**Centalized Remote Agriculture Pump Control System**

Team**: DEVELOPERS**

S.Chirudeep

R.Pavani

P.Manikanteshwar

**INDEX:**

* **Aim**
* **Introduction**
* **Technical Specifiations**
* **Hardware Tools**
* **Software Tools**
* **Block Diagram**
* **Block Diagram Explination**
* **Node Mcu**
* **Submersible Pump**
* **Relay Module 2 Channel**
* **Connectors**
* **SOFTWARE PROGRAM**
* **Node RED**
* **MIT App Inventor**

**ABSTARCT**

In Egypt increasing demand for desert land reclamation has resulted in the increasing use of submersible pumps. And that's where the operation and monitoring of pump malfunction takes a lot of effort and time. The purpose of this paper to design an embedded system which can remotely ON/OFF and monitoring pump status through sending and receiving messages on mobile phones to help the farmer to supply water to the crops and get information about the type of faults. This system, saving time, less Physical effort and inconvenience Loss/ Frequent damage of irrigation equipment Wastage of water and electricity Objective. The design consists of the a Node MCU ESP8266 board connected with a mobile phone.

Remote control systems are a very useful way to control and monitor devices rapidly and effortlessly. This paper proposes a new architecture for remote control of agriculture devices and provides security to the farm through online-streaming. Proposed system makes use of few technologies which allow remote control of the agricultural motor, pesticides thereby making a farmer's work much easier and less dependent of the conditions present. As outcome of this paper an Android application running on a smart phone, connected to the server vie internet is build. This server can be controlled through a Arduino IDE application written in C++.

* **AIM:**

The main objective of this project is to develop a remote control pump in agricultural field for farmers.

* **INTRODUCTION:**

In modern days, as a result of advances in technology, human beings are

interested to remotely control different systems and applications. Remote controlling is one of the greatest inventions of humankind. It is a method

of controlling an electronic or electromechanical system remotely without the

need to physically touch and operate the controlled device. While most of the

earlier remote controllers were connected to the device being controlled by wires, the first wireless remote control for a consumer electronics device was invented in

the 1950’s .

Nowadays, remote controls are most commonly used in many

applications and consumer electronic devices such as television sets, DVD

players, air conditioners, automatic garage door opener systems with most of

them operating wirelessly from a short Line-of-Sight (LoS) distance. The main

technology used in home remote controls is infrared (IR) light. While IR

transmission is limited to LoS operation, radio-wave transmission is used for

controlling devices located in remote or distant locations. As a complementary

method to infrared remote controls, the radio remote control is used with electric

garage door or gate openers, automatic barrier systems, burglar alarms, industrial

automation systems, military applications such as satellite linked remote

controlling of unmanned airplanes (drones), space travels and other related

applications. It is obvious that in the case of remote controlling using a radio, we

need to have a transmitter that generates and transmits the control commands and

a receiver at the remote site that receives the transmitted commands from the transmitter, interprets, and produces different states of operation depending on the

transmitted commands. In the case of radio-communication, the control command

can be transmitted using the wired or wireless network making it capable of

controlling an application of interest at any time and any place.

Beyond the normal use of voice and multimedia communications, in recent

years the use of wireless and mobile phone devices is becoming prevalent in

remote controlling applications such as house and property security surveillance

system, theft control and monitoring systems, remote motor speed control, remote

real-time industrial process control & monitoring, remote door locking system,

remote controlling of electrical apparatus control in offices and homes, remote

operation of robotic systems, remote vehicular security systems, remote

switching systems and other relevant applications. Therefore, remote

controlling of systems using telephone signals is not new however the

implementation differs from application to application.

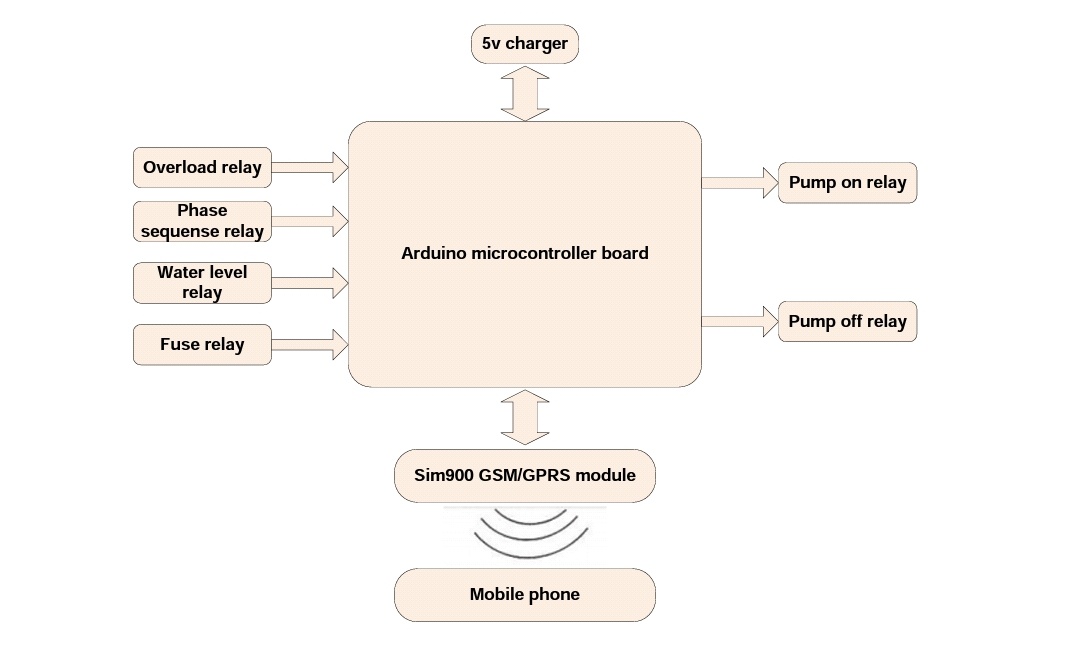
* **TECHNICAL SPECIFICATIONS :**
* **HARDWARE TOOLS:**

1. Node MCU
2. Water Pump
3. Relay Module
4. Connecting Wires

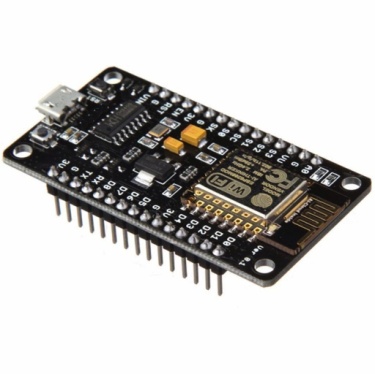
* **SOFTWARE TOOLS:**

1. Arduino IDE
2. Node RED
3. MIT App Inventor

* **BLOCK DIAGRAM:**

****

* **BLOCK DIAGRAM EXPLANATION:**
* **NodeMCU:**



NodeMCU is an open source IoT platform.It includes firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module. The term "NodeMCU" by default refers to the firmware rather than the development kits.NodeMCU Dev Kit has Arduino like Analog (i.e. A0) and Digital (D0-D8) pins on its board.It supports serial communication protocols i.e. UART, SPI, I2C etc.Using such serial protocols we can connect it with serial devices like I2C enabled LCD display, Magnetometer HMC5883, MPU-6050 Gyro meter + Accelerometer, RTC chips, GPS modules, touch screen displays, SD cards etc.NodeMCU Development board is featured with wifi capability, analog pin, digital pins and serial communication protocols.The way of developing NodeMCU with a well-known IDE i.e. Arduino IDE. We can also develop applications on NodeMCUusing Arduino development environment. ESP8266 comes with capabilities of:

2.4 GHz Wi-Fi (802.11 b/g/n, supporting WPA/WPA2),general-purpose input/output (16 GPIO),

Inter-Integrated Circuit (I²C) serial communication protocol,

analog-to-digital conversion (10-bit ADC)

Serial Peripheral Interface (SPI) serial communication protocol,

I²S (Inter-IC Sound) interfaces with DMA(Direct Memory Access) (sharing pins with GPIO),

UART (on dedicated pins, plus a transmit-only UART can be enabled on GPIO2), and

pulse-width modulation (PWM).

It employs a 32-bit RISC CPU based on the Tensilica Xtensa L106 running at 80 MHz (or overclocked to 160 MHz). It has a 64 KB boot ROM, 64 KB instruction RAM and 96 KB data RAM. External flash memory can be accessed through SPI.

ESP8266 module is low cost standalone wireless transceiver that can be used for end-point IoT developments.

To communicate with the ESP8266 module, microcontroller needs to use set of AT commands. Microcontroller communicates with ESP8266-01 module using UART having specified Baud rate.

There are many third-party manufacturers that produce different modules based on this chip. So, the module comes with different pin availability options like,

ESP-01 comes with 8 pins (2 GPIO pins) – PCB trace antenna. (shown in above figure)

ESP-02 comes with 8 pins, (3 GPIO pins) – U-FL antenna connector.

ESP-03 comes with 14 pins, (7 GPIO pins) – Ceramic antenna.

ESP-04 comes with 14 pins, (7 GPIO pins) – No ant.

**ESP8266-01 Module Pin Description ESP8266 pins:**

ESP8266-01 Module Pins

3V3: - 3.3 V Power Pin.

GND: - Ground Pin.

RST: - Active Low Reset Pin.

EN: - Active High Enable Pin.

TX: - Serial Transmit Pin of UART.

RX: - Serial Receive Pin of UART.

GPIO0 & GPIO2: - General Purpose I/O Pins.

These pins decide what mode (boot or normal) the module starts up in.

It also decides whether the TX/RX pins are used for Programming the module or for serial I/O purpose.

To program the module using UART, Connect GPIO0 to ground and GPIO2 to VCC or leave it open.

To use UART for normal Serial I/O leave both the pins open (neither VCC nor Ground).

**Features:**

Communication interface voltage: 3.3V.

Antenna type: Built-in PCB antenna is available.

Wireless 802.11 b/g/n standard

WiFi at 2.4GHz, support WPA / WPA2 security mode

Support STA/AP/STA + AP three operating modes

Built-in TCP/IP protocol stack to support multiple TCP Client connections (5 MAX)

D0 ~ D8, SD1 ~ SD3: used as GPIO, PWM, IIC, etc., port driver capability 15mA

AD0: 1 channel ADC

Power input: 4.5V ~ 9V (10VMAX), USB-powered

Current: continuous transmission: ˜70mA (200mA MAX), Standby: <200uA

Transfer rate: 110-460800bps

Support UART / GPIO data communication interface

Remote firmware upgrade (OTA)

Flash size: 4MByte

* **SUBMERSIBLE PUMP :**

A submersible pump (or sub pump, electric submersible pump (ESP)) is a device which has a hermetically sealed motor close-coupled to the pump body. The whole assembly is submerged in the fluid to be pumped. The main advantage of this type of pump is that it prevents pump cavitations’, a problem associated with a high elevation difference between pump and the fluid surface. Small DC Submersible water pumps push fluid to the surface as opposed to jet pumps having to pull fluids. Submersibles are more efficient than jet pumps.

* **Working principle:-**

The submersible pumps used in ESP installations are multistage centrifugal pumps operating in a vertical position.

Although their constructional and operational features underwent a continuous evolution over the years, their basic operational principle remained the same.

Produced liquids, after being subjected to great centrifugal forces caused by the high rotational speed of the impeller, lose their kinetic energy in the diffuser where a conversion of kinetic to pressure energy takes place.

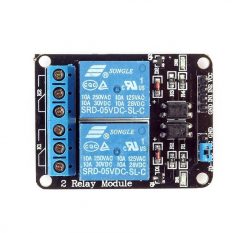
This is the main operational mechanism of radial and mixed flow pumps.

The pump shaft is connected to the gas separator or the protector by a mechanical coupling at the bottom of the pump.

Well fluids enter the pump through an intake screen and are lifted by the pump stages.

Other parts include the radial bearings (bushings) distributed along the length of the shaft providing radial support to the pump shaft turning at high rotational speeds.

* **Relay Module 2 Channel:**

Relays are switches that open and close circuits electromechanically or electronically.Relays control one electrical circuit by opening and closing contacts in another circuit.As relay diagrams show, when a relay contact is normally open (NO), there is an open contact when the relay is not energized.

Classification or the types of relays depend on the function for which they are used. Some of the categories include protective, reclosing, regulating, auxiliary and monitoring relays.

2-Channel Relay Interface Board

This item is a 5V 4 Channel Relay board, Can be able to control various appliances, and the other equipments with large current. It can be controlled directly by Microcontroller Arduino , 8051, AVR, PIC, DSP, ARM, ARM, MSP430, TTL logic .

2-Channel Relay interface board

Each one needs 15-20mA Driver Current

Equiped with high-current relay : DC30V 10A AC250V 10A AC125V 10A

Standard interface that can be controlled directly by microcontroller

Indication LED’s for Relay output status.

* **CONNECTORS:**

A simple device that physically links, couples, or connects, two things together. A male connector has pins that fit into the sockets, or receptacles, of a female connector, as the connectors mate. A male connector sometimes is referred to as a plug, and a female connector as a jack.



A Universal Serial Bus (**USB**) is a common interface that enables communication between devices and a host controller such as a personal computer (PC). It connects peripheral devices such as digital cameras, mice, keyboards, printers, scanners, media devices, external hard drives and flash drives.

* **SOFTWARE:**

#include <ESP8266WiFi.h>

#include <PubSubClient.h>

const char\* ssid = "GOOD BOY";

const char\* password = "mani2810";

#define ORG "ppql51"

#define DEVICE\_TYPE "SONY"

#define DEVICE\_ID "IOT"

#define TOKEN "pavanisony"

String command

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

char topic[] = "iot-2/cmd/home/fmt/String";

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

char token[] = TOKEN;

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

//Serial.println(clientID);

WiFiClient wifiClient;

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

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

void setup() {

Serial.begin(115200);

Serial.println();

pinMode(D1,OUTPUT);

pinMode(D2,OUTPUT);

wifiConnect();

mqttConnect();

}

void loop() {

if (!client.loop()) {

mqttConnect();

}

}

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(topic)) {

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(topic);

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

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

command += (char)payload[i];

}

Serial.println(command);

if(command == "Motor1ON"){

digitalWrite(D1,HIGH);

Serial.println("Motor 1 is Switched ON");

}

else if(command == "Motor1OFF"){

digitalWrite(D1,LOW);

Serial.println("Motor 2 is Switched OFF");

}

if(command == "Motor2ON"){

digitalWrite(D2,HIGH);

Serial.println("Motor 2 is Switched ON");

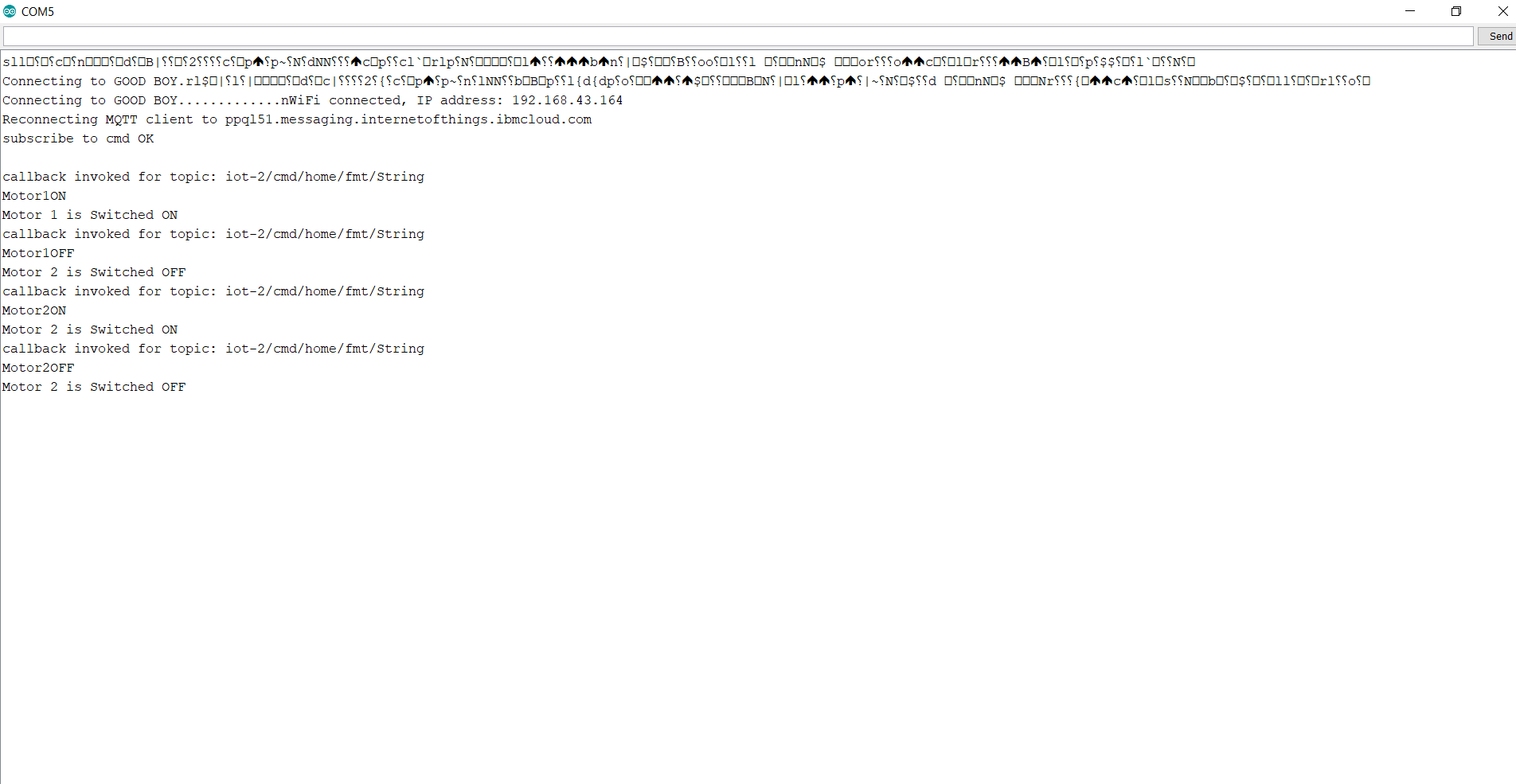
}

else if(command == "Motor2OFF"){

digitalWrite(D2,LOW);

Serial.println("Motor 2 is Switched OFF");

}

 Fig: serial monitor

**Node RED Block:**

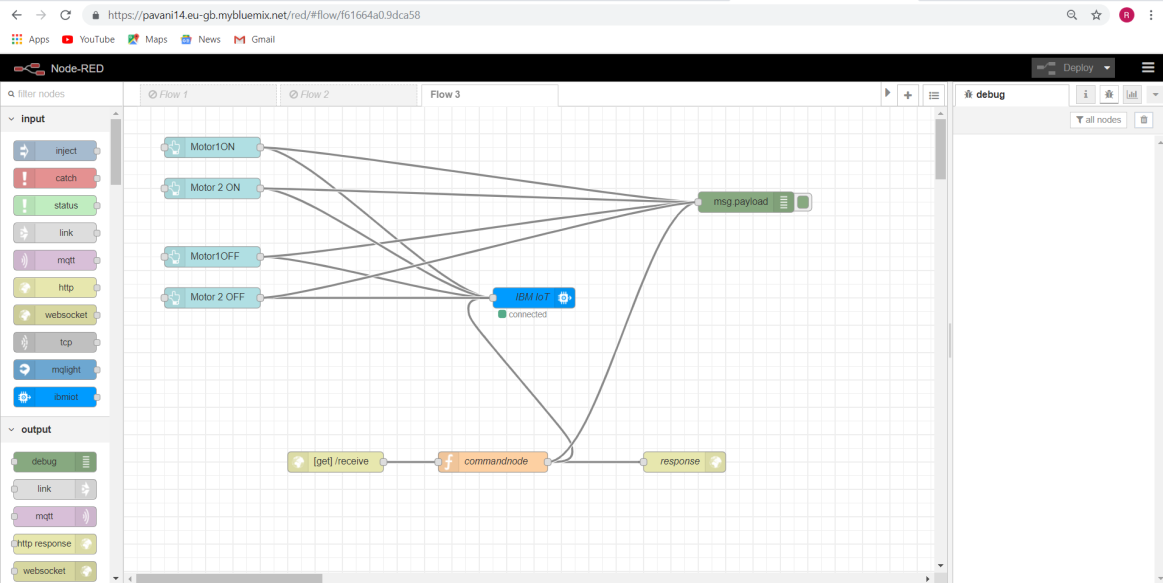
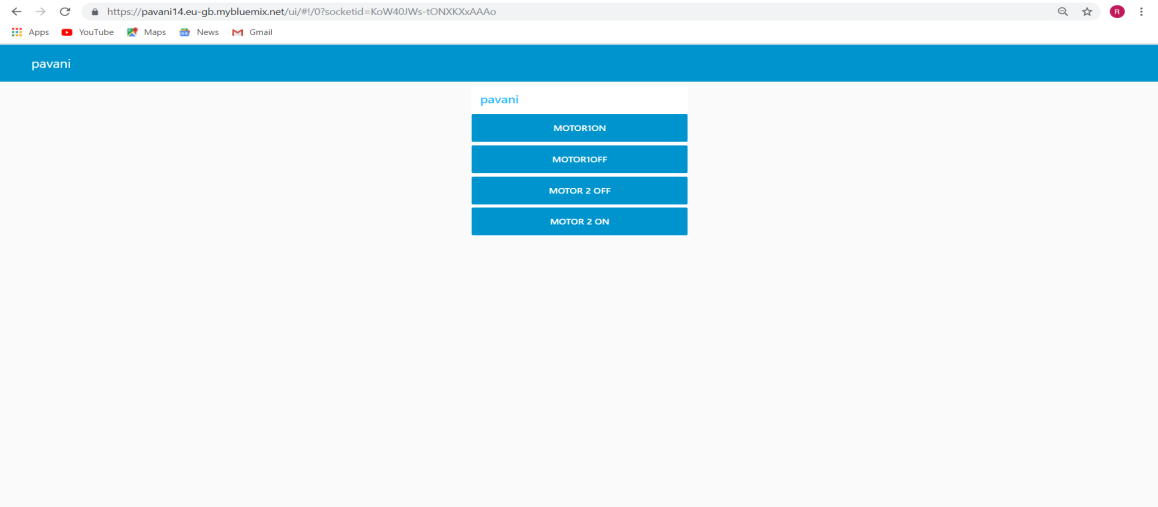
****

Fig: Before deploying

Fig: output window

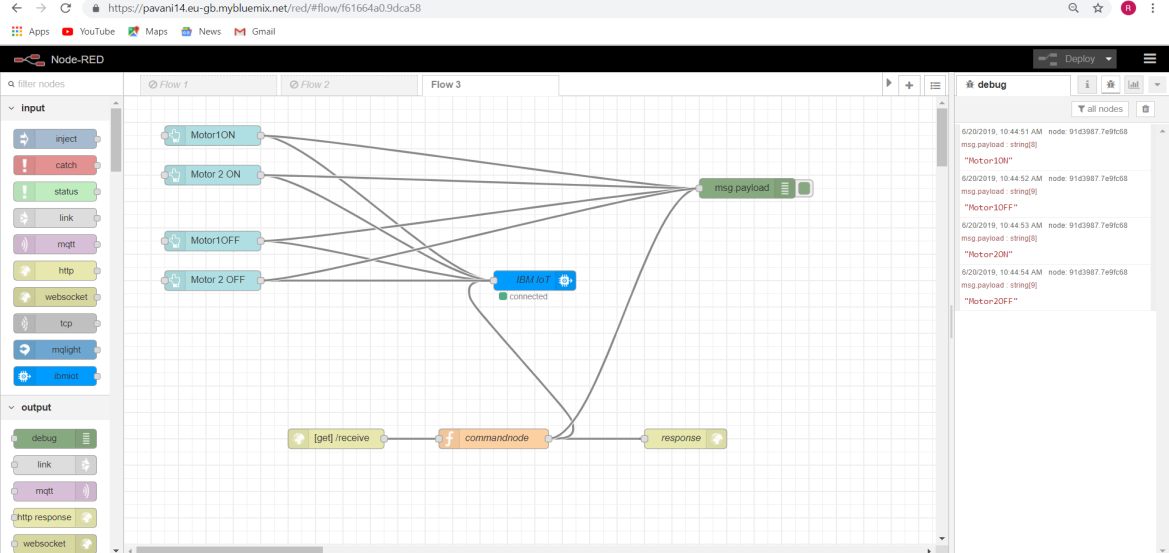
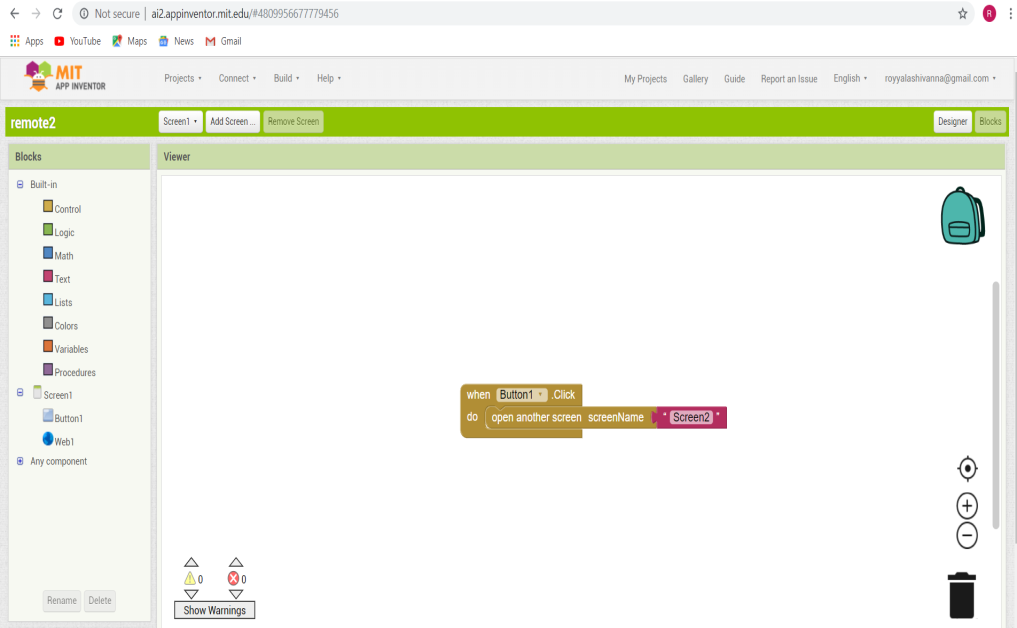


fig: output obtained after the switching action of above figure

* **MIT App Inventor:**

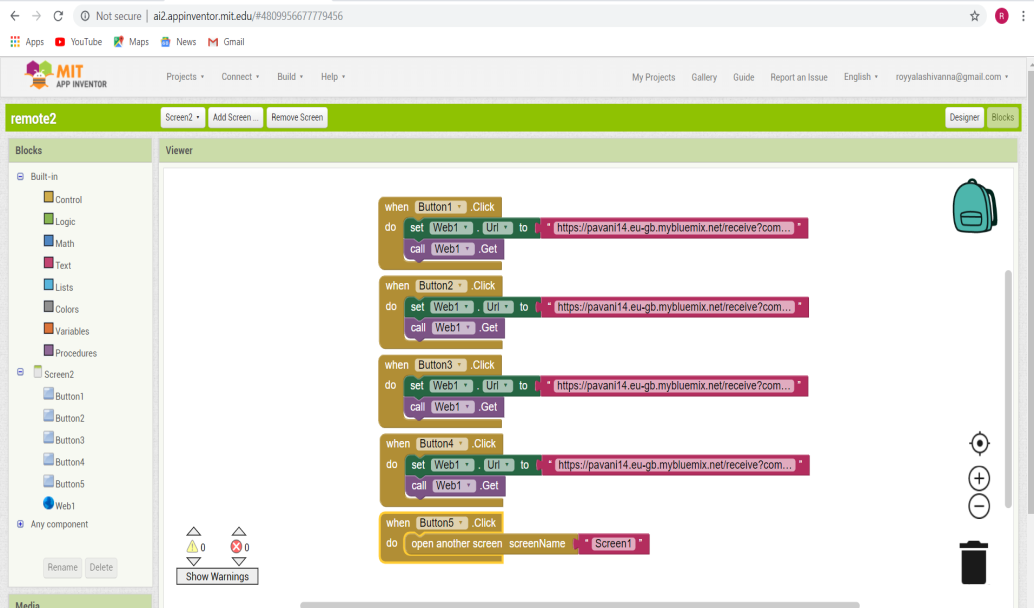
Fig: screen1 Block

Fig:screen2 Block

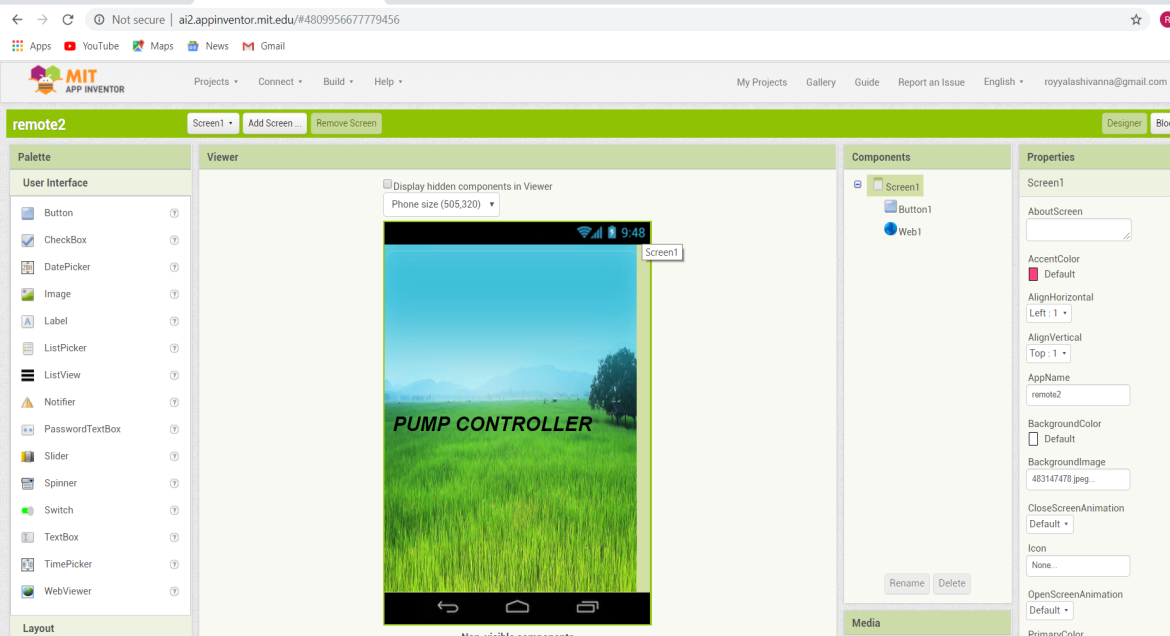
****

Fig:screen1 Designer

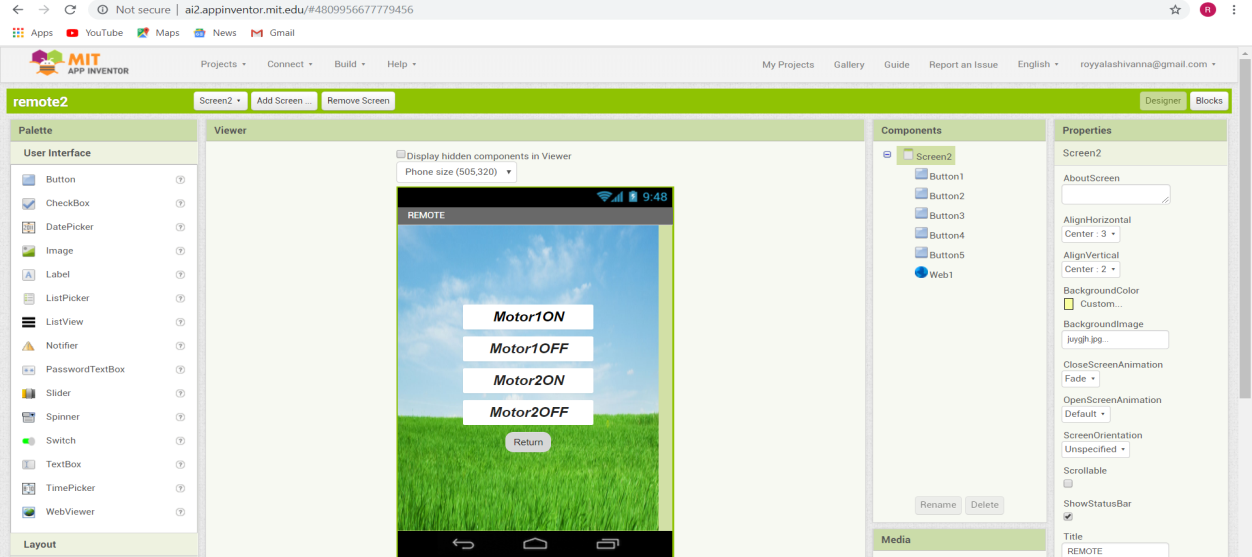


Fig:screen2 Designer