# Project-Based Learning Report

on

## Using ALEXA to Switch ON/OFF Water Sprinkler Pump with ESP8266.

Submitted in the partial fulfillment of the requirements.

For the Project-based learning in

**INDUSTRIAL INTERNET OF THINGS &**

**MACHINE LEARNING**

# INDEX

**Serial No. Title Page No.**

|  |  |  |
| --- | --- | --- |
| **1** | Problem Statement | 1 |
| **2** | Project Description | 1-4 |
| **3** | Software Used | 5 |
| **4** | Circuit Diagram | 6 |
| **5** | Program Code | 7-8 |
| **6** | Result & Output | 9-11 |
| **7** | Project Conclusion & Outcome | 12 |

## PROBLEM STATEMENT

To Switch ON/OFF water pump which is used to sprinkle water on the land using ALEXA.

# Project Description

The ESP8266 is a low-cost Wi-Fi chip. It can be used as a standalone device, or as a UART to Wi-Fi adaptor to allow other microcontrollers to connect to a Wi-Fi network.

* **Low-cost**: you can get ESP8266 boards starting at $3 (or less) depending on the model.
* **Low power**: the ESP8266 consumes very little power when compared with other microcontrollers and can even go into deep sleep mode to consume less power.
* **Wi-Fi**: the ESP8266 can generate its own Wi-Fi network (access point) or connect to other Wi-Fi networks (station) to get access to the internet. This means the ESP8266 can access online services to make HTTP requests or save data to the cloud, for example. It can also act as a web server so that you can access it using a web browser and be able to control and monitor your boards remotely.
* **Compatible with the Arduino “programming language”**: those that are already familiar with programming the Arduino board, we’re happy to know that they can program the ESP8266 in the Arduino style.
* **Compatible with Micro Python**: you can program the ESP8266 with Micro Python firmware, which is a re-implementation of Python 3 targeted for microcontrollers and embedded systems.

ESP8266 Pinout in Arduino IDE

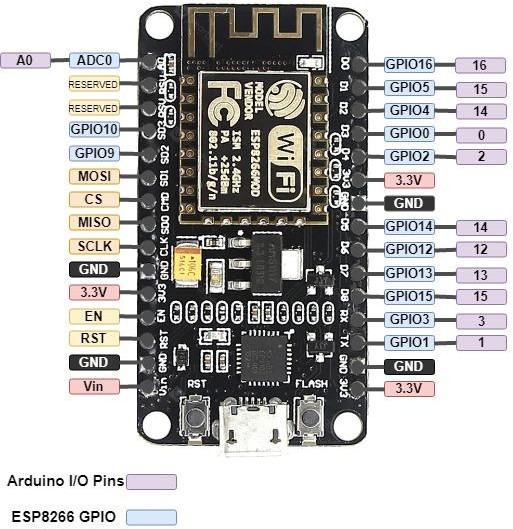


Fig. 1. A

Consider the following table and the NodeMCU picture to use the right number of the pin in your sketch otherwise you will face the problems in using the correct GPIOs.

|  |  |  |  |
| --- | --- | --- | --- |
| Arduino IDE Pins Index | ESP8266 GPIO | | |
| A0 | **A**DC 0 | | |
| 0 | GPIO 0 | | |
| 1 |  | GPIO 1 |  |
| 2 | GPIO 2 | | |
| 3 |  | GPIO 3 |  |
| 4 | GPIO 4 | | |
| 5 |  | GPIO 5 |  |
| 12 | GPIO 12 | | |
| 13 |  | GPIO 13 |  |
| 14 | GPIO 14 | | |
| 15 |  | GPIO 15 |  |
| 16 | GPIO 16 | | |

**Components Used**

* + NodeMCU ESP8266
  + Relay Module
  + Water Pump
  + Breadboard
  + Jumper Wire
  + Micro USB cable

**Relay Module**

The relay module with a single channel board is used to manage high voltage, current loads like [solenoid](https://www.elprocus.com/different-types-of-solenoid-working-applications/) valves, motor, AC load & lamps. This module is mainly designed to interface through different microcontrollers like PIC, Arduino, etc.

## 5V Relay Module Pin Configuration

The pin configuration of the 5V relay module is shown below. This module includes 6-pins where each pin and its functionality are discussed below.



Fig.1. B. Relay Module Pin Diagram

**Normally Open (NO):** This pin is normally open unless we provide a signal to the relay modules signal pin. So, the common contact pin smashes its link through the NC pin to make a connection through the NO pin

**Common Contact:** This pin is used to connect through the load that we desire to switch by using the module.

**Normally Closed (NC):** This NC pin is connected through the COM pin to form a closed circuit. However, this NC connection will break once the relay is switched through providing an active high/low signal toward the signal pin from a [microcontroller.](https://www.elprocus.com/atmega16-next-generation-micro-controller/)

**Signal Pin:** The signal pin is mainly used for controlling the relay. This pin works in two cases like active low otherwise active high. So, in active low case, the relay activates once we provide an active low signal toward the signal pin, whereas, in an active high case, the relay will trigger once we provide a high signal toward the signal pin.

However, these modules generally work on an active high signal which will strengthen the relay coil to make contact with the common terminal with the normally open terminal.

**5V VCC:** This pin needs 5V DC to work. So 5V DC power supply is provided to this pin.

**Ground:** This pin connects the GND terminal of the power supply.

## Connection Diagram:

The ESP8266 NodeMCU is a development board based on the ESP8266 microcontroller. It is designed to simplify the process of building projects with the ESP8266 microcontroller by supplying an easy-to-use platform that includes all the necessary hardware components and a USB interface for programming and debugging.

The NodeMCU board includes a built-in Wi-Fi module that allows it to connect to a local Wi- Fi network and communicate with other devices over the network. It also includes a USB-to- serial converter that allows the board to be programmed using the Arduino IDE or other programming languages.

The data pins of the relay have been connected to the D1 pin of the NodeMCU. ‘Vin’ of the relay is connected to 5v Power Supply through the ESP8266 and the Ground Pin of the relay module is connected to the ground pin of the NodeMCU. The Middle Output Pin of the Relay Module is shorted to each other and gone through 1 node of the Plug. The First Output node of each relay has gone through water pump and other node is connected to each other. Other node of the water pump is connected to each other and gone through another node of the Plug.

## Working of Alexa Smart Water Sprinkler system using ESP8266 & Relay Module:

To connect to Alexa App web server on the ESP8266, the first step is to establish a Wi- Fi connection to the local network. Once the connection is established, a web server is created using the ESP8266WiFi.h library. The server is set up to listen on a specific port, typically port 80, which is the default HTTP port.

When a client sends an automation request to the server, the ESP8266 microcontroller receives the request and processes it. The request is typically in the form of a GET or POST request and includes a URL that specifies the resource to be retrieved or modified.

To control the LED, the ESP8266 microcontroller needs to read the incoming request and determine the desired action. This is typically done by parsing the URL for a specific parameter or value. In the case of Water Sprinkler control, the URL might include a parameter named "Water Pump" with a value of "ON" or "OFF".

Once the desired action is determined, the ESP8266 microcontroller can then control the relay by setting the appropriate pin high or low. For example, if the desired action is to turn the relay on, the microcontroller can set the pin connected to the relay to a high logic level, which will turn the relay on.

To control these devices, the ESP8266 microcontroller needs to be programmed to read and write to their respective input and output pins. This can be done using the Arduino programming language, which is commonly used with the ESP8266 microcontroller.

Finally, the ESP8266 microcontroller sends a response back to the client with the updated status of the relay. This response is typically in the form of an HTML page that includes a status message and a link or button to return to the main control page.

## Software Used

### Arduino IDE and its overview:

The Arduino IDE is an open-source software, which is used to write and upload code to the Arduino boards. The IDE application is suitable for different operating systems such as Windows, Mac OS X, and Linux. It supports the programming languages C and C++. Here, IDE stands for Integrated Development Environment.



The program or code written in the Arduino IDE is often called as sketching. We need to connect the Genuino and Arduino board with the IDE to upload the sketch written in the Arduino IDE software. The sketch is saved with the extension '.ino'.

It is an official Arduino software, making code compilation too easy that even a common person with no prior technical knowledge can get their feet wet with the learning process. Each of them contains a microcontroller on the board that is actually programmed and accepts the information in the form of code. The main code, also known as a sketch, created on the IDE platform will ultimately generate a Hex File which is then transferred and uploaded in the controller on the board.

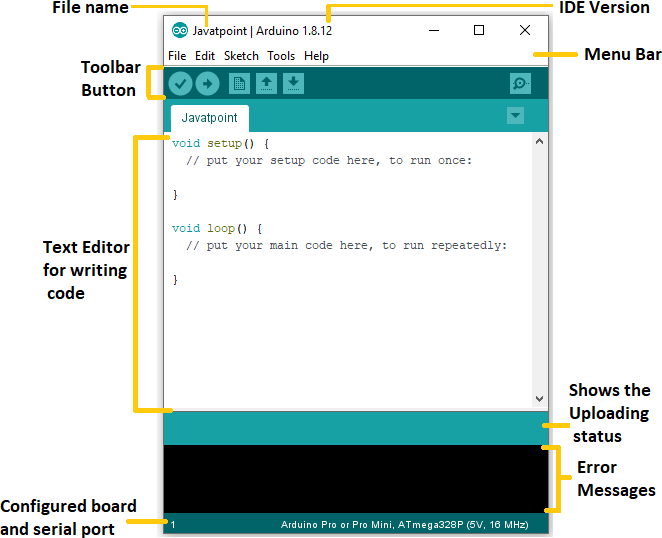


Fig.1.C

## Circuit Diagram

**Program Code**

#ifdef ARDUINO\_ARCH\_ESP32

#include <WiFi.h> #else

#include <ESP8266WiFi.h> #endif

#include <Espalexa.h>

// define the GPIO connected with Relays #define RelayPin1 5 //D1

// prototypes

boolean connectWifi();

//callback functions

void firstLightChanged(uint8\_t brightness);

// WiFi Credentials

const char\* ssid = "Squad-204";

const char\* password = "airtel8051127884";

// device names

String Device\_1\_Name = "Water pump"; boolean wifiConnected = false; Espalexa espalexa;

void setup()

{

Serial.begin(115200);

pinMode(RelayPin1, OUTPUT);

// Initialise wifi connection wifiConnected = connectWifi();

if (wifiConnected)

{

espalexa.addDevice(Device\_1\_Name, firstLightChanged); //simplest definition, default state off

espalexa.begin();

}

else

{

while (1)

{

Serial.println("Cannot connect to WiFi. Please check data and reset the ESP.");

delay(2500);

}

}

}

void loop()

{

espalexa.loop(); delay(1);

}

//our callback functions

void firstLightChanged(uint8\_t brightness)

{

//Control the device if (brightness == 255)

{

digitalWrite(RelayPin1,LOW); Serial.println("Device1 ON");

}

else

{

digitalWrite(RelayPin1, HIGH); Serial.println("Device1 OFF");

}

}

// connect to wifi – returns true if successful or false if not boolean connectWifi()

{

boolean state = true; int i = 0;

WiFi.mode(WIFI\_STA); WiFi.begin(ssid, password); Serial.println(""); Serial.println("Connecting to WiFi");

// Wait for connection Serial.print("Connecting...");

while (WiFi.status() != WL\_CONNECTED) { delay(500);

Serial.print("."); if (i > 20) {

state = false; break;

} i++;

}

Serial.println(""); if (state) {

Serial.print("Connected to "); Serial.println(ssid); Serial.print("IP address: "); Serial.println(WiFi.localIP());

}

else {

Serial.println("Connection failed.");

}

return state;

}

## Screenshots of code Performed on Arduino

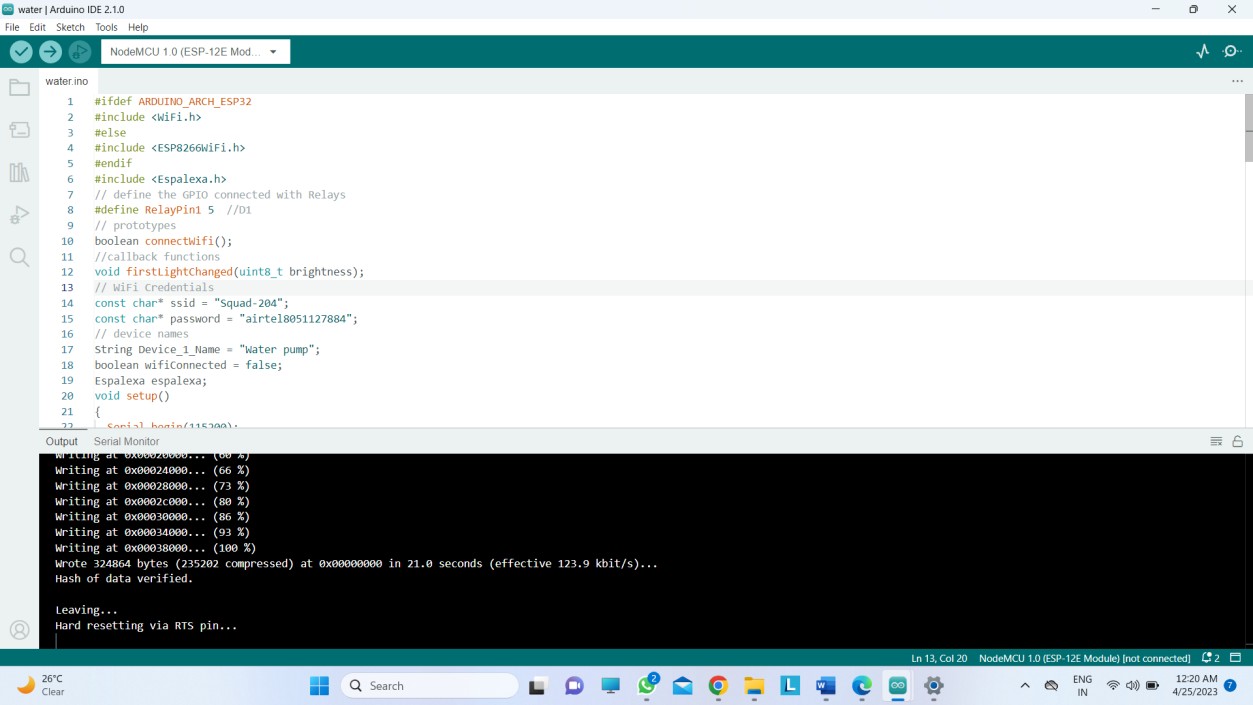


Fig.2

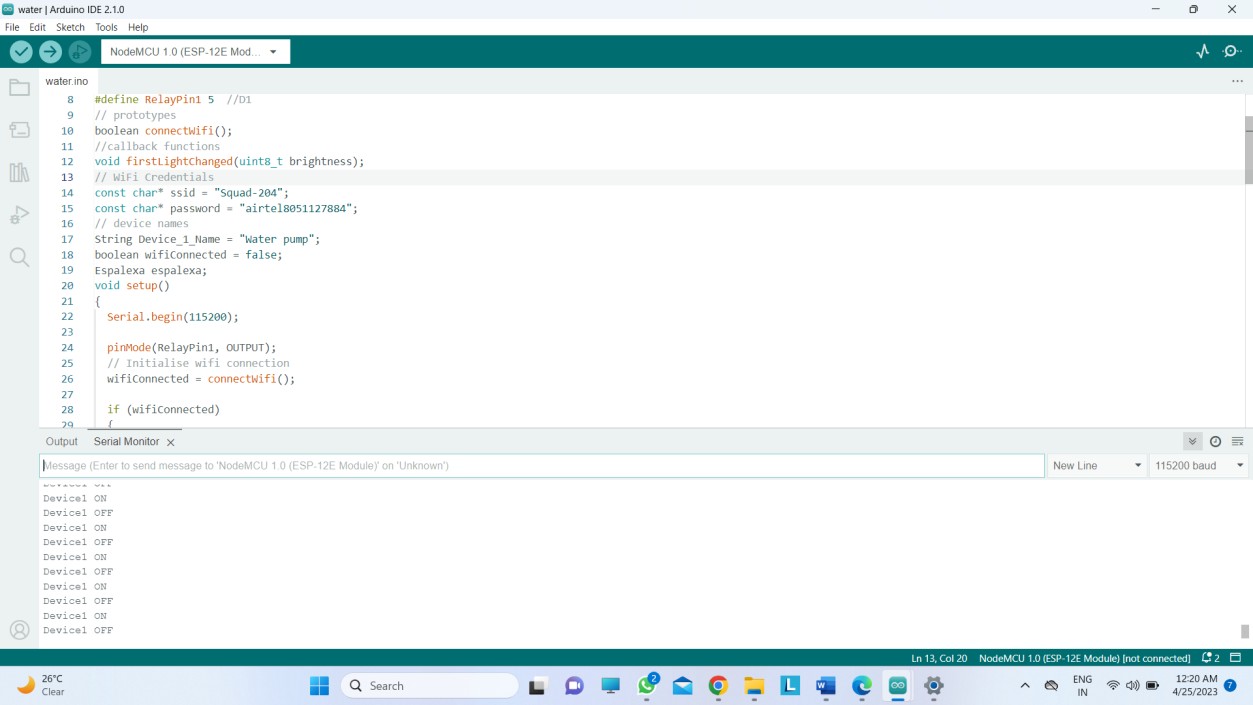


Fig.3

## Screenshots of Circuits

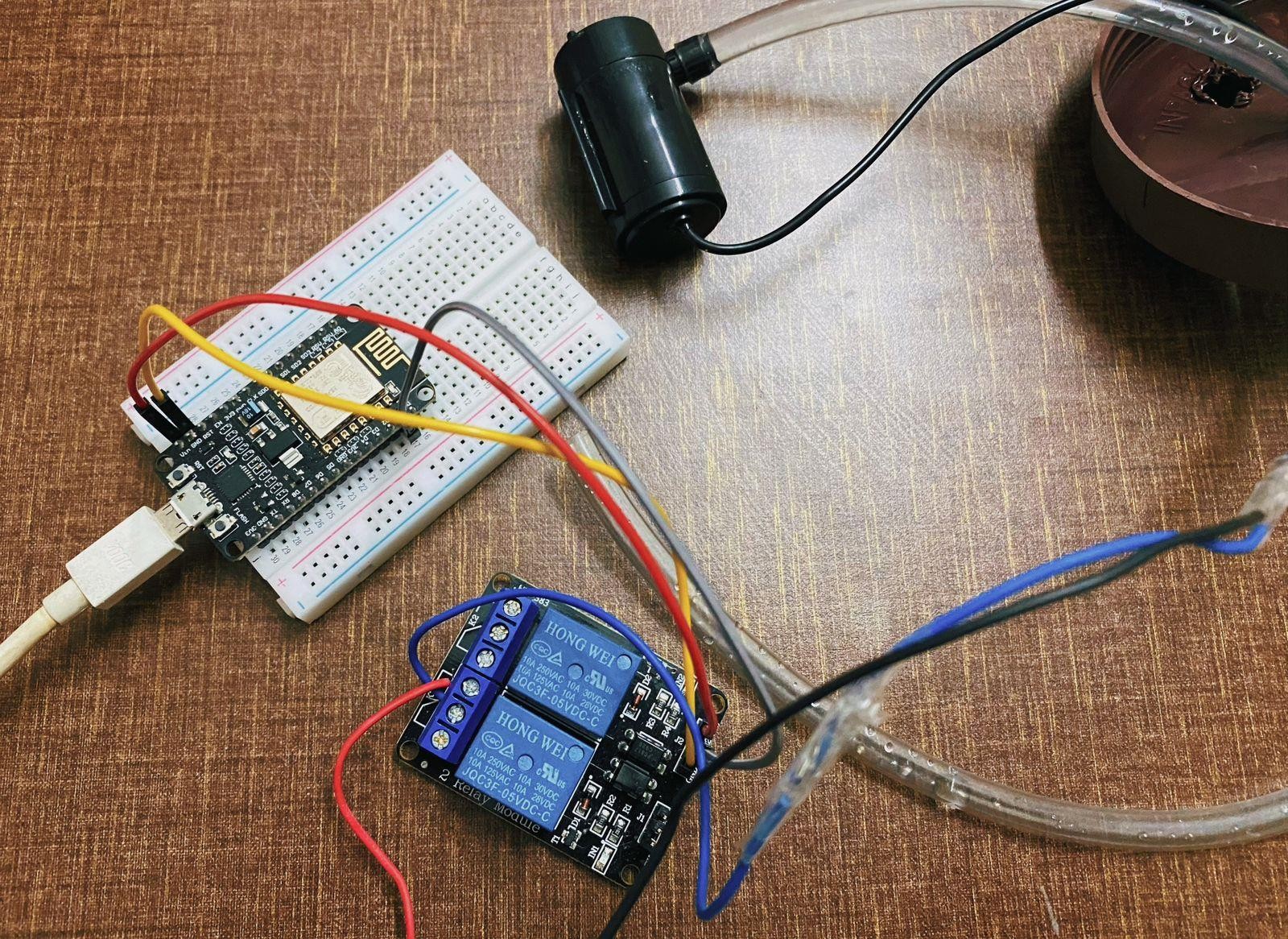


Fig.4



Fig.5



Fig.6

## Project Conclusion

The project is “**Using ALEXA to Switch ON/OFF Water Sprinkler pump with ESP8266**.” has helped us to understand the basics of ESP8266 & real world examples of IOT. We used Arduino IDE and NodeMCU ESP8266 with each step, and we explored and learned the various concepts of each of its pins. We learned how to use these concepts for solving real-world-based problems through Sensors. Overall, it was an enriching experience for all of us.

## Project Outcome

The PBL project is helped us to understand the real world application. The project maps to the course outcome 6.

CO6: Design an IOT application with ML and Arduino /Raspberry Pi.

In this project, we have used ESP8266.