

A PROJECT REPORT

ON

**SMART WATER MANAGEMENT USING IBM IOT WATSON PLATFORM**

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS OF THE DEGREE OF

B. Tech

BY

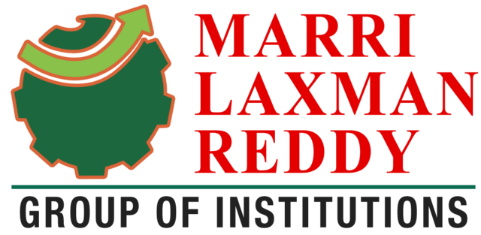
B.PRASHANTH REDDY 177Y1A05F0

G. MAHESHWARI 187Y1A0484

M.ANUSHA 187Y1A04C4

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 Nagabhushanam**, 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- 03

B.PRASHANTH REDDY 177Y1A05F0

G.MAHESHWARI 187Y1A0484

M.ANUSHA 187Y1A04C4

**INDEX**

### ABSTRACT 4

### INTRODUCTION 5

### HARDWARE DEVICES 6-11

### 1.Node MCU

### 2.Ultrasonic Sensor:

### 3.Water Flow Sensor:

4.Servo Motor:

### SOFTWARES 12-15

1.Arduino IDE

2.IBM Cloud

* + IBM IOT Watson Platform
  + Node Red

3.MIT App Inventor 2

DESCRIPTION ABOUT THE PROJECT 16-27

### Software Implementation

### Arduino IDE

### Node-RED

### Mit app inventor

### CONCLUSION 27

### ABSTRACT

### During the past decade, water needs have increased unpredictably in India. Increasing demand of water supply has become a major challenge for the world. Wasteful usage of water, climatic changes and Urbanization has further depleted the resource. Conservation and management of the resource must be given utmost importance.

### In this paper, we present an IoT design for water monitoring and control approach which supports internet based data collection on real time bases. The system addresses new challenges in the water sector -flow rate measuring and the need for a study of the supply of water in order to curb water wastage and encourage its conservation.

### The traditional water metering systems require periodic human intervention for maintenance making it inconvenient and often least effective. For shortcoming of the existing models for a ubiquitous usage of wireless systems for smart quality monitoring and communicate data wirelessly.

### INTRODUCTION

### Water is an important resource for all the livings on the earth. In that, some people are not getting sufficient amount of water because of unequal distribution. We can use this approach so that everyone gets the equal amount of water. It is also used to avoid the wastage of water during the distribution period. In the previous method, the employee will go to that place and open the valve for a particular duration, then again the employee will go to the same place and close the valve, it is waste of time. The proposed system is fully automated. Here human work and time are saved.

### The project Smart Water Management system, as the name says it is all about management of water supply throughout the scale, right from small societies, townships to entire urban infrastructure and also for irrigation water supply management. Firstly, we can check the water levels in the tank using ultrasonic sensors and based on water levels we can start and stop the water supply from the main tank to the individual houses automatically.

### And later at the house level we can check the flow rate of water by using flow sensors and all these data can be sent to cloud and these parameters can be shown on the User Interface which is created in the Node Red.

### From user interface/App, based on flowrate value from the Flow Sensor we provide individuals whether the tap in the room has to be on/off so that they will be able to store the water in home without manual interaction even when they are not present in home.

### HARDWARE DEVICES

### Hardware devices used in this project are:

### NodeMCU

### Ultrasonic Sensor

### Flow Sensor

### Servo Motor

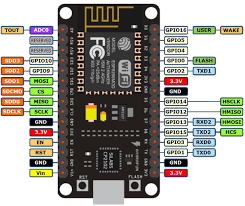
### Node MCU

NodeMCU V3 is an open-source firmware and development kit that plays a vital role in designing your own IoT product using a few Lua script lines. Multiple GPIO pins on the board allow you to connect the board with other peripherals and are capable of generating PWM, I2C, SPI, and UART serial communications. The interface of the module is mainly divided into two parts including both Firmware and Hardware where former runs on the ESP8266 Wi-Fi SoC and later is based on the ESP-12 module.

* NodeMCU V3 comes with a number of GPIO Pins. Following figure shows the Pinout of the board.There is a candid difference between Vin and VU where former is the regulated voltage that may stand somewhere between 7 to 12 V while later is the power voltage for USB that must be kept around 5 V.

**Features:**

* Open-source
* Arduino-like hardware
* Status LED
* MicroUSB port
* Reset/Flash buttons
* Interactive and Programmable
* Low cost
* ESP8266 with inbuilt wifi
* USB to UART converter
* GPIO Pins



### Ultrasonic Sensor:



**Ultrasonic Sensor Pin Configuration**

|  |  |  |
| --- | --- | --- |
| **Pin Number** | **Pin Name** | **Description** |
| 1 | Vcc | The Vcc pin powers the sensor, typically with +5V |
| 2 | Trigger | Trigger pin is an Input pin. This pin has to be kept high for 10us to initialize measurement by sending US wave. |
| 3 | Echo | Echo pin is an Output pin. This pin goes high for a period of time which will be equal to the time taken for the US wave to return back to the sensor. |
| 4 | Ground | This pin is connected to the Ground of the system. |

### HC-SR04 Sensor Features

* Operating voltage: +5V
* Theoretical  Measuring Distance: 2cm to 450cm
* Practical Measuring Distance: 2cm to 80cm
* Accuracy: 3mm
* Measuring angle covered: <15°
* Operating Current: <15mA
* Operating Frequency: 40Hz

### HC-SR04 Ultrasonic Sensor – Working

As shown above the **HC-SR04 Ultrasonic (US) sensor** is a 4 pin module, whose pin names are Vcc, Trigger, Echo and Ground respectively. This sensor is a very popular sensor used in many applications where measuring distance or sensing objects are required. The module has two eyes like projects in the front which forms the Ultrasonic transmitter and Receiver. The sensor works with the simple high school formula that

**Distance = Speed × Time**

The Ultrasonic transmitter transmits an ultrasonic wave, this wave travels in air and when it gets objected by any material it gets reflected back toward the sensor this reflected wave is observed by the Ultrasonic receiver module as shown in the picture below.



Now, to calculate the distance using the above formulae, we should know the Speed and time. Since we are using the Ultrasonic wave we know the universal speed of US wave at room conditions which is 330m/s. The circuitry inbuilt on the module will calculate the time taken for the US wave to come back and turns on the echo pin high for that same particular amount of time, this way we can also know the time taken. Now simply calculate the distance using a microcontroller or microprocessor.

### Water Flow Sensor



**Description**

Water flow sensor consists of a plastic valve body, a water rotor, and a hall-effect sensor. When water flows through the rotor, rotor rolls. Its speed changes with different rate of flow. The hall-effect sensor outputs the corresponding pulse signal. This one is suitable to detect flow in water dispenser or coffee machine. We have a comprehensive line of water flow sensors in different diameters. Check them out to find the one that meets your need most.

**Features**

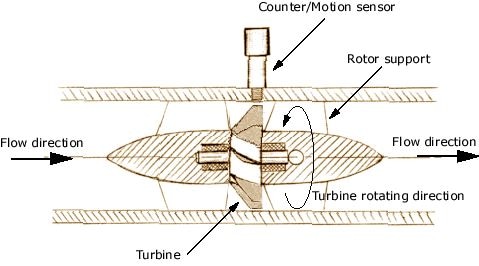
* Compact, Easy to Install
* High Sealing Performance
* High Quality Hall Effect Sensor
* RoHS Compliant

**Specifications**

* Mini. Wokring Voltage: DC 4.5V
* Max. Working Current: 15mA (DC 5V)
* Working Voltage: DC 5V~24V
* Flow Rate Range: 1~30L/min
* Load Capacity: ?10mA (DC 5V)
* Operating Temperature: ?80?
* Liquid Temperature: ?120?
* Operating Humidity: 35%~90%RH
* Water Pressure: ?1.75MPa
* Storage Temperature: -25~+ 80?
* Storage Humidity: 25%~95%RH

### Working Principle:

Flow meters perform flow measurement through turbine rotation with a shunt, propeller, or paddle wheel design. The mechanical types of water flow meters work by measuring the speed of water flowing through the pipe that causes a piston or turbine to rotate. The volumetric flow rate of the water is proportional to the rotational speed of the blades. The disadvantage of mechanical water flow meters for water measurement is that they may clog up when the water is dirty or contain larger particles, which leads to increased maintenance costs. Mechanical water meters also do not work well when the water flow is low.



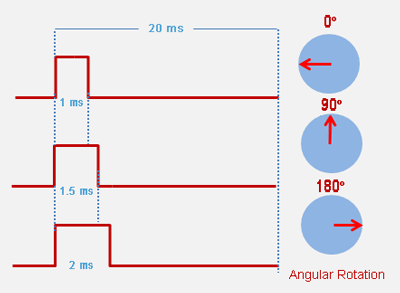
**Servo Motor**



A **servo motor** is an electrical device which can push or rotate an object with great precision. If you want to rotate and object at some specific angles or distance, then you use servo motor. It is just made up of simple motor which run through **servo mechanism**. If motor is used is DC powered then it is called DC servo motor, and if it is AC powered motor then it is called AC servo motor. We can get a very high torque servo motor in a small and light weight packages. Doe to these features they are being used in many applications like toy car, RC helicopters and planes, Robotics, Machine etc.

### Working Principle:

Servo motor works on **PWM (Pulse width modulation)** principle, means its angle of rotation is controlled by the duration of applied pulse to its Control PIN. Basically servo motor is made up of **DC motor which is controlled by a variable resistor (potentiometer) and some gears**. High speed force of DC motor is converted into torque by Gears. We know that WORK= FORCE X DISTANCE, in DC motor Force is less and distance (speed) is high and in Servo, force is High and distance is less. Potentiometer is connected to the output shaft of the Servo, to calculate the angle and stop the DC motor on required angle.



Servo motor can be rotated from 0 to 180 degree, but it can go up to 210 degree, depending on the manufacturing. This degree of rotation can be controlled by applying the **Electrical Pulse** of proper width, to its Control pin. Servo checks the pulse in every 20 milliseconds. Pulse of 1 ms (1 millisecond) width can rotate servo to 0 degree, 1.5ms can rotate to 90 degree (neutral position) and 2 ms pulse can rotate it to 180 degree.

All servo motors work directly with your +5V supply rails but we have to be careful on the amount of current the motor would consume, if you are planning to use more than two servo motors a proper servo shield should be designed.

**Specifications:**

* Power: 4.8V - 6V DC max (5V works well)
* Average Speed: 60 degrees in 0.20 sec (@ 4.8V), 60 degrees in 0.16 sec (@ 6.0V)
* Weight: 62.41g
* Torque: At 4.8V: 8.5 kg-cm / 120 oz-in, and at 6V: 10 kg-cm / 140 oz-in.
* Size mm: (L x W x H) 40.7 x 19.7 x 42.9
* Spline Count: 25

### SOFTWARES

Software used to implement this project are:

* Arduino IDE
* IBM Cloud
  + IBM IOT Watson Platform
  + Node Red
* MIT App Inventor 2

**Arduino IDE:**

The [Arduino](https://en.wikipedia.org/wiki/Arduino) integrated development environment ([IDE](https://en.wikipedia.org/wiki/Integrated_development_environment)) is a [cross-platform](https://en.wikipedia.org/wiki/Cross-platform) application (for [Windows](https://en.wikipedia.org/wiki/Windows), [macOS](https://en.wikipedia.org/wiki/MacOS), [Linux](https://en.wikipedia.org/wiki/Linux)) that is written in the programming language [Java](https://en.wikipedia.org/wiki/Java_(programming_language)). 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.

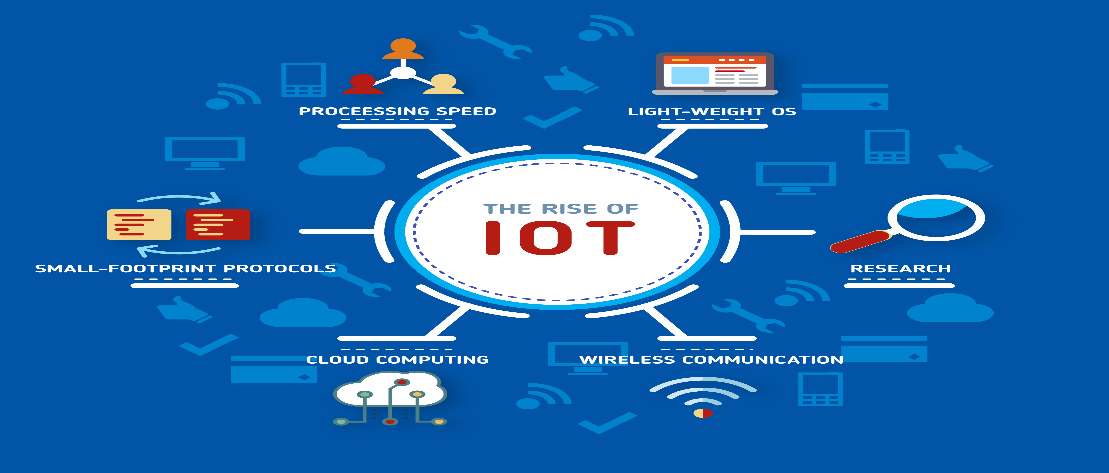
The source code for the IDE is released under the [GNU General Public License](https://en.wikipedia.org/wiki/GNU_General_Public_License), version 2. The Arduino IDE supports the languages [C](https://en.wikipedia.org/wiki/C_(programming_language)) and [C++](https://en.wikipedia.org/wiki/C%2B%2B) using special rules of code structuring. The Arduino IDE supplies a [software library](https://en.wikipedia.org/wiki/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 andlinked with a program stub *main()*  into an executable [cyclic executive](https://en.wikipedia.org/wiki/Cyclic_executive) program with the [GNU toolchain](https://en.wikipedia.org/wiki/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**

1. **IOT Platform:**

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](https://en.wikipedia.org/wiki/Flow-based_programming) development tool for [visual programming](https://en.wikipedia.org/wiki/Visual_programming_language) developed originally by [IBM](https://en.wikipedia.org/wiki/IBM) for wiring together hardware devices, [APIs](https://en.wikipedia.org/wiki/Application_programming_interface) and [online services](https://en.wikipedia.org/wiki/Online_services) as part of the [Internet of Things](https://en.wikipedia.org/wiki/Internet_of_Things).



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

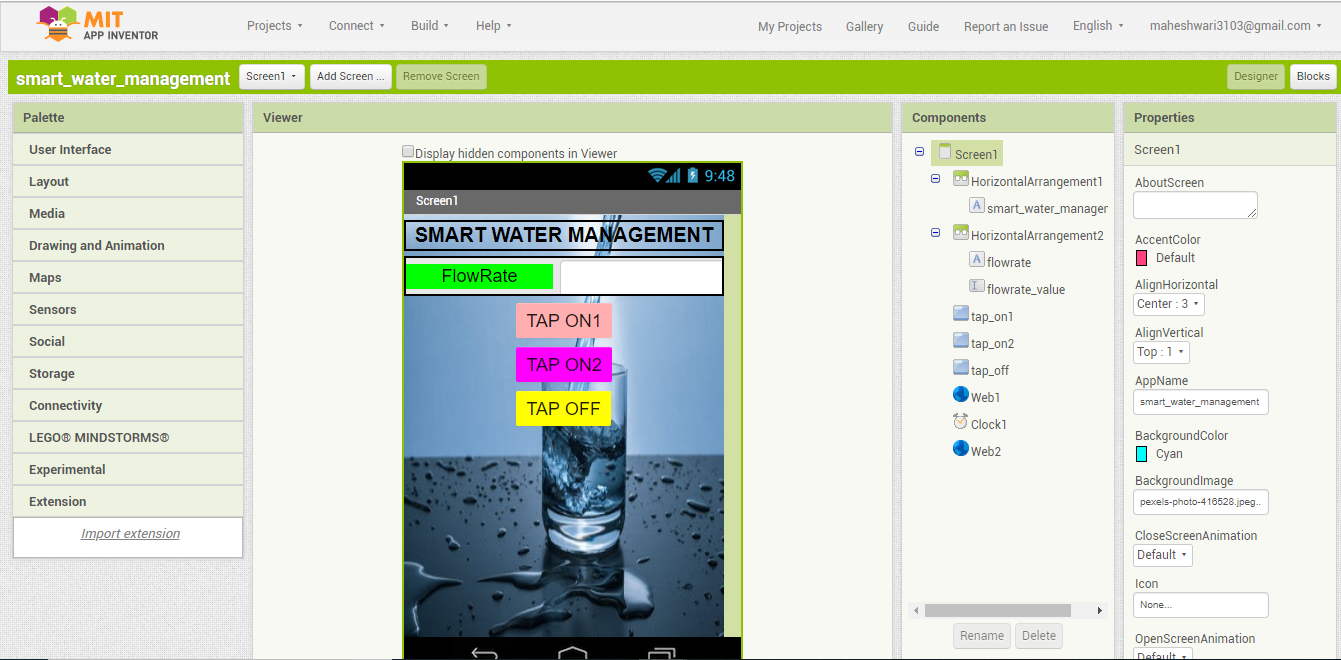
### MIT APP INVENTOR 2

**App Inventor for Android** is an open-source web application originally provided by [Google](https://en.wikipedia.org/wiki/Google), and now maintained by the Massachusetts Institute of Technology (MIT), which allows newcomers to [computer programming](https://en.wikipedia.org/wiki/Computer_programming) to create applications for the [Android](https://en.wikipedia.org/wiki/Android_(operating_system)) operating system (OS).

It uses a graphical interface very similar to [Scratch](https://en.wikipedia.org/wiki/Scratch_(programming_language)) and the StarLogo TNG [user interface](https://en.wikipedia.org/wiki/User_interface), which allows users to 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.[[1]](https://en.wikipedia.org/wiki/App_Inventor_for_Android#cite_note-MIT_roots-1)

App Inventor and the projects on which it is based are informed by [constructionist learning](https://en.wikipedia.org/wiki/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](https://en.wikipedia.org/wiki/Seymour_Papert) and the MIT Logo Group in the 1960s and has also manifested itself with [Mitchel Resnick](https://en.wikipedia.org/wiki/Mitchel_Resnick)'s work on [Lego Mindstorms](https://en.wikipedia.org/wiki/Lego_Mindstorms) and [StarLogo](https://en.wikipedia.org/wiki/StarLogo).

App Inventor also supports the use of [cloud data](https://en.wikipedia.org/wiki/Cloud_database) via an experimental FirebaseDB component.



**DESCRIPTION ABOUT THE PROJECT**

### In our project, Smart Water Management using IBM Watson Platform, as the name says it is all about management of water supply throughout the scale, right from small societies, townships to entire urban infrastructure and also for irrigation water supply management.

### Firstly, we can check the water levels in the tank using ultrasonic sensors and based on water levels we can start and stop the water supply from the main tank to the individual houses automatically.

### And later at the house level we can check the flow rate of water by using flow sensors and all these data can be sent to cloud and these parameters can be shown on the User Interface which is created in the Node Red.

### From user interface/App, based on flowrate value from the Flow Sensor we provide individuals whether the tap in the room has to be on/off so that they will be able to store the water in home without manual interaction even when they are not present in home.

**MAIN TANK MONITORING:**

### Firstly, we have to check the water levels in the tank using ultrasonic sensors and based on water levels we can start and stop the water supply from the main tank to the individual houses automatically.

### For checking the water levels in the tank, we use ultrasonic sensor

### For rotating the tap directly, we use a Servo Motor

### For checking the water levels in the tank, we connect an ultrasonic sensor. It gives the result about water level in the tank. Based on the level of water i.e., if water level is high in the tank, the tap (i.e., servo motor rotates 90degrees) is kept on directly and if water level in the tank is low, the tap (i.e., servo motor) is kept off.

### 

### SOFTWARE IMPLEMENTATION

**Arduino IDE:**

We need to write a program to develop any project. Now first step is to write program in Arduino IDE. Ultrasonic sensor is interfaced with the nodemcu. The NodeMCU is connected to the system and the NodeMCU board has to be selected to get the output. And we need to select the port(ex-COM3).

### #include <Servo.h>

### Servo servo1;

### int trigPin = D1;

### int echoPin = D2;

### long distance;

### long duration;

### 

### void setup ()

### {

### servo1.attach(D3);

### pinMode (trigPin, OUTPUT);

### pinMode (echoPin, INPUT);

### Serial.begin(9600);

### }

### 

### void loop()

### {

### ultra();

### servo1.write(0);

### if(distance <= 10)

### {

### servo1.write(90);

### }

### }

### 

### void ultra()

### {

### digitalWrite(trigPin, LOW);

### delayMicroseconds(2);

### digitalWrite(trigPin, HIGH);

### delayMicroseconds(10);

### digitalWrite(trigPin, LOW);

### duration = pulseIn(echoPin, HIGH);

### distance = duration\*0.034/2;

### Serial.print("object distance is: ");

### Serial.println(distance);

### delay(5000);

### }

### 

### By this code we can tap on or tap off the motor in the tank.

### In House Monitoring

### After tap on in the main tank, we should know the flowrate of water.so to know the flowrate of water we use flow sensor with is interfaced with NodeMCU and uploaded to cloud. whenever the flowrate is high the servo motor rotates to 180degrees.whenever the flowrate is moderate the servo motor rotates to 90degrees.whenever we want to tap off the servo motor position will be 0degrees.

### 

### The coding to this logic is as follows

### #include <ESP8266WiFi.h>

### #include <PubSubClient.h>

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

### const char\* ssid = "SIRI JioFi";

### const char\* password = "bmr12345";

### //////////////////////

### #define flow\_pin D2 //gpio4

### #define LED\_PIN D7 //gpio13

### volatile long flow\_count=0;

### float calibrationFactor = 4.5;

### float flowRate;

### unsigned int flowMilliLitres;

### unsigned long totalMilliLitres;

### float totalLitres;

### unsigned long oldTime;

### void ICACHE\_RAM\_ATTR flow\_counter()

### {

### flow\_count++;

### }

### ///////////////////////////

### #define ORG "1lpuff"

### #define DEVICE\_TYPE "water555"

### #define DEVICE\_ID "555"

### #define TOKEN "9876543210"

### String data;

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

### char subtopic[] = "iot-2/cmd/tap/fmt/String";

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

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

### char token[] = TOKEN;

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

### WiFiClient wifiClient;

### PubSubClient client(server, 1883, callback, wifiClient);

### void setup()

### {

### Serial.begin(115200);

### Serial.println();

### pinMode(D1,OUTPUT);

### wifiConnect();

### mqttConnect();

### /////////////////

### 

### flow\_count = 0;

### flowRate = 0.0;

### flowMilliLitres = 0;

### totalMilliLitres = 0;

### oldTime = 0;

### pinMode(LED\_PIN, OUTPUT);

### digitalWrite(LED\_PIN, HIGH); // We have an active-low LED attached

### 

### pinMode(flow\_pin, INPUT);

### attachInterrupt(flow\_pin, flow\_counter, FALLING);

### 

### //////////////////////////////////////

### 

### Serial.print("Connecting to ");

### Serial.print(ssid);

### WiFi.begin(ssid, password);

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

### {

### delay(500);

### Serial.print(".");

### }

### Serial.println("");

### 

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

### Serial.println(WiFi.localIP());

### }

### void loop()

### {

### if (!client.loop())

### {

### mqttConnect();

### }

### delay(100);

### 

### ////////////

### 

### if((millis() - oldTime) > 1000) // Only process counters once per second

### {

### detachInterrupt(flow\_pin);

### flowRate = ((1000.0 / (millis() - oldTime)) \* flow\_count) / calibrationFactor;

### oldTime = millis();

### flowMilliLitres = (flowRate / 60) \* 1000;

### totalMilliLitres += flowMilliLitres;

### totalLitres = totalMilliLitres \* 0.001;

### unsigned int frac;

### Serial.print("flowrate: ");

### Serial.print(int(flowRate)); // Print the integer part of the variable

### Serial.println("L/min");

### 

### /\*Serial.print("."); // Print the decimal point

### frac = (flowRate - int(flowRate)) \* 10;

### Serial.print(frac, DEC) ; // Print the fractional part of the variable

### Serial.print("L/min");

### 

### Serial.print("Current Liquid Flowing: "); // Output separator

### Serial.print(flowMilliLitres);

### Serial.print("mL/Sec");

### Serial.print(" Output Liquid Quantity: "); // Output separator

### Serial.print(totalLitres);

### Serial.println("L");\*/

### 

### flow\_count = 0;

### attachInterrupt(flow\_pin, flow\_counter, FALLING);

### }

### 

### 

### ///////////

### 

### PublishData(flowRate);

### 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("nWiFi 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();

### Serial.println();

### }

### }

### void initManagedDevice()

### {

### if (client.subscribe(subtopic))

### {

### Serial.println("subscribe to cmd OK");

### }

### else

### {

### Serial.println("subscribe to cmd FAILED");

### }

### }

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

### {

### Serial.print("callback invoked for topic: ");

### Serial.println(subtopic);

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

### {

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

### }

### Serial.println(data);

### if(data == "servo\_on")

### {

### // digitalWrite(D1,HIGH);

### Serial.println("tap is on");

### }

### else if(data == "servo\_off")

### {

### //digitalWrite(D1,LOW);

### Serial.println("tap is off");

### }

### data ="";

### }

### void PublishData(float flowRate)

### {

### 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 = "{\"d\":{\"house1\_flowRate\":";

### payload += flowRate;

### payload += "}}";

### Serial.print("Sending payload: ");

### Serial.println(payload);

### 

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

### {

### Serial.println("Publish ok");

### }

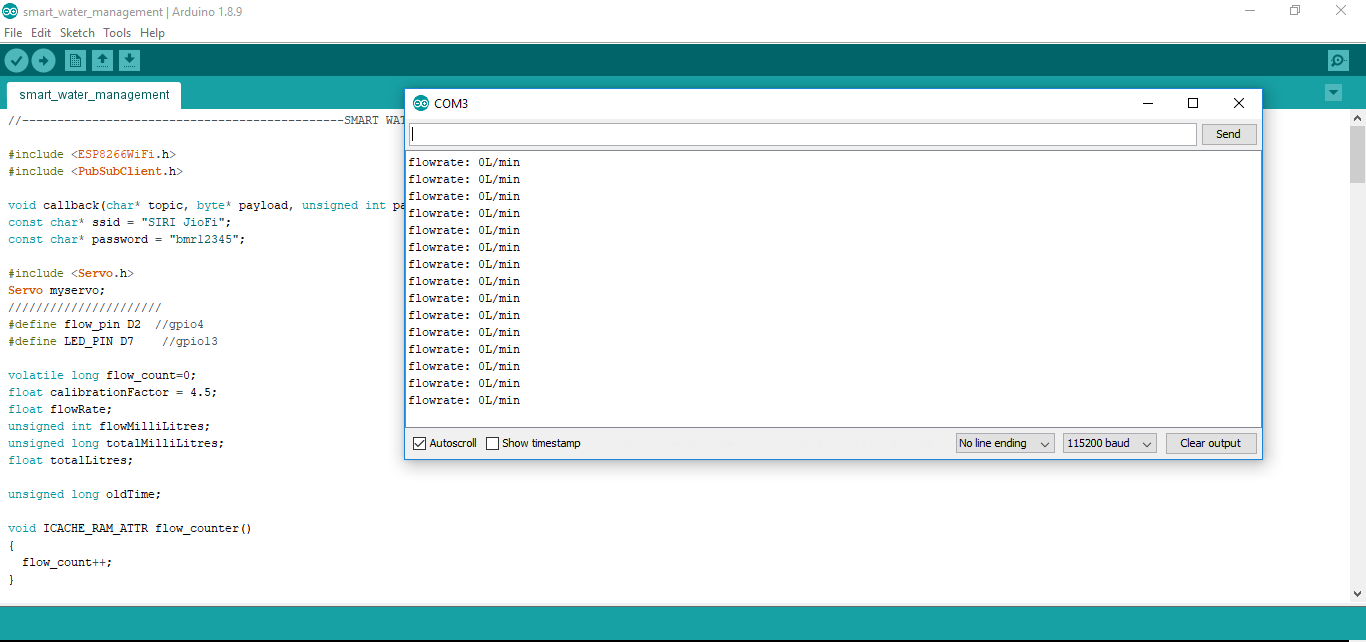
### else

### {

### Serial.println("Publish failed");

### }

### }

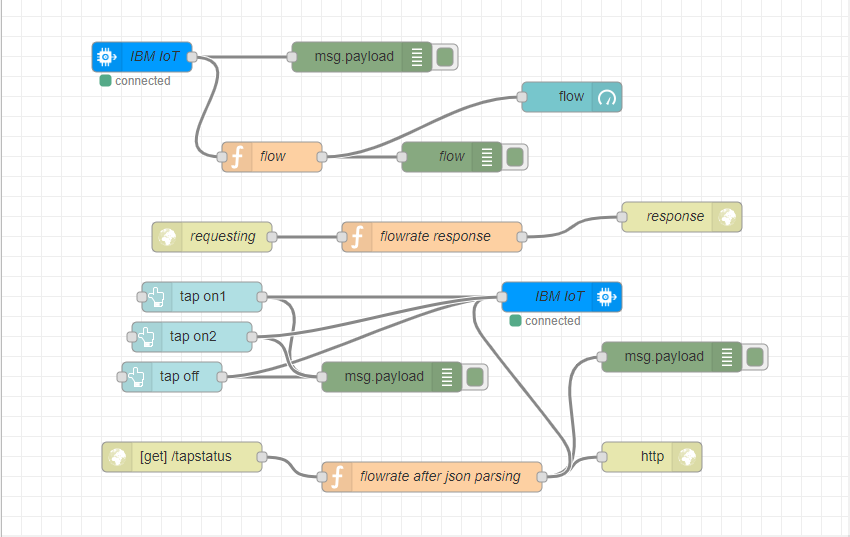


**Node-RED:**

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.



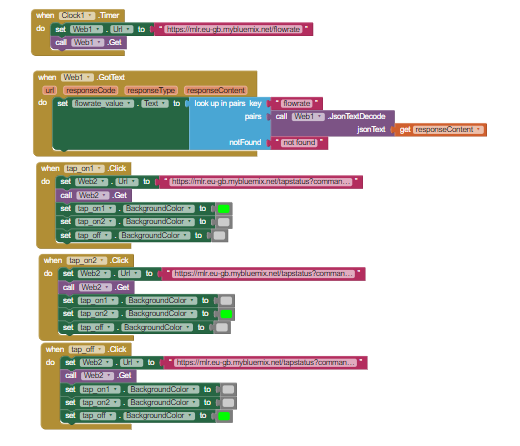
**Mit app inventor 2**

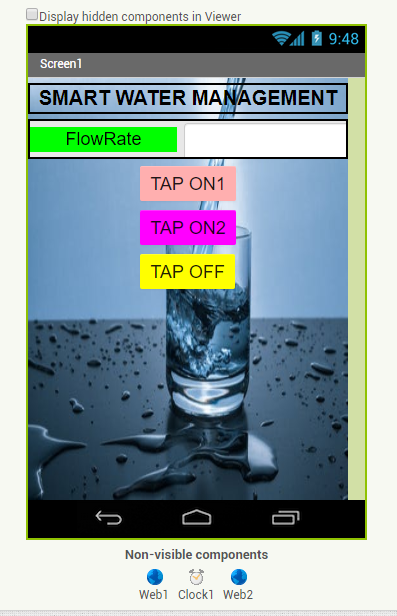
**App Inventor for Android** is an open-source web application originally provided by Google, and now maintained by the Massachusetts Institute of Technology(MIT), which allows newcomers to computer programming to create software applications for the Android operating system (OS).

It uses a graphical interface very similar to Scratch and the Star Logo TNG user interface, which allows users to 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.





The data will be shown in the mobile app.so we can monitor the flowrate and we can operate the tap on, tap off.

### CONCLUSION

On the basis of comparative of system used for Water Management System. For Smart Water Management Using IoT, our system is efficient in term of cost accuracy, time. And it needs less man power.