Setup & Operation

Table of Contents

1. Introduction

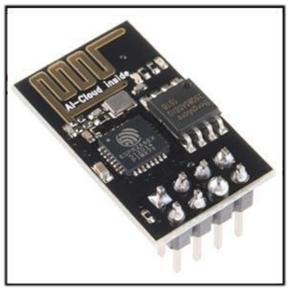
The following document contains information about three Arduino Uno examples, one about Raspberry Pi, one about NodeMcu, and one about Wemos D1 R2. All are relatively cheap, ubiquitous open-source hardware that are programmed with free open-source software, and enjoy a wealth of information and documentation over the web. The Raspberry Pi is a single-board-computer (www.raspberrypi.org) that features an operating system, and its example is programmed with Python language in the Raspberry Pi itself. All the other examples are programmed with C language in the Arduino IDE (www.arduino.cc). The examples depict configuring both hardware and software to connect to the Takano cloud-based platform, exchange data and send emails and push notifications via http(s) based commands. The user has first to register on the takano platform, get the hardware registration key, and can use the Takano app (freely available on both Ios and Android) to send data back and forth with the hardware. Information on how to setup and use the Takano platform is available at: http://iotsprint.com/takano_setup/takano_platform.pdf.
Information on how to setup and run the Takano mobile app is available at: http://iotsprint.com/takano_setup/takano_app.pdf.
Please make you read these two documents before going after the examples herein. The examples here do not teach Arduino or Raspberry Pi programming but are rather intended for users who already have some basic experience in programming and handling them, but would like to connect them to the cloud and remotely control/gauge them, without having to neither write server script nor worry about server hosting or mobile apps. You are urged to follow the examples exactly as they are presented below. An http link is provided for every compiler code used. Use this link to easily copy and paste the program into the proper Software (Arduino or Python IDE). Also don’t forget to change SSID, password and registration key to match yours.

2. Arduino UNO Example#1

The first step is to procure the items used in the experiment, these are:

- Arduino Uno with its USB cable
- 2x16 LCD (Hitachi HD44780 compatible)
- ESP8266 Wi-Fi module
- LM35 temperature sensor
- Breadboard
- 10Kohm potentiometer
- 10Kohm, 2.2Kohm, 1Kohm resistors
- A bunch of connector cables

Please start by connecting the different items exactly as per Fig 1. One particular attention is to the ground pin of the LM35: it must be connected to a ground separate from the other items. Connecting it to the ground of the breadboard will result in erroneous reading. It provides a linear 10mV/°C on its output pin. It requires no additional components and hence can be directly connected to any of the Arduino analogue inputs (0 to +5V). The ESP8266 is a serial to Wi-Fi modem controlled by AT commands sent to it by an external controller (Arduino in our case) to its serial port, shown in the pic below. It should be set at a baud rate of 9600 for this particular example. If its baud rate is 115200, the user has to connect it to a USB to serial board, such as Sparkfun USB to serial Breakout board, open a terminal on the PC and send any of the following commands, depending on the ESP8266 firmware: AT+CIOBAUD, AT+IPR, or AT+UART_DEF. More information is available on the web on how to change the ESP8266 baud rate. Please note that the ESP8266 is powered by +3.3V which is fortunately provided by the Arduino Uno board. Supplying the ESP8266 with +5V will burn it! Since the Arduino itself requires a +5V, a voltage divider is required to interface the RX of the ESP8266 to the TX of Arduino (1Kohm and 2.2Kohm on the breadboard). Arduino serial RX is able on the other hand to directly interface with the TX pin of ESP8266.


ESP8266 Wi-Fi module

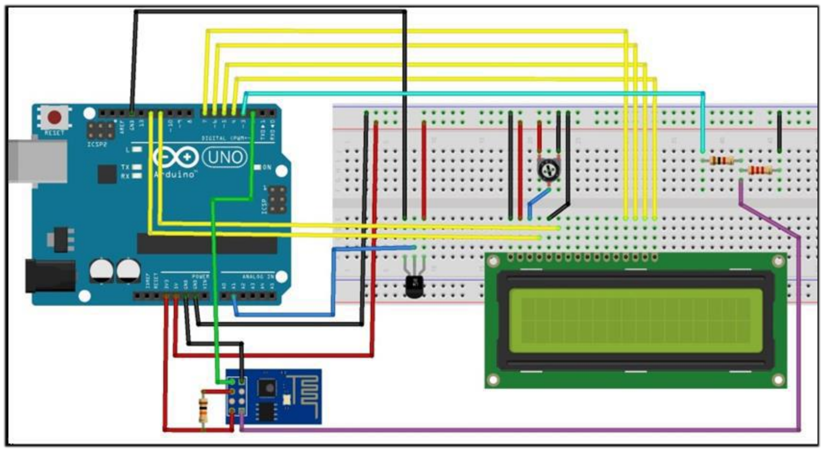


Fig 1.

The Arduino Uno has a single serial port (pin0 as RX and pin1 as TX), which is also directly connected to the PC programming cable. I chose not to connect it to the ESP8266 and hence to leave it for program troubleshooting and use other Arduino pins not dedicated for serial use to connect to the ESP8266 instead. These are pin2 and pin3, which are mimicking a serial port in software. The downside of this approach is that the baud rate cannot reach 115200. This is the main reason why the ESP8266 baud was fixed at 9600.. Now power up the connected assembly and adjust the potentiometer for

best LCD contrast. You should of course have already downloaded/installed the latest Arduino IDE from https://www.arduino.cc/en/Main/Software.
Now press on this link to Copy/Paste the below code to the Arduino IDE and save it by any name.
Don't forget to change SSID, password and registration key in the declaration section of the below Arduino program to match those of your network.

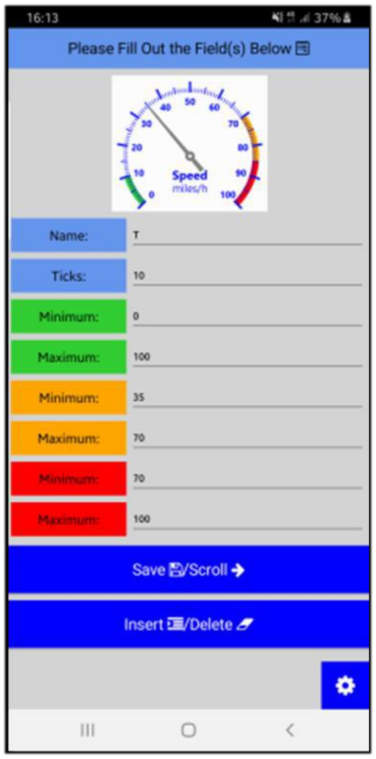
```
#include <SoftwareSerial.h> //needed for mimicking serial port on normal I/O pins
#include <LiquidCrystal.h> //needed for LCD control
#define RX 2 //pin2 of Arduino is serial RX pin
#define TX 3 //pin3 of Arduino is serial TX pin

// initialize the library by associating any needed LCD interface pin
// with the arduino pin number it is connected to
const int rs = 12, en = 11, d4 = 7, d5 = 6, d6 = 5, d7 = 4;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

String AP = "TPLINK_90210";       // SSID name, enter your own  network name
String PASS = "Abracadabra@2"; // and enter your network Password
String REG_KEY = "witjmvmox0x9";   // hardware registration key on takano (do not confuse with
user registration key!)
String HOST = "iotsprint.com"; //website Arduino should send http request to
String PORT = "80"; //default port

int countTrueCommand;
int countTimeCommand;
boolean found = false;
int tempPin = 1;
int timeout = 0;
int first, last, first1, last1;
String stringTwo, stringThree, data, data1, data2, content;
SoftwareSerial esp8266(RX, TX);

void negotiate_modem() //sequence of AT commands from Arduino to ESP8266
{  //this function allows the Arduino to connect to the router
  sendCommand("AT+CWMODE=1", 50, "OK");
  sendCommand("AT+CIPMUX=1", 50, "OK");
  sendCommand("AT+CWJAP=\"" + AP + "\",\"" + PASS + "\"", 150, "OK");
}
void setup() {
  Serial.begin(115200); //serial port between Arduino and PC
  esp8266.begin(9600); //software serial port between Arduino and ESP8266
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print(" takano example");
  delay(2000);
  sendCommand("AT", 50, "OK");
  lcd.clear();
}
```

```
void loop() {
  //read the LM35 temperature sensor analog input on analogue pin1
  data = "";
  data1 = "";
  data2 = "";
  content = "";
  if (found == false) negotiate_modem(); //if Arduino did not get a proper string back from cloud,
then try to connect to modem again. Maybe power was recyled to modem for example..
  float celsius = (500.0 * analogRead(tempPin)) / 1024.0; //5V or 5000mV correspond to 1024
steps of 10-bit AD converter, and LM35 increases by 10mV per degree C..
  stringTwo = String(celsius);
  stringThree= String(stringTwo);
  if (stringTwo.length() == 5) stringThree = stringTwo.substring(0, stringTwo.length() - 1); //trim it
to 4 characters
  if (stringThree.length() < 2) stringThree = "000" + stringThree; //to make sure it has 4 characters
length
  else if (stringThree.length() < 3) stringThree = "00" + stringThree; //to make sure it has 4
characters length
  else if (stringThree.length() < 4) stringThree = "0" + stringThree; //to make sure it has 4
characters length
  stringThree = "T=" + stringThree; // stringThree = “T=23.4” for example
  lcd.setCursor(0, 0);
  lcd.print(stringThree);
  String getData = "GET /tk.php?r="+REG_KEY+"&m=" + stringThree + " HTTP/1.0";
sendCommand("AT+CIPSTART=4,\"TCP\",\"iotsprint.com\",80", 10, "OK");
sendCommand("AT+CIPSEND=4,71", 10, ">"); //this command is crucial, read at the end of the
program
  esp8266.println(getData); delay(2000); countTrueCommand++;
  read_cloud("Host:iotsprint.com:80");
  delay(3000);
}

void read_cloud(String command)
{
  data = "";
  data1 = "";
  esp8266.println(command);
  esp8266.println();
  timeout = 0;
  while (Serial.available() == 0) {
    if ( ++timeout > 10000) { // set this to your timeout value in milliseconds
      Serial.print(". at command => ");
      break;
    }
  }
  timeout = 0; // got a char so reset timeout
  content = esp8266.readString(); // read the incoming data as string
  Serial.print(content);
  if (content.indexOf("L=") >= 0) //slider value is 0 to 100
  {
    first = content.indexOf("L=") + 2;
    last = content.lastIndexOf("S=");
    data = content.substring(first, last - 1);
  }
  if (content.indexOf("S=") >= 0) //switch value is either 0 (off) or 1 (on)
  {
    first1 = content.indexOf("S=") + 2;
    last1 = first1 + 1;
    data1 = content.substring(first1, last1);
  }
  if ((data != "") && (data1 != ""))
  {
    data2 = "S="+data1+" "+"L=" +data+"      ";
    lcd.setCursor(0, 1);
    lcd.print(data2);
  }
}

void sendCommand(String command, int maxTime, char readReplay[]) {
  found = false;
  Serial.print(countTrueCommand);
  Serial.print(". at command => ");
  Serial.print(command);
  Serial.print(" ");
  while (countTimeCommand < (maxTime * 1))
  {
    esp8266.println(command);
    if (esp8266.find(readReplay)) //ok
    {
      found = true;
      break;
    }
    countTimeCommand++;
  }
  if (found == true)
  {
    Serial.println("OK");
    countTrueCommand++;
    countTimeCommand = 0;
  }
  if (found == false)
  {
    Serial.println("FAIL");
    countTrueCommand = 0;
    countTimeCommand = 0;
  }
}
```

The number ‘71’ in the command “AT+CIPSEND=4,71” determines the number of characters that should be send to the ESP8266. These are:

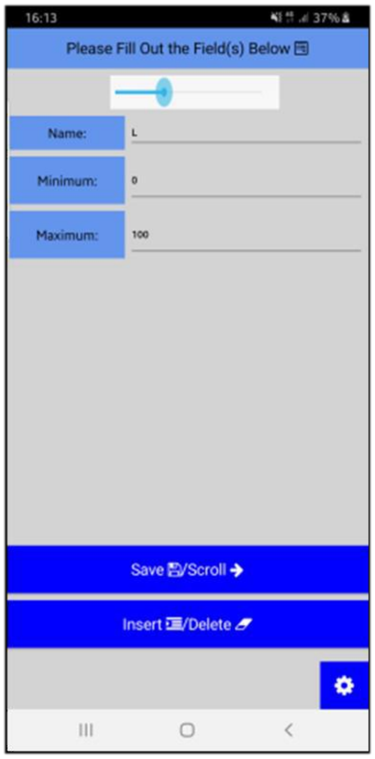
"GET /tk.php?r=witjmvmox0x9&m=T=19.2 HTTP/1.0" :44 characters+linefeed+carriage return = 46 characters
"Host:iotsprint.com:80" :21 characters+linefeed+carriage return+linefeed+carriage return= 25 characters
So the total is 71 characters (the println function prints a newline and a carriage return, so actually it counts as two characters..)

The code has been written more for readability and less for optimization. Now upload it to Arduino, and it everything goes fine, the LCD should display 'takano example' for few seconds before showing the LM35 temperature on line1. Arduino will then negotiate the SSID and Password that were entered in the program. If they are correct and the Arduino is within the router range, then on line2 you should see 'S=0 L=0'. S is an on/off control, and L is an analogue control value sent by the Takano app.
In order to start reading the temperature sensor on your phone app, and in parallel send commands from the app to Arduino, download Takano app and enter three fields, a Gauge 270, a linear slider and a checkbox in the same order as presented in the below pics.



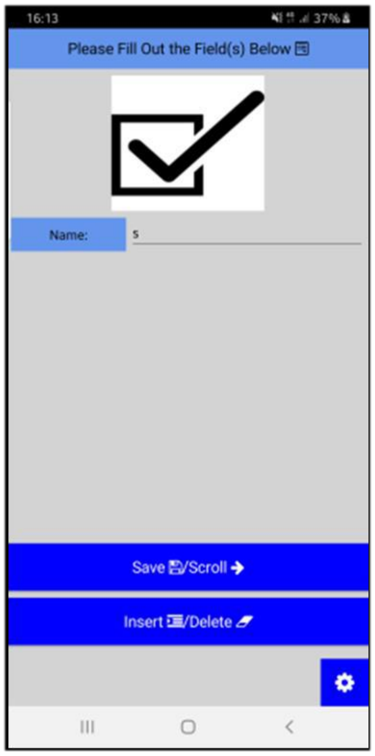
Name:	T
Ticks:	10
Minimum:	0
Maximum:	100
Minimum:	35
Maximum:	70
Minimum:	70
Maximum:	100

Fig 2.



Name:	L
Minimum:	0
Maximum:	100

Fig 3.



Name:	S

Fig 4.

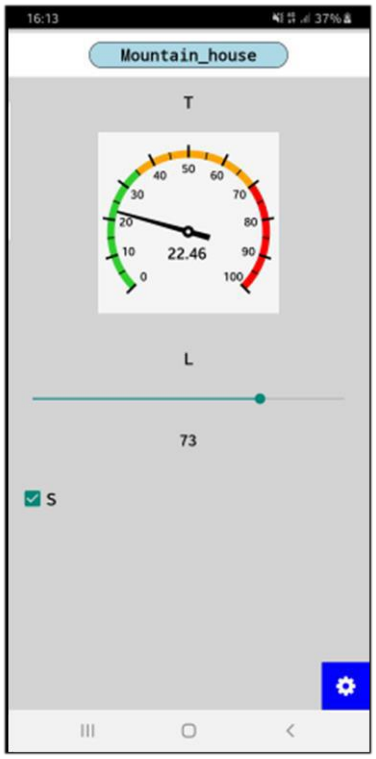


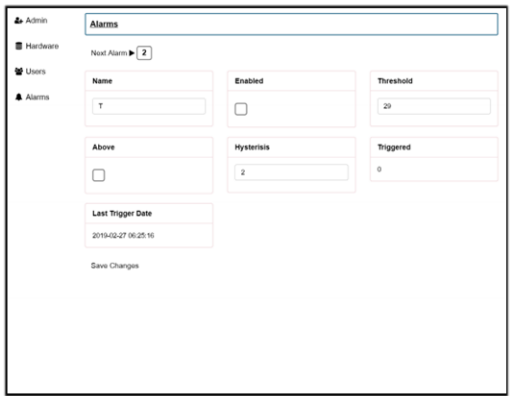
Fig 5.

The name of each graphic interface must exactly match that in the Arduino program, that is capital T, L and S, otherwise the app won't be able to exchange data with the arduino via the Takano cloud. The setup and operation instructions for the Takano app are found in another pdf on the iotsprint website. The running page should look like Fig 5. In parallel, go to 'Tools' in the Arduino IDE and select 'Serial Monitor'. This will open a window that will allow you to track the packets of data between Arduino and ESP8266, as well as the packets between ESP8266 and the Takano cloud. Make sure 115200 as baud rate is selected at the bottom right corner of the window. The Arduino sends the temperature to the cloud in the stringTwo variable through the http request: "GET /tk.php?r="+REG_KEY+"&m=" + stringTwo + " HTTP/1.0", and gets the S and L commands from the cloud after it sends the request: "Host:iotsprint.com:80\n".
On the serial monitor, a packet of the form:

```
+IPD,4,145:HTTP/1.1 200 OK
Server: nginx/1.14.1
Content-Type: text/html; charset=UTF-8
Connection: close
```

L=73,S=14,CLOSED

Indicates the cloud response where, L has a value of 73 and S has a value of 1. “4,CLOSED” is the closing message string. The Arduino program looks for the location of “L=” and “S=”. Once both are found, then the value of L is between “L=” and up to one character before “S=”, whereas the value of S is one character after “S=” etc… String manipulation functions inherent to the Arduino programming language are used to localize them (such as indexOf, lastIndexOf and substring). When setting up the Takano platform, you can choose a threshold limit for the variables the hardware is sending to the platform, that when exceeded, an email will be sent to the user. You can for example set a ‘29’ limit for T, and set it Above in the Alarms page as in Fig 6.



Name	Enabled	Threshold
T	☐	29
Above	Hysterisis	Triggered
☐	2	0
Last Trigger Date		
2019-02-27 06:25:16		

Fig 6.

3. Arduino UNO Example#2

This example is a slightly modified version of the above one. Here the ESP8266 is maintained at its default baud rate of 115200, and the Arduino Uno in this case uses its only serial port, namely pin0 (RX) and pin1 (TX) to connect to it. Note that prior to uploading the program to Arduino, you should remove RX and TX connectors to the ESP8266, otherwise the sketch upload will be unsuccessful. Fig 7 depicts the schematic.

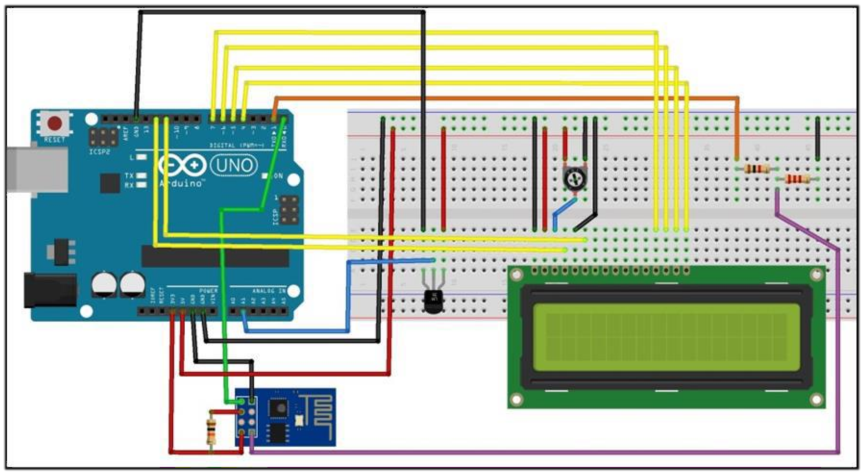


Fig 7.

Press on this link to Copy/Paste the below code to the Arduino IDE and save it by any name:

```
#include <LiquidCrystal.h> //needed for LCD control

// initialize the library by associating any needed LCD interface pin
// with the arduino pin number it is connected to
const int rs = 12, en = 11, d4 = 7, d5 = 6, d6 = 5, d7 = 4;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

String AP = "D_LINK_259701";      // SSID name, enter your own  network name
String PASS = "Abracadabra@2"; // and enter your network Password
String REG_KEY = "witjmvmox0x9";   // hardware registration key on takano (do not confuse with
user registration key!)
String HOST = "iotsprint.com"; //website Arduino should send http request to
String PORT = "80"; //default port

int countTrueCommand;
int countTimeCommand;
boolean found = false;
int tempPin = 1;
int timeout = 0;
int first, last, first1, last1;
String stringTwo, stringThree, data, data1, data2, content;

void negotiate_modem() //sequence of AT commands from Arduino to ESP8266
{  //this function allows the Arduino to connect to the router
  sendCommand("AT+CWMODE=1", 50, "OK");
  sendCommand("AT+CIPMUX=1", 50, "OK");
  sendCommand("AT+CWJAP=\"" + AP + "\",\"" + PASS + "\"", 150, "OK");
}
void setup() {
  Serial.begin(115200);
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print(" takano example");
  delay(2000);
  sendCommand("AT", 50, "OK");
  lcd.clear();
}

void loop() {
  //read the LM35 temperature sensor analog input on analogue pin1
  data = "";
  data1 = "";
  data2 = "";
  content = "";
  if (found == false) negotiate_modem(); //if Arduino did not get a proper string back from cloud,
then try to connect to modem again. Maybe power was recyled to modem for example..
  float celsius = (500.0 * analogRead(tempPin)) / 1024.0; //5V or 5000mV correspond to 1024
steps of 10-bit AD converter, and LM35 increases by 10mV per degree C..
  stringTwo = String(celsius);
  stringThree= String(stringTwo);
  if (stringTwo.length() == 5) stringThree = stringTwo.substring(0, stringTwo.length() - 1); //trim it
to 4 characters
  if (stringThree.length() < 2) stringThree = "000" + stringThree; //to make sure it has 4 characters
length
  else if (stringThree.length() < 3) stringThree = "00" + stringThree; //to make sure it has 4
characters length
  else if (stringThree.length() < 4) stringThree = "0" + stringThree; //to make sure it has 4
characters length
  stringThree = "T=" + stringThree; // stringThree = “T=23.4” for example
  lcd.setCursor(0, 0);
  lcd.print(stringThree);

  String getData = "GET /tk.php?r="+REG_KEY+"&m=" + stringThree + " HTTP/1.0";
  sendCommand("AT+CIPSTART=4,\"TCP\",\"iotsprint.com\",80", 10, "OK");
  sendCommand("AT+CIPSEND=4,71", 10, ">");
  Serial.println(getData); delay(2000); countTrueCommand++; //46
  read_cloud("Host:iotsprint.com:80");  //25
  delay(3000);
}

void read_cloud(String command)
{
  data = "";
  data1 = "";
  Serial.println(command);
  Serial.println();
  timeout = 0;
  while (Serial.available() == 0) {
   if ( ++timeout > 10000) { // set this to your timeout value in milliseconds
     break;
   }
```

```
    }
    timeout = 0; // got a char so reset timeout
    content = Serial.readString(); // read the incoming data as string

    if (content.indexOf("L=") >= 0) //slider value is 0 to 100
    {
      first = content.indexOf("L=") + 2;
      last = content.lastIndexOf("S=");
      data = content.substring(first, last - 1);
    }
    if (content.indexOf("S=") >= 0) //switch value is either 0 (off) or 1 (on)
    {
      first1 = content.indexOf("S=") + 2;
      last1 = first1 + 1;
      data1 = content.substring(first1, last1);
    }

    if ((data != "") && (data1 != ""))
    {
      data2 = "S="+data1+" "+"L=" +data+"      ";
      lcd.setCursor(0, 1);
      lcd.print(data2);
    }
  }

  void sendCommand(String command, int maxTime, char readReplay[]) {
    found = false;
    while (countTimeCommand < (maxTime * 1))
    {
      Serial.println(command);
      if (Serial.find(readReplay)) //ok
      {
        found = true;
        break;
      }
      countTimeCommand++;
    }

    if (found == true)
    {
      countTrueCommand++;
      countTimeCommand = 0;
    }

    if (found == false)
    {
      countTrueCommand = 0;
      countTimeCommand = 0;
    }
  }
```

The program flow of the previous example is valid for this example as well. Also the same applies to the phone app.

4. Arduino UNO Example#3

In this example, we will use a DHT11 temperature+humidity sensor, and a photoresistor light intensity sensor. In addition we will control a small servo motor remotely. The circuit is shown in Fig 8. Note that you will definitely need to use an external 9 to 12V/1A power supply to make up for the additional power constraints imposed but the servo motor and not to stretch the USB supply. Here the ESP8266 baud rate should be set to 9600, and the Arduino Uno in this case uses its only serial port, namely pin0 (RX) and pin1 (TX) to connect to it. Note that prior to uploading the program to Arduino, you should remove RX and TX connectors to the ESP8266, otherwise the upload won't be successful. And better press on the Arduino Uno reset button after successful upload.

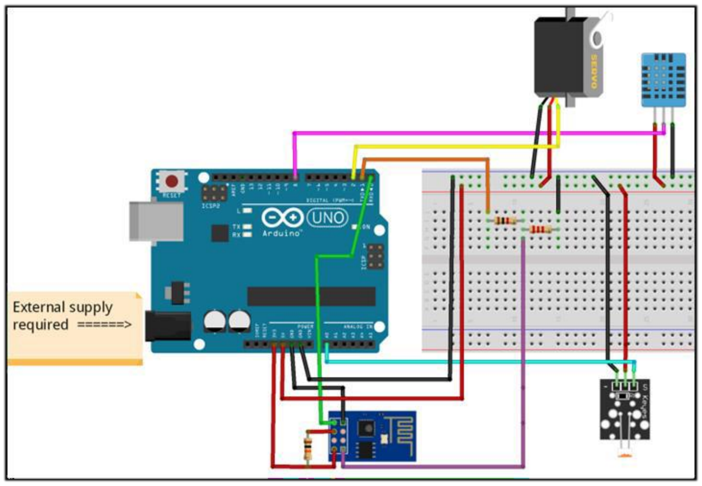
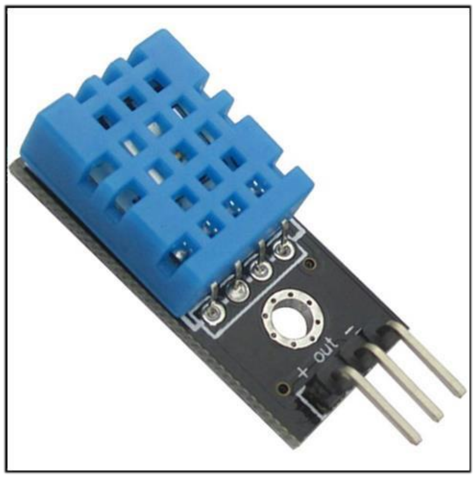


Fig 8.

The servo motor we used is pictured below. It has 3 wires, one for +5V (red), one for ground (brown) and one for signal (yellow) connected to Arduino pin2. Make sure you get a similar one otherwise it won't run directly from the Arduino Uno board if it is bigger, and would require a separate driver and power source. Refer to the datasheet of the one you have as not all servo motors have the same cabling color.

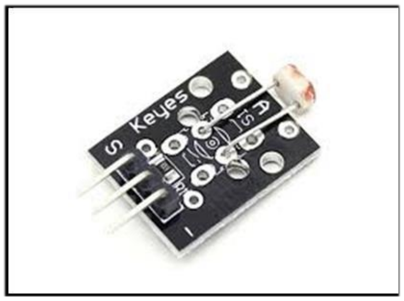
Servo motor

The DHT11 sensor we used is a 3-wire device with its signal output in the middle as shown below, and with an integrated pull-up resistor. Several models are available in the market (including a 4-wire type) and not all have the same pin-out sequence and might require an external 5KΩ to 10KΩ pull-up resistor. So make sure you inspect the one you have before connecting it to Arduino.



DHT11

The photoresistor we used is mounted on a small pcb with three pins as shown below. As seen, the signal output ‘S’ is to the left, the ground to the right, and +5V is in the middle. Its output is an analogue voltage wired to A0 of Arduino. The value of this voltage is inversely proportional to the incoming light i.e it decreases with more light and increases with less. If you can’t procure it as a pcb, you can reproduce it with an LDR (light dependent resistor) in a voltage divider configuration with a 10 KΩ resistor as seen in Fig 9. The middle node is connected to Arduino as the analogue signal.



Photoresistor

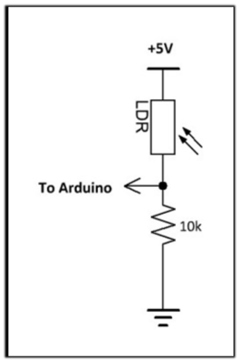


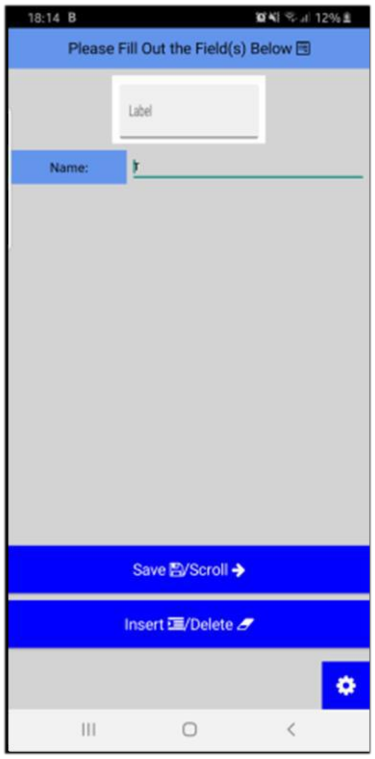
Fig 9.

Press on this link to Copy/Paste the below code to the Arduino IDE and save it by any name, this link for the DHT11 library and save it as dht.h in the same directory, and this link also for the DHT11 library and save it as dht.cpp in the same directory.

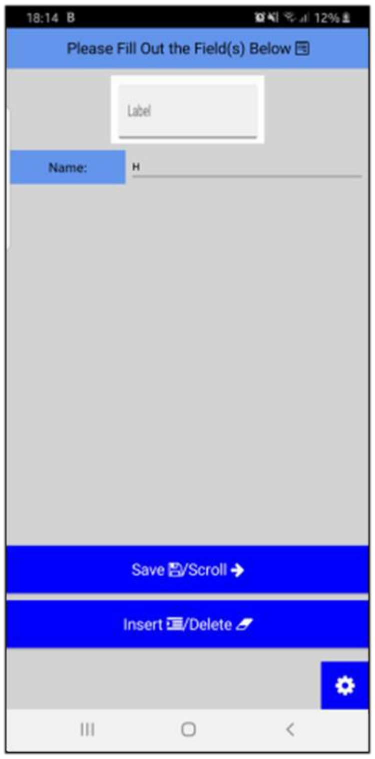
```
#include <Servo.h>  //needed library for servo control
#include "dht.h"  //needed library for DHT11 sensor. Should be downloaded online and placed in
the same directory as this Arduino code
#define DHT11_PIN 8 //Connect DHT11 data line to Arduino pin 8

String AP = "D_LINK1234";       // SSID name, enter your own  network name
String PASS = "Abracadabra@2"; // and enter your network Password
String REG_KEY = "witjmvmox0x9";   // hardware registration key on takano (do not confuse with
user registration key!)
String HOST = "iotsprint.com"; //website Arduino should send http request to
String PORT = "80"; //default port

String string_number_of_characters_to_send;
String command_to_send;
int number_of_characters_to_send;
int countTrueCommand;
int countTimeCommand;
int steps;
boolean found = false;
int tempPin = 1;
int timeout = 0;
int first, last, first1, last1;
String stringT2, stringT3, stringH2, stringH3, stringL2, stringL3, data, content;

Servo myservo;  // create servo object to control a servo
dht DHT;
int photoresistor = A0; //Photoresistor connected to A0 Arduino analogue pin
int light = 0;  // variable to store the value read on A0
int pos;

void negotiate_modem() //sequence of AT commands from Arduino to ESP8266
{  //this function allows the Arduino to connect to the router
  sendCommand("AT+CWMODE=1", 50, "OK");
  sendCommand("AT+CIPMUX=1", 50, "OK");
  sendCommand("AT+CWJAP=\"" + AP + "\",\"" + PASS + "\"", 150, "OK");
}

boolean isNumeric(String str) { //function used to test if a string can be converted to a number
    unsigned int stringLength = str.length();

    if (stringLength == 0) {
        return false;
    }

    boolean seenDecimal = false;

    for(unsigned int i = 0; i < stringLength; ++i) {
        if (isDigit(str.charAt(i))) {
            continue;
        }

        if (str.charAt(i) == '.') {
            if (seenDecimal) {
                return false;
            }
            seenDecimal = true;
            continue;
        }
        return false;
    }
    return true;
}

void setup(){
  // open the serial port at 9600 bps: MAKE SURE ESP8266 HAS SAME BAUD RATE!
  Serial.begin(9600);
}

void loop()
{
  data = "";
  content = "";
  int chk = DHT.read11(DHT11_PIN); //read DHT11 data
  light = analogRead(photoresistor);  //read the photoresistor

  if (found == false) negotiate_modem(); //if Arduino did not get a proper string back from cloud,
then try to connect to modem again.

  float float_temp = DHT.temperature;
  stringT2 = String(float_temp);
  stringT3= String(stringT2);
  if (stringT2.length() == 5) stringT3 = stringT2.substring(0, stringT2.length() - 1); //trim it to 4
characters
  if (stringT3.length() < 2) stringT3 = "000" + stringT3; //to make sure it has 4 characters length
  else if (stringT3.length() < 3) stringT3 = "00" + stringT3; //to make sure it has 4 characters length
  else if (stringT3.length() < 4) stringT3 = "0" + stringT3; //to make sure it has 4 characters length
  stringT3 = "T=" + stringT3; // stringT3 = “T=23.4” for example

  float float_humi = DHT.humidity;
  stringH2 = String(float_humi);
  stringH3= String(stringH2);
  if (stringH2.length() == 5)  stringH3 = stringH2.substring(0, stringH2.length()  - 1); //trim it to 4
characters
  if (stringH3.length() < 2) stringH3 = "000" + stringH3; //to make sure it has 4 characters length
else if (stringH3.length() < 3) stringH3 = "00" + stringH3; //to make sure it has 4 characters
length
  else if (stringH3.length() < 4) stringH3 = "0" + stringH3; //to make sure it has 4 characters length
  stringH3 = "H=" + stringH3; // stringH3 = “H=64.5” for example

  stringL2 = String(light);
  stringL3= String(stringL2);
  if (stringL2.length() == 5) stringL3 = stringL2.substring(0, stringL2.length() - 1); //trim it to 4
characters
  if (stringL3.length() < 2) stringL3 = "000" + stringL3; //to make sure it has 4 characters length
  else if (stringL3.length() < 3) stringL3 = "00" + stringL3; //to make sure it has 4 characters length
  else if (stringL3.length() < 4) stringL3 = "0" + stringL3; //to make sure it has 4 characters length
  stringL3 = "L=" + stringL3; // stringH3 = “L=0456” for example

  String getData = "GET /tk.php?r=" + REG_KEY + "&m=" + stringT3 + "," + stringH3 + "," + stringL3
+ " HTTP/1.0"; //String getData = "GET /tk.php?r=witjmvmox0x9&m=T=23.4,H=64.5,L=0456
HTTP/1.0"; //58 +2
  sendCommand("AT+CIPSTART=4,\"TCP\",\"iotsprint.com\",80", 10, "OK");

  number_of_characters_to_send= getData.length()+27;
//unlike previous Arduino examples, here we automatically
//calculate the number of characters that ESP8266 is to send to the cloud
  string_number_of_characters_to_send= "";
  string_number_of_characters_to_send = String(number_of_characters_to_send);
  command_to_send= "AT+CIPSEND=4," + string_number_of_characters_to_send;
  sendCommand(command_to_send, 10, ">");
  Serial.println(getData); delay(2000); countTrueCommand++;
  read_cloud("Host:iotsprint.com:80");
  delay(3000);  //wait 3seconds before rehearsing
}

void read_cloud(String command)
{
  data = "";
  Serial.println(command);
  Serial.println();
  timeout = 0;

  while (Serial.available() == 0) {
    if ( ++timeout > 10000) { // set this to your timeout value in milliseconds
      break;
    }
  }
  timeout = 0; // got a char so reset timeout
  content = Serial.readString(); // read the incoming data as string

  if ((content.indexOf("M=") >= 0) && (content.indexOf("M=") <= 180)) //servo angle value is
between 0 and 180 degrees
  {
    first = content.indexOf("M=") + 2;
    last = content.lastIndexOf("4,CLOSE");
    data = content.substring(first, last);
    if (isNumeric(data))
     {
      myservo.attach(2); //servo is connected to pin 2
      steps= data.toInt();
      myservo.write(steps);              // tell servo to go to position in variable 'steps'
      delay(1000);                       // waits 1000ms for the servo to reach the position
      myservo.detach();   //unconnect servo to stop it. Please read about servo motors online to
know more about their behavior..
     }
  }
}

void sendCommand(String command, int maxTime, char readReplay[])
{
  found = false;
  while (countTimeCommand < (maxTime * 1))
  {
    Serial.println(command);
    if (Serial.find(readReplay)) //ok
    {
      found = true;
      break;
    }
    countTimeCommand++;
  }

  if (found == true)
```

```
      {
        countTrueCommand++;
        countTimeCommand = 0;
      }

      if (found == false)
      {
        countTrueCommand = 0;
        countTimeCommand = 0;
      }
    }
```

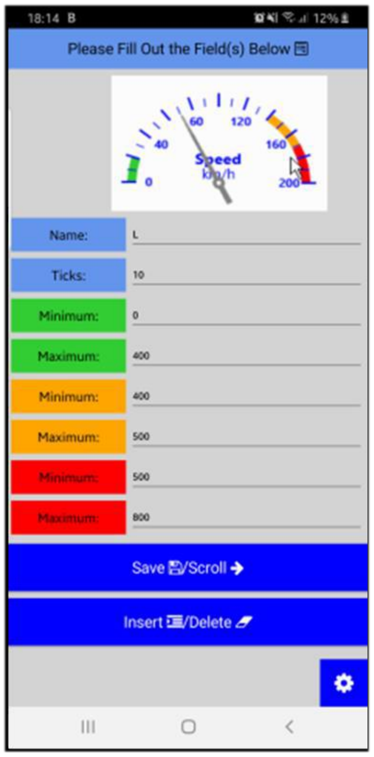
Now we have to setup the mobile app. Start by setting up 2 'Text/Numeric Label' in app, 1 'Gauge 180°' and 1 'Linear Slider' control in the same order as the pics below. The run mode should look like Fig 10.



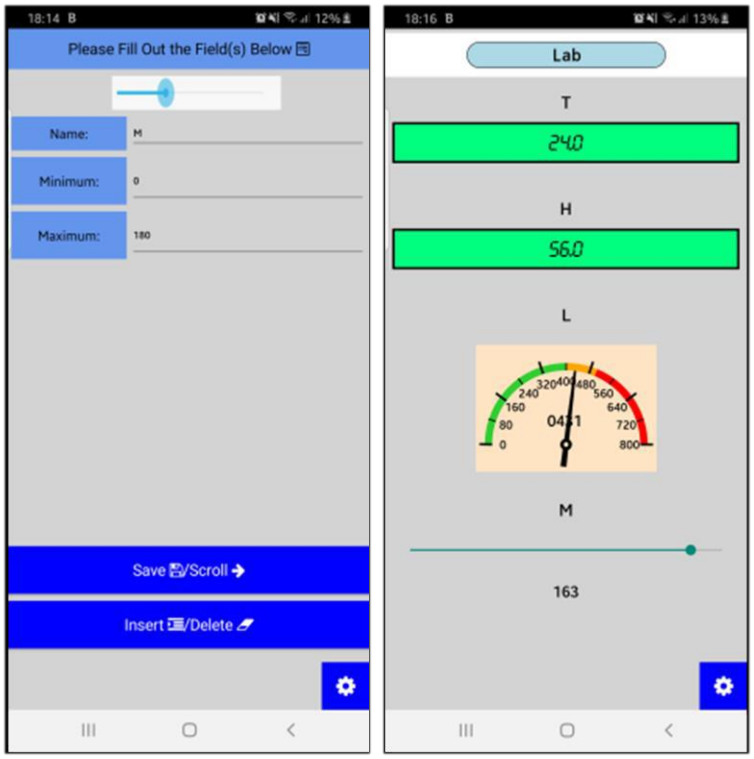
Name:	I



Name:	H



Name:	L
Ticks:	10
Minimum:	0
Maximum:	400
Minimum:	400
Maximum:	500
Minimum:	500
Maximum:	800



Name: M
Minimum: 0
Maximum: 180

T
24.0
H
56.0
L
0431
M
163

Fig 10.

You should be able to read the temperature 'T', the humidity 'H' and the light as a linear analogue value. The light value increases as its surrounding gets darker and diminishes as it gets brighter. A notification can be set in the Takano platform to send an email or a push notification if the light level gets beyond or below a certain value. The servo motor angle can be controlled directly by changing the slider value, which has a minimum of 0 and a maximum of 180.

5. Raspberry Pi Example

The Raspberry Pi is a powerful single-board computer that was chosen due to its huge popularity among students and professionals alike. The example presented below is written in Python in the Thonny IDE that is built-in the Raspberry Pi running the Raspbian operating system. It is a simple example that intends to show the ease of which a Raspberry Pi can take advantage of the Takano platform connectivity. It involves a led that is turned on remotely via the Takano app and a Tilt switch, i.e. a contact that is activated if it is tilted by a certain angle that in turn will send an email to the Takano registered user. We will use https requests since the Raspberry Pi can handle these much better than the ESP8266 used in the Arduino examples. The model we used in this example is a Raspberry Pi 3 Model B+. Please connect it as in Fig 11.

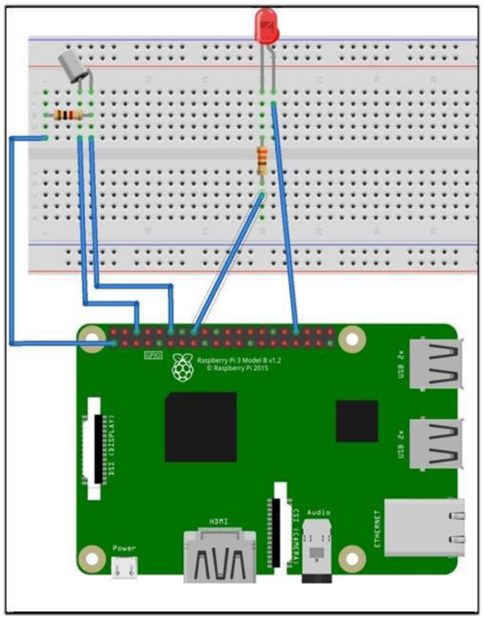


Fig 11.

The Raspberry Pi has of course to be connected to the internet and the Takano app properly setup with the right credentials, i.e. email and user registration key that were used and generated respectively from the Takano platform. The user has to create a two graphic controls, a Check Box for the led and an on/off lamp for the tilt switch. The run mode should look like Fig 12.

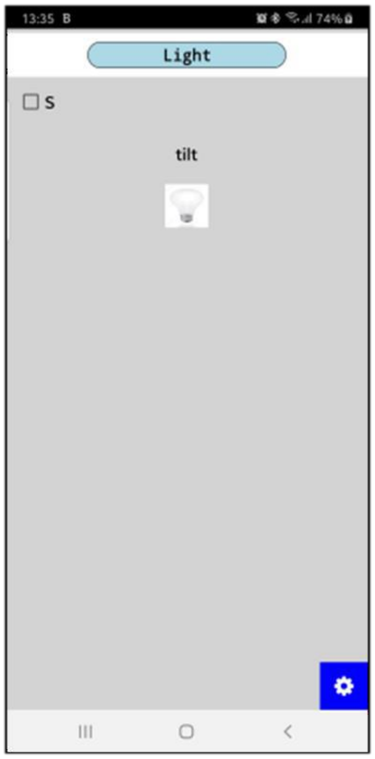


Fig 12.

Press on this link to Copy/Paste the below code to the Python IDE and save it by any name:

```
import requests
import time
from gpiozero import LED, Button
led= LED(23)
tilt= Button(18)
email_sent=0
time.sleep(1)

while True:
    if ((tilt.is_pressed) and (email_sent==0)):
        email_sent=1
    elif not(tilt.is_pressed):
        email_sent=0

    if email_sent==1:
        email_sent=2
        #send email to user
        print("sending email")
        r = requests.get("https://iotsprint.com/tk.php?r=witjmvmox0x9&m=ALARM_TRIPPED|e")
        print(r.text)
    else:
        if (tilt.is_pressed):
            r = requests.get("https://iotsprint.com/tk.php?r=witjmvmox0x9&m=tilt=1")
            print(r.text)
        else:
            r = requests.get("https://iotsprint.com/tk.php?r=witjmvmox0x9&m=tilt=0")
            print(r.text)

        if (r.text == "L=1"):
            led.on()
            print("led is on")
        elif (r.text == "L=0"):
            led.off()
            print("led is off")
        else:
            led.off()
            print("led is undetermined")

        if (tilt.is_pressed):
            print("tilt is on")
        else:
            print("tilt is off")

    print("repeating")
    time.sleep(5)
```

6. NodeMcu Example

The ‘Wemos Lolin NodeMcu v3’ (Fig 13) is another small, cheap and popular open source single board microcontroller that can be programmed with the Arduino IDE. Its main feature is the embedded ESP8266 Wi-Fi and has a micro USB connector that can be used to both power and program it from the PC. The first step is to include the NodeMcu to the Arduino IDE. First open the Arduino IDE and go to “Files” and click on “Preferences”. In the opened window, enter http://arduino.esp8266.com/stable/package_esp8266com_index.json next to “Additional Boards Manager URLs:” then click “OK”. Back to the Arduino main menu, go to “Tools” -> “Board: …” and select “Boards Manager”. In The “Boards Manager” window, navigate to esp8266 by esp8266 community and install the software for Arduino. Once done, we are ready to program the NodeMcu with Arduino IDE. From main menu, go to “Tools” -> “Board: …” and make sure “NodeMcu 1.0” is selected. In this example we will use a tri-color (reg, green blue) led, connected to the NodeMcu together with a 4.7KΩ potentiometer. The led has three anodes: one for red, one for green and one for blue. Each anode has a 220Ω series resistance to limit its current. The fourth led connector is the common cathode for all the led colors, and should be connected to ground. Connect them exactly as in Fig 14 below. We will control the led color and read the potentiometer analog value (anywhere between 0 and 1024) via the internet and Takano platform using the Takano app.

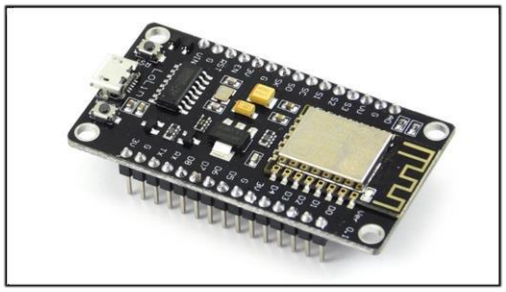


Fig 13.

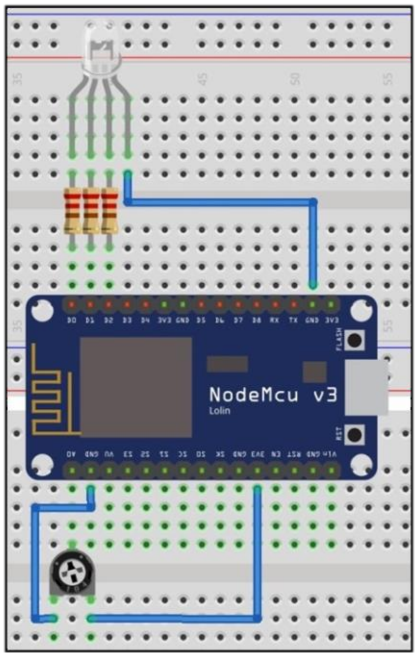


Fig 14.

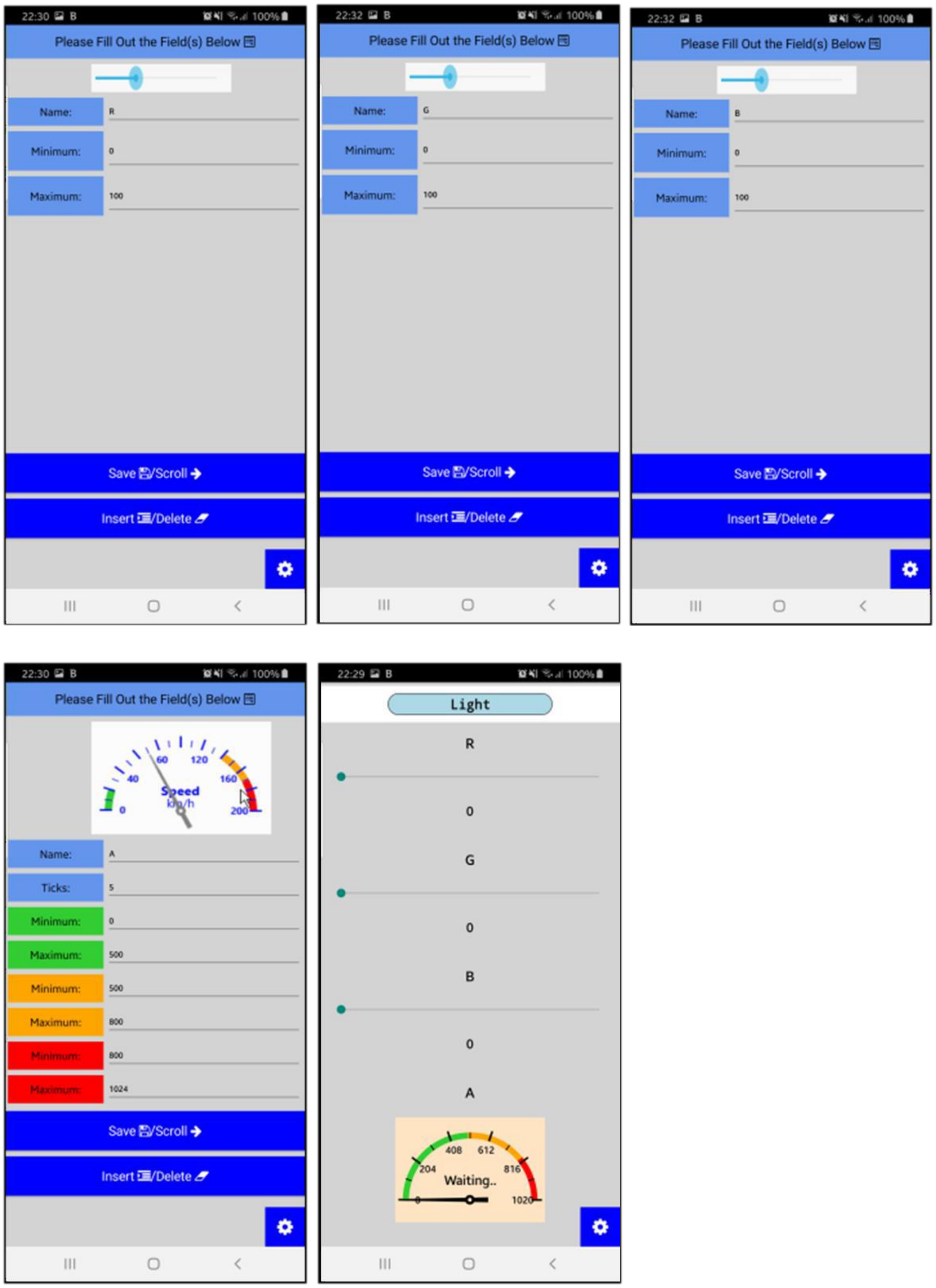
Press on this link to Copy/Paste the below code to the Arduino IDE and save it by any name:

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266HTTPClient.h>

/* Set these to your desired credentials. */
const char *ssid = "Link_4763F";  //enter your Wi-Fi SSID
const char *password = "D4AS7B1AB4"; //enter your Wi-Fi PASSWORD

//Web/Server address to read/write from
const char *host = "iotsprint.com";

String REG_KEY = "witjmvmox0x9";   // hardware registration key on takano (do not confuse with
user registration key!)
```

```
int ledB = D0; //blue led pin
int ledG = D1; //green led pin
int ledR = D2; //red led pin
int R;
int G;
int B;

void  setup() {
  delay(1000);
  R=0;
  G=0;
  B=0;
  pinMode(D0, OUTPUT); //configure D0 pin as output
  pinMode(D1, OUTPUT); //configure D1 pin as output
  pinMode(D2, OUTPUT); //configure D2 pin as output
  //Set PWM frequency 500, default is 1000
  //Set range 0~100, default is 0~1023
  analogWriteFreq(500);
  analogWriteRange(100);

  Serial.begin(115200); //setg serial port baud rate at 115200
  WiFi.mode(WIFI_OFF);  //Prevents reconnection issue (taking too long to connect)
  delay(1000);
  WiFi.mode(WIFI_STA);  //This line hides the viewing of ESP as wifi hotspot

  WiFi.begin(ssid, password);  //Connect to your WiFi router
  Serial.println("");

  Serial.print("Connecting");
  // Wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  //If connection successful show IP address in serial monitor
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());  //IP address assigned to your ESP
}

// Main Program Loop
void loop() {
  HTTPClient http;     //Declare object of class HTTPClient

  String getData, Link;
  int adcvalue= analogRead(A0);  //Read Analog value of 4K7 potentiometer
  Serial.println("Analog: " +String(adcvalue));     //Print request response payload
  //GET Data
  getData = "r=" + REG_KEY + "&m=A=" + String(adcvalue); //prepare message to send to
Takano cloud
```

```
  Link = "http://iotsprint.com/tk.php?" + getData;

  http.begin(Link);  //Specify request destination

  int httpCode = http.GET();             //Send the GET request
  String payload = http.getString();    //Get the response payload

  Serial.println(httpCode);   //Print HTTP return code
  Serial.println(payload);     //Print request response payload

  //look for red, green and blue values in server response
   R= payload.indexOf("R=");
   if (R != -1) { R= payload.substring(R+2).toInt();   analogWrite(ledR, R); }

  G= payload.indexOf("G=");
  if (G != -1) { G= payload.substring(G+2).toInt(); analogWrite(ledG, G); }

  B= payload.indexOf("B=");
  if (B != -1) { B= payload.substring(B+2).toInt(); analogWrite(ledB, B); }

  http.end();  //Close connection
  delay(5000);  //Repeat every 5 seconds
}
```

Now we need to setup the Takano app:
Launch the app from your smartphone, and after entering your Takano credentials (email, user reg.key and hardware name in the setup form), press on "Choose Parameter Display" and enter the three graphic items in the same order as presented in the below pics. Here we chose three sliders, one to named "R" to control the red colour, one name "G" to control the green colour and one named "B" to control the blue colour. And finally an analog 180 degrees needle gauge to read the analog potentiometer value. Then going back to the run mode we should have a form like that in Fig 14.



22:30 B 100%

Please Fill Out the Field(s) Below

Name:	R
Minimum:	0
Maximum:	100

Save /Scroll

Insert /Delete

22:32 B 100%

Please Fill Out the Field(s) Below

Name:	G
Minimum:	0
Maximum:	100

Save /Scroll

Insert /Delete

22:32 B 100%

Please Fill Out the Field(s) Below

Name:	B
Minimum:	0
Maximum:	100

Save /Scroll

Insert /Delete

22:30 B 100%

Please Fill Out the Field(s) Below

Name:	A
Ticks:	5
Minimum:	0
Maximum:	500
Minimum:	500
Maximum:	800
Minimum:	800
Maximum:	1024

Save /Scroll

Insert /Delete

22:29 B 100%

Light

R

0

G

0

B

0

A

Fig 14.

If the NodeMcu is properly connected to the internet, we should read the analog potentiometer value directly on the app, and changing the sliders should be reflected back on the tri-color led, with the combination of red, green and blue colors blending to give a unique color. Just remember that the NodeMcu refreshes every 5 seconds.

7. Wemos D1 R2 Example

The WeMos D1 R2 is another cheap hardware based on the ESP8266 Wi-Fi microcontroller development board, and is compatible with the Arduino IDE. Its dimensions are identical to the Arduino Uno, with 2 major differences: it runs on 3.3V instead of 5V, so not all shields and accessories will directly work for both, and its pin-outs arrangement are a bit dissimilar, so care should be taken when porting an Arduino Uno design to the Wemos D1 R2 board. It is powered by a micro USB connector.
The purpose of this example is to show how to use SSL encryption on the low-cost ESP8266 to connect to the Takano platform, by connecting a push button and a led to the Wemos. Just a quick word about SSL: Secure Sockets Layer (SSL) is a standard security protocol for establishing an encrypted link between a server (Takano platform in our case) and a client (Wemos D1 R2). When exchanging sensitive data with a server, like credit card number, login credentials such as passwords, or a patient medical record for example, you wouldn't want an 'attacker' listening to you connection to intercept those information. In other words, the data sent from your browser to the server and vice versa should be coded in a way that would make it 'unreadable' by anyone else except your browser and server only. SSL does exactly that by involving a complex handshaking mechanism between server and client. It should be noted that SSL comes at a cost: the encryption protocol incurs additional network bandwidth and CPU resources that is palpable on a small device like the ESP8266. The time taken to connect and start sending and receiving data from the server could translate into several seconds of latency.
The first step is to include the ESP8266 to the Arduino IDE if it hasn't been done before. First open the Arduino IDE and go to "Files" and click on "Preferences". In the opened window, enter http://arduino.esp8266.com/stable/package_esp8266com_index.json next to "Additional Boards Manager URLs:" then click "OK". Back to the Arduino main menu, go to "Tools" -> "Board: …" and select "Boards Manager". In The "Boards Manager" window, enter "esp8266" in the input field and select the version 2.5.0 or more if available, even if it is still a beta version. Then press on "Install" button. This version is required for SSL encryption to properly work on the ESP8266. Again go to "Tools"-> "Board: …" and select "LOLIN(WEMOS)D1 R2 & mini". The wiring is shown in Fig 15. Make sure you have an internet connection before uploading the sketch to Wemos.

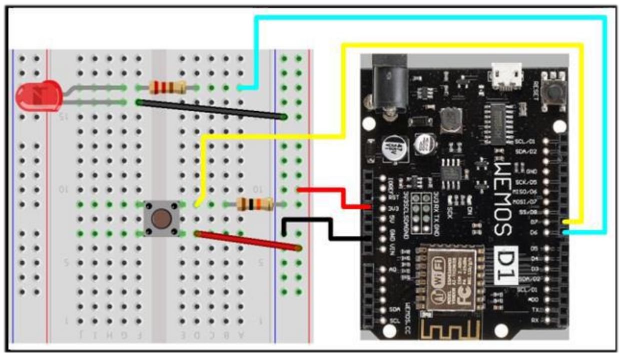


Fig 15.

Press on this link to Copy/Paste the below code to the Arduino IDE and save it by any name:

```
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
#include <ESP8266WebServer.h>
#include <ESP8266HTTPClient.h>

/* Set these to your desired credentials. */
const char *ssid = "DLink007";  /* ENTER YOUR WIFI SSID */
const char *password = "Abracadabra$02"; /* ENTER YOUR WIFI PASSWORD */

const char *host = "iotsprint.com";
const int httpsPort = 443;  /* HTTPS= 443 */

/* SHA1 finger print of certificate of iotsprint.com */
const char fingerprint[ ] PROGMEM = "F3 CE FA 78 EC 4A E2 F2 BE 50 69 07 04 0B EA 74 B8
48 1C 62";

String REG_KEY = "witjmvmox0x9";   // hardware registration key on takano (do not confuse with
user registration key!)

int Led = D6;  // LED connected to digital pin D6
int Button = D7;  // pushbutton connected to digital pin D7
int val = 0;  // variable to store the read value
int L;

void setup() {
pinMode(Led, OUTPUT);  // sets the digital pin 6 as output
pinMode(Button, INPUT);     // sets the digital pin 7 as input
  delay(1000);
  Serial.begin(115200);
  WiFi.mode(WIFI_OFF);        //Prevents reconnection issue (taking too long to connect)
  delay(1000);
  WiFi.mode(WIFI_STA);        //Only Station No AP, This line hides the viewing of ESP as wifi
hotspot

  WiFi.begin(ssid, password);     //Connect to your WiFi router
  Serial.println("");
  Serial.print("Connecting");
  // Wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  //If connection successful show IP address in serial monitor
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());  //IP address assigned to your ESP
}

void loop() {
  WiFiClientSecure httpsClient;    //Declare object of class WiFiClient

  Serial.println(host);

  Serial.printf("Using fingerprint '%s'\n", fingerprint);
  httpsClient.setFingerprint(fingerprint);
  httpsClient.setTimeout(15000); // 15 Seconds
  delay(1000);

  Serial.print("HTTPS Connecting");
  int r=0; //retry counter
  while((!httpsClient.connect(host, httpsPort)) && (r < 30)){
      delay(100);
      Serial.print(".");
      r++;
  }
  if(r==30) {
    Serial.println("Connection  failed");
  }
  else {
    Serial.println("Connected to web");
  }

String string_val, getData, Link;
string_val= "";
val = digitalRead(Button);   // read the push button input pin
string_val = String(val);   //String to integer conversion

getData = "r=" + REG_KEY + "&m=tilt=" + string_val; //prepare message to send to Takano cloud
Link = "/tk.php?" + getData;

  Serial.print("requesting URL: ");
  Serial.println(host+Link);

  httpsClient.print(String("GET ") + Link + " HTTP/1.1\r\n" +
               "Host: " + host + "\r\n" +
               "Connection: close\r\n\r\n");
  Serial.println("request sent");

  while (httpsClient.connected()) {
    String line = httpsClient.readStringUntil('\n');
    if (line == "\r") {
      Serial.println("headers received");
      break;
    }
  }

  Serial.println("reply was:");
  Serial.println("==========");
  String line;
  while(httpsClient.available()){
    line = httpsClient.readStringUntil('\n');  //Read Line by Line

     //look for "S=" in server response
     L= line.indexOf("S=");
     if (L != -1)
      {
       L= line.substring(L+2).toInt();
       if (L==1) digitalWrite(Led, HIGH);
       else if (L==0) digitalWrite(Led, LOW);
      }

    Serial.println(line); //Print response
  }
  Serial.println("==========");
  Serial.println("closing connection");

  delay(5000);  //Repeat every 5 seconds
}
//=======================================================================
```

After the upload is successfully over, go to “Tools”->”Serial Monitor” in Arduino IDE, and watch the exchanged packets between the Wemos and the Takano server (making sure that the Serial Monitor baud rate is set to 115200).
The same mobile app setup for the Raspberry pi example can be used here without any modification.

8. Contact information

For more information or comments, please do not hesitate to contact the Takano technical team by email at: info@iotsprint.com
We are available from 8:00AM (GMT+3) to 6:00PM (GMT+3).