# PROJECT REPORT ON

**HOME AUTOMATION SYSTEM USING ARDUINO AND ESP8266**

**SUBMITTED BY:**

|  |  |
| --- | --- |
| **NAME** | **COLLEGE** |
| Sukalyan Maiti | Calcutta Institute Of Technology |
| Sourav Das | Calcutta Institute Of Technology |
| Riju Guchhait | Calcutta Institute Of Technology |
| Subhajit Hazra | Calcutta Institute Of Technology |
| Souvik Maji | Calcutta Institute Of Technology |
| Tanay Hazra | Calcutta Institute Of Technology |

PROJECT GUIDE AND COORDINATOR: MR. ARITRA CHOUDHURY

(SR. EXECUTIVE TRAINER)



## SKILLECTRON INDIA PVT LTD

156/A/54, Dunlop, Saket Nagar, Bonhooghly Government Colony, Baranagar, Kolkata, West Bengal 700108

**Topics covered in the project report**

1. Acknowledgement
2. Introduction
3. Embedded Systems
4. Internet of Things
5. Arduino Development Platform
6. Arduino Uno microcontroller development board
7. ESP8266 as a standalone Microcontroller and a WiFi module
8. Objective of the Project
9. Block Diagram
10. Circuit Diagram
11. List of Components
12. Flow Chart
13. Program
14. Conclusion
15. Bibliography

ACKNOWLEDGEMENT

During these training of embedded systems and IoT, I learnt a lot on microcontroller architecture and control its behaviour by changing the status of special purpose resisters, they will be able to interface the microcontroller with various sensors and actuators and also learnt about IoT and interfacing Wi-Fi modules with microcontrollers and how this devices communicate via the Internet.

I have to thank Aritra Choudhury Sir for advising me during this project.

Therefore, I am grateful to the people in the skillectron for the chance to make this experience. I especially want to thanks Shuva Bhowmik Sir for giving me the possibility to attend the training for embedded systems and IoT

Further on, I want to thanks the students and interns in the skillectron who made this demanding time joyful but always efficient

INTRODUCTION

Embedded systems in the Internet of Things: The growth potential for the embedded industry is enormous. And the path forward is becoming clearer every day. It’s time that we start building IoT systems, and provide value to our customers. The Internet of Things (IoT) is no longer a fanciful vision. It is very much with us, in everything from factory automation to on-demand entertainment. IoT software solutions have largely had to be built from scratch with a high degree of customization to specific requirements, which has driven up the cost and complexity of development and deterred many prospective entrants to the market. What have been missing are developer tools that alleviate the costs associated with building the foundational infrastructure—the “plumbing” of their solutions—so they can focus on optimizing the core functionality and bring solutions to market more quickly with less cost. Benison Technologies is addressing these challenges with new solutions that have the potential to expand the market for IoT by reducing the cost and complexity of development.

***EMBEDDED SYSTEM***

## Embedded System

As its name suggests, Embedded means something that is attached to another thing. An embedded system can be thought of as a computer hardware system having software embedded in it. An embedded system can be an independent system or it can be a part of a large system. An embedded system is a microcontroller or microprocessor based system which is designed to perform a specific task. For example, a fire alarm is an embedded system; it will sense only smoke.

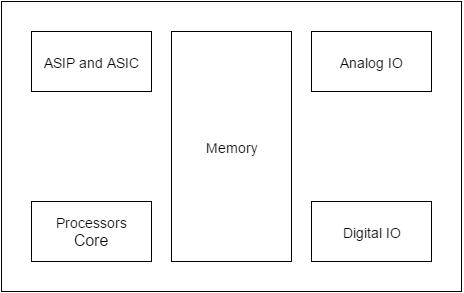
An embedded system has three components −

* It has hardware.
* It has application software.
* It has Real Time Operating system (RTOS) that supervises the application software and provide mechanism to let the processor run a process as per scheduling by following a plan to control the latencies. RTOS defines the way the system works. It sets the rules during the execution of application program. A small scale embedded system may not have RTOS.

So we can define an embedded system as a Microcontroller based, software driven, reliable, real-time control system.

Characteristics of an Embedded System

* **Single-functioned** − An embedded system usually performs a specialized operation and does the same repeatedly. For example: A pager always functions as a pager.
* **Tightly constrained** − All computing systems have constraints on design metrics, but those on an embedded system can be especially tight. Design metrics is a measure of an implementation's features such as its cost, size, power, and performance. It must be of a size to fit on a single chip, must perform fast enough to process data in real time and consume minimum power to extend battery life.
* **Reactive and Real time** − Many embedded systems must continually react to changes in the system's environment and must compute certain results in real time without any delay. Consider an example of a car cruise controller; it continually monitors and reacts to speed and brake sensors. It must compute acceleration or de-accelerations repeatedly within a limited time; a delayed computation can result in failure to control of the car.
* **Microprocessors based** − It must be microprocessor or microcontroller based.
* **Memory** − It must have a memory, as its software usually embeds in ROM. It does not need any secondary memories in the computer.
* **Connected** − It must have connected peripherals to connect input and output devices.
* **HW-SW systems** − Software is used for more features and flexibility. Hardware is used for performance and security



Advantages

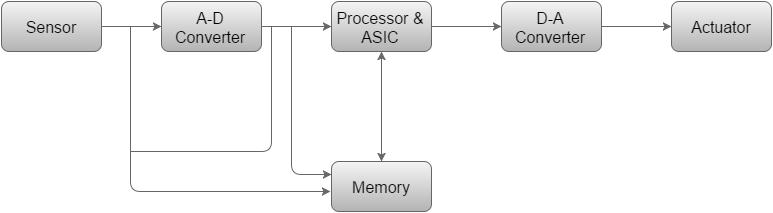
* Easily Customizable
* Low power consumption
* Low cost
* Enhanced performance

Disadvantages

* High development effort
* Larger time to market

Basic Structure of an Embedded System

The following illustration shows the basic structure of an embedded system –



* **Sensor** − It measures the physical quantity and converts it to an electrical signal which can be read by an observer or by any electronic instrument like an A2D converter. A sensor stores the measured quantity to the memory.
* **A-D Converter** − An analog-to-digital converter converts the analog signal sent by the sensor into a digital signal.
* **Processor & ASICs** − Processors process the data to measure the output and store it to the memory.
* **D-A Converter** − A digital-to-analog converter converts the digital data fed by the processor to analog data
* **Actuator** − An actuator compares the output given by the D-A Converter to the actual (expected) output stored in it and stores the approved output.

***INTERNET OF THINGS***

The **Internet of things** (**IoT**) is a system of interrelated computing devices, mechanical and digital machines provided with unique [identifiers](https://en.wikipedia.org/wiki/Identifiers) (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.[[1]](https://en.wikipedia.org/wiki/Internet_of_things#cite_note-Linux_Things-1)[[2]](https://en.wikipedia.org/wiki/Internet_of_things#cite_note-Linux_21OSP-2)[[3]](https://en.wikipedia.org/wiki/Internet_of_things#cite_note-ITU-3)[[4]](https://en.wikipedia.org/wiki/Internet_of_things#cite_note-4)

The definition of the Internet of things has evolved due to the convergence of multiple technologies, real-time [analytics](https://en.wikipedia.org/wiki/Analytics), [machine learning](https://en.wikipedia.org/wiki/Machine_learning), [commodity](https://en.wikipedia.org/wiki/Commodity) [sensors](https://en.wikipedia.org/wiki/Sensors), and [embedded systems](https://en.wikipedia.org/wiki/Embedded_system).[[1]](https://en.wikipedia.org/wiki/Internet_of_things#cite_note-Linux_Things-1) Traditional fields of [embedded systems](https://en.wikipedia.org/wiki/Embedded_system), [wireless sensor networks](https://en.wikipedia.org/wiki/Wireless_sensor_network), [control systems](https://en.wikipedia.org/wiki/Control_system), [automation](https://en.wikipedia.org/wiki/Automation) (including [home](https://en.wikipedia.org/wiki/Home_automation) and [building automation](https://en.wikipedia.org/wiki/Building_automation)), and others all contribute to enabling the Internet of things. In the consumer market, IoT technology is most synonymous with products pertaining to the concept of the "[smart home](https://en.wikipedia.org/wiki/Smart_home_technology)", including devices and [appliances](https://en.wikipedia.org/wiki/Home_appliance) (such as lighting fixtures, [thermostats](https://en.wikipedia.org/wiki/Thermostats), home [security systems](https://en.wikipedia.org/wiki/Security_systems) and cameras, and other home appliances) that support one or more common ecosystems, and can be controlled via devices associated with that ecosystem, such as [smartphones](https://en.wikipedia.org/wiki/Smartphone" \o "Smartphone) and [smart speakers](https://en.wikipedia.org/wiki/Smart_speaker).

There are a number of serious concerns about dangers in the growth of IoT, especially in the areas of [privacy](https://en.wikipedia.org/wiki/Digital_privacy) and [security](https://en.wikipedia.org/wiki/Digital_security), and consequently industry and governmental moves to address these concerns have begun.

***IOT KEY FEATURES :-***

The most important features of IoT include artificial intelligence, connectivity, sensors, active engagement, and small device use. A brief review of these features is given below:

* **AI -** IOT essentially makes virtually anything “smart”, meaning it enhances every aspect of life with the power of data collection, artificial intelligence algorithms, and networks. This can mean something as simple as enhancing your refrigerator and cabinets to detect when milk and your favorite cereal run low, and to then place an order with your preferred grocer.
* **CONNECTIVITY -** New enabling technologies for networking, and specifically IoT networking, mean networks are no longer exclusively tied to major providers. Networks can exist on a

much smaller and cheaper scale while still being practical. IoT creates these small networks between its system devices.

**SENSORS -**IoT loses its distinction without sensors. They act as defining instruments which transform IoT from a standard passive network of devices into an active system capable of real-world integration.

***IOT ADVANTAGES* :-**

* **IMPROVED CUSTOMER ENGAGEMENT -**– Current analytics suffer from blind-spots and significant flaws in accuracy; and as noted, engagement remains passive. IoT completely transforms this to achieve richer and more effective engagement with audiences.
* **TECHNOLOGY OPTIMIZATION -** The same technologies and data which improve the customer experience also improve device use, and aid in more potent improvements to technology. IoT unlocks a world of critical functional and field data.
* **REDUCED WASTE -** IoT makes areas of improvement clear. Current analytics give us superficial insight, but IoT provides real-world information leading to more effective management of resources.

***IOT DISADVANTAGES :-***

* **SECURITY -**IOT makes areas of improvement clear. Current analytics give us superficial insight, but IoT provides real-world information leading to more effective management of resources.
* **PRIVACY -** The sophistication of IoT provides substantial personal data in extreme detail without the user's active participation.
* **FLEXIBILITY -** Many are concerned about the flexibility of an IoT system to integrate easily with another. They worry about finding themselves with several conflicting or locked systems.





**ARDUINO DEVELOPMENT PLATFORM**

* **INTRODUCTION**:-

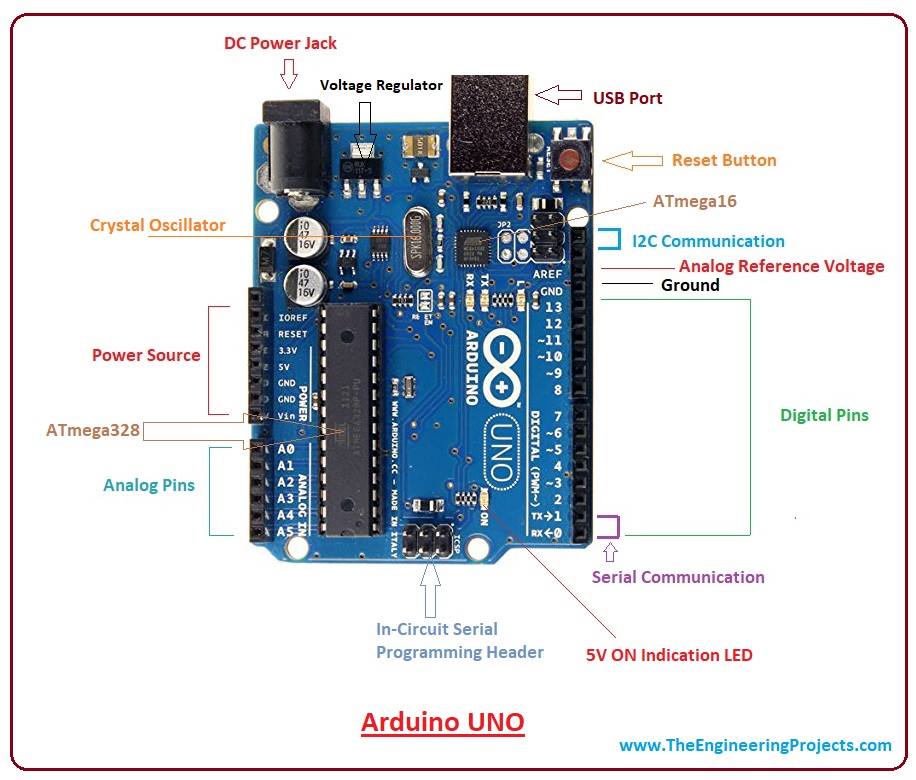
Nowadays the control engineering study usually relies on the available simulation packages and the virtual laboratories. Both of them have their irreplaceable position in the educational process. The Arduino platform is a powerful open source control tool used in control engineering to implement control laws in flexible way. This platform consists of integrated development environment and hardware part used to interact with the physical world via its inputs and outputs. This paper aims to give an overview about Arduino platform include both software and hardware parts. Thereafter, two applications using this platform in power electronics and robotics field are presented. The first application deals with the implementation of space vector pulse width modulation using Arduino UNO board. The second application is an implementation of control law of Planar Cable-Direct-Driven Robot using Arduino Mega board.

* **DESIGN OF ARDUINO DEVELOPMENT PLATFORM:-**

 The microcontrollers were designed in such a way as to obtain a small chip size to reduce costs and to include memory slots and IO (Input / Output)interfaces on the chip: the digital control system consists of UCP (Universal Computer Protocol -microprocessor core), data and program memory, interfaces and peripheral controllers. Today, both passionate people and professionals around the world use it to design and implement electronic projects, from prototypes to complex systems (Kermadi et al., 2015). Arduino consists of a microcontroller and an electronic board, it can connect various sensor types, such as sensors: alcohol, of fire, liquid petroleum gas, for the detection of carbon monoxide; accelerometer; gravity; temperature; humidity; sound, vibration, distance estimation (Doroftei et al., 2018;Akyildiz et al., 2002).

* **APPLY AND EXAMPAL**:-

 In order to determine what can be done with the help of Arduino and the sensors mentioned above, we list some project examples: an alcohol sensor that can be connected to a PC-integrated Arduino which gives a warning before letting the user post on Facebook based on the amount of alcohol in the user's expired air; an autonomous robot that can avoid obstacles; a robot that can be controlled by Bluetooth, using your mobile phone or laptop; a device for painting eggs, for checking temperature, humidity and atmospheric pressure (temperature sensor humidity sensor, atmospheric pressure sensor, and Ethernet card can be used to transmit the data collected from the environment to Google Docs within 10 seconds). Also a robotic hand can be constructed based on a glove with bending sensors and servomotors; a drone, that can capture information on the temperature and wind speed of the environment it is in; a system built with Arduino, which can monitor the electricity that is consumed in a particular room or home; an access control system based on RFID with Twitter notificatification



* **OTHER INFORMATION**:-

A partir de la Fig.1 It is the first Arduino board based on a 32-bit ARM core microcontroller [10]. It contains 54 digital input/output pins (of which 12 can be used as PWM outputs), 12 analog inputs, 4 UARTs (hardware serial ports), a 84 MHz clock, an USB OTG capable connection, 2 DAC (digital to analog), 2 TWI, a power jack, an SPI header, a JTAG header, a reset button and an erase button [11] [12]. From table IV, we notice that the state of each switch is symmetrical with respect to the half sampling period with unique rising or falling edge, so we will exploit this feature to implement the SVM algorithm using the PWM built-in module of the DUE, which will be configured to center aligned mode

**ARDUINO UNO MICROCONTROLLER DEVELOPMENT BOARD**

## The **Arduino Uno** is one kind of microcontroller board based on ATmega328, and Uno is an Italian term which means one. Arduino Uno is named for marking the upcoming release of microcontroller board namely **Arduino Uno Board 1.0**. This board includes digital I/O pins-14, a power jack, analogi/ps-6, ceramic resonator-A16 MHz, a USB connection, an RST button, and an ICSP header. All these can support **the microcontroller** for further operation by connecting this board to the computer. The power supply of this board can be done with the help of an AC to DC adapter, a USB cable, otherwise a battery.

## **What is Arduino Uno ATmega328?**

The **ATmega328** is one kind of single-chip microcontroller formed with Atmel within the **megaAVR family**. The architecture of this Arduino Uno is a customized Harvard architecture with 8 bit [**RISC processor**](https://www.elprocus.com/difference-between-risc-and-cisc-architecture/) core. [**Other boards of Arduino**](https://www.elprocus.com/different-types-of-arduino-boards/)**Uno** include Arduino Pro Mini, Arduino Nano, Arduino Due, Arduino Mega, and Arduino Leonardo.

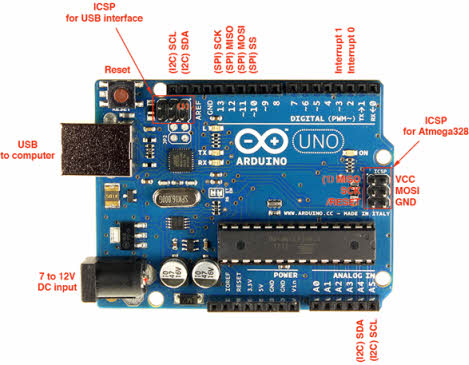
**Features of Arduino Uno Board**

The **features of Arduino Uno ATmega328** includes the following.

* The operating voltage is 5V
* The recommended input voltage will range from 7v to 12V
* The input voltage ranges from 6v to 20V
* Digital input/output pins are 14
* Analog i/p pins are 6
* DC Current for each input/output pin is 40 mA
* DC Current for 3.3V Pin is 50 mA
* Flash Memory is 32 KB
* SRAM is 2 KB
* EEPROM is 1 KB
* CLK Speed is 16 MHz

## **Arduino Uno Pin Diagram**

The Arduino Uno board can be built with power pins, analog pins, ATmegs328, ICSP header, Reset button, [power LED](https://www.elprocus.com/solar-powered-led-street-light-control-circuit/), digital pins, test led 13, TX/RX pins, USB interface, an external [power supply](https://www.elprocus.com/difference-between-single-phase-and-three-phase-ac-power-supply/). The **Arduino UNO board description** is discussed below.



**Power Supply**

The **Arduino Uno power supply** can be done with the help of a USB cable or an external power supply. The external power supplies mainly include AC to DC adapter otherwise a battery. The adapter can be connected to the Arduino Uno by plugging into the power jack of the Arduino board. Similarly, [**the battery**](https://www.elprocus.com/an-overview-of-bio-battery-working-principle-types-applications/)leads can be connected to the Vin pin and the GND pin of the POWER connector. The suggested voltage range will be 7 volts to 12 volts.

**Pin1 (TX) & Pin0 (RX) (Serial):** This pin is used to transmit & receive TTL serial data, and these are connected to the ATmega8U2 USB to TTL Serial chip equivalent pins.

**Pin 2 & Pin 3 (External Interrupts):** External pins can be connected to activate an interrupt over a low value, change in value.

**Pins 3, 5, 6, 9, 10, & 11 (PWM):** This pin gives 8-bit PWM o/p by the function of analogWrite().

**SPI Pins (Pin-10 (SS), Pin-11 (MOSI), Pin-12 (MISO), Pin-13 (SCK):** These pins maintain SPI-communication, even though offered by the fundamental hardware, is not presently included within the Arduino language.

**Pin-13(LED):** The inbuilt LED can be connected to pin-13 (digital pin). As the HIGH-value pin, the light emitting diode is activated, whenever the pin is LOW.

**Pin-4 (SDA) & Pin-5 (SCL) (I2C):** It supports TWI-communication with the help of the Wire library.

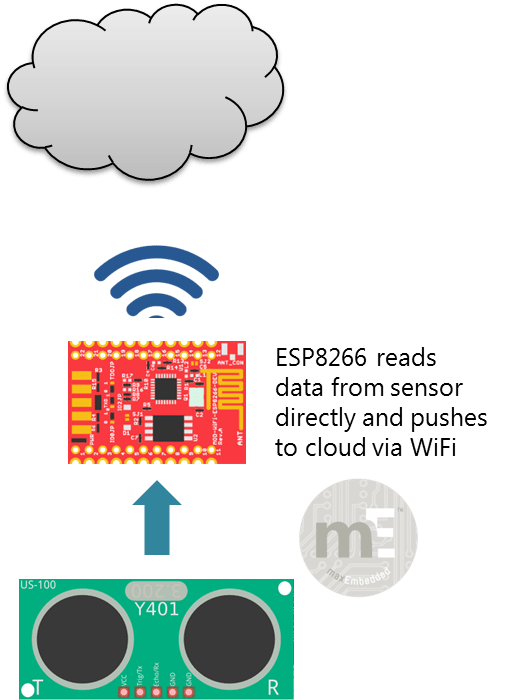
**AREF (Reference Voltage):** The reference voltage is for the analogi/ps with analogReference().

**Reset Pin:** This pin is used for reset (RST) the microcontroller.

**ESP8266 AS A STANDALONE MICROCONTROLLER AND A WIFI MODULE**

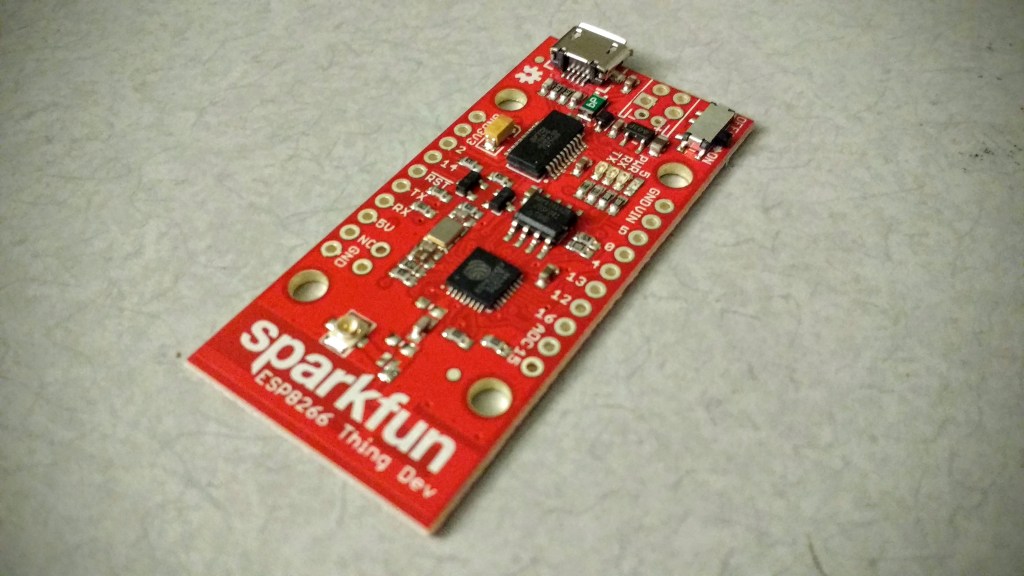
* **ESP8266 as a Standalone Microcontroller** :-

you could get rid of the Arduino completely and replace it with the ESP8266 module and run your application code on it. Now this does has limitations – the ESP8266 doesn’t has as many peripherals as the Arduino, so it can only be used for smaller applications. But it works perfectly fine when all it needs to do is to send data from a bunch of sensors to the internet. You can also run a web server on the ESP8266 that can listen to incoming connections and serve web pages. **We will be using ESP8266 in standalone mode for this post.** Once you get an idea of how to program it, you can use it the way you like, doesn’t really matter.



# The Sparkfun ESP8266 Thing :-

But first you need the module! You can either get the cheap $2 module directly, or get a development board offering more functionality like [NodeMCU](http://nodemcu.com/index_en.html" \t "_blank), [Adafruit HUZZAH](https://www.adafruit.com/products/2471" \t "_blank), or [Sparkfun Thing](https://www.sparkfun.com/products/13231" \t "_blank). **We will be using SparFun ESP8266 Thing board for this post**, but you should be able to get any generic ESP8266 module to work with little changes.

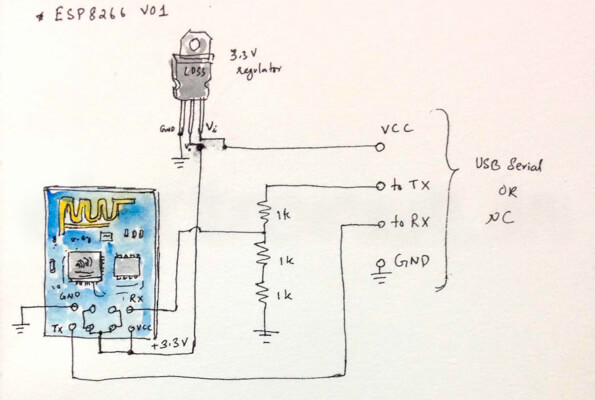


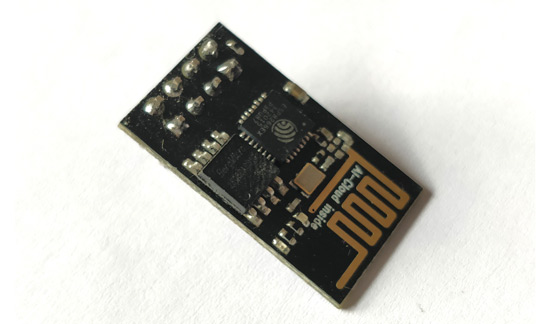
### The ESP8266 WiFi Module :-

The ESP8266 is a WiFi module that costs less than 5 USD. This makes putting your sensors on the net actually feasible. (Hooking up the $75 Arduino Yun to each of your sensors – not no feasible.) There’s a lot of excitement about this sensor on the Internet currently, and people have done an amazing job deciphering the obscure command structure of this device that comes from China. There seems to be three ways of using this module, in order of increasing complexity:

1. Sending it *AT* commands from a computer via an USB to serial adapter. This is mostly useful for testing and setup.
2. Interfacing with an Arduino or any other microcontoller and using this board as a peripheral.
3. Programming the module directly and use its GPIO pins to talk to your sensors, eliminating the need for a second controller.

### Setting up the ESP8266 :-

The first thing you want to do with ESP8266 (as with any aliens) is to establish communication. For this, you hook up a USB to TTL adapter to the module, and talk to it using a serial port terminal application like [CoolTerm](http://freeware.the-meiers.org/" \t "_blank). One thing to be careful about when you hook up this module is to remember that this module operates at 3.3 V – even the serial lines should not exceed this voltage. So here is how I hooked up my ESP8266:

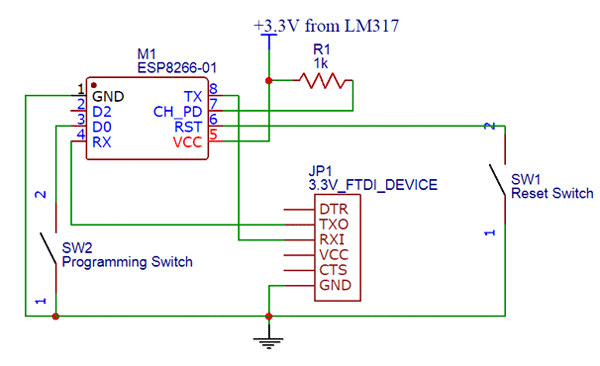
In the above circuit, you can see that I used a 3.3 V regulator to power the board, and a resistor dividor on the RX line to keep the voltages on spec. The sessions below use CoolTerm. My board worked with a baud rate of 9600, since its firmware was already upgraded to 0.9.2.2. You may need to try other baud rates –115200, for instance. 

### ESP8266 Pin Configuration :-

* Low cost, compact and powerful Wi-Fi Module
* Power Supply: +3.3V only
* Current Consumption: 100mA
* I/O Voltage:  3.6V (max)
* I/O source current: 12mA (max)
* Built-in low power 32-bit MCU @ 80MHz
* 512kB Flash Memory
* Can be used as Station or Access Point or both combined
* Supports Deep sleep (<10uA)
* Supports serial communication hence compatible with many development platform like Arduino
* Can be programmed using Arduino IDE or AT-commands or Lua Script

### How to use the ESP8266 Module :-

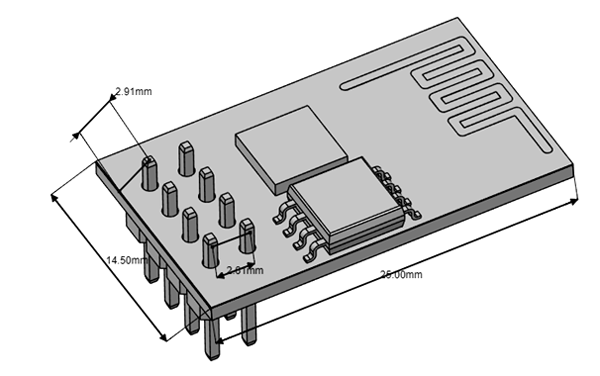
* There are so many methods and IDEs available to with ESP modules, but the most commonly used on is the Arduino IDE. So let us discuss only about that further below.
* The **ESP8266 module** works with 3.3V only, anything more than 3.7V would kill the module hence be cautions with your circuits. The best way to program an **ESP-01** is by using the FTDI board that supports 3.3V programming. If you don’t have one it is recommended to buy one or for time being you can also use an Arduino board. One commonly problem that every one faces with ESP-01 is the powering up problem. The module is a bit power hungry while programming and hence you can power it with a 3.3V pin on Arduino or just use a potential divider. So it is important to make a small voltage regulator for 3.31v that could supply a minimum of 500mA. One recommended regulator is the [LM317](https://components101.com/lm317-pinout-equivalent-datasheet) which could handle the job easily. A **simplified circuit diagram for using the ESP8266-01 module** is given below



### **Applications :-**

* IOT Projects
* Access Point Portals
* Wireless Data logging
* Smart Home Automation
* Learn basics of networking
* Portable Electronics
* Smart bulbs and Sockets

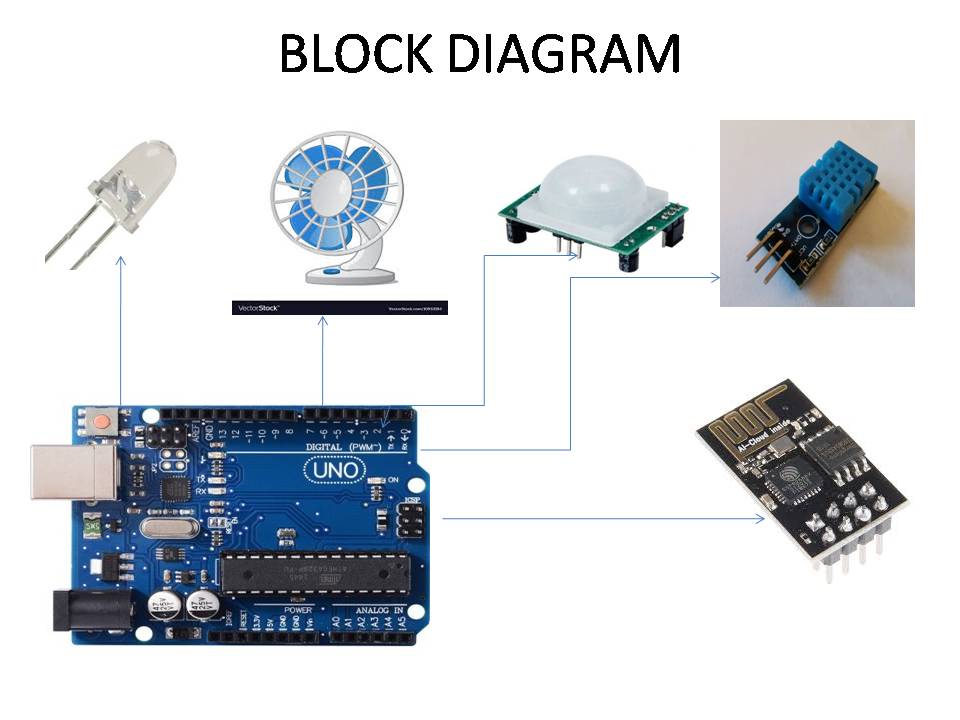
### 2D – Model :-

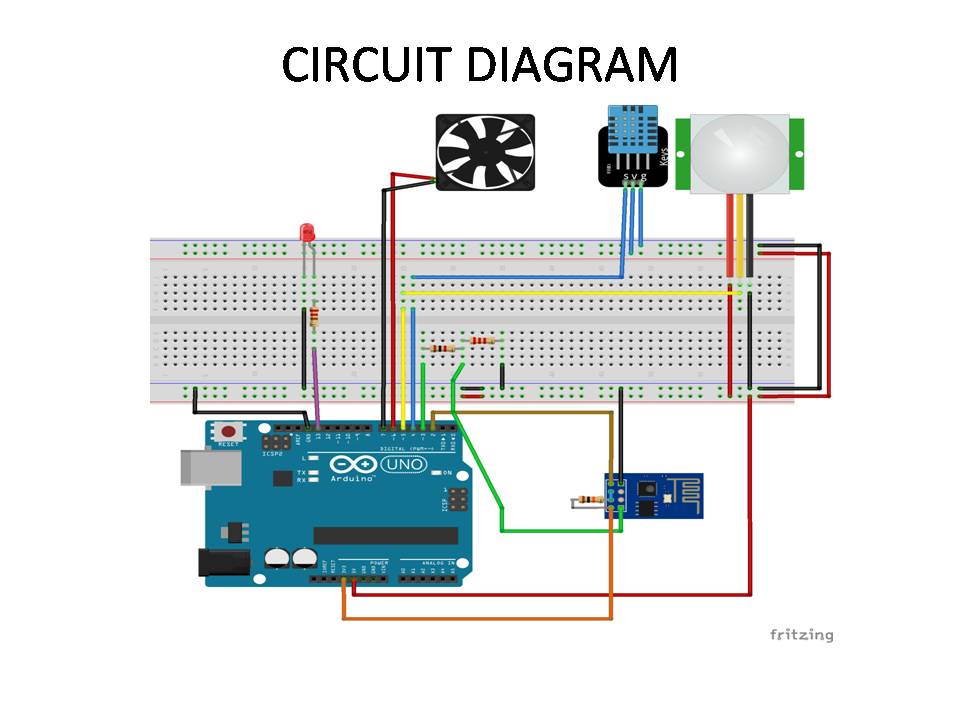


***OBJECTIVE OF THE PROJECT***

To introduce students a set of advanced topics in **embedded IOT** and lead them to understand research in network . To develop comprehensive approach towards building small low cost **embedded IOT system**.To understand fundamentals of security in **IOT** . To learn to implement secure infrastructure for **IOT .**

* To understand fundamentals of IOT and EMBEDDED system including essence, basic design strategy and process modelling.
* To introduce students a set of advanced topics in embedded IOT and lead them to understand research in network .
* To develop comprehensive approach towards building small low cost EMBEDDED IOT SYSTEM .
* To understand fundamentals of security in IOT .
* TO learn to implement secure infrastructure for IOT .
* To learn real world application scenarios of IOT along with its societal and economic impact using case studies.





**List of Components**

1.Bread board

2.Arduino UNO board

3. PIR sensors

4 . DHT11 sensors

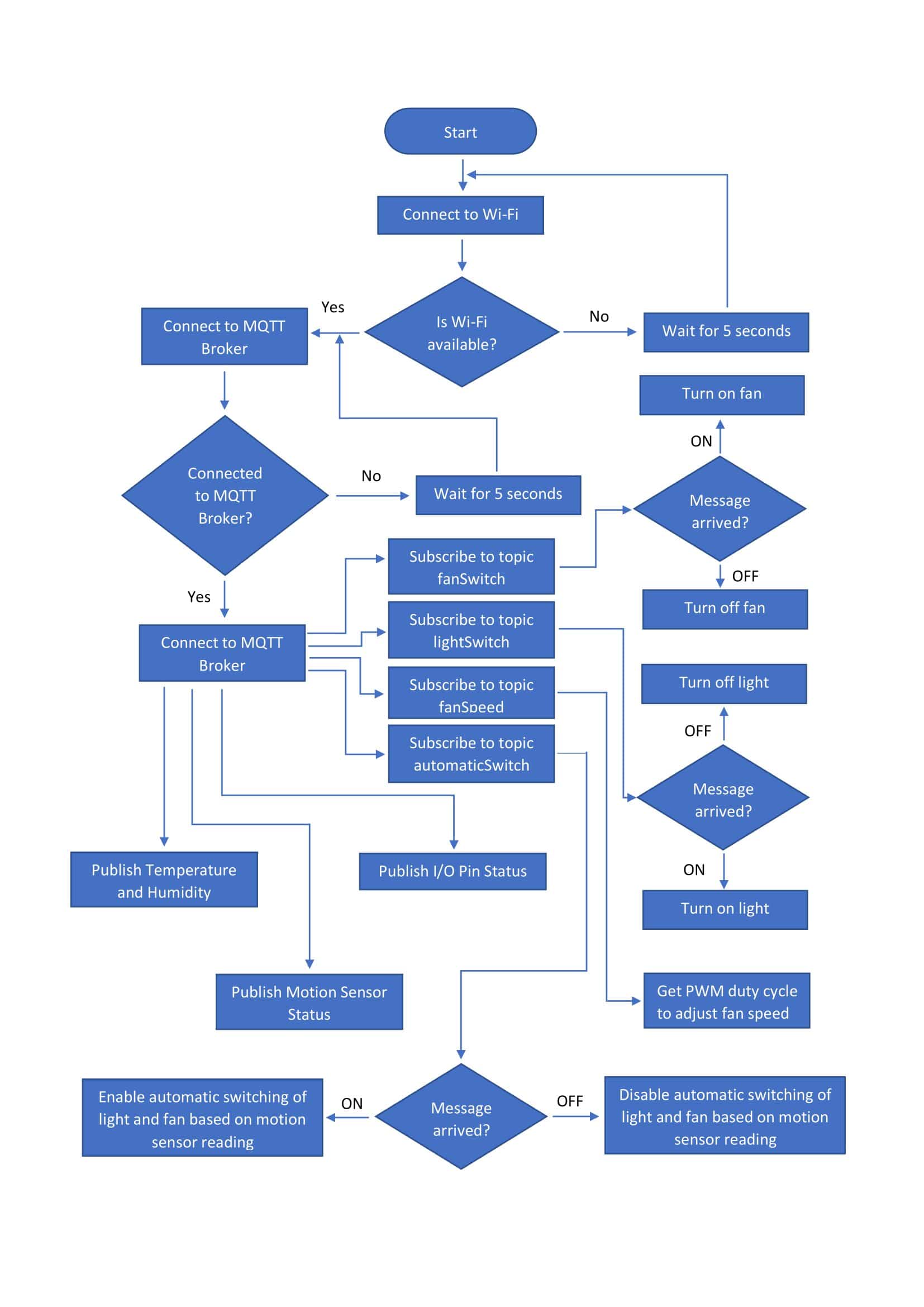
5. LED light

6. Fan

7. ESP8266

8. Wire

9.Resistor



**MQTT PROGRAMME**

//Adding DHT library for DHT11 sensor

#include <DHT.h>

//Adding WiFiEspClient library for communicating with ESP8266 WiFi module

#include <WiFiEspClient.h>

//Adding PubSubClient library to connect with MQTT broker

#include <PubSubClient.h>

//Adding SoftwareSerial library to enable software serial communication with ESP8266 WiFi module

#include "SoftwareSerial.h"

//Adding WiFi credentials to connect to the internet using ESP8266 module

#define WIFI\_AP "IoTHub"

#define WIFI\_PASSWORD "qwertyuiop"

//Adding MQTT broker address

const char\* mqtt\_server = "broker.hivemq.com";

// Setting DHT on pin 4

#define DHTPIN 4

#define DHTTYPE DHT11

// Creating an instance of DHT

DHT dht(DHTPIN, DHTTYPE);

// Creating an instance of WiFiEspClient

WiFiEspClient espClient;

// Creating an instance of PubSubClient and passing espClient as parameter to connect with MQTT broker

PubSubClient client(espClient);

// Creating an instance of SoftwareSerial on pin 2 and 3

SoftwareSerial soft(2, 3); // RX, TX

//Initializing few flags

int status = WL\_IDLE\_STATUS;

unsigned long lastSend = 0;

unsigned long lastCheck = 0;

const int BUILTIN\_LED = 13;

const int FANP = 6;

const int FANN = 7;

const int pirPin = 5; // PIR Out pin

int pirStatus = 0; // PIR status

bool detected = false;

bool automatic = false;

//Setting up WiFi module

void setup\_wifi() {

// initialize serial for ESP module

soft.begin(9600);

// initialize ESP module

WiFi.init(&soft);

// check for the presence of the shield

if (WiFi.status() == WL\_NO\_SHIELD) {

Serial.println("WiFi shield not present");

// don't continue

while(true);

}

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

// attempt to connect to WiFi network

while( status != WL\_CONNECTED){

Serial.print("Attempting to connect to WPA SSID: ");

Serial.println(WIFI\_AP);

// Connect to WPA/WPA2 network

status = WiFi.begin(WIFI\_AP, WIFI\_PASSWORD);

delay(500);

}

Serial.println("Connected to AP");

}//The callback function is triggered when a new message is arrived to a subscribed topic

void callback(char\* topic, byte\* payload, unsigned int length) {

if (strcmp(topic,"test1/ls") == 0){

if (strncmp((const char\*)payload, "ON", 2) == 0) {

digitalWrite(BUILTIN\_LED, HIGH);

Serial.println("LED ON");

}

if (strncmp((const char\*)payload, "OFF", 3) == 0) {

digitalWrite(BUILTIN\_LED, LOW);

Serial.println("LED OFF");

}

}

if (strcmp(topic,"test1/fs") == 0){

if (strncmp((const char\*)payload, "ON", 2) == 0) {

digitalWrite(FANN, LOW);

Serial.println("FAN ON");

}

if (strncmp((const char\*)payload, "OFF", 3) == 0) {

digitalWrite(FANN, HIGH);

Serial.println("FAN OFF");

}

}

if (strcmp(topic,"test1/auto") == 0){

if (strncmp((const char\*)payload, "1", 1) == 0) {

automatic = true;

Serial.print("Automatic: ");

Serial.println(automatic);

}

if (strncmp((const char\*)payload, "0", 1) == 0) {

automatic = false;

Serial.print("Automatic: ");

Serial.println(automatic);

}

}

if (strcmp(topic,"test1/fanSpeed")==0){

payload[length] = '\0';

String s = String((char\*)payload);

int i = s.toInt();

analogWrite(FANP, i);

}

}

// When MQTT broker is disconnected, the reconnect function is called

void reconnect(){

// Loop until we're reconnected

while (!client.connected()) {

Serial.print("Attempting MQTT connection...");

// Create a random client ID

String clientId = "ArduinoClient-";

clientId += String(random(0xffff), HEX);

// Attempt to connect

if (client.connect(clientId.c\_str())) {

Serial.println("connected");

// Once connected, publish an announcement...

client.publish("test1", "reconnected");

// ... and resubscribe

client.subscribe("test1/fs");

client.subscribe("test1/ls");

client.subscribe("test1/fanSpeed");

client.subscribe("test1/auto");

} else {

Serial.print("failed, rc=");

Serial.print(client.state());

Serial.println(" try again in 5 seconds");

// Wait 5 seconds before retrying

delay(5000);

}

}

}

//The getAndSendTemperatureAndHumidityData reads data from DHT sensor and sends it to MQTT broker on subtopic weather

void getAndSendTemperatureAndHumidityData()

{

float h = dht.readHumidity();

float t = dht.readTemperature();

if (isnan(h) || isnan(t)) { //not a number

Serial.println("Failed to read from DHT sensor!");

return;

}

String temperature = String(t);

String humidity = String(h);

// Prepare a JSON payload string JavaScript Object Notation -> Key Value pair

String payload = "{";

payload += "\"t\":";

payload += temperature;

payload += ",";

payload += "\"h\":";

payload += humidity;

payload += "}";

// Send payload

char weather[50];

payload.toCharArray(weather, 50);

client.publish("test1/weather", weather);

Serial.println(weather);

}

//The checkPIRStatus detects motion inside room and sends the information to pir subtopic

void checkPIRStatus(){

pirStatus = digitalRead(pirPin);

if (pirStatus == 1) {

detected = true;

lastCheck = millis();

}

//Wait for 15 minutes before setting the detected flag to false

else if( millis() - lastCheck >= 15\*60\*1000){

detected = false;

lastCheck = millis();

}

client.publish("test1/pir", detected?"1":"0");

if(!detected){

digitalWrite(FANN, HIGH);

digitalWrite(BUILTIN\_LED, LOW);

}else if(automatic && detected){

digitalWrite(FANN, LOW);

digitalWrite(BUILTIN\_LED, HIGH);

}

}

//Get the switch status of light and fan pins on arduino and send information on switchState subtopic

void getSwitchState(){

int ls = digitalRead(BUILTIN\_LED);

int fs = digitalRead(FANN);

String light = String(ls);

String fan = String(fs);

String payload = "{";

payload += "\"l\":";

payload += light;

payload += ",";

payload += "\"f\":";

payload += fan;

payload += ",";

payload += "\"a\":";

payload += automatic;

payload += "}";

char switchState[20];

payload.toCharArray(switchState, 20);

client.publish("test1/switchState", switchState);

}

//Setup the input/output devices to the arduino microcontroller

void setup(){

pinMode(BUILTIN\_LED, OUTPUT);

pinMode(FANP, OUTPUT);

pinMode(FANN, OUTPUT);

pinMode(pirPin, INPUT);

analogWrite(FANP, 255);

digitalWrite(FANN, HIGH);

dht.begin();

Serial.begin(115200);

setup\_wifi();

client.setServer(mqtt\_server, 1883);

client.setCallback(callback);

}

//Loops through the code repeatedly

void loop(){

//reconnect function is called when client is disconnected from broker

if (!client.connected()) {

reconnect();

}

delay(40);

//client.loop function continuously checks for new messages on the subscribed topics

client.loop();

//Send the data after every 5 seconds

if ( millis() - lastSend > 5000 ) {

getAndSendTemperatureAndHumidityData();

checkPIRStatus();

getSwitchState();

lastSend = millis();

}

}

**MQTT OVER WEBSOCKETS**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>MQTT Over Websocket</title>

<!-- CSS Styling for webpage -->

<style>

body{

background-color: rgb(157, 187, 184);

}

.switch {

position: relative;

display: inline-block;

width: 60px;

height: 34px;

bottom: 5px;

left: 10px;

}

.switch input {

opacity: 0;

width: 0;

height: 0;

}

.slider {

position: absolute;

cursor: pointer;

top: 0;

left: 0;

right: 0;

bottom: 0;

background-color: rgb(240, 35, 35);

-webkit-transition: .4s;

transition: .4s;

}

.slider:before {

position: absolute;

content: "";

height: 26px;

width: 26px;

left: 4px;

bottom: 4px;

background-color: white;

-webkit-transition: .4s;

transition: .4s;

}

input:checked + .slider {

background-color: #2196F3;

}

input:focus + .slider {

box-shadow: 0 0 1px #2196F3;

}

input:checked + .slider:before {

-webkit-transform: translateX(26px);

-ms-transform: translateX(26px);

transform: translateX(26px);

}

/\* Rounded sliders \*/

.slider.round {

border-radius: 34px;

}

.slider.round:before {

border-radius: 50%;

}

</style>

</head>

<body>

<!-- Including the Paho JavaScript client library for MQTT over websockets -->

<script src="https://cdnjs.cloudflare.com/ajax/libs/paho-mqtt/1.0.1/mqttws31.js" type="text/javascript"></script>

<script>

//Using the HiveMQ public Broker, with a random client Id

var client = new Paho.MQTT.Client("broker.hivemq.com", 8000, "myclientid\_" + parseInt(Math.random() \* 100, 10));

//Gets called if the websocket/mqtt connection gets disconnected for any reason

client.onConnectionLost = function (responseObject) {

//Alert user that the connection was disconnected

alert("connection lost: " + responseObject.errorMessage);

};

//Gets called whenever you receive a message for your subscriptions

client.onMessageArrived = function (message) {

//Check for messages from the DHT11 sensor and display them on webpage

if(message.destinationName == "test1/weather"){

var res = JSON.parse(message.payloadString);

document.getElementById("weather").innerHTML = '<span>Temperature: ' + res.t + '&deg;C</span><br/>Humidity: ' + res.h + '%</span><br/>';

}

//Check for messages from the PIR sensor for motion detected inside room

if(message.destinationName == "test1/pir"){

document.getElementById("pir").innerHTML = '<span>PIR Status: '+ (message.payloadString=="1"?'Motion detected inside room</span><br/>':'No motion detected for last 30 seconds</span><br/>');

}

//Check for the status of the light, fan and automatic switch send from the Arduino Uno

if(message.destinationName == "test1/switchState"){

var switchState = JSON.parse(message.payloadString);

document.getElementById("lightSwitch").checked = (switchState.l == 1) ? true : false;

document.getElementById("fanSwitch").checked = (switchState.f == 0) ? true : false;

document.getElementById("autoSwitch").checked = (switchState.a == true) ? true : false;

}

};

//Connect Options

var options = {

timeout: 3,

//Gets Called if the connection has sucessfully been established

onSuccess: function () {

//Alert the user that the websocket connection was successful

alert("Connected");

},

//Gets Called if the connection could not be established

onFailure: function (message) {

//Alert the user with error messages for disconnection from the broker

alert("Connection failed: " + message.errorMessage);

}

};

//Creates a new Paho.MQTT.Message Object and sends it to the HiveMQ MQTT Broker

function publish(payload, topic, qos) {

//Publish messages to the MQTT broker

var message = new Paho.MQTT.Message(payload);

message.destinationName = topic;

message.qos = qos;

client.send(message);

}

</script>

<center>

<!-- Button to connect to the MQTT Broker -->

<button onclick="client.connect(options);">Connect</button>

<!-- Button to subscribe to all messages in the 'test1' and it's subtopics with quality of service as 0 -->

<button onclick="client.subscribe('test1/#', {qos: 0}); alert('Subscribed');">Subscribe</button>

<!-- Button to disconnect from the MQTT Broker and close the websocket connection -->

<button onclick="client.disconnect();">Disconnect</button>

<br>

<hr>

<br>

<!-- Heading 3 for displaying Temperature and Humidity -->

<h3 id="weather"></h3>

<!-- Heading 3 for displaying the status of PIR motion sensor -->

<h3 id="pir"></h3>

<!-- A slider switch for switching the Light -->

<h3>Light switch:<label class="switch">

<input type="checkbox" id="lightSwitch">

<span class="slider round"></span>

</label>

</h3>

<!-- A slider switch for switching the Fan -->

<h3>Fan switch:<label class="switch">

<input type="checkbox" id="fanSwitch">

<span class="slider round"></span>

</label>

</h3>

<!-- A range slider switch for controlling fan speed using PWM -->

<h3>Fan Speed:

<input type="range" min="0" max="255" value="128" id="fanSpeed"

onchange="showVal(this.value)"

><span id="valBox"></span>

</h3>

<!-- A slider switch for switching between Manual and Automatic operation -->

<h3>Automatic switching:<label class="switch">

<input type="checkbox" id="autoSwitch">

<span class="slider round"></span>

</label>

</h3>

</center>

<script>

// Assigning variables to store the references of the three slider switches

var checkboxLight = document.getElementById("lightSwitch");

var checkboxFan = document.getElementById("fanSwitch");

var checkboxAuto = document.getElementById("autoSwitch");

//Listening for change in the state of the slider switches

//and publish the state on their respective sub topics

checkboxLight.addEventListener( 'change', function() {

this.checked?publish('ON','test1/ls',0):publish('OFF','test1/ls',0);

});

checkboxFan.addEventListener('change', function(){

this.checked?publish('ON','test1/fs',0):publish('OFF','test1/fs',0);

});

checkboxAuto.addEventListener('change', function(){

this.checked?publish('1','test1/auto',0):publish('0','test1/auto',0);

});

//Publish the PWM value of the range slider switch to the broker

function showVal(newVal){

document.getElementById("valBox").innerHTML=newVal;

publish(newVal,'test1/fanSpeed',0);

}

</script>

</body>

</html>

**CONCLUSION**

Just as the Internet of Things presents businesses in all industries with opportunities for innovation, it also presents tremendous opportunities for developers of embedded systems and IoT solutions. Developers who can meet the demand for reliable, secure, and readily manageable IoT solutions, quickly and at a fair cost, are the ones who stand to benefit. Benison Technologies IoT-optimized solutions, backed by industry-leading expertise in embedded technology, give developers the competitive edge needed to capitalize on opportunities in this rapidly growing market

**BIBLIOGRAPHY**

* **Contents:-**

1: The Internet of Things

2: Security of the Internet of Things

3: Encryption and the Internet of Things

4: Data Ownership and Collection in the Internet of Things

5: Consumer Convenience and the Internet of Things

**1: The Internet of Things:-** As IoT begins to make its mark on the tech industry and revolutionize not only consumer goods but virtually every aspect of human life (author suggests things from cars to street lights to seaports will eventually be “smart”), the importance of IoT, or IoE, lies, as the author points out, in the sensors and cloud computing inherent in the technology. While smart devices have the capacity to “talk to” or communicate with other smart devices (M2M), they also have the capacity to gather and monitor data (vis-à-vis sensors) and analyze that data in real-time (vis-àvis cloud-based software applications). Below is a poignant example provided by the author of the importance of monitoring and analyzing data in real-time through the novelty of a “smart” infrastructure

**2: Security of the Internet of Things:-** A panel discussion geared at finding a balance between security and privacy at Disrupt New York 2016 tackled important topics such as strategies for securing consumer data and risks associated with an expansion of the IoT. There is a trend in the private sector, especially with messaging companies such as Whatsapp, of increasingly viewing the government as an oppositional force. It is clear that data must be protected, and some experts suggest that one way of doing it is simply not to store the information at all – a zero knowledge model. The contention between privacy and security has increased dramatically in the past few years given the large shift in volume and type of data being put online. This shift has incentivized hackers to search for and exploit an evergrowing list of vulnerabilities. Nate Cardozo, a senior staff attorney at the Electronic Frontier Foundation, highlights the human risk factor in securing the IoT and worries about how the lack of knowledge or consensus in securing data could impact embedded systems, such as medical devices.

**3: Encryption and the Internet of Things:-** This article discusses the issues that arise in law enforcement with the increasing use of end-to-end encryption in the IoT, specifically in messaging devices. The authors compare the U.S. and Germany’s encryption debates, outlining the differences between the two in terms of how each country has approached the situation. In the U.S. there is a war between Washington and Silicon Valley regarding the installation of “back doors” that would allow the government access to encrypted information in exceptional circumstances. James Comey, director of the FBI, has referred to the dangers “going dark” could pose for law enforcement. Civil liberties advocates, on the other hand, argue that installing such backdoors could set a worrying precedent in terms of data privacy. The German government, unlike that of the U.S., has publicly declared its support for more widespread and effective encryption. However, policymakers in Berlin are beginning to worry about the practical realities of more encryption, especially in light of the recent terrorist attacks in France and Belgium. The author suggests that Germany should take an “encrypted world” as a given and foster discussion in this context, focusing on policy that would allow law enforcement to fulfill its duties in a world of ubiquitous encryption. To do so, they must address legal frameworks, technical capabilities and staffing

**4: Data Ownership and Collection in the Internet of Things:-**

There is no exact measure to reference when it comes to data ownership in the IoT. Previous legislation surrounding user data from apps, such as Facebook or Twitter, are outdated and are becoming increasingly irrelevant when it comes to smart, sensor-based devices in the IoT. Within the industry, the topic of data ownership is highly debated and not as clear-cut as many consumers might like. In most instances, it is not clear who owns the data being collected from IoT devices. Often, an app will require users to agree to hand over data being collected by the software company, for storage in a collection database. This is the classic example of targeted-advertising on platforms such as Facebook, Twitter, etc. However, many connected devices do and will gather more types of data: 1) internal data – data provided to vendor from consumer site detailing how product is being used; 2) external data – data relevant to customer and/or broader market. An emerging line of thought, as expressed by Eric Harper of ABB, is that the first type of data should be owned by the vendor, since it is used in enhancing the product/service the device delivers. The second type of data, the external, should be used by the consumer however they please; the consumer should own this data. Data ownership in the IoT is a constantly evolving process that needs to be revisited and addressed often as the industry matures.

**5: Consumer Convenience and the Internet of Things:-** Rival chip makers Intel and Qualcomm have agreed to consolidate their respective standards groups for IoT protocols into one entity, Open Connectivity Foundation (OCF). Industry leaders hope this new group will streamline IoT devices, ensuring that all connected devices will be able to “talk” to each other seamlessly, regardless of the chip manufacturer or operating system of the device. To ensure this, the standards group will define industry regulations regarding communication protocols, software, hardware, and licensing agreements. Microsoft has also played a key role in the formation of OCF; according to a spokesperson, “Despite the opportunity and promise of IoT to connect devices in the home or in businesses, competition between various open standards and closed company protocols have slowed adoption and innovation.” Microsoft has a large stake in the future of IoT: Windows 10 will be released for low-powered devices in the near future and will implement the IoT standards defined by the OCF on all of its devices.