INTELLIGENT AND WEATHER ADAPTIVE STREET LIGHTING SYSTEM

**BY**

T. Rashmika

D. Suma Bala

D. A. Vineela

**ABSTRACT**

In present scenario the street lights are being operated manually i.e., when the switch is on the lights gets illuminated and vice versa .In most of the cases there is a chance of forgetting the things which results in wastage of Electricity.

So in order to eradicate this problem we have come up with a new solution which makes our lives easier .Through this project we can make the smart street lights and control these street lights accordingly. We can control the lights wirelessly.

The smart street lighting also controls the luminosity of light and performs automatic turning on and turning off of lights which is an aspect that serves to reduce energy consumption. Logically, this system may save a large amount of the electrical power.

We are creating a mesh network with the NRF modules so that if one street light which acts as coordinator is made ON then automatically the other street lights which are acting as routers will also receive data from the main module and switch on the street lights which are present there. From this router the data will be transmitted to other router and process will be continued.

**PROBLEM STATEMENT**

We have seen in the number of cities where the street lights is the one of the huge energy expense for a city. Currently we have manual system where the light will be switched ON in the evening before the sunset and they are switched OFF next day morning after there is sufficient light outside. So there is lot of energy waste between ON and OFF timing.

**WORKING**

The system architecture consists of NRF24L01 modules, NODEMCU 1.0, ARDUINO UNO, BASIC SHIELDS. LDR‟s are light dependent devices whose resistance decreases when light falls on them and increases in the dark. When a light dependent resistor is kept in dark, its resistance is very high. Here the signal gets transmitted from NRF module of NODEMCU to NRF module of Arduino Uno through antenna.

NODE MCU 1.0

NRF24L01

BASIC SHIELD

ARDUINO UNO

NRF24L01

BASIC SHIELD

SIGNAL

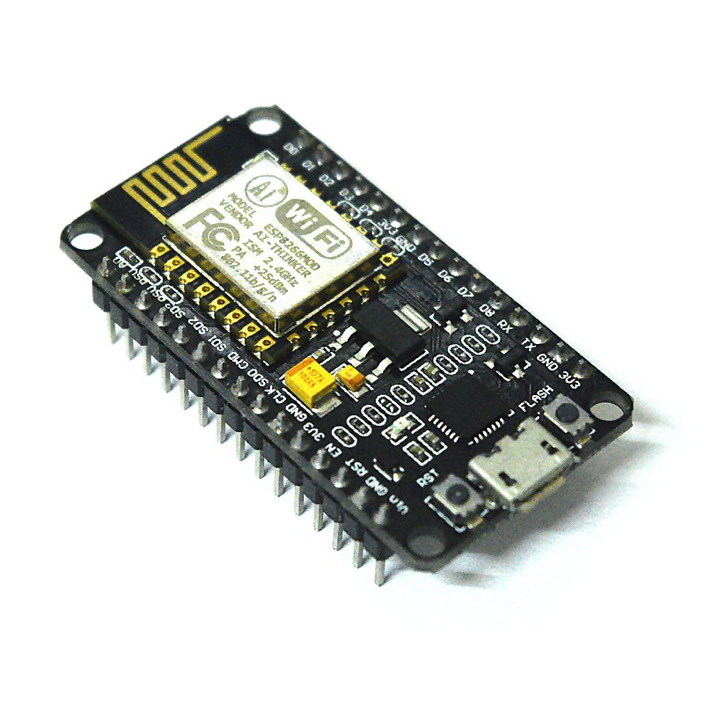
**HARDWARE COMPONENTS**

**NodeMCU ESP8266 ESP-12E**

NodeMCU is an open source IoT platform. It includes firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module. The term “NodeMCU” by default refers to the firmware rather than the DevKit. The firmware uses the Lua scripting language. It is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266. It uses many open source projects, such as lua-cjson, and spiffs.

**Features**

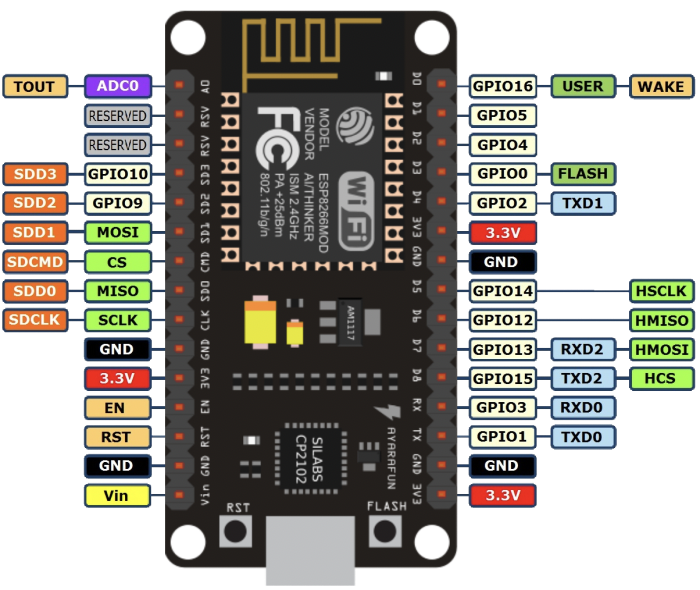
* Version : DevKit v1.0
* Breadboard Friendly
* Light Weight and small size.
* 3.3V operated, can be USB powered.
* Uses wireless protocol 802.11b/g/n.
* Built-in wireless connectivity capabilities.
* Built-in PCB antenna on the ESP-12E chip.
* Capable of PWM, I2C, SPI, UART, 1-wire, 1 analog pin.
* Uses CP2102 USB Serial Communication interface module.
* Arduino IDE compatible (extension board manager required).
* Supports Lua (alike node.js) and Arduino C programming language.



**Working Principle:**

The ESP8266 can be controlled from your local Wi-Fi network or from the internet (after port forwarding). The ESP-01 module has GPIO pins that can be programmed to turn an LED or relay ON/OFF through the internet. The module can be programmed using an Arduino/USB-to-TTL converter through the serial pins (RX, TX).

**PINOUT DIAGRAM**



**Specification:**

* Voltage:3.3V.
* Wi-Fi Direct (P2P), soft-AP.
* Current consumption: 10uA~170mA.
* Flash memory attachable: 16MB max (512K normal).
* Integrated TCP/IP protocol stack.
* Processor: Tensilica L106 32-bit.
* Processor speed: 80~160MHz.
* RAM: 32K + 80K.
* GPIOs: 17 (multiplexed with other functions).
* Analog to Digital: 1 input with 1024 step resolution.
* +19.5dBm output power in 802.11b mode
* 802.11 support: b/g/n.
* Maximum concurrent TCP connections: 5.

**ARDUINO UNO R3**

* Arduino is used for building different types of electronic circuits easily using of both a physical programmable circuit board usually microcontroller and piece of code running on computer with USB connection between the computer and Arduino.
* Programming language used in Arduino is just a simplified version of C++ that can easily replace thousands of wires with words.

**FEATURES:**

**Peripheral Features**:

* Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
* One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
* Real Time Counter with Separate Oscillator
* Six PWM Channels
* 8-channel 10-bit ADC in TQFP and QFN/MLF package
* Temperature Measurement
* 6-channel 10-bit ADC in PDIP Package
* Temperature Measurement
* Programmable Serial USART
* Master/Slave SPI Serial Interface
* Byte-oriented 2-wire Serial Interface (Philips I2 C compatible)
* Programmable Watchdog Timer with Separate On-chip Oscillator
* On-chip Analog Comparator
* Interrupt and Wake-up on Pin Change

**Special Microcontroller Features:**

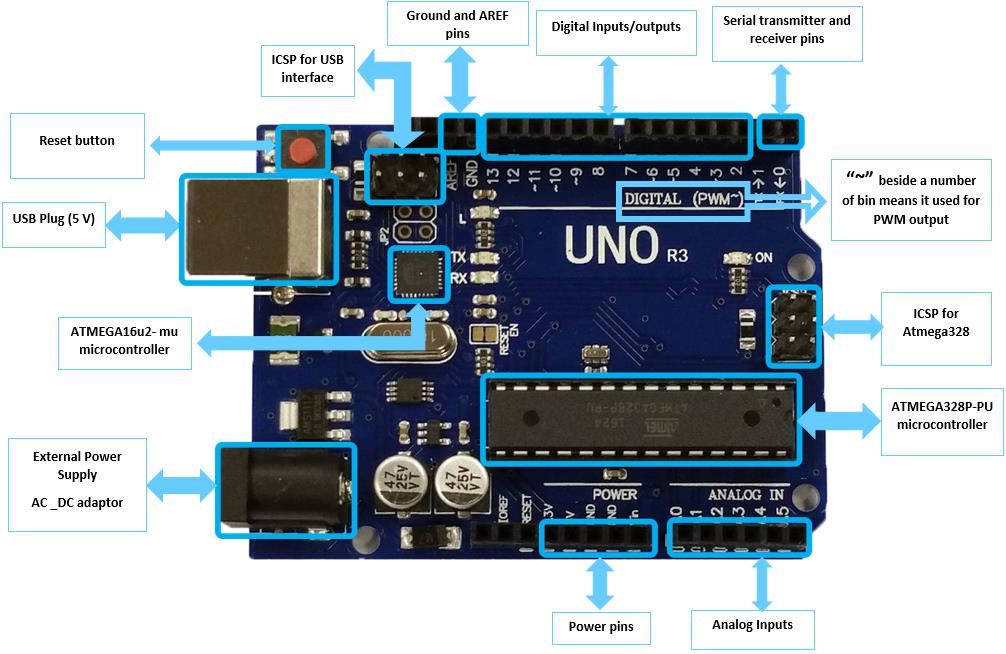
* Power-on Reset and Programmable Brown-out Detection
* Internal Calibrated Oscillator
* External and Internal Interrupt Sources
* Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby

**Working Principle:**

The **Arduino** Uno board is a microcontroller based on the ATmega328. It has 14 digital input/output pins in which 6 can be used as PWM outputs, a 16 MHz ceramic resonator, an ICSP header, a USB connection, 6 analog inputs, a power jack and a reset button. This contains all the required support needed for microcontroller.

**Specifications:**

* Microcontroller: ATmega328
* Operating Voltage: 5V
* Input Voltage (recommended): 7-12V
* Input Voltage (limits): 6-20V
* Digital I/O Pins: 14 (of which 6 provide PWM output)
* Analog Input Pins: 6
* DC Current per I/O Pin: 40 mA
* DC Current for 3.3V Pin: 50 mA
* Flash Memory: 32 KB of which 0.5 KB used by bootloader
* SRAM: 2 KB (ATmega328)
* EEPROM: 1 KB (ATmega328)
* Clock Speed: 16 MHz



**nRF24L01 MODULE:**

nRF24L01 is a single chip radio transceiver for the world wide 2.4 - 2.5 GHz ISM band. The transceiver consists of a fully integrated frequency synthesizer, a power amplifier, a crystal oscillator, a demodulator, modulator and Enhanced ShockBurstTM protocol engine. Output power, frequency channels, and protocol setup are easily programmable through a SPI interface. Current consumption is very low, only 9.0mA at an output power of -6dBm and 12.3mA in RX mode. Built-in Power Down and Standby modes makes power saving easily realizable.

**Features:**

* Worldwide 2.4GHz ISM band operation
* 250kbps, 1Mbps and 2Mbps on air data rates
* Ultra low power operation
* 11.3mA TX at 0dBm output power
* 13.5mA RX at 2Mbps air data rate
* 900nA in power down
* 26μA in standby-I
* On chip voltage regulator
* 1.9 to 3.6V supply range
* Enhanced ShockBurstTM
* Automatic packet handling
* Auto packet transaction handling
* 6 data pipe MultiCeiverTM
* Drop-in compatibility with nRF24L01
* On-air compatible in 250kbps and 1Mbps with nRF2401A, nRF2402, nRF24E1 and nRF24E2
* Low cost BOM
* ±60ppm 16MHz crystal
* 5V tolerant inputs
* Compact 20-pin 4x4mm QFN package

**Working principle:**

It uses the 2.4 GHz band and it can operate with baud rates from 250 kbps up to 2 Mbps. If used in open space and with lower baud rate its range can reach up to 100 meters.

**Specifications:**

* Antenna load impedance = 15Ω+j88Ω.
* Average data rate 10kbps and max. payload length packets. Average current consumption during TX startup (130μs) and when changing mode from RX to TX (130μs).
* Average current consumption during RX startup (130μs) and when changing mode from TX to RX (130μs).



**Code:**

**Transmitter code**

#include <ESP8266WiFi.h>-------

#include <PubSubClient.h>

#include <SPI.h>

#include <nRF24L01.h>

#include <RF24.h>

String command;

RF24 radio(D4,D8); // CE, CSN

const byte address[6] = "RSV76";

//-------- Customise these values -----------

const char\* ssid = "Bala";

const char\* password = "suma1899";

#include "DHT.h"

#define DHTPIN D2 // what pin we're connected to

#define DHTTYPE DHT11 // define type of sensor DHT 11

DHT dht (DHTPIN, DHTTYPE);

#define ORG "pgx60q"

#define DEVICE\_TYPE "vineela"

#define DEVICE\_ID "1229"

#define TOKEN "123456789"

//-------- Customise the above values --------

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";

char topic[] = "iot-2/evt/Data/fmt/json";

char authMethod[] = "use-token-auth";

char token[] = TOKEN;

char clientId[] = "d:" ORG ":" DEVICE\_TYPE ":" DEVICE\_ID;

WiFiClientwifiClient;

PubSubClientclient(server, 1883,wifiClient);

void setup() {

Serial.begin(115200);

Serial.println();

dht.begin();

Serial.print("Connecting to ");

Serial.print(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.print("WiFi connected, IP address: ");

Serial.println(WiFi.localIP());

radio.begin();

radio.openWritingPipe(address);

radio.setPALevel(RF24\_PA\_MIN);

radio.stopListening(); //disabling receiving mode

Serial.begin(9600);

pinMode(D0,OUTPUT);

}

void loop() {

intldrvalue=analogRead(A0);

PublishData(ldrvalue);

radio.write(&ldrvalue, sizeof(ldrvalue));

if(ldrvalue<200)

{

digitalWrite(D0,LOW);

}

else

digitalWrite(D0,HIGH);

delay(1000);

}

void PublishData(intldrvalue){

if (!!!client.connected()) {

Serial.print("Reconnecting client to ");

Serial.println(server);

while (!!!client.connect(clientId, authMethod, token)) {

Serial.print(".");

delay(500);

}

Serial.println();

}

String payload = "{\"d\":{\"ldrvalue\":";

payload += ldrvalue;

payload += "}}";

Serial.print("Sending payload: ");

Serial.println(payload);

if (client.publish(topic, (char\*) payload.c\_str())) {

Serial.println("Publish ok");

} else {

Serial.println("Publish failed");

}

}

**Receiver code:**

include <SPI.h>

#include <nRF24L01.h>

#include <RF24.h>

RF24 radio(7,8); // CE, CSN

const byte address[6] = "RSV76";

void setup() {

Serial.begin(9600);

radio.begin();

radio.openReadingPipe(0, address);

radio.setPALevel(RF24\_PA\_MIN);

radio.startListening();

pinMode(4, OUTPUT);

}

void loop() {

if (radio.available()) {

char text[32] = "";

radio.read(&text, sizeof(text));

Serial.println(text);

delay(1000);

}

digitalWrite(2,HIGH);

}