

Index

Sl.No	Experiment Name	Category	Page No
1	Transmit a string using UART	Serial Communication	3-5
2	Point-to-Point communication of two Motes over the radio frequency.	RF Motes Integration	6-9
3	Multi-point to single point communication of Motes over the radio frequency.LAN (Subnetting).	RF Motes Integration	10-13
4	I2C protocol study	Serial Communication	14-19
5	Reading Temperature and Relative Humidity value from the sensor	DHT11 Sensor Integration	20-23

Hardware List

Sl.No	Components
1	Arduino UNO Boards
2	RF Transmitters/Receivers
3	DHT11 Temperature & Humidity Sensor
4	LCD Display
5	LED Lights
6	Bread Board
7	Connecting Wires

Experiment No 1: Transmit a string using UART.

Aim:

A **UART's** (Universal Asynchronous Receiver/Transmitter) main purpose is to transmit and receive serial String data using serial communication.

Objective:

To learn how serial communication is achieved using UART .(serial string data).

Components Required:

- 1.Arduino UNO.
- 2.Aduino UNO USB cable.

Connections:

Arduino UNO to be connect with computer through USB cable.

Procedure :

1. Open the Arduino UNO IDE and create and save a sketch source code as follows.
2. Compile sketch.
3. 3.Upload sketch on to Arduino UNO.

Source Code:

```

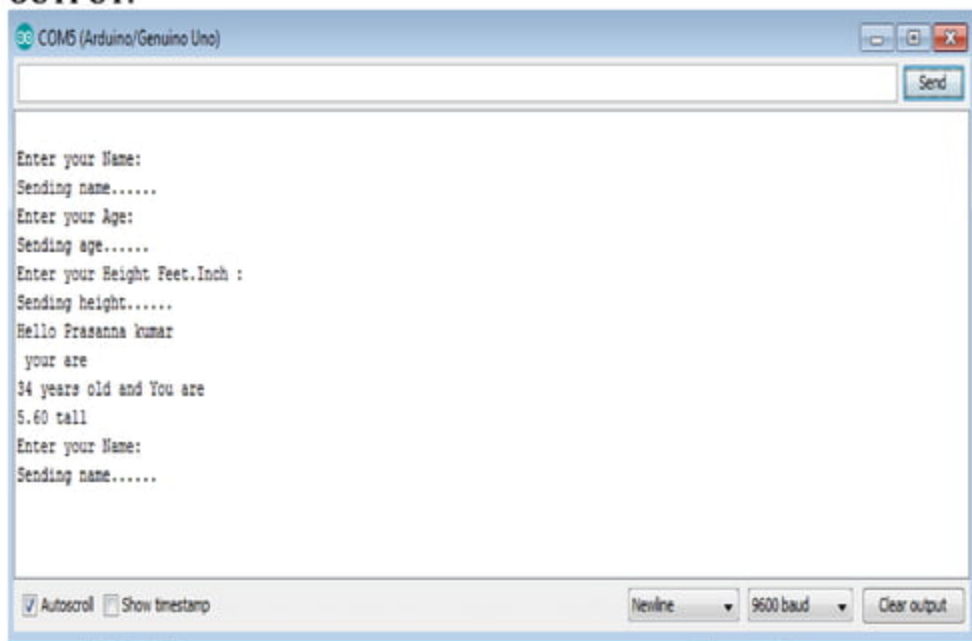
String name;
int age;
float height;
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
}
void loop() {
  // put your main code here, to run repeatedly:
  Serial.println("\nEnter your Name:");
  while(Serial.available() == 0)
  {

  }
  name=Serial.readString();
  Serial.println("Sending name.....");
  delay(10000);
  Serial.println("Enter your Age:");
  while(Serial.available() == 0)
  {

  }
  age=Serial.parseInt();
  Serial.println("Sending age.....");
  delay(10000);
  Serial.println("Enter your Height Feet.Inch :");
  while(Serial.available() == 0)
  {

  }
  height=Serial.parseFloat();
  Serial.println("Sending height.....");
  delay(5000);
  display();
}
void display()
{
  Serial.print("Hello ");
  Serial.print(name);
  Serial.println(" your are ");
  Serial.print(age);
  Serial.print(" years old and");
  Serial.println(" You are ");
  Serial.print(height);
  Serial.print(" tall");
}

```

OUTPUT:**Applications :**

UARTs are used for devices including GPS units, modems, wireless communication and Bluetooth modules, amongst many other applications.

Conclusion:

Serial data communication is achieved by transmitting and receiving string data using UART.

Experiment No 2. Point-to-Point communication of two Motes over the radio frequency.

Aim:

To communicate two Motes node over the radio frequency.

Objective:

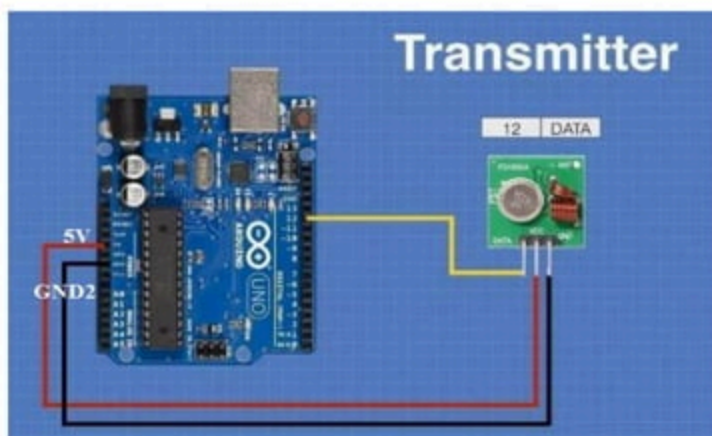
To learn how wireless communication is achieved via point to point communication of two motes using RF Transmitter/ Receiver.

Components Required:

1. Arduino UNO's (Two).
2. RF transmitter 433MHz.
3. RF receiver 433MHz
4. Connecting wires

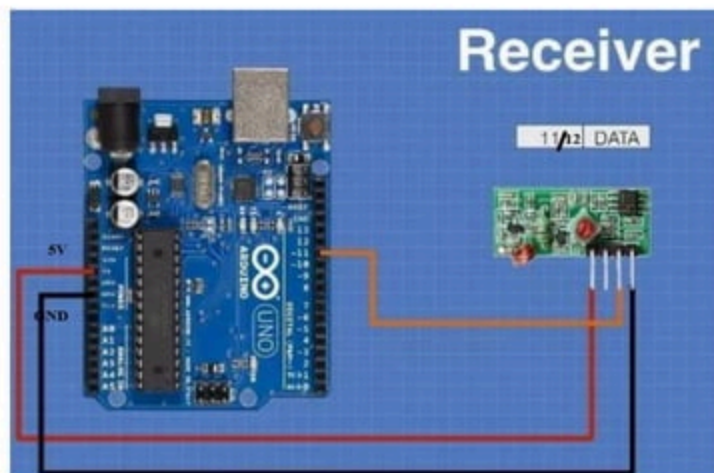
Arduino Sender Connections:

Arduino Pins	RF Transmitter
12	Data
5V	VCC
GND2	GND



Arduino Receiver Connections:

Arduino Pins	RF Receiver
11	Data
5V	VCC
GND2	GND



Procedure :

1. Open the Arduino UNO IDE and create and save a sketch source code of transmitter and receiver.
2. Compile the sketches.
3. Upload sketches on to Arduino UNO's through connected COM ports.

Source Code of Reciever:

```
#include <RH_ASK.h>
#include <SPI.h> // Not actually used but needed to compile

RH_ASK driver;

void setup()
{
  Serial.begin(9600); // Debugging only
  if (!driver.init())
    Serial.println("init failed");
}

void loop()
{
  uint8_t buf[12];
  uint8_t buflen = sizeof(buf);
  if (driver.recv(buf, &buflen)) // Non-blocking
  {
    int i;
    // Message with a good checksum received, dump it.
    Serial.print("Message: ");
    Serial.println((char*)buf);
  }
}
```

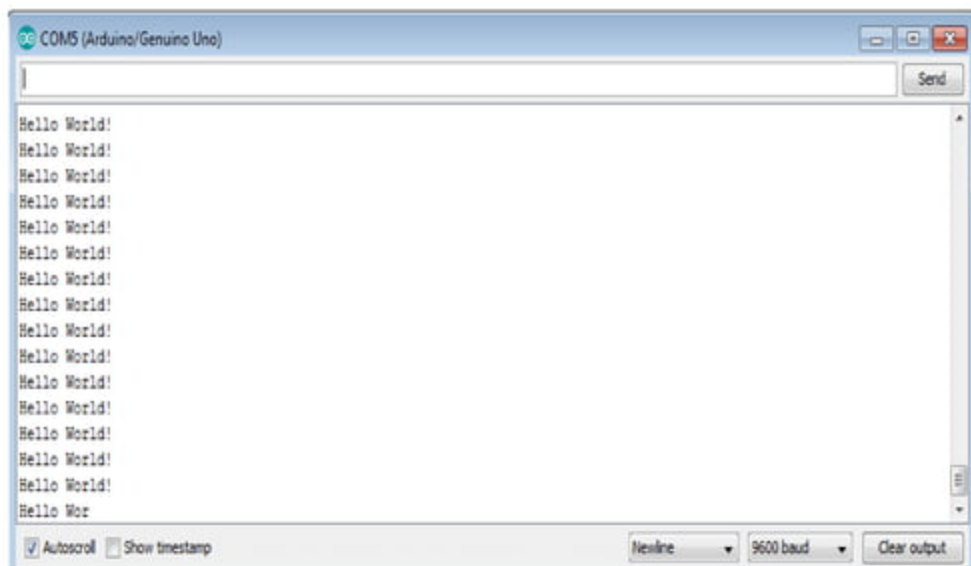
Source Code of Transmitter:

```
#include <RH_ASK.h>
#include <SPI.h> // Not actually used but needed to compile

RH_ASK driver;

void setup()
{
  Serial.begin(9600); // Debugging only
  if (!driver.init())
    Serial.println("init failed");
}

void loop()
{
  const char *msg = "Hello World!";
  driver.send((uint8_t *)msg, strlen(msg));
  driver.waitPacketSent();
  delay(1000);
}
```


OUTPUT:**Applications :**

RF technology, the information can be transmitted through the air without requiring any cable or wires or other electronic conductors, by using electromagnetic waves like IR, RF, satellite, etc. a variety of wireless communication devices and technologies ranging from smart phones to computers, tabs, laptops, Bluetooth Technology, printers.

Conclusion:

Wireless communication is achieved by RF transmitter and RF receiver message passed in the frequency range of 433MHZ.

Experiment No 3. Multi-point to single point communication of Motes over the radio frequency. LAN (Subnetting).

Aim:

To study and analysis of the Wireless Communication of Motes over the radio frequency through multi-point to single point in a network.

Objective:

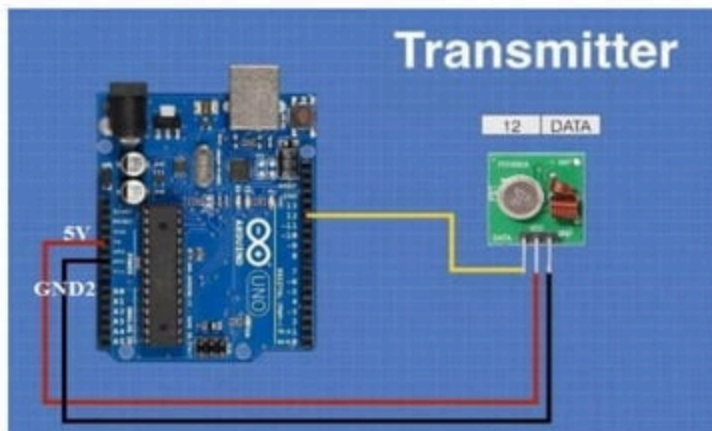
Point-to-multipoint (PMP) communication refers to communication that is accomplished through a distinct and specific form of one-to-many vice versa connections, offering several paths from one single location to various locations. Single transmitter broad casts message across the multiple receiver using wireless communication of motes through RF transmitter and receiver 433 MHZ in a network. Wireless communication is achieved.

Components Required:

Sl.No	Components Required	QTY
1	Arduino UNO Transmitter	1
2	Arduino UNO Receiver 1	1
3	Arduino UNO Receiver 2	1
4	Aduino UNO USB cables	3
5	RF Receiver 433 MHZ	2
6	RF Transmitter 433 MHZ	1
7	Connecting Wires	9

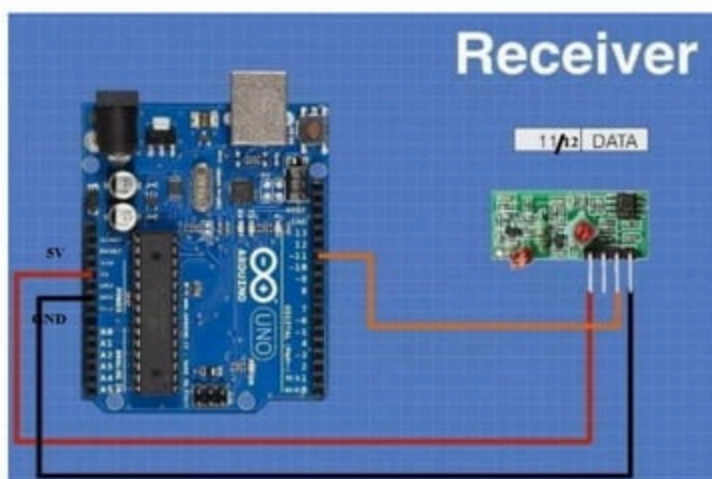
Arduino Sender Connections:

Arduino Pins	RF Transmitter
12	Data
5V	VCC
GND2	GND



Arduino Receiver 1 & 2 Connections:

Arduino Pins	RF Receiver
11	Data
5V	VCC
GND2	GND



Procedure :

1. Open the Arduino UNO IDE and create and save a sketch of Master transmitter and receiver 1 and receiver 2 source code.
2. Compile sketch of transmitter and receivers.
3. Upload sketch through COM ports (COM4, COM3, COM5).
4. See the output message on to the Serial monitor by using COM ports.

Source Code Sender:

```
#include <RH_ASK.h>
#include <SPI.h> // Not actually used but needed to compile
RH_ASK driver;

void setup()
{
  Serial.begin(9600); // Debugging only
  if (!driver.init())
    Serial.println("init failed");
}

void loop()
{
  const char *msg = "Welcome to IoT Laboratory";
  driver.send((uint8_t *)msg, strlen(msg));
  driver.waitPacketSent();
  Serial.println("Sender : Sending Data....");

  delay(1000);
}
```

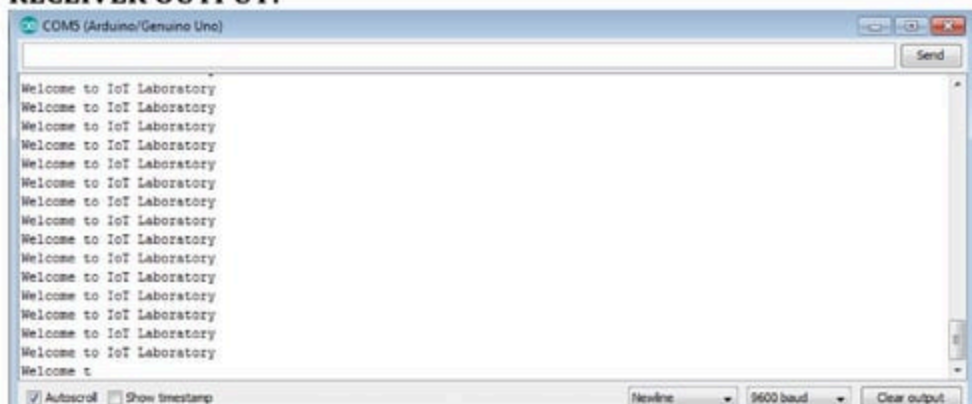
Source Code Multiple Receivers:

```
#include <RH_ASK.h>
#include <SPI.h> // Not actually used but needed to compile
RH_ASK driver;

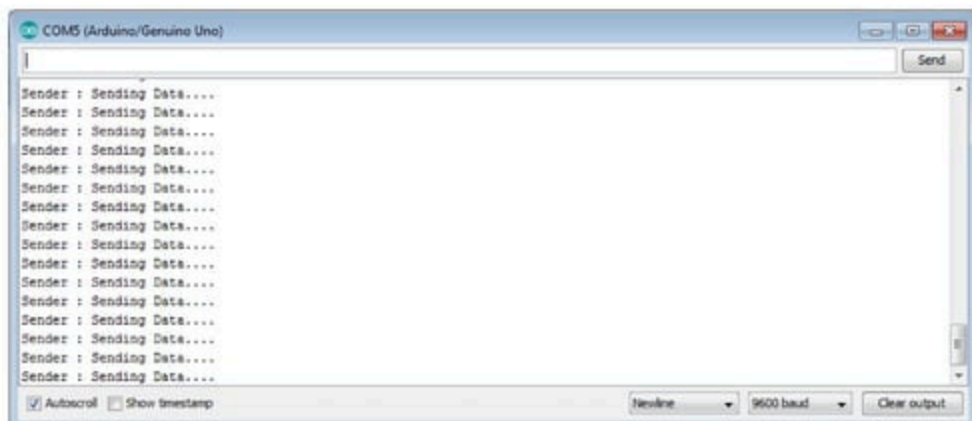
void setup()
{
  Serial.begin(9600); // Debugging only
  if (!driver.init())
    Serial.println("init failed");
}

void loop()
{
  uint8_t buf[20];
  uint8_t buflen = sizeof(buf);
  if (driver.recv(buf, &buflen)) // Non-blocking
  {
    int i;
    // Message with a good checksum received, dump it.
    Serial.print("Message: ");
    Serial.println((char*)buf);
  }
}
```

RECEIVER OUTPUT:



SENDER OUPUT:



Applications :

PMP wireless networks are employed in distribution amenities, huge corporate campuses, school districts, public safety applications, etc.

Conclusion:

Wireless Network communication is achieved through multi point to single point communication nodes over radio frequency by RF transmitter and receiver. Multiple receiver and a transmitter communicate each other by broadcasting message in a network via RF transmitter and receiver nodes in the frequency 433 MHz range.

Experiment No 4. I2C protocol study

Aim:

The study of I2C protocol to communicate (in turn) with many devices, or other boards, sensors networks.

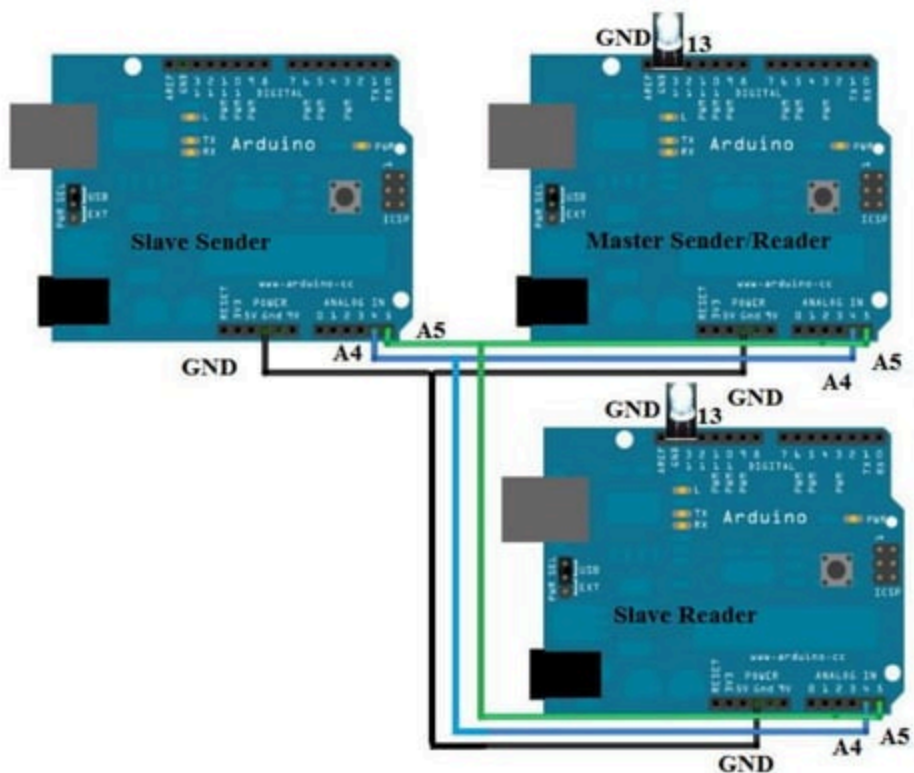
Objective:

In some situations, it can be helpful to set up two (or more!) Arduino and Genuino boards to share information with each other. In this example, two boards are programmed to communicate with one another in a Master Reader/Slave Sender configuration via the I2C synchronous serial protocol. Several functions of Arduino's Wire Library (<http://www.arduino.cc/en/Reference/Wire>) are used to accomplish this. Arduino 1, the Master, is programmed to request, and then read, 6 bytes of data sent from the uniquely addressed Slave Arduino. Once that message / a character is received, it can then be viewed in the Arduino Software (IDE) serial monitor window and LED will glow in both boards.

The I2C protocol involves using two lines to send and receive data: a serial clock pin (SCL) that the Arduino or Genuino Master board pulses at a regular interval, and a serial data pin (SDA) over which data is sent between the two devices. As the clock line changes from low to high (known as the rising edge of the clock pulse), a single bit of information - that will form in sequence the address of a specific device and a command or data - is transferred from the board to the I2C device over the SDA line. When this information is sent - bit after bit -, the called upon device executes the request and transmits it's data back - if required- to the board over the same line using the clock signal still generated by the Master on SCL as timing.

Components Required:

Sl.No	Components Required	QTY
1	Arduino UNO Master and PORT(COM4)	1
2	Arduino UNO Slave Sender and PORT(COM3)	1
3	Arduino UNO Slave Sender and PORT(COM3)	1
4	Aduino UNO USB cables	3
5	LED lights	2
6	Bread board	1
7	Connecting Wires	9



Connections:

Arduino Master Pins	Arduino Slaves Pins
A4	A4
A5	A5
GND	GND
Arduino Master Pins	LED
13	+Ve
GND	-Ve

Procedure :

1. Set up the connection as per diagram through bread board and connect LEDs to Master and Slave Reader.
2. Compile and upload sketch Arduino UNO Master Sender/Reader through COM4.
3. Compile and upload sketch Arduino UNO Slave Sender through COM3.
4. Compile and upload sketch Arduino UNO Slave Reader through COM5.
5. See the Output status message on Serial Monitor and LED light status ON (LED ONs) of Master Reader and Slave Reader UNO boards.

Master Source Code:

```

// Wire Master Reader
// Demonstrates use of the Wire library
// Reads data from an I2C/TWI slave device
// Refer to the "Wire Slave Sender/Receiver" example for use with this
#include <Wire.h>

void setup()
{
  Wire.begin();    // join i2c bus (address optional for master)
  Serial.begin(9600); // start serial for output
  pinMode(13,OUTPUT);
  digitalWrite(13,LOW);
  Wire.begin(5);    // join i2c bus with address #8
  Wire.onRequest(requestEvent); // register event
}

void loop()
{
  Wire.requestFrom(8, 6); // request 6 bytes from slave device #8
  while (Wire.available())
  {
    // slave may send less than requested
    char c = Wire.read(); // receive a byte as character
    // Serial.print(c);    // print the character
    if(c == 'H')
    {
      digitalWrite(13,HIGH);
      Serial.println("Command Accepted, Master- LED ON");
    }
    else if(c == 'L')
    {
      digitalWrite(13,LOW);
      Serial.println("Command Accepted LED OFF");
    }
  }
  delay(500);
}

void requestEvent()
{
  Wire.write('H'); // respond with message of 6 bytes
  // as expected by master
}

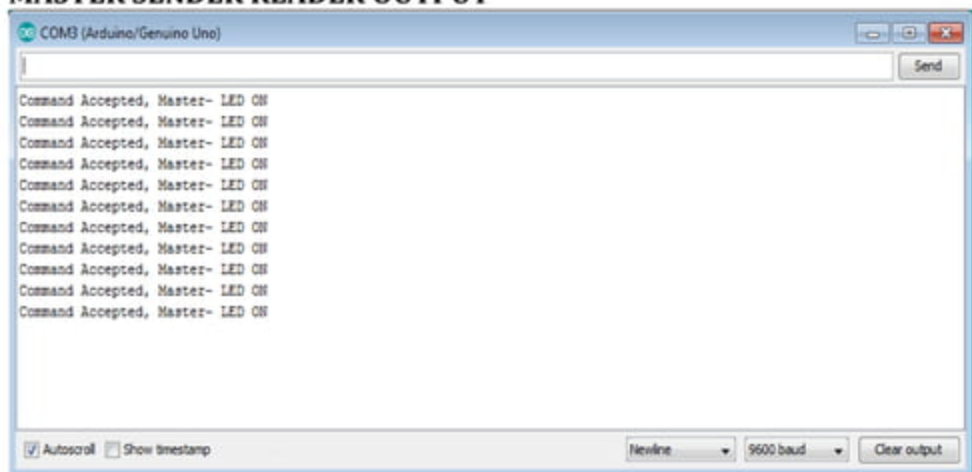
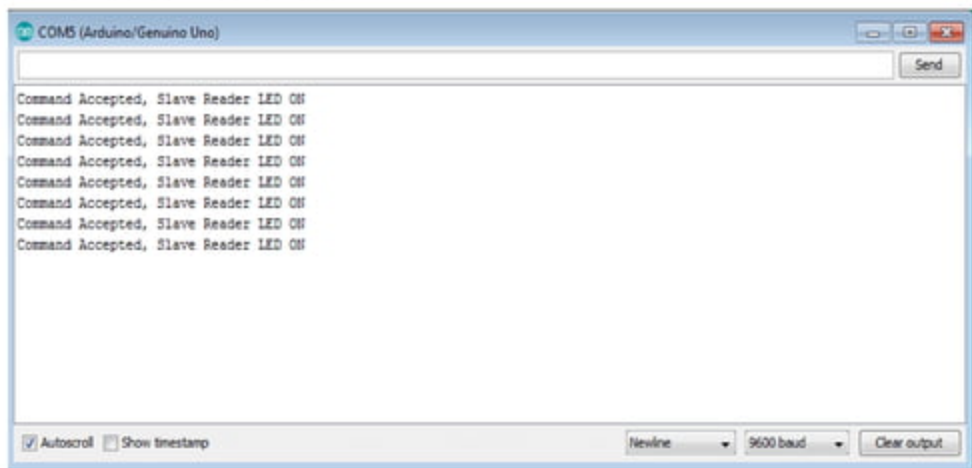
```


Slave Sender Source Code:

```
// Wire Slave Sender
// Demonstrates use of the Wire library
// Sends data as an I2C/TWI slave device
// Refer to the "Wire Master Reader" example for use with this
#include <Wire.h>
char c;
void setup()
{
  Wire.begin(8);          // join i2c bus with address #8
  Wire.onRequest(requestEvent); // register event
}
void loop()
{
  delay(100);
}
// function that executes whenever data is requested by master
// this function is registered as an event, see setup()
void requestEvent()
{
  Wire.write('H'); // respond with message of 6 bytes
  // as expected by master
}
```

Slave Reader Code:

```
// Wire Slave Reader
// Demonstrates use of the Wire library
// Reads data from an I2C/TWI master device
#include <Wire.h>
void setup()
{
  Wire.begin();          // join i2c bus (address optional for master)
  Serial.begin(9600);    // start serial for output
  pinMode(13,OUTPUT);
  digitalWrite(13,LOW);
}
void loop()
{
  Wire.requestFrom(5, 6); // request 6 bytes from slave device #5
  while (Wire.available())
  {
    // slave may send less than requested
    char c = Wire.read(); // receive a byte as character
    if(c == 'H')
    {
      digitalWrite(13,HIGH);
      Serial.println("Command Accepted, Slave Reader LED ON");
    }
    else if(c == 'L')
    {
      digitalWrite(13,LOW);
      Serial.println("Command Accepted, Slave Reader LED OFF");
    }
  }
  delay(500);
}
```

MASTER SENDER READER OUTPUT**SLAVE READER OUTPUT****Applications :**

The I2C protocol used to connect a maximum of 128 devices that are all connected to communicate with the SCL and SDL lines of the master unit as well as the slave devices. It supports Multi master communication, which means two masters are used to communicate the external devices.

Conclusion:

The I2C protocol allows for each enabled device to have its own unique address, and as both master and slave devices to take turns communicating over a single line, it is possible for your Arduino or Genuino board to communicate (in turn) with many devices, or other boards, while using just two pins of microcontroller.

Experiment No 5 Reading Temperature and Relative Humidity value from the sensor.

Aim:

Read the sensor data to measure Temperature and Relative Humidity value.

Objective:

To study the Temperature and Relative Humidity value of sensor data.

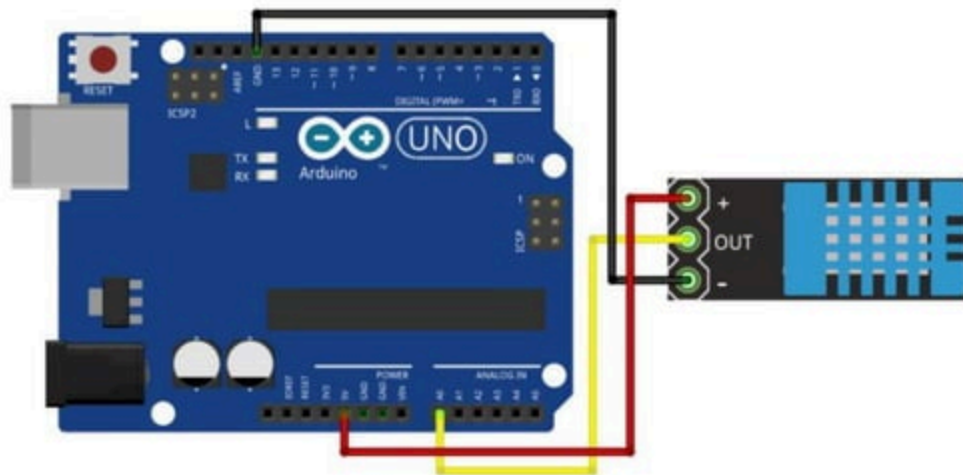
Components Required:

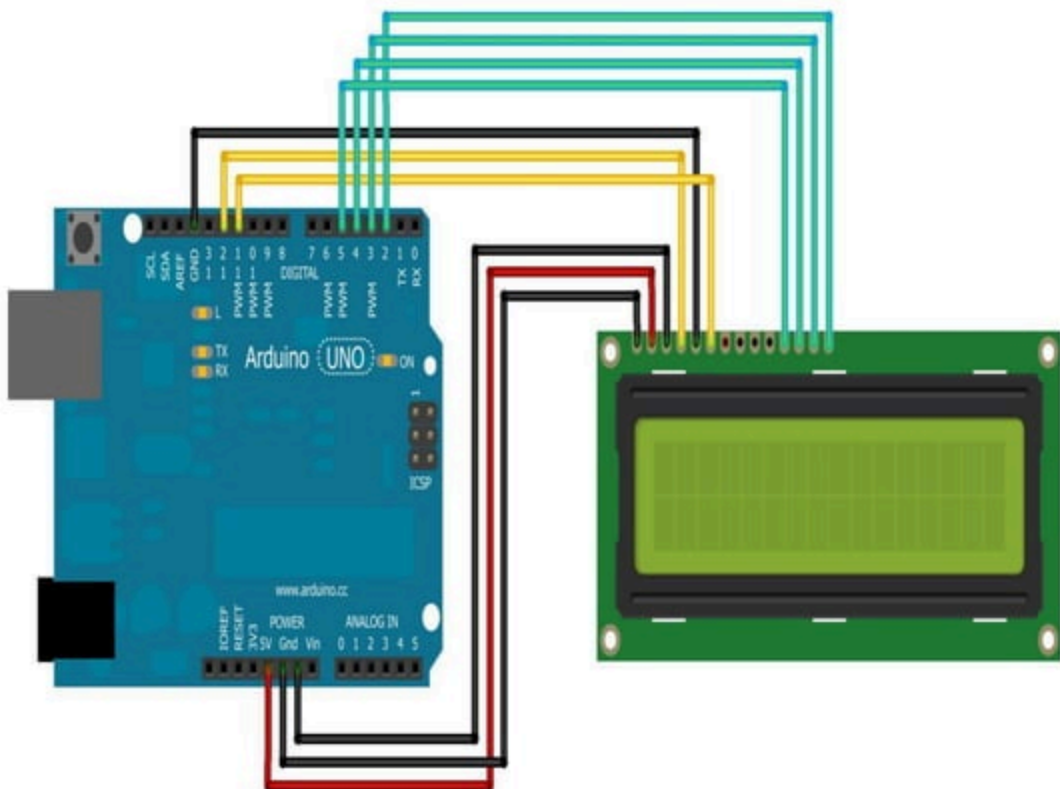
1. Arduino UNO.
2. DHT11 Temperature and Humidity sensor data .

Connections:

Arduino Pins	DHT11 Sensor
5V	+VE
A0	OUT
GND	-VE

CONNECTIONS





Procedure :

1. Set up pin connection DHT11 Humidity Temperature sensor and Arduino UNO as shown in figure.
2. Open the Arduino UNO IDE and create and save a sketch source code as follows.
3. Compile sketch.
4. Upload sketch on to Arduino UNO.

Source Code:

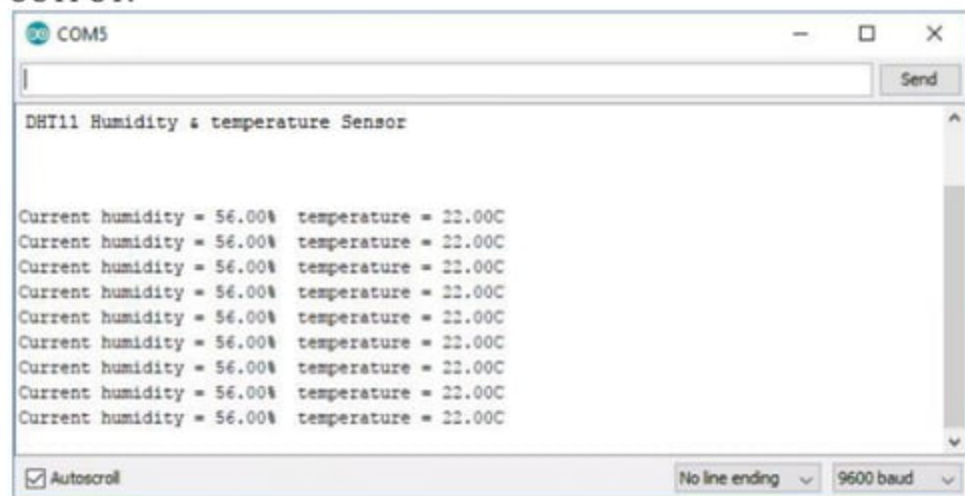
```

#include <dht.h>
#include <LiquidCrystal.h>
#define dht_apin A0 // Analog Pin sensor is connected to
char ch;
int Contrast=15;
// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
dht DHT;
void setup()
{
  Serial.begin(9600);
  delay(500); //Delay to let system boot
  Serial.println("DHT11 Humidity & temperature Sensor\n\n");
  pinMode(13,OUTPUT);
  analogWrite(6,Contrast);
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  delay(1000); //Wait before accessing Sensor
} //end "setup()"

void loop(){
  //Start of Program
  DHT.read11(dht_apin);
  //Print on Serial Monitor
  Serial.print("Current humidity = ");
  Serial.print(DHT.humidity);
  Serial.print("% ");
  Serial.print("temperature = ");
  Serial.print(DHT.temperature);
  Serial.println("C ");

  lcd.setCursor(0,0); //goto first column (column 0) and first line (Line 0)
  lcd.print("Temp C = "); //Print at cursor Location
  lcd.print(DHT.temperature);
  lcd.setCursor(0,1); //goto first column (column 0) and second line (line 1)
  lcd.print("Hum % = "); //Print at cursor location
  lcd.print(DHT.humidity);
  delay(5000); //Wait 5 seconds before accessing sensor again.
  //Fastest should be once every two seconds.
} // end loop()

```

OUTPUT:

The screenshot shows a serial monitor window titled 'COM5'. It displays the output of a DHT11 Humidity & temperature Sensor. The data is repeated ten times, showing a constant humidity of 56.00% and a constant temperature of 22.00C. The window includes a 'Send' button, an 'Autoscroll' checkbox (which is checked), and dropdown menus for 'No line ending' and '9600 baud'.

```
COM5  
DHT11 Humidity & temperature Sensor  
  
Current humidity = 56.00% temperature = 22.00C  
Current humidity = 56.00% temperature = 22.00C  
Current humidity = 56.00% temperature = 22.00C  
Current humidity = 56.00% temperature = 22.00C  
Current humidity = 56.00% temperature = 22.00C  
Current humidity = 56.00% temperature = 22.00C  
Current humidity = 56.00% temperature = 22.00C  
Current humidity = 56.00% temperature = 22.00C  
Current humidity = 56.00% temperature = 22.00C  
Current humidity = 56.00% temperature = 22.00C  
  
☒ Autoscroll No line ending 9600 baud
```

Applications :

A humidity sensor senses, measures and reports both moisture and air temperature. The ratio of moisture in the air to the highest amount of moisture at a particular air temperature is called relative humidity. Relative humidity becomes an important factor, when looking for comfort Automated humidity control systems use humidity sensors to test humidity levels and can control processes, such as chemical dryers, required to maintain a specific relative humidity in industries.

Conclusion:

Air temperature and humidity sensors provide the benefit of obtaining measurement data for two parameters humidity and temperature using a single sensor. Sensor data help us to study and measure the humidity and temperature.