IOT Home Automation

Six Months Industrial Training Report

On

Amplify Software Solutions Pvt Ltd

Submitted in partial fulfilment of the requirements for the award of degree

of

Bachelor of Technology

in

COMPUTER SCIENCE & ENGINEERING

TRANSFORMING THE ENGINEERING EDUCATION

Submitted By:

Abhiraj Sharma

18015004006

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING ECHELON INSTITUTE OF TECHNOLOGY, FARIDABAD

JAN 2022- JULY 2022

Amplify

Amplify your capabilities

AMPLIFY SOFTWARE SOLUTIONS PRIVATE LIMITED

Registered Address:	Corporate Address:	CIN : U72900DL2017PTC327572
A-34/A, First Floor,	419-424, 4th Floor, JMD Megapolis,	https://amplifysoftwaresolutions.com
Rama Park, Uttam Nagar,	Sector 48, Sohna Road, Gurgaon,	E-mail: info@amplifyss.com
New Delhi 110059, India.	Haryana 122018, India.	Contact No: 9599821144

Date: 05-Jul-2022

To Whomsoever It May Concern

Sub: Industrial training completion certificate.

This is to certify that **Mr. Abhiraj Sharma** doing B. Tech CSE from Echelon institute of technology having college Roll no. **18015004006**, have done the training on ERP Software: "Microsoft Dynamics Business Central" as part of his training cum project at Amplify Software Solution Pvt. Ltd.

His industrial training started from **01-Jan-2022** and completed on **30-Jun-2022** at our Gurgaon branch (419-424, 4th Floor, JMD Megapolis, Sector 48, Sohna Road, Gurgaon, Haryana – 122018, India). He is sincere, hardworking and his conduct during the project has been commendable.

Industrial training duration was **01-Jan-2022** to **30-Jun-2022**.

Authorised Signatory

SKMalhan

Shiv Kumar Malhan

Director

CERTIFICATE

TO WHOM IT MAY CONCERN

This is to certify that the dissertation entitled **"IOT Home Automation"** submitted by **Abhiraj Sharma,** Roll No.: **18015004006,** Department of Computer Science and Engineering, Echelon Institute of Technology Under YMCA UNIVERSITY for partial fulfillment of the requirements for the degree of Bachelor of Technology in Computer Science & Engineering is a bonafide record of the work and investigations carried out by him/her under my supervision and guidance.

Signature of H.O.D.	Signature of the Supervisor
MS. SHEFALI MADAN	**MS. SUMAN CHANDILA**
	AP, CSE Department

CANDIDATE DECLARATION

I hereby declare that the project report entitled **IOT Home Automation** submitted by me to Echelon Institute of Technology, Faridabad in partial fulfillment of the requirement for the award of the degree of B. Tech in Computer Science and Engineering. is a record of Bonafide project work carried out by me under the guidance of **Ms. Suman Chandila**, CSE 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 degree or diploma in this institute or any other institute or university.

Signature of Candidate

Abhiraj Sharma

18015004006

ACKNOWLEDGEMENTS

I take this opportunity to thank all those who have helped me in completing the project successfully.

I would like to express my gratitude to **Ms. Suman Chandila**, who as my guide/mentor provided me with every possible support and guidance throughout the development of project. This project would never have been completed without her encouragement and support.

I would also like to show my gratitude to **Ms. Shefali Madan**, Head of Department for providing me with well-trained Team members and giving me all the required resources and a healthy environment for carrying out my project work

.

We sincerely thank **Dean Academics & Principal of Echelon Institute of Technology**, Faridabad for providing us a platform to build this project.

I sincerely thank to **Amplify Software Solutions** for helping me during the industrial training.

Abhiraj Sharma

18015004006

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER – 1

COMPANY PROFILE

Chapter-1

Company Profile

1.1 Amplify Software Solutions

Amplify Software Solutions is a start-up., which basically works on adding customizations to Microsoft Dynamics Business Central ERP according to client's requirement. Started from 2017, Amplify has built and made custom modifications in Business Central for worldwide user. Our code is very well optimized and gives businesses industry standard ERP system. We are using most trending technologies in terms of development and marketing to increase our reach. We at Amplify., always prefer Team Management, Time Management, Low work life pressure and smoothness in work. We all work on Microsoft Azure and its only visible to our employees.

Our Apps: -

1. Microsoft Dynamics Business Central
2. Microsoft Azure
3. Docker
4. Teams and Outlook for team coordination

Amplify

Fig 1.1

At Amplify, we also give support and training to new users who will be using our extensions in future. We have a team of over 10+ developers who works for best experience out there.

Business Central is a business management solution for small and mid-sized organizations that automates and streamlines business processes and helps you manage your business. Highly adaptable and rich with features, Business Central enables companies to manage their business, including finance, manufacturing, sales, shipping, project management, services, and more. Companies can easily add functionality that is relevant to the region of operation, and that is customized to support even highly specialized industries. Business Central is fast to implement, easy to configure, and simplicity guides innovations in product design, development, implementation, and usability.

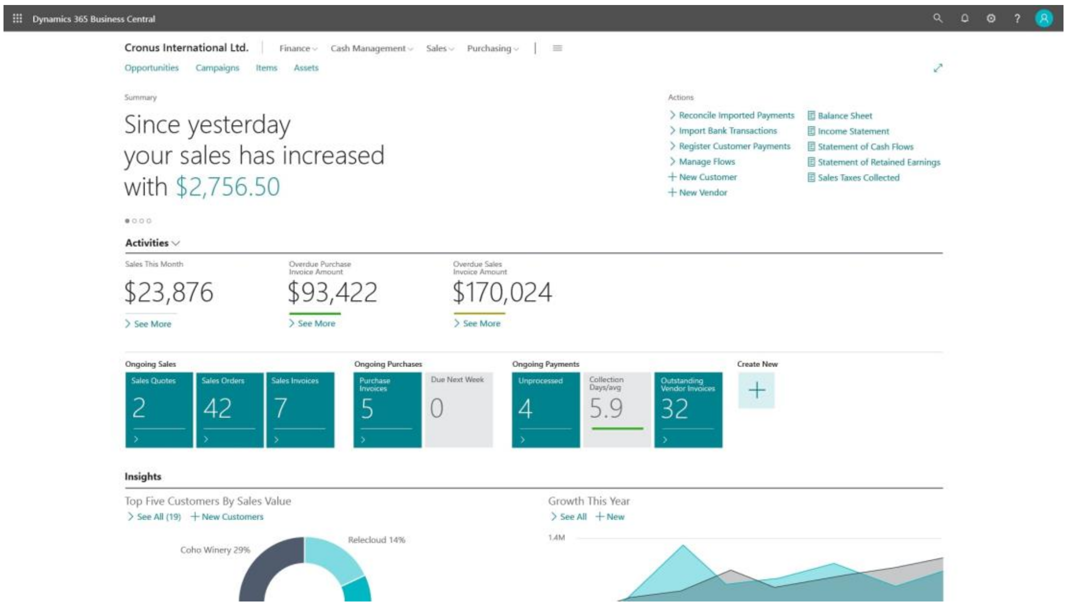


Fig 1.2

Business Central includes tooltips for fields and actions that can help guide you through the various business processes. Some pages also have teaching tips and tours to help you. On each tooltip and teaching tip, choose the **Learn more** link to open the Help pane where you find information about the current page and related tasks. On all pages, use *Ctrl+F1* on your keyboard to open the Help pane. On any device, use the question mark in the upper right corner to get to the Help.

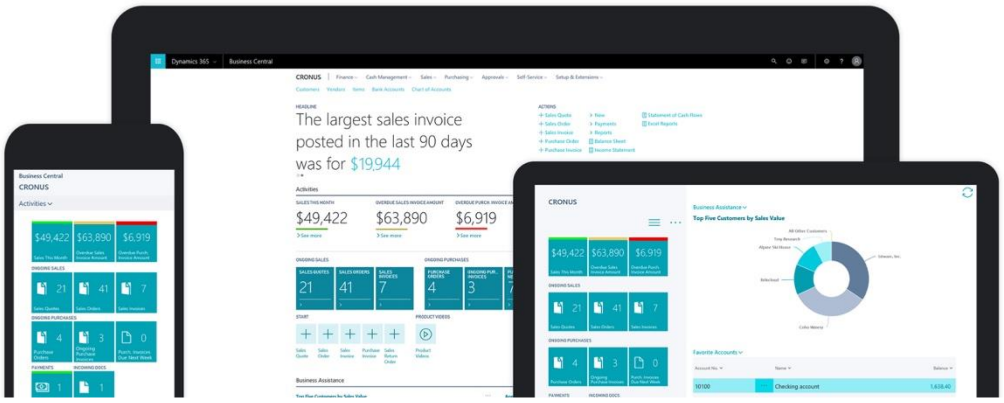


Fig 1.3

We are continuously working on our apps to give our users best updates and experience and premium feeling, when you use it. We now have almost 50+ clients and we are continuously working to increasing this number day by day.

CHAPTER – 2

INTRODUCTION TO PROJECT

Chapter – 2

Introduction to Project

2.1 What is IOT?

The internet of things, or IoT, is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.

A *thing* in the internet of things can be a person with a heart monitor implant, a farm animal with a biochip transponder, an automobile that has built-in sensors to alert the driver when tire pressure is low or any other natural or man-made object that can be assigned an Internet Protocol (IP) address and is able to transfer data over a network.

Fig 2.1

2.1.1 Why is IoT important?

The internet of things helps people live and work smarter, as well as gain complete control over their lives. In addition to offering smart devices to automate homes, IoT is essential to business. IoT provides businesses with a real-time look into how their systems really work, delivering insights into everything from the performance of machines to supply chain and logistics operations.

IoT enables companies to automate processes and reduce labour costs. It also cuts down on waste and improves service delivery, making it less expensive to manufacture and deliver goods, as well as offering transparency into customer transactions.

As such, IoT is one of the most important technologies of everyday life, and it will continue to pick up steam as more businesses realize the potential of connected devices to keep them competitive.

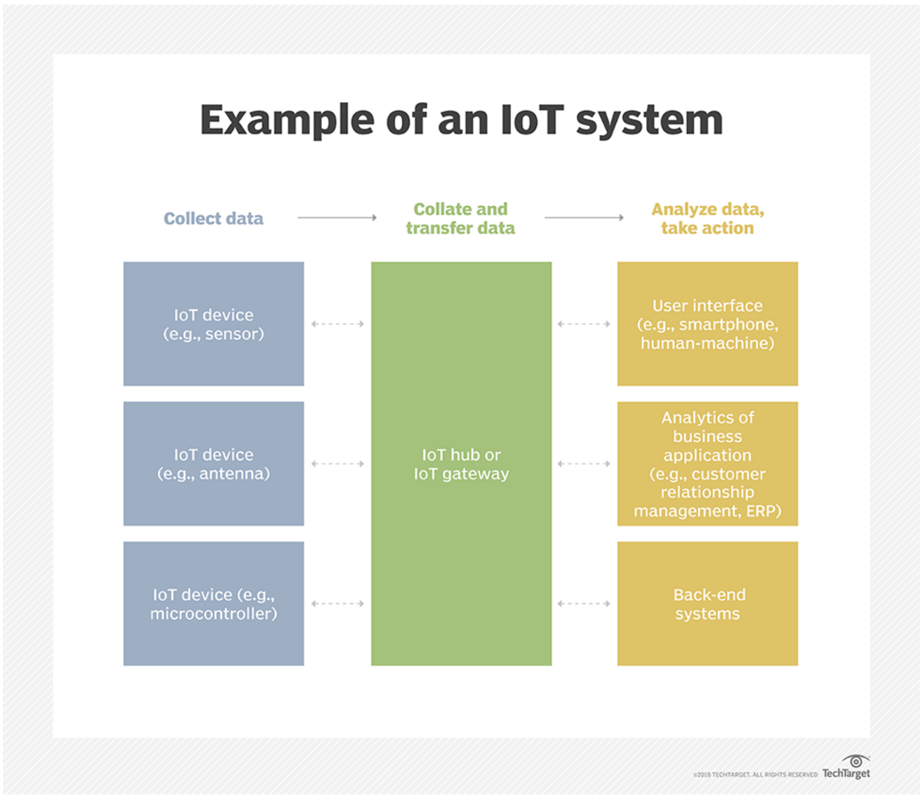

Example of an IoT system
Collect data
Collate and transfer data
Analyze data, take action
IoT device (e.g., sensor)
IoT device (e.g., antenna)
IoT device (e.g., microcontroller)
IoT hub or IoT gateway
User interface (e.g., smartphone, human-machine)
Analytics of business application (e.g., customer relationship management, ERP)
Back-end systems
©2018 TECHTARGET. ALL RIGHTS RESERVED
TechTarget

Fig 2.2

2.2 PROJECT DETAILS

This is an IoT based home automatic system for plants and home appliances using NodeMCU microcontroller. With the help of WIFI, the moisture and other data can be monitored and controlled from anywhere around the world.

2.2.1 About NodeMCU: -

NodeMCU is a low-cost open source IoT platform. It initially included firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which was based on the ESP-12 module. Later, support for the ESP32 32-bit MCU was added.

NodeMCU is an open source firmware for which open source prototyping board designs are available. The name "NodeMCU" combines "node" and "MCU" (micro-controller unit). The term "NodeMCU" strictly speaking refers to the firmware rather than the associated development kits.

Both the firmware and prototyping board designs are open source.

The firmware uses the Lua scripting language. The firmware 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. Due to resource constraints, users need to select the modules relevant for their project and build a firmware tailored to their needs. Support for the 32-bit ESP32 has also been implemented.

The prototyping hardware typically used is a circuit board functioning as a dual in-line package (DIP) which integrates a USB controller with a smaller surface-mounted board containing the MCU and antenna. The choice of the DIP format allows for easy prototyping on breadboards. The design was initially based on the ESP-12 module of the ESP8266, which is a Wi-Fi SoC integrated with a Tensilica Xtensa LX106 core, widely used in IoT applications (see related projects).

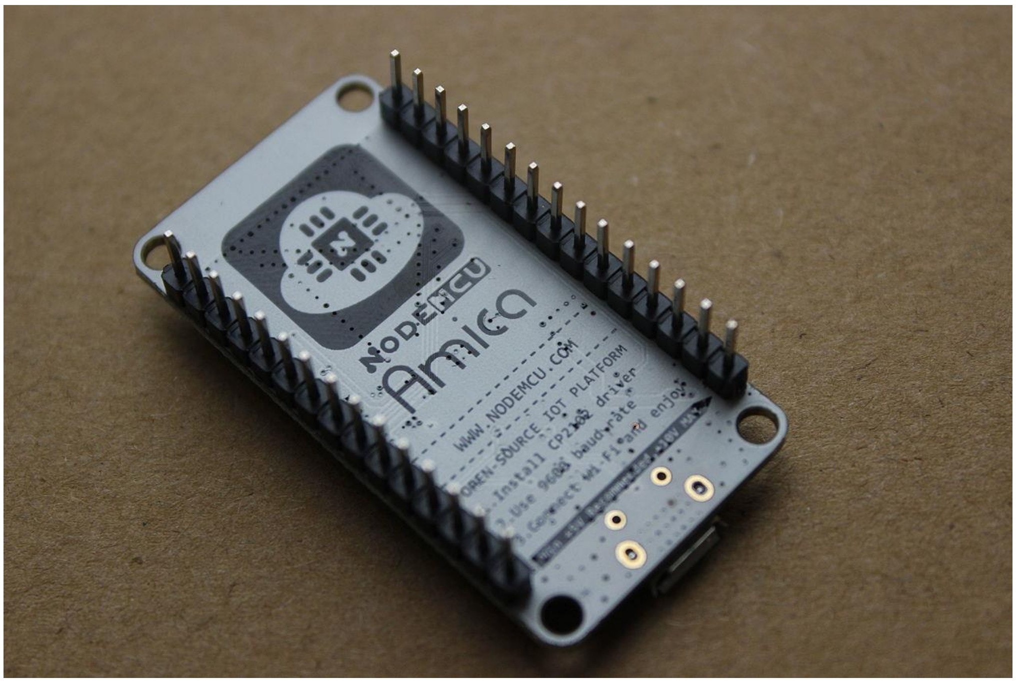


Fig 2.3

NodeMCU provides access to the GPIO (General Purpose Input/Output) and a pin mapping table is part of the API documentation.

O index	SP8266 pin
[*]	PIO16
	PIO5
	PIO4
	PIO0
	PIO2
	PIO14
	PIO12
	PIO13

	PIO15
	PIO3
)	PIO1
1	PIO9
2	PIO10

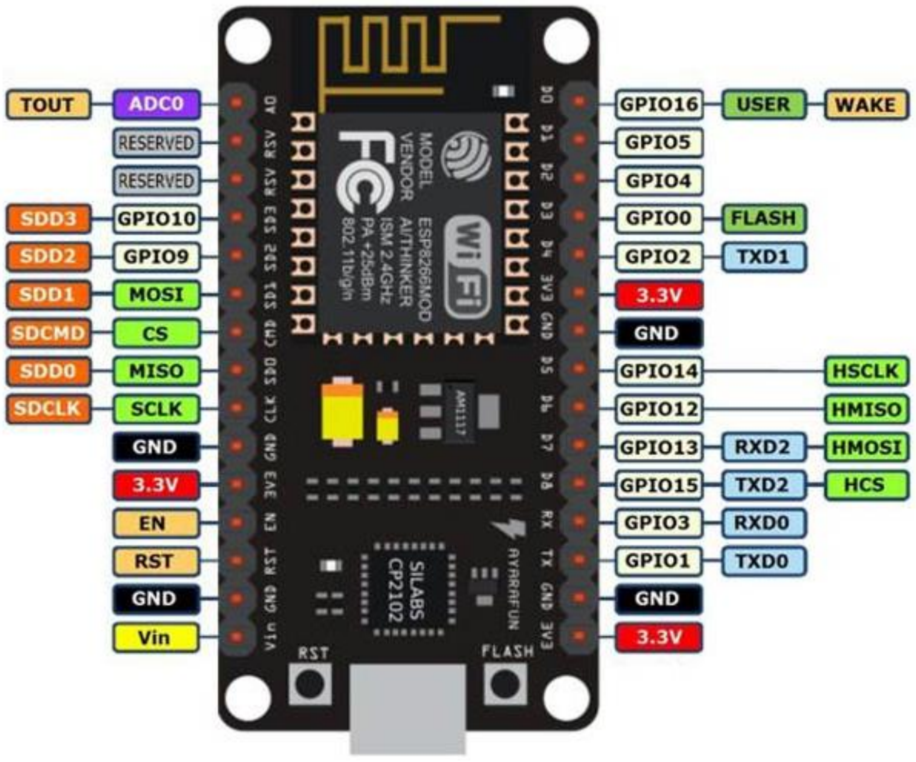


Fig 2.4

The **NodeMCU ESP8266 development board** comes with the ESP-12E module containing the ESP8266 chip having Tensilica Xtensa 32-bit LX106 RISC microprocessor. This microprocessor supports RTOS and operates at 80MHz to 160 MHz adjustable clock frequency. NodeMCU has 128 KB RAM and 4MB of Flash memory to store data and programs. Its high processing power with in-built Wi-Fi / Bluetooth and Deep Sleep Operating features make it ideal for IoT projects.

NodeMCU can be powered using a Micro USB jack and VIN pin (External Supply Pin). It supports UART, SPI, and I2C interface.

Programming NodeMCU ESP8266 with Arduino IDE

The NodeMCU Development Board can be easily programmed with Arduino IDE since it is easy to use.

Programming NodeMCU with the Arduino IDE will hardly take 5-10 minutes. All you need is the Arduino IDE, a USB cable and the NodeMCU board itself. You can check this Getting Started Tutorial for NodeMCU to prepare your Arduino IDE for NodeMCU.

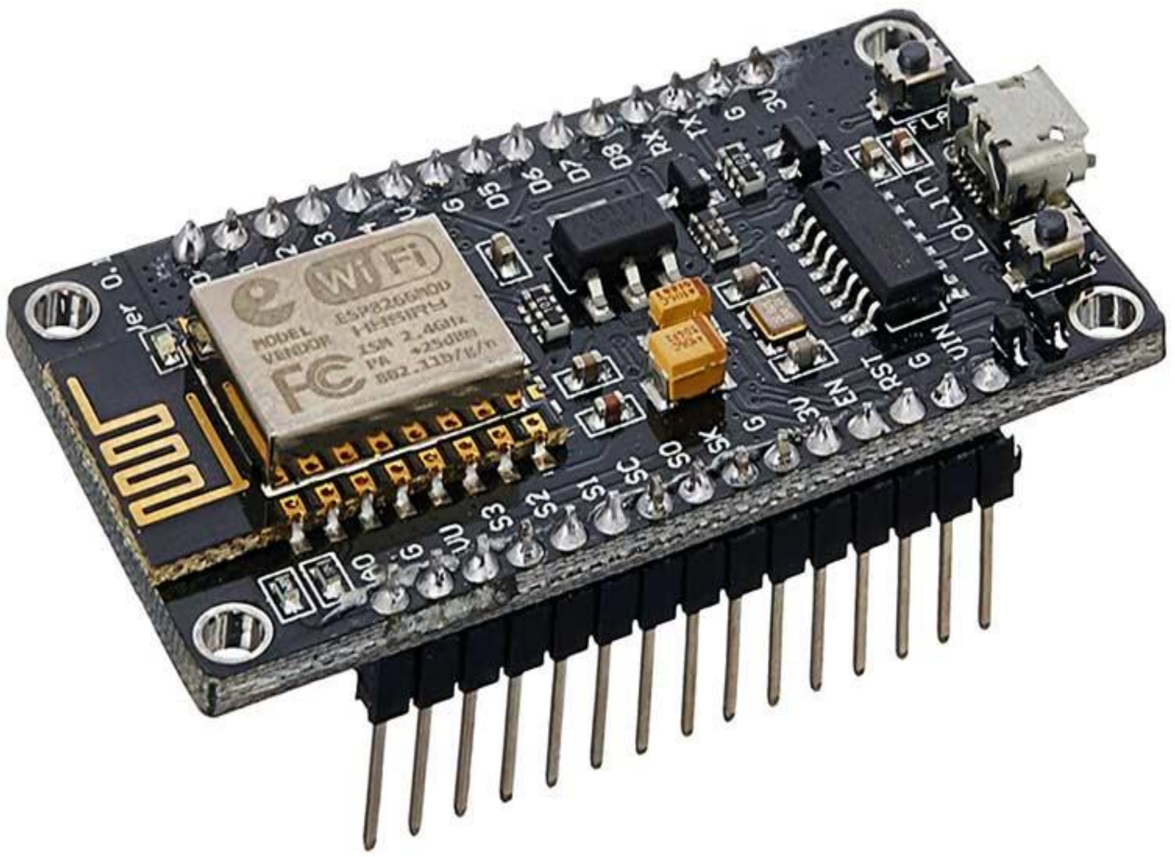


Fig 2.5

NodeMCU Development Board Pinout Configuration

in Category	ame	escription
ɔwer	Iicro-USB, 3.3V, GND, Vin	**Iicro-USB:** NodeMCU can be ɔwered through the USB port **3V:** Regulated 3.3V can be ıpplied to this pin to power th ɔard **ND:** Ground pins **in:** External Power Supply
ontrol Pins	**N, RST**	ıe pin and the button resets th icrocontroller
nalog Pin	0	sed to measure analog voltag e range of 0-3.3V
PIO Pins	PIO1 to GPIO16	odeMCU has 16 general purp put-output pins on its board
PI Pins	D1, CMD, SD0, CLK	odeMCU has four pins availa PI communication.

ART Pins	XD0, RXD0, TXD2, RXD2	odeMCU has two UART inte ART0 (RXD0 & TXD0) and ART1 (RXD1 & TXD1). UA sed to upload the firmware/pr
C Pins		odeMCU has I2C functionalit ipport but due to the internal inctionality of these pins, you find which pin is I2C.

NodeMCU ESP8266 Specifications & Features

- Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106
- Operating Voltage: 3.3V
- Input Voltage: 7-12V
- Digital I/O Pins (DIO): 16
- Analog Input Pins (ADC): 1
- UARTs: 1
- SPIs: 1
- I2Cs: 1
- Flash Memory: 4 MB
- SRAM: 64 KB
- Clock Speed: 80 MHz
- USB-TTL based on CP2102 is included onboard, Enabling Plug n Play
- PCB Antenna
- Small Sized module to fit smartly inside your IoT projects

2.2.2 Relay Module: -

The relay is the device that open or closes the contacts to cause the operation of the other electric control. It detects the undesirable condition with an assigned area and gives the commands to the circuit breaker to disconnect the affected area through ON or OFF.

Every electromechanical relay consists of

1. Electromagnet

3. Mechanically movable contact

3. Switching points and

4. Spring

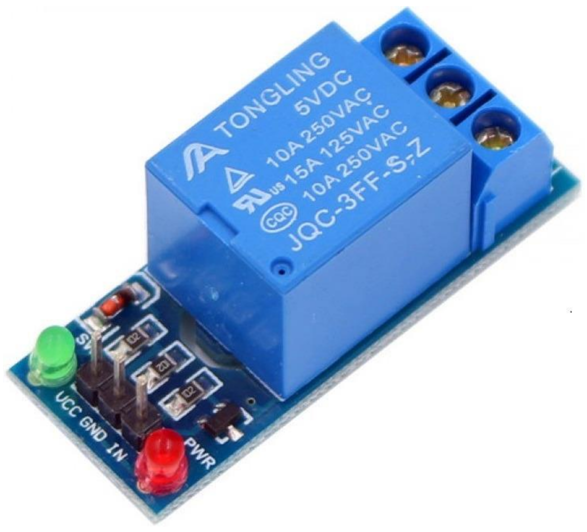


Fig 2.6

COM: common pin

NO: Normally open – there is no contact between the common pin and the normally open pin. So, when you trigger the relay, it connects to the COM pin and power is provided to the load.

NC: Normally closed – there is contact between the common pin and the normally closed pin. There is always connection between the COM and NC pins, even when the relay is turned off. When you trigger the relay, the circuit is opened and there is no supply provided to the load.

Working principle of relay:

It works on the principle of an electromagnetic attraction. When the circuit of the relay senses the fault current, it energizes the electromagnetic field which produces the temporary magnetic field. This magnetic field moves the relay armature for opening or closing the connections. The small power relay has only one contacts, and the high-

power relay has two contacts for opening the switch.
The inner section of the relay is shown in the figure below. It has an iron core which is wound by a control coil. The power supply is given to the coil through the contacts of the load and the control switch.

The current flow through the coil produces the magnetic field around it.
Due to this magnetic field, the upper arm of the magnet attracts the lower arm. Hence close the circuit, which makes the current flow through the load. If the contact is already closed, then it moves oppositely and hence open the contacts.
Types of Relays Based on the principle of operation

1. Electrothermal relay:
2. Electromechanical relay:
3. Solid State relay:
4. Hybrid relay:

Applications of relay:

A. They can be used for both ac and dc systems for protection of ac and dc equipment’s

B. Electromagnetic relays operating speeds which has the ability to operate in milliseconds are also can be possible

C. They have the properties such as simple, robust, compact and most reliable

D. These relays are almost instantaneous. Though instantaneous the operating time of the relay varies with the current. With extra arrangements like dashpot, copper rings.

E. Electromagnetic relays have fast operation and fast reset

Disadvantages:

a. High burden level instrument transformers are required (CTs and PTs of high burden is required for operating the electromagnetic relays compared to static relays)

b. The directional feature is absent in electromagnetic relays

c. Requires periodic maintenance and testing unlike static relays

d. Relay operation can be affected due to ageing of the components and dust, pollution resulting in spurious trips

e. Operation speed for an electromagnetic relay is limited by the mechanical inertia of the component.

2.2.3 Tech Stacks Used: -

In our application, we have used React which secure and fast. And for designing purposes we have used CSS and HTML so that user can feel a better UI. And for Backend we have used IOT Cloud.

CHAPTER -3

REQUIREMENT ANALYSIS AND TECHNOLOGIES USED

Chapter – 3

Requirement Analysis

3.1 Project Aims:

- To control home appliances or other sources connected to system remotely.
- To show details like room temperature and humidity on display pane.
- To build a custom layout for the IOT system for adding further enhancements in future.

3.2 Expected outcomes for this project are:

- Written report of findings and searching
- Demonstration of a working prototype involving either one or more methods (ideally with a desktop interface because android libraries are deprecated).

3.3 Software Requirements:

- Processor: Pentium 2.4 GHz or above
- Memory: 256 MB RAM or above
- Cache Memory: 128 KB or above
- Printer: Laser Printer
- Pen Drive: 5 GB

3.4 Hardware Requirements:

- Microcontroller: NodeMCU ESP 8266
- Sensors: DHT11
- Others: 4 Channel Relay Module, Jumper Wires, Bulb Sockets

3.5 Working on data from the following indicators-

- Access to basic home power supply
- Access to laptop
- Access to Internet Connectivity

CHAPTER – 4

DESIGN

Chapter-4

Design

Software design is a process to transform user requirements into some suitable form, which helps the programmer in software coding and implementation.

For assessing user requirements, an SRS (Software Requirement Specification) document is created whereas for coding and implementation, there is a need of more specific and detailed requirements in software terms. The output of this process can directly be used into implementation in programming languages.

Software design is the first step in SDLC (Software Design Life Cycle), which moves the concentration from problem domain to solution domain. It tries to specify how to fulfill the requirements mentioned in SRS.

Software design yields three levels of results:

- **Architectural Design -** The architectural design is the highest abstract version of the system. It identifies the software as a system with many components interacting with each other. At this level, the designers get the idea of proposed solution domain.
- **High-level Design-** The high-level design breaks the 'single entity-multiple component' concept of architectural design into less-abstracted view of sub-systems and modules and depicts their interaction with each other. High-level design focuses on how the system along with all its components can be implemented in forms of modules. It recognizes modular structure of each sub-system and their relation and interaction among each other.
- **Detailed Design-** Detailed design deals with the implementation part of what is seen as a system and its sub-systems in the previous two designs. It is more detailed towards modules and their implementations. It defines logical structure of each module and their interfaces to communicate with other modules.

- Smaller components are easier to maintain
- Program can be divided based on functional aspects
- Desired level of abstraction can be brought in the program
- Components with high cohesion can be re-used again
- Concurrent execution can be made possible

- Desired from security aspect

4.1 Data Flow: -

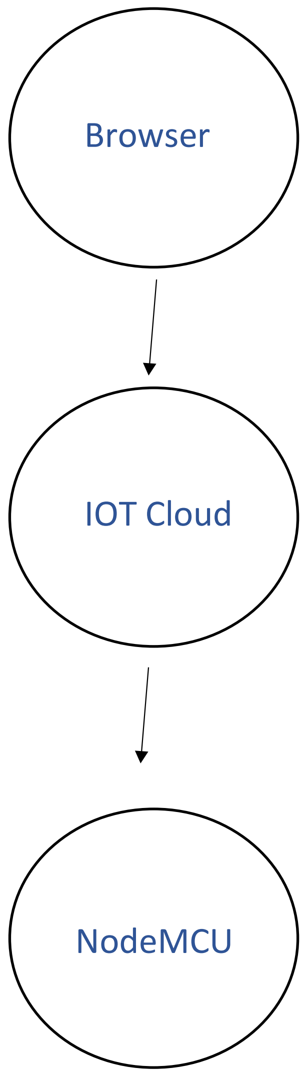


Fig 4.1

This microcontroller is getting its data from the IOT Cloud, after that on application we have applied certain conditions to get the desired output means we will have only that type of data on the screens which we want to see. Microcontroller has its own code unit, which we'll need to apply separately.

4.2 Quality Assurance: -

This QA step is also conducted on the behalf of our team. Here, we keep in mind that there’s no data leak and user get a quick response on every gesture he makes on his laptop. Our home automation is very reliable and has good encryption implementation hence making it difficult for hacking.

CHAPTER – 5

DATABASE

Chapter-5

Database

The Arduino IoT Cloud is an online platform that makes it easy for you to create, deploy and monitor IoT projects.
Connected devices around the world are increasing by billions every year. The Arduino IoT Cloud is a platform that allows anyone to create IoT projects, with a user-friendly interface, and an all-in-one solution for configuration, writing code, uploading and visualization.

How does it work?

Setting up the Arduino IoT Cloud and accessing the different features available involves a few simple steps. So, let’s look at how to go from start to finish!

1. Creating an Arduino Account

   To starting using the Arduino IoT cloud, we first need to log in or sign up to Arduino.

2. Go to the Arduino IoT Cloud

   After we have signed up, you can access the Arduino IoT Cloud from any page on arduino.cc by clicking on the four dots menu in the top right corner. You can also go directly to the Arduino IoT Cloud.

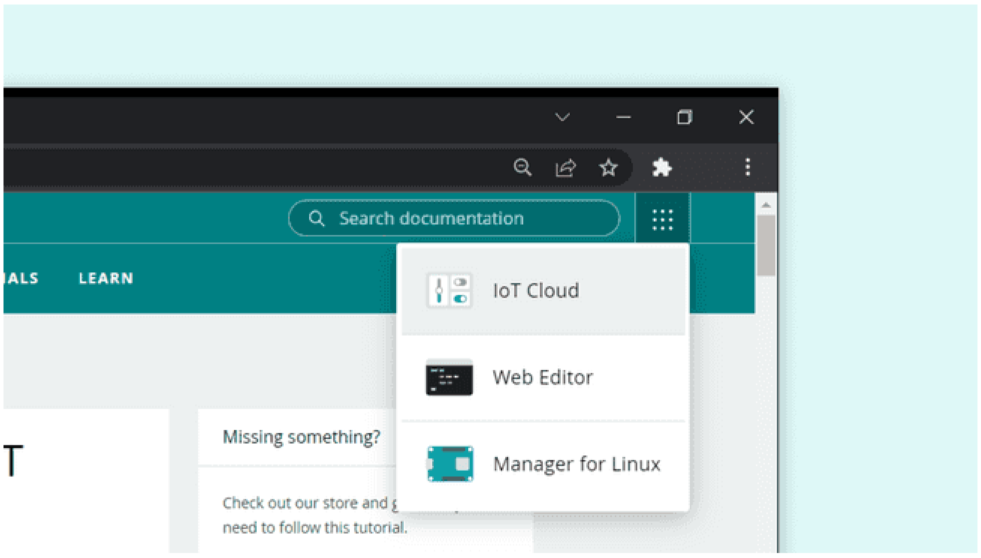


Fig 5.1

3. Creating a Thing

   The journey always begins by creating a new Thing. In the Thing overview, we can choose what device to use, what Wi-Fi network we want to connect to, and create variables that we can monitor and control. This is the main configuration space, where all changes we make are automatically generated into a special sketch file.

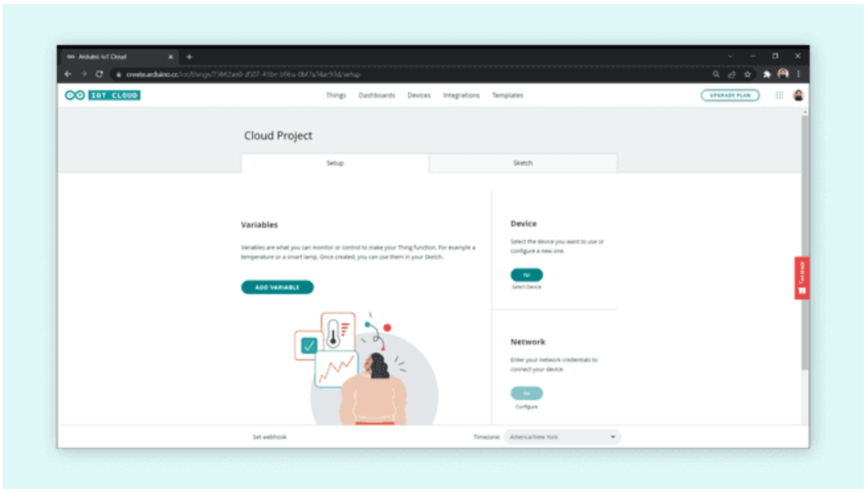
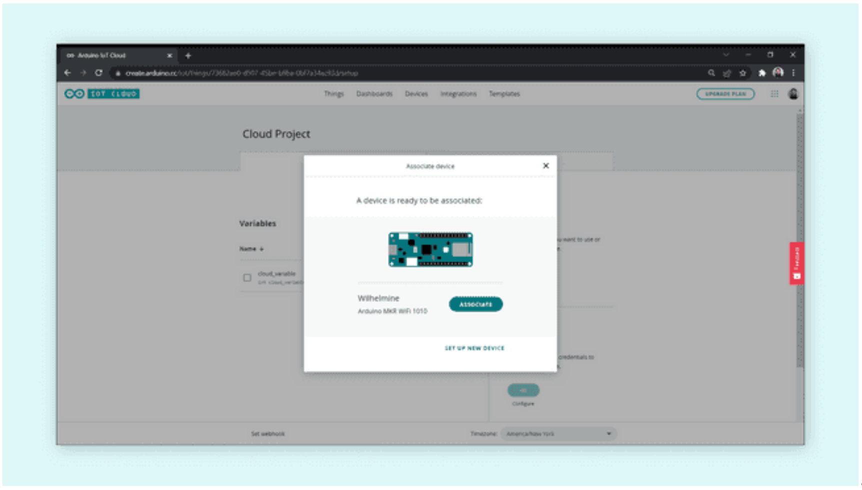


Fig 5.2

4. Configuring a Device

   Devices can easily be added and linked to a Thing. The Arduino IoT Cloud requires your computer to have the Arduino Agent installed. The configuration process is quick and easy and can be done by clicking on the "Select device" button in the Thing overview. Here, we can choose from any board that has been configured or select the "Configure new device" option.



.

Fig 5.3

We can also get a complete overview of our devices by clicking the “Devices" tab at the top of the Arduino IoT Cloud interface. Here we can manage and add new devices.

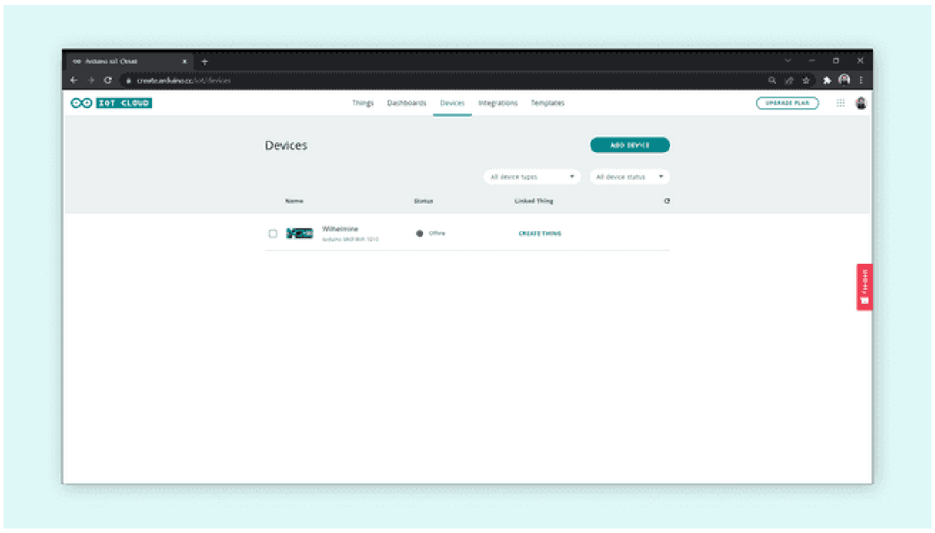


Fig 5.4

5. Creating Variables

The variables we create are automatically generated into a sketch file. There are several data types we can choose from, such as int, float, Boolean, long, char. There are also special variables, such as Temperature, Velocity, Luminance that can be used. When clicking on the “Add variable” button, we can choose name, data type, update setting and interaction mode.

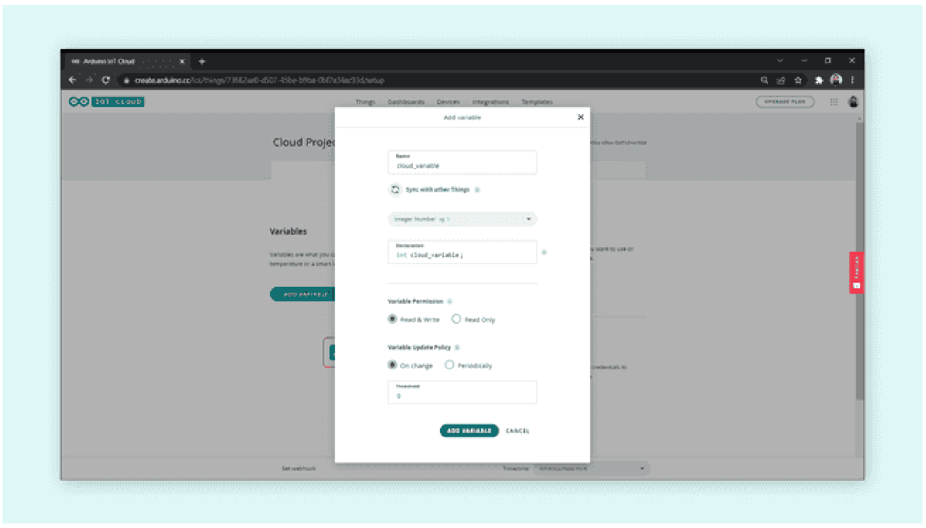
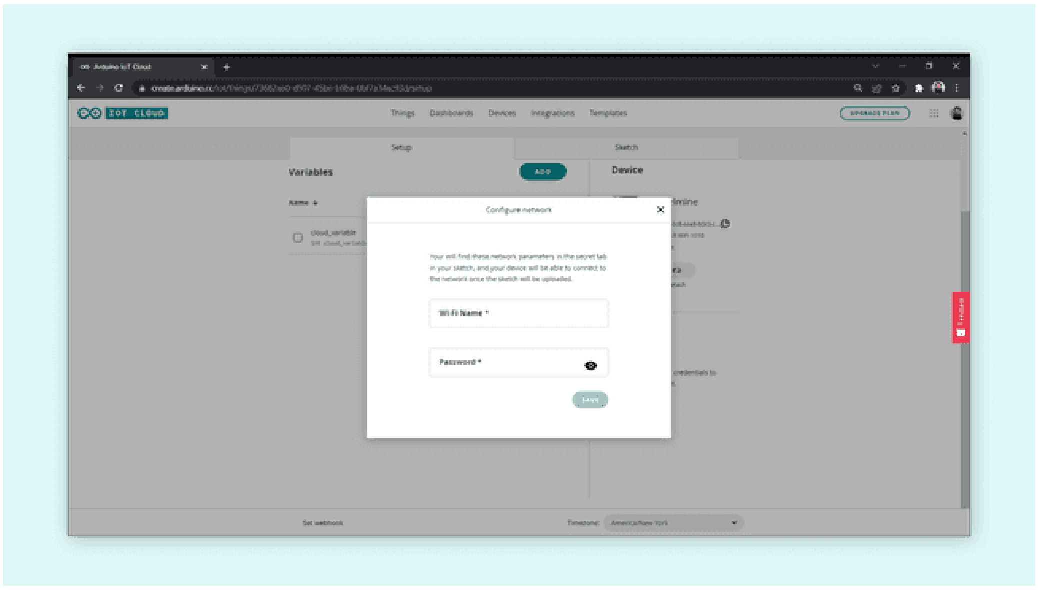


Fig 5.5

6. Connecting to a Network

   To connect to a Wi-Fi network, simply click the "Configure" button in the network section. Enter the credentials and click "Save". This information is also generated into your sketch file!



Your will find these network parameters in the secret tab in your sketch, and your device will be able to connect to the network once the sketch will be uploaded.

Fig 5.6

7. Editing the Sketch

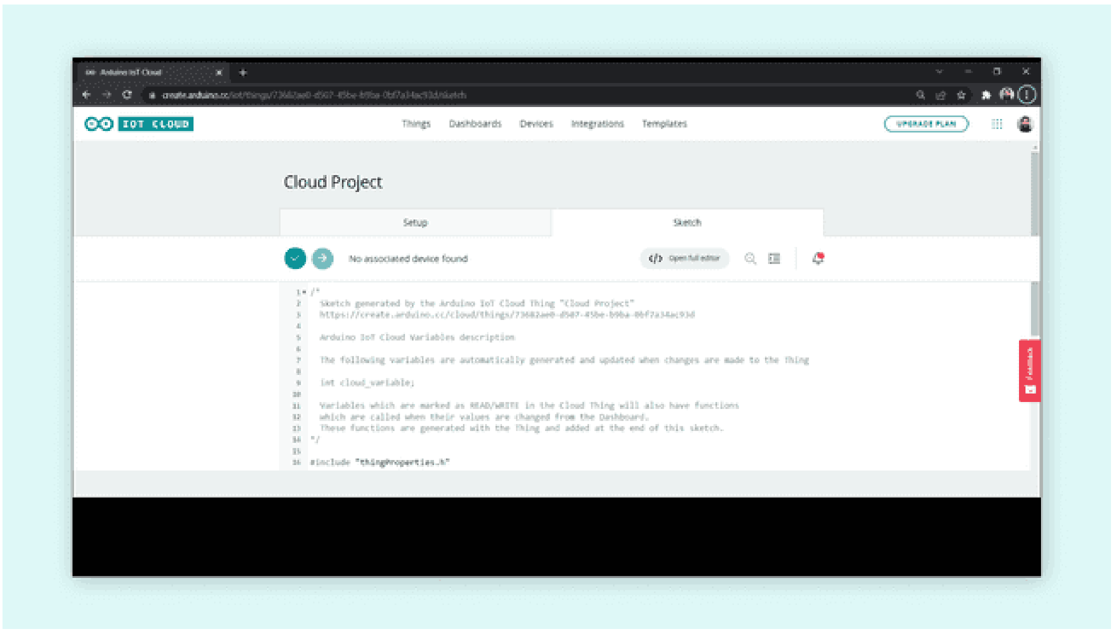
   Now that we have configured variables, devices, and network settings, we can get to programming our devices!

   An automatically generated sketch file can be found in the "Sketch" tab. It has the same structure as a typical **.ino** file, but with some additional code to make the connection to your network and to the cloud.

   A sketch that, for example, reads an analog sensor, and use the cloud variable to store it. When the sketch has been uploaded, it will work as a regular sketch, but it will also update the cloud variables that we use!

   Additionally, each time we create a variable that has the Read & Write permission enabled, a function is also generated, at the bottom of your sketch file. Each time these variable changes, it will execute the code within this function! This means that we can leave most of the code out of the loop () and only run code when needed.

To upload the program to our board, simply click the "Upload" button.

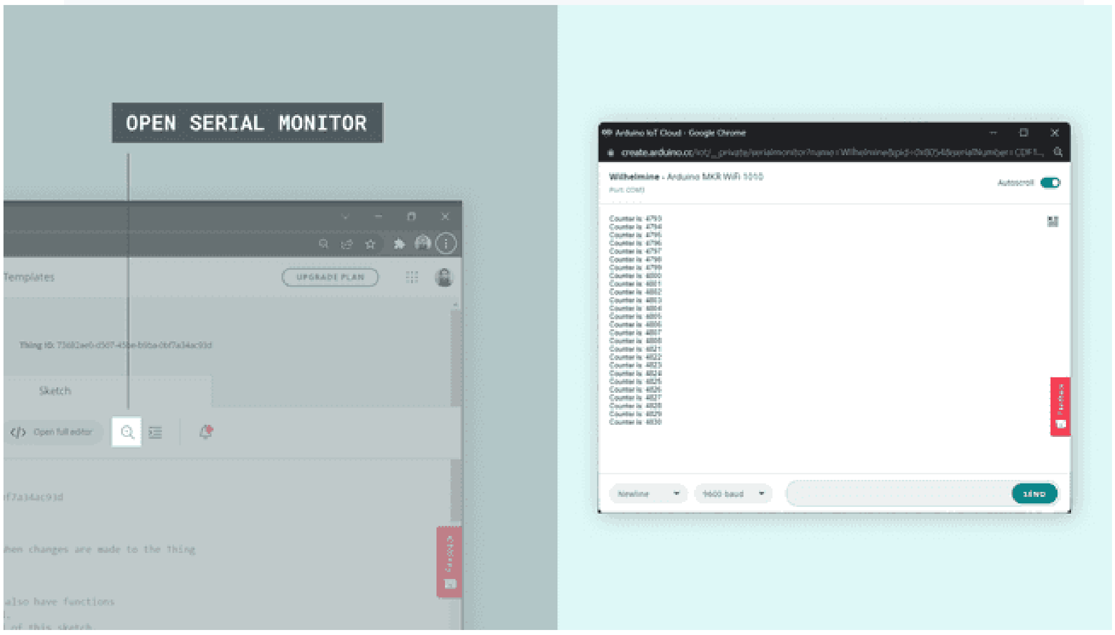


```
/*
  Sketch generated by the Arduino IoT Cloud Thing "Cloud Project"
  https://create.arduino.cc/cloud/things/73682ae0-d507-45be-b9ba-0bf7a34ac93d

  Arduino IoT Cloud Variables description

  The following variables are automatically generated and updated when changes are made to the Thing

  int cloud_variable;

  Variables which are marked as READ/WRITE in the Cloud Thing will also have functions
  which are called when their values are changed from the Dashboard.
  These functions are generated with the Thing and added at the end of this sketch.
*/

#include "thingProperties.h"
```

Fig 5.7

The editor also has a Serial Monitor Tool, which can be opened by clicking the magnifying glass in the toolbar. Here you can view information regarding your connection, or commands printed via

Serial.print()



```
Counter is 4793
Counter is 4794
Counter is 4795
Counter is 4796
Counter is 4797
Counter is 4798
Counter is 4799
Counter is 4800
Counter is 4801
Counter is 4802
Counter is 4803
Counter is 4804
Counter is 4805
Counter is 4806
Counter is 4807
Counter is 4808
Counter is 4821
Counter is 4822
Counter is 4823
Counter is 4824
Counter is 4825
Counter is 4826
Counter is 4827
Counter is 4828
Counter is 4829
Counter is 4830
```

Fig 5.8

After we have successfully uploaded the code, we can open the “Serial Monitor” tab to view information regarding our connection. If it is successful, it will print “connected to network_name” and “connected to cloud”. If it fails to connect, it will print the errors here as well.

The cloud editor is a mirrored "minimal" version of the Web Editor. Any changes you make will also be reflected in the Web Editor, which is more suitable for developing more advanced sketches.

8. Creating a Dashboard

   Now that we have configured the device & network, created variables, completed the sketch and successfully uploaded the code, we can move on to the fun part: creating dashboards!

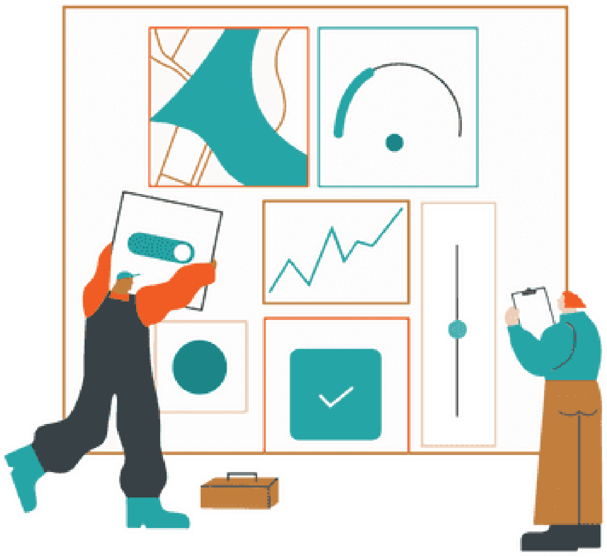


Fig 5.9

Dashboards are visual user interface for interacting with your boards over the cloud, and we can setup many different setups depending on what your IoT project needs. We can access our dashboards by clicking on the “Dashboards” tab at the top of the Arduino IoT Cloud interface, where we can create new dashboards, and see a list of dashboards created for other Things.

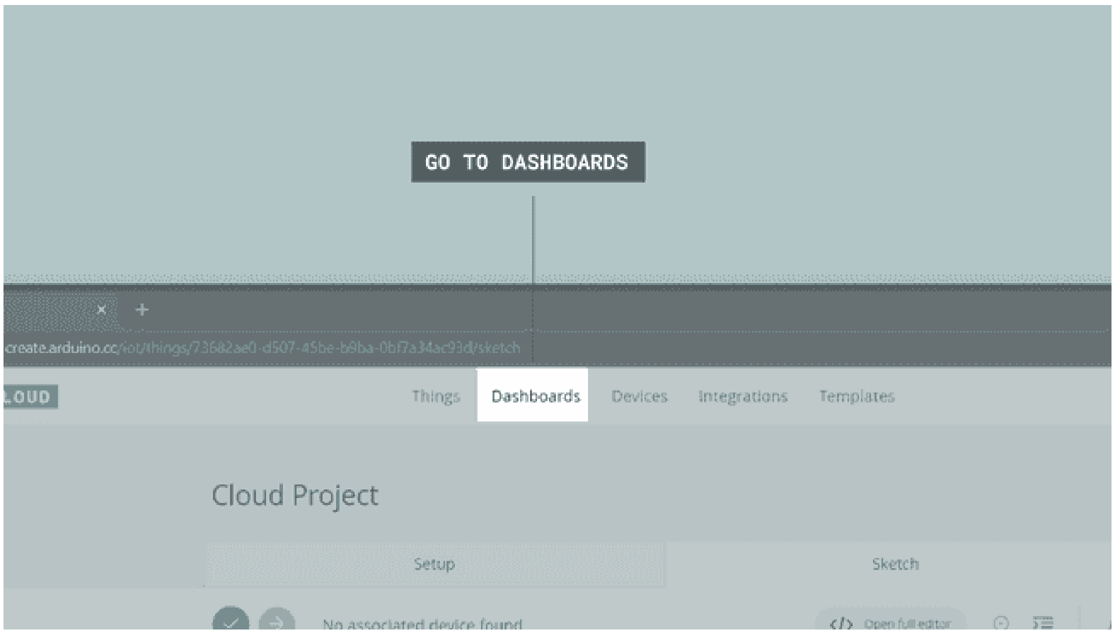
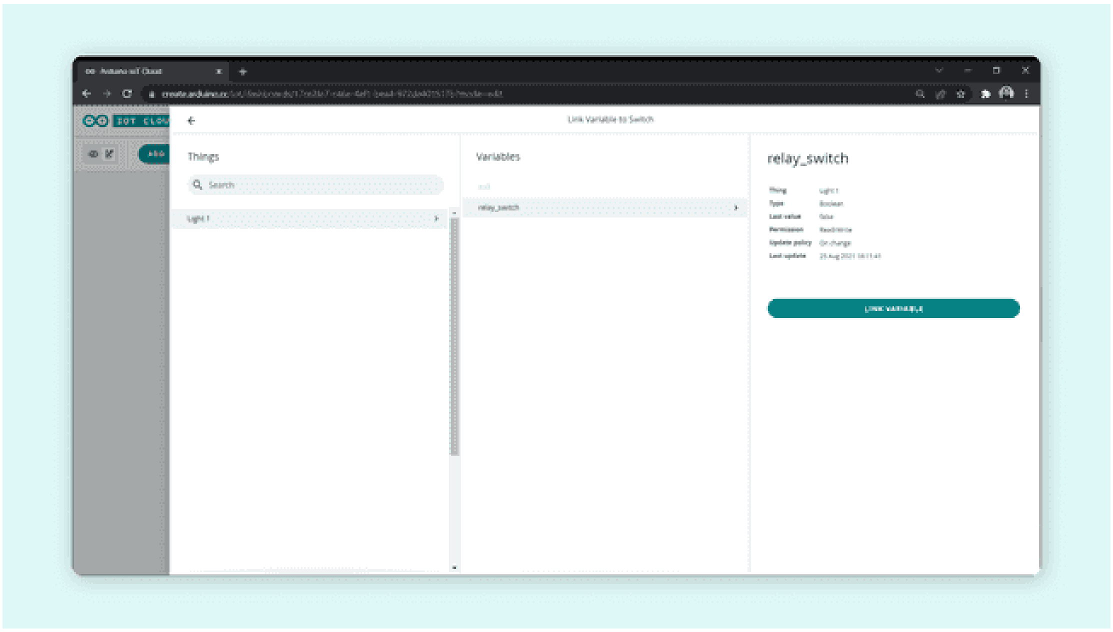


Fig 5.10

If we click on “Create new dashboard”, we enter a dashboard editor. Here, we can create something called widgets. Widgets are the visual representation of our variables we create, and there are many different to choose from. Below is an example using several types of widgets.

When we create widgets, we also need to link them to our variables. This is done by clicking on a widget we create, select a Thing, and select a variable that we want to link. Once it is linked, we can either interact with it, for example a button, or we can monitor a value from a sensor. If our board is connected to the cloud, the values will update!

Let's say we have a temperature widget that we want to link to the temperature variable inside the Cloud project thing.



relay_switch

Thing	Light 1
Type	Boolean
Last value	false
Permission	Read/Write
Update policy	On change
Last update	25 Aug 2021 [illegible]

Fig 5.11

Note that not all widgets and variables are compatible. A switch and an integer can for example not be linked and will not be an option while setting up your dashboard.

We can also have several things running at once, depending on your Arduino IoT Cloud plan, which we can include in the same dashboard. This is a great feature for tracking multiple boards in for example a larger sensor network, where boards can be connected to different networks around the world but be monitored from the same dashboard.

CHAPTER - 6

IMPLEMENTATION

Chapter No. 6
Implementation of Modules

6.1 NodeMCU Module

For the microcontroller, here we have used a unique Arduino programming language, especially designed for Arduino microcontrollers. The Arduino board is connected to a computer via USB, where it connects with the Arduino development environment (IDE). The user writes the Arduino code in the IDE, then uploads it to the microcontroller which executes the code, interacting with inputs and outputs such as sensors, motors, and lights.

Arduino code is written in C++ with an addition of special methods and functions, which we'll mention later on. C++ is a human-readable programming language. When you create a 'sketch' (the name given to Arduino code files), it is processed and compiled to machine language.

The Arduino Integrated Development Environment (IDE) is the main text editing program used for Arduino programming. It is where you'll be typing up your code before uploading it to the board you want to program. Arduino code is referred to as **sketches**.

Here for Wi-Fi connection, we'll create a file with SSID, and Password defined.

```
#define SECRET_SSID "Panda"

#define SECRET_PASS "12345678"

#define SECRET_DEVICE_KEY "VQKGG3BHBVD1F4SVFCR3"
```

Here for connecting our NodeMCU to IOT Cloud, we'll use the device ID and things ID generated when we created a project in IOT Cloud. We'll also mention the variables(things) we are dealing with.

```
#include <ArduinoIoTCloud.h>

#include <Arduino_ConnectionHandler.h>

const char THING_ID[]  = "7b536364-90c4-4712-8115-b05bf5b9aa75";  //Enter THING ID
const char DEVICE_LOGIN_NAME[]  = "544a27a9-3199-46c7-abdb-5a47b220c356"; //Enter
DEVICE ID

const char SSID[] = SECRET_SSID;    // Network SSID (name)

const char PASS[] = SECRET_PASS;    // Network password (use for WPA, or use as key for
WEP)

const char DEVICE_KEY[]  = SECRET_DEVICE_KEY;    // Secret device password

void onSwitch1Change();

void onSwitch2Change();

void onSwitch3Change();

void onSwitch4Change();

CloudSwitch switch1;

CloudSwitch switch2;

CloudSwitch switch3;

CloudSwitch switch4;

void initProperties(){

  ArduinoCloud.setBoardId(DEVICE_LOGIN_NAME);

  ArduinoCloud.setSecretDeviceKey(DEVICE_KEY);

  ArduinoCloud.setThingId(THING_ID);

  ArduinoCloud.addProperty(switch1, READWRITE, ON_CHANGE, onSwitch1Change);

  ArduinoCloud.addProperty(switch2, READWRITE, ON_CHANGE, onSwitch2Change);

  ArduinoCloud.addProperty(switch3, READWRITE, ON_CHANGE, onSwitch3Change);

  ArduinoCloud.addProperty(switch4, READWRITE, ON_CHANGE, onSwitch4Change);
}

WiFiConnectionHandler ArduinoIoTPreferredConnection(SSID, PASS);
```

Here is the main code for IOT cloud connection and switch functioning in the system.

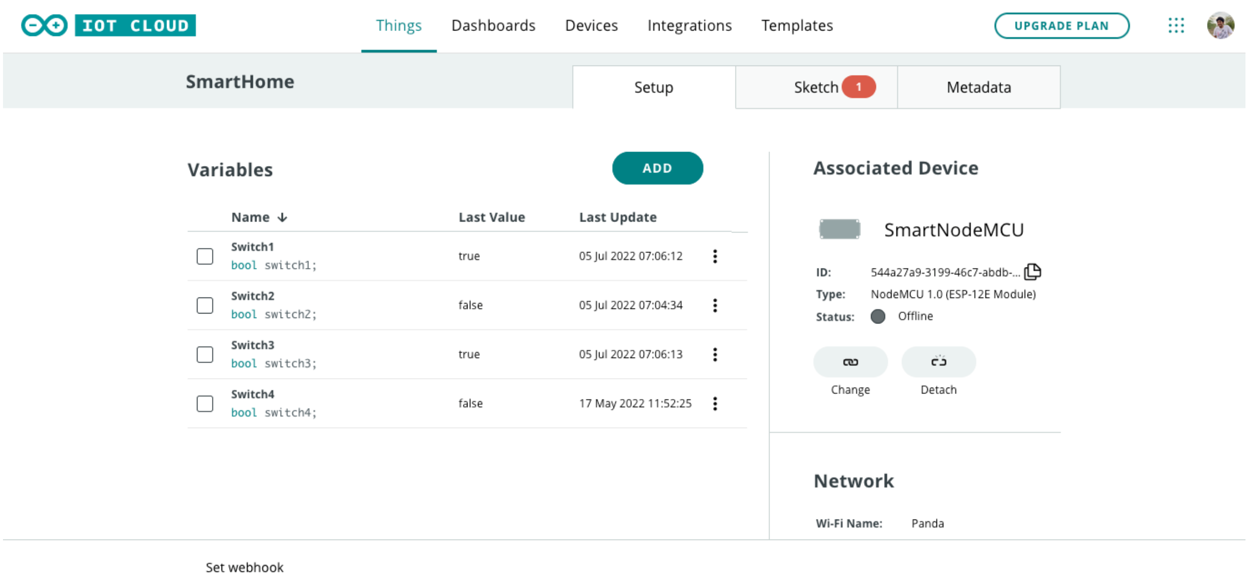
```
#include "arduino_secrets.h"

#include "thingProperties.h"

#include <DHT.h>

#include <DallasTemperature.h>

#define RelayPin1 5  //D1

#define RelayPin2 4  //D2

#define RelayPin3 14 //D5

#define RelayPin4 12 //D6

#define SwitchPin1 10  //SD3

#define SwitchPin2 0   //D3

#define SwitchPin3 13  //D7

#define SwitchPin4 3   //RX

#define wifiLed   16   //D0

int toggleState_1 = 0; //Define integer to remember the toggle state for relay 1

int toggleState_2 = 0; //Define integer to remember the toggle state for relay 2

int toggleState_3 = 0; //Define integer to remember the toggle state for relay 3

int toggleState_4 = 0; //Define integer to remember the toggle state for relay 4
```

```
void relayOnOff(int relay) {
  switch (relay) {
    case 1:
      if (toggleState_1 == 0) {
        digitalWrite(RelayPin1, LOW); // turn on relay 1
        toggleState_1 = 1;
        Serial.println("Device1 ON");
      }
      else {
        digitalWrite(RelayPin1, HIGH); // turn off relay 1
        toggleState_1 = 0;
        Serial.println("Device1 OFF");
      }
      delay(100);
      break;
    case 2:
      if (toggleState_2 == 0) {
        digitalWrite(RelayPin2, LOW); // turn on relay 2
        toggleState_2 = 1;
        Serial.println("Device2 ON");
      }
      else {
        digitalWrite(RelayPin2, HIGH); // turn off relay 2
        toggleState_2 = 0;
        Serial.println("Device2 OFF");
      }
      delay(100);
      break;
  }
}
void setup() {
  // Initialize serial and wait for port to open:
  Serial.begin(9600);
  // This delay gives the chance to wait for a Serial Monitor without blocking if none is found
  delay(1500);
   initProperties();
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();
  pinMode(RelayPin1, OUTPUT);
  pinMode(RelayPin2, OUTPUT);

  pinMode(wifiLed, OUTPUT);

  pinMode(SwitchPin1, INPUT_PULLUP);

  pinMode(SwitchPin2, INPUT_PULLUP);

  //During Starting all Relays should TURN OFF

  digitalWrite(RelayPin1, HIGH);

  digitalWrite(RelayPin2, HIGH);

  digitalWrite(wifiLed, HIGH);  //Turn OFF WiFi LED

}

void sendTemps()

{

  sensor=analogRead(A0);

  sensors.requestTemperatures();

  float temp = sensors.getTempCByIndex(0);

  Serial.println(temp);

  Serial.println(sensor);

  Blynk.virtualWrite(V1, temp);

  Blynk.virtualWrite(V2,sensor);

  delay(1000);

}
```

```
 if (WiFi.status() != WL_CONNECTED)
  {
    digitalWrite(wifiLed, HIGH); //Turn OFF WiFi LED
  } else{
    digitalWrite(wifiLed, LOW); //Turn ON WiFi LED
  }
}
void onSwitch1Change() {
  if (switch1 == 1)
  {
    digitalWrite(RelayPin1, LOW);
    Serial.println("Device1 ON");
    toggleState_1 = 1;
  }
  else
  {
    digitalWrite(RelayPin1, HIGH);
    Serial.println("Device1 OFF");
    toggleState_1 = 0;
  }
```

}

6.2 IOT Cloud



SmartHome

Variables

Name	Last Value	Last Update
Switch1 `bool switch1;`	true	05 Jul 2022 07:06:12
Switch2 `bool switch2;`	false	05 Jul 2022 07:04:34
Switch3 `bool switch3;`	true	05 Jul 2022 07:06:13
Switch4 `bool switch4;`	false	17 May 2022 11:52:25

Associated Device

SmartNodeMCU

ID: 544a27a9-3199-46c7-abdb-...
Type: NodeMCU 1.0 (ESP-12E Module)
Status: Offline

Network

Wi-Fi Name: Panda

Fig 6.1

As you can see, the IDE has a minimalist design. There are only 5 headings on the menu bar, as well as a series of buttons underneath which allow you to verify and upload your sketches. Essentially, the IDE translates and compiles your sketches into code that Arduino can understand. Once your Arduino code is compiled it's then uploaded to the board's memory.

All the user has to do to start compiling their sketch is press a button (a guide to this can be found below).

If there are any errors in the Arduino code a warning message will flag up prompting the user to make changes. Most new users often experience difficulty with compiling because of Arduino's stringent syntax requirements. If you make any mistakes in your punctuation when using Arduino, the code won't compile and you'll be met with an error message.

As any other piece of productized software, the modern API has its own software development lifecycle (SDLC) of designing, testing, building, managing, and versioning. Also, modern APIs are well documented for consumption and versioning.

6.3 React Module

First we’ll create a HTML and a CSS module for switch.

Here we’ll import the stylesheet vc-toggle-switch.css in the HTML page.

```
<link href="vc-toggle-switch.css" rel="stylesheet" />
```

The markup structure to create a toggle switch and define the on/off text in the data attributes:

```
<div class="vc-toggle-container">
  <label class="vc-small-switch">
    <input type="checkbox" class="vc-switch-input" />
    <span class="vc-switch-label" data-on="Yes" data-off="No"></span>
    <span class="vc-switch-handle"></span>
  </label>
</div>
```

Customize the switch.

```
<div style="
  /*Background color when it's turned off*/
  --vc-off-color: #d1d3d4;
  /*Background color when it's turned on*/
  --vc-on-color: #38cf5b;

  /*Animation speed and type*/
  --vc-animation-speed: 0.15s ease-out;
  /*Font used by the text*/
  --vc-font-family: Arial;
  /*The size used*/
  --vc-font-size: 11px;
  /*The font weight*/
  --vc-font-weight: 300;
```

```
/*Font color when the switch is on*/
--vc-on-font-color: white;
/*Font color when the switch is off*/
--vc-off-font-color: white;
/*How far the OFF text is from the right side*/
--vc-label-position-off: 12px;
/*How far the ON text is from the left side*/
--vc-label-position-on: 11px;
/*Small switch width*/
--vc-width: 50px;
/*Small switch height*/
--vc-height: 25px;
/*Border radius for the handle*/
--vc-handle-border-radius: 20px;
/*Border radius for the box*/
--vc-box-border-radius: 18px;
/*Shadow for the handle*/
--vc-handle-shadow: 1px 1px 5px rgba(0, 0, 0, 0.2);

/*Handle color*/
--vc-handle-color: white;
/*Handle width*/
--vc-handle-width: 15px;
/*Handle height*/
--vc-handle-height: 15px;
/*The handle's width while the toggle is clicked*/
--vc-onclick-width: 30px;
/*Hadle's distance from the top*/
--vc-handle-top: 5px;
">
<div class="vc-toggle-container">
  <label class="vc-small-switch">
    <input type="checkbox" class="vc-switch-input" />
```

```
      <span class="vc-switch-label" data-on="Yes" data-off="No"></span>
      <span class="vc-switch-handle"></span>
    </label>
  </div>
</div>
```

For creating the toggle switch using react, we’ll not use any library instead, we’ll design it from scratch. After creating directory and html files, we’ll alter changes to app file.

```
import React from 'react';
import ToggleSwitch from './ToggleSwitch/ToggleSwitch'
function App() {
  return (
    <ToggleSwitch />
  );
}
export default App;
```

Now for converting HTML components to react, we’ll make a class component.

```
class ToggleSwitch extends Component {
  render() {
    return (
      <div className="toggle-switch">
        <input
          type="checkbox"
          className="toggle-switch-checkbox"
          name="toggleSwitch"
          id="toggleSwitch"
        />
        <label className="toggle-switch-label" htmlFor="toggleSwitch">
          <span className="toggle-switch-inner" />
          <span className="toggle-switch-switch" />
        </label>
      </div>
    );
  }
}
export default ToggleSwitch;
```

We’ll then apply a style to the Toggle Switch’s inner <span> element when it’s focused with the keyboard, but not when it’s clicked. We’ll theme the same for all 4 switches

CHAPTER - 7

TESTING STRATEGY

Chapter – 7

Information about Testing Strategy

7.1 Types for Data:

The data is selected on the current trends, what people like to watch in news and what they want to hear in a news. This was our targeting strategy that which type of news are running on market and will remain evergreen that people will always love to listen that type of news.

7.2 Software Testing:

Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is Defect free. It involves execution of software/system components using manual or automated tools to evaluate one or more properties of interest. The purpose of software testing is to identify errors, gaps, or missing requirements in contrast to actual requirements.

7.2.1 Why Software Testing is Important?

Software Testing is Important because if there are any bugs or errors in the software, it can be identified early and can be solved before delivery of the software product. Properly tested software product ensures reliability, security, and high performance which further results in time saving, cost effectiveness and customer satisfaction.

- **Cost-Effective:** It is one of the important advantages of software testing. Testing any IT project on time helps you to save your money for the long term. In case if the bugs caught in the earlier stage of software testing, it costs less to fix.
- **Security:** It is the most vulnerable and sensitive benefit of software testing. People are looking for trusted products. It helps in removing risks and problems earlier.

- **Product quality:** It is an essential requirement of any software product. Testing ensures a quality product is delivered to customers.
- **Customer Satisfaction:** The main aim of any product is to give satisfaction to their customers. UI/UX Testing ensures the best user experience.

7.3 Testing Strategies in Software Engineering

Here are important strategies in software engineering:

Unit Testing: This software testing basic approach is followed by the programmer to test the unit of the program. It helps developers to know whether the individual unit of the code is working properly or not.

Integration testing: It focuses on the construction and design of the software. You need to see that the integrated units are working without errors or not.

System testing: In this method, your software is compiled as a whole and then tested as a whole. This testing strategy checks the functionality, security, portability, amongst others.

7.4 Outliers or Extreme Observations

An outlier denotes an observation much farther away from the rest of the observations. In quantitative terms, an observation could be so high that it falls beyond an acceptable upper threshold level and vice-versa. The two simplest ways of detecting outliers is:

1) EDA (exploratory data analysis) using boxplot, histogram, or 2-d scatterplot

2) Knowledge coupled with experience in domain.

We will not be discussing the intricacies involved in how to accurately classify an observation as extreme using exploratory data analysis as part of this document. Instead, another detailed document may be provided for enhanced understanding.

7.5 Different Levels of Software Testing

Software level testing can be majorly classified into 4 levels:

1. **Unit Testing:** A level of the software testing process where individual units/components of a software/system are tested. The purpose is to validate that each unit of the software performs as designed.
2. **Integration Testing:** A level of the software testing process where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units.
3. **System Testing:** A level of the software testing process where a complete, integrated system/software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements.
4. **Acceptance Testing:** A level of the software testing process where a system is tested for acceptability. The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery.

Fig 7.1

7.6 Software Testing Techniques

Software techniques can be majorly classified into two categories:

1. **Black Box Testing:** The technique of testing in which the tester doesn't have access to the source code of the software and is conducted at the software interface without concern with the internal logical structure of the software is known as black-box testing. For this we tested every screen which can appear, basically our application has 4 major screens. First One, the Splash Screen which will appear for 2 seconds, and this screen shows a logo of our application, our application name is stylish font and a subtitle for our application. Second Screen which shows news types of selection bar and new cards which contains a little regarding news. Third Screen shows news in little brief, and the fourth screen basically is our chrome's tab which show news in detail.

2. **White-Box Testing:** The technique of testing in which the tester is aware of the internal workings of the product, has access to its source code, and is conducted by making sure that all internal operations are performed according to the specifications is known as white box testing.
Our complete code is tested by taking the concepts of OOPs and Data Structure in mind, that's why our code has short length and has multiple use of Data Structures like Array List, Stacks, Queues and Linked List.

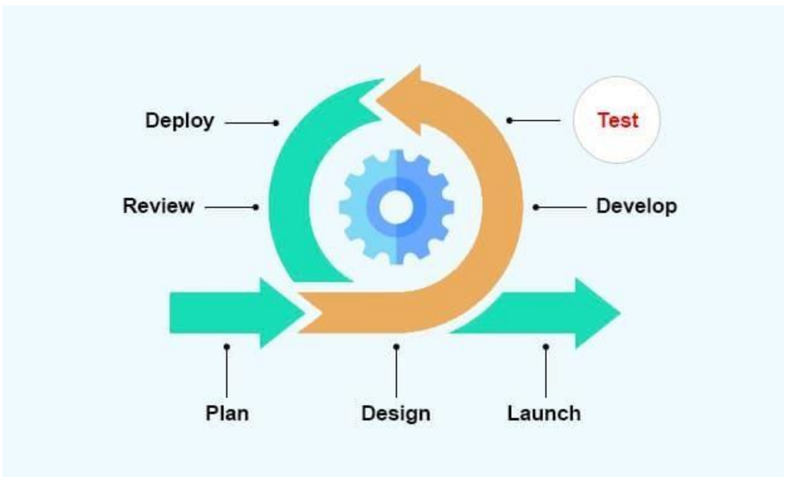


Fig 7.2

7.7 Feature Labels:

A feature label gives context to data turning it into a meaningful object. We have observed the data from various site., So that we can provide our user with the best types of news and current demanding or trending news.

7.8 Hardware Testing:

Hardware Testing is a specific test designed to verify the expected environmental conditions and maintenance activities should be used. Some of the may have issues due to a bad digital pin. Most people's first reaction is that there is something wrong with the code. We can save a lot of time if we take a few minutes to test our microcontroller before our project.

- Pick a Logic Analyzer

It includes a 2-Channel Oscilloscope, 2-Channel Waveform Generator, 16-Channel Logic Analyzer, 16-Channel Digital Pattern Generator, ±5VDC Power Supplies, Spectrum Analyzer, Network Analyzer, Voltmeter, Digital I/O, and it is supported by MATLAB. Overall it is a great tool for learning and helps us check the input in the pins of microcontroller.

- Pick a Development Board

Pick any development board. Press on the reset button for refreshing the board.

- Upload the Code

Just upload the code to your microcontroller. For the testing microcontroller, then have each of the pins trigger high for a second and then switch to low. After cycling through all of the pins individually, simultaneously trigger all the pins high for a second and then low.

- Test

We’ll connect each digital pin of your microcontroller to a channel of the logic analyzer. Switch on the microcontroller with the uploaded code. Check to see that each pin is functioning as intended. If something is wrong, we will see it immediately. Whether the pin is not triggering at all or the timing is wrong, you can easily find the problem using a logic analyzer.

We will usually test a few at a time and mark them with a green paint marker if they pass the test. We can also mark bad pins with a red paint marker.

CHAPTER – 8

SNAPSHOTS OF GUI

Chapter – 8

Snapshots of GUI

Here's our complete project.

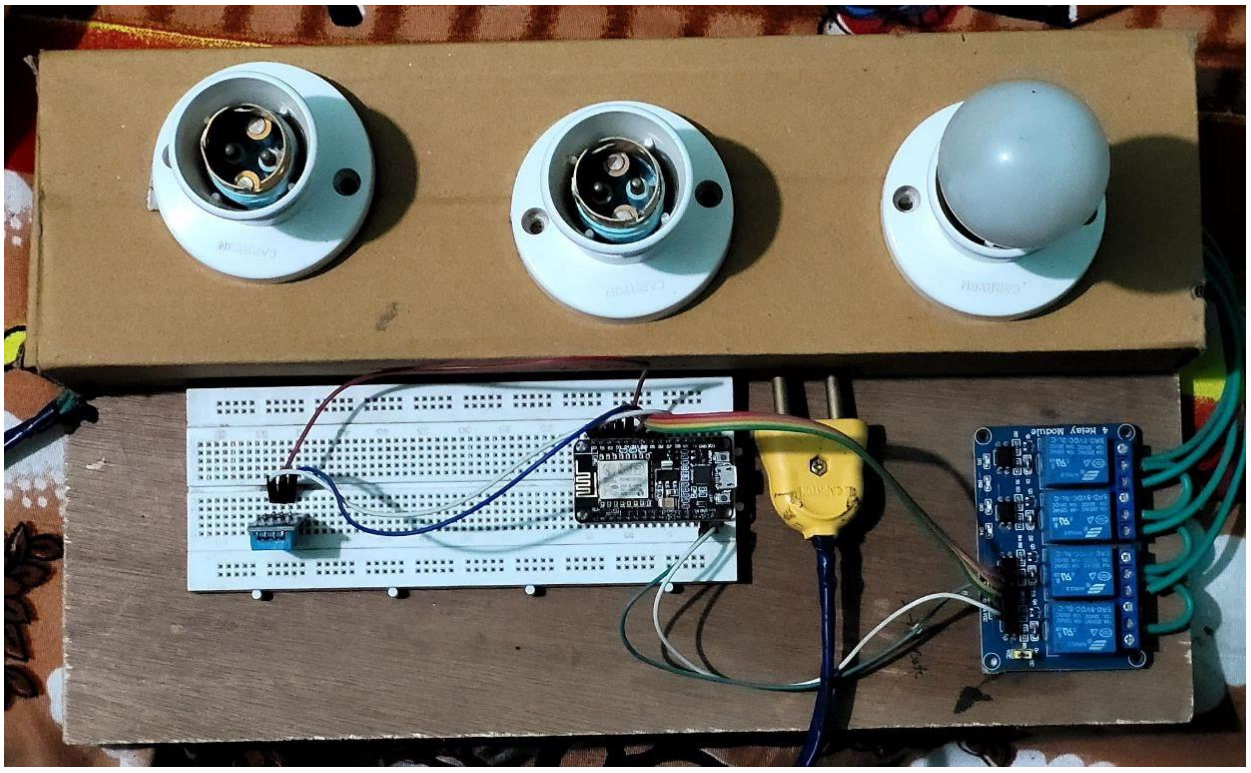


Fig 8.1

Here we've connected three sockets to demonstrate the connection of a 220V supply with relay module. The source is controlled by relay module, while the relay module is controlled by NodeMCU. We've also connected a DHT11 sensor to sense the temperature and humidity in the room, which refreshes the data every 5 minutes.

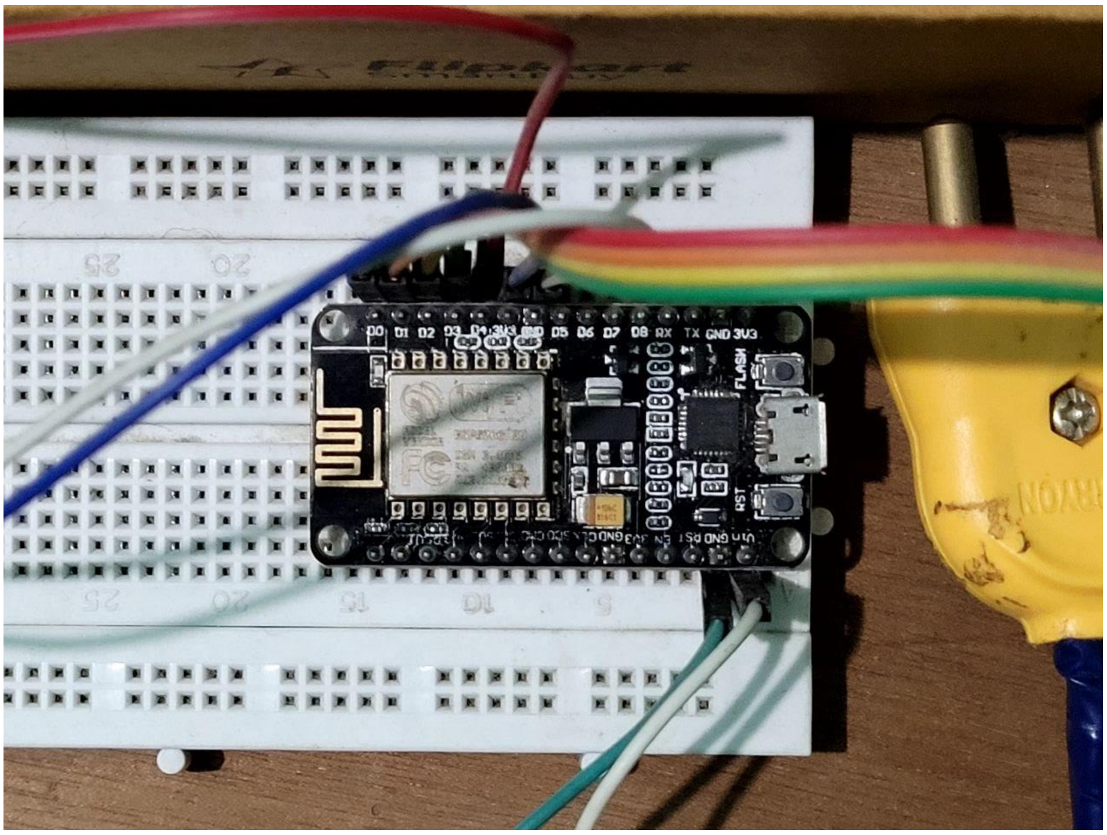


Fig 8.2

Here is the NodeMCU Amica connected to a breadboard. An ESP 8266 chip is embedded in the microcontroller itself.

A breadboard, solderless breadboard, protoboard, or terminal array board is a construction base used to build semi-permanent prototypes of electronic circuits. The holes in a breadboard are connected by metal clips that span five holes, horizontally. These metal clips allow each row of five holes to be connected. There are no vertical connections on a terminal strip. Horizontal rows on either side of the center groove are also not connected to each other.

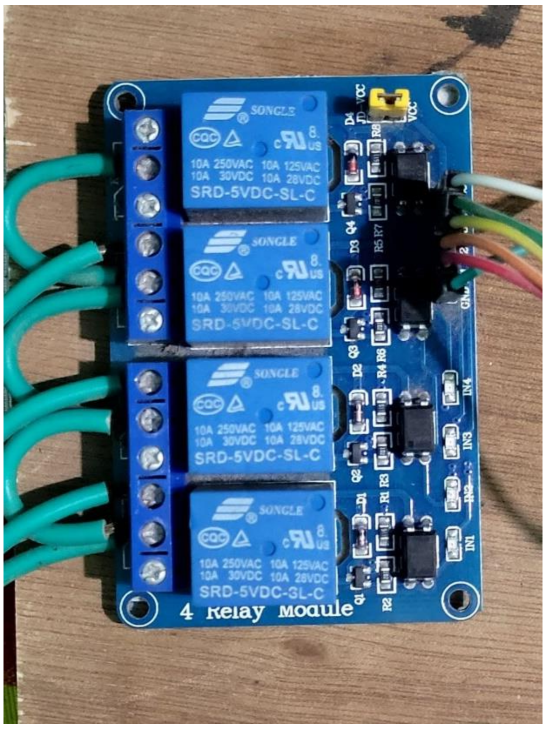


Fig 8.3

Here we are using a 4 channel, 5V relay module. Down here is the DHT 11 sensor for sensing temperature and humidity.

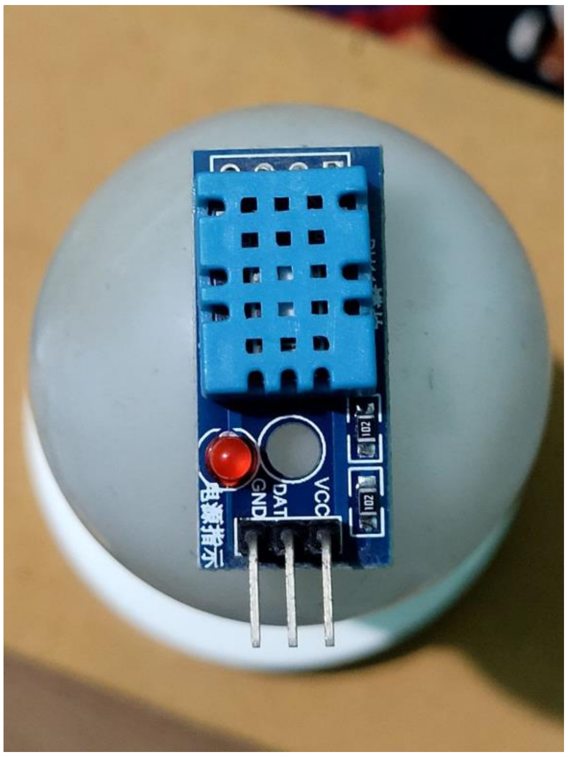


Fig 8.4

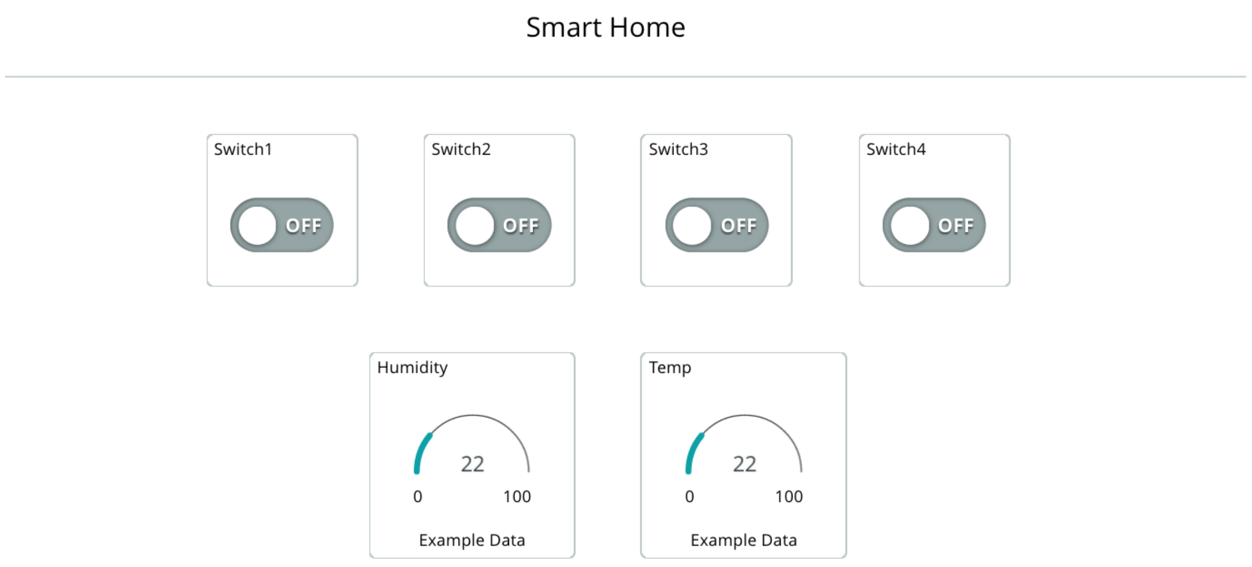


Fig 8.5

Here's the frontend of the project. User can control the relay using the switches. As this is based on cloud, we can control the microcontroller from all around the world, condition being that both NodeMCU and the device are connected to the internet. First picture is when Home Automation is in off stage, and second one is when it's on.

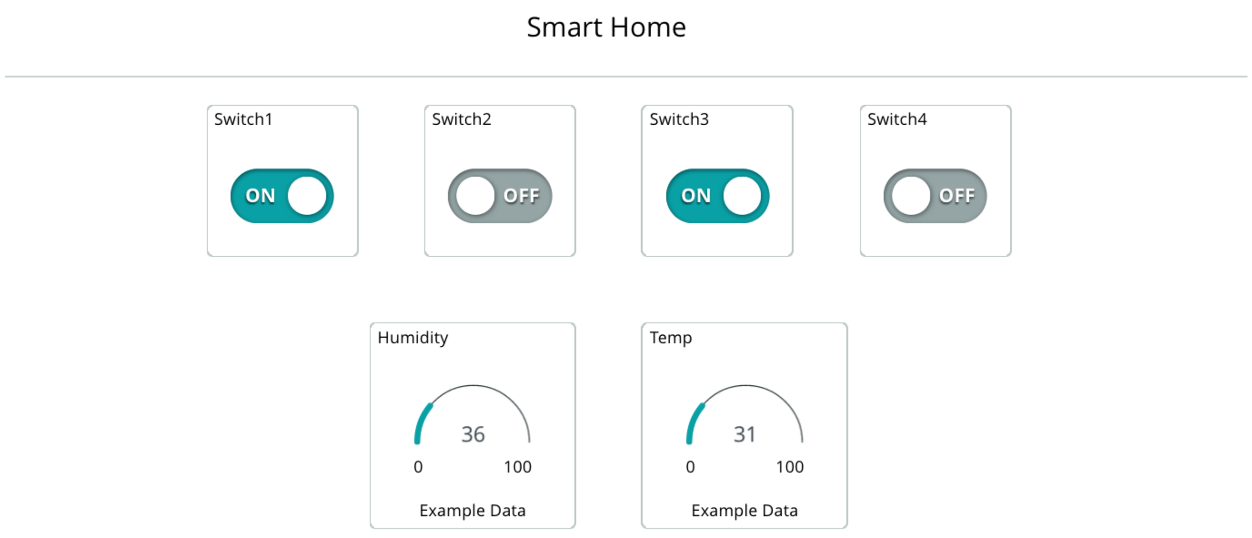


Fig 8.6

CHAPTER – 9

REFERENCES

Chapter – 9

References

[1] For getting the project components: - https://robu.in/

[2] For Arduino IDE: - https://www.arduino.cc/en/software

[3] For Arduino Cloud: https://create.arduino.cc/iot/

[4] For Arduino Cloud Tutorial :- https://www.viralsciencecreativity.com/

[5] For Testing hardware: -
https://www.instructables.com/How-To-Easily-Test-a- Microcontroller-with-a-Logic-/

[6] For Testing software:- https://www.geeksforgeeks.org/software-testing-basics/

[7] For Project Report Formatting: - http://remtechproject.weebly.com/final-report.html

CHAPTER – 10

BRIEF PROFILE

Chapter – 10

Brief Profile

I, Abhiraj Sharma, student of B. Tech Computer Science and Engineering, have created the above project report during my 8th semester. I also did industrial training at Amplify Software Solutions, which is an ERP development company for the partial fulfillment of award for B. Tech Degree. During my industrial training, my role as Microsoft Dynamics Business Central Consultant includes-

- Use of the software Business Central to develop modules for clients.
- Add any custom extension client requests for.
- Experienced working in an agile environment and completed tasks under deadline.
- Able to work with professional repository softwires like Microsoft Azure.
- Show willpower and tenacity to explore all existing data for the specific indicator I am working on.
- Maintain clear and coherent communication, both verbal and written, to understand data needs and report results
- Had direct contact with clients and their employees, for further training them to use the extensions.
- Create clear summaries reporting the progress of my work

I am hereby submitting my project report.

Yours Sincerely,

Abhiraj Sharma