Environmental Wireless Sensor Node with ESP8266, DHT and LDR for Humidity, Temperature and Light Intensity Monitoring.

Table of Contents

The following is a brief description of the setup and functionality of the sensor node responsible for collecting and sending temperature, light intensity and humidity values to the cloud platform.

Objective

The main purpose of the designed system is to collect temperature, humidity and light intensity readings from the environment and upload this data to a cloud platform database. The system is designed to function wirelessly with minimal overhead management from a user.

Components

The system is composed of the following major modules: the power supply, the sensors and the microprocessor/communication module.

Sensors

This consist of the actual physical devices that interface with the physical world. Three measurements were of interest in the design: temperature, humidity and light intensity. The following two components were used to determine the readings

DHT Sensor

The DHT11 is a basic, ultra-low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analogue input pins needed).

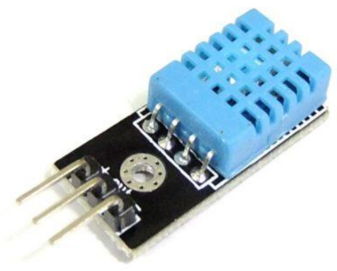

Figure 1:DHT Sensor

Light Dependent Resistor (LDR)

An LDR or photo-resistor is basically a variable resistor whose resistance depends on the amount of light incident on its surface; the higher the incident light intensity, the lower the resistance. This relationship was used in the design of the light sensor used in the system.

Figure 2:Light Dependent Resistor

Microprocessor/Communication Module

This is the most important unit of the node. It provides both the computing capabilities of the system and the ability to connect to the network and send the required information. The ESP8266 MOD was chosen as the preferred hardware due to its unique capability of being both a microprocessor and a communication module.

The ESP8266 Wi-Fi Module is a self-contained SOC with integrated TCP/IP protocol stack that can give any microcontroller access to a Wi-Fi network. The ESP8266 is capable of either hosting an application or offloading all Wi-Fi networking functions from another application processor.

This module has powerful enough on-board processing and storage capability that allows it to be integrated with the sensors and other application specific devices through its GPIOs with minimal development up-front and minimal loading during runtime. Its high degree of on-chip integration allows for minimal external circuitry, including the front-end module, is designed to occupy minimal PCB area.

The following is an image of ESP8266 MOD12

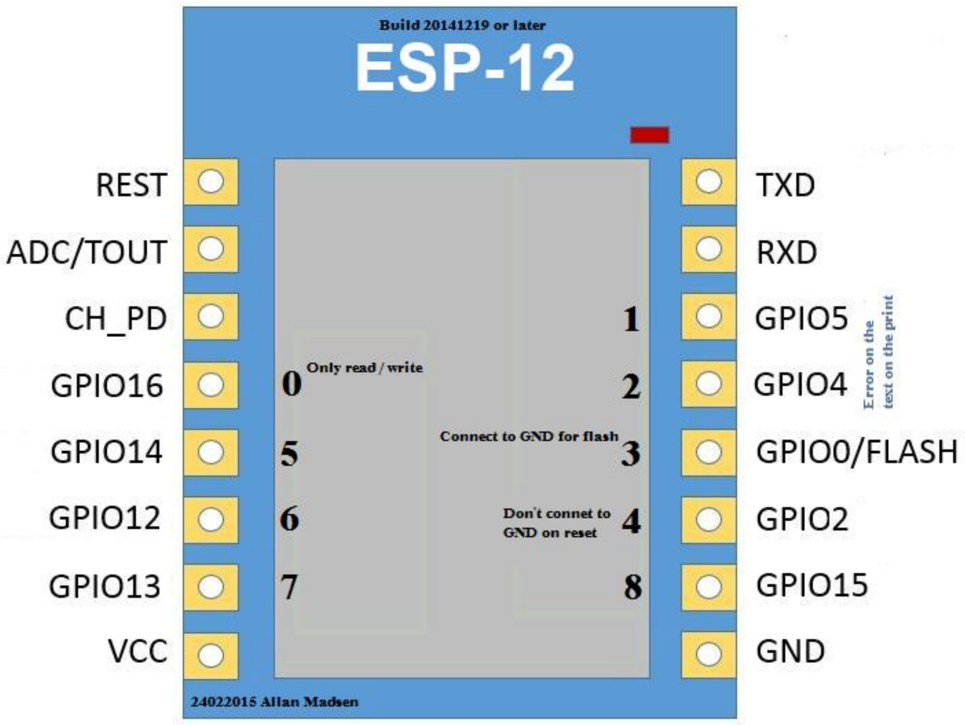

Figure 3:ESP8266 MOD

The device has several GPIO pins: GPIO5, GPIO4, GPIO12, GPIO13, GPIO14 can all be used to connect digital sensors, LEDs or actuators. The module has one analogue input (ADC) with 10bit resolution thus only one analogue sensor can be attached per module.

Power Supply

This is the unit that supplies power to the whole circuitry. A 1000mAH 3.7V LiPo battery was employed as the power supply. The unit could thus be portable to anywhere with Wi-Fi connection without the need of wires to supply the power.

Figure 4:LiPo Battery

Circuit Design

Circuit Diagram

The following image describes the circuit used to make the system

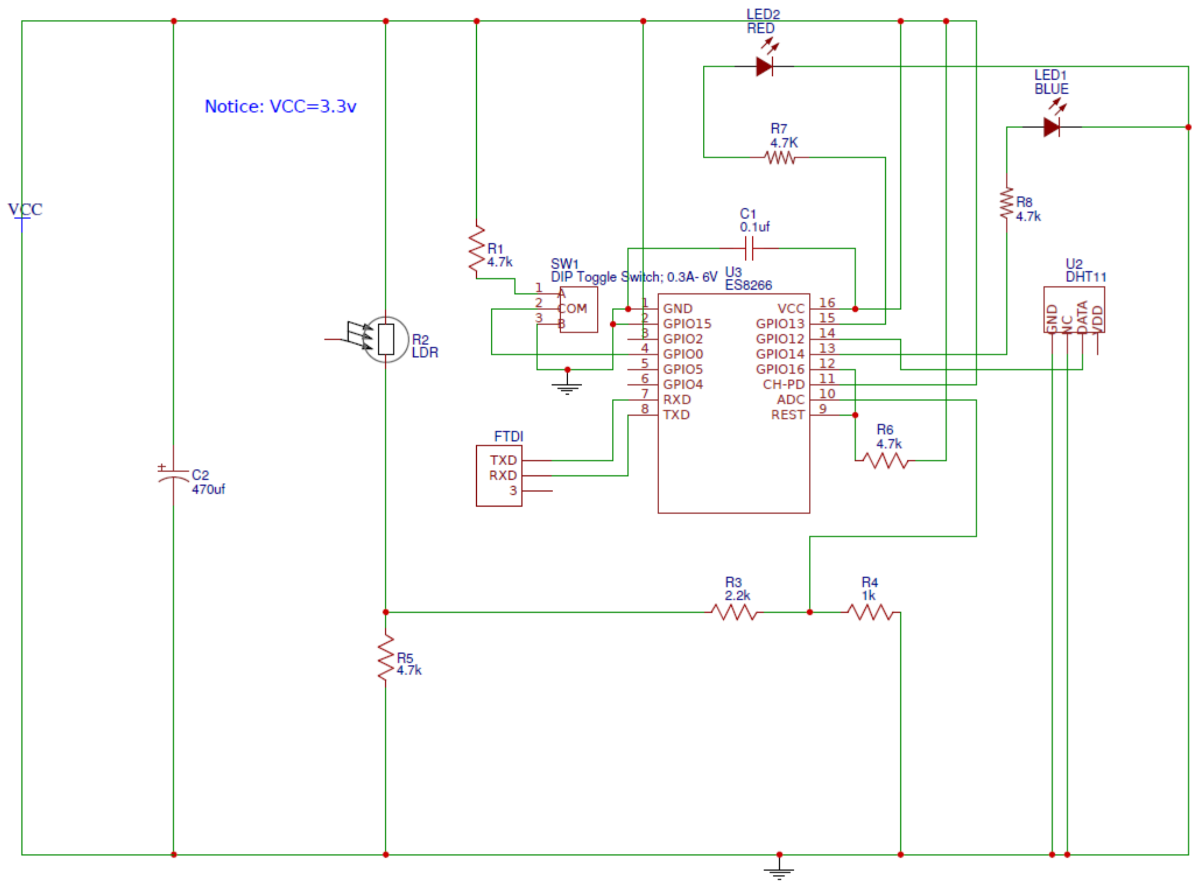

Figure 5:Circuit Diagram of the System

Circuit Description

A brief explanation of the major components of the circuit is attempted below. Obvious items such as how the battery has been connected have been left out.

1. In order for the ESP8266 to work as desired the following vital connection must be made
   - GPIO15 must be connected to the ground
   - GPIO2 must be pulled high
   - GPIO 16 and RESET pin must be tied together and then pulled high through 4.7K resistor. The system will still work without this connection but if we want

the module to periodically go to sleep mode, this connection must be made otherwise the system will not get out of sleep mode.

- GPIO 0 MUST be connected to ground when uploading the program. After uploading the code, GPIO 0 must be left hanging or pulled high through a 4.7K resistor (either way apparently works) If GPIO 0 pin is left connected to the ground after the code is uploaded, the ESP8266 module will function as long as power is still supplied to it. Once it is switched off and then back on again, it cannot automatically load the program in memory thus making it necessary to make the pin floating or pull it up though the 4.7K resistor. This necessitates the use of the 3-way switch shown. It is advisable not to use this pin to connect any sensors.

2. GPIO 4,5,12,13,14 can be used for sensors, actuators, LEDs etc. Please note that these are digital pins thus only digital sensors can be attached to them.
3. There is only one analogue input for the ESP8266 module which proved quite problematic to use. This pin is basically a 10bit ADC. The tricky bit is that it only quantizes voltages from 0 to 1V thus anything above 1V will produce a level of 1023. To solve this problem, a voltage divider circuit is placed before the pin which scales down the voltage in the ratio

$$\frac{R4}{R4+R5} = \frac{1}{1+2.2} = 0.3125$$

which is not very accurate but it does what we want it to do. Moreover, it is not super important that we accurately measure the light intensity, our point is to determine approximate levels of it.

4. The two LEDs were used to show the status of the system. If the red LED is on, the system is on (out of sleep mode), if the blue LED comes on, the system is connected to a WIFI network. If the blue LED blinks three times consecutively, the module has made connection to the server and posted the sensor readings to the database. If both the LEDs are off, it means the system is off (or in sleep mode). This on/off behaviour of the LED is controlled from the code loaded in the ESP8266.
5. A 470uF capacitor was used across the voltage lines to minimize voltage fluctuations. A 0.1uF decoupling capacitor was placed across the ESP8266 Vcc to Gnd.

The rest of the circuit we believe is self-explanatory. The code will not be shown here but can be requested from @iLabAfrica, IOT Lab.

The picture of the system is shown below, it's not quite so pretty to look at but it does the job:

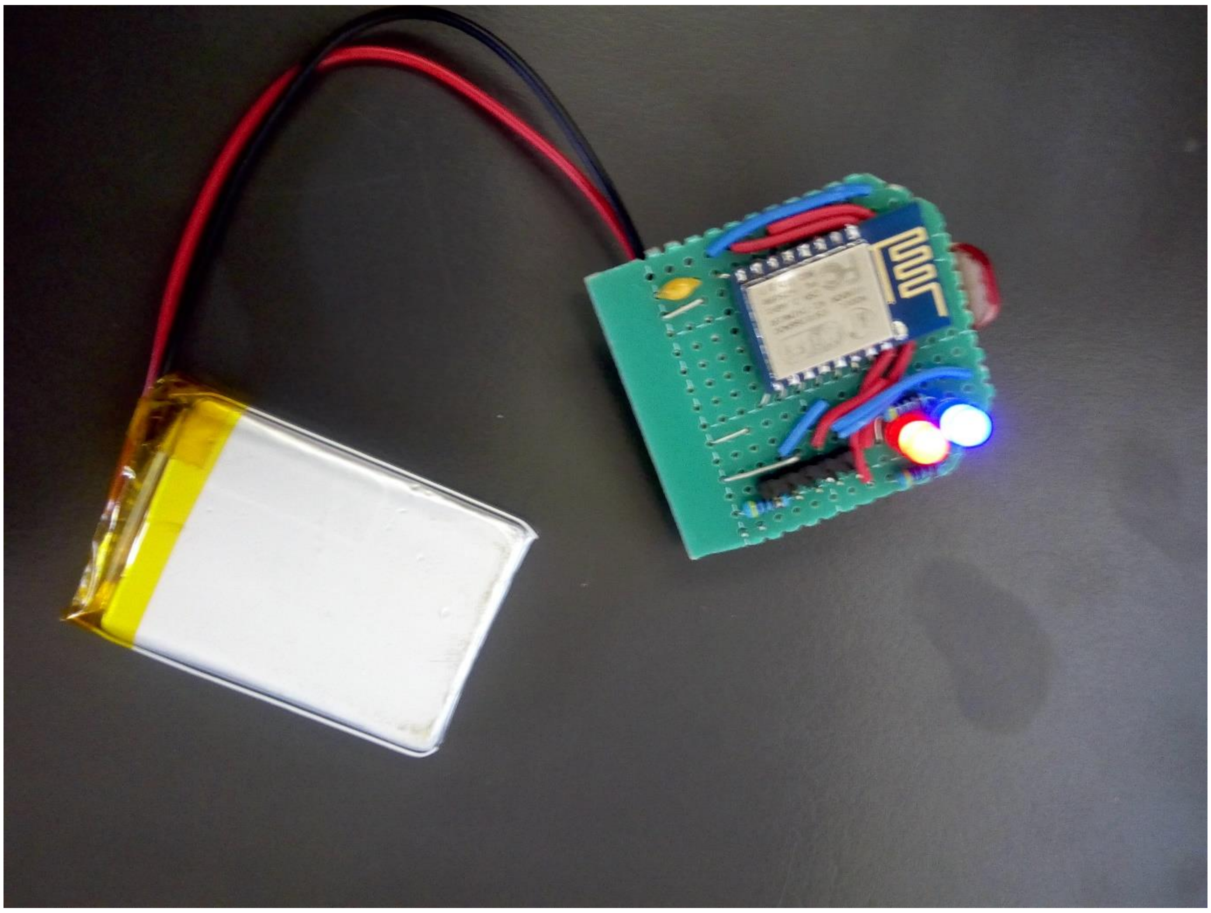

Figure 6:Picture of the System

Power Consumptions and Tests on Functionality

The ESP8266 MOD12 does not seem to have a specific time it takes before connecting to the Wi-Fi. The time ranges from as few as 5 seconds to as long as 2 minutes. This time is also dependent on whether there is a strong Wi-Fi coverage. The stronger the Wi-Fi, the faster the module connects to the Wi-Fi but from tests, there is no one specific time the module will take to connect to the Wi-Fi even when you place the module in the exact same position. This poses a problem, the ESP8266 module will keep trying to make a connection to a Wi-Fi, thereby draining the battery.

-The code was modified such that after every successful upload of the readings from the sensors it goes into deep sleep mode for 8 minutes and when it boots up again, it reconnects to the Wi-Fi afresh.

The module consumes 70mA when it is idle, when the program starts, it consumes about 80mA.

The ESP8266 MOD12 was programmed to go in deep sleep mode for 8 minutes. In deep sleep mode, the current consumption goes below 1mA. The exact current draw when in deep sleep mode could not be accurately measured because the multi-meter in the lab cannot measure current in micro Amperes. There is no significant difference in current draw when the module has connected to Wi-Fi and when it is disconnected to Wi-Fi.

There were noted inconsistencies in the time the module takes to drain the 1000mAH battery. After several tests with the module, it was discovered that the Wi-Fi strength is the biggest limitations in terms of how long the battery lasts. If the Wi-Fi signal is weak and the battery takes too long to connect to the network, it ends up draining more current than it should.

The maximum continuous operation time of the system was recorded as 144 hours (and still counting)-it was made to sleep every 8 minutes. If the module in made to work without going to sleep mode (continuously sending data), the battery is drained in about 12 hours.