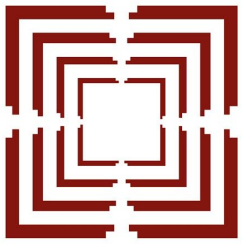


Ahmedabad University

IoT based GreenHouse Environment Control and Monitoring System

Report - 3

Under Guidance of,

Prof. Anurag Lakhlani

By,

Group - 13

Saloni Chudgar - 1641013

Abhi Patel - 1641021

Nildeep Jadav - 1641024

Manav Chotalia - 1641036

Motivation:

Most of the agricultural sector in the country is facing the low economical resources. Moreover, seasonal changes also affect the agricultural growth of the plants. A greenhouse is a modern off season, cultivating method that gives high yields at any season. So basically, it is a covered area where plants grow and cultivate. It is also known as land of controlled crops and plants.

Recently, the use of greenhouse has increased tremendously. A greenhouse is a structure that is built of walls and a transparent roof and is designed to maintain regulated climatic conditions. These structures are used for the cultivation of plants, fruits, and vegetables which require a particular level of sunlight, temperature, humidity and soil moisture. The temperature and humidity readings are critical to determine if heating/cooling curves in each greenhouse are optimal or not. However, measuring these parameters becomes a tedious job. With the use of IoT devices this processes can be automated or at least semi-automated with human monitoring.

Description:

Greenhouse effect is a natural phenomenon and beneficial to human beings. Numerous farmers fail to get good profits from the greenhouse crops for the reason that they can't manage two essential factors, which determines plant growth as well as productivity. Greenhouse temperature should not go below a certain degree, High humidity can result to crop transpiration, condensation of water vapour on various greenhouse surfaces, and water evaporation from the humid soil. To overcome such challenges, this greenhouse monitoring and control system comes to the rescue.

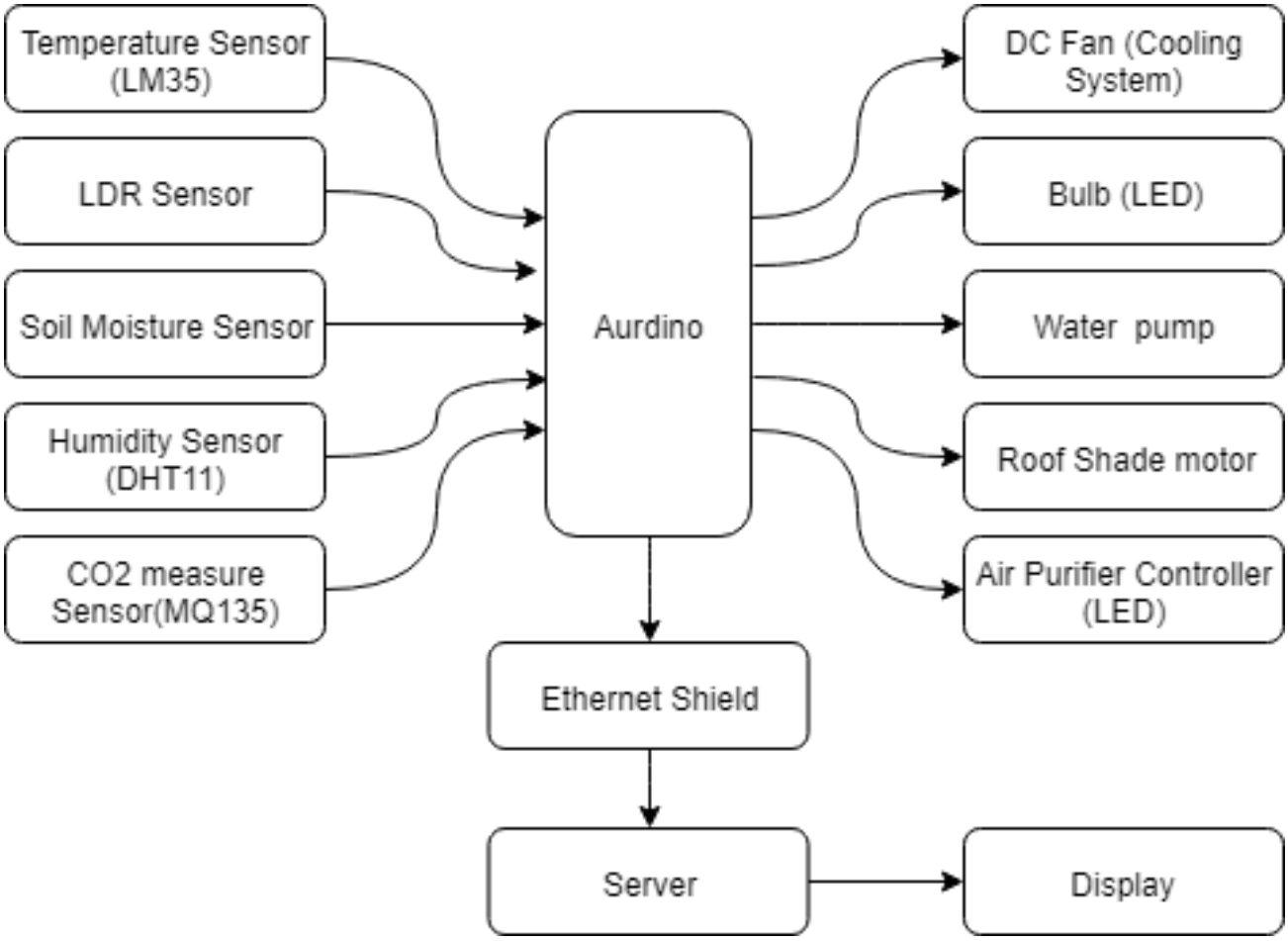
Devices like Moisture sensor, Temperature sensor, CO2 sensor, Light Sensor, Humidity Sensor can be attached to an Arduino Microcontroller. The devices needed to maintain the parameters are then turned on/off automatically by the Arduino based on the data obtained by the sensors. Thus, a conrolled environment is efficiently created with minimum effort.

Final Outcome:

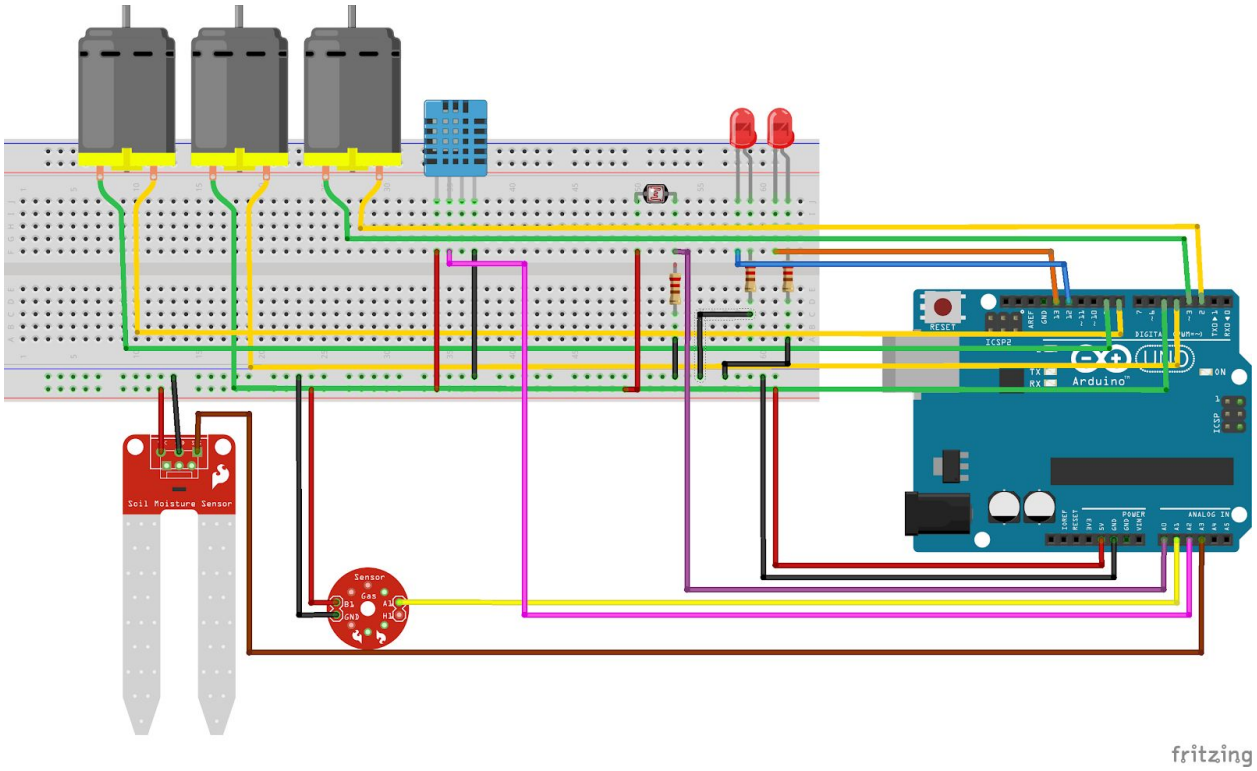
A controlled environment is autonomously created without human intervention and the data can be accessed on a remote server by anyone for monitoring purposes. Based on the necessary range of temperature, humidity, CO2 , moisture already entered, the arduino device will maintain the parameters by controlling the fan. Water Pump, dehumidifier, air-purifier or any other devices necessary. The live streaming of the data of the sensors will be sent to the user through internet and the user can monitor it on his/her own laptop/mobile device.

Also if there is a major change in the parameters or if anything abnormal occurs then one user with administrator control will be alerted and can change the output.

Block Diagram:



Circuit Diagram:

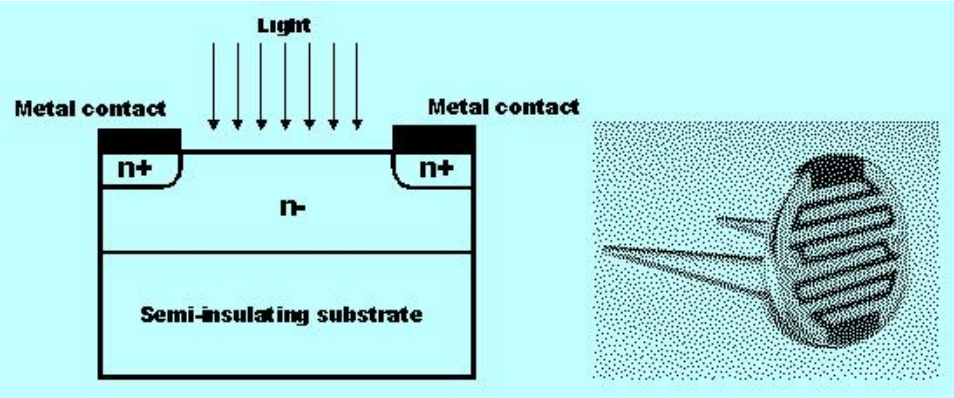


List of Components :

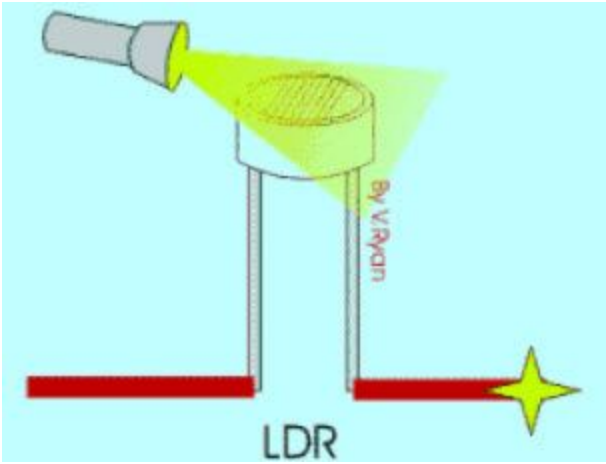
- ➔ Arduino Board
- ➔ LDR Sensor
- ➔ Humidity Sensor
- ➔ Air Quality Sensor
- ➔ Moisture Sensor
- ➔ Wi-Fi Module
- ➔ Temperature Sensor
- ➔ Fan
- ➔ LEDs
- ➔ Water Pump
- ➔ Breadboard
- ➔ Resistors
- ➔ Connecting Wires

LDR :

Operating principles



The controlling of lights and home appliances are generally operated and maintained manually on several occasions. But the process of appliances controlling may cause wastage of power due to the carelessness of human beings or unusual circumstances. To overcome this problem we can use the light-dependent



resistor circuit for controlling the loads based on the intensity of light. An LDR or a photoresistor is a device that is made up of high resistance semiconductor material.

Code

```
const int LDRPin = A0;  // Analog input pin that the LDR is attached to
const int LEDPin = 9; // Analog output pin that the LED is attached to

int LDRValue = 0;       // value read from the LDR
int LEDValue = 0;       // value output to the PWM (analog out)
```

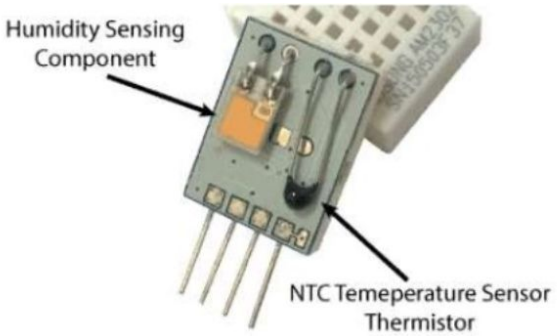
```
void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);
}

void loop() {
  // read the analog in value:
  LDRValue = analogRead(LDRPin);
  // map it to the range of the analog out:
  LEDValue = map(LDRValue, 0, 1023, 0, 255);
  // change the analog out value:
  analogWrite(LEDPin, LEDValue);

  // print the results to the serial monitor:
  Serial.print("sensor = ");
  Serial.print(LDRValue);
  Serial.print("\t output = ");
  Serial.println(LEDValue);

  // wait 2 milliseconds before the next loop
  // for the analog-to-digital converter to settle
  // after the last reading:
  delay(2);
}
```

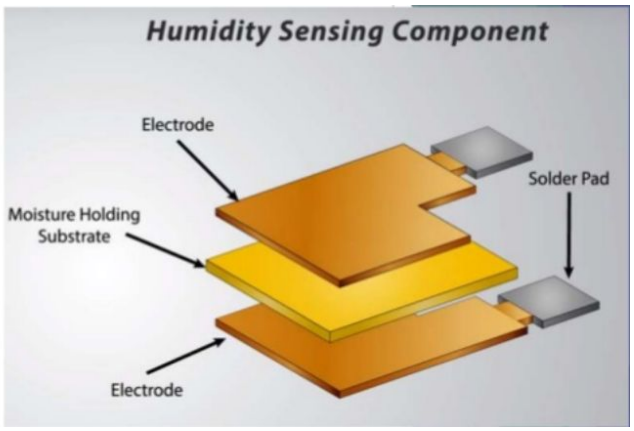
Moisture and Temperature Sensor (DHT11)

Operating Principle

DHT11 sensor consists of a capacitive humidity sensing element and a thermistor for sensing temperature. The humidity sensing capacitor has two electrodes with a moisture-holding substrate as a dielectric between them. Change in the capacitance value occurs with the change in humidity levels. The IC measure, process changed resistance values and change them into digital form.

For measuring temperature this sensor uses a Negative Temperature coefficient thermistor, which causes a decrease in its resistance value with an increase in temperature. To get larger resistance value even for the smallest change in temperature, this sensor is usually made up of semiconductor ceramics or polymers.





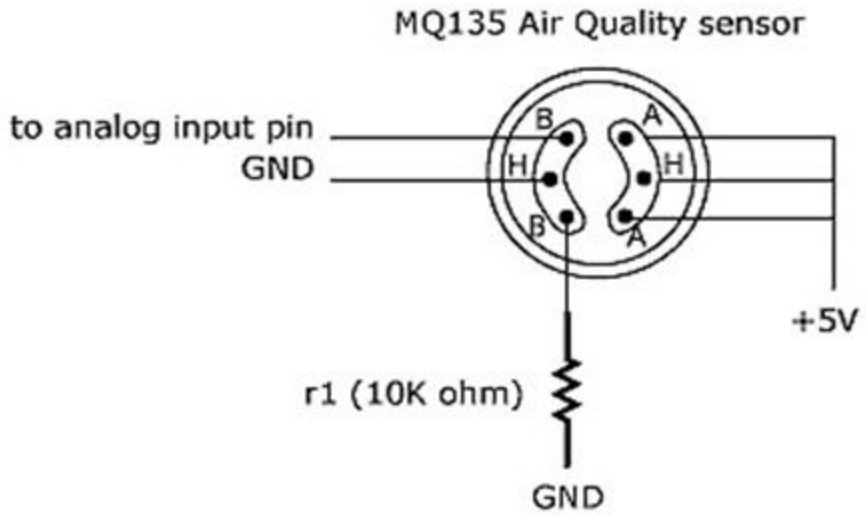
The temperature range of DHT11 is from 0 to 50 degree Celsius with a 2-degree accuracy. Humidity range of this sensor is from 20 to 80% with 5% accuracy. The sampling rate of this sensor is 1Hz .i.e. it gives one reading for every second. DHT11 is small in size with operating voltage from 3 to 5 volts. The maximum current used while measuring is 2.5mA.

Code

```
#include <dht11.h>
#define DHT11PIN A1

dht11 DHT11;
void setup()
{
  Serial.begin(9600);

}
void loop()
{
  Serial.println();
  int chk = DHT11.read(DHT11PIN);

  Serial.print("Humidity (%): ");
  Serial.println((float)DHT11.humidity, 2);

  Serial.print("Temperature (C): ");
  Serial.println((float)DHT11.temperature, 2);
  delay(1000);
}
```

Air Quality Sensor (MQ135)

Operating principle



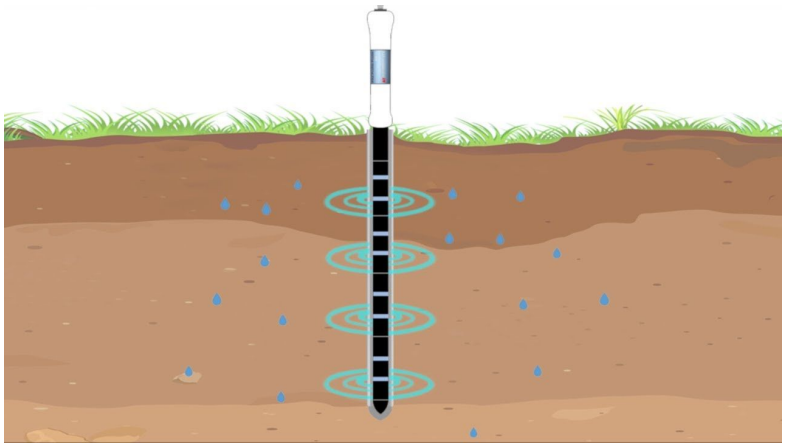
The air quality sensor is also a MQ-135 sensor for detecting venomous gases that are present in the air in homes and offices. The gas sensor layer of the sensor unit is made up of tin dioxide (SnO2); it has lower conductivity compare to clean hair and due to air pollution the conductivity is increases. The air quality sensor detects ammonia, nitrogen oxide, smoke, CO2 and other harmful gases. The air quality sensor has a small potentiometer that permits the adjustment of the load resistance of the sensor circuit.

Code

```
int AirQualitySensorValue;

void setup()
{
  Serial.begin(9600);                    // sets the serial port to 9600
}
  void loop(){AirQualitySensorValue = analogRead(A3);       // read analog input pin 0
  Serial.print("AirQua= ");
  Serial.print(AirQualitySensorValue, DEC);               // prints the value read
  Serial.println(" PPM");

  delay(100);                            // wait 100ms for next reading
}
```

Soil Moisture Sensor

Operating Principle



This sensor mainly utilizes capacitance to gauge the water content of the soil (dielectric permittivity). The working of this sensor can be done by inserting this sensor into the earth and the status of the water content in the soil can be reported in the form of a percent.

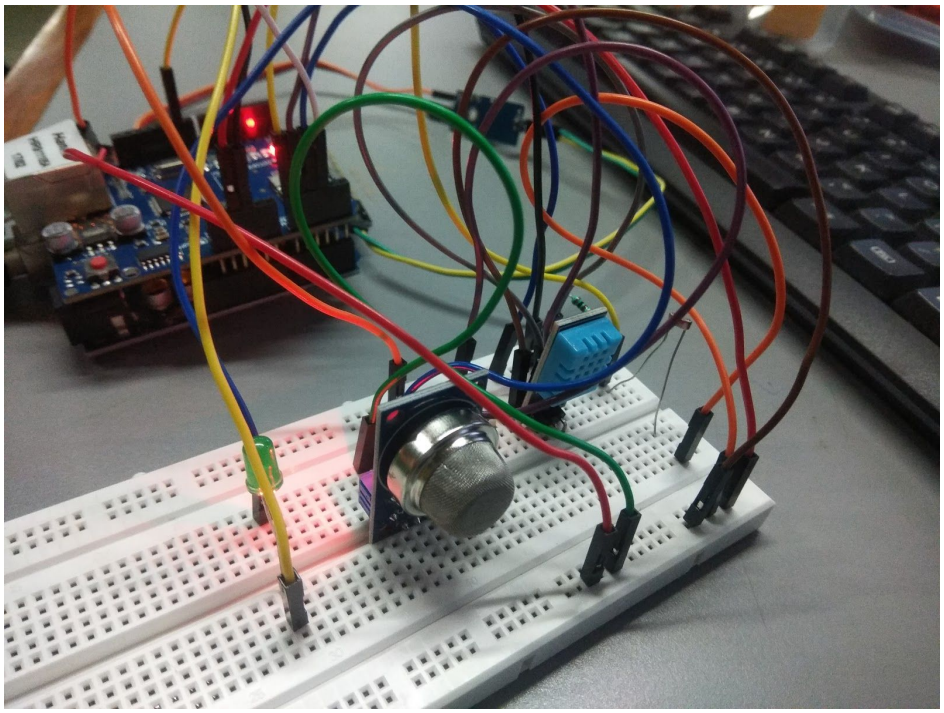
Code

```
const int sensor_pin = A3;  /* Soil moisture sensor O/P pin */

void setup() {
  Serial.begin(9600); /* Define baud rate for serial communication */
}
void loop() {
  float moisture_percentage;
  int sensor_analog;
  sensor_analog = analogRead(sensor_pin);
  moisture_percentage = ( 100 - ( (sensor_analog/1023.00) * 100 ) );
  Serial.print("Moisture Percentage = ");
  Serial.print(moisture_percentage);
  Serial.print("%\n\n");
  Serial.print(sensor_analog);
  Serial.print("\n\n");
  delay(1000);
}
```

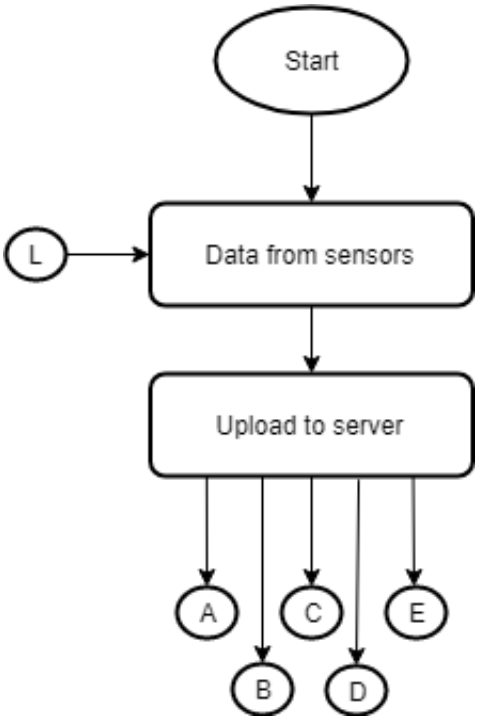
Transfer data to server code

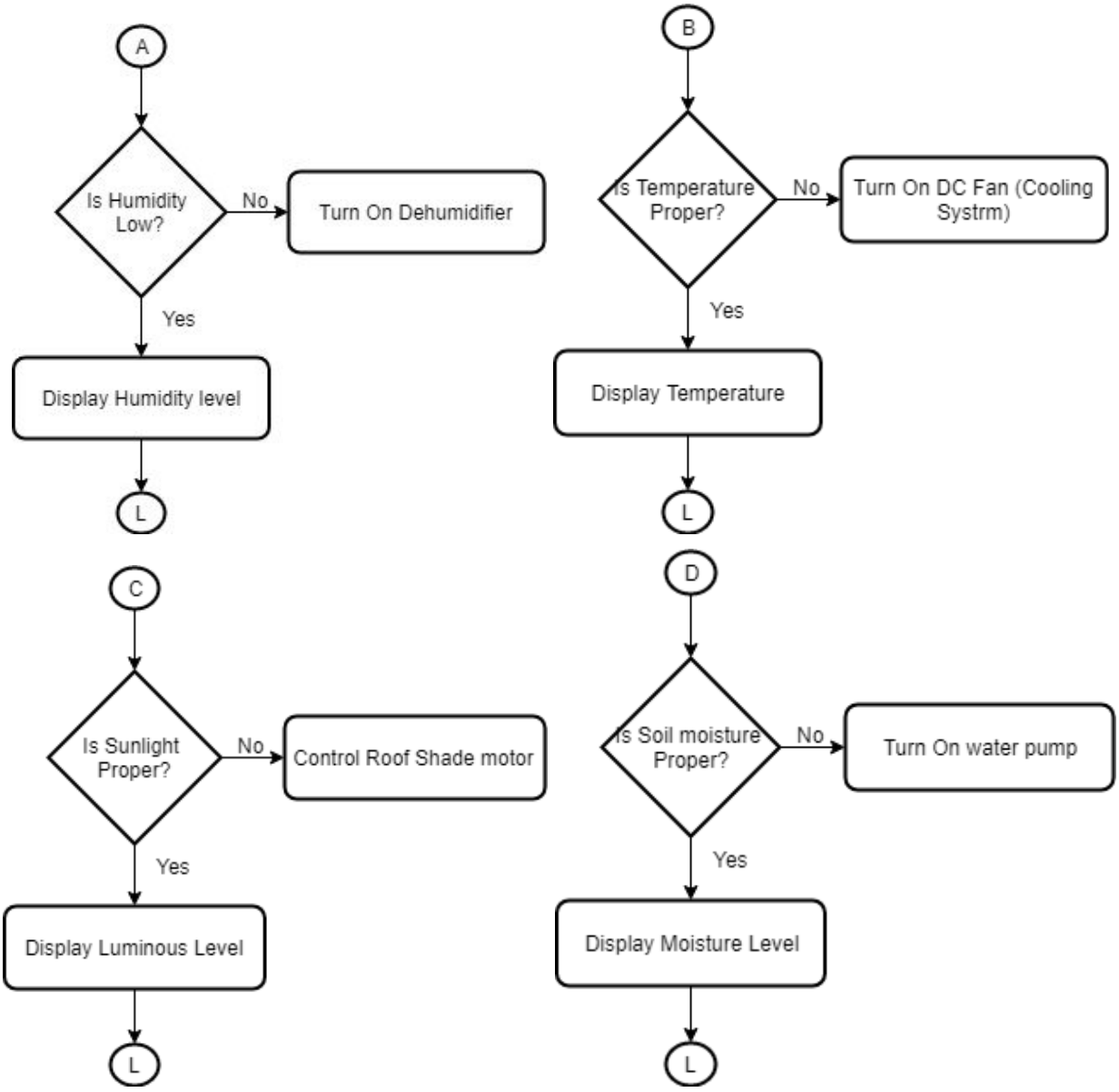
```
#include <SPI.h>
#include <Ethernet.h>


byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED
};
IPAddress ip(10,20,24,59);

// initialize the library instance:
EthernetClient client;

char server[] = "10.20.37.53";
const size_t maxSize = 512;
char response[512];
int pin = A0;
int sensorValue = 0;  // variable to store the value coming from the sensor
void setup() {
  // start serial port:
  Serial.begin(9600);

  Serial.println("--- Start ---");

  // give the ethernet module time to boot up:
  delay(1000);
  // start the Ethernet connection using a fixed IP address and DNS server:
  Ethernet.begin(mac, ip);
  // print the Ethernet board/shield's IP address:
  Serial.print("My IP address: ");
  Serial.println(Ethernet.localIP());

}

void loop() {
  //sensorValue = analogRead(pin);
  //Serial.println(sensorValue);
 httpRequest();
}
```

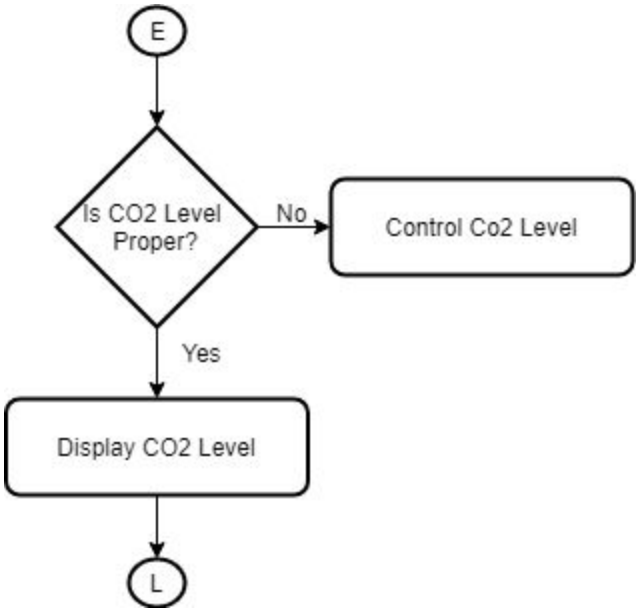
```
// this method makes a HTTP connection to the server:
void httpRequest() {
  sensorValue = analogRead(pin);

    client.stop();
  if (client.connect(server, 80)) {
    Serial.println("connected");
    String s ="GET /temp.php?request="+String(sensorValue);
    Serial.println(s);
    client.println(s);

    client.println("Host: 10.20.37.53");
    delay(1000);
    size_t len = client.readBytes(response, 512);
    response[len] = 0;
    client.println("Connection: close");
    client.println();
    Serial.println(response);
  }
  else
    Serial.println("connection failed");
}
```



Flow Chart:




A
Is Humidity Low?
No
Turn On Dehumidifier
Yes
Display Humidity level
L
B
Is Temperature Proper?
No
Turn On DC Fan (Cooling Systrm)
Yes
Display Temperature
L
C
Is Sunlight Proper?
No
Control Roof Shade motor
Yes
Display Luminous Level
L
D
Is Soil moisture Proper?
No
Turn On water pump
Yes
Display Moisture Level
L


E
Is CO2 Level Proper?
No
Control Co2 Level
Yes
Display CO2 Level
L

Final Code :

```
#include <SPI.h>
#include <Ethernet.h>
#include <dht11.h>


byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED
};
IPAddress ip(10,20,12,47);
char srvr1[] = "www.dweet.io";

// initialize the library instance:
EthernetClient client;

char server[] = "10.1.22.74";
const size_t maxSize = 512;
char response[512];
int LDRpin = A0;        // input pin A0 to LDR
#define DHT11PIN A1  // input pin A1 to DHT11
int SoilMoisturepin = A3;  // input pin A3 to Soil moisture

int LDRValue = 0;// variable to store the value coming from the LDR
int AirQualitySensorValue = 0;
int HumidValue = 0;
int TempValue = 0;
int moisture_percentage = 0;
int SoilMoistAnalog = 0;

dht11 DHT11;

String rcv="";
int Rcvflg = 5;

int TempRcv = 30;
int HumidRcv = 75;
int LDRRcv = 300;
int SoilRcv = 78;
```

```
int AqiRcv = 700;

void setup() {
  // start serial port:
  Serial.begin(9600);

  Serial.println("--- Start ---");

  // give the ethernet module time to boot up:
  delay(1000);
  // start the Ethernet connection using a fixed IP address and DNS server:
  //Ethernet.begin(mac, ip);
  Serial.println(Ethernet.begin(mac));
  // print the Ethernet board/shield's IP address:
  Serial.print("My IP address: ");
  Serial.println(Ethernet.localIP());

  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(13, OUTPUT);

}

void loop() {

  /* LDR */
  LDRValue = analogRead(LDRpin); // read analog input of LDR

  /* DHT11 */
  int chk = DHT11.read(DHT11PIN); // read analog input of temp and humidity
  HumidValue = (int)DHT11.humidity;

  TempValue = (int)DHT11.temperature;
```

```
/* MQ135 */
AirQualitySensorValue = analogRead(A2);       // read analog input pin A2

/* Soil Moisture Sensor */
SoilMoistAnalog = analogRead(SoilMoisturepin);   // read analog input of Soil moisture
moisture_percentage = ( 100 - ( (SoilMoistAnalog/1023.00) * 100 ) );

httpSendRequest();

if(Rcvflg == 5){
  httpReceiveRequest();
  Rcvflg = 0;
}
if(Rcvflg>9){
  Rcvflg = 0;
}
Rcvflg++;

if(TempValue > TempRcv||1){  // with temprature data cooling system(motor) will On/Off
  digitalWrite(2, HIGH);
  digitalWrite(3, LOW);
  Serial.println("Cooling System(motor) is ON");
  delay(2000);
}
else{
  digitalWrite(3, LOW);
  digitalWrite(2, LOW);
  Serial.println("Cooling System(motor) is OFF");
}

if(LDRValue > LDRRcv){  // With LDR Value Lighting System (LED) Will Operate
  digitalWrite(13, HIGH);
  Serial.println("Lighting System (LED) is ON");
  delay(2000);
}
else{
  digitalWrite(13, LOW);
  Serial.println("Lighting System (LED) is OFF");
}
```

```
  if(moisture_percentage < SoilRcv || 1){  // with Soil Moisture data Water pump(motor) will
On/Off
    digitalWrite(6, HIGH);
    digitalWrite(7, LOW);
    Serial.println("Water pump(motor) is ON");
    delay(2000);
  }
  else{
    digitalWrite(6, LOW);
    digitalWrite(7, LOW);
    Serial.println("Water pump(motor) is OFF");
  }

  if(HumidValue > HumidRcv || 1){  // with Humidity Sensor data Dehumidifier(motor) will
On/Off
    digitalWrite(8, HIGH);
    digitalWrite(9, LOW);
    Serial.println("Dehumidifier(motor) is ON");
    delay(2000);
  }
  else{
    digitalWrite(8, LOW);
    digitalWrite(9, LOW);
    Serial.println("Dehumidifier(motor) is OFF");
  }

  if(AirQualitySensorValue > AqiRcv){  // With MQ135 sensor Value Air purifier System (LED)
Will Operate
    digitalWrite(12, HIGH);
    Serial.println("Air purifier System (LED) is ON");
    delay(2000);
  }
  else{
    digitalWrite(12, LOW);
    Serial.println("Air purifier System (LED) is OFF");
  }
}
```

```
// this function Send data to HTTP connection to the server:
void httpSendRequest() {

  /* HTTP Request */
    client.stop();
  if (client.connect(server, 80)) {
    Serial.println("connected");
    String s ="GET
/send.php?temp="+String(TempValue)+"&ldr="+String(LDRValue)+"&humid="+

String(HumidValue)+"&soil="+String(moisture_percentage)+"&aqi="+String(AirQualitySensor
Value);
    Serial.println(s);
    client.println(s);

    client.println("Host: 10.1.22.74");
    delay(1000);
    size_t len = client.readBytes(response, 512);
    response[len] = 0;
    client.println("Connection: close");
    client.println();
    Serial.println(response);
  }
  else
    Serial.println("connection failed");
/* HTTP Request 2nd */
    client.stop();
  if (client.connect(srvr1, 80)) {
    Serial.println("connected to server 2");
    String s ="POST
/dweet/for/arduinotest?temp="+String((int)DHT11.temperature)+"&ldr="+String(LDRValue)+"
&humid="+

String((int)DHT11.humidity)+"&soil="+String(moisture_percentage)+"&aqi="+String(AirQuali
tySensorValue);
    Serial.println(s);
    client.println(s);

    client.println("Host: www.dweet.io");
```

```
   delay(2000);
   size_t len = client.readBytes(response, 512);
   response[len] = 0;
   client.println("Connection: close");
   client.println();
   Serial.println(response);
  }
  else
   Serial.println("connection failed");

}

// this function Receive data to HTTP connection from the server:
void httpReceiveRequest()
{
  if (client.connect(server, 80))
  {
   Serial.println("Connection established 1");
   client.println("GET /receive.php");

   client.println("Host: 10.1.22.74");
   delay(1000);
   unsigned long timeout = millis();
   while (client.available() == 0)
   {
    if (millis() - timeout > 2500) //If nothing is available on server for 25 seconds, close the
connection.
     {
      return;
     }
   }
   while(client.available())
   {
    String line = client.readStringUntil('\r'); //Read the server response line by line..
    rcv+=line; //And store it in rcv.
   }
   client.println("\r\nConnection: close\r\n\r\n"); //GET request for server response.
   client.stop(); // Close the connection.
  }
```

```
  else
  {
   Serial.println("Connection failed 1");
  }

  String s2=rcv.substring((rcv.indexOf('[')),rcv.indexOf(']')); // Extract the line returned by JSON object.

  int l=s2.length();
  String s="";
  String s1="";
  int m,i=0;
  while(i<l)
  {
   m=s2.indexOf("",i+1);
   if(m!=-1)
   {
    for(int j=m+1;j<l;j++)
    {
     if(s2.charAt(j)!="")
     {
      s+=s2.charAt(j);
     }
     else
     {
      i=j;
      break;
     }
    }
    if(s=="temp" || s=="humid" || s=="soil" || s=="aqi" || s=="ldr") {
     s1 = s;
     s = "";
     continue;
    }
    if(s1=="temp") {
     Serial.print("temp : " + s + " ");
     TempRcv = s.toInt();
     Serial.println(TempRcv);
    }
```

```
    else if(s1=="humid") {
      Serial.print("humid : " + s + " ");
      HumidRcv = s.toInt();
      Serial.println(HumidRcv);
    }
    else if(s1=="soil") {
      Serial.print("soil : " + s + " ");
      SoilRcv = s.toInt();
      Serial.println(SoilRcv);
    }
    else if(s1=="aqi") {
      Serial.print("aqi : " + s + " ");
      AqiRcv = s.toInt();
      Serial.println(AqiRcv);
    }
    else if(s1=="ldr") {
      Serial.print("ldr : " + s + " ");
      LDRRcv = s.toInt();
      Serial.println(LDRRcv);
    }
    s="";
   }
   else {
    break;
   }
  }
}
```

Timeline:

Tasks	19/8 - 25/8	26/8 - 1/9	2/9 - 9/9	10/9 - 14/9
Gather Components	Done			
Build circuit	Done			
Test individual Components		Done (except water pump)		
Connect via internet (Code for remote server)		Done and tested with temperature sensor	Done and tested with all the components	
Combine all the components			Done	
Testing and Debugging				Done
Project Report				Done

References:

Dan, L. I. U., et al. "Intelligent agriculture greenhouse environment monitoring system based on IOT technology." *2015 International Conference on Intelligent Transportation, Big Data and Smart City*. IEEE, 2015.

Pallavi, S., Jayashree D. Mallapur, and Kirankumar Y. Bendigeri. "Remote sensing and controlling of greenhouse agriculture parameters based on IoT." *2017 International Conference on Big Data, IoT and Data Science (BID)*. IEEE, 2017.

Li, Li, et al. "Application of IoT technology in greenhouse management." *2012 Dallas, Texas, July 29-August 1, 2012*. American Society of Agricultural and Biological Engineers, 2012

Appendix for DataSheet:

TECHNICAL DATA MQ-135 GAS SENSOR

FEATURES

Wide detecting scope Fast response and High sensitivity
Stable and long life Simple drive circuit

APPLICATION

They are used in air quality control equipments for buildings/offices, are suitable for detecting of NH3,NOx, alcohol, Benzene, smoke,CO_2 ,etc.

SPECIFICATIONS

A. Standard work condition

Symbol	Parameter name	Technical condition	Remarks
Vc	Circuit voltage	5V±0.1	AC OR DC
V_H	Heating voltage	5V±0.1	ACOR DC
R_L	Load resistance	can adjust	
R_H	Heater resistance	33Ω±5%	Room Tem
P_H	Heating consumption	less than 800mw	

B. Environment condition

Symbol	Parameter name	Technical condition	Remarks
Tao	Using Tem	-10 -45	
Tas	Storage Tem	-20 -70	
R_H	Related humidity	less than 95%Rh	
O_2	Oxygen concentration	21%(standard condition)Oxygen concentration can affect sensitivity	minimum value is over 2%

C. Sensitivity characteristic

Symbol	Parameter name	Technical parameter	Ramark 2
Rs	Sensing Resistance	30KΩ-200KΩ (100ppm NH_3)	Detecting concentration scope 10ppm-300ppm NH_3 10ppm-1000ppm Benzene 10ppm-300ppm Alcohol
α (200/50) NH_3	Concentration Slope rate	≤0.65	
Standard Detecting Condition	Temp: 20 ±2 Vc:5V±0.1 Humidity: 65%±5% Vh: 5V±0.1		
Preheat time	Over 24 hour		

D. Structure and configuration, basic measuring circuit

	Parts	Materials
1	Gas sensing layer	SnO_2
2	Electrode	Au
3	Electrode line	Pt
4	Heater coil	Ni-Cr alloy
5	Tubular ceramic	Al_2O_3
6	Anti-explosion network	Stainless steel gauze (SUS316 100-mesh)
7	Clamp ring	Copper plating Ni
8	Resin base	Bakelite
9	Tube Pin	Copper plating Ni

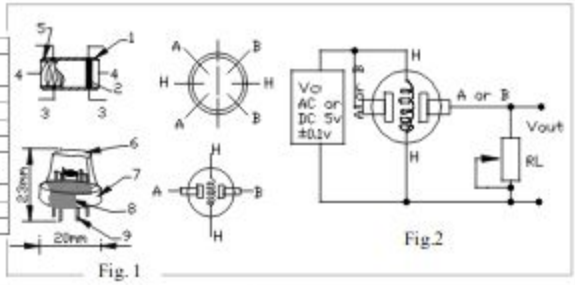
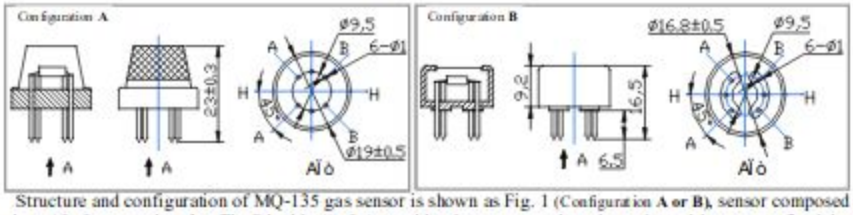


Fig.1 Fig.2



Structure and configuration of MQ-135 gas sensor is shown as Fig. 1 (Configuration **A or B**), sensor composed by micro AL_2O_3 ceramic tube, Tin Dioxide (SnO_2) sensitive layer, measuring electrode and heater are fixed into a crust made by plastic and stainless steel net. The heater provides necessary work conditions for work of sensitive

MQ135 Semiconductor Sensor for Air Quality Control

Sensitive material of MQ135 gas sensor is SnO_2, which with lower conductivity in clean air. When the target combustible gas exist, The sensor's conductivity is more higher along with the gas concentration rising. Please use simple electrocircuit, Convert change of conductivity to correspond output signal of gas concentration.

MQ135 gas sensor has high sensitity to Ammonia, Sulfide and Benze steam, also sensitive to smoke and other harmful gases. It is with low cost and suitable for different application.

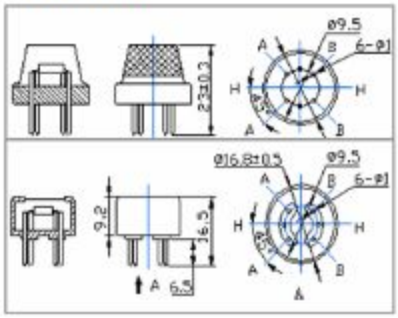
Character

* Good sensitivity to Harmful gases in wide range
* High sensitivity to Ammonia, Sulfide and Benze
* Long life and low cost
* Simple drive circuit

Application

* Domestic air pollution detector
* Industrial air pollution detector
* Portable air pollution detector

Configuration

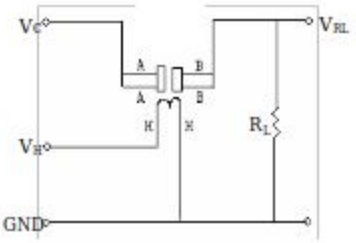


Technical Data

Model No.			MQ135
Sensor Type			Semiconductor
Standard Encapsulation			Bakelite (Black Bakelite)
Detection Gas			Ammonia, Sulfide, Benze steam
Concentration			10-10000ppm (Ammonia, Benze, Hydrogen)
Circuit	Loop Voltage	V_C	≤24V DC
	Heater Voltage	V_H	5.0V±0.2V AC or DC
	Load Resistance	R_L	Adjustable
Character	Heater Resistance	R_H	31Ω±3Ω (Room Tem.)
	Heater consumption	P_H	≤900mW
	Sensing Resistance	R_s	2KΩ-20KΩ(in 100ppm NH_3)
	Sensitivity	S	Rs(in air)/Rs(100ppm NH_3)≥5
	Slope	α	≤0.6 (R_{300ppm}/R_{100ppm} NH_3)
Condition	Tem. Humidity		20℃±2℃; 65%±5%RH
	Standard test circuit		Vc:5.0V±0.1V; V_H: 5.0V±0.1V
	Preheat time		Over 48 hours

$Ps=Vc^2 \times Rs/(Rs+RL)^2$

Basic test loop

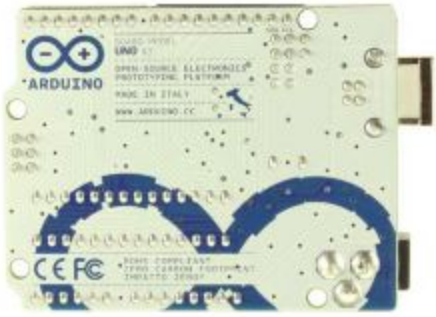


The above is basic test circuit of the sensor. The sensor need to be put 2 voltage, heater voltage(VH) and test voltage(VC). VH used to supply certified working temperature to the sensor, while VC used to detect voltage (VRL) on load resistance (RL) whom is in series with sensor. The sensor has light polarity, Vc need DC power. VC and VH could use same power circuit with precondition to assure performance of sensor. In order to make the sensor with better performance, suitable RL value is needed:

Power of Sensitivity body(Ps):

Arduino Uno

Arduino Uno R3 Front



Arduino Uno R3 Back

Arduino Uno R2 Front

Arduino Uno SMD

Arduino Uno Front

Arduino Uno Back

Overview

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.
The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.
Revision 2 of the Uno board has a resistor pulling the 8U2 HWB line to ground, making it easier to put into DFU mode.
Revision 3 of the board has the following new features:

- 1.0 pinout: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible both with the board that use the AVR, which operate with 5V and with the Arduino Due that operate with 3.3V. The second one is a not connected pin, that is reserved for future purposes.
- Stronger RESET circuit.
- Atmega 16U2 replace the 8U2.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V

SUNROM TECHNOLOGIES
Your Source for Embedded Systems

Email: info@sunrom.com or sunrom@gmail.com
Visit us at http://www.sunrom.com

Document: Datasheet Date: 28-Jul-08 Model #: 3190 Product's Page: www.sunrom.com/p-510.html

Light Dependent Resistor - LDR

Two cadmium sulphide(cds) photoconductive cells with spectral responses similar to that of the human eye. The cell resistance falls with increasing light intensity. Applications include smoke detection, automatic lighting control, batch counting and burglar alarm systems.

Applications

Photoconductive cells are used in many different types of circuits and applications.

Analog Applications

- Camera Exposure Control
- Auto Slide Focus - dual cell
- Photocopy Machines - density of toner
- Colorimetric Test Equipment
- Densitometer
- Electronic Scales - dual cell
- Automatic Gain Control – modulated light source
- Automated Rear View Mirror

Digital Applications

- Automatic Headlight Dimmer
- Night Light Control
- Oil Burner Flame Out
- Street Light Control
- Absence / Presence (beam breaker)
- Position Sensor

Electrical Characteristics

Parameter	Conditions	Min	Typ	Max	Unit
Cell resistance	1000 LUX 10 LUX	- -	400 9	- -	Ohm K Ohm
Dark Resistance	-	-	1	-	M Ohm
Dark Capacitance	-	-	3.5	-	pF
Rise Time	1000 LUX 10 LUX	- -	2.8 18	- -	ms ms
Fall Time	1000 LUX 10 LUX	- -	48 120	- -	ms ms
Voltage AC/DC Peak		-	-	320	V max
Current		-	-	75	mA max
Power Dissipation				100	mW max
Operating Temperature		-60	-	+75	Deg. C

Soil Moisture Sensor

This sensor can be used to test the moisture of soil, when the soil is having water shortage, the module output is at high level, else the output is at low level. By using this sensor one can automatically water the flower plant, or any other plants requiring automatic watering technique. Module triple output mode, digital output is simple, analog output more accurate, serial output with exact readings.

Features

- Sensitivity adjustable.
- Has fixed bolt hole, convenient installation.
- Threshold level can be configured.
- Module triple output mode, digital output is simple, analog output more accurate, serial output with exact readings.

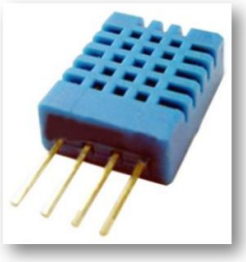
Applications

- Agriculture
- Landscape irrigation

Specifications

Parameter	Value
Operating Voltage	+5v dc regulated
Soil moisture	Digital value is indicated by out pin

Humidity and Temperature Sensor



Each DHT11 element is strictly calibrated in the laboratory that is extremely accurate on humidity calibration. The calibration coefficients are stored as programmes in the OTP memory, which are used by the sensor's internal signal detecting process. The single-wire serial interface makes system integration quick and easy. Its small size, low power consumption and up-to-20 meter signal transmission making it the best choice for various applications, including those most demanding ones. The component is 4-pin single row pin package. It is convenient to connect and special packages can be provided according to users' request.

2. Technical Specifications:

Overview:

Item	Measurement Range	Humidity Accuracy	Temperature Accuracy	Resolution	Package
DHT11	20-90%RH 0-50 ℃	±5%RH	±2℃	1	4 Pin Single Row

Detailed Specifications:

Parameters	Conditions	Minimum	Typical	Maximum
Humidity				
Resolution		1%RH	1%RH	1%RH
			8 Bit	
Repeatability			±1%RH	
Accuracy	25℃		±4%RH	
	0-50℃			±5%RH
Interchangeability	Fully Interchangeable			
Measurement Range	0℃	30%RH		90%RH
	25℃	20%RH		90%RH
	50℃	20%RH		80%RH
Response Time (Seconds)	1/e(63%)25℃, 1m/s Air	6 S	10 S	15 S
Hysteresis			±1%RH	
Long-Term Stability	Typical		±1%RH/year	
Temperature				
Resolution		1℃	1℃	1℃
		8 Bit	8 Bit	8 Bit
Repeatability			±1℃	
Accuracy		±1℃		±2℃
Measurement Range		0℃		50℃
Response Time (Seconds)	1/e(63%)	6 S		30 S