```
// article version 5/6/18
// Compile using Generic 8266
//RCv disabled alolcated as GPIO
// 0 button,2 output, 3 HC12 Set, 1 is still TX
/*
Using this to disable RCV
Serial.begin(115200,SERIAL_8N1,SERIAL_TX_ONLY);
//********** CHANGE PIN FUNCTION TO GPIO **********
//GPIO 1 (TX) swap the pin to a GPIO.
*/
#include <ESP8266WiFiMulti.h>
#include <WebSocketsClient.h> //  get it from https://github.com/Links2004/arduinoWebSockets/releases
#include <ArduinoJson.h> // get it from https://arduinojson.org/ or install via Arduino library manager
ESP8266WiFiMulti WiFiMulti;
WebSocketsClient webSocket;
WiFiClient client;
#define MyApiKey "6651f98d-938e-493b-a66b-0bc04xxxxxxxx" // TODO: Change to your sinric API Key. Your API Key
is displayed on sinric.com dashboard
#define MySSID "NETXXXXXXX" // TODO: Change to your Wifi network SSID
#define MyWifiPassword "ma@#$%^&*()" // TODO: Change to your Wifi network password

#define API_ENDPOINT "http://sinric.com"
#define HEARTBEAT_INTERVAL 120000 // 2 Minutes

uint64_t heartbeatTimestamp = 0;
bool isConnected = false,bounce=0,done=0,flag=0;
int  bcnt=10,temp=1,SETpin=3,j;
int  aa=0,lastdim=0,baseChan=50,lastdimh;
int  Dchan[5]={50,50,50,50,50},i=50;  // Dchan is sized for number of actions provided thru this controller
//
String gvalue="",DSTR="",gdeviceId="",gaction="";
String saddr="6a6c"; // uses last 4 characters of APIkey
float rtn;
//
void turnOn(String deviceId)
{
  if (gdeviceId == "5ae9d5d1fdc5543e490d2cbf") // Device ID of 1st device dimmer'
  {
    if (gvalue=="ON")
    {
     rtn=100;
    }
    if (gvalue=="OFF")
    {
     rtn=0;
    }
    dimmer(); //*
  }
  else if (deviceId == "5ae65fdefdc5543e496d4bea") // Device ID of second device 'outlet (attic) '
  {
     gdeviceId="SERV1";
     j=0;
     Setchan();
     sendRemote();
  }
  else if (deviceId == "5ae68294fdc5543e4978b299") // Device ID of first device  'fan hi ovrd'
  {
     gdeviceId="SERV2";
```

```
      j=1;
      Setchan();
      sendRemote();
    }
    else if (deviceId == "5ae682b7fdc5543e4978bdea") // Device ID of 'fan low ovrd'
    {
      gdeviceId="SERV3";
      j=2;
      Setchan();
      sendRemote();
    }
    else if (deviceId == "5ae68253fdc5543e49789cfc") // Device ID of 'profile'
    {
      gdeviceId="SERV4";
      j=3;
      Setchan();
      sendRemote();
    }
    // added
    else if (deviceId == "5ae68253fdc5543e499809") // Device ID of ' temp override'
    {
      gdeviceId="SERV5";
      j=4;
      Setchan();
      sendRemote();
    }
}
// ********************************
void turnOff(String deviceId)
{
   if (gdeviceId == "5ae9d5d1fdc5543e490d2cbf")
// Device ID of 1st device ف dimmer'
   {  // this is PWM local for GPIO0
      if (gvalue=="OFF")
      {
       rtn=0;
      }
      dimmer(); // this is a PWM local service on GPIO0
   }
   else if (gdeviceId == "5ae65fdefdc5543e496d4bea") // Device ID of 'attic'
   {
      gdeviceId="SERV1";
      j=0;
      Setchan();
      sendRemote();
   }
   else if (deviceId == "5ae68294fdc5543e4978b299") // Device ID of fan Hi ovrd'
   {
      gdeviceId="SERV2";
      j=1;
      Setchan();
      sendRemote();
   }
   if (deviceId == "5ae682b7fdc5543e4978bdea") // Device ID of 'Fan Low ovrf'
   {
      gdeviceId="SERV3";
      j=2;
      Setchan();
```

```
      sendRemote();
   }
   else if (deviceId == "5ae68253fdc5543e49789cfc") // Device ID of 'profile'
   {
      gdeviceId="SERV4";
      j=3;
      Setchan();
      sendRemote();
   }
// added trip override 'OFF' Not allowed
   else
   {
    // for unknown
   }
}
void webSocketEvent(WStype_t type, uint8_t * payload, size_t length) {
  switch(type) {
    case WStype_DISCONNECTED:
      isConnected = false;
//      Serial.printf("[WSc] Webservice disconnected from sinric.com!\n");
      break;
    case WStype_CONNECTED: {
      isConnected = true;
//      Serial.printf("[WSc] Service connected to sinric.com at url: %s\n", payload);
//      Serial.printf("Waiting for commands from sinric.com ...\n");
      }
      break;
    case WStype_TEXT: {
//        Serial.printf("[WSc] get text: %s\n", payload);
        // Example payloads
        // For Light device type
        // {"deviceId": xxxx, "action": "setPowerState", value: "ON"} // https://developer.amazon.com/docs/device-apis/alexa-powercontroller.html
        // {"deviceId": xxxx, "action": "AdjustBrightness", value: 3} // https://developer.amazon.com/docs/device-apis/alexa-brightnesscontroller.html
        // {"deviceId": xxxx, "action": "setBrightness", value: 42} // https://developer.amazon.com/docs/device-apis/alexa-brightnesscontroller.html
        // {"deviceId": xxxx, "action": "SetColor", value: {"hue": 350.5,  "saturation": 0.7138, "brightness": 0.6501}} // https://developer.amazon.com/docs/device-apis/alexa-colorcontroller.html
        // {"deviceId": xxxx, "action": "DecreaseColorTemperature"} // https://developer.amazon.com/docs/device-apis/alexa-colortemperaturecontroller.html
        // {"deviceId": xxxx, "action": "IncreaseColorTemperature"} // https://developer.amazon.com/docs/device-apis/alexa-colortemperaturecontroller.html
        // {"deviceId": xxxx, "action": "SetColorTemperature", value: 2200} // https://developer.amazon.com/docs/device-apis/alexa-colortemperaturecontroller.html

        DynamicJsonBuffer jsonBuffer;
        JsonObject& json = jsonBuffer.parseObject((char*)payload);
        String deviceId = json ["deviceId"];
        gdeviceId=deviceId; // string copy of returned id
        String action = json ["action"];
        gaction = action;
        String value = json ["value"];
        gvalue=value;       // string copy of returned value
        rtn=gvalue.toFloat(); // number of return value

// at this point you must decide if you are going to code as if a local control or remote link
// ... which deviceId matches which process. the following sorts by the passed 'action' parameter
```

```
      if(action == "setPowerState") { //  for Switch
         if(value == "ON") {
            turnOn(deviceId);
         } else {
            turnOff(deviceId);
         }
      }
      else if((action == "SetBrightness") || (action == "AdjustBrightness") ) // this type is for lamp brightness/change
      { // different actions have different constraints on the passed values
         //String value = json ["value"];
         turnOn(deviceId);
      }
      else if (action == "test")
      {
//          Serial.println("[WSc] received test command from sinric.com");
      }
     }
     break;
   case WStype_BIN:
//     Serial.printf("[WSc] get binary length: %u\n", length);
     break;
  }
}
// ****************************
void setup()
{ //*
  Serial.begin(9600,SERIAL_8N1,SERIAL_TX_ONLY); //disable RCV and set baud rate 115200 or (9600) HC12 rate
  //Serial.begin(115200);   // leave this out, delete references to print and  gpio pin1 and 3 are available directly off the ESP-
01
  WiFiMulti.addAP(MySSID, MyWifiPassword);
  delay(100);
//  Serial.println();
//  Serial.print("Connecting to Wifi: ");
//  Serial.println(MySSID);
// Waiting for Wifi connect
  while(WiFiMulti.run() != WL_CONNECTED)
  {
    delay(500);
//    Serial.print(".");
  }
  if(WiFiMulti.run() == WL_CONNECTED)
  {
//*   Serial.println("");
//    Serial.print("WiFi connected. ");
//    Serial.print("IP address: ");
//    Serial.println(WiFi.localIP());
  }
  // server address, port and URL
  webSocket.begin("iot.sinric.com", 80, "/");
  // event handler
  webSocket.onEvent(webSocketEvent);
  webSocket.setAuthorization("apikey", MyApiKey); // <-enter key
  // try again every 5000ms if connection has failed
  // If you see 'class WebSocketsClient' has no member named 'setReconnectInterval' error update arduinoWebSockets
  webSocket.setReconnectInterval(5000);
//GPIO 3 (RX) swap the pin to a GPIO 3.
  pinMode(3, FUNCTION_3); // not sure if this is necessary
  pinMode(0,INPUT_PULLUP);
```

```
  digitalWrite(2,HIGH);
  pinMode(2,OUTPUT);
  digitalWrite(3,HIGH);
  pinMode(3,OUTPUT); // now HC12 set pin
}
// ************************************************************
//  subservice(s)
void dimmer() // passed as 0-100
{
  aa=1023-(rtn*1023/100);     // adjust for low true PWM
  if(gaction == "setPowerState")
  {
    analogWrite(3,aa);
  }
  else
  {
    fadethis(); // fades into and outof lastdim to aa
  }
  lastdim=aa;
}
// ************************
void fadethis()
{
  bool flg=0;
  while (!flg)
  {
    int temp=abs((lastdim-aa)/2);
    if (temp>15)
    {
      if(aa>lastdim)
      {
        temp=lastdim+temp;
      }
      else
      {
        temp=lastdim-temp;
      }
      lastdim=temp;
      analogWrite(3,temp);
      delay(200);
    }
    else
    {
      analogWrite(3,aa);
      flg=1;
    }
  }
}
// ************************************
void loop()
{
  aa=digitalRead(0);
  if (aa==temp)
  {
    if (bounce)
    {
      bcnt--;
      if (bcnt==0  && !done) // not serviced and debounced
```

```
      {             // service button(s)
       if (aa==1) //3)
       {  // released
         bounce=0;
       }
       else if(aa==2)
       { //GPIO0 asserted

          if(lastdim<665) // assume low true : power set >~ 35%
          {
            analogWrite(3,1023); // set off(low true)
            lastdimh=lastdim;
            lastdim=1023;
            flag=1;
          }
          else
          {
           analogWrite(3,lastdimh); // set on(low true) or 0
           lastdim=0;
           flag=0;
          }
       }
       done=1;
       bcnt++;
      }
    }
  }
  else // not same
  {
     bounce=1;
     bcnt=15;
     temp=aa;
     done=0;
  }
  webSocket.loop();
  if(isConnected)
  {
     uint64_t now = millis();
     // Send heartbeat in order to avoid disconnections during ISP resetting IPs over night. Thanks @MacSass
     if((now - heartbeatTimestamp) > HEARTBEAT_INTERVAL)
     {
        heartbeatTimestamp = now;
        webSocket.sendTXT("H");
     }
  }
  delay(2);

}
// ************************
void sendRemote()   //send sendM to remote
{
  DSTR="E50,"+saddr+","+ gdeviceId+ "," + gvalue + ","+ char(0x5E);
  // eg:  E50,0,6a6c,SERVn,ON,^   ; n=1 to number of service types
  flushthis();
  Serial.println(DSTR); // set to transmit of HC12
}
//*****************************
void Setchan() // pass channel number in i
```

```
{
   if(baseChan != Dchan[j])
   {
    i=Dchan[j];
    if (i<10)
    {
       DSTR="AT+C00"+String(i);
    }
    else if (i<100)
    {
      DSTR="AT+C0"+String(i);
    }
    else
    {
     DSTR="AT+C"+String(i);
    }
    digitalWrite(SETpin,LOW);
    delay(40);
    Serial.print(DSTR); // was HC12Link.print(DSTR);
    digitalWrite(SETpin,HIGH);
    Serial.println();     // was HC12Link.println();
    delay(80);
    baseChan=i;
    flushthis();
   }
}
// **********************
void flushthis()
{
  while(Serial.available()){Serial.read();} // flush
}

// ***************************

/* NOTES
 *
 *   pinMode(LED_BUILTIN, x )  will reference pin 1 for the esp01, this pin also is TX !
Analog output
analogWrite(pin, value) enables software PWM on the given pin.
Call analogWrite(pin, 0) to disable PWM on the pin. PWM range may be changed by calling
analogWriteRange(new_range).
PWM frequency is 1kHz by default. Call analogWriteFreq(new_frequency) to change the frequency.

*/
```