EMCOG SOLUTIONS

EV THROTTLE MONITOR AND CONTROL

PROJECT DOCUMENTATION

Submitted by

P. ARUNKUMAR

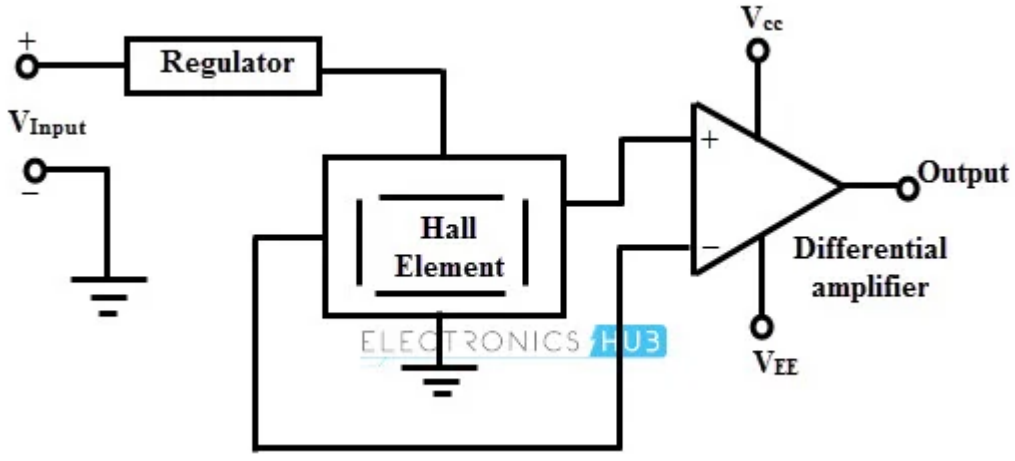
TABLE OF CONTENTS

INTRODUCTION:

This project focuses on the monitoring and control of throttle and direction of movement for an electric bike using the Internet of Things (IoT) technology. To achieve this, a linear hall effect sensor is utilized to measure the throttle position, and the collected data is transmitted to an Arduino UNO developmental board (ATmega328P) through a 10-bit analog-to-digital converter (ADC). Additionally, an ESP8266 module is serially connected to the Arduino UNO to establish a remote connection with the server, enabling users to monitor and control the bike's parameters remotely and conveniently from anywhere and at any time. To provide a user-friendly graphical interface, the project utilizes the RemoteXY platform, which is compatible with both Android and iOS devices. With this integrated system, users can easily access and interact with the electric bike, enhancing their overall experience and control over its functionality. This documentation will outline the various components, connections, and functionalities of the project, providing a comprehensive guide for its implementation and usage

a) HALL SENSOR:

A Hall effect sensor is an electronic device that is designed to detect the Hall effect, and convert its findings into electronic data, either to switch a circuit on and off, provide a measurement of a varying magnetic field, be processed by an embedded computer or displayed on an interface

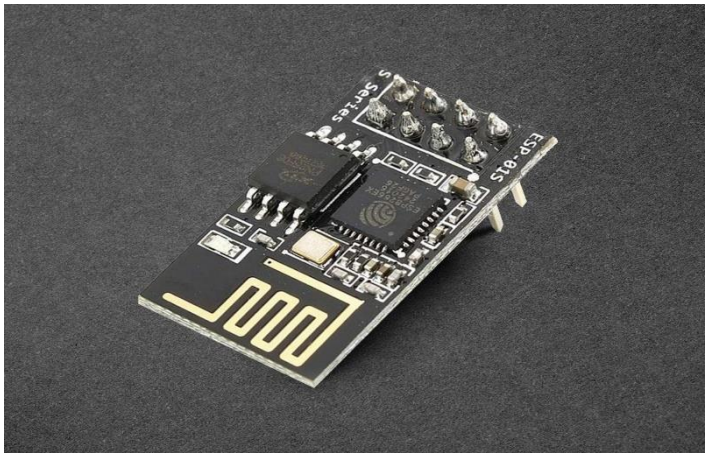


Hall effect sensor diagram

Linear Hall sensors (analog) render precise and continuous measurements based on magnetic field strength; they do not switch on and off. Within the Hall effect sensor, the Hall element sends the electric potential difference (voltage brought about by the magnetic interference) to an amplifier in order to make the change in voltage large enough to be perceived by the embedded system.

b) ESP8266 Wi-Fi Module

ESP8266 is Wi-Fi enabled system on chip (SoC) module developed by Espressif system. It is mostly used for development of IoT (Internet of Things) embedded applications.



3V3: - 3.3 V Power Pin.

GND: - Ground Pin.

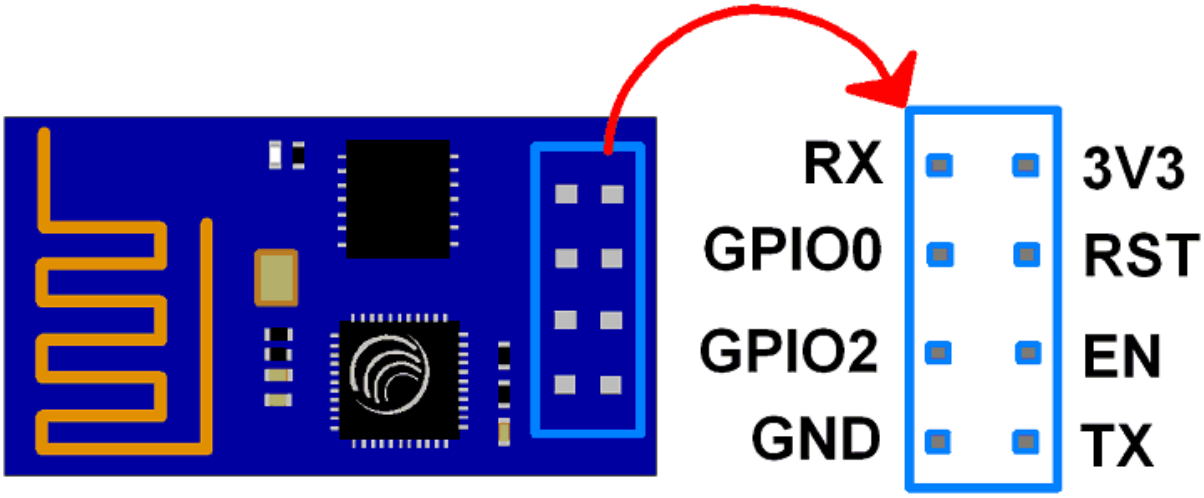
RST: - Active Low Reset Pin.

EN: - Active High Enable Pin.

TX: - Serial Transmit Pin of UART.

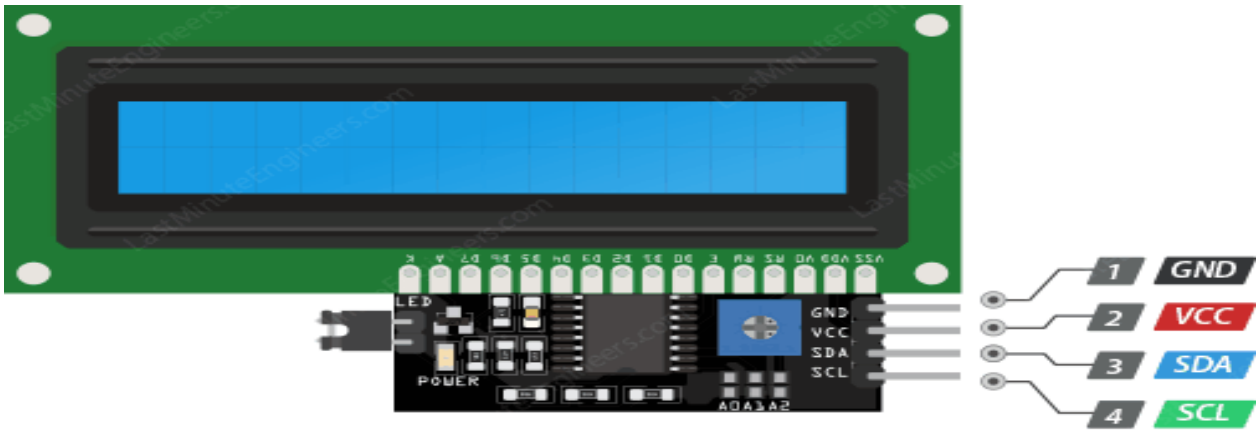
RX: - Serial Receive Pin of UART. GPIO0 & GPIO2: - General Purpose I/O Pins. These pins decide what mode (boot or normal) the module starts up in. It also decides whether the TX/RX pins are used for Programming the module or for serial I/O purpose.

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).



I2C LCD DISPLAY:

The I2C 16×2 Arduino LCD Screen is using an I2C communication interface. It is able to display 16×2 characters on 2 lines, white characters on blue background.



This display overcomes the drawback of LCD 1602 Parallel LCD Display in which you'll waste about 8 Pins on your Arduino for the display to get working. Luckily in this product, an I2C adapter is directly soldered right onto the pins of the display. So all you need to connect are the I2C pins, which shows a good library and little of coding.

PREREQUISITES:

A. HARDWARE REQUIREMENTS :

1.Arduino UNO developmental board

2.ESP8266 2.4GHz Wi-Fi module

3.HALL sensor(A3144)

4.I2C 16X2 LCD display

5.Switches to control direction and online cloud control

6.Serial USB communication cable for Arduino to burn .hex file to MCU

B. SOFWARE REQUIREMENTS :

1.Arduino IDE – download link

2.RemoteXY library installed in Arduino IDE - download link

3.RemoteXY android / iOS app - download link

4.RemoteXY GUI editor – download link

4.LiquidCrystal_I2C - in Arduino library manager

PROCEDURE:

Step 1: Ensure you have all the necessary hardware and software components for the project.

Step 2: Hardware Connection:

ESP8266:

a) Configure the ESP8266 to a baud rate of 9600 and set it's mode to Station mode as per the instructions provided in the configuration section.

b) Connect the VCC (3.3V) pin and enable pin of the module to a regulated 3.3V power supply available on the Arduino UNO. If not available, use a 7803.3 regulator IC to provide a 3.3V supply. Connect the ground pin to the common ground.

c) Establish a hard serial connection between the ESP8266 module and the Arduino UNO by connecting the transmission pin (Tx pin) of the module to the Rx pin of the Arduino UNO and the receiver pin (Rx pin) of the module to the Tx pin of the Arduino UNO.

I2C LCD Display:

a) The I2C LCD display requires a 5V power supply, which can be obtained from the UNO development board.

b) Connect the serial clock line (SCL) of the display to the A5 pin of the Arduino UNO and the serial data line (SDA) to the A4 pin of the Arduino UNO.

c) Note down the I2C address of the I2C_LCD. For example, if the address is 0x27, take note of it.

Hall Sensor:

a) The hall sensor used in the project requires a 5V power supply, which can be obtained from the development board or an external supply with a 7805 IC.

b) Connect the output data pin of the hall sensor to the A0 pin of the Arduino UNO.

Direction Control:

a) The direction control is achieved by using a simple switch. Two ends of the switch are connected to the output terminals. When the switch is in the reverse direction, the two terminals are short-circuited. When the switch is in the forward direction, the two terminals are open-circuited. Connect one pin to a digital input pin with internal pull-up on the Arduino board and connect the other pin to ground.

After connecting the hardware components, the schematic would resemble the one provided.

Step 3: Software Setup:

a) Install the Arduino IDE, which is available for Windows, macOS, and Linux. You can find the download link in the prerequisites page.

b) In the Arduino IDE, go to Tools and search for the Library Manager. Search for the "LiquidCrystal_I2C" library and install the latest version.

c) Download and install the RemoteXY library, which is also provided in the prerequisites page.

d) Additionally, install the RemoteXY app on your handheld device. The download link is available in the prerequisites page.

Step 4: RemoteXY Editor Section (Not required):

a) Access the RemoteXY editor portal using the provided link in the prerequisites section.

b) Log in to your RemoteXY account or create a new one if you don't have an account.

c) Configure the settings according to your Wi-Fi module, create a server, and make a note of its token.

d) The RemoteXY editor provides various GUI components, but you can only use a maximum of five components in the free version. If you need more, you will need to purchase the PRO version.

e) After creating the GUI, click on "Get Source Code" to download the source code.

Note: The reference link and project link are provided in the references section.

f) Copy and paste the downloaded source code into your Arduino IDE.

Step 5: Fuse the Normal Program:

Write the normal program to determine the throttle percentage and direction of rotation. Fuse this code with the source code provided by the RemoteXY editor. The fused code should already be written with explanatory comments. Simply copy the code to your Arduino IDE.

Step 6: Upload the Code:

Upload the code to the Arduino UNO development board using the Arduino IDE.

Step 7: Set Up the Hotspot:

Set up the hotspot specified in the code. Open the serial monitor and reset the microcontroller. The ESP8266 module will automatically connect to the hotspot and access the RemoteXY server.

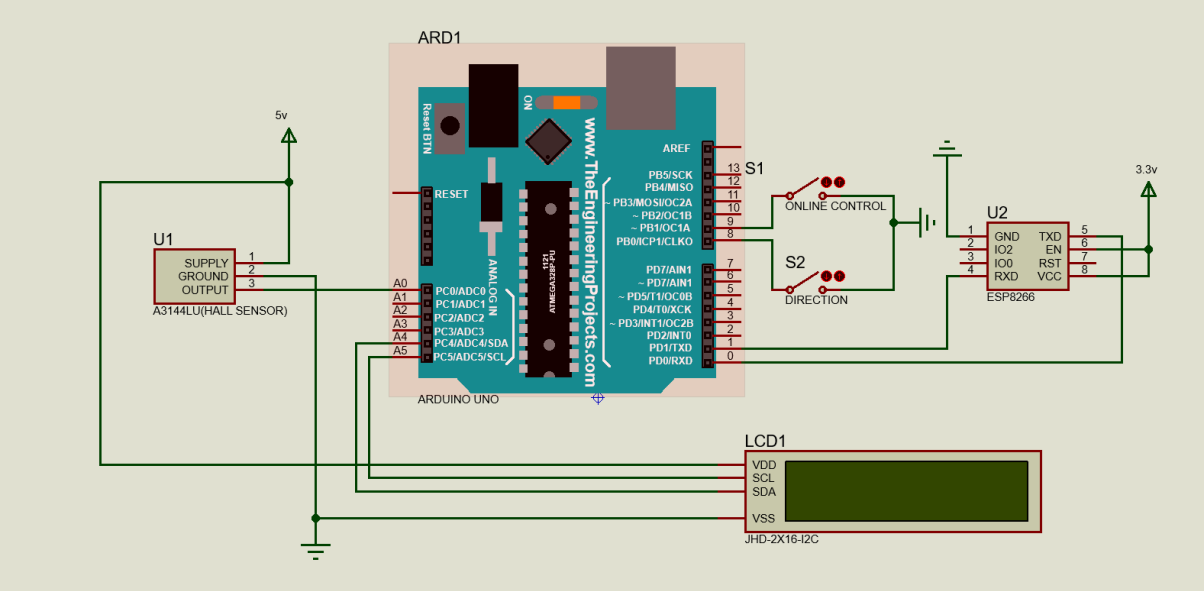
Step 8: Connect to RemoteXY App:

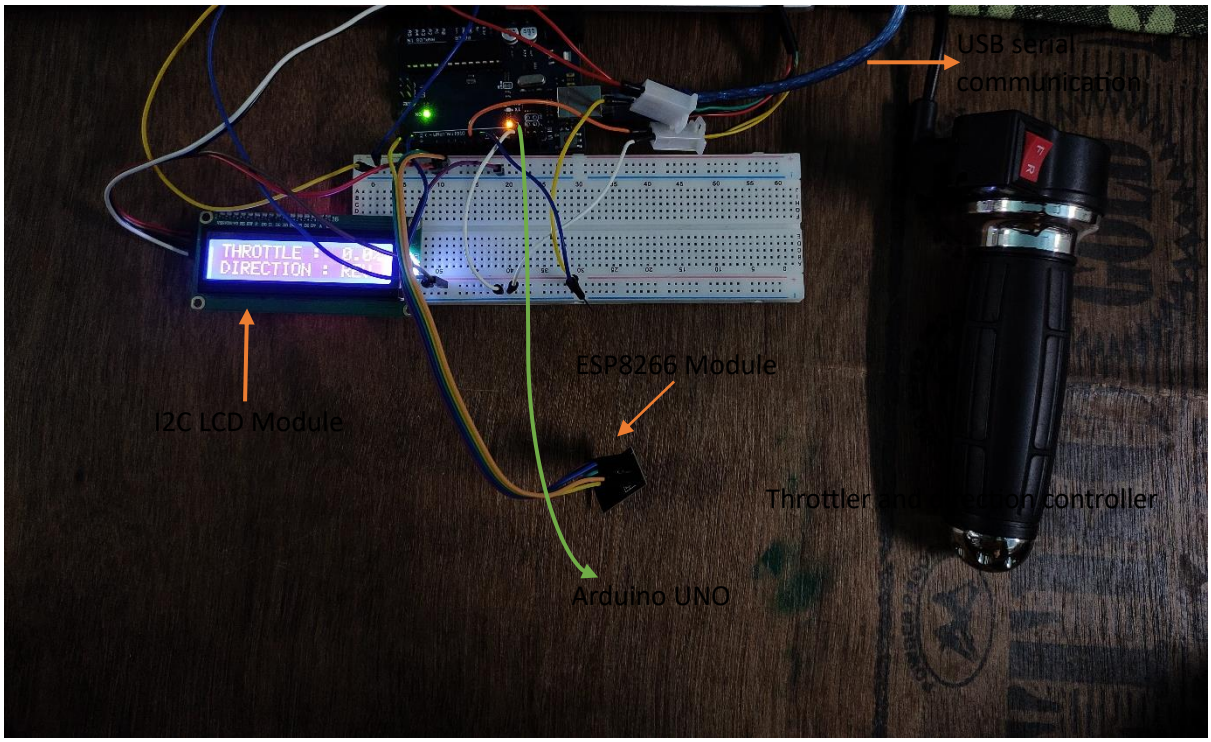
Open the RemoteXY app on your handheld device and select the "Cloud" section. Enter the server address and the token provided in the RemoteXY editor.

Step 9: Retrieve the GUI:

The GUI from the cloud will be retrieved, and now you can monitor and control the data using the RemoteXY app.

SCHEMATICS:


ARD1
5v
U1
SUPPLY
GROUND
OUTPUT
A3144LU(HALL SENSOR)
ARDUINO UNO
www.TheEngineeringProjects.com
RESET
AREF
S1
ONLINE CONTROL
S2
DIRECTION
U2
ESP8266
3.3v
LCD1
VDD
SCL
SDA
VSS
JHD-2X16-I2C


USB serial communication
I2C LCD Module
ESP8266 Module
Arduino UNO

Code:

remote

```
// RemoteXY select connection mode and include library
#define REMOTEXY_MODE__ESP8266_HARDSERIAL_CLOUD                    //esp8266 connected to arduino via hardserial

#include <RemoteXY.h>                                              //including RemoteXY library
#include<Wire.h>                                                   //including wire library to communicate with I2C devices
#include<LiquidCrystal_I2C.h>                                      //including I2C LCD library to control LCD
#include<string.h>                                                 //including strings library for string manipulation

// RemoteXY connection settings
#define REMOTEXY_SERIAL Serial
#define REMOTEXY_SERIAL_SPEED 9600                                 //baud rate
#define REMOTEXY_WIFI_SSID "ARUN-PAVILION"                         //my  wifi name
#define REMOTEXY_WIFI_PASSWORD "thankyoutokitou"                   //my wifi password
#define REMOTEXY_CLOUD_SERVER "cloud.remotexy.com"                 //RemoteXY cloud server web address
#define REMOTEXY_CLOUD_PORT 6376                                   //server port for cloud server
#define REMOTEXY_CLOUD_TOKEN "fea02a733d3f8c13f06993c062f0307d"    //token to log in


// RemoteXY configurate
#pragma pack(push, 1)
uint8_t RemoteXY_CONF[] =                                          // 86 bytes
  { 255,1,0,26,0,79,0,16,180,0,68,17,28,32,68,28,8,36,4,0,
  9,31,6,28,2,26,129,0,3,6,92,6,38,84,72,82,79,84,84,76,
  69,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,68,73,
  82,69,67,84,73,79,78,0,67,1,6,15,28,5,32,26,11,67,1,63,
  15,32,5,32,180,11 };
```

remote

```

// this structure defines all the variables and events of your control interface
struct {

    // input variables
  int8_t slide;                                                    // =0..100 slider position

    // output variables
  float graph;
  char acc[11];                                                    // string UTF8 end zero
  char dir[11];                                                    // string UTF8 end zero

    // other variable
  uint8_t connect_flag;                                            // =1 if wire connected, else =0

} RemoteXY;
#pragma pack(pop)


LiquidCrystal_I2C lcd(0x27,16,2);                                  //defining lcd => I2C address = 0x27, 16 x 2 display

#define start_hall 179                                             //value of ADC(hall sensor) with 0% throttle
#define end_hall 880                                               //value of ADC(hall sensor) with 100% throttle
#define aver 3                                                     //number of times hall sensor reading should be taken for a loop
#define tolerance 5                                                //tolerance for middle values of throttle
#define tolerance2 10                                              //tolerance for initial and final values throttle
```

remote

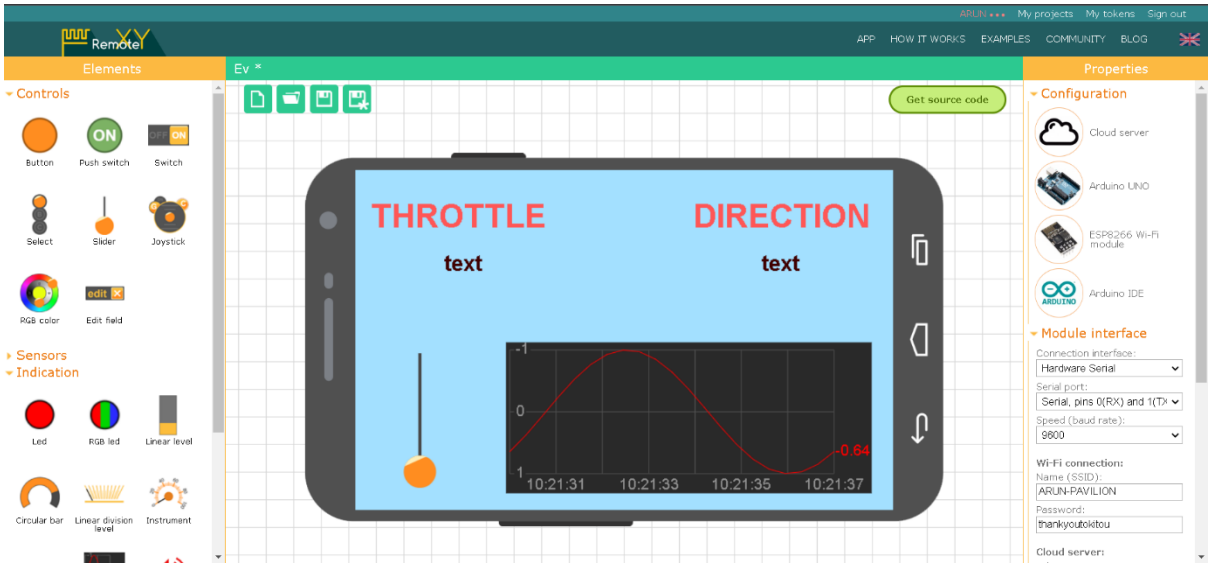
```

int value_prev,value_pres,direct,control;
control = 0;                                                       //variable specifing the online control
float percent;                                                     //throttle percent
void setup()
{
  RemoteXY_Init ();                                                //initialization of REMOTEXY module
  pinMode(8,INPUT_PULLUP);                                         //8th pin set as digital input pin with iternal pullup
  pinMode(9,INPUT_PULLUP);                                         //9th pin set as digital input pin with iternal pullup
  lcd.init();                                                      //initialize the lcd
  lcd.backlight();                                                 //open the backlight
  lcd.setCursor(0,0);                                              //set lcd cursor to row: 0 and col : 0
  lcd.print("THROTTLE :     %");
  lcd.setCursor(0,1);                                              //set lcd cursor to row: 1 and col : 0
  lcd.print("DIRECTION : ");

}

void loop()
{
  RemoteXY_Handler ();                                             //updating the RemoteXY structure variables to cloud in loop
    direct = digitalRead(8);                                       //read direction pin (forward or reverse)
  lcd.setCursor(12,1);                                             //set lcd cursor to row: 0 and col : 0
  if(direct == 0){                                                 //if reverse
    lcd.print("REV");
    strcpy(d,"REVERSE");
```

CONTINUATION OF CODE:

```
remote
    strcpy(RemoteXY.dir,"REVERSE");                          //store the string "REVERSE" in RemoteXY structure variable - dir
  }
  else{
    lcd.print("FOR");
    strcpy(RemoteXY.dir,"FORWARD");                          //store the string "REVERSE" in RemoteXY structure variable - dir
  }

  control = digitalRead(9);                                  //read online control pin (enable online control or not)
  if(control){                                               //if online control was not enabled
    value_pres = 0;
  for(int i = 0;i < aver;i++){                               //do read the analog pin A0(output of HALL sensor) for n number of time
      value_pres += analogRead(A0);                          //make sum of hall sensor reading
    }
  value_pres /= aver;                                        //make average of the HALL sensor reading


  if(abs(value_pres - hall_start) < tolerance2)              //if throttle is near 0%
    value_pres = hall_start;                                 //set throttle to 0%
  else if(abs(value_pres - hall_end) < tolerance2)           //if throttle is near 100%
    value_pres = hall_end;                                   //set throttle to 100%
  else if(abs(value_pres - value_prev) < tolerance)          //if throttle change value is lesser than tolerace
    value_pres = value_prev;                                 //don't change throttle value
  value_prev = value_pres;

  percent = ((value_pres - hall_start) * 100.0) / (hall_end - hall_start);        //ADC hall sensor value to percent
  }
```

```
remote §
  else                                                       //if online control is enabled
    percent = (float)RemoteXY.slide;                         //read slider value from cloud

  dtostrf(percent , 5 , 1 , RemoteXY.acc);                   //convert percent to string and send to cloud
  RemoteXY.graph = percent;                                  //send percent value to cloud for graph plotting
  lcd.setCursor(10,0);                                       //set lcd cursor to row: 0 and col : 10
  lcd.print(t);
}
```

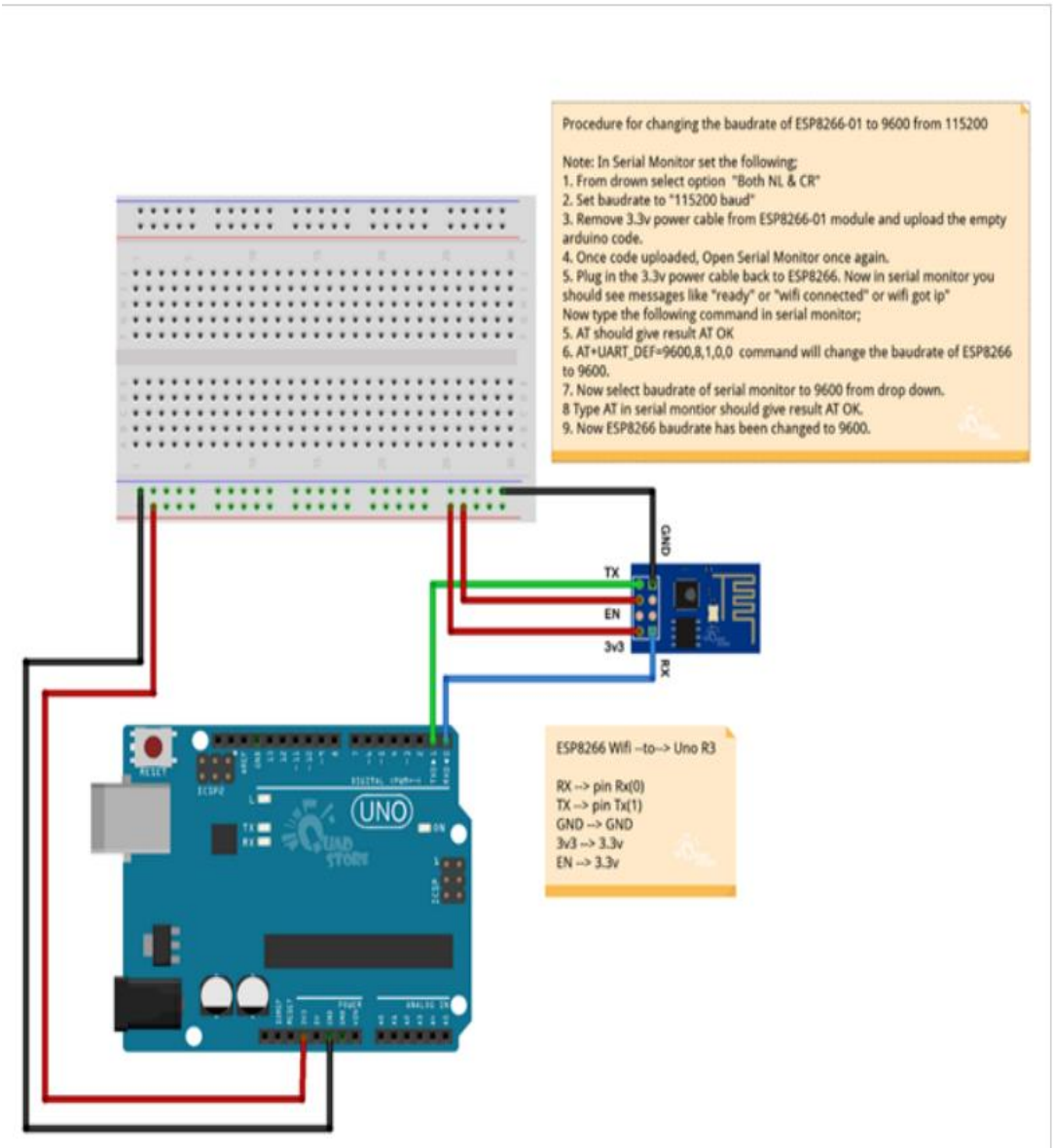
RemoteXY GUI:



CONFIGURATION:

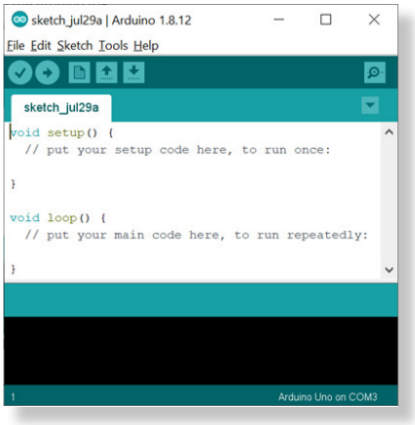
1.ESP8266 Wi-Fi Module:

a) Baud rate change to 9600:

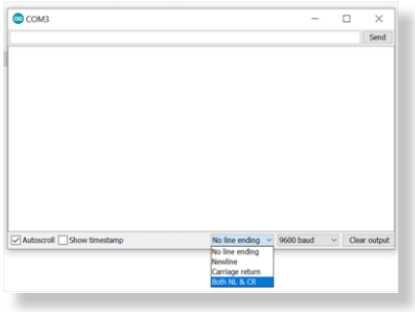


ACTION-1: PROCEDURE FOR CHANGING THE BAUDRATE OF ESP8266-01 TO 9600 FROM 115200

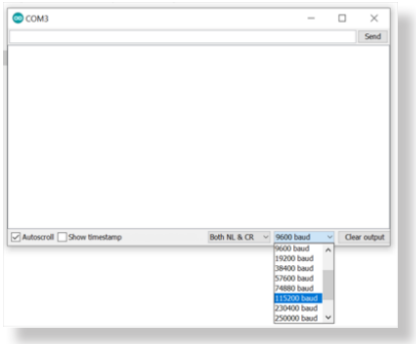
1. Open New Sketch from the Arduino IDE.



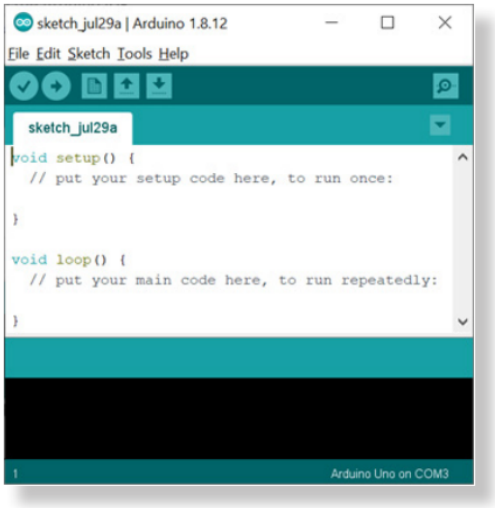
2. Open Serial Monitor andchange the following options
3. From drown select option "Both NL & CR"



4. Set baudrate to "115200 baud"

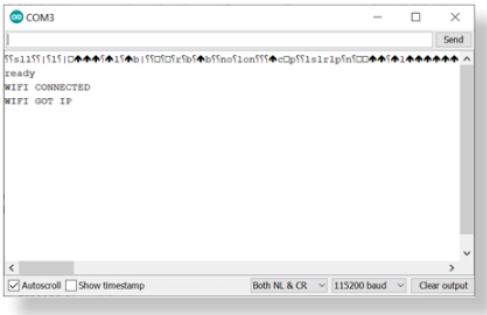


5. Remove **3.3v power cable** from ESP8266-01 module.
6. Upload the New empty Sketch/Code to the Uno R3 board.

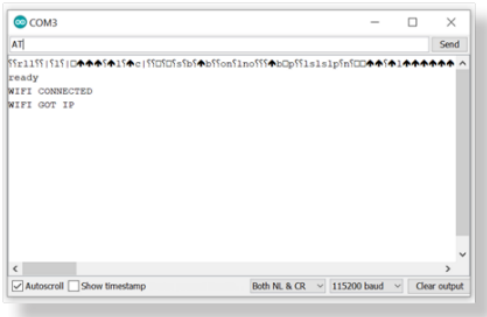


7. Once code uploaded, Open Serial Monitor once again.

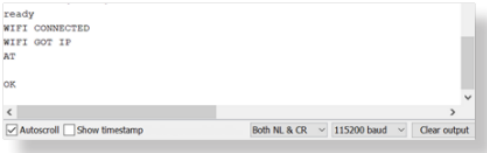
8. Plug in the **3.3v power cable** back to ESP8266. Now in serial monitor you should see messages like "ready" or "WIFI CONNECTED" or "WIFI GOT IP"



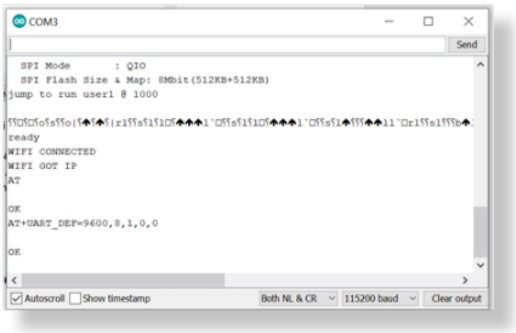
9. Now type the commandAT in serial monitor and click Send button.



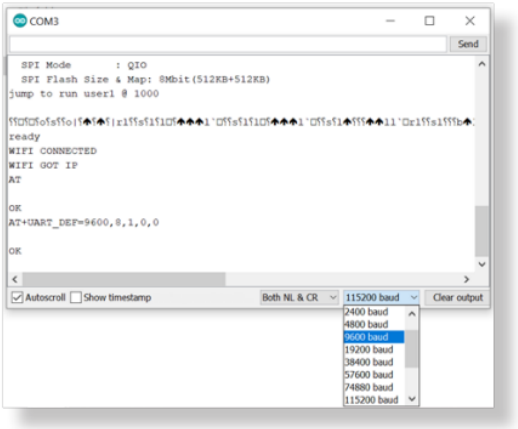
10. AT should give result AT OK



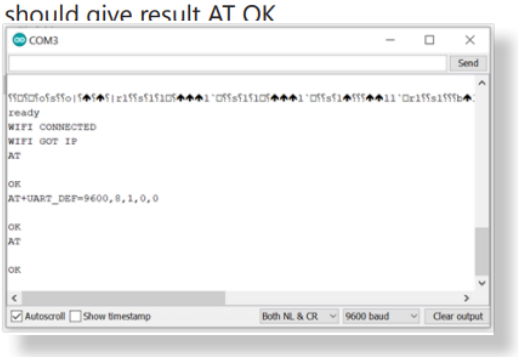
11. Now type command AT+UART_DEF=9600,8,1,0,0 command will change the baudrate of ESP8266 to 9600.



12. Now select baudrate of serial monitor to 9600 from drop down.



13. Type AT in serial monitor should give result AT OK



14. Success! Now the baudrate of ESP8266-01 has been changed to 9600.

b) Mode change to Station Mode (STA):

1.with the same circuit diagram as above, open serial monitor and type in the following commands

First, type the following command to test whether the communication is successful or not.

```
AT
```

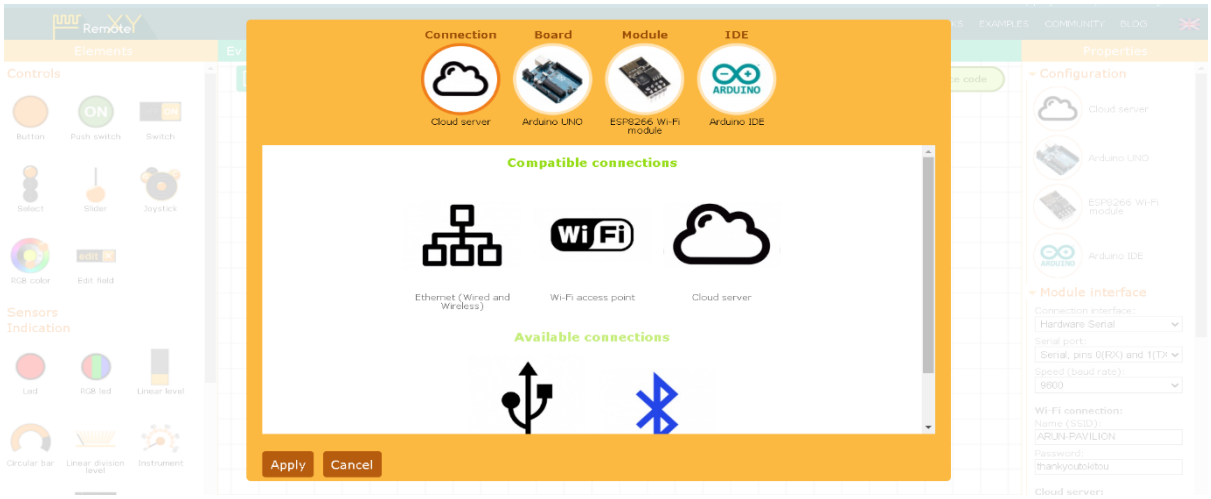
Then, I will restart the ESP8266 Module using the following command, just to make sure that I start fresh.

```
AT+RST
```

Now, I need to set the Mode of operation as Station Mode. For this, use the following command.

```
AT+CWMODE=1
```

2. a) REMOTEXY CONFIGURATION:



SELECT THE FOLLWING:

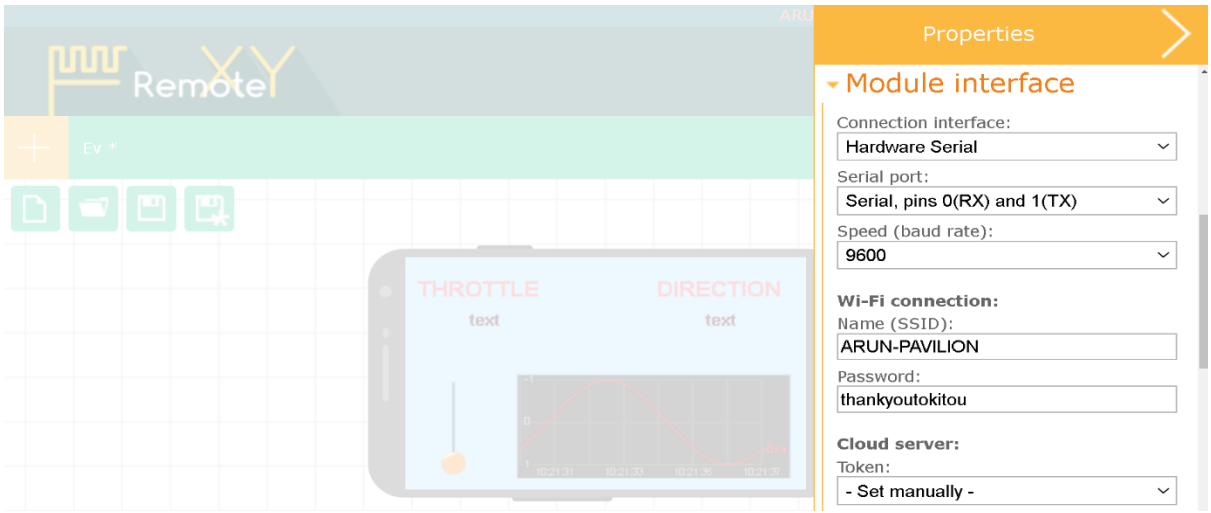
1.CONNECTION -> CLOUD SERVER

2.BOARD -> ARDUINO UNO

3.MODULE -> ESP8266 Wi-Fi Module

4.IDE -> ARDUINO IDE

b) REMOTEXY MODULE INTERFACE:



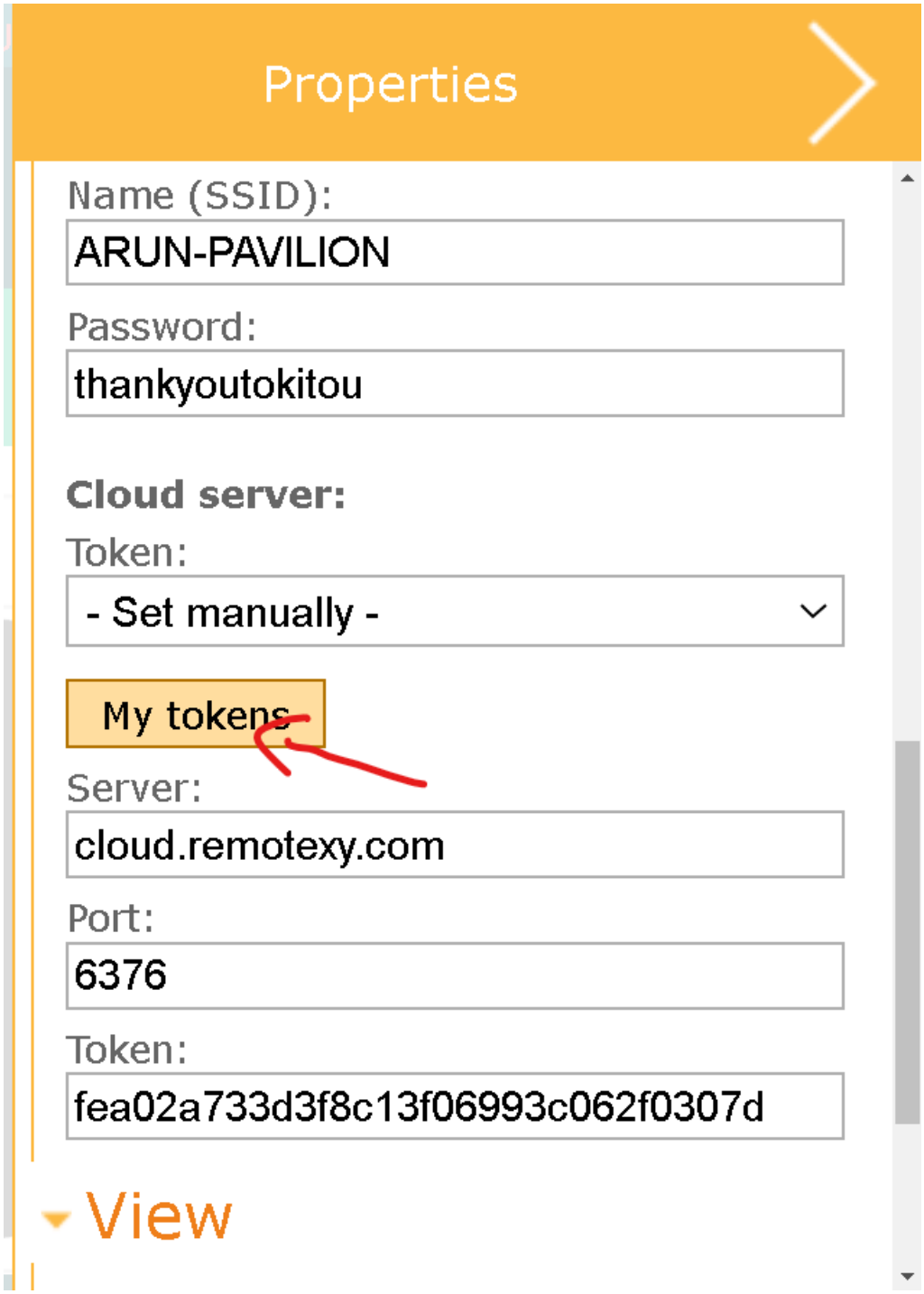
SELECT THE FOLLOWING AND ENTER YOUR HOTSPOT NAME AND PASSWORD

1.CONNECTION INTERFACE -> Hardware Serial

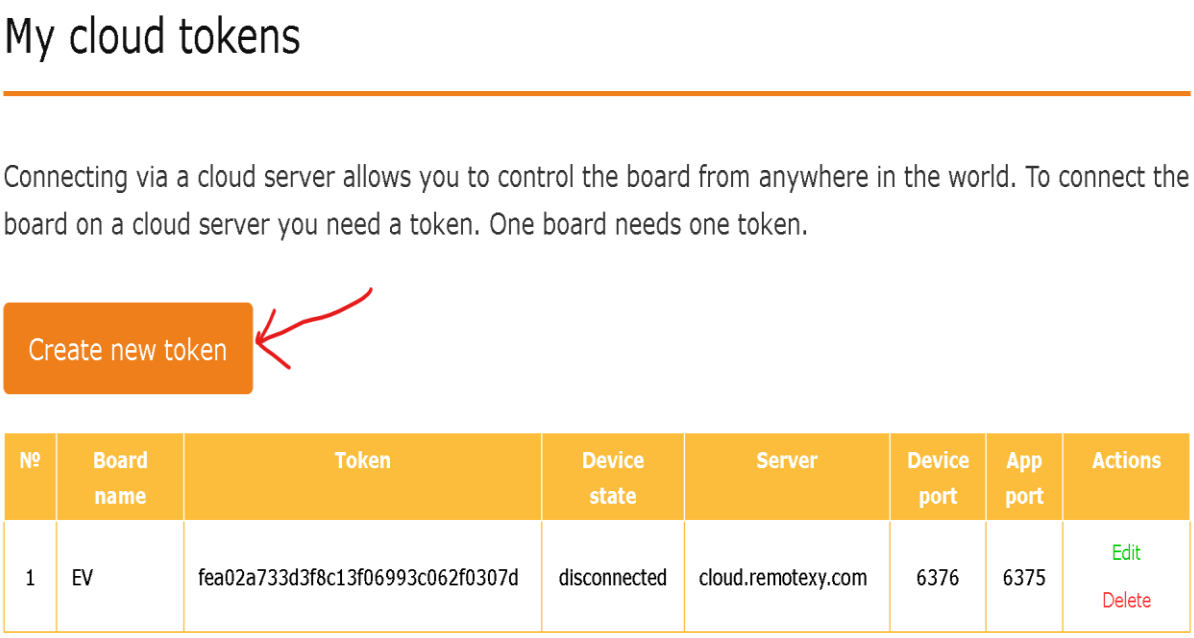
2.BAUD RATE -> 9600 bits per second

c) REMOTEXY CLOUD SERVER CREATION:

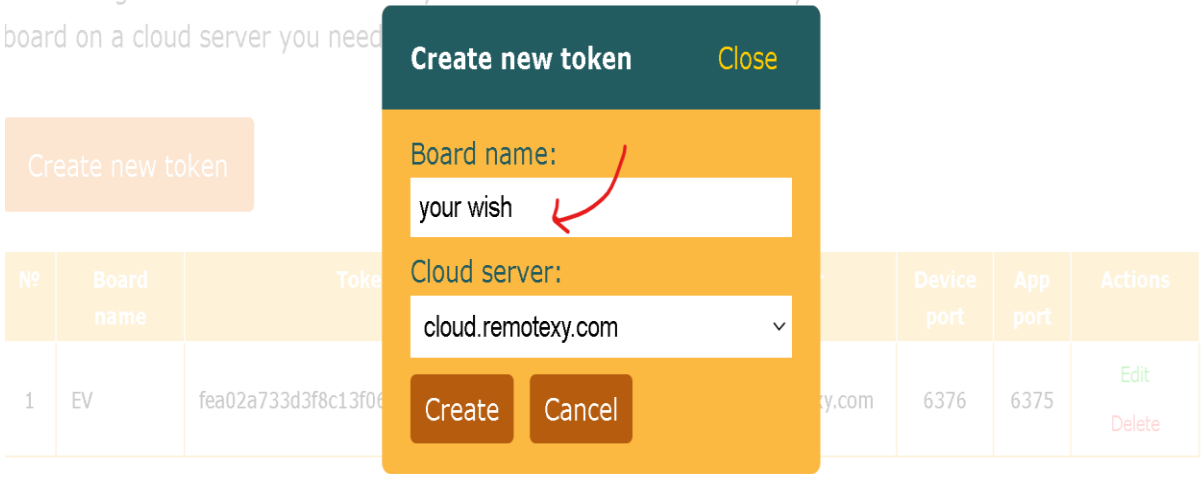
Step 1: Click on PROPERTIES > MODULE INTERFACE > CLOUD SERVER > MY TOKENS



Step 2: Click on create token



Step 3: Enter a board name and click create



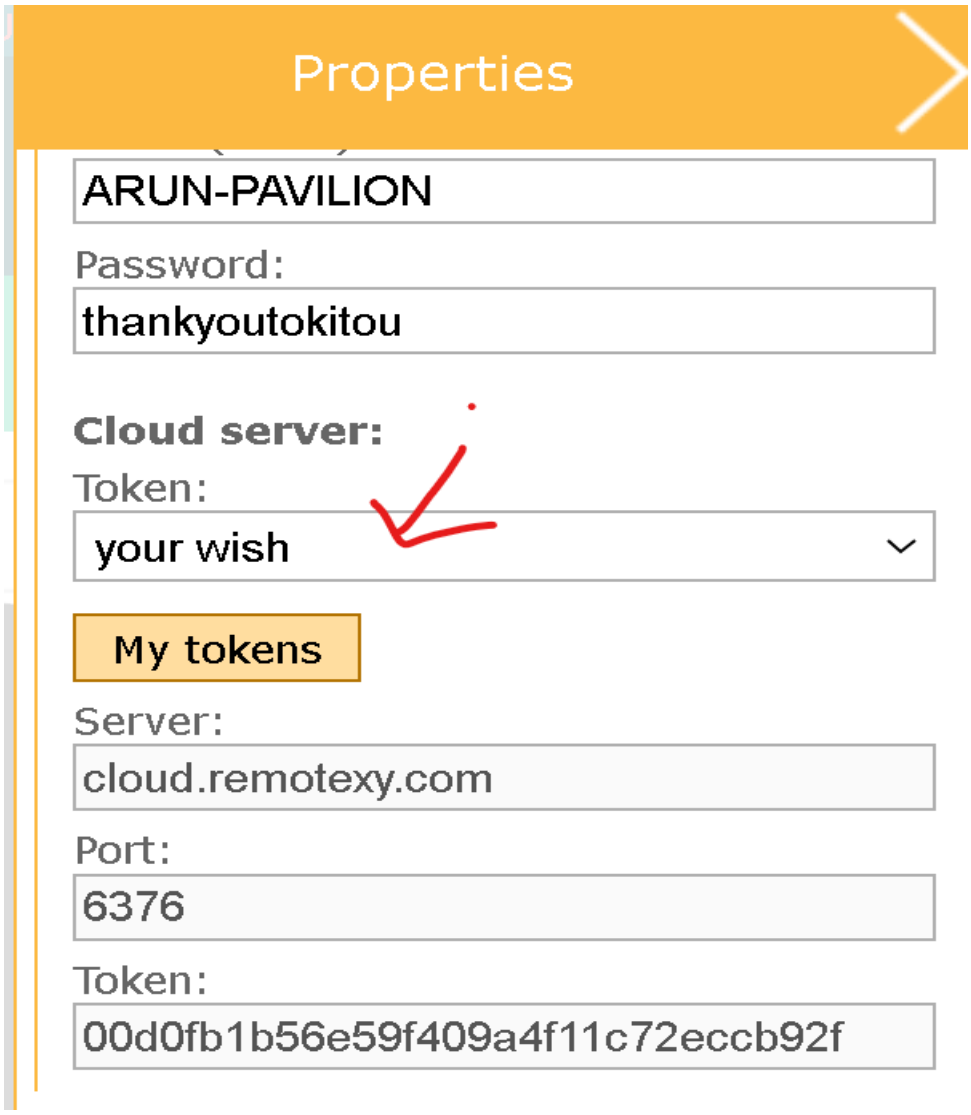
Step 4: Copy the token code and app port to enter into handheld device

Create new token

№	Board name	Token	Device state	Server	Device port	App port	Actions
1	EV	fea02a733d3f8c13f06993c062f0307d	disconnected	cloud.remotexy.com	6376	6375	Edit Delete
2	your wish	00d0fb1b56e59f409a4f11c72eccb92f	disconnected	cloud.remotexy.com	6376	6375	Edit Delete

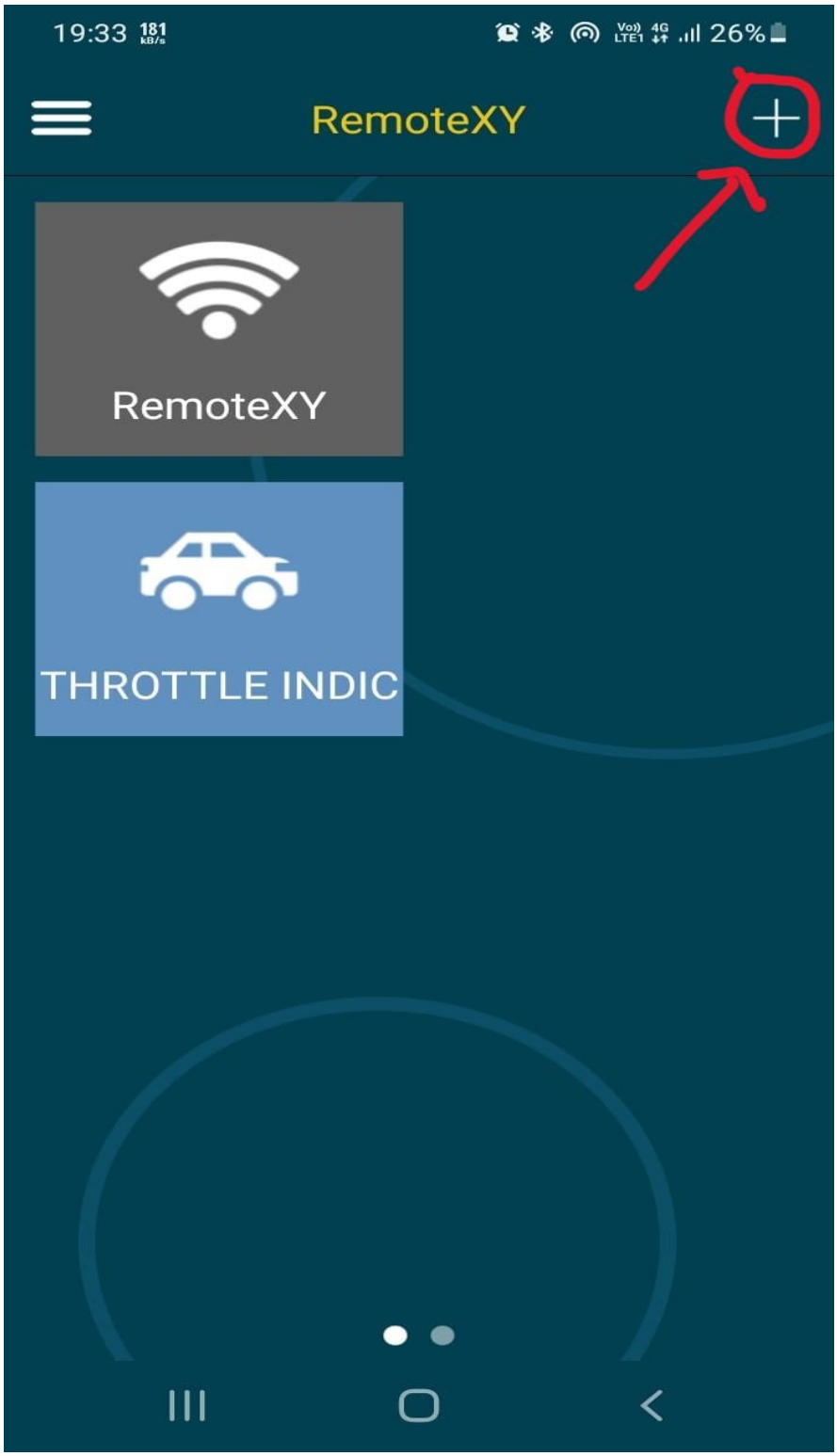
Copy this token

Step 5: Select the created token

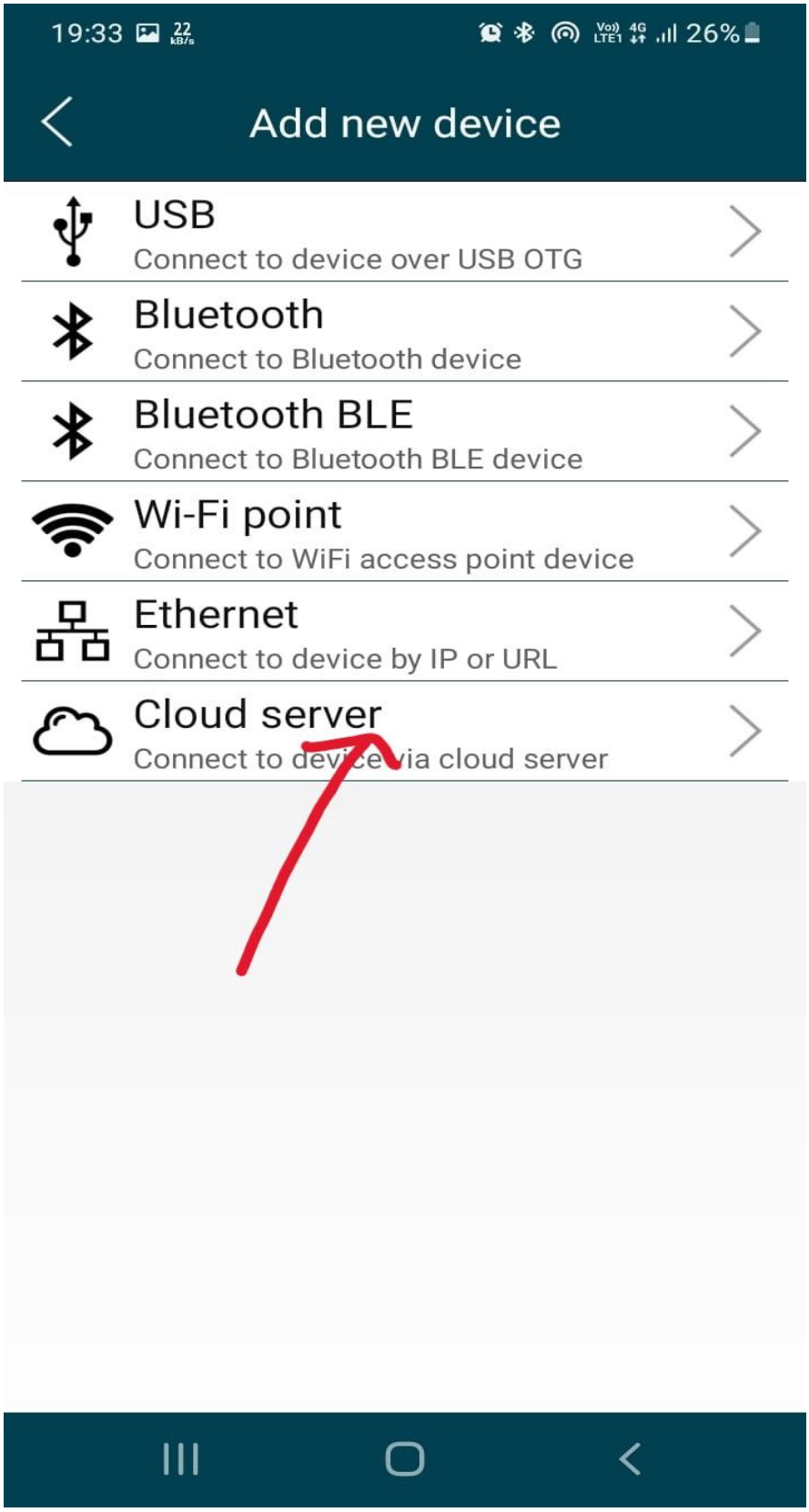


d) HANDHELD DEVICE SETUP:

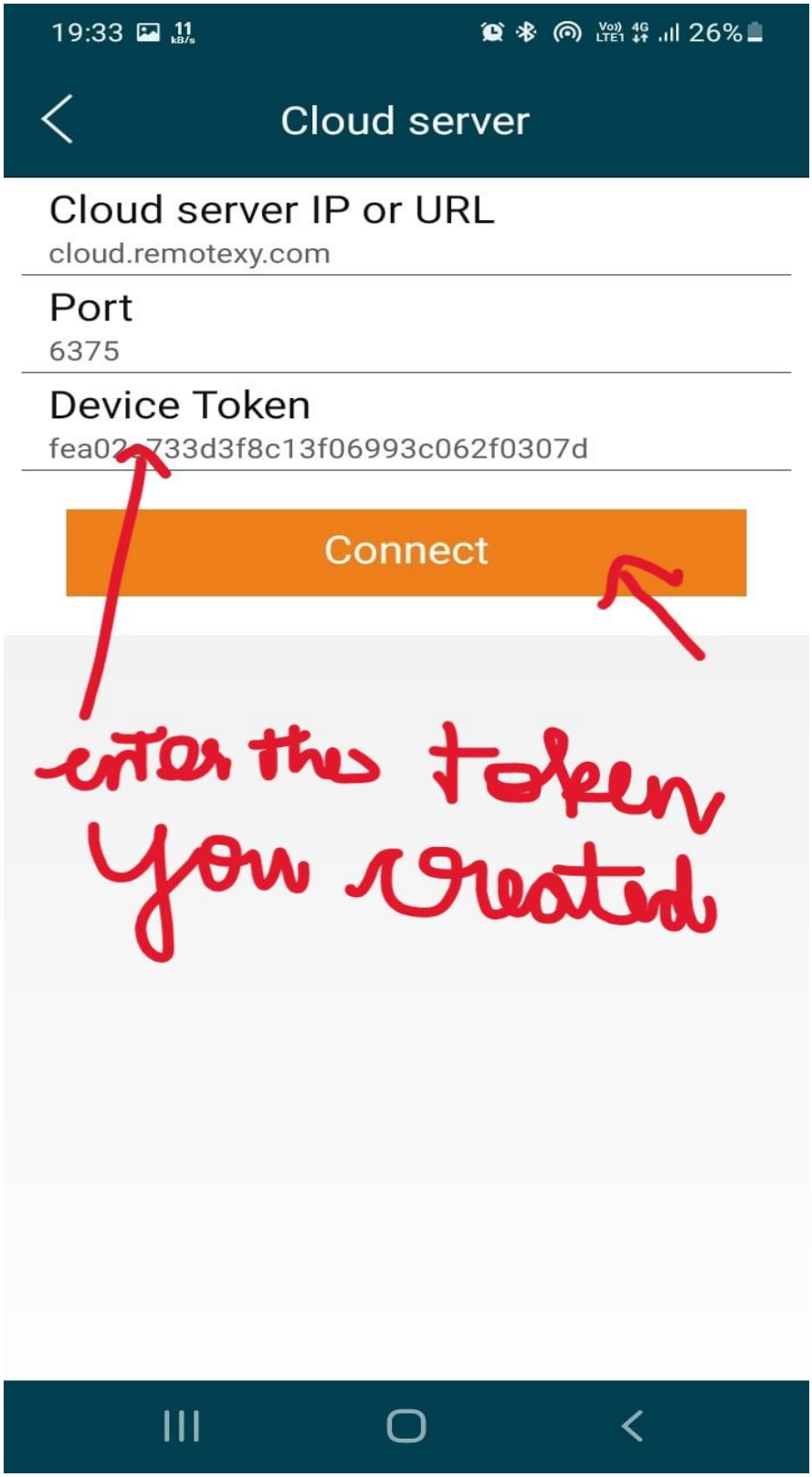
Step 1: Open RemoteXY app and select the plus button at the top right



Step 2: Click on Cloud Server



Step 3: Enter the cloud URL, Port and Device Token. Click connect



VIDEO OUTPUT RESULTS:

PROJECT FILES:

https://drive.google.com/drive/folders/1RN5Zp48HR9sYU5qhANGlrBA2D3XONe7u?usp=sharing

SERIAL PLOTTER:

https://drive.google.com/file/d/1RogiELwpbxLyFDIuV3yPZftIKWQmjnt-/view?usp=sharing

IOT CONTROL:

https://drive.google.com/file/d/1RXrcCTrvS5IUQw5AU2gZoZq8N2VFAuwj/view?usp=sharing

REFERENCES:

[1] ARDUINO DOCUMENTATION : https://docs.arduino.cc/

[2] RemoteXY DOCUMENTATION : https://remotexy.com/en/help/

[3] ESP8266 CONFIGURATION REFERENCE :
How to Connect ESP8266 to WiFi | A Beginner's Guide (electronicshub.org)

[4] ARDUINO KIT REFERENCE : https://quadstore.in/

[5] GITHUB LINK : https://github.com/P-Arun02/EV-throttle-monior-and-control