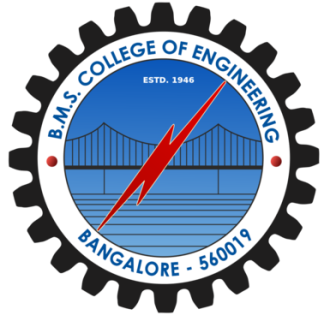
**BMS COLLEGE OF ENGINEERING**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

Academic Semester Aug-Dec 2016

****

**LAB MANUAL**

**ON**

**INTERNET OF THINGS**

15CS5DCIOT

Faculty In-Charge

| Panimozhi K  4th Sem B, Section | Dr. Kayarvizhy N  4th Sem C, Section | Antara De  4th Sem A, Section |
| --- | --- | --- |

**Manual Content**

| **Sl.No** | **Content** | **Pg. No** |
| --- | --- | --- |
|  | **IoT Overview** |  |
|  | **Introduction to Arduino** |  |
|  | **Arduino Development Board** |  |
|  | **General Purpose Input / Output Interfacing**  LED Blinking |  |
|  | **Experiment List** |  |
|  | 1. LED Fading |  |
|  | 1. Traffic Controller |  |
|  | 1. LDR |  |
|  | 1. PIR |  |
|  | 1. Home Automation |  |
|  | 1. IR remote controlled LED |  |
|  | 1. Fire Alert |  |
|  | 1. Automatic irrigation controller simulation |  |
|  | 1. Reverse parking sensor (Using LCD Display) |  |
|  | 1. Color recognition |  |
|  | 1. RFID reader and RFID tag count //Debugging |  |
|  | 1. Arduino with Vibration sensor |  |
|  | 1. Enthernet / Wifi |  |
|  |  |  |

**1. Internet of Things**

The Internet of Things (IoT) describes the phenomenon of everyday devices connecting to the Internet through tiny embedded sensors and computing power

We’re entering a new era of computing technology that many are calling the Internet of Things (IoT). Machine to machine, machine to infrastructure, machine to environment, the Internet of Everything, the Internet of Intelligent Things, intelligent systems—call it what you want, but it’s happening, and its potential is huge.  We see the IoT as billions of smart, connected “things” that will encompass every aspect of our lives, and its foundation is the intelligence that embedded processing provides. The IoT is comprised of smart machines interacting and communicating with other machines, objects, environments and infrastructures.

**Applications of IoT**

* Machine-to-machine communication
* Machine-to-infrastructure communication
* Telehealth: remote or real-time pervasive monitoring of patients, diagnosis and drug delivery
* Continuous monitoring of, and firmware upgrades for, vehicles
* Asset tracking of goods on the move
* Automatic traffic management
* Remote security and control
* Environmental monitoring and control
* Home and industrial building automation
* “Smart” applications, including cities, water, agriculture, buildings, grid, meters, broadband, cars, appliances, tags, animal farming and the environment, to name a few.

**2. Introduction to Arduino**

The Arduino board is a small microcontroller board, which is a small circuit (the board) that contains a whole computer on a small chip (the microcontroller). Arduino is composed of two major parts: the Arduino board, which is the piece of hardware you work on when you build your objects; and the Arduino IDE, the piece of software you run on your computer. You use the IDE to create a sketch (a little computer program) that you upload to the Arduino board. The sketch tells the board what to do.

The Arduino software is easy-to-use for beginners, yet flexible enough for advanced users. It runs on Mac, Windows, and Linux. Teachers and students use it to build low cost scientific instruments, to prove chemistry and physics principles, or to get started with programming and robotics. It has the specific advantages such as

• **Inexpensive** - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50

• **Cross-platform** - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.

• **Simple, clear programming environment** - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.

• **Open source and extensible software** - The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.

• **Open source and extensible hardware** - The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

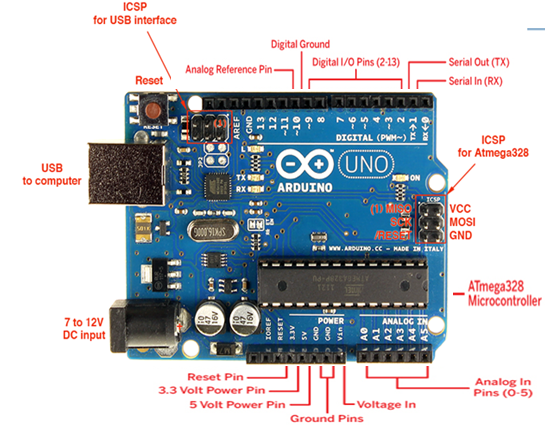
**Installing Arduino on Your Computer**

To program the Arduino board, you must first download the development environment (the IDE) from here: [www.arduino.cc/en/Main/Software](http://www.arduino.cc/en/Main/Software). Choose the right version for your operating system. Download the file and double-click on it to open it it; on Windows or Linux, this creates a folder named arduino-[version], such as arduino-1.0. Drag the folder to wherever you want it: your desktop, your Program Files folder (on Windows), etc. On the Mac, double-clicking it will open a disk image with an Arduino application (drag it to your Applications folder). Now whenever you want to run the Arduino IDE, you’ll open up the arduino (Windows and Linux) or Applications folder (Mac), and double-click the Arduino icon.

**Difference between Microprocessor and Microcontroller**

| **Microprocessor** | **Micro Controller** |
| --- | --- |
| 1G mobile phone | 1G mobile phone |
| Microprocessor is heart of Computer system. | Micro Controller is a heart of embedded system. |
| It is just a processor. Memory and I/O components have to be connected externally | Micro controller has external processor along with internal memory and i/O components |
| Since memory and I/O has to be connected externally, the circuit becomes large. | Since memory and I/O are present internally, the circuit is small. |
| Cannot be used in compact systems and hence inefficient | Can be used in compact systems and hence it is an efficient technique |
| Cost of the entire system increases | Cost of the entire system is low |
| Due to external components, the entire power consumption is high. Hence it is not suitable to used with devices running on stored power like batteries. | Since external components are low, total power consumption is less and can be used with devices running on stored power like batteries. |
| Most of the microprocessors do not have power saving features. | Most of the micro controllers have power saving modes like idle mode and power saving mode. This helps to reduce power consumption even further. |
| Since memory and I/O components are all external, each instruction will need external operation, hence it is relatively slower. | Since components are internal, most of the operations are internal instruction, hence speed is fast. |
| Microprocessor have less number of registers, hence more operations are memory based. | Micro controller have more number of registers, hence the programs are easier to write. |
| Microprocessors are based on von Neumann model/architecture where program and data are stored in same memory module | Micro controllers are based on Harvard architecture where program memory and Data memory are separate |
| Mainly used in personal computers | Used mainly in washing machine, MP3 players |

## 3. The Arduino Development board



### Digital Pins

The digital pins on an Arduino board can be used for general purpose input and output via the [pinMode()](https://www.arduino.cc/en/Reference/PinMode), [digitalRead()](https://www.arduino.cc/en/Reference/DigitalRead), and [digitalWrite()](https://www.arduino.cc/en/Reference/DigitalWrite) commands. Each pin has an internal pull-up resistor which can be turned on and off using digitalWrite() (w/ a value of HIGH or LOW, respectively) when the pin is configured as an input. The maximum current per pin is 40 mA.

Some other specific functions of pins are listed below:

* **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data
* **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
* **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the [analogWrite()](https://www.arduino.cc/en/Reference/AnalogWrite) function. On boards with an ATmega8, PWM output is available only on pins 9, 10, and 11.
* **BT Reset: 7.** (Arduino BT-only) Connected to the reset line of the bluetooth module.
* **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.
* **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

### Analog Pins

In addition to the specific functions listed below, the analog input pins support 10-bit analog-to-digital conversion (ADC) using the [analogRead()](https://www.arduino.cc/en/Reference/AnalogRead) function. Most of the analog inputs can also be used as digital pins: analog input 0 as digital pin 14 through analog input 5 as digital pin 19. Analog inputs 6 and 7 (present on the Mini and BT) cannot be used as digital pins.

### Power Pins

* **VIN**: The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
* **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
* **3V3.**  A 3.3 volt supply generated by the on-board FTDI chip.
* **GND.** Ground pins.

### Other Pins

* **AREF.** Reference voltage for the analog inputs. Used with [analogReference](https://www.arduino.cc/en/Reference/AnalogReference)().
* **Reset.**  Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

**4. General Purpose Input / Output Interfacing**

**LED Blinking**

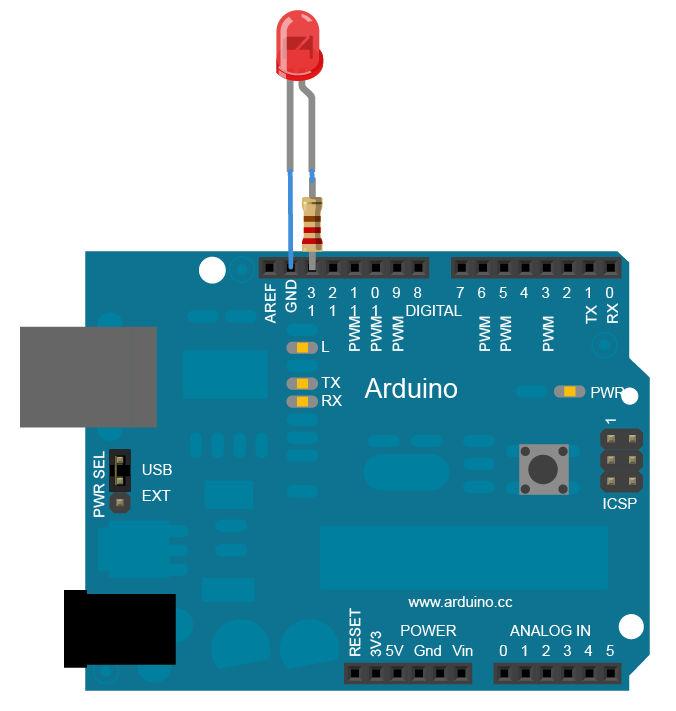
**Aim**

Turns on an LED on for one second, then off for one second, repeatedly

**Hardware Required**

* Arduino Board
* LEDs

**Circuit Diagram**



**Code**

// Pin 13 has an LED connected on most Arduino boards

int led = 13;  
  
void **setup**()  // the setup routine runs once when you press reset

{                  
// initialize the digital pin as an output.  
  pinMode(led, OUTPUT);       
}  
  
void **loop**() { // the loop routine runs over and over again forever  
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)  
  delay(1000);               // wait for a second  
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW

  delay(1000);               // wait for a second  
}

**5. Experiment list**

**1. Traffic Controller**

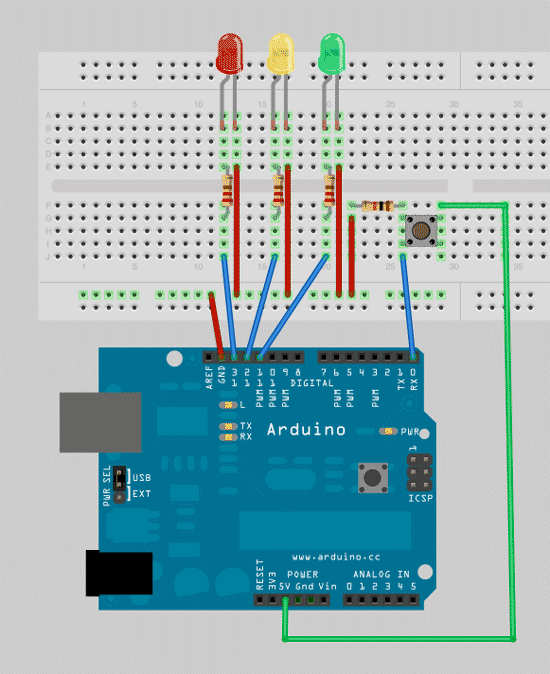
**Aim**

Traffic Signal Simulation

**Hardware Required**

* Arduino Board
* LEDs
* Resistors – 10 K
* Bread board

**Circuit Diagram**



**Code**

**// defining variables so that we can address the lights by name rather than a number**

int red = 13;

*int yellow = 12;*

*int green = 11;*

**//** **add the setup function, where’ll we define the red, yellow and green LEDs to be output mode**

void setup()

{

pinMode(red,OUTPUT);

pinMode(yellow,OUTPUT);

pinMode(green,OUTPUT);

}

void loop()

{

changeLights();

delay(15000);

}

void changeLights(){

// green off, yellow for 3 seconds

digitalWrite(green,HIGH);

digitalWrite(yellow,LOW);

delay(3000);

// turn off yellow, then turn red on for 5 seconds

digitalWrite(yellow,LOW);

digitalWrite(red,HIGH);

delay(5000);

// red and yellow on for 2 seconds (red is already on though)

digitalWrite(yellow,HIGH);

delay(2000);

// turn off red and yellow, then turn on green

digitalWrite(yellow,LOW);

digitalWrite(red,LOW);

digitalWrite(green,HIGH);

}

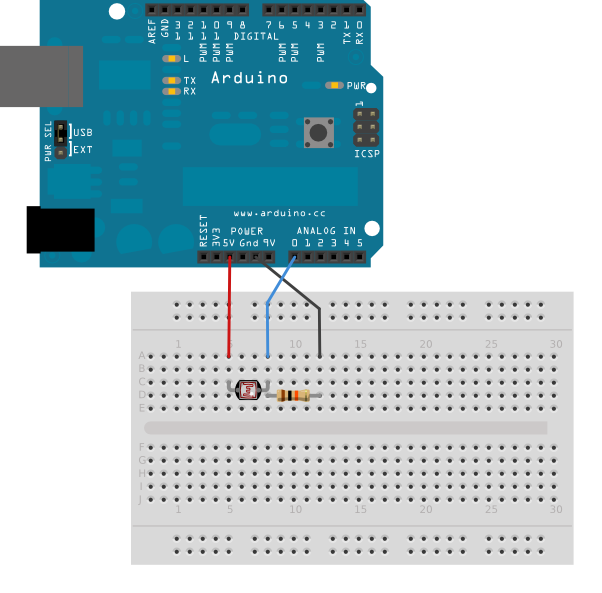
# 2. Nightlight Simulation

**Aim**

Simulating a night light using LDR and PIR

**Hardware Required**

* 1 LED
* 1 LDR
* 110K register



**Connection:**

1. Attach one leg of LDR to 5V and another leg to Arduino Analog pin A0
2. Attach one leg of 110K register with that leg of LDR connected to A0
3. Attach another leg of register to the ground
4. Connect the positive leg of LED to pin 11 and negative to GND

**Code:**

int LDR = 0; //analog pin to which LDR is connected, here we set it to 0 so it means A0

int LDRValue = 0; //that’s a variable to store LDR values

int light\_sensitivity = 500; //This is the approx value of light surrounding your LDR

void setup()

{

Serial.begin(9600); //start the serial monitor with 9600 buad

pinMode(11, OUTPUT); //attach positive leg of LED to pin 11

}

void loop()

{

LDRValue = analogRead(LDR); //reads the ldr’s value through LDR

Serial.println(LDRValue); //prints the LDR values to serial monitor

delay(50); //This is the speed by which LDR sends value to arduino

if (LDRValue < light\_sensitivity)

{

digitalWrite(11, HIGH);

}

else

{

digitalWrite(11, LOW);

}

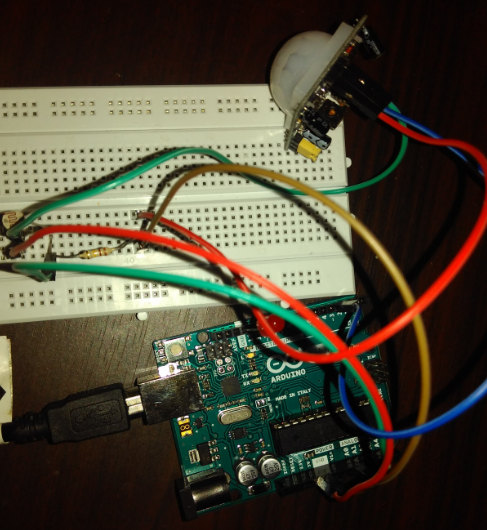
delay(1000);

}

**Observation:**

While lights are switched off in the room, LED should switch ON, when lights are switched on in the room, LED should switch off immediately.

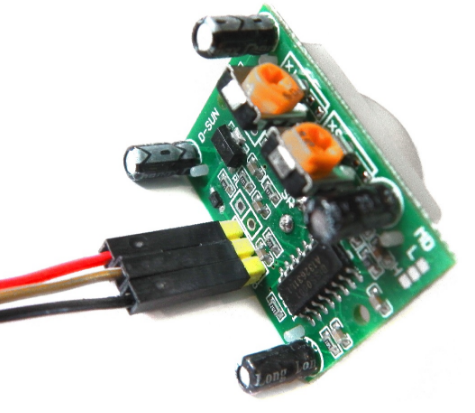
# Nightlight Simulation with Human Presence Detection



**Hardware Required**

* 1 LED
* 1 LDR
* 110K register
* 1 PIR

**Understanding the configuration of PIR sensor:**

****

**Connection:**

1. Attach one leg of LDR to 5V and another leg to Arduino Analog pin A0
2. Attach one leg of 110K register with that leg of LDR connected to A0
3. Attach another leg of register to the ground
4. Connect the positive leg of LED to pin 11 and negative to GND
5. Connect positive leg of PIR to 5V and negative leg to GND
6. Connect output pin of PIR to digital pin 3

**Code:**

int LDR = 0; //analog pin to which LDR is connected, here we set it to 0 so it means A0

int LDRValue = 0; //that’s a variable to store LDR values

int light\_sensitivity = 500; //This is the approx value of light surrounding your LDR

//the time we give the sensor to calibrate (10-60 secs according to the datasheet)

int calibrationTime = 30;

//the time when the sensor outputs a low impulse

long unsigned int lowIn;

//the amount of milliseconds the sensor has to be low

//before we assume all motion has stopped

long unsigned int pause = 5000;

boolean lockLow = true;

boolean takeLowTime;

int pirPin = 3; //the digital pin connected to the PIR sensor's output

int ledPin = 11;

void setup()

{

Serial.begin(9600); //start the serial monitor with 9600 buad

pinMode(11, OUTPUT);

pinMode(pirPin, INPUT);

pinMode(ledPin, OUTPUT);

digitalWrite(pirPin, LOW);

Serial.print("calibrating sensor ");//give the sensor some time to calibrate

for(int i = 0; i<calibrationTime; i++){

Serial.print(".");

delay(1000);

}

Serial.println(" done");

Serial.println("SENSOR ACTIVE");

delay(50);

}

void loop()

{

LDRValue = analogRead(LDR); //reads the ldr’s value through LDR

// Serial.println(LDRValue); //prints the LDR values to serial monitor

if(digitalRead(pirPin) == HIGH && LDRValue < light\_sensitivity){

digitalWrite(ledPin, HIGH);

if(lockLow){

//makes sure we wait for a transition to LOW before any further output is made:

lockLow = false;

Serial.println("---");

Serial.print("motion detected at ");

Serial.print(millis()/1000);

Serial.println(" sec");

delay(50);

}

takeLowTime = true;

}

if(digitalRead(pirPin) == LOW || LDRValue >= light\_sensitivity){

digitalWrite(ledPin, LOW); //the led visualizes the sensors output pin state

if(takeLowTime){

lowIn = millis(); //save the time of the transition from high to LOW

takeLowTime = false; //make sure this is only done at the start of a LOW phase

}

//if the sensor is low for more than the given pause,

//we assume that no more motion is going to happen

if(!lockLow&&millis() - lowIn> pause){

//makes sure this block of code is only executed again after

//a new motion sequence has been detected

lockLow = true;

Serial.print("motion ended at "); //output

Serial.print((millis() - pause)/1000);

Serial.println(" sec");

delay(50);

}

delay(100);

}

}

**Steps and Observation:**

1. Upload the program to Arduino and open Serial monitor
2. It will show in Serial monitor: **calibrating sensor .............................. done**
3. Wait until it is shown: **SENSOR ACTIVE**
4. Caution: Avoid any movements near the sensor**. It should not show: “Motion detected at” in monitor**
5. Switch off lights avoiding motion near sensor, LED should not glow
6. Make some movement near PIR
7. LED should keep glowing for some time
8. While LED is ON, switch on the light, LED should switch off immediately.

**3. Home automation**

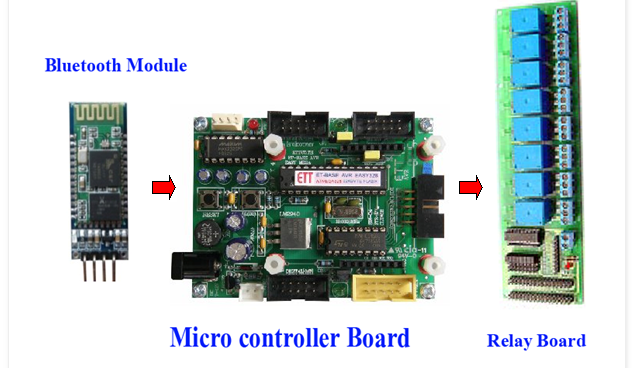
**Aim:**

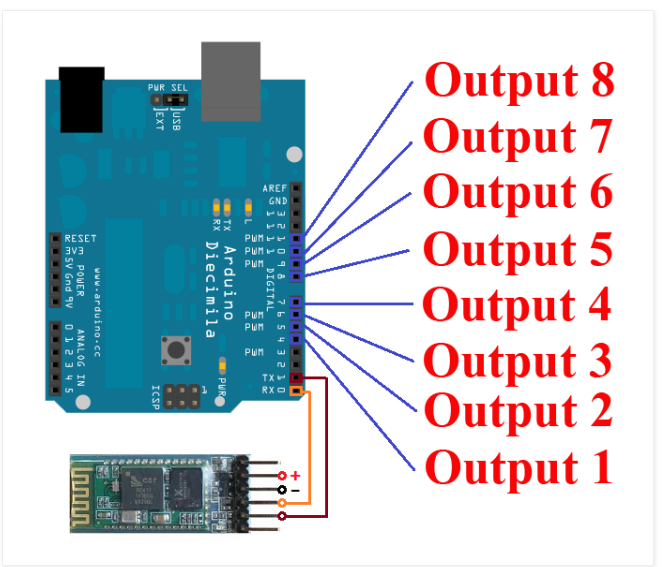
To control the working of relay through Android Mobile.

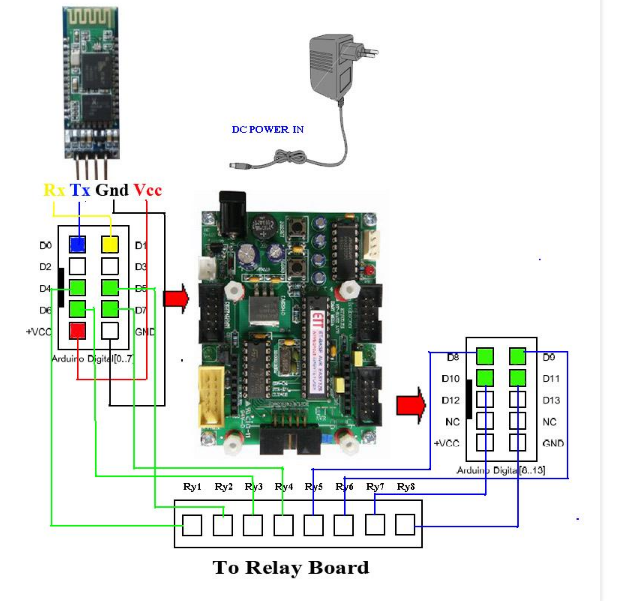
**Hardware Required:**

* Arduino
* 4-Channel relay
* Bluetooth Module
* Android phone

**Circuit diagram**







Wiring Arduino Pin Diagram

Output 1 to Pin 4 ( Arduino Board )  
Output 2 to Pin 5  
Output 3 to Pin 6  
Output 4 to Pin 7

Bluetooth Module Tx to Pin 0  
Bluetooth Module Rx to Pin 1  
======================

VCC of Bluetooth & relay should be connected to Arduino 5V(through breadboard)

GND of Bluetooth & relay should be connected to Arduino GND

**Important:**

* **First upload the code to Arduino and then connect Bluetooth module**
* **Install Arduino Bluetooth device App in your android mobile**

A data transmission via Bluetooth.  
Device1 ON sent “a” , Device1 OFF sent “A”   
Device2 ON sent “b” , Device2 OFF sent “B”   
Device3 ON sent “c” , Device3 OFF sent “C”   
Device4 ON sent “d” , Device4 OFF sent “D”

**Code:**

/\*

Simple LED test

\*/

char val; // variable to receive data from the serial port

int ledpin = 2; // LED connected to pin 2 (on-board LED)

void setup()

{

pinMode(ledpin = 2, OUTPUT); // pin 2 (on-board LED) as OUTPUT

pinMode(ledpin = 3, OUTPUT); // pin 3 (on-board LED) as OUTPUT

pinMode(ledpin = 4, OUTPUT); // pin 4 (on-board LED) as OUTPUT

pinMode(ledpin = 5, OUTPUT); // pin 5 (on-board LED) as OUTPUT

Serial.begin(9600); // start serial communication at 9600bps

}

void loop()

{

if( Serial.available() ) // if data is available to read

{

;

}

val = Serial.read(); // read it and store it in 'val'

if( val == 'a' ) // if 'a' was received led 2 is switched off

{

digitalWrite(ledpin = 2, HIGH); // turn Off pin 2

}

if( val == 'A' ) // if 'A' was received led 2 on

{

digitalWrite(ledpin = 2, LOW); // turn ON pin 2

}

if( val == 'b' ) // if 'b' was received led 3 is switched off

{

digitalWrite(ledpin = 3, HIGH); // turn Off pin 3

}

if( val == 'B' ) // if 'B' was received led 3 on

{

digitalWrite(ledpin = 3, LOW); // turn ON pin 3

}

if( val == 'C' ) // if 'C' was received

{

digitalWrite(ledpin=4, LOW); // turn Off pin 4

}

if( val == 'D' ) // if 'D' was received

{

digitalWrite(ledpin=5, LOW); // turn Off pin 5

}

if( val == 'c' ) // if 'c' was received

{

digitalWrite(ledpin = 4, HIGH); // turn On pin 4

}

if( val == 'd' ) // if 'd' was received

{

digitalWrite(ledpin = 5, HIGH); // turn ON pin 5

}

}

**4. IR Remote Controlled LED**

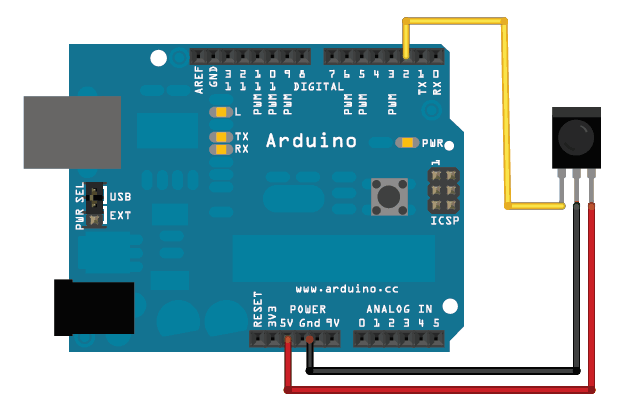
**Aim:**

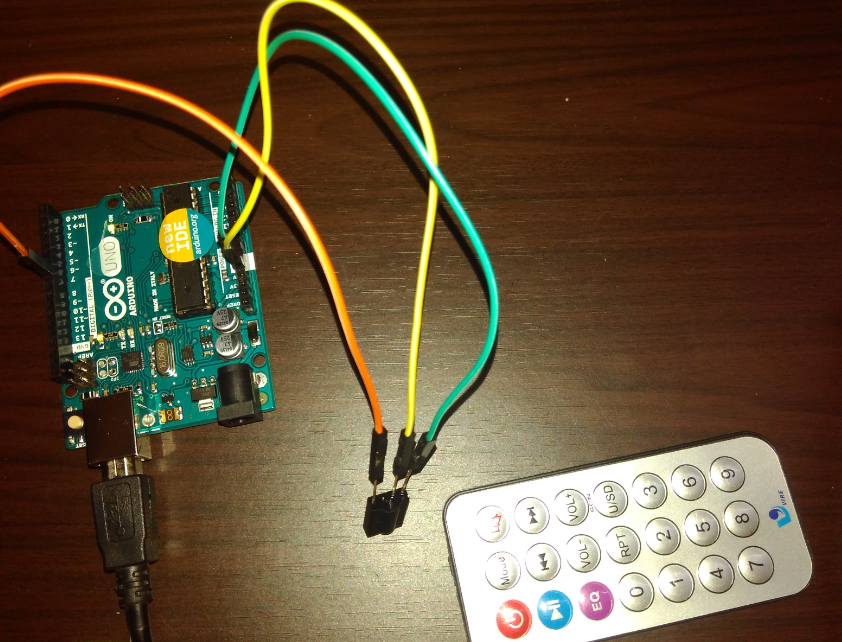
Controlling the LED using IR remote

**Hardware Required:**

* Arduino
* Any IR remote
* IR receiver
* Breadboard
* Jumper Cables
* LED

**Circuit diagram:**





**Code:**

Arduino has been making some changes to the IDE, and now there is a conflicting IR library. I won't be using this library, because I still want to use the IRremote by Ken Sheriff. I've had great success with it in the past, and so have many others. So let's get this file deleted.

**Windows:** C:\Program Files (x86)\Arduino\libraries\RobotIRremote

Once you have located the folder RobotIRremote, delete it. Restart the Arduino IDE and your RobotIRremote library should be gone.

## Installing the IR Library

The very first thing that we need to do associating with Arduino is to download the IR library. To make things simpler, I have included a .zip of the IR library. Download it to your computer, unzip it, then place it in your Arduino libraries folder. Don't know where it is?

Open up the Arduino IDE and on the menu select **Sketch>IncludeLibrary>Add Library** and select the 'IRremote' folder.

Here is the IR remote folder

<http://www.mediafire.com/download/jd5j7911amju36g/IRremote.zip>

## Connection:

The IR sensor's pins are attached to Arduino as so: (from left to right with the sensor's head facing you)

(Vout) Pin 1 to pin 11(Arduino)  
(GND) Pin 2 to GND(Arduino)  
(Vcc) Pin 3 to 5v(Arduino)

**Code:**

/\*

Some Sample code of how to use your IR remote

\* Lets get started:

The IR sensor's pins are attached to Arduino as so:

Pin 1 to Vout (pin 7 on Arduino)

Pin 2 to GND

Pin 3 to Vcc (+5v from Arduino)

\*/

#include <IRremote.h>

int IRpin = 7; // pin for the IR sensor

int LED = 13; // LED pin

IRrecv irrecv(IRpin);

decode\_results results;

boolean LEDon = true; // initializing LEDon as true

void setup()

{

Serial.begin(9600);

irrecv.enableIRIn(); // Start the receiver

pinMode(LED, OUTPUT);

}

void loop()

{

if (irrecv.decode(&results))

{

Serial.println(results.value);

irrecv.resume(); // Receive the next value

}

if (results.value == 33444015) // change the number according to your IR remote button number

{

if (LEDon == true) // is LEDon equal to true?

{

LEDon = false;

digitalWrite(LED, HIGH);

delay(100); // keeps the transistion smooth

}

else

{

LEDon = true;

digitalWrite(LED, LOW);

delay(100);

}

}

}

## Instruction: Upload the program to Arduino board. Open Serial Monitor. Press any one

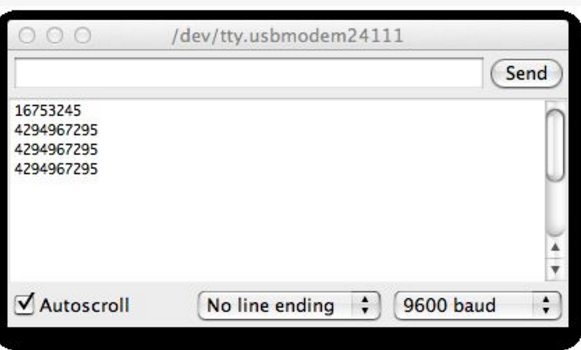
## button(say 0). See the first number getting printed when you press 0 each time. Note the

## number, and replace the number at the portion highlighted in red color in the code.

## Now upload the code once again. Now you should see LED attached to pin 13 getting

## toggled each time you press button 0 in remote !!!

## Recognizing IR Signals



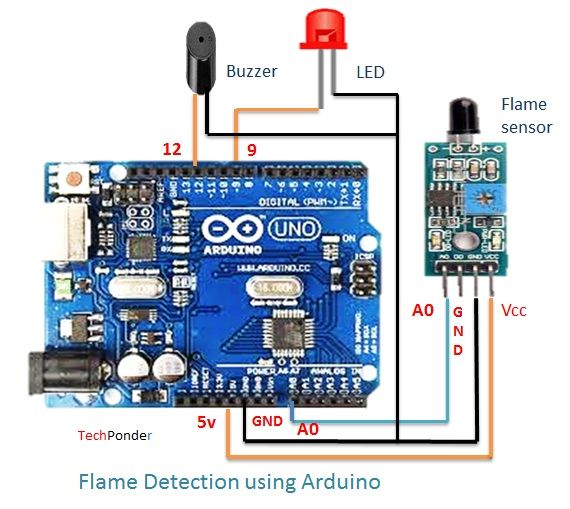
**5. Fire Alert**

**Aim**

Fire alarm simulation

**Hardware Required**

* Flame sensor (Analogue Output)
* Arduino
* Bread board
* LED
* Buzzer
* Connecting wires

**Circuit Diagram**

**Flame sensor interfacing to Arduino**

Flame sensor to Arduino

vcc -> vcc

gnd -> gnd

A0 -> A0

**Led interfacing to Arduino**

**LED +ve** is connected to **9th pin** of Arduino

**LED -ve** is connected to **gnd pin** of arduino

**Buzzer interfacing to Arduino**

**Buzzer +ve** is connected to **12th pin** of Arduino

**Buzzer -ve** is connected to **GND** pin of Arduino

**Code:**

#include<SoftwareSerial.h>

int sensorPin = A0; // select the input pin for the LDR

int sensorValue = 0; // variable to store the value coming from the sensor

int led = 9; // Output pin for LED

int buzzer = 12; // Output pin for Buzzer

void setup() {

// declare the ledPin and buzzer as an OUTPUT:

pinMode(led, OUTPUT);

pinMode(buzzer,OUTPUT);

Serial.begin(9600);

}

void loop()

{

sensorValue = analogRead(sensorPin);

Serial.println(sensorValue);

if (sensorValue < 100)

{

Serial.println("Fire Detected");

Serial.println("LED on");

digitalWrite(led,HIGH);

digitalWrite(buzzer,HIGH);

delay(1000);

}

digitalWrite(led,LOW);

digitalWrite(buzzer,LOW);

delay(sensorValue);

}

**6. Automatic irrigation controller simulation**

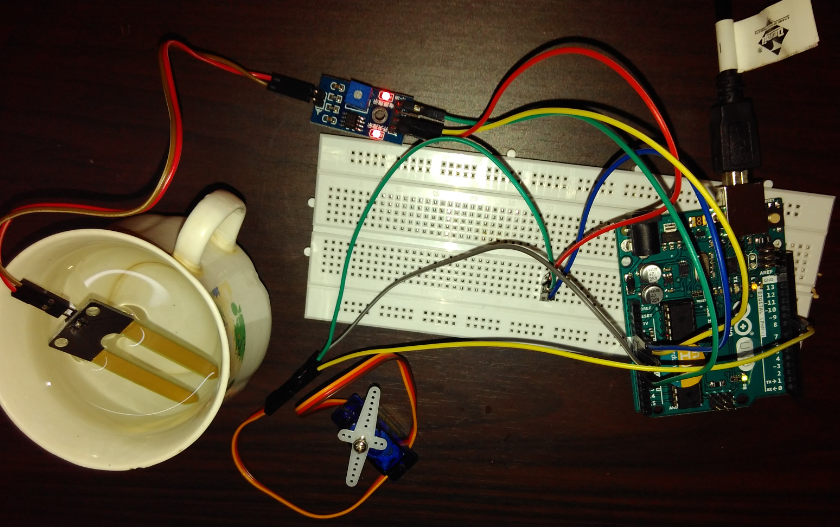
**Aim**

Sensing the soil moisture and sprinkling the Water simulation

**Hardware Required**

* Arduino
* Moisture Sensor
* Breadboard
* Min servo motor

**Circuit diagram**



Moisture sensor VCC to Arduino 5V

Moisture sensor GND to Arduino GND

Moisture sensor A0 to Arduino A0

Servo motor VCC to Arduino 5V

Servo motor GND to Arduino GND

Servo Motor Signal to Arduino digital pin 9

Code:

#include <Servo.h>

Servo myservo; // create servo object to control a servo

// twelve servo objects can be created on most boards

int pos = 0; // variable to store the servo position

int sensorPin = A0; // select the input pin for the potentiometer

int sensorValue = 0; // variable to store the value coming from the sensor

void setup() {

myservo.attach(9); // attaches the servo on pin 9 to the servo object

Serial.begin(9600);

}

void loop() {

// read the value from the sensor:

sensorValue = analogRead(sensorPin);

Serial.println (sensorValue);

if(sensorValue>500)

{

for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees

// in steps of 1 degree

myservo.write(pos); // tell servo to go to position in variable 'pos'

delay(15); // waits 15ms for the servo to reach the position

}

for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees

myservo.write(pos); // tell servo to go to position in variable 'pos'

delay(15); // waits 15ms for the servo to reach the position

}

}

delay (1000);

}

**7. Reverse parking sensor (Using LCD)**

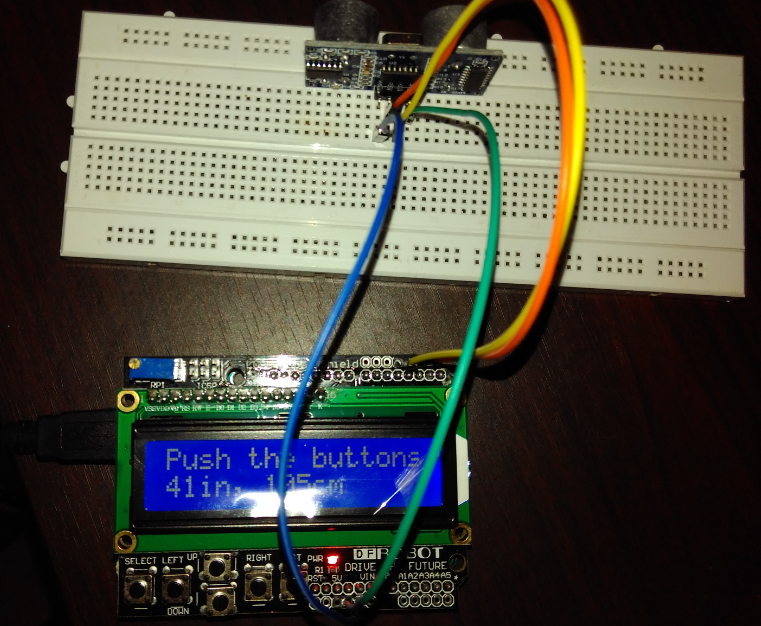
**Aim**

Measuring the distance using Ultrasonic sensor which helps in reverse parking system.

**Hardware Required**

* Arduino
* Ultrasonic Sensor
* Breadboard
* LCD

**Circuit diagram**

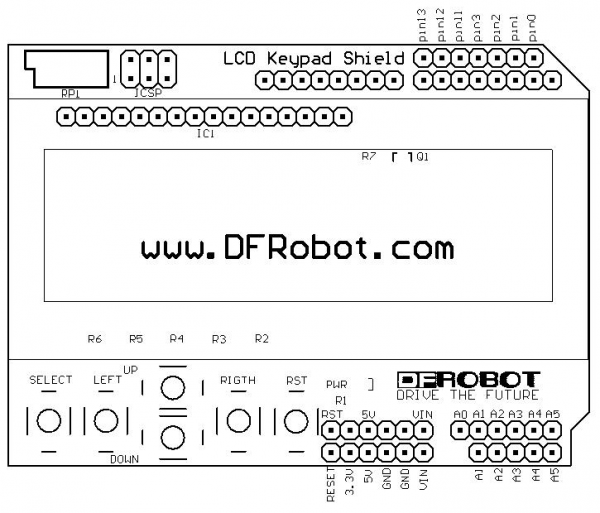
****

**Echo-pin 2**

**Trigger-pin 3**

**Vcc-5 V**

**Gnd-Gnd**



**Code:**

#include <LiquidCrystal.h>

const int trigPin = 3;

const int echoPin = 2;

LiquidCrystal lcd(8, 9, 4, 5, 6, 7); // select the pins used on the LCD panel

void setup() {

lcd.begin(16, 2); // start the library

lcd.setCursor(0,0); // set the LCD cursor position

}

void loop()

{

// establish variables for duration of the ping,

// and the distance result in inches and centimeters:

long duration, inches, cm;

lcd.setCursor(0,1); // move cursor to second line "1" and 0 spaces over

// The sensor is triggered by a HIGH pulse of 10 or more microseconds.

// Give a short LOW pulse beforehand to ensure a clean HIGH pulse:

pinMode(trigPin, OUTPUT);

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

// Read the signal from the sensor: a HIGH pulse whose

// duration is the time (in microseconds) from the sending

// of the ping to the reception of its echo off of an object.

pinMode(echoPin, INPUT);

duration = pulseIn(echoPin, HIGH);

// convert the time into a distance

inches = microsecondsToInches(duration);

cm = microsecondsToCentimeters(duration);

lcd.print(inches);

lcd.print("in, ");

lcd.print(cm);

lcd.print("cm");

delay(1000);

}

long microsecondsToInches(long microseconds)

{

return microseconds / 74 / 2;

}

long microsecondsToCentimeters(long microseconds)

{ return microseconds / 29 / 2;}

**8. Color recognition**

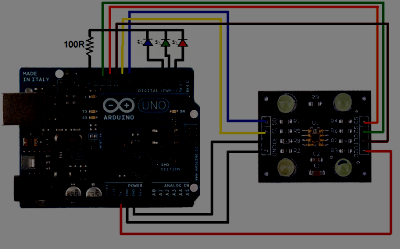
**Aim**

Identify the RGB Colors

**Hardware Required**

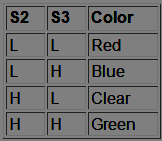
* RGB LED  
  Resistor 100 ohm  
  Color sensor TCS230  
  Arduino board

**Circuit diagram:**



Recognizes the basic colors RGB.

This is color sensor that used TCS230 and RGB LED.  
color detection in this module depend on S2 and S3 output as follow:

`  
L means low or 0 and the H means high or 1. you must use a 100ohm resistor to protect the LEDs.  


**Code:**

/\*// TCS230 color recognition sensor

// Sensor connection pins to Arduino are shown in comments

Color Sensor      Arduino

-----------      --------

 VCC               5V

 GND               GND

 s0                8

 s1                9

 s2                12

 s3                11

 OUT               10

 OE                GND

\*/

const int s0 = 8;

const int s1 = 9;

const int s2 = 12;

const int s3 = 11;

const int out = 10;

// LED pins connected to Arduino

int redLed = 2;

int greenLed = 3;

int blueLed = 4;

// Variables

int red = 0;

int green = 0;

int blue = 0;

void setup()

{

  Serial.begin(9600);

  pinMode(s0, OUTPUT);

  pinMode(s1, OUTPUT);

  pinMode(s2, OUTPUT);

  pinMode(s3, OUTPUT);

  pinMode(out, INPUT);

  pinMode(redLed, OUTPUT);

  pinMode(greenLed, OUTPUT);

  pinMode(blueLed, OUTPUT);

  digitalWrite(s0, HIGH);

  digitalWrite(s1, HIGH);

}

void loop()

{

  color();

  Serial.print("R Intensity:");

  Serial.print(red, DEC);

  Serial.print(" G Intensity: ");

  Serial.print(green, DEC);

  Serial.print(" B Intensity : ");

  Serial.print(blue, DEC);

  //Serial.println();

  if (red < blue && red < green && red < 20)

  {

   Serial.println(" - (Red Color)");

   digitalWrite(redLed, HIGH); // Turn RED LED ON

   digitalWrite(greenLed, LOW);

   digitalWrite(blueLed, LOW);

  }

  else if (blue < red && blue < green)

  {

   Serial.println(" - (Blue Color)");

   digitalWrite(redLed, LOW);

   digitalWrite(greenLed, LOW);

   digitalWrite(blueLed, HIGH); // Turn BLUE LED ON

  }

  else if (green < red && green < blue)

  {

   Serial.println(" - (Green Color)");

   digitalWrite(redLed, LOW);

   digitalWrite(greenLed, HIGH); // Turn GREEN LED ON

   digitalWrite(blueLed, LOW);

  }

  else{

  Serial.println();

  }

  delay(300);

  digitalWrite(redLed, LOW);

  digitalWrite(greenLed, LOW);

  digitalWrite(blueLed, LOW);

 }

void color()

{

  digitalWrite(s2, LOW);

  digitalWrite(s3, LOW);

  //count OUT, pRed, RED

  red = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);

  digitalWrite(s3, HIGH);

  //count OUT, pBLUE, BLUE

  blue = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);

  digitalWrite(s2, HIGH);

  //count OUT, pGreen, GREEN

  green = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);

}

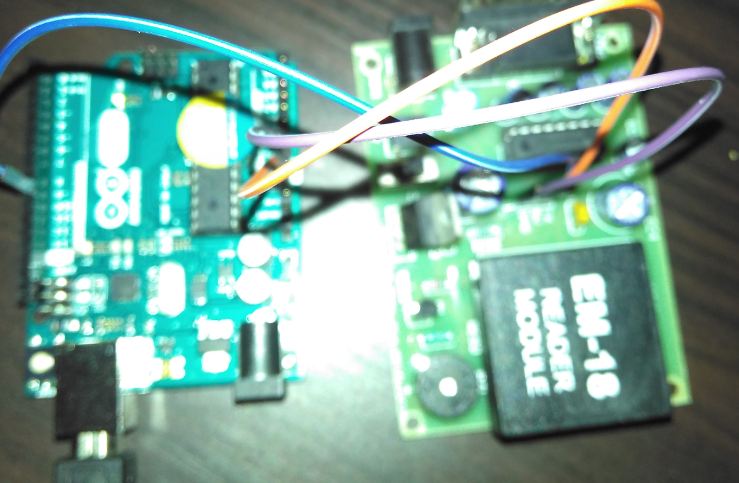
**9. RFID Reader and RFID tag count**

**Aim:**

To count the number of RFID tags read by the RFID reader

**Hardware Required:**

* Arduino
* RFID Reader module
* RFID Tags

**Circuit diagram:**

**Code:**

**#include <SoftwareSerial.h> //Reading sinngle tag**

SoftwareSerial mySerial(9, 10);

void setup()

{

  mySerial.begin(9600); // Setting the baud rate of Software Serial Library

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

 }void loop()

{

 if(mySerial.available()>0)

  {

  Serial.write(mySerial.read());

  }

}

**// Code for reading multiple tags:**

#include<SoftwareSerial.h>

SoftwareSerial mySerial(9,10);

int read\_count=0,tag\_count=0;

int j=0,k=0; // Variabvles to iterate in for loops

char data\_temp, RFID\_data[12], data\_store[10][12];

boolean disp\_control;

void setup()

{

mySerial.begin(9600);

Serial.begin(9600);

}

void loop()

{

RecieveData();

StoreData();

PrintData();

}

void RecieveData()

{

if(mySerial.available()>0)

{

data\_temp=mySerial.read();

RFID\_data[read\_count]=data\_temp;

read\_count++;

}

}

void StoreData()

{

if(read\_count==12)

{

disp\_control=true;

for(k=tag\_count;k<=tag\_count;k++)

{

for(j=0;j<12;j++)

{

data\_store[k][j]=RFID\_data[j];

}

}

read\_count=0;

tag\_count++;

}

}

void PrintData()

{

if(disp\_control==true)

{

  for(k=0;k<=tag\_count;k++)

{

    for(j=0;j<12;j++)

  {

    Serial.write(data\_store[k][j]);

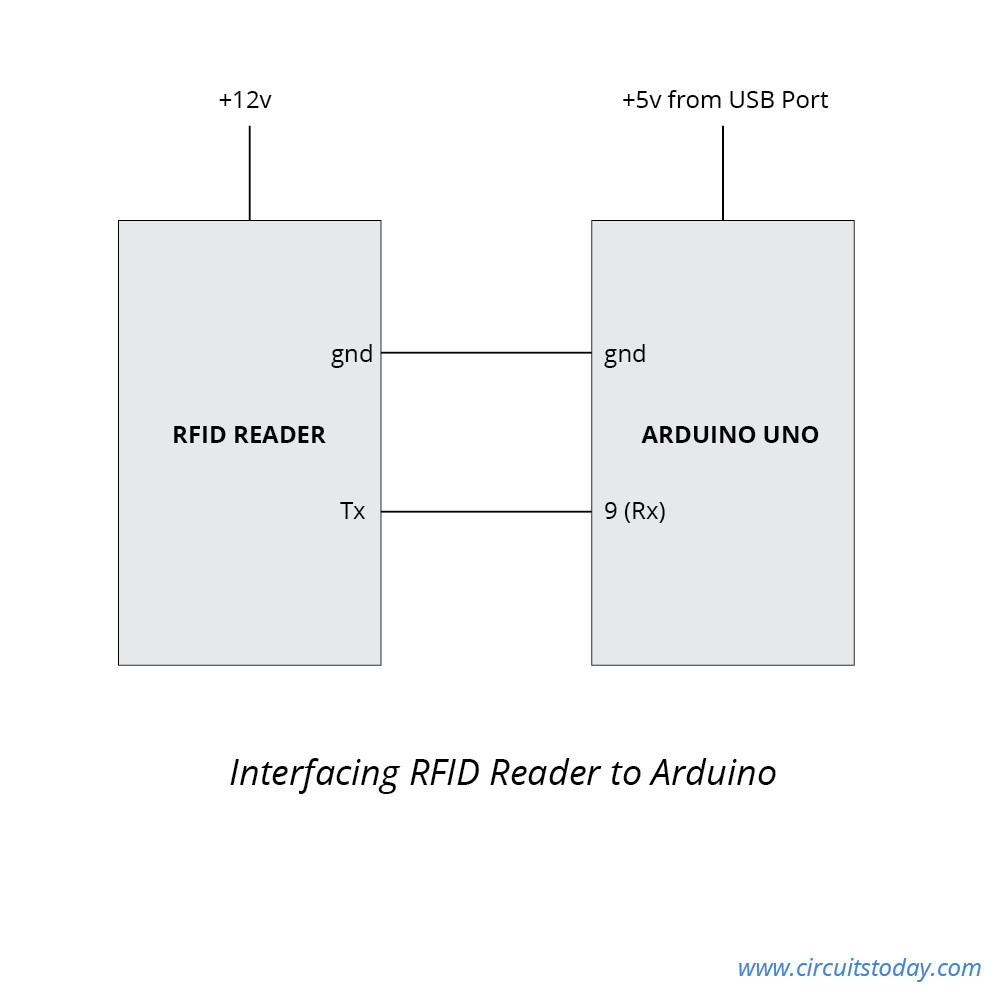
  }

  Serial.println();

}

disp\_control=false;

}

}

**10. Arduino with Vibration sensor**

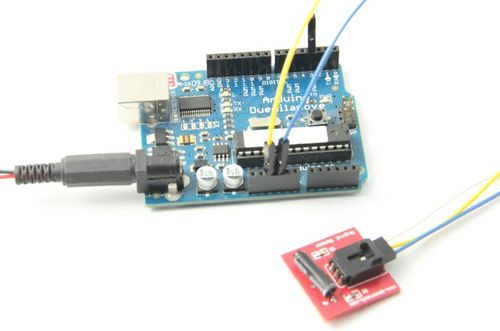
**Aim:**

Simulated the purpose of Vibration sensor by tapping

**Hardware Required**

* Arduino
* Vibration sensor

**Circuit diagram**



**Code:**

int ledPin = 13; // Connect LED to pin 13

int switcher = 3; // Connect Tilt sensor to Pin3

void setup()

{

pinMode(ledPin, OUTPUT); // Set digital pin 13 to output mode

pinMode(switcher, INPUT); // Set digital pin 3 to input mode

}

void loop()

{

if(digitalRead(switcher)==HIGH) //Read sensor value

{

digitalWrite(ledPin, HIGH); // Turn on LED when the sensor is tilted

delay(300);

}

else {

digitalWrite(ledPin, LOW); // Turn off LED when the sensor is not triggered } }

**11. Humidity and Temperature logging in a cloud**

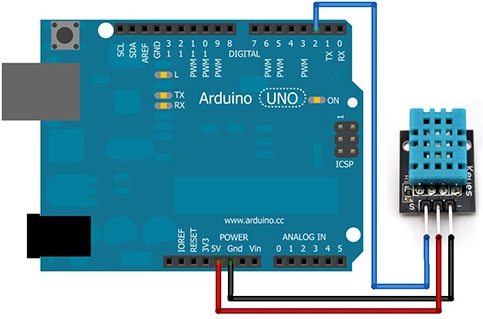
**Aim:**

Storing the sensed sensor reading in the cloud.

**Hardware Required:**

* Arduino
* Humidity and temperature sensor

**Circuit diagram: (Using Ethernet shield)**



1. Create an account in <https://thingspeak.com/>
2. Create one new channel
3. Note down WriteAPI key and Channel ID
4. Attach Ethernet Shield on Arduino
5. Make connection with temperature and humidity sensor as in circuit diagram.
6. Install DHT11 library – See

http://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-an-arduino/

1. Include the library in your code. Go to sketch->Include library and include DHT library

**Code:**

#include <dht.h>

#include <SPI.h>

#include <Ethernet.h>

dht DHT;

#define DHT11\_PIN 7

// Local Network Settings

byte mac[] = { 0xD4, 0x28, 0xB2, 0xFF, 0xA0, 0xA1 }; // Must be unique on local network

// ThingSpeak Settings

char thingSpeakAddress[] = "api.thingspeak.com";

String writeAPIKey = "**IZMTTVC12FFAHTZH**"; //Replace with yours

unsigned long myChannelNumber = **132940**; //Replace with yours

const int updateThingSpeakInterval = 16 \* 1000; // Time interval in milliseconds to update ThingSpeak (number of seconds \* 1000 = interval)

// Variable Setup

long lastConnectionTime = 0;

boolean lastConnected = false;

int failedCounter = 0;

// Initialize Arduino Ethernet Client

EthernetClient client;

void setup(){

Serial.begin(9600);

// Start Ethernet on Arduino

startEthernet();

}

void loop()

{

int chk = DHT.read11(DHT11\_PIN);

Serial.print("Temperature = ");

Serial.println(DHT.temperature);

Serial.print("Humidity = ");

Serial.println(DHT.humidity);

char t\_buffer[10];

char h\_buffer[10];

float t=(DHT.temperature);

String temp=dtostrf(t,0,5,t\_buffer);

float h=(DHT.humidity);

String humid=dtostrf(h,0,5,h\_buffer);

delay(1000);

// Disconnect from ThingSpeak

if (!client.connected() && lastConnected)

{

Serial.println("...disconnected");

Serial.println();

client.stop();

}

// Update ThingSpeak

if(!client.connected() && (millis() - lastConnectionTime > updateThingSpeakInterval))

{

updateThingSpeak("field1="+temp+"&field2="+humid);

}

// Check if Arduino Ethernet needs to be restarted

if (failedCounter > 3 ) {startEthernet();}

lastConnected = client.connected();

}

void updateThingSpeak(String tsData)

{

if (client.connect(thingSpeakAddress, 80))

{

client.print("POST /update HTTP/1.1\n");

client.print("Host: api.thingspeak.com\n");

client.print("Connection: close\n");

client.print("X-THINGSPEAKAPIKEY: "+writeAPIKey+"\n");

client.print("Content-Type: application/x-www-form-urlencoded\n");

client.print("Content-Length: ");

client.print(tsData.length());

client.print("\n\n");

client.print(tsData);

lastConnectionTime = millis();

if (client.connected())

{

Serial.println("Connecting to ThingSpeak...");

Serial.println();

failedCounter = 0;

}

else

{

failedCounter++;

Serial.println("Connection to ThingSpeak failed ("+String(failedCounter, DEC)+")");

Serial.println();

}

}

else

{

failedCounter++;

Serial.println("Connection to ThingSpeak Failed ("+String(failedCounter, DEC)+")");

Serial.println();

lastConnectionTime = millis();

} }

void startEthernet()

{

client.stop();

Serial.println("Connecting Arduino to network...");

Serial.println();

delay(1000);

// Connect to network amd obtain an IP address using DHCP

if (Ethernet.begin(mac) == 0)

{

Serial.println("DHCP Failed, reset Arduino to try again");

Serial.println();

}

else

{

Serial.println("Arduino connected to network using DHCP");

Serial.println();

}

delay(1000);

}

After uploading, check your channel status in thingsspeak.

**11.2 Using Wifi**

**CLOUD (SPARKFUN)**

**Cloud Data:**

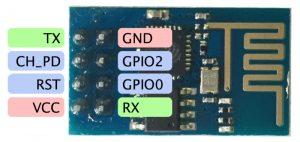
[**https://data.sparkfun.com/streams/QGyLQKYb71F2Q1qMqQER**](https://data.sparkfun.com/streams/QGyLQKYb71F2Q1qMqQER)

**Pushing data to cloud (Only for reference)**

Pusing data to cloud is done by the program in Arduino. However sometime we might want to push manually from browser just for testing. Use below url for that

[**https://data.sparkfun.com/input/QGyLQKYb71F2Q1qMqQER?private\_key=JqyBXeWg9ViqeBy9yevR&temperature=33.33**](https://data.sparkfun.com/input/QGyLQKYb71F2Q1qMqQER?private_key=JqyBXeWg9ViqeBy9yevR&temperature=33.33)

**Pin Diagram of ESP8266**

****

**Connections**

| **Arduino** | **ESP8266** | **DHT Temperature sensor** |
| --- | --- | --- |
| 5V | CH\_PD, RST, VCC |  |
| 3.3V |  | VCC |
| GND | GND | GND |
| 2 (Software Serial RX) | TX |  |
| 3 (Software Serial TX) | RX |  |
| 5 |  | DATA |

**IMPORTANT**

Change the WIFI name and password in the program before uploading. Also the program might show disconnect few times

POST /input/QGyLQKYb71F2Q1qMqQER?private\_key=JqyBXeWg9ViqeBy9yevR&temperature=29.0000 HTTP/1.0

Host: data.sparkfun.com

connect timeout

**but will succeed and say**

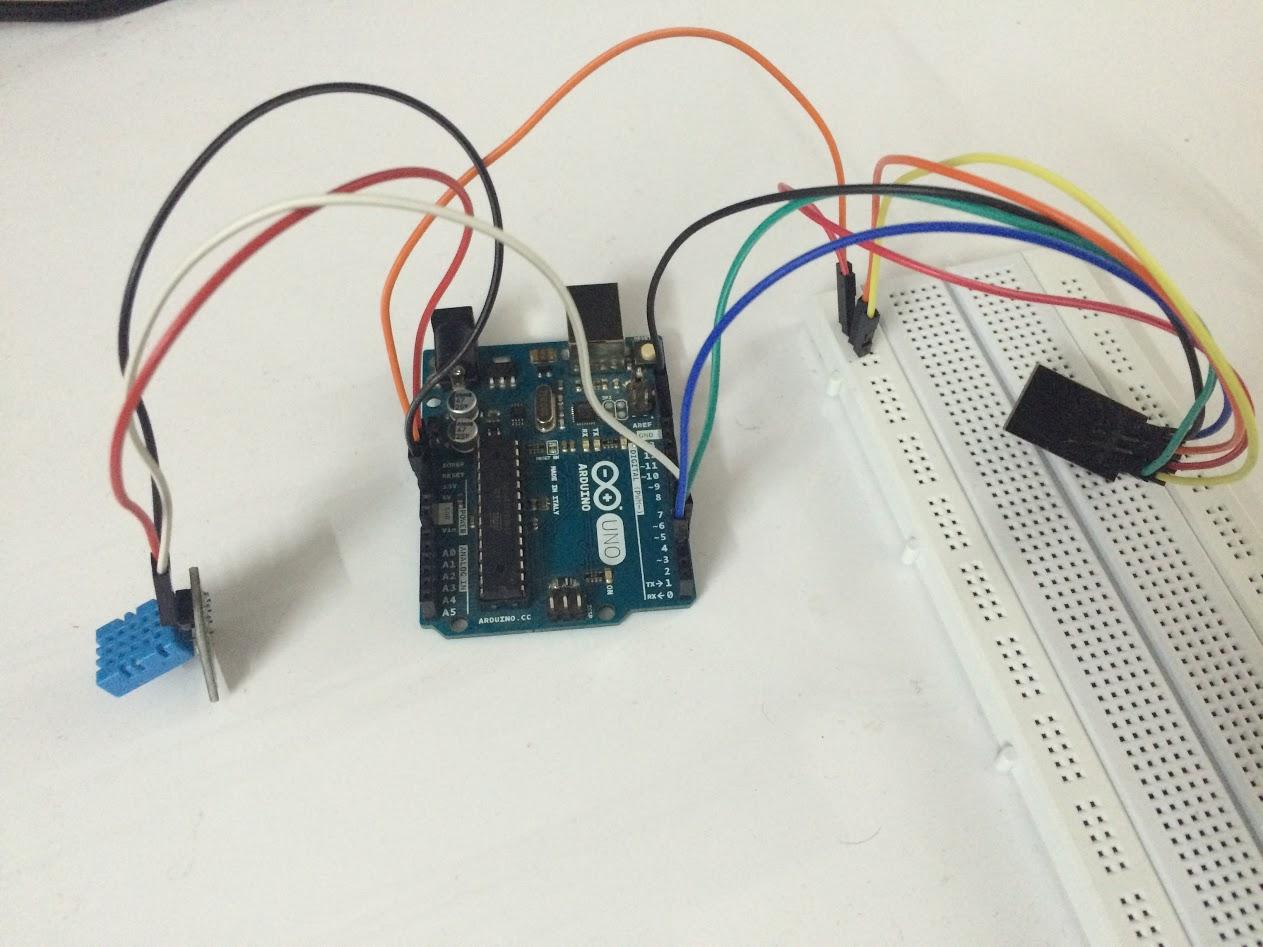
POST /input/QGyLQKYb71F2Q1qMqQER?private\_key=JqyBXeWg9ViqeBy9yevR&temperature=29.0000 HTTP/1.0

Host: data.sparkfun.com

>

Recv 126 aytes

SEND OK



**IMPORTANT**

**When uploading the sketch you might have to disconnect the VCC to ESP8266 to avoid upload failure**

**Code**

#include <SoftwareSerial.h>

//use mega Serial 2 for serial monitor; Serial 1 on pins 19 (RX) and 18 (TX);// Serial2 on pins 17 (RX) and 16 (TX), Serial3 on pins 15 (RX) and 14 (TX).

//#define SSID "SSID"

//#define PASS "Password"

#include <dht.h>

dht DHT;

#define DHT11\_PIN 5

#define SSID "NAME OF WIFI"

#define PASS "PASSWORD"

#define DST\_IP "data.sparkfun.com" //baidu.com

SoftwareSerialSerial2(2,3);

float temp;

intpintemp = A0;

//SoftwareSerialdbgSerial(10, 11); // RX, TX

// Use DHT11 sensor

void setup()

{

// Open serial communications and wait for port to open:

//serial 2 is to esp8266

Serial2.begin(115200);//9600 (mine), 57600, 115200

Serial2.setTimeout(1000);

//serial 0 is to usb

Serial.begin(9600);

while(!Serial);

while(!Serial2);

//dbgSerial.begin(9600); //can't be faster than 19200 for softserial

//dbgSerial.println("ESP8266 Demo");

Serial.println("ESP8266 Demo on Mega2560");

while(Serial2.available()>0)

Serial2.read();

Serial2.println("AT");

delay(1000);

while (Serial2.available())

{

char c = Serial2.read();

//dbgSerial.write(c);

Serial.write(c);

//if(c=='\r') dbgSerial.print('\n');

if(c=='\r') Serial.print('\n');

}

Serial.print('\n');

delay(1000);

Serial2.flush();

Serial2.println("AT+RST");

delay(1000);

while (Serial2.available())

{

char c = Serial2.read();

//dbgSerial.write(c);

Serial.write(c);

//if(c=='\r') dbgSerial.print('\n');

if(c=='\r') Serial.print('\n');

}

Serial.print('\n');

delay(1000);

Serial2.flush();

Serial2.println("AT+GMR");

delay(1000);

while (Serial2.available())

{

char c = Serial2.read();

//dbgSerial.write(c);

Serial.write(c);

//if(c=='\r') dbgSerial.print('\n');

if(c=='\r') Serial.print('\n');

}

Serial.print('\n');

delay(1000);

Serial2.flush();

Serial2.println("AT+CIFSR");

delay(1000);

while (Serial2.available())

{

char c = Serial2.read();

//dbgSerial.write(c);

Serial.write(c);

//if(c=='\r') dbgSerial.print('\n');

if(c=='\r') Serial.print('\n');

}

Serial.print('\n');

delay(1000);

Serial2.flush();

Serial2.println("AT+CWMODE=3");

delay(1000);

while (Serial2.available())

{

char c = Serial2.read();

//dbgSerial.write(c);

Serial.write(c);

//if(c=='\r') dbgSerial.print('\n');

if(c=='\r') Serial.print('\n');

}

Serial.print('\n');

delay(2000);

Serial2.flush();

Serial2.println("AT+CWLAP");

delay(2000);

while (Serial2.available())

{

char c = Serial2.read();

//dbgSerial.write(c);

Serial.write(c);

//if(c=='\r') dbgSerial.print('\n');

if(c=='\r') Serial.print('\n');

}

Serial.print('\n');

delay(2000);

Serial2.flush();

String cmd="AT+CWJAP=\"";

cmd+=SSID;

cmd+="\",\"";

cmd+=PASS;

cmd+="\"";

Serial2.println(cmd);

Serial.println(cmd);

delay(2000);

while (Serial2.available())

{

char c = Serial2.read();

//dbgSerial.write(c);

Serial.write(c);

//if(c=='\r') dbgSerial.print('\n');

if(c=='\r') Serial.print('\n');

}

delay(2000);

Serial.print('\n');

Serial2.flush();

/\*

//test if the module is ready

Serial2.println("AT+RST");

//delay(1000);

//delay(1000);

Serial.println("Resetting module");

Serial2.flush();

//if(Serial2.find("ready"))

if(Serial2.find("Ready")||Serial2.find("ready"))

{

//dbgSerial.println("Module is ready");

Serial.println("Module is ready");

}

else

{

//dbgSerial.println("Module have no response.");

Serial.println("Module have no response.");

while(1);

}

delay(1000);

//connect to the wifi

boolean connected=false;

for(inti=0;i<5;i++)

{

if(connectWiFi())

{

connected = true;

break;

}

}

if (!connected){while(1);}

delay(5000);

//print the ipaddr

/\*

Serial2.println("AT+CIFSR");

Serial.println("ip address:");

while (Serial2.available())

Serial.write(Serial2.read());

//set the single connection mode

Serial2.println("AT+CIPMUX=0");

\*/

}

void loop()

{

// Read temperature as Celsius

Serial.print("DHT11, \t");

intchk = DHT.read11(DHT11\_PIN);

switch (chk)

{

case DHTLIB\_OK:

Serial.print("OK,\t");

break;

case DHTLIB\_ERROR\_CHECKSUM:

Serial.print("Checksum error,\t");

break;

case DHTLIB\_ERROR\_TIMEOUT:

Serial.print("Time out error,\t");

break;

case DHTLIB\_ERROR\_CONNECT:

Serial.print("Connect error,\t");

break;

case DHTLIB\_ERROR\_ACK\_L:

Serial.print("Ack Low error,\t");

break;

case DHTLIB\_ERROR\_ACK\_H:

Serial.print("Ack High error,\t");

break;

default:

Serial.print("Unknown error,\t");

break;

}

int temp = DHT.temperature;

#if 1

String cmd = "AT+CIPSTART=\"TCP\",\"";

cmd += DST\_IP;

cmd += "\",80";

Serial2.println(cmd);

//dbgSerial.println(cmd);

Serial.println(cmd);

if(Serial2.find("Error")) return;

//cmd = "GET / HTTP/1.0\r\n\r\n";

//cmd = "POST /input/QGyLQKYb71F2Q1qMqQER?private\_key=JqyBXeWg9ViqeBy9yevR&temperature=40 HTTP/1.0\r\n\r\n Host: data.sparkfun.com\r\n\r\n";

char s[200];

Serial.println(temp);

int t1 = temp;

float f2 = temp-t1;

int t2 = trunc(f2 \* 10000);

sprintf (s, "POST /input/QGyLQKYb71F2Q1qMqQER?private\_key=JqyBXeWg9ViqeBy9yevR&temperature=%d.%04d HTTP/1.0\r\n\r\n Host: data.sparkfun.com\r\n\r\n", t1,t2);

Serial.println(s);

cmd = s;

Serial.println(cmd);

Serial2.print("AT+CIPSEND=");

Serial2.println(cmd.length());

if(Serial2.find(">"))

{

//dbgSerial.print(">");

Serial.print(">");

}else

{

Serial2.println("AT+CIPCLOSE");

//dbgSerial.println("connect timeout");

Serial.println("connect timeout");

delay(1000);

return;

}

Serial2.print(cmd);

delay(2000);

//Serial.find("+IPD");

while (Serial2.available())

{

char c = Serial2.read();

//dbgSerial.write(c);

Serial.write(c);

//if(c=='\r') dbgSerial.print('\n');

if(c=='\r') Serial.print('\n');

}

//dbgSerial.println("====");

Serial.println("====");

delay(1000);

#endif

}

booleanconnectWiFi()

{

Serial2.println("AT+CWMODE=1");

String cmd="AT+CWJAP=\"";

cmd+=SSID;

cmd+="\",\"";

cmd+=PASS;

cmd+="\"";

//dbgSerial.println(cmd);

Serial2.println(cmd);

Serial.println(cmd);

delay(2000);

if(Serial2.find("OK"))

{

//dbgSerial.println("OK, Connected to WiFi.");

Serial.println("OK, Connected to WiFi.");

return true;

}else

{

//dbgSerial.println("Can not connect to the WiFi.");

Serial.println("Can not connect to the WiFi.");

return false;

}

}