ELECTRONICS ENGINEERING

Session 2022-23

A PROJECT REPORT
on

AC Load Control System with Wi Fi Control and IR Remote

Submitted in
Partial Fulfillment of the Requirements
For the Diploma of
ELECTRONICS ENGINEERING

UNDER THE ABLE GUIDANCE OF
Er. MUKESH KUMAR YADAV
(PRINCIPAL)

SUBMITTED TO: -
Er. BHUPENDRA KUMAR
H.O.D. OF ELECTRONICS ENGG. DEPT.

UNDER THE GUIDANCE OF:
Er. SANGEETA CHAUDHARY
LECT. OF ELECTRONICS ENGG. DEPT.

PREPARED BY:

VIPAN KUMAR KUSHAWAHA
S/O RADHA CHARAN
ENROLLEMENT No. E20112033000049

Department of Electronics Engineering
M. G. POLYTECHNIC HATHRAS (U. P.) 204101

DECLARATION

I hereby declare that the work presented in this report entitled "AC load control system with wifi control and IR remote ", submitted by me to M.G. Polytechnic , Vipan Kumar Kushawaha in partial fulfillment of the requirement for the award of the diploma in electronics engineering is a record of bon fide project work carried out by me under the guidance of Mr. Mukesh Kumar Yadav (principal), Mr. Bhupendra Kumar H.O.D. (Electronics Engg. Department), miss. Sangeeta Chaudhary & Mr. Har Veer Singh lecturer (Electronics Engg. Department). I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other diploma or degree in this institute or any other institute or university.

Name: Vipan Kumar Kushawaha
Roll No. E20112033000049
Branch: Electronics Engineering
Vipan Kumar
(Candidate Signature)

CERTIFICATE

This is to Certify that the project titled " **AC Load Control System with WiFi Control and IR Remote** " is the bona fide work **VIPAN KUMAR KUSHAWAHA(E20112033000049**) a student of diploma (Electronics Engineering) of M. G. Polytechnic, Hathras affiliated to Board of Technical Education Lucknow during the academic year 2023, in fulfillment of the requirements for the award of the diploma of Electronics Engineering and that the project has not formed the basis for the award previously of any other degree, diploma, fellowship or any other similar titles.

Place: Hathras

Date: --/08/2023

(Signature of the Examiner) (Signature of the Guide)

ABSTRACT

The AC Load Control System with WiFi Control and IR Remote is an innovative project designed to enable efficient control of four AC load switches using a relay module, with options for WiFi control, IR remote control, and manual switchboard control. This project utilizes a 5V power supply, an ESP8266 microcontroller, an IR sensor, a relay unit, and a switchboard. This report presents a comprehensive overview of the project, including the circuit diagram, Arduino code, and a professional analysis of the system's functionality.

The objective of this project is to create a versatile system that allows users to control AC load switches remotely. The system offers multiple control options for convenience and flexibility. With WiFi and IR remote control capabilities, users can easily operate the AC loads from a distance, eliminating the need for physical interaction.

ACKNOWLEDGEMENTS

It is the contribution of many persons that make a work successful. I wish to express my gratitude to individuals who have contributed their ideas, time and energy in this work.

First, I wish to express my heartfelt gratitude towards**, Er. Sangeeta Chaudhary,** my supervisors for their active interest, constructive guidance and advice during every stage of this work. Their valuable guidance coupled with active and timely review of my work provided the necessary motivation for me to work on and successfully complete the project.

I sincerely thank to the Principal of M. G. Polytechnic Hathras, **Er. Mukesh Kumar Yadav** to provide a project Oriented infrastructure.

I would like to express my special gratitude and thanks to the H.O.D. of the department, Electronics Engineering, **Er. Bhupendra Kumar** for giving me attention and time as and when needed.

I would like to thank the other faculty member of the department, namely **Er. Har Veer Singh and Er. Poonam Yadav** for their valuable suggestions.

Words are insufficient to express my gratitude to my beloved parents for their inspiration and blessings. I thank God, who has supported me at every moment. Last but not least I would like to thank all my friends and well-wishers who were involved directly or indirectly in successful completion of the present work.

VIPAN KUMAR KUSHAWAHA

TABLE OF CONTENTS

LIST OF FIGURES

Figure Name :

CHAPTER 1: INTRODUCTION

1.1. What is AC Load Control System with WiFi Control and IR Remote?

The AC Load Control System with WiFi Control and IR Remote is a sophisticated project that aims to provide remote control capabilities for controlling four AC load switches. The system incorporates a relay module, allowing users to conveniently operate the AC loads from a distance using either WiFi or an IR remote control.

1.2. Objective

The objective of this project is to develop a versatile and user-friendly AC load control system that offers multiple control options. The system aims to provide remote control functionality through WiFi and IR remote control, while also offering manual control through a switchboard.

1.3. Literature Survey

A comprehensive literature survey was conducted to understand the existing technologies and systems related to AC load control. The survey focused on studying relevant research papers, articles, and technical documentation related to WiFi-based control systems, IR remote control, and microcontroller applications. This survey helped in identifying the current trends, challenges, and potential solutions in the field of AC load control.

1.4. Motivation

The motivation behind developing the AC Load Control System with WiFi Control and IR Remote stems from the increasing demand for efficient and convenient control of AC loads in various settings. By integrating WiFi and IR remote control, this system aims to enhance user convenience and flexibility in operating AC loads remotely, thereby improving energy management and overall efficiency.

1.5. Proposed Idea

The proposed idea for this project involves the development of a control system that combines the power of WiFi, IR remote control, and manual switchboard control. By utilizing the ESP8266 microcontroller, the system enables users to remotely control four AC load switches. The WiFi control allows users to send commands to the microcontroller over a WiFi network, while the IR remote control allows them to use an IR remote to send specific commands. Additionally, a switchboard is provided for manual control of the AC loads. The integration of these control options provides users with flexibility and convenience in managing their AC loads.

CHAPTER 2: HARDWARE

2.1 Components of the AC Load Control System

The AC Load Control System comprises several key components that are essential for its operation.

2.2.1 5V Power Supply

2.2.2 Microcontroller (ESP8266)

2.2.3 IR Sensor

2.2.4 Relay Module

2.2.5 Switchboard

2.2.1 5V Power Supply

The 5V power supply is responsible for providing a stable DC voltage to power the microcontroller, relay module, and other components. It converts the available input voltage into the required 5V DC voltage, ensuring the smooth functioning of the system.

Input voltage: 220-250 V AC

Output voltage: 5V DC

1.1 5V Power Supply

2.2.2 Microcontroller (ESP8266)

The microcontroller plays a crucial role in the system as it serves as the control unit. In this project, the ESP8266 microcontroller is used due to its integrated WiFi functionality. It receives commands from the WiFi module and IR sensor, processes them, and controls the relay module accordingly. The microcontroller acts as the interface between the user and the AC loads.

ESP8266

The ESP8266 is a low-cost Wi-Fi microchip, with built-in TCP/IP networking software, and microcontroller capability, produced by Espressif Systems in Shanghai, China.

The chip was popularized in the English-speaking maker community in August 2014 via the ESP-01 module, made by a third-party manufacturer Ai-Thinker.

This small module allows microcontrollers to connect to a Wi-Fi network and make simple TCP/IP connections using Hayes-style commands.

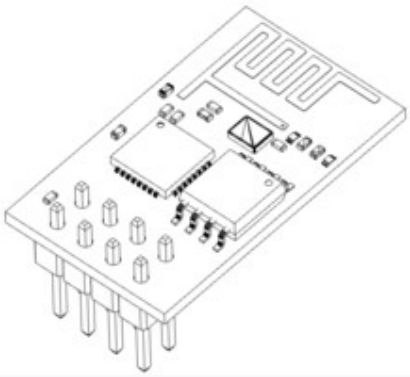
However, at first, there was almost no English-language documentation on the chip and the commands it accepted.

The very low price and the fact that there were very few external components on the module, which suggested that it could eventually be very inexpensive in volume, attracted many hackers to explore the module, the chip, and the software on it, as well as to translate the Chinese documentation.

The ESP8285 is a similar chip with a built-in 1 MiB flash memory, allowing the design of single-chip devices capable of connecting via Wi-Fi.

These microcontroller chips have been succeeded by the ESP32 family of devices.

Features



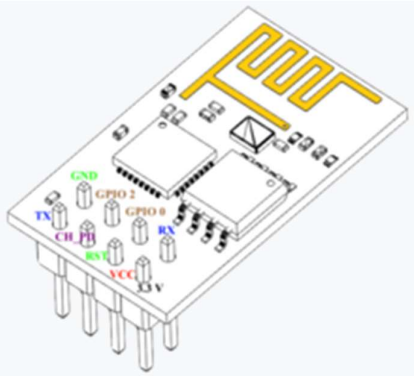
ESP-12e

- Processor: L106 32-bit RISC microprocessor core based on the Tensilica Diamond Standard 106Micro running at 80 or 160 MHz
- Memory:[6]
  - 32 KB instruction RAM

- o 32 KB instruction cache RAM
- o 80 KB user-data RAM
- o 16 KB ETS system-data RAM
- External QSPI flash: up to 16 MiB is supported (512 KiB to 4 MiB typically included)
- IEEE 802.11 b/g/n Wi-Fi
- o Integrated TR switch, balun, LNA, power amplifier and matching network
- o WEP or WPA/WPA2 authentication, or open networks
- 17 GPIO pins
- Serial Peripheral Interface Bus (SPI)
- I²C (software implementation)
- I²S interfaces with DMA (sharing pins with GPIO)
- UART on dedicated pins, plus a transmit-only UART can be enabled on GPIO2
- 10-bit ADC (successive approximation ADC)

Standard 106Micro running at 80 or 160 MHz

Pinout of ESP-01



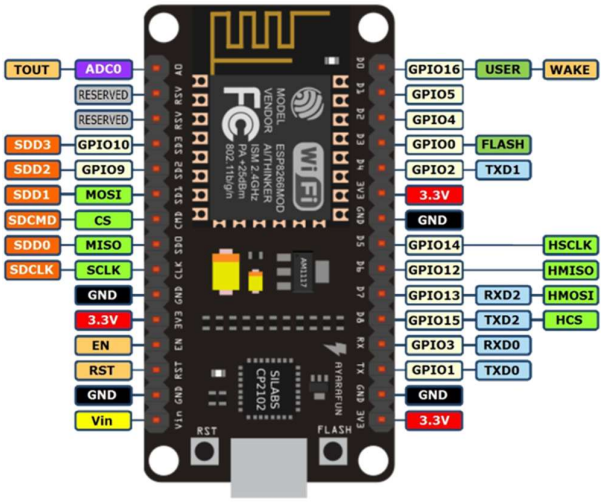
1.3 ESP-01 module pinout

The pinout is as follows for the common ESP-01 module:

1. GND, Ground (0 V)
2. GPIO 2, General-purpose input/output No. 2
3. GPIO 0, General-purpose input/output No. 0
4. RX, Receive data in, also GPIO3

5. VCC, Voltage (+3.3 V; can handle up to 3.6 V)
6. RST, Reset
7. CH_PD, Chip power-down
8. TX, Transmit data out, also GPIO1

Pinout of esp8266 wifi module



1.4 Pinout of ESP8266

2.2.3 IR Sensor

The IR sensor detects infrared signals from an IR remote control. It receives the commands transmitted by the remote control and converts them into electrical signals. These signals are then interpreted by the microcontroller to determine the appropriate action to be taken for controlling the AC loads.

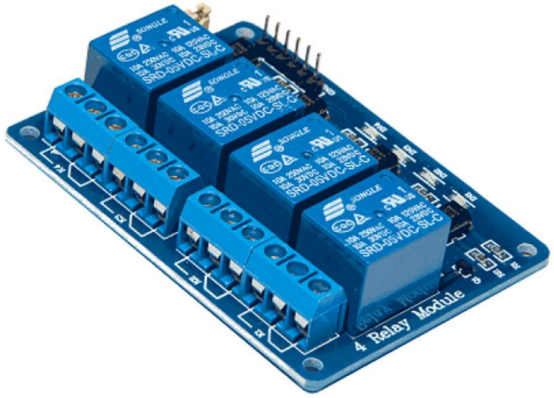
1.5 IR SENSOR

2.2.4 Relay Module

The relay module is responsible for switching the AC loads on and off. It provides isolation between the low-voltage microcontroller and the high-voltage AC loads. The microcontroller controls the relay module by sending appropriate signals to energize or de-energize the relays, thereby controlling the flow of current to the AC loads.

Load configuration:

Single relay: 220V AC 7 Ampere



1.6 Relay Module

2.2.5 Switchboard

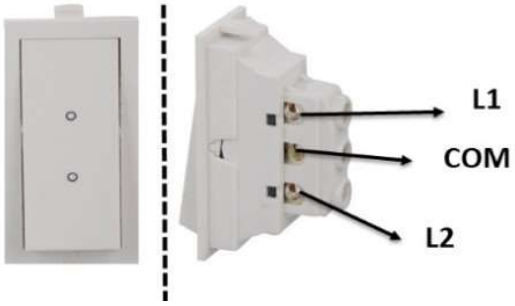
The switchboard serves as a manual control option for the AC loads. It consists of physical switches that allow users to directly control the state of the AC loads. This serves as a backup control method if WiFi or IR control is not available or preferred.

1.7 Switchboard

In this project we use two way switch board which are compatible with relay module to provide 2 way control.

Two way switch

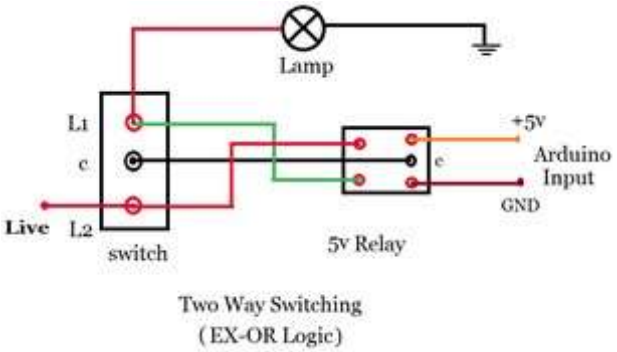
A two-way switch, also known as a double-pole switch or two-way light switch, is a type of electrical switch commonly used in residential and commercial settings. It allows control of a single electrical load, such as a light fixture or a set of lights, from two different locations.



1.8 Two way switch

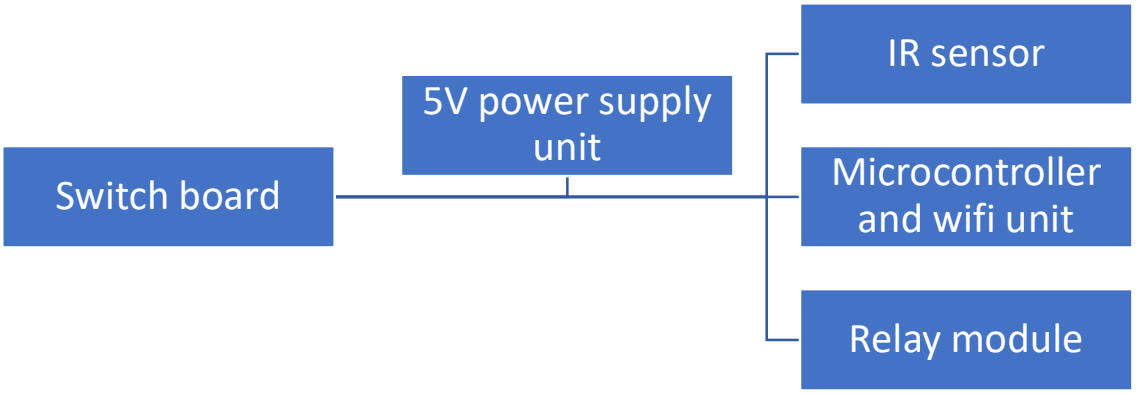
To Connect the Two-Way Switch to the Relay Module:

Start by identifying the common terminal of the two-way switch. This terminal should be connected to the power source or electrical load. Connect one terminal of the common terminal to the "Normally Open" (NO) or "Normally Closed" (NC) terminal of the relay module, depending on the desired functionality. The other terminal of the common terminal should be connected to the common ground of the power supply and the relay module.



1.10 Connection of Two-Way Switch to the Relay Module

CHAPTER 3: BLOCK DIAGRAM



2.0 Block diagram of ac load control system

Description of Block Diagram Components:

1. Switch Board:

The switch board serves as a physical interface for manual control of the AC loads. It consists of switches corresponding to each load switch, allowing users to manually turn them on or off as needed. The switch board provides a direct and immediate control method without relying on wireless or remote access.

2. 5V Power Supply Unit:

The 5V power supply unit is responsible for providing a stable and regulated power source to all the components of the system. It converts the available input voltage into a consistent 5V output, ensuring proper functioning of the microcontroller, WiFi unit, IR sensor, and relay module.

3. Microcontroller:

The microcontroller acts as the central processing unit of the system. It receives input signals from various sources, such as the WiFi unit, IR sensor, and manual switch board, and processes them to control the AC load switches accordingly. The microcontroller contains a built-in program that dictates the system's behavior and coordinates the communication between different components.

4. WiFi Unit:

The WiFi unit enables wireless communication and control of the AC load switches using a smartphone or any other WiFi-enabled device. It connects to the local WiFi network and receives commands or instructions from the user's device. The WiFi unit then transmits these commands to the microcontroller, allowing remote control of the AC loads from anywhere within the network range.

5. IR Sensor:

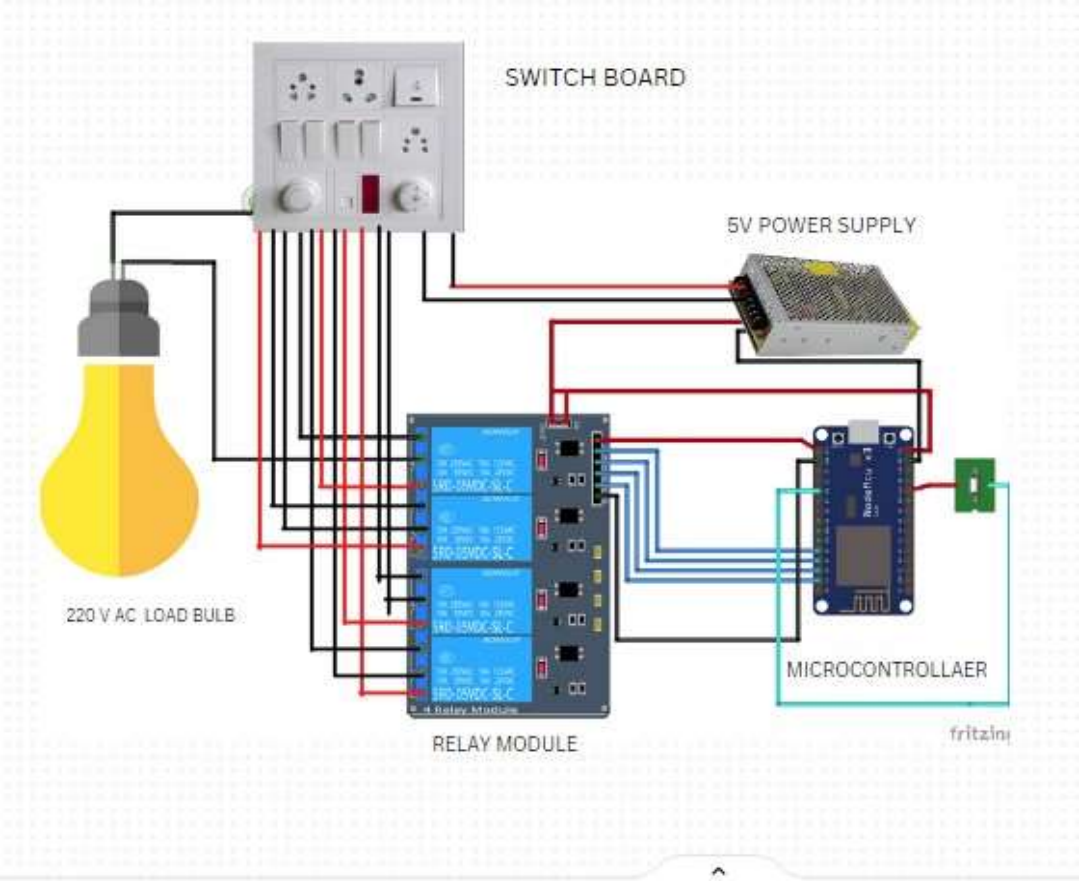
The IR sensor is responsible for receiving infrared signals from the IR remote control. When a button on the remote control is pressed, it emits an infrared signal that is detected by the IR sensor. The sensor translates this signal into electrical impulses and sends them to the microcontroller. Based on the received signal, the microcontroller executes the corresponding action, such as turning on or off the AC load switches.

6. Relay Module:

The relay module serves as an interface between the microcontroller and the AC load switches. It consists of multiple relays, each connected to a separate load switch. The microcontroller controls the relays, which, in turn, open or close the circuit of the AC loads. By controlling the relays, the microcontroller effectively controls the power supply to the AC loads, allowing for remote or manual switching of the devices.

3.1 Circuit Diagram Overview

2.1 Circuit diagram



3.2 Detailed Circuit Connections

1.1 Connection of power supply :

Firstly connect power supply ac input to the switch board ac input terminals and then 5v dc of power supply output to the input of microcontroller

Power supply +VE → VCC (microcontroller and IR sensor and relay module)

Power supply -VE → GND (microcontroller and IR sensor and relay module)

1.2 Connection of relay module and microcontroller

Microcontroller pins	Relay module pins
D2	IN1
D3	IN2
D4	IN3
D5	IN4

1.3 Connection of IR sensor and microcontroller

IR sensor pins	Microcontroller pins
OUT	D7
GND	GND

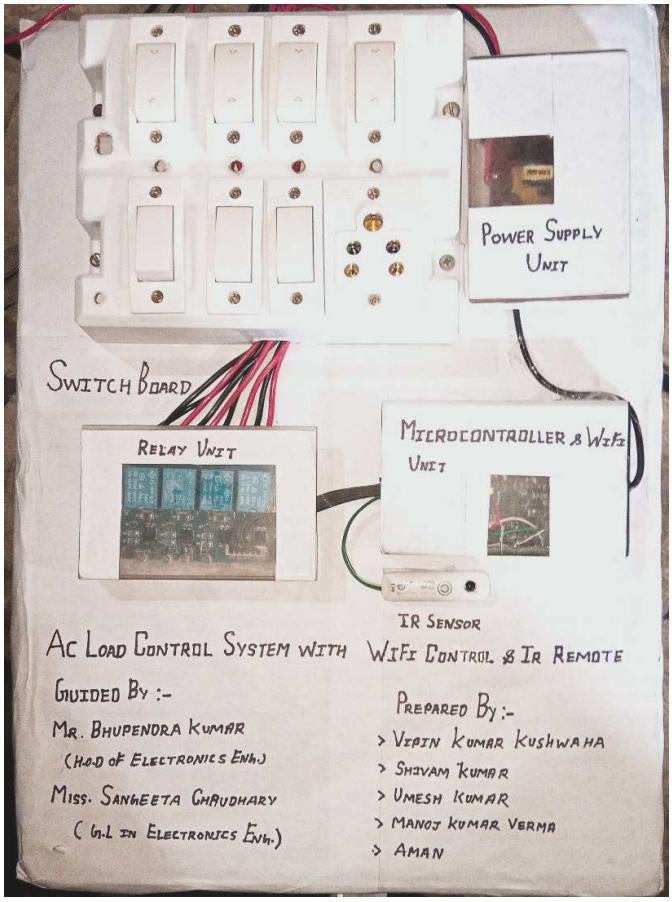
1.4 Connections of Switchboard and relay module

Connections of Switchboard and relay module are shown in the fig no. 1.9

1.5 Microcontroller programming :

To program microcontroller all steps are given in the chapter 4 according to that simply copy the given code and paste it in Arduino IDE and select board and upload the code

1.6 Photographic view of final project :



2.2 Photographic view of final project

1.6 CODE :

```
#include<ESP8266WiFi.h>

#include <EEPROM.h>

#include <IRremoteESP8266.h>

#include <IRrecv.h>

#include <IRutils.h>

WiFiClient client;

WiFiServer server(80);
```

```
int state1,state2,state3,state4;
#define EEPROM_SIZE 5
#define relay_1 D1
#define relay_2 D2
#define relay_3 D3
#define relay_4 D6
#define LED D7
// Define the IR receiver pin
const int IR_PIN = D5;
// Create an instance of the IR receiver
IRrecv irrecv(IR_PIN);
// Create a variable to store the IR code received
decode_results results;
void setup()
{
  // put your setup code here, to run once:
  EEPROM.begin(EEPROM_SIZE);
  Serial.begin(9600);
    // Start the IR receiver
  irrecv.enableIRIn();
  WiFi.softAP("ELectronics LAB", "******");
  Serial.println();
  Serial.println("NodeMCU Started!");
  Serial.println(WiFi.softAPIP());
  server.begin();
  pinMode(relay_1, OUTPUT);
  pinMode(relay_2, OUTPUT);
  pinMode(relay_3, OUTPUT);
```

```
  pinMode(relay_4, OUTPUT);
  pinMode(LED_BUILTIN,OUTPUT);
  pinMode(LED,OUTPUT);
    state1 = EEPROM.read(1);
    state2 = EEPROM.read(2);
    state3 = EEPROM.read(3);
    state4 = EEPROM.read(4);
    digitalWrite(relay_1, state1);
    digitalWrite(relay_2, state2);
    digitalWrite(relay_3, state3);
    digitalWrite(relay_4, state4);
    digitalWrite(LED, HIGH);
    digitalWrite(LED_BUILTIN, HIGH);
}
void loop()
{
  // put your main code here, to run repeatedly:
    // Check if an IR code has been received
  if (irrecv.decode(&results)) {
    // Print the received IR code to the serial monitor
    Serial.println(results.value,DEC);
    // Control the relay based on the received IR code
    switch (results.value) {
      case 33441975:  // button 1
        digitalWrite(relay_1, HIGH);
        digitalWrite(relay_2, HIGH);
        digitalWrite(relay_4, HIGH);
        digitalWrite(relay_3, HIGH);
```

```
  EEPROM.write(1, digitalRead(relay_1));
  EEPROM.write(2, digitalRead(relay_2));
  EEPROM.write(3, digitalRead(relay_3));
  EEPROM.write(4, digitalRead(relay_4));
  EEPROM.commit();
  break;
    case 33454215:  // button 1
  digitalWrite(relay_1, LOW);
  digitalWrite(relay_2, LOW);
  digitalWrite(relay_4, LOW);
  digitalWrite(relay_3, LOW);
  EEPROM.write(1, digitalRead(relay_1));
  EEPROM.write(2, digitalRead(relay_2));
  EEPROM.write(3, digitalRead(relay_3));
  EEPROM.write(4, digitalRead(relay_4));
  EEPROM.commit();
  break;
 case 33444015:  // button 1
  digitalWrite(relay_1, !digitalRead(relay_1));
  EEPROM.write(1, digitalRead(relay_1));
  EEPROM.commit();
  break;
 case 33478695:  // button 2
  digitalWrite(relay_2, !digitalRead(relay_2));
  EEPROM.write(2, digitalRead(relay_2));
  EEPROM.commit();
  break;
 case 33486855:  // button 3
```

```
      digitalWrite(relay_3, !digitalRead(relay_3));
      EEPROM.write(3, digitalRead(relay_3));
      EEPROM.commit();
      break;
    case 33435855:  // button 4
      digitalWrite(relay_4, !digitalRead(relay_4));
      EEPROM.write(4, digitalRead(relay_4));
      EEPROM.commit();
      break;
    default:
      break;
  }
  irrecv.resume();
}
 client = server.available();  //Gets a client that is connected to the server and has data available for reading.
  if(client == 1)
 {
  String request =  client.readStringUntil('\n');
  Serial.println(request);
  request.trim();
    if(request == "GET /a HTTP/1.1")  // Requested to the server 192.168.4.1
  {
   digitalWrite(relay_1, HIGH);
    EEPROM.write(1, HIGH );
    EEPROM.commit();
  }
  if(request == "GET /A HTTP/1.1")
  {
```

```
 digitalWrite(relay_1, LOW);
 EEPROM.write(1, LOW);
   EEPROM.commit();
 }
if(request == "GET /b HTTP/1.1")
{
 digitalWrite(relay_2,HIGH);
  EEPROM.write(2, HIGH);
  EEPROM.commit();
 }
if(request == "GET /B HTTP/1.1")
 {
 digitalWrite(relay_2,LOW);
  EEPROM.write(2, LOW);
  EEPROM.commit();
   }
if(request == "GET /c HTTP/1.1")
 {
  EEPROM.write(3, HIGH);
  EEPROM.commit();
}
if(request == "GET /C HTTP/1.1")
 {
  digitalWrite(relay_3,LOW);
  EEPROM.write(3, LOW);
  EEPROM.commit();
   }
```

```
   if(request == "GET /d HTTP/1.1")
       EEPROM.write(4, HIGH);
 EEPROM.commit();
       }
    if(request == "GET /D HTTP/1.1")
    {
      digitalWrite(relay_4,LOW);
EEPROM.write(4, LOW);
      EEPROM.commit();
       }
   }

}
```

CHAPTER 4

SOFTWARE:

Connecting hardware according to the circuit diagram doesn't work really. You need to program the components accordingly so as to work according to the algorithm you have designed in order to make the project work properly as required. Following is the description of software used.

4.1 Arduino IDE

The Arduino IDE is a cross-platform application written in Java, and is derived from the IDE for the Processing programming language and the Wiring project. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and is also capable of compiling and uploading programs to the board with a single click. There is typically no need to edit makefiles or run programs on a command-line interface. Although building on command-line is possible if required with some third-party tools such as Ino. The Arduino IDE comes with a C/C++ library called "Wiring" (from the project of the same name), which makes many common input/output operations much easier.

4.1.1 Key Features

- Cross-platoform program
- Open-Source
- Different instructions for different OS
- Example Programs

4.1.2 Blink Program

Once we have downloaded/unzipped the arduino IDE, we can Plug the Arduino to our PC via USB cable. We are actually ready to "burn" our first program on the arduino board. To select "blink led", the physical translation of the well-known programming "hello world", select

File>Sketchbook>Arduino-0017>Examples>Digital>Blink

Once we have our sketch we'll see something very close to the screenshot on the following page.

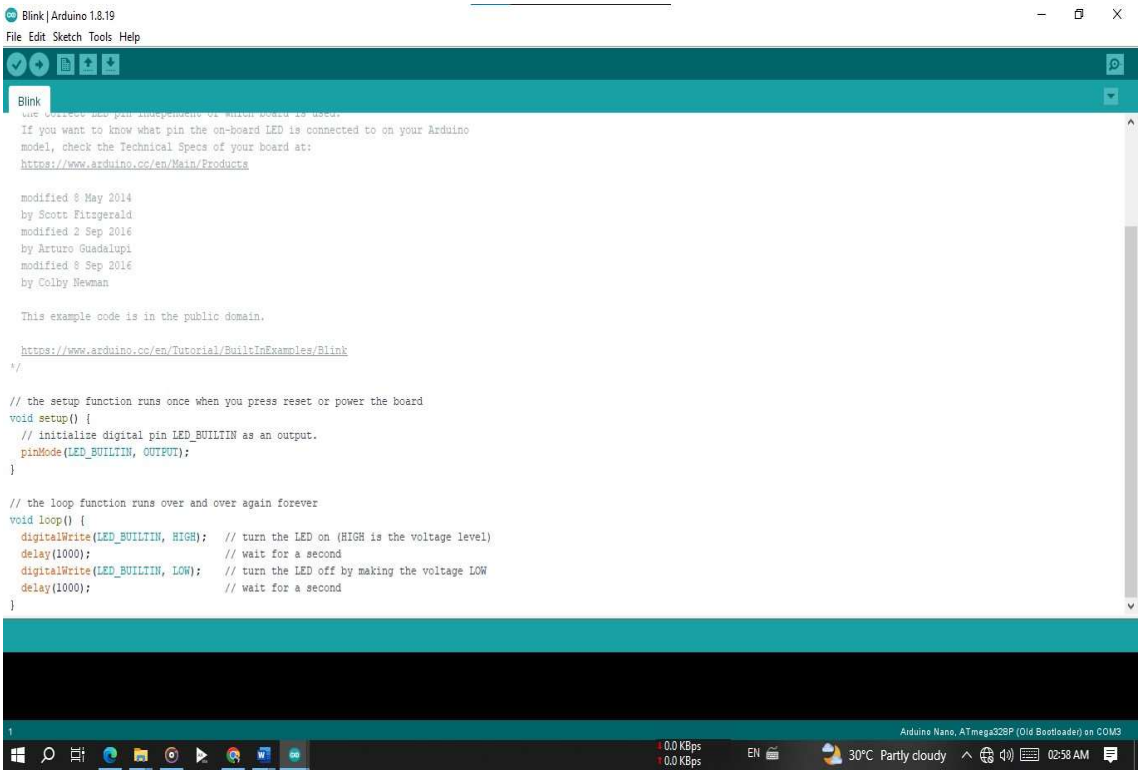


Fig 2.2 A screenshot of the Arduino IDE showing the "Blink" program, a simple beginner program

4.1.3 Functions Library

[10] Arduino programs are written in C/C++, although users only need define two functions to make a runnable program:

- setup () – a function run once at the start of a program that can initialize settings
- loop () – a function called repeatedly until the board powers off

A typical first program for a microcontroller simply blinks an LED on and off. In the Arduino environment, the user might write a program like this:

```
#define LED_PIN 13

void setup () {
   pin Mode (LED_PIN, OUTPUT);     // enable pin 13 for digital output
}

void loop () {
   digital Write (LED_PIN, HIGH);  // turn on the LED
```

```
   delay (1000);                // wait one second (1000 milliseconds)
   digital Write (LED_PIN, LOW);   // turn off the LED
   delay (1000);                // wait one second
}
```

It is a feature of most Arduino boards that they have an LED and load resistor connected between pin 13 and ground, a convenient feature for many simple tests. The above code would not be seen by a standard C++ compiler as a valid program, so when the user clicks the "Upload to I/O board" button in the IDE, a copy of the code is written to a temporary file with an extra include header at the top and a very simple main() function at the bottom, to make it a valid C++ program. The Arduino IDE uses the GNU toolchain and AVR Libc to compile programs, and uses avrdude to upload programs to the board. As the Arduino platform uses Atmel microcontrollers Atmel's development environment, AVR Studio or the newer Atmel Studio, may also be used to develop software for the Arduino.

- setup ()

The setup () function is called when a sketch starts. Use it to initialize variables, pin modes, start using libraries, etc. The setup function will only run once, after each powerup or reset of the Arduino board.

Example

```
int button Pin = 3;

void setup ()
{
  Serial. Begin (9600);
  pin Mode (button Pin, INPUT);
}

void loop ()
{
  // ...
}
```

- **loop ()**

After creating a setup () function, which initializes and sets the initial values, the loop() function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to actively control the Arduino board.

Example

```
const int button Pin = 3;
```

```
// setup initializes serial and the button pin
void setup ()
{
  Serial.begin(9600);
  pinMode (button Pin, INPUT);
}

// loop checks the button pin each time,
// and will send serial if it is pressed
void loop ()
{
  if (digitalRead(buttonPin) == HIGH)
    Serial.write('H');
  else
    Serial.write('L');

  delay(1000);
}
```

- Define

#define is a useful C component that allows the programmer to give a name to a constant value before the program is compiled. Defined constants in arduino don't take up any program memory space on the chip. The compiler will replace references to these constants with the defined value at compile time. This can have some unwanted side effects though, if for example, a constant name that had been #defined is included in some other constant or variable name. In that case the text would be replaced by the #defined number (or text).

In general, the const keyword is preferred for defining constants and should be used instead of #define.

Arduino defines have the same syntax as C defines:

Syntax

#define constantName value

Note that the # is necessary.

Example

```
#define ledPin 3
// The compiler will replace any mention of ledPin with the value 3 at compile time.
```

- #include

#include is used to include outside libraries in your sketch. This gives the programmer access to a large group of standard C libraries (groups of pre-made functions), and also libraries written especially for Arduino.

The main reference page for AVR C libraries (AVR is a reference to the Atmel chips on which the Arduino is based) is here.

Note that **#include**, similar to **#define**, has no semicolon terminator, and the compiler will yield cryptic error messages if you add one.

Example

This example includes a library that is used to put data into the program space *flash* instead of *ram*. This saves the ram space for dynamic memory needs and makes large lookup tables more practical.

```
#include <avr/pgmspace.h>
prog_uint16_t myConstants [] PROGMEM = {0, 21140, 702, 9128, 0, 25764, 8456,
0,0,0,0,0,0,0,0,29810,8968,29762,29762,4500};
```

- pin Mode ()

Description

Configures the specified pin to behave either as an input or an output. See the description of digital pins pains for details on the functionality of the pins.

As of Arduino 1.0.1, it is possible to enable the internal pullup resistors with the mode INPUT_PULLUP. Additionally, the INPUT mode explicitly disables the internal pullups.

Syntax

Pin Mode (pin, mode)

Parameters

pin: the number of the pin whose mode you wish to set

mode: INPUT, OUTPUT, or INPUT_PULLUP. (See the digital pins page for a more complete description of the functionality.)

Returns

None

Example

```
int led Pin = 13;              // LED connected to digital pin 13
void setup()
{
  pin Mode(led Pin, OUTPUT);      // sets the digital pin as output
}
void loop()
{
  digital Write(led Pin, HIGH);   // sets the LED on
  delay(1000);                    // waits for a second
  digital Write(led Pin, LOW);    // sets the LED off
  delay(1000);                    // waits for a second
}
```

- digital Read ()

Description

Reads the value from a specified digital pin, either HIGH or LOW.

Syntax

digital Read(pin)

Parameters

pin: the number of the digital pin you want to read (*int*)

Returns

HIGH or LOW

Example

```
int ledPin = 13; // LED connected to digital pin 13
int inPin = 7; // pushbutton connected to digital pin 7
int val = 0;   // variable to store the read value

void setup()
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin 13 as output
  pinMode(inPin, INPUT);      // sets the digital pin 7 as input
}
void loop()
{
  val = digitalRead(inPin);   // read the input pin
  digitalWrite(ledPin, val);    // sets the LED to the button's value
}
```

Sets pin 13 to the same value as the pin 7, which is an input.

digital Write ()

Description

Write a HIGH or a LOW value to a digital pin.

If the pin has been configured as an OUTPUT with pinMode (), its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW. If the pin is configured as an INPUT, writing a HIGH value with digitalWrite() will enable an internal 20K pullup resistor (see the tutorial on digital pins). Writing LOW will disable the pullup. The pullup resistor is enough to light an LED dimly, so if LEDs appear to work, but very dimly, this is a likely cause. The remedy is to set the pin to an output with the pinMode() function.

NOTE: Digital pin 13 is harder to use as a digital input than the other digital pins because it has an LED and resistor attached to it that's soldered to the board on most boards. If you enable its internal 20k pull-up resistor, it will hang at around 1.7 V instead of the expected 5V because the onboard LED and series resistor pull the voltage level down, meaning it always returns LOW. If you must use pin 13 as a digital input, use an external pull down resistor.

Syntax

digitalWrite (pin, value)

Parameters

pin: the pin number

value: HIGH or LOW

Returns

none

Example

```
int ledPin = 13;                // LED connected to digital pin 13
void setup()
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin as output
}
void loop()
{
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite (ledPin, LOW);    // sets the LED off
  delay(1000);                  // waits for a second
}
```

Sets pin 13 to HIGH, makes a one-second-long delay, and sets the pin back to LOW.

- Serial

Used for communication between the Arduino board and a computer or other devices. All Arduino boards have at least one serial port (also known as a UART or USART): **Serial**. It communicates on digital pins 0 (RX) and 1 (TX) as well as with the computer via USB. Thus, if you use these functions, you cannot also use pins 0 and 1 for digital input or output. You can use the Arduino environment's built-in serial monitor to communicate with an Arduino board. Click the serial monitor button in the toolbar and select the same baud rate used in the call to begin().

The Arduino Mega has three additional serial ports: **Serial1** on pins 19 (RX) and 18 (TX), **Serial2** on pins 17 (RX) and 16 (TX), **Serial3** on pins 15 (RX) and 14 (TX). To use these pins to communicate with your personal computer, you will need an additional USB-to-serial adaptor, as they are not connected to the Mega's USB-to-serial adaptor. To use them to communicate with an external TTL serial device, connect the TX pin to your device's RX pin, the RX to your device's TX pin, and the ground of your Mega to your device's ground. (Don't connect these pins directly to an RS232 serial port; they operate at +/- 12V and can damage your Arduino board.)

The Arduino Due has three additional 3.3V TTL serial ports: **Serial1** on pins 19 (RX) and 18 (TX); **Serial2** on pins 17 (RX) and 16 (TX), **Serial3** on pins 15 (RX) and 14 (TX). Pins 0 and 1 are also connected to the corresponding pins of the ATmega16U2 USB-to-TTL Serial chip, which is connected to the USB debug port. Additionally, there is a native USB-serial port on the SAM3X chip, *SerialUSB*'. The Arduino Leonardo board uses **Serial1** to communicate via TTL (5V) serial on pins 0 (RX) and 1 (TX). **Serial** is reserved for USB CDC communication. For more information, refer to the Leonardo getting started page and hardware page.

- analogRead()

Description

Reads the value from the specified analog pin. The Arduino board contains a 6 channel (8 channels on the Mini and Nano, 16 on the Mega), 10-bit analog to digital converter. This means that it will map input voltages between 0 and 5 volts into integer values between 0 and 1023. This yields a resolution between readings of: 5 volts / 1024 units or, .0049 volts (4.9 mV) per unit. The input range and resolution can be changed using analogReference().

It takes about 100 microseconds (0.0001 s) to read an analog input, so the maximum reading rate is about 10,000 times a second.

Syntax

analogRead(pin)

Parameters

pin: the number of the analog input pin to read from (0 to 5 on most boards, 0 to 7 on the Mini and Nano, 0 to 15 on the Mega)

Returns

int (0 to 1023)

Note

If the analog input pin is not connected to anything, the value returned by analogRead() will fluctuate based on a number of factors (e.g. the values of the other analog inputs, how close your hand is to the board, etc.).

Example

```
int analogPin = 3;     // potentiometer wiper (middle terminal) connected to analog pin 3
                       // outside leads to ground and +5V
int val = 0;           // variable to store the value read
void setup()
{
  Serial.begin(9600);          //  setup serial
}
void loop()
{
  val = analogRead(analogPin);    // read the input pin
  Serial.println(val);             // debug value
}
```

Boolean Operators

These can be used inside the condition of an if statement.

- && (logical and)

True only if both operands are true, e.g.

```
if (digitalRead(2) == HIGH  && digitalRead(3) == HIGH) { // read two switches.
}
```

is true only if both inputs are high.

- if (conditional) and ==, !=, <, > (comparison operators)

if, which is used in conjunction with a comparison operator, tests whether a certain condition has been reached, such as an input being above a certain number. The format for an if test is:

```
if (someVariable > 50)
{
  // do something here
}
```

The program tests to see if someVariable is greater than 50. If it is, the program takes a particular action. Put another way, if the statement in parentheses is true, the statements inside the brackets are run. If not, the program skips over the code.

The brackets may be omitted after an *if* statement. If this is done, the next line (defined by the semicolon) becomes the only conditional statement.

```
if (x > 120) digitalWrite(LEDpin, HIGH);

if (x > 120)
digitalWrite(LEDpin, HIGH);
if (x > 120){ digitalWrite(LEDpin, HIGH); }
if (x > 120){
  digitalWrite(LEDpin1, HIGH);
  digitalWrite(LEDpin2, HIGH);
}       // all are correct
```

CHAPTER 5

LIST OF REFRENCES

1. *"ESP8266 Overview". Espressif Systems. Retrieved 2017-10-02. https://en.wikipedia.org/wiki/ESP8266*
2. *^ Brian Benchoff (August 26, 2014). "New Chip Alert: The ESP8266 WiFi Module (It's $5)". Hackaday. Retrieved 2015-06-24.*
3. Conection of two way switch with relay image https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.instructables.com%2FTwowaySwitching%2F&psig=AOvVaw38IvqJKFg8UiQkzcvju8CN&ust=1684786508730000&source=images&cd=vfe&ved=0CBIQjhxqFwoTCMCtg_ych_8CFQAAAAAdAAAAABAE
4. Esp8266 image https://www.pngwing.com/en/search?q=esp8266
5. Relay module image https://www.pngwing.com/en/free-png-tgagt
6. Swich board image https://www.pngegg.com/en/png-ewfqh
7. IR sensor image https://www.pngwing.com/en/search?q=IRsensor6