENTS REQUIRED:

Required equipment: Microcontroller / one LED / resistor with 200 Ohm / Breadboard / piezo speaker / cables

Setup:



CODE :

```
int LED=4; //this time we also going to use the first part of the program. Here
//we are going to put in variables. This means that there will be a letter or a
//word standing for a number. In this example the LED is connected to pin 4 and
//the speaker to pin 5, so we rename pin 4 and pin 5, to avoid confusion. The
//word "LED" now stands for the number 4 and the word "beep" for the number 5.
int beep=5;
void setup()
{ //We are starting with the setup
pinMode(LED, OUTPUT); //pin 4 (pin "LED") is supposed to be an output
pinMode(beep, OUTPUT); //Pin 5 (pin "beep") is supposed to be an output
}
void loop()
{ //The main part starts
digitalWrite(LED, HIGH); //turn on the LED
digitalWrite(beep, HIGH); //turn on the speaker
delay(1000); //wait for 1000 milliseconds (sound and light)
digitalWrite(LED, LOW); //turn off the LED
digitalWrite(beep, LOW); //turn off the speaker
delay(1000); //wait for 1000 milliseconds (no sound and no light)
} //Here at the end of the loop the program starts again from the beginning of
```

//the loop. So it will beep and light up again. If you change the break (delay)

//it will be either beep and light up faster or slower.

PARTICULARS	MARKS ALLOTTED	MARKS OBTAINED
PERFORMANCE		
VIVA-VOCE		
RECORD		
TOTAL		

RESULT: