A PROJECT REPORT



ON

***SMART MEDICINE REMINDER FOR ELDERLY PEOPLE USING IoT***

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS OF THE DEGREE OF

B. Tech

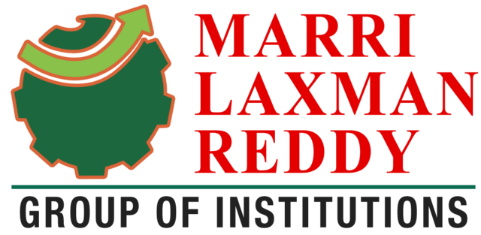
BY

P. Bhavana Roll No: 187Y1A04C6

B. Nikhitha Roll No: 187Y1A0490

Shaik KhajaAyaz Roll No: 187Y1A0411

UNDER THE ESTEEMED GUIDANCE OF



Marri Laxman Reddy Institute of Technology and Management

2019

**ACKNOWLEDGEMENT**

A dissertation work, which integrated skill full and hard work need a great deal of guidance and helping hand. I would like to add a few heartfelt words for the people who have directly or indirectly in preparing in this dissertation work in numerous ways.

My heartfelt thanks to **Marri Laxman Reddy**, chairman of our college for providing me an opportunity to care out this project work form this college.

My sincere thanks to **Dr. K Venkateswara Reddy**, the principal of our college for providing me the required facilities to complete my project.

With profound sense of gratitude and regards, I acknowledge with great pleasure the Guidance and support extended by, **Mr. K Abdul Basith & Mr. K Naga Bhushanam**, heads of the Computer Science &Electronics and Communication Engineering departments and internal guide, **Mr. B Prasad** for his timely suggestion, logical thought that have greatly contributed in bringing out the project in a systematic way. He has been inculcating innovative skills and technicalities at all stages at performing this project.

My sincere thanks to all the teaching and non- teaching staff of our department for their constant co-operation extended to me to complete the project successfully. My thanks to all my classmates and friends for their constant help and support during my project work.

My heartfelt gratitude to my parents and my family for their constant encouragement throughout the work carried out to prepare this project report.

SUBMITTED BY

BATCH NO- 05

P. BHAVANA 187Y1A04C6

B. NIKHITHA 187Y1A0490

SHAIK KHAJA AYAZ 187Y1A0411

INDEX

Introduction……………………………………………… 6-7

Literature Survey…………………………………………………………… 8

Block Diagram……………………………………………………………….. 9

Hardware

\*Node MCU………………………………………………………….. 11

\*RTC Clock Module……………………………………………….. 12-13

\*Leds…………………………………………………………………….. 14

\*Buzzer…………………….……………………………………………. 14

\*Connecting Wires….…………………………………………….. 15

Software

\*Arduino IDE…………………………………………………………. 16

\*IBM Cloud Services………………………………………………. 17

\*Node Red…………………………………………………………….. 18

\*MIT App inventer-2……………………………………………… 18-19

Alarm System…………………………………………………………………. 20

Discription of the project.………………………………………………. 20-35

Advantages…………………………………………………………………….. 35

Conclusion……………………………………………………………………… 36

**ABSTRACT**

In modern society, busy life has made people forget many things in day to day life. The elderly people and the people victims of chronicle diseases who need to take the medicines timely without missing are suffering from dementia, which is forgetting things in their daily routine. Considering this situation study has been done in this. Paper reviewing the technologies of home health care which are currently used for improving this situation by reminding the scheduled of medicine, remote monitoring and update new medicine Consumption data of patients, which can be done by pre-scriber through IOT.

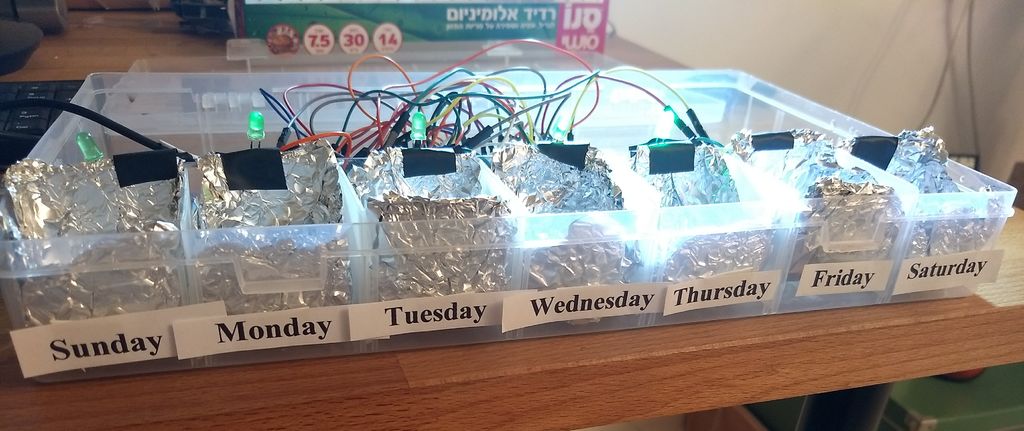
**Key Words:** Smart medicine box, old age patients, permanent diseases, setting up alarm, bright light, sound with buzzer, sensing capability.

**INTRODUCTION:**

In day-to-day life most of the people need to take medicines which was not there in past couple of years and the reason behind this is diseases are increasing in large amount. So sooner or later many people come in contact with these diseases. Some diseases are temporary diseases while many are permanent life threatening diseases. Life threatening diseases gets mixes with the human body in such a way that they can’t leave the body ever and they increases in rapid time. Life span of humans became less because of such diseases and to overcome or to live a better life we need to take medicines regularly and also in large amount. We need to be in advice of Doctor who tells us to take desired pills in desired way so that patients face problems like forgetting pills to take at right time and also when Doctor changes the prescription of medicine patients have to remember the new schedule of medicine. This problem of forgetting to take pills at right time, taking wrong medicines and accidentally taking of expired medicine causes health issues of patient and this leads to suffer from unhealthy life.



Our project is to make Smart medicine reminder box which uses Real time clock. It is compulsory for the patient to take pills from the box at the right time otherwise our systems continues to make large sound until the medicine is taken out from the box. This feature adds life years to the patient and thus this thing is not available in any device which is the necessity for present days.



**Literature Survey:**

In this section, a combination between electronic and mechanical pill boxes or dispensers is presented. It’s been included certain traditional pills organizers, which represents a first step in these developments and allowed us to obtain ideas about design useful patterns in development of this solution. In is presented a pill dispenser which has different prescribed administration schedules. It includes a plurality of pill storage compartments, each of them capable of holding more than one pill.

A current design, shows a device that allows the storing and dispensing of pills and various supplements. In this section, a combination between electronic and mechanical pill boxes or dispensers is presented. It’s been included certain traditional pills organizers, which represents a first step in these developments and allowed us to obtain ideas about design useful patterns in development of this solution.

This device works such as an alarm clock and may work with blister packed pills or alternatively uses an encapsulated compartment to hold and dispense loose pills. Also, it can be connected by wireless to external environments (cellphones, computers).

The dosages are dispensed when an alarm is activated, this device does not use referential diseases, just use dosages per days, and is also not programmable for any schedule. ii) It is a reminder medication product focused on patients, caregivers or medical health professionals. This device locks automatically. It has alarm and text message reminders perceived are to close device by interaction of keeper and is not Independent.

***Block Diagram :-***

IBM Cloud Platform,

Node Red

RTC Clock Module

Node MCU

ESP8266

Mobile App

Leds

Buzzer

For this project the hardware and software used are:

**Hardware:**

* NodeMCU.
* RTC Clock Module.
* Leds.
* Buzzer.
* Connecting Wires.

**Software:**

* Arduino IDE.
* IBM Cloud Services.
* Node Red
* MIT App inventer-2.

**Hardware:**

NodeMCU:-

The Node MCU device kit is a development board with the ESP8266 mounted on it. It also has a USB to Serial convertor chip on board. This removes the need of the FTDI USB to Serial Converter. Also, it has a voltage converter on board for converting the 5V supplied by the USB port to 3.3V input required by the ESP8266. So all you have to do is plug the USB cable from the computer right into the micro USB slot of the Node MCU dev. board, and you can start with your ESP8266 programming / prototyping. The Node MCU provides easier way to program the ESP8266 module.



ESP8266 offers a complete and self-contained Wi-Fi networking solution, allowing it to either host the application or to offload all Wi-Fi networking functions from another application processor. When ESP8266 hosts the application, and when it is the only application processor in the device, it is able to boot up directly from an external flash. It has integrated cache to improve the performance of the system in such applications, and to minimize the memory requirements.802.11 b/g/n/d/e/i/k/r support; Wi-Fi Direct (P2P) support,P2P Discovery, P2P Group Owner mode, P2P Power Management. Dual and single antenna Bluetooth co-existence support with optional simultaneous receive (Wi-Fi/Bluetooth) capability.

RTC Clock Module:-

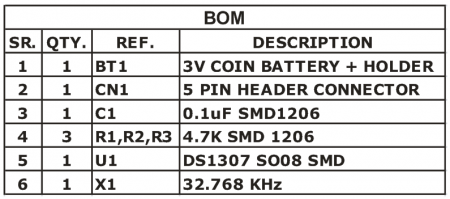
DS1307 (64 x 8, Serial, I2C Real-Time Clock):

The DS1307 serial real-time clock (RTC) is a low-power, full binary-coded decimal (BCD) clock/calendar plus 56 bytes of NV SRAM. Address and data are transferred serially through an I2C, bidirectional bus. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The end of the month date is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24- hour or 12-hour format with AM/PM indicator[2]. The DS1307 has a built-in power-sense circuit that detects power failures and automatically switches to the backup supply. Timekeeping operation continues while the part operates from the backup supply.



SPECIFICATIONS

* Supply 5V DC.
* Completely Manages All Timekeeping Functions.
* Real-Time Clock Counts Seconds, Minutes, Hours, Date of the Month, Month, Day of the Week, and Year with Leap-Year Compensation Valid Up to 2100.
* 56-Byte, Battery-Backed, General-Purpose RAM with Unlimited Writes.
* Programmable Square-Wave Output Signal.
* Simple Serial Port Interfaces to Most Microcontrollers.
* I2C Serial Interface.
* Low Power Operation Extends Battery Backup Run Time.
* Consumes Less than 500nA in Battery-Backup Mode with Oscillator Running.
* Automatic Power-Fail Detect and Switch Circuitry.



Leds :-

Diode LED emitting infrared waves, this component has high reliability and high radiant intensity, its peak wavelength is p=3.7e-5in and 1.00e-7in lead spacing. Its applications are in free air transmission system or infrared applied system.



Buzzer :-

A **buzzer** or **beeper** is an [audio](https://en.wikipedia.org/wiki/Sound) signalling device, which may be [mechanical](https://en.wikipedia.org/wiki/Machine), [electromechanical](https://en.wikipedia.org/wiki/Electromechanics), or [piezoelectric](https://en.wikipedia.org/wiki/Piezoelectricity) (*piezo* for short). Typical uses of buzzers and beepers include [alarm devices](https://en.wikipedia.org/wiki/Alarm_devices), [timers](https://en.wikipedia.org/wiki/Timer), and confirmation of user input such as a mouse click or keystroke.



Connecting Wires :-

A **jump wire** (also known as jumper wire, or jumper) is an [electrical wire](https://en.wikipedia.org/wiki/Electrical_wire), or group of them in a cable, with a connector or pin at each end (or sometimes without them – simply "tinned"), which is normally used to interconnect the components of a [breadboard](https://en.wikipedia.org/wiki/Breadboard) or other prototype or test circuit, internally or with other equipment or components, without soldering.

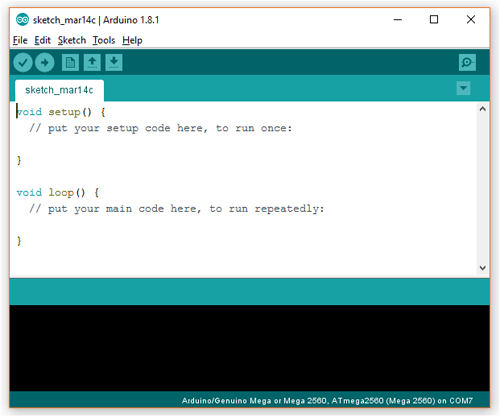


**Software:**-

Arduino IDE :-

The **Arduino integrated development environment (IDE)** is a cross-platform application (for Windows, macOS, Linux) that is written in the programming language Java. It is used to write and upload programs to Arduino compatible boards, but also, with the help of 3rd party cores, other vendor development boards. The source code for the IDE is released under the GNU General Public License, version.

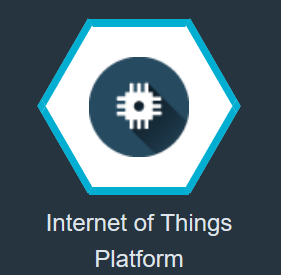
2. The Arduino IDE supports the languages [C](https://en.wikipedia.org/wiki/C_(programming_language)) and [C++](https://en.wikipedia.org/wiki/C%2B%2B) using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub *main()* into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program *avrdude* to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.



IBM Cloud Services :-

An IoT platform is a multi-layer technology that enables straightforward provisioning, management, and automation of connected devices within the Internet of Things universe. It basically connects your hardware, however diverse, to the cloud by using flexible connectivity options, enterprise-grade security mechanisms, and broad data processing powers. For developers, an IoT platform provides a set of ready-to-use features that greatly speed up development of applications for connected devices as well as take care of scalability and cross-device compatibility.

Thus, an IoT platform can be wearing different hats depending on how you look at it. It is commonly referred to as middleware when we talk about how it connects remote devices to user applications (or other devices) and manages all the interactions between the hardware and the application layers. It is also known as a cloud enablement platform or IoT enablement platform to pinpoint its major business value, that is empowering standard devices with cloud-based applications and services. Finally, under the name of the IoT application enablement platform, it shifts the focus to being a key tool for IoT developers.

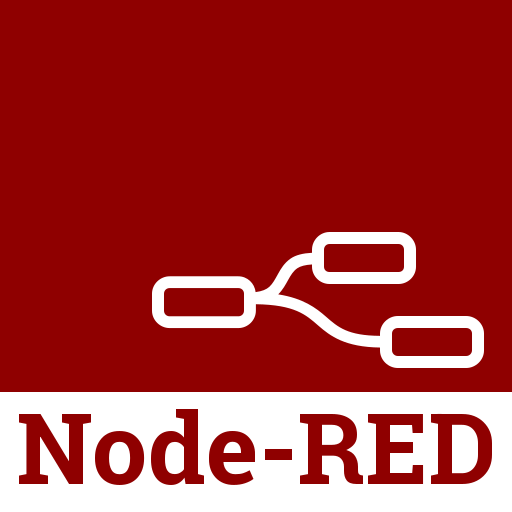


Node Red :-

**Node-RED** is a flow-based development tool for visual programming developed originally by IBM for wiring together hardware devices, APIs and online services as part of the Internet of Things.

Node-RED provides a web browser-based flow editor, which can be used to create JavaScript functions. Elements of applications can be saved or shared for re-use. The runtime is built on Node.js. The flows created in Node-RED are stored using JSON. Since version 0.14 MQTT nodes can make properly configured TLS connections.

In 2016, IBM contributed Node-RED as an open source JS Foundation project.



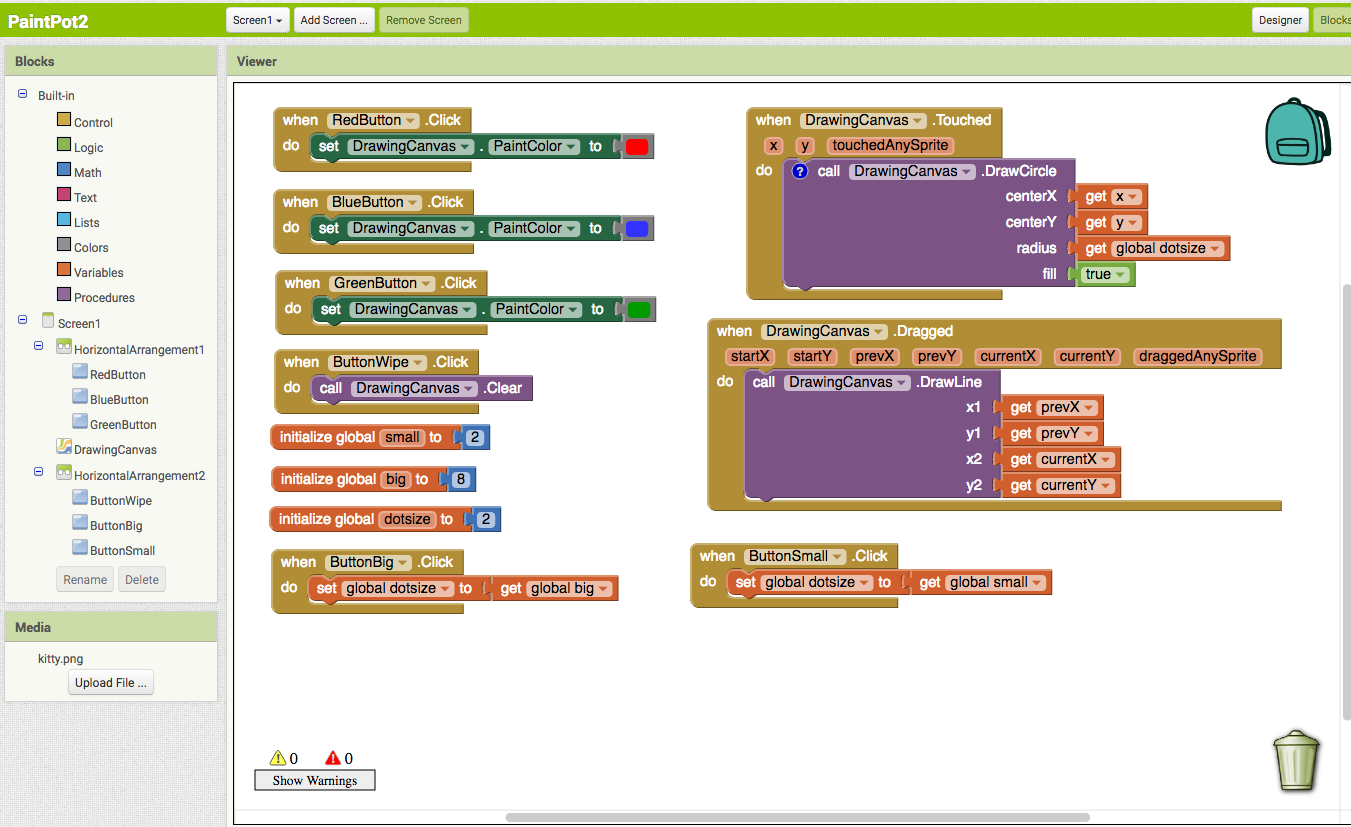
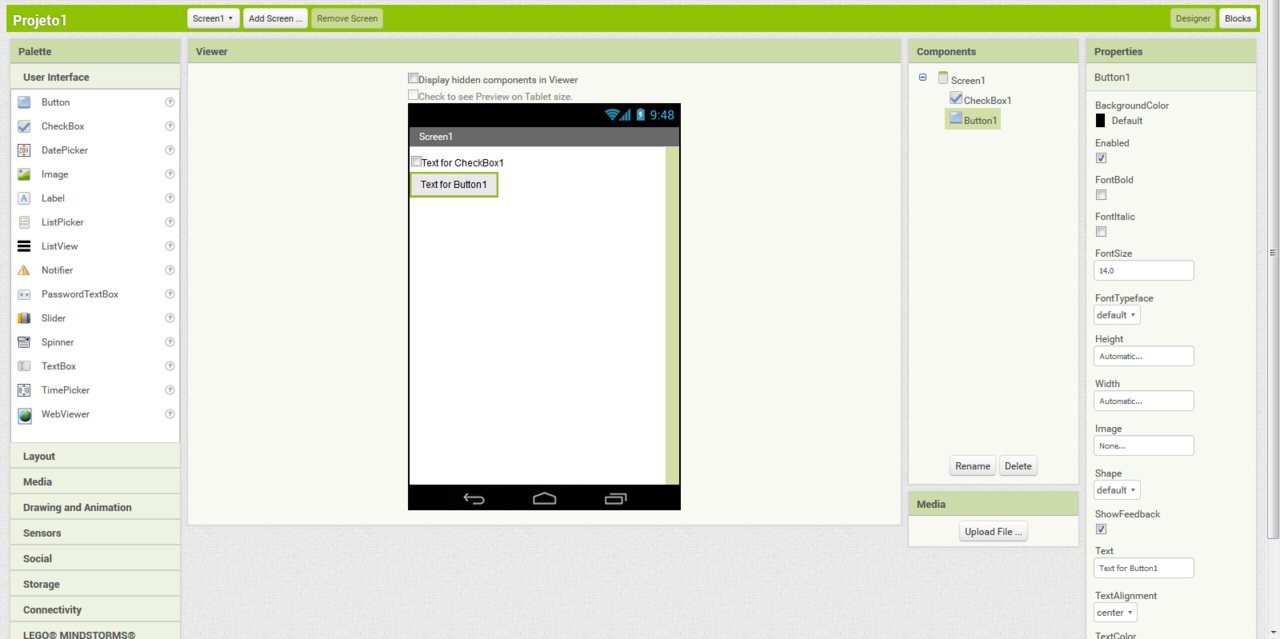
MIT App inventer-2 :-

**App Inventor for Android** is an open-source web application originally provided by Google, and now maintained by the Massachusetts Institute of Technology (MIT), which allows newcomers to computer programming to create software applications for the Android operating system (OS).

It uses a graphical interface very similar to Scratch and the Star Logo TNG user interface, which allows users to drag-and-drop visual objects to create an application that can run on Android devices. In creating App Inventor, Google drew upon significant prior research in educational computing, as well as work done within Google on online development environments.

App Inventor and the projects on which it is based are informed by constructionist learning theories, which emphasizes that programming can be a vehicle for engaging powerful ideas through active learning. As such, it is part of an ongoing movement in computers and education that began with the work of Seymour Papert and the MIT Logo Group in the 1960s and has also manifested itself with Mitchel Resnick's work on Lego Mind storms and Star Logo.

App Inventor also supports the use of cloud data via an experimental FirebaseDB component.



**ALARM SYSTEM:-**

The alarm system consists of a buzzer. The buzzer used in proposed system is piezoelectric buzzer. A buzzer is a mechanical, electromechanical, magnetic, electromagnetic, electro-acoustic or piezoelectric audio signaling device. A piezo electric buzzer can be driven by a noscillating electronic circuit or other audio signal source. A click, beep or ring can indicate that a button has been pressed. The buzzer is also output device and gives a beep when current time meets the set time.

***Procedure:-***

Arduino IDE:

We need to write a program to develop any project. Now first step is to write program in Arduino IDE. The program consists of transmitting and receiving of data from RTC Clock Module. The Node MCU is connected to the system through USB cord and the Node MCU board has to be selected in the arduino IDE to get the output. And we need to select the port(ex-COM3).

RTC Clock module program for getting the time:

#include <Wire.h>

#include <RtcDS1307.h>

RtcDS1307<TwoWire> Rtc(Wire);

void setup ()

{

Serial.begin(57600);

Serial.print("compiled: ");

Serial.print(\_\_DATE\_\_);

Serial.println(\_\_TIME\_\_);

Rtc.Begin();

RtcDateTime compiled = RtcDateTime(\_\_DATE\_\_, \_\_TIME\_\_);

printDateTime(compiled);

Serial.println();

if (!Rtc.IsDateTimeValid())

{

if (Rtc.LastError() != 0)

{

Serial.print("RTC communications error = ");

Serial.println(Rtc.LastError());

}

else

{

Serial.println("RTC lost confidence in the DateTime!");

Rtc.SetDateTime(compiled);

}

}

if (!Rtc.GetIsRunning())

{

Serial.println("RTC was not actively running, starting now");

Rtc.SetIsRunning(true);

}

RtcDateTime now = Rtc.GetDateTime();

if (now < compiled)

{

Serial.println("RTC is older than compile time! (Updating DateTime)");

Rtc.SetDateTime(compiled);

}

else if (now > compiled)

{

Serial.println("RTC is newer than compile time. (this is expected)");

}

else if (now == compiled)

{

Serial.println("RTC is the same as compile time! (not expected but all is fine)");

}

Rtc.SetSquareWavePin(DS1307SquareWaveOut\_Low);

}

void loop ()

{

if (!Rtc.IsDateTimeValid())

{

if (Rtc.LastError() != 0)

{

Serial.print("RTC communications error = ");

Serial.println(Rtc.LastError());

}

else

{

Serial.println("RTC lost confidence in the DateTime!");

}

}

RtcDateTime now = Rtc.GetDateTime();

printDateTime(now);

Serial.println();

delay(5000); // five seconds

}

#define countof(a) (sizeof(a) / sizeof(a[0]))

void printDateTime(const RtcDateTime& dt)

{

char Hou[20];

char Min[20];

snprintf\_P(Hou,countof(Hou),PSTR("%02u"),dt.Hour());

snprintf\_P(Min,countof(Min,PSTR("%02u"),dt.Minute());

Serial.print(Hou);

Serial.print(Min);

}

The RTC time value should be pushed to the node red so the following program has been used for the value pushing.

#include <Wire.h>

#include <RtcDS1307.h>

RtcDS1307<TwoWire> Rtc(Wire);

#include <ESP8266WiFi.h>

#include <PubSubClient.h>

void callback(char\* subtopic, byte\* payload, unsigned int payloadLength);

const char\* ssid = "MLRITM@DGL";

const char\* password = "digi@123";

String data;

#define ORG "fwtttv"

#define DEVICE\_TYPE "Reminder-2019"

#define DEVICE\_ID "nodemcu\_reminder"

#define TOKEN "123456789"

#define led D3

#define buzzer D4

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

char pubtopic[] = "iot-2/evt/reminder/fmt/json";

char subtopic[] = "iot-2/evt/reminder/fmt/json";

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

char token[] = TOKEN;

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

char Hou;

char Min ;

WiFiClient wifiClient;

PubSubClient client(server, 1883,wifiClient);

#define led D3

#define buzzer D4

void setup ()

{

Serial.begin(57600);

Serial.print("compiled: ");

Serial.print(\_\_DATE\_\_);

Serial.println(\_\_TIME\_\_);

Rtc.Begin();

RtcDateTime compiled = RtcDateTime(\_\_DATE\_\_, \_\_TIME\_\_);

printDateTime(compiled);

Serial.println();

pinMode(led,OUTPUT);

pinMode(buzzer,OUTPUT);

wifiConnect();//user define function

mqttConnect();// user define function

if (!Rtc.IsDateTimeValid())

{

if (Rtc.LastError() != 0)

{

Serial.print("RTC communications error = ");

Serial.println(Rtc.LastError());

}

else

{

Serial.println("RTC lost confidence in the DateTime!");

Rtc.SetDateTime(compiled);

}

}

if (!Rtc.GetIsRunning())

{

Serial.println("RTC was not actively running, starting now");

Rtc.SetIsRunning(true);

}

RtcDateTime now = Rtc.GetDateTime();

if (now < compiled)

{

Serial.println("RTC is older than compile time! (Updating DateTime)");

Rtc.SetDateTime(compiled);

}

else if (now > compiled)

{

Serial.println("RTC is newer than compile time. (this is expected)");

}

else if (now == compiled)

{

Serial.println("RTC is the same as compile time! (not expected but all is fine)");

}

Rtc.SetSquareWavePin(DS1307SquareWaveOut\_Low);

}

void loop ()

{

if (!Rtc.IsDateTimeValid())

{

if (Rtc.LastError() != 0)

{

Serial.print("RTC communications error = ");

Serial.println(Rtc.LastError());

}

else

{

Serial.println("RTC lost confidence in the DateTime!");

}

}

RtcDateTime now = Rtc.GetDateTime();

printDateTime(now);

Serial.println();

delay(5000); // five seconds

if (isnan(Hou) || isnan(Min))

{

Serial.println("Failed to read from RTC module!");

delay(1000);

return;

}

if (!client.loop()) {

mqttConnect();

}

delay(100);

PublishData(Hou,Min);

delay(1000);

}

void wifiConnect() {

Serial.print("Connecting to ");

Serial.print(ssid);

WiFi.begin(ssid, password);

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

{

delay(500);

Serial.print(".");

}

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

Serial.println(WiFi.localIP());

}

void mqttConnect()

{

if (!client.connected())

{

Serial.print("Reconnecting MQTT client to ");

Serial.println(server);

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

{

Serial.print(".");

delay(500);

}

initManagedDevice();//user defined function

Serial.println();

}

}

void initManagedDevice()

{

if (client.subscribe(subtopic))

{

Serial.println("subscribe to command OK");

}

else

{

Serial.println("subscribe to command FAILED");

}

}

void callback(char\* subtopic, byte\* payload, unsigned int payloadLength)

{

Serial.print("callback involed for topic: ");

Serial.println(subtopic);

for (int i = 0; i < payloadLength; i++)

{

//Serial.println((char)payload[i]);

data += (char)payload[i];// data = data +(char)payload[i]

}

Serial.println(data);//if user input is light\_on then data = light\_on

if(data == "hour")

{

digitalWrite(led,HIGH);

tone (buzzer,100);

Serial.println("Light is Switched ON");

Serial.println("Buzzer is Switched ON");

}

else if(data != "")

{

digitalWrite(led,LOW);

noTone(buzzer);

Serial.println("Light is Switched OFF");

Serial.println("Buzzer is Switched OFF");

}

data ="";

}

#define countof(a) (sizeof(a) / sizeof(a[0]))

void printDateTime(const RtcDateTime& dt)

{

char Hou[20];

char Min[20];

snprintf\_P(Hou,countof(Hou),PSTR("%02u"),dt.Hour());

snprintf\_P(Min,countof(Min,PSTR("%02u"),dt.Minute());

Serial.print(Hou);

Serial.print(Min);

}

void PublishData(char Hou, char Min){

if (!client.connected()) {

Serial.print("Reconnecting client to ");

Serial.println(server);

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

Serial.print(".");

delay(500);

}

Serial.println();

}

String payload = "{\"u\":{\"hour\":";

payload += Hou;

payload+="," "\"minutes\":";

payload += Min;

payload += "}}";

Serial.print("Sending payload: ");

Serial.println(payload);

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

{

Serial.println("Publish ok");

}

else

{

Serial.println("Publish failed");

}

}

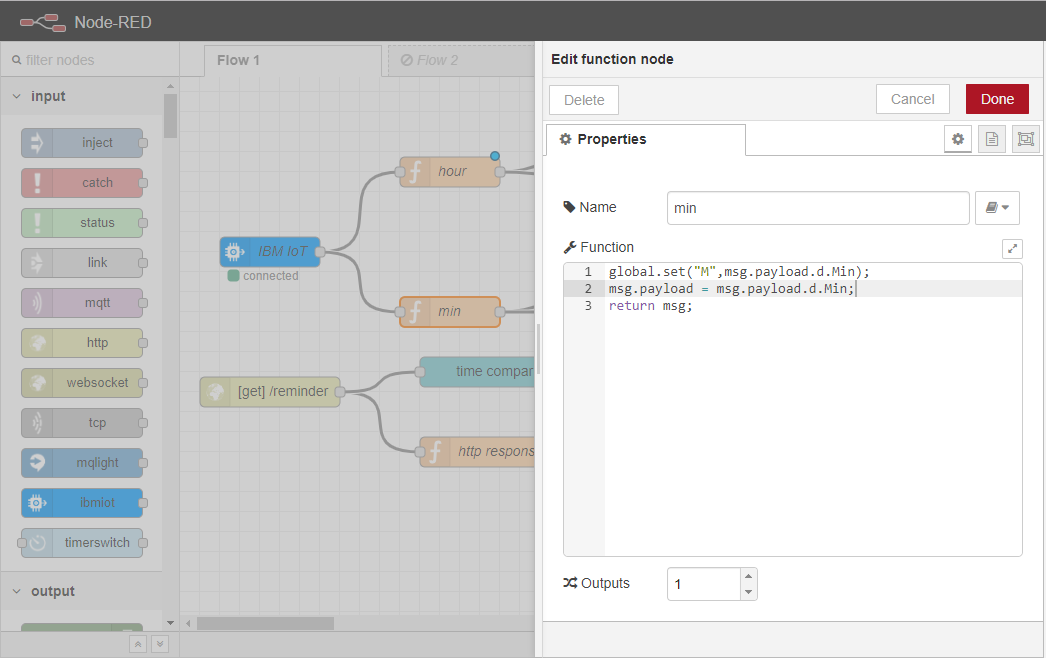
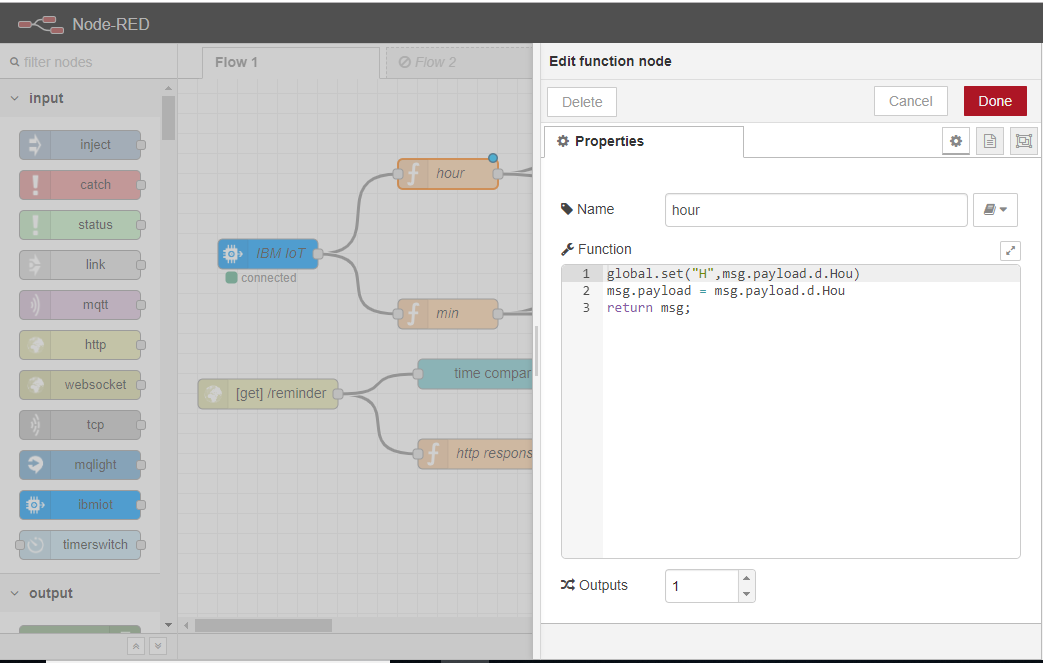
Node-RED:

Node-RED is a flow-based development tool for visual programming developed originally by IBM for wiring together hardware devices, APIs and online services as part of the Internet of Things. Node-RED provides a web browser-based flow editor, which can be used to create JavaScript functions. Elements of applications can be saved or shared for re-use. The runtime is built on Node.js. The flows created in Node-RED are stored using JSON.

We need to create a flow and we need to implement the process in given space. The following steps are used to implement the program.

First, drag the input node Input node that can be used with Watson IoT Platform to receive events sent from devices, receive commands sent to devices, or receive status updates concerning devices or applications. It produces an object called msg and sets msg. payload to be a String containing the payload of the incoming message.

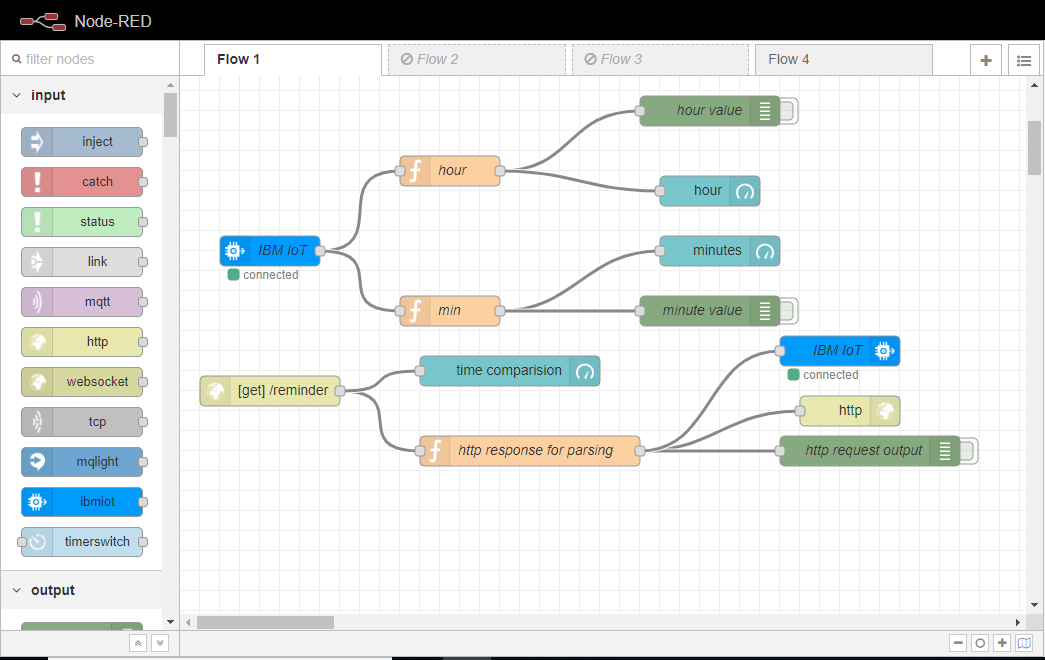
Set the properties of the input node as shown below.



Next drag the Debug node. The debug sidebar provides a structured view of the messages it is sent, making it easier to understand their structure. JavaScript objects and arrays can be collapsed and expanded as required. Buffer objects can be displayed as raw data or as a string if possible.

Function node: A JavaScript function block to run against the messages being received by the node. The messages are passed in as a JavaScript object called msg. By convention it will have a msg. payload property containing the body of the message. The function is expected to return a message object (or multiple message objects), but can choose to return nothing in order to halt a flow.

The overall view of the project will be as follows:



MIT App Inventer :-

App Inventor for Android is an open-source web application originally provided by Google, and now maintained by the Massachusetts Institute of Technology (MIT), which allows newcomers to computer programming to create software applications for the Android operating system (OS). It uses a graphical interface very similar to Scratch and the Star Logo TNG user interface, which allows users to drag-and-drop visual objects to create an application that can run on Android devices. In creating App Inventor, Google drew upon significant prior research in educational computing, as well as work done within Google on online development environments.

Steps:

\* Create new project.

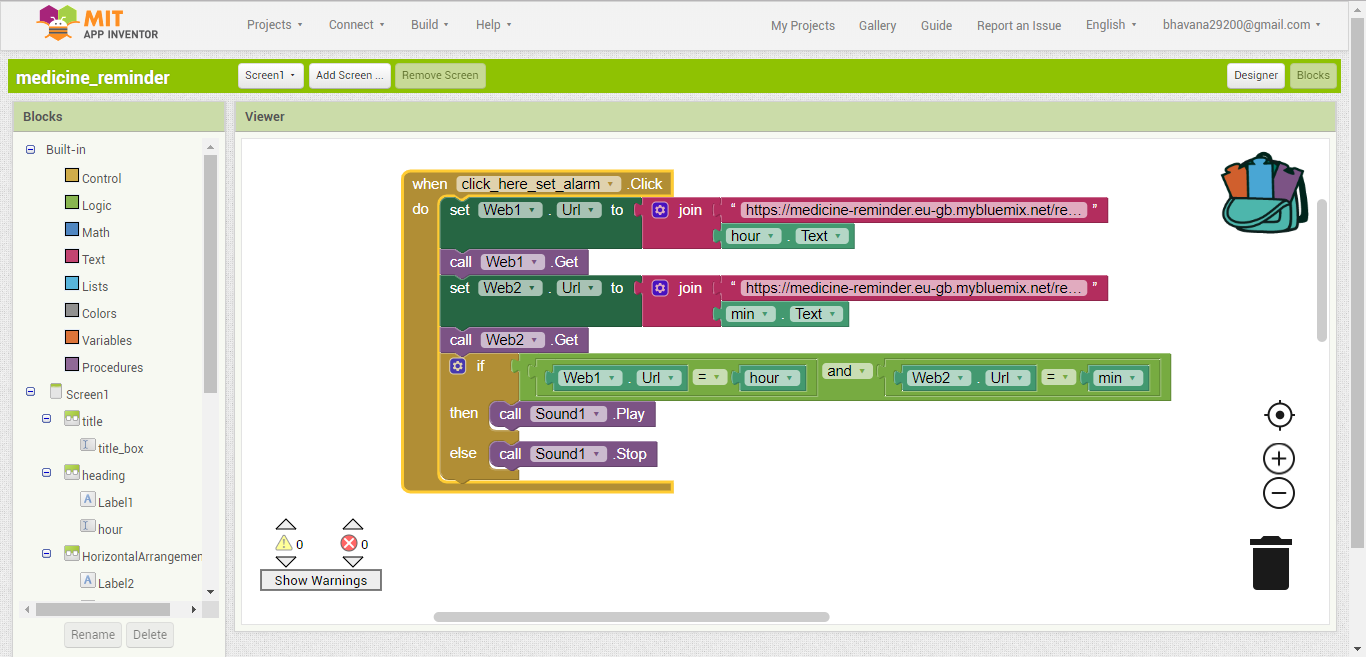
\* Drag the label to the screen and edit the properties.

\* Drag the layout components and in it drag textbox, labels and set its properties.

\* And add buttons as our requirement.

\* In blocks give conditions to each of the component.

\* Add screens as per our requirement.



**Hardware Connections:-**

Connections are between the Node MCU, RTC Clock module, leds and buzzer. The connections are as follows:

Connections between node mcu and rtc clock module

**Node MCU** - **RTC Clock Module**

GND - GND

3V3 - VCC

D2 - SDA

D1 - SCL

Connections between node mcu, leds and buzzer

**Node MCU** - **Leds, Buzzer**

GND - GND

3V3 - VCC

D3 - Led

D4 - Buzzer

**ADVANTAGES:-**

• Monitoring of health statistics Medicine, alarms and medication non-compliance control.

• Emergency and medical management services.

• Wireless identifiable Embedded healthcare systems.

**Conclusion:-**

Older people play an important role in the society. They are part of the priority group of healthcare. Therefore, creating new devices using the emerging technology in order to improve their lives quality is necessary. The creation of alternatives of medication devices looks promising and necessary due to that today only 1 of each 10 people in need have access to such system due to high costs and a lack of awareness, availability, personal trainee, policy and financing. The introduction of such devices in IOT could lead us to a future where important information of patients would be available anytime and anywhere, in order to make a correct treatment and to prevent calamities. Based on open source solutions, a new alternative to remind medicine dosages was raised. The objective of creating a device that allows the organization of several medication schedules was reached. IoT is an important aim pretended in this device, finding a way to keep pillbox connected to Internet and it will help surely to manage better form of treatments in patients, mainly in elderly patients. Scientific validation method used is dedicated to validate electronic equipment and applications, for the future works this method will change and it let us evaluate response between elder patient and keepers with pillbox and this one with the biggest network is Internet.