Sunfounder Smart Home Kit for Rpi and Arduino

About SunFounder

SunFounder is a technology company focused on Raspberry Pi and Arduino open source community development. Committed to the promotion of open source culture, we strive to bring the fun of electronics making to people all around the world and enable everyone to be a maker. Our products include learning kits, development boards, robots, sensor modules and development tools. In addition to high quality products, SunFounder also offers video tutorials to help your own project. If you have interest in open source or making something cool, welcome to join us! Visit www.sunfounder.com for more!

Kit Overview

With this kit tutorial, you will learn how to DIY a simple Smart Home system. In this Smart Home system, we constantly upload temperature and humidity, light intensity, gas values, gas pressure, elevation, RFID status values, human pyroelectric infrared sensor status values and other data to DeviceBit platform at a certain time interval. With the controller options on the DeviceBit platform, you will be able to control your remote relay in real time.

You can go to our official website www.sunfounder.com to download related code by clicking LEARN -> Get Tutorials and watch related videos by clicking VIDEO.

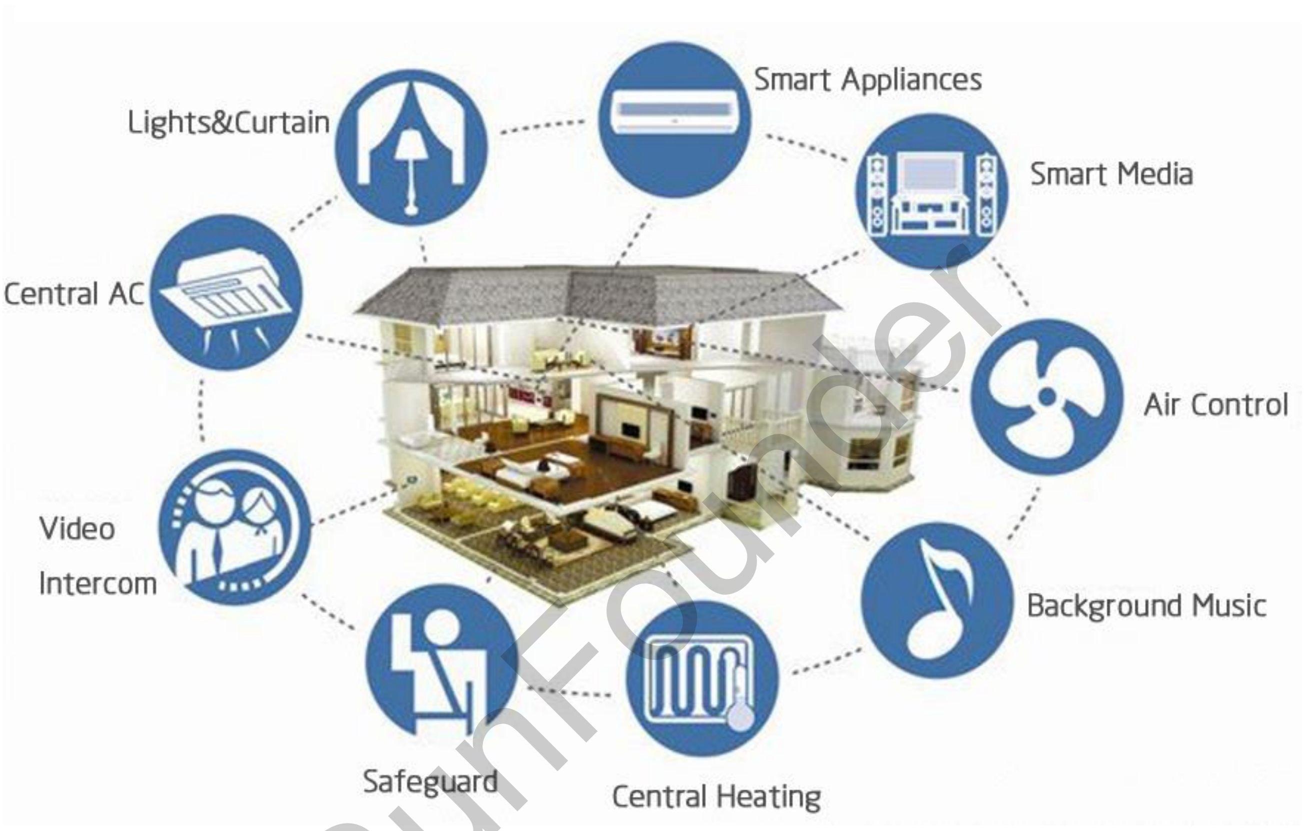
If you have any questions, please send an email to support@sunfounder.com. You can also leave a message and share your projects on our forum.

Contents

What is Smart Home?........2
The Purpose for Developing This Kit........3
Pre-requisites........4
    Some Linux Basic Knowledge........4
    Required Components........4
Operation Principle and System Structure........12
DeviceBit Platform........13
    DeviceBit Platform Introduction........13
    Register Devicebit Account........13
    Quick Start........14
    Add Sensors Required........14
An Introduction to Data Acquisition Node........16
    Circuit Connection........17
    Program and Debug........18
Data Processing and Relay Node........20
    Circuit Connection........21
    Install Raspberry Pi SPI Driver........22
    Program and Debug........23
System Joint Debugging........28
Expansion........29
Summary........29

What is Smart Home?

Smart Home is a management system by using comprehensive wiring technology, network communication technology, smart furnishing system, safeguard system, and audio-video technology to provide home with a safe, convenient, comfortable, eco-friendly and energy-efficient environment.



As the development of Internet and technology, the concept of smart home emerges. It can integrate all your home facilities together, provide various functions like household appliances control, lighting control, curtain control, remote telephone control, indoor and outdoor remote control, burglar alarm, environment monitoring, HVAC control, programmable timing control and so on. Compared with ordinary home, smart home not only has basic living function, but also has a feature of comprehensive information interaction. It will optimize the way people live and help people to arrange time effectively, improve the safety standard, even cutting the energy cost.

The Purpose for Developing This Kit

When it comes to Smart Home, the most critical problem is usually how to really interact with these devices and how to control them conveniently. It seems that all OEMs have their own set of processes. Although all of them are very smart, a whole set of ready-made solution is often not cheap.

With current hot open source hardware field and the rapid development of Internet of Things technology, it is very convenient for us to realize smart home. For those who have a basic knowledge of programming and plenty of time, they can design a set of smart home system with complete functions and low cost by Raspberry Pi and Arduino. At the same time, you can enjoy a lot of fun in the process of DIY. The two open source tools are not only for makers and hardware development hobbyist but also for anyone new to hardware development. They can learn how to build their own project with this open source hardware platform in shortest time.

SunFounder's smart home kit is not a collection of boring electronic components. By learning our tutorial for this kit, you will not only have a deeper understanding about open source hardware and Internet of Things, but also can make some real, useful things.

Pre-requisites

Some Basic Knowledge about Linux

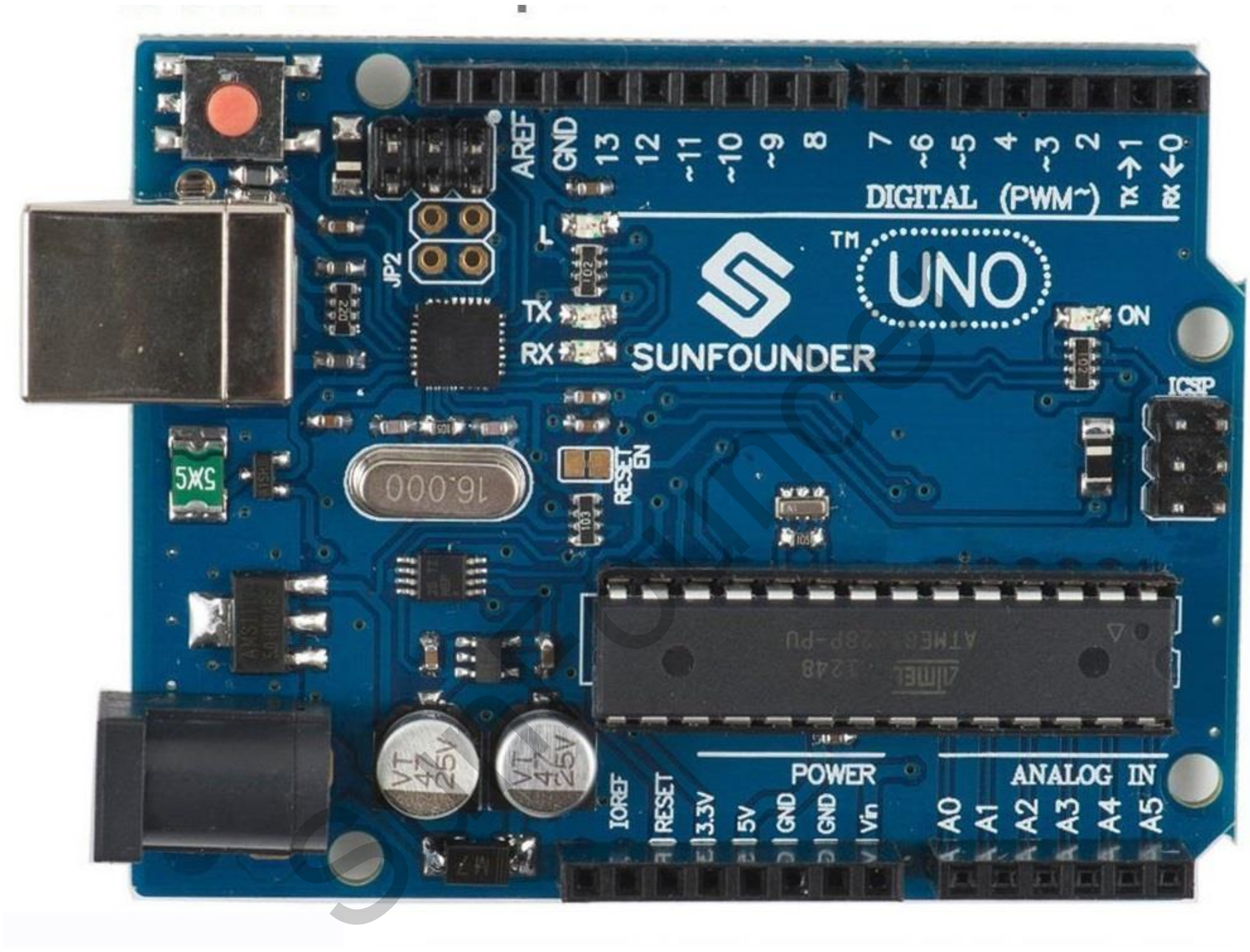
To play with Raspberry Pi, you must have some basic knowledge about Linux. Since the operation system of Raspberry Pi is based on Linux, so if you know about Linux, it will be very convenient for you to realize your ideas.

Required Components

1* Raspberry Pi

The Raspberry Pi is a credit-card sized computer that plugs into your TV and a keyboard. It is a capable little computer which can be used in electronics projects, and for many of the things that your desktop PC does, like spreadsheets, word-processing and games. It also plays high-definition video. We want to see it being used by kids all over the world to learn how computers work, how to manipulate the electronic world around them, and how to program.

1* Arduino Uno



Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It's intended for artists, designers, hobbyists and anyone interested in creating interactive objects or environments.

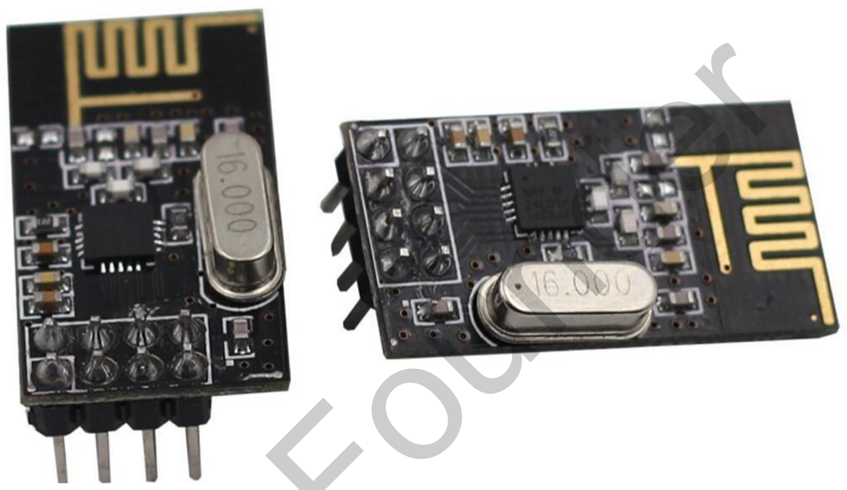
Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing). Arduino projects can be stand-alone or they can communicate with software running on a computer (e.g. Flash, Processing, MaxMSP).

The boards can be built by hand or purchased preassembled; the software can be downloaded for free. The hardware reference designs (CAD files) are available under an open-source license, you are free to adapt them to your needs.

Arduino received an Honorary Mention in the Digital Communities section of the 2006 Ars Electronica Prix.

The Arduino founders are: Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, and David Mellis.

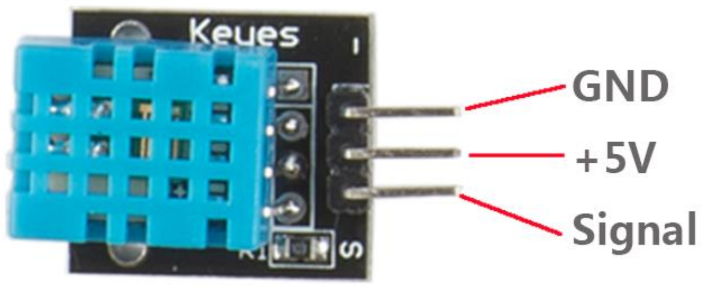
2* nRF24L01+ Module



The Nordic nRF24L01+ is a highly integrated, ultra low power (ULP) 2Mbps RF transceiver IC for the 2.4GHz ISM (Industrial, Scientific and Medical) band. With peak RX/TX currents lower than 14mA, a sub µA power down mode, advanced power management, and a 1.9 to 3.6V supply range, the nRF24L01+ provides a true ULP solution enabling months to years of battery life from coin cell or AA/AAA batteries. The Enhanced ShockBurst™ hardware protocol accelerator offloads time critical protocol functions from the application microcontroller enabling the implementation of advanced and robust wireless connectivity with low cost 3rd-party microcontrollers.

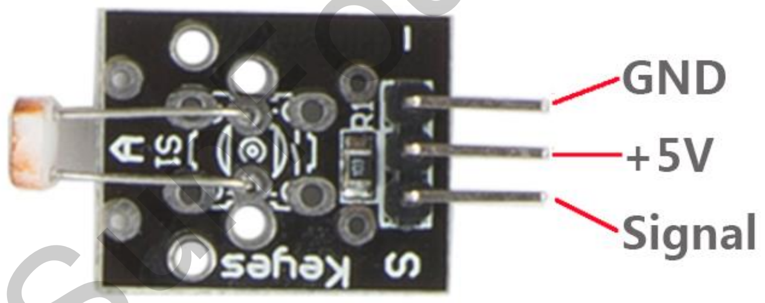
The Nordic nRF24L01+ integrates a complete 2.4GHz RF transceiver, RF synthesizer, and baseband logic including the Enhanced ShockBurst™ hardware protocol accelerator supporting a high-speed SPI interface for the application controller. No external loop filter, resonators, or VCO varactor diodes are required, only a low cost ±60ppm crystal, matching circuitry, and antenna.

1* DHT-11



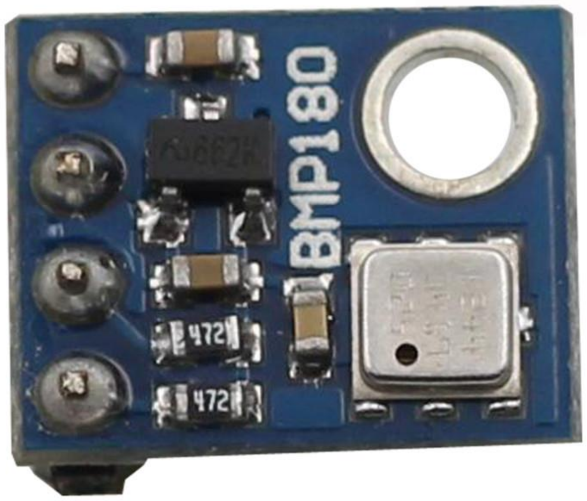
The DHT-11 is a basic, ultra low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins needed). It's fairly simple to use, but requires careful timing to grab data. The only real downside of this sensor is that you can only get new data from it once every 2 seconds.

1* Photoresistor



In this project, a photoresistor is used as detection component, combined with Arduino Uno built-in ADC to realize light intensity detection. By automatically detecting light intensity, systems and users can get to know current light intensity to adjust it reasonably. The basic principle of light intensity detection is that the resistance of photoresistor decreases with the increase of light intensity, thus the voltage at the two ends of the photoresistor decreases. You can conveniently obtain current light intensity by doing AD conversion to the voltages with Arduino Uno built-in ADC and then doing some simple conversion and processing with applications. The light intensity detection circuit is as shown below:

1* BMP180



The BMP180 is the function compatible successor of the BMP085, a new generation of high precision digital pressure sensors for consumer applications. The ultra-low power, low voltage electronics of the BMP180 is optimized for use in mobile phones, PDAs, GPS navigation devices and outdoor equipment. With a low altitude noise of merely 0.25m at fast conversion time, the BMP180 offers superior performance. The I2C interface allows for easy system integration with a microcontroller. The BMP180 is based on piezo-resistive technology for EMC robustness, high accuracy and linearity as well as long term stability. Robert Bosch is the world market leader for pressure sensors in automotive applications. Based on the experience of over 400 million pressure sensors in the field, the BMP180 continues a new generation of micro-machined pressure sensors.

1* MQ-2

Coal gas, natural gas, liquefied petroleum gas and other combustible gas are often used as kitchen fuels in city. It often causes fire, poisoning and explosion accidents due to improper use and management, thus causes great losses of life and property to residents. To monitor combustible gas leakage, you can design a combustible gas detector in automatic fire alarm system and install it in kitchen or room with combustible gas valve. This design uses MQ-2 combustible gas sensor with IR phototransistor to realize coal gas leakage and fire monitoring. MQ-2 features wide detection range, high sensitivity, fast response and recovery, excellent stability, long service life and simple drive circuit. It is widely used in gas leakage monitoring device in homes and factories, and suitable for liquefied petroleum gas, butane, propane, methane, alcohol, hydrogen, and smoke detection.

1* PIR

To prevent strangers from illegally entering, a human body pyroelectric infrared (PIR) sensor module is installed in a proper place of the house. Once illegal person enters, it will alarm and inform the house owners by their alarm controller. This design uses the PIR sensor module to realize infrared security and protection and intrusion detection.

The PIR sensor is used as a detecting component with a size of 2*1mm, which is mainly made of a kind of high thermoelectric coefficient material, such as lead zirconate titanate ceramics, lithium tantalate, triglycine sulfate, etc. The central wavelength of IR rays radiated by human body is 9~10 um, and the wavelength sensitivity of the detecting component is almost stable within the range of 0.2 ~ 20 um. A window with filter is set up on top of the sensor. The filter can pass through light with wavelength in the range of 7 ~ 10 um, which is just suitable for human body infrared radiation detection. For IR rays with other wavelength, it will be absorbed by the filter to block lamplight, sunlight and other IR radiation. Thus it is formed an IR sensor specially used for human body infrared radiation detection.

Mount one or two detecting components in each detector and connect them in series with polarity reversed to suppress interferences generated due to their temperature increase. The detecting components transform IR radiation signals detected and received into weak voltage signals and then amplify them through FET packaged in the probe to output. The main purpose for introducing FET in the module is to realize impedance transformation. In order to improve detection sensitivity of the detector to increase detection distance, we usually install a Fresnel lens which is made of transparent plastic in the front of the detector. Divide the upper and lower parts of the lens into several equal parts and make it into a kind of lens with special optical system. The Fresnel lens uses the special optical principle of lens to produce an alternate changing "blind area" and "high sensitive area" in front of the detector to improve its detecting and receiving sensitivity. When someone walks through in front of the lens, the IR rays emitted by human body will constantly alternate from "blind area" to "high sensitive area". It makes the IR signals received input in the form of strong and weak pulses to enhance its energy range. Using Fresnel lens with amplifying circuit can amplify signals more than 70dB, thus you can measure human motion in the range of 10 to 20 meters.

1* RFID Module

When entering the door, you need to swipe the card. This module is used to identify a visitor’s identity.

1* Power Adapter (5V, 1500mA)

It is used to supply power for the Raspberry Pi, which requires at least 700mA current supply, therefore, power shortages may cause it cannot work normally.

1* USB WIFI Adapter

It is used to make the Raspberry Pi access wireless router and become a terminal device in Local Area Network (LAN).

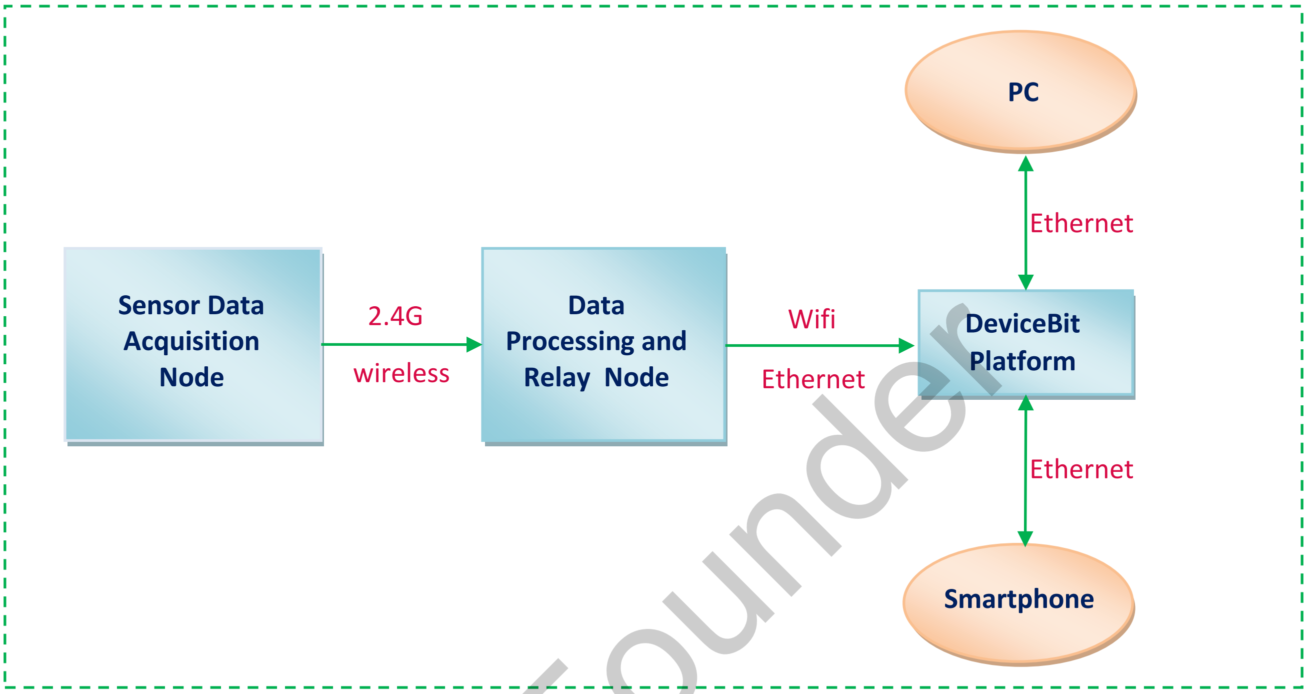
1* USB Cable

It is used to connect Arduino with PC, download applications for Arduino, and communicate with the serial port of PC.

Several Dupont Cables

Operation Principle and System Structure

The Smart Home system is mainly divided into sensor data acquisition node using Arduino Uno as master controller and data processing and relay node using Raspberry Pi as master controller. The system structure diagram is as shown below:



Combined with all kinds of sensors, the Arduino Uno board collects and detects home environment parameters in real time, e.g. temperature, humidity, light intensity, coal gas (natural gas concentration) and so on. Then package the data and send them to Raspberry Pi by nRF24L01 wireless RF module. Raspberry Pi further packages various sensor data and finally sends them to open Internet of Things platform – DeviceBit through network to enable users to access sensor data by PCs and mobile phones so as to control everything at home in real time.

DeviceBit Platform

DeviceBit Platform Introduction

The DeviceBit platform (http://www.devicebit.com/) is a real-time data brokerage platform for the Internet of Things (IoT), providing most of its functionality via its Application Programming Interface (API). It is quick and easy to add Devices and Applications to the DeviceBit platform. It provides real-time data storage and remote control at scale. The DeviceBit platform is not just an easy way to prototype new Internet-enabled sensors; it's also a service that helps companies bring products to market at scale.

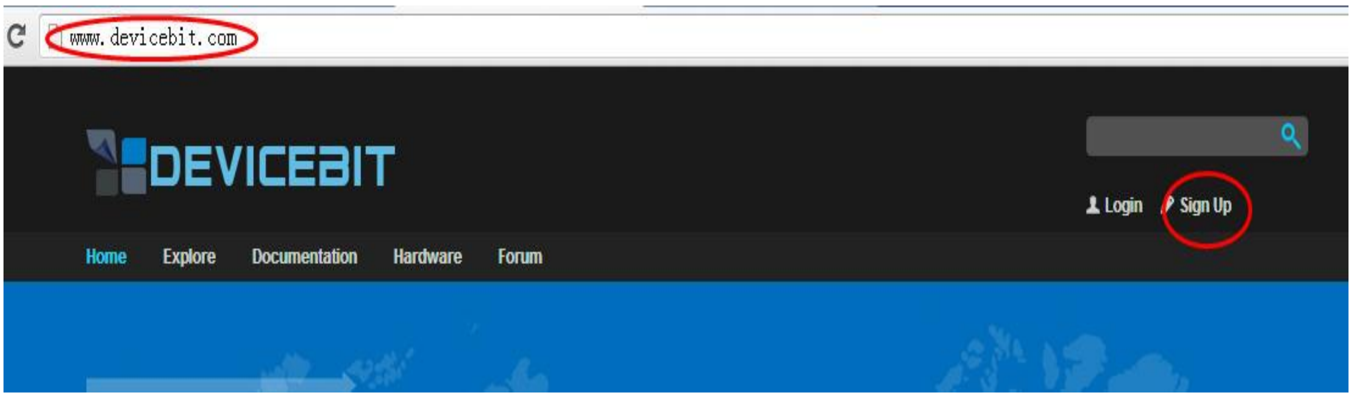
The DeviceBit platform provides basic data analysis tools for rapid data evaluation, as well as real-time alerts and notifications if sensors report "abnormal" conditions.

You can realize your own ideas and develop your own devices relying on this platform. You can focus on hardware instead of software infrastructure.

The DeviceBit platform also communicates with existing social network, such as Twitter and Facebook, allowing you to share what you do with your friend, which might be very helpful to their research on similar field.

The DeviceBit platform uses three-stage IoT architecture, i.e. cloud server – gateway - device. The cloud server is the backstage server of DeviceBit. Gateway is a device used to bridge measurement or control devices and Internet. It refers to Arduino Uno + Raspberry Pi in this application. The device is front-end measuring device or control device. It refers to measuring devices (i.e. all kinds of sensors) in this application.

Register DeviceBit Account



Click **Sign Up** on DeviceBit official website, then register and activate an exclusive DeviceBit account according to the prompt.

Quick Start

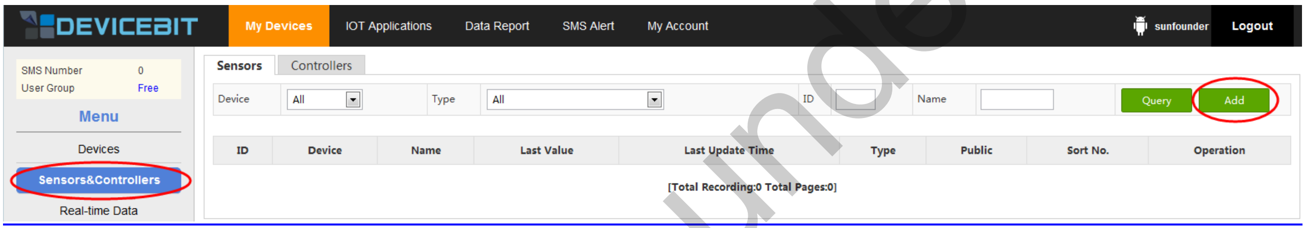
The link below will teach you how to upload data to the DeviceBit platform:
http://www.devicebit.com/dev/doc/171

For more information about how to use the DeviceBit platform, please click the following link:
http://www.devicebit.com/dev/content/about?sk=63

Add Sensors Required

For the detailed steps, please refer to the link above http://www.devicebit.com/dev/doc/171 .

First, add a temperature sensor:

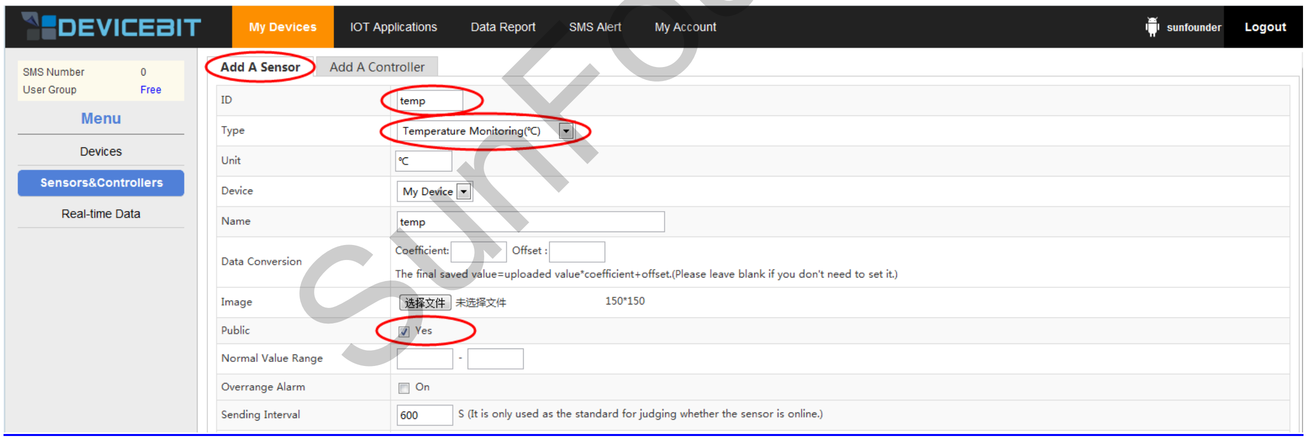


SMS Number 0
User Group Free

Device: All | Type: All | ID | Name

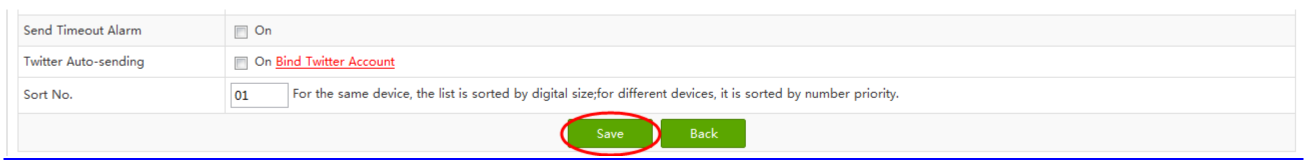
ID	Device	Name	Last Value	Last Update Time	Type	Public	Sort No.	Operation

[Total Recording:0 Total Pages:0]



SMS Number 0
User Group Free

ID	temp
Type	Temperature Monitoring(℃)
Unit	℃
Device	My Device
Name	temp
Data Conversion	Coefficient: Offset: The final saved value=uploaded value*coefficient+offset.(Please leave blank if you don't need to set it.)
Image	选择文件 未选择文件 150*150
Public	[✓ Yes]
Normal Value Range	-
Overrange Alarm	On
Sending Interval	600 S (It is only used as the standard for judging whether the sensor is online.)



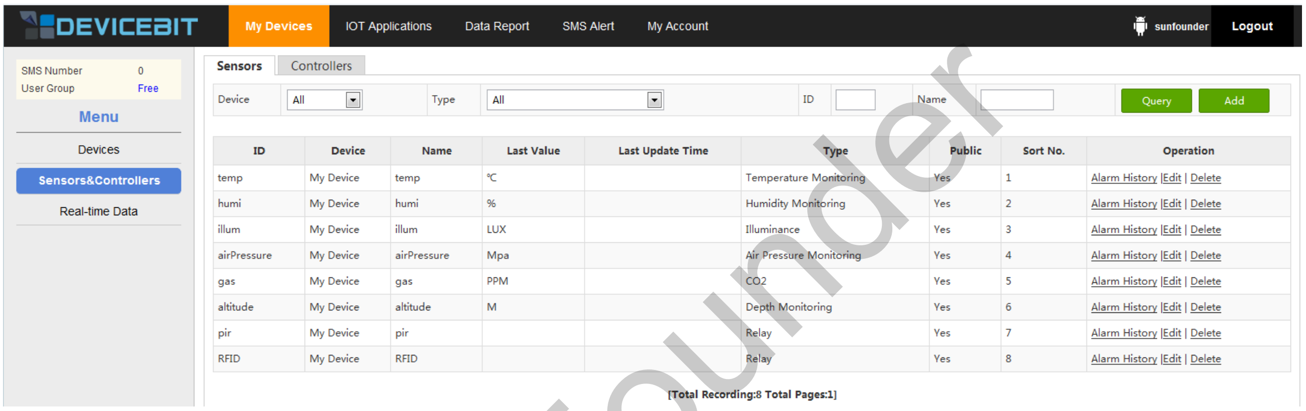
Send Timeout Alarm	On
Twitter Auto-sending	On Bind Twitter Account
Sort No.	01 For the same device, the list is sorted by digital size;for different devices, it is sorted by number priority.

Add the following sensors in turn according to the method above:

Name	ID
humidity	humi

illumination intensity	illum
air pressure	airPressure
noxious gas	gas
altitude	altitude
infrared	pir
RFID	RFID

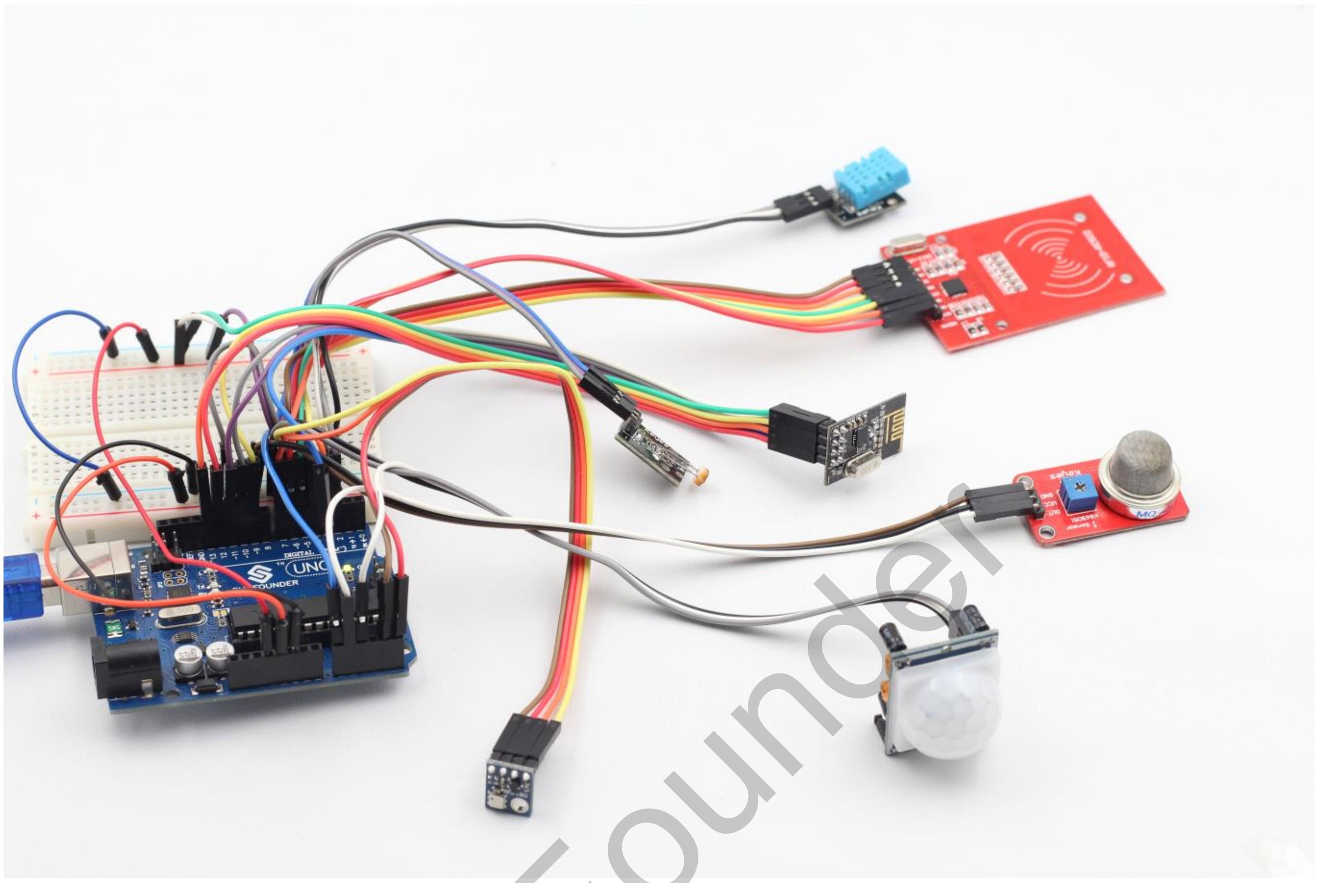
After all the sensors have been added, you will see:



ID	Device	Name	Last Value	Last Update Time	Type	Public	Sort No.	Operation
temp	My Device	temp	℃		Temperature Monitoring	Yes	1	Alarm History |Edit | Delete
humi	My Device	humi	%		Humidity Monitoring	Yes	2	Alarm History |Edit | Delete
illum	My Device	illum	LUX		Illuminance	Yes	3	Alarm History |Edit | Delete
airPressure	My Device	airPressure	Mpa		Air Pressure Monitoring	Yes	4	Alarm History |Edit | Delete
gas	My Device	gas	PPM		CO2	Yes	5	Alarm History |Edit | Delete
altitude	My Device	altitude	M		Depth Monitoring	Yes	6	Alarm History |Edit | Delete
pir	My Device	pir			Relay	Yes	7	Alarm History |Edit | Delete
RFID	My Device	RFID			Relay	Yes	8	Alarm History |Edit | Delete

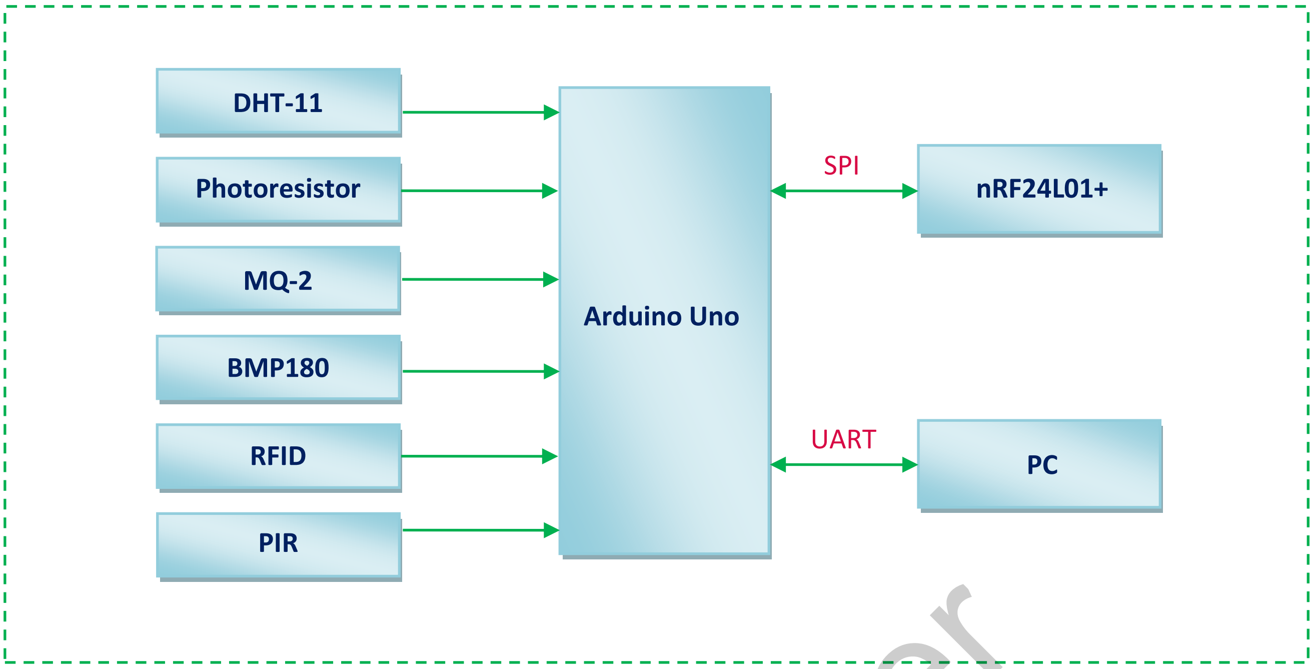
[Total Recording:8 Total Pages:1]

An Introduction to Data Acquisition Node



Data acquisition node uses Arduino Uno as the master controller and combines DHT-11 temperature and humidity sensor, photoresistor, MQ-2 harmful gas sensor, BMP180 gas pressure sensor, RFID module, human body PIR sensor module to collect a variety of environmental data at home in real time, and then send the collected data to data processing and relay node through nRF24L01.

The structure chart of data acquisition node is as shown below:



Circuit Connection

How to connect nRF24L01:

Arduino Uno	nRF24L01 Module
3.3v	3.3v
GND	GND
D9	CE
D10	CSN
D13	SCK
D11	MOSI
D12	MISO

How to connect the sensor:

Arduino Uno	Sensor
D2	DHT-11

A0	Photoresistor
A1	MQ-2
D5	RFID MISO Pin
D4	RFID MOSI Pin
D3	RFID SCK Pin
D6	RFID NSS Pin
D7	RFID RST Pin
A2	PIR
A5	BMP180 SCL Pin
A4	BMP180 SDA Pin

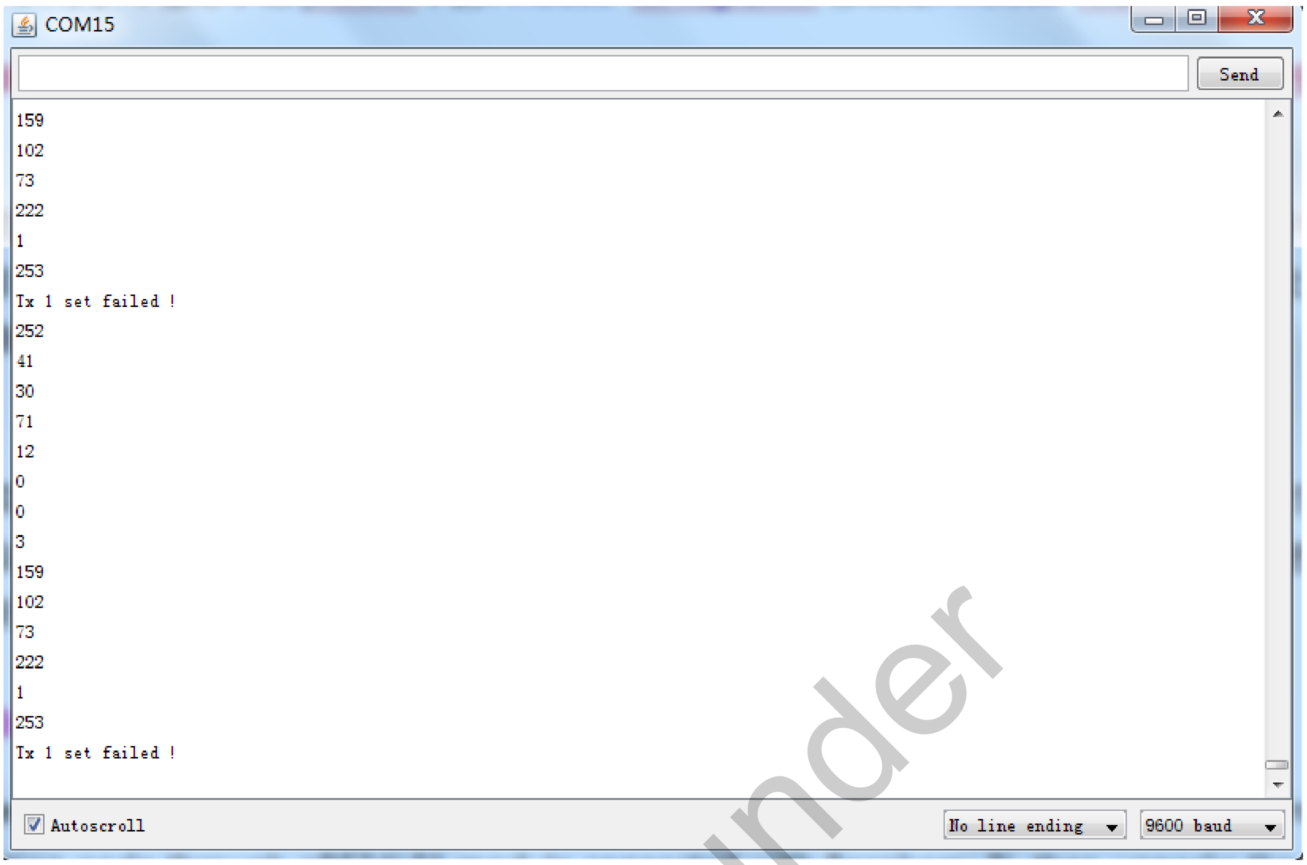
Program and Debug

Please visit the following link to download RF24 library for Arduino platform:
https://github.com/maniacbug/RF24

What we need is five files of RF24.h, RF24.cpp, RF24_config.h, nRF24L01.h and printf.h. Create RF24 folder in *libraries* folder of Arduino, copy the above five files to the folder, then you can see RF24 in Import Library of Arduino IDE. Do not forget to modify #include "WProgram.h" in *printf.h* as #include "Arduino.h".

Find out two zip files named Dht11.zip and RFID.zip in our CD, unzip them, and then copy them to the Arduino *libraries* folder. Close Arduino IDE and reopen it, then the two library files is available.

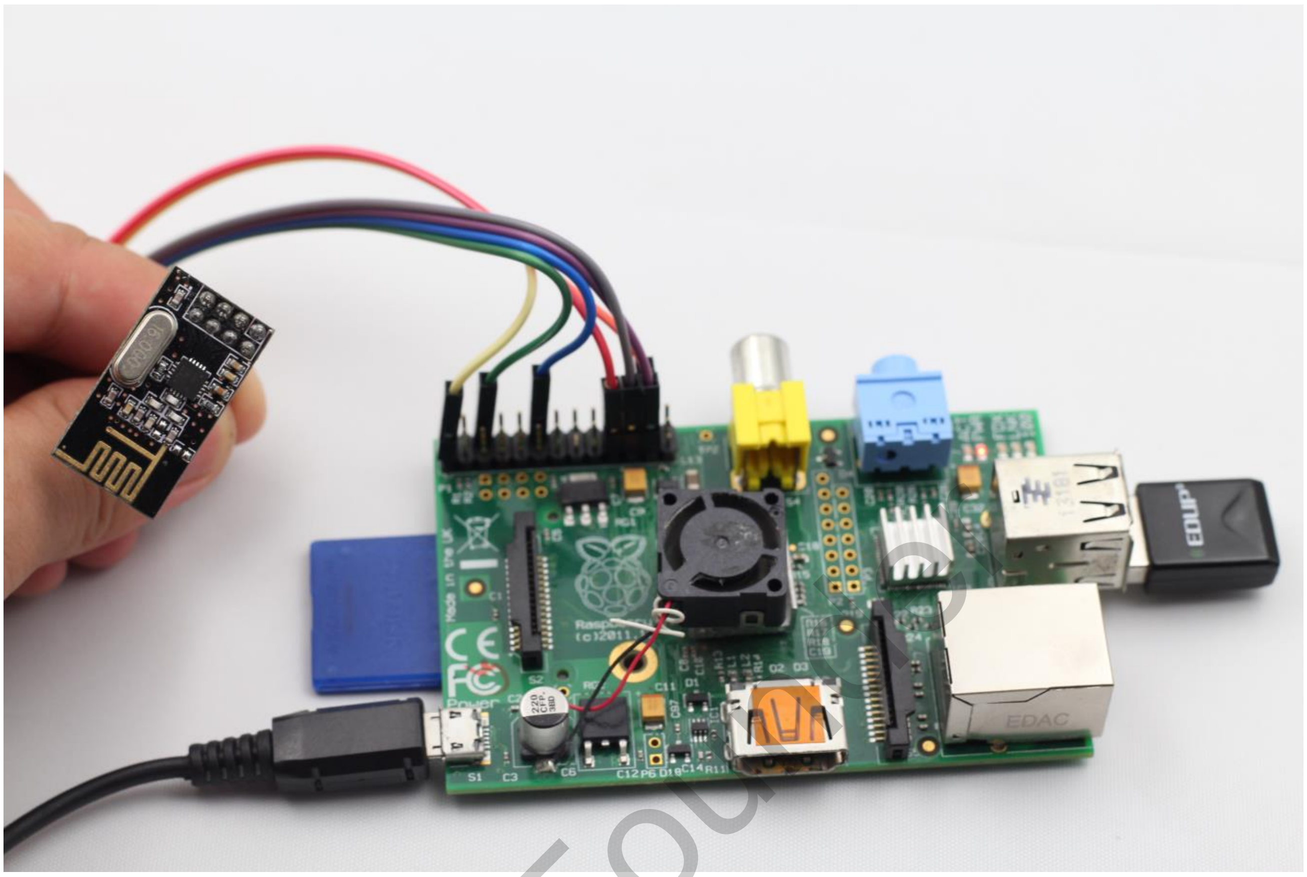
Copy the smartHome_for_Arduino.zip file in the CD attached with Smart Home kit to desktop, unzip the file and open the project in this file, then block nRF24L01 transmitting code and do not start 2.4G wireless transmitting. Compile and run the program. Meanwhile open the serial port debugging interface and print the obtained sensor data on screen via serial port. Check carefully to ensure the sensor work as normal.



```
159
102
73
222
1
253
Tx 1 set failed !
252
41
30
71
12
0
0
3
159
102
73
222
1
253
Tx 1 set failed !
```

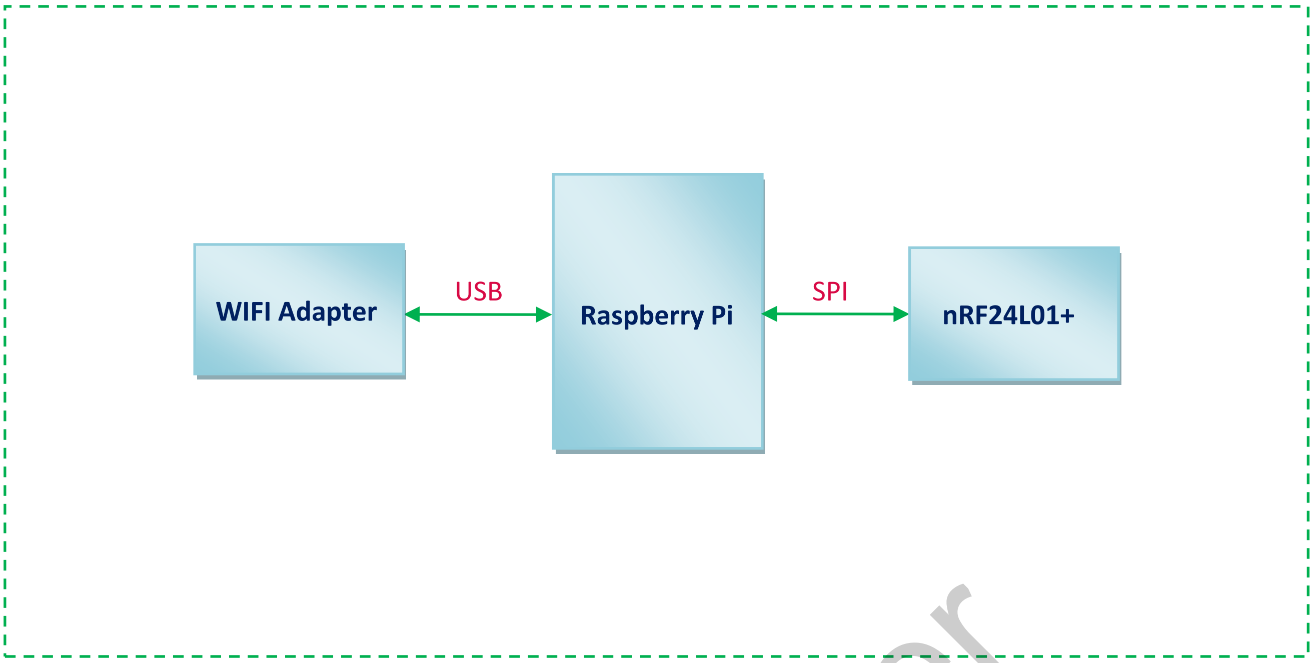
For simplicity, the sensor data is packaged in the form of array. The first element and the last element of the array are used to verify. When receiving data packet, the receiving terminal will judge the first element and the last element of the array. If the two data elements are not appointed data, it will discard the data packet and wait for the arrival of next packet.

Data Processing and Relay Node



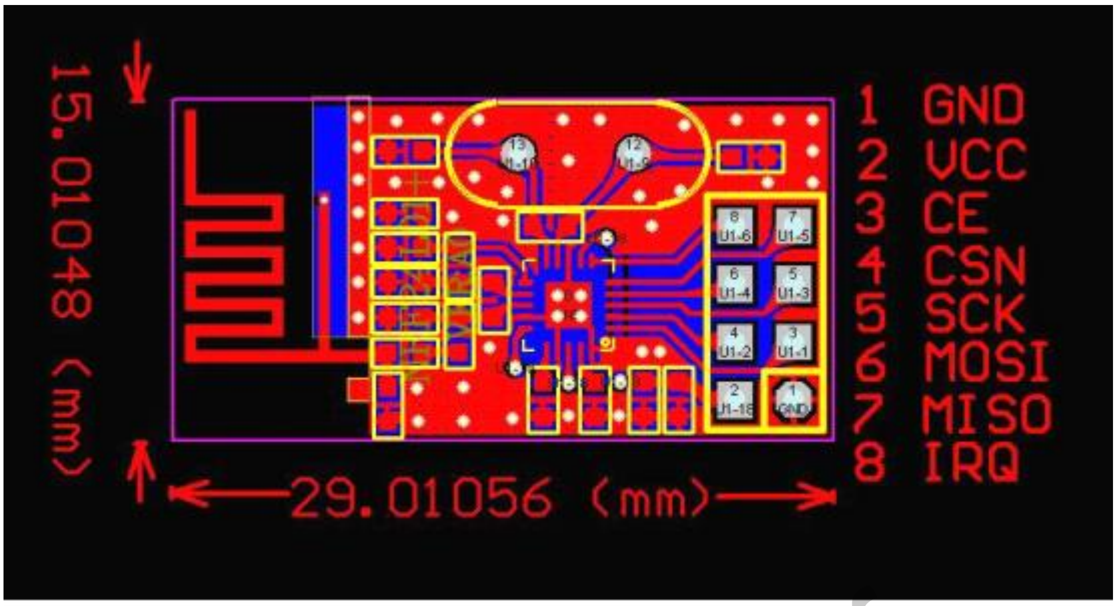
Data processing and relay node uses Raspberry Pi as master controller, and receive data sent from data acquisition node through nRF24L01 module connected with Raspberry Pi, then unpacks the data and writes the file (write sensor data into sensorDat.txt) according to the format specified by DeviceBit platform, and finally call script (upload.sh) program to upload the file with the sensor data saved to DeviceBit server.

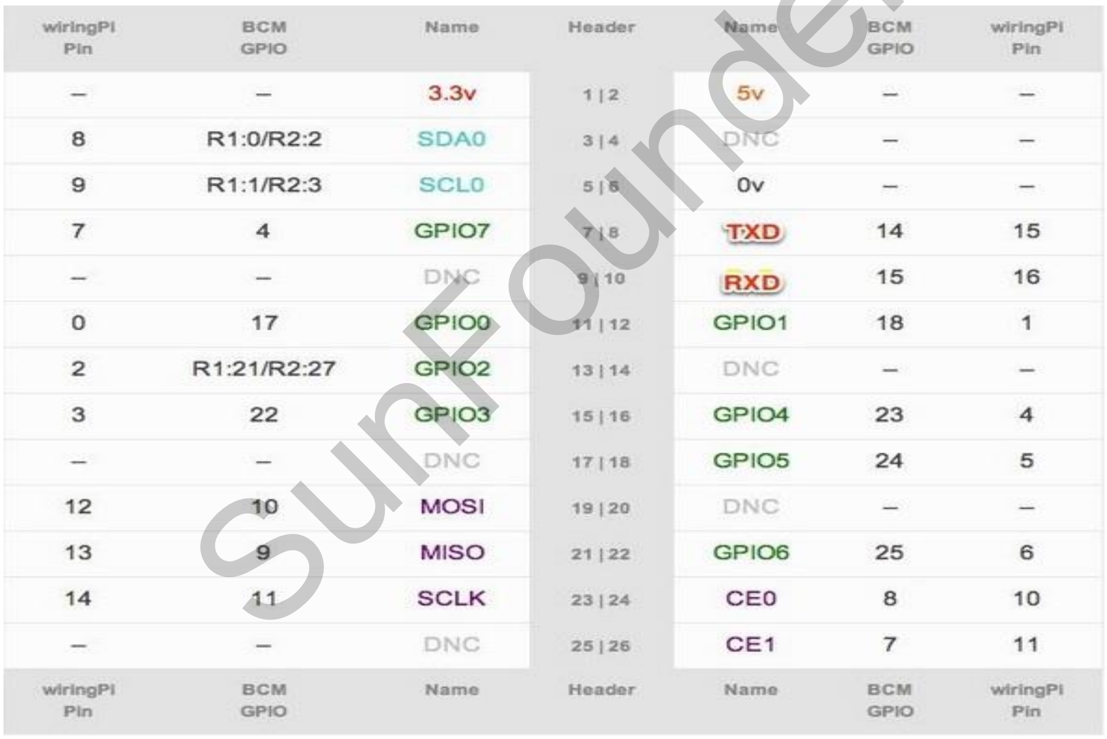
The structure chart of data processing and relay node is as shown below:



Circuit Connection

Raspberry Pi	nRF24L01 Module
3.3v	3.3v
GND	GND
GPIO 18	CE
GPIO 8	CSN
GPIO 11	SCK
GPIO 10	MOSI
GPIO 9	MISO





wiringPi Pin	BCM GPIO	Name	Header	Name	BCM GPIO	wiringPi Pin
–	–	3.3v	1 | 2	5v	–	–
8	R1:0/R2:2	SDA0	3 | 4	DNC	–	–
9	R1:1/R2:3	SCL0	5 | 6	0v	–	–
7	4	GPIO7	7 | 8	TXD	14	15
–	–	DNC	9 | 10	RXD	15	16
0	17	GPIO0	11 | 12	GPIO1	18	1
2	R1:21/R2:27	GPIO2	13 | 14	DNC	–	–
3	22	GPIO3	15 | 16	GPIO4	23	4
–	–	DNC	17 | 18	GPIO5	24	5
12	10	MOSI	19 | 20	DNC	–	–
13	9	MISO	21 | 22	GPIO6	25	6
14	11	SCLK	23 | 24	CE0	8	10
–	–	DNC	25 | 26	CE1	7	11
wiringPi Pin	BCM GPIO	Name	Header	Name	BCM GPIO	wiringPi Pin

Install Raspberry Pi SPI Driver

First, remove SPI driver from system driver blacklist

Open the blacklist file of driver

```
$ sudo vim /etc/modprobe.d/raspi-blacklist.conf
```

Modify the file as follows:

```
# blacklist spi and i2c by default (many users don't need them)

#blacklist spi-bcm2708
```

```
#blacklist i2c-bcm2708
```

Save the file and exit.

Second, load the driver module

Open the file:

```
$ sudo vim /etc/modules
```

Add:

```
snd-bcm2835

spidev
```

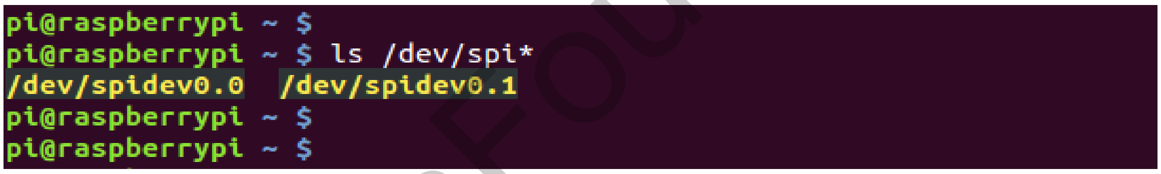
Save the file and exit.

Third, reboot Raspberry Pi

```
$ sudo reboot
```

Fourth, check whether SPI driver is installed successfully or not

```
$ ls /dev/spi*
```



```
pi@raspberrypi ~ $
pi@raspberrypi ~ $ ls /dev/spi*
/dev/spidev0.0  /dev/spidev0.1
pi@raspberrypi ~ $
pi@raspberrypi ~ $
```

If you see the two files (spidev0.0 and spidev0.1), it means you have successfully loaded SPI driver. Next you can normally use SPI device.

Program and Debug

Our code is modified based on an open source nRF24L01 driver code (https://github.com/gnulnulf/RF24) on github. We only modified pingtest.cpp. Please copy smartHome_for_Rpi.tar.gz in CD attached with Smart Home kit to your /home.

```
$ cp your_path/smartHome_for_Rpi.tar.gz /home
```

Unzip the file:

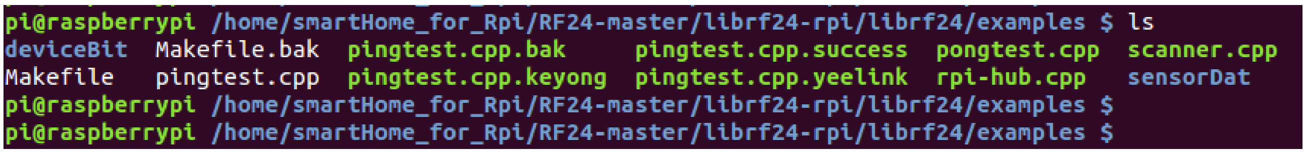
```
$ cd /home

$ sudo tar xvf smartHome_for_Rpi.tar.gz
```

Install nRF24L01 library:

```
$ cd smartHome_for_Rpi/RF24-master/librf24-rpi/librf24/
$ sudo touch -m *

$ sudo make

$ sudo make install
```

Enter the program directory:

```
$ cd yourpath/smartHome_for_Rpi/RF24-master/librf24-rpi/librf24/examples/

$ ls
```



```
pi@raspberrypi /home/smartHome_for_Rpi/RF24-master/librf24-rpi/librf24/examples $ ls
deviceBit  Makefile.bak  pingtest.cpp.bak     pingtest.cpp.success  pongtest.cpp  scanner.cpp
Makefile   pingtest.cpp  pingtest.cpp.keyong  pingtest.cpp.yeelink  rpi-hub.cpp   sensorDat
pi@raspberrypi /home/smartHome_for_Rpi/RF24-master/librf24-rpi/librf24/examples $
pi@raspberrypi /home/smartHome_for_Rpi/RF24-master/librf24-rpi/librf24/examples $
```

Pingtest.cpp is the revised program file. It is used to receive and parse 2.4G wireless data, and upload the sensor data to the DeviceBit platform.

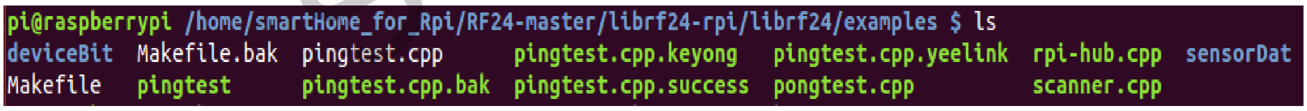
Next, compile the program:

```
$ sudo touch -m *

$ sudo make clean

$ sudo make
```

Wait for a moment until the compiling process is finished.

List the files:

```
$ ls
```



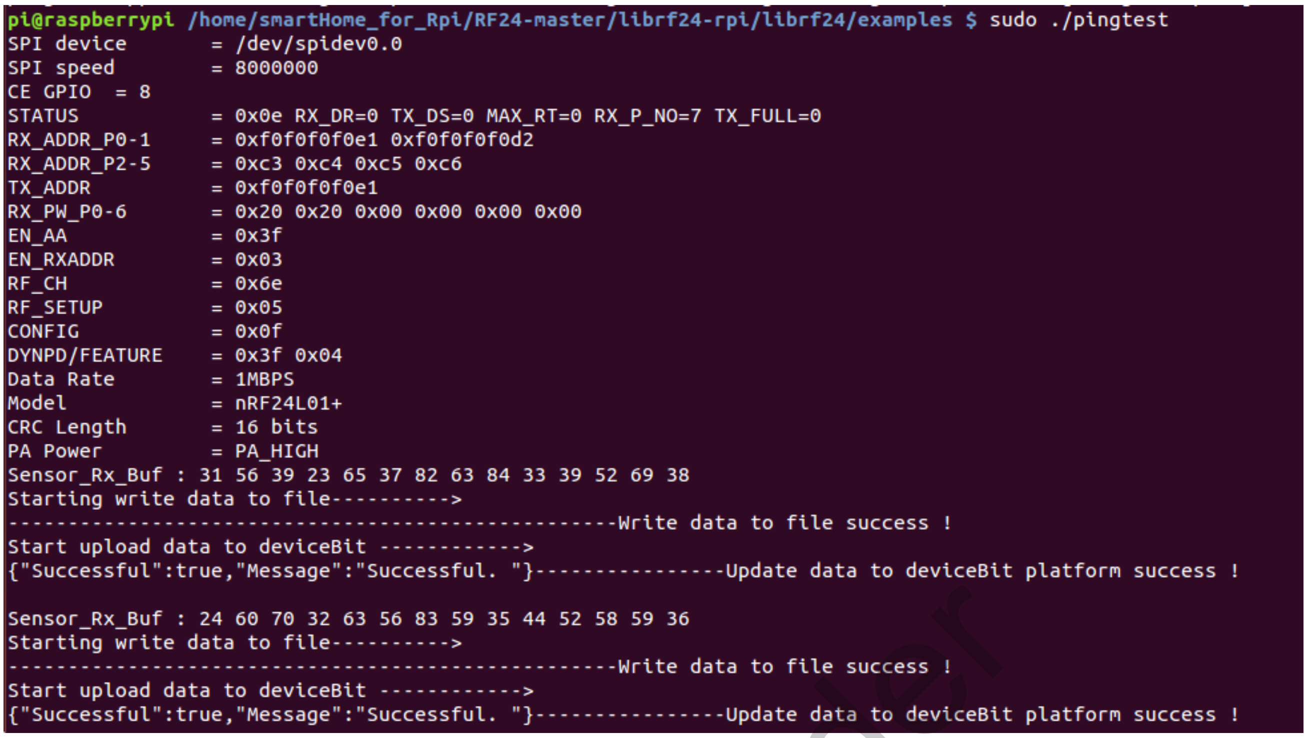
```
pi@raspberrypi /home/smartHome_for_Rpi/RF24-master/librf24-rpi/librf24/examples $ ls
deviceBit  Makefile.bak  pingtest.cpp      pingtest.cpp.keyong   pingtest.cpp.yeelink  rpi-hub.cpp  sensorDat
Makefile   pingtest      pingtest.cpp.bak  pingtest.cpp.success  pongtest.cpp          scanner.cpp
```

The `pingtest` is the executable program file generated after compiling.

Run the program:

```
$ sudo ./pingtest
```

Press Enter, you will see received sensor acquisition data printed on the screen and prompt a message that the sensor data is successfully written to the file and uploaded to DeviceBit platform.

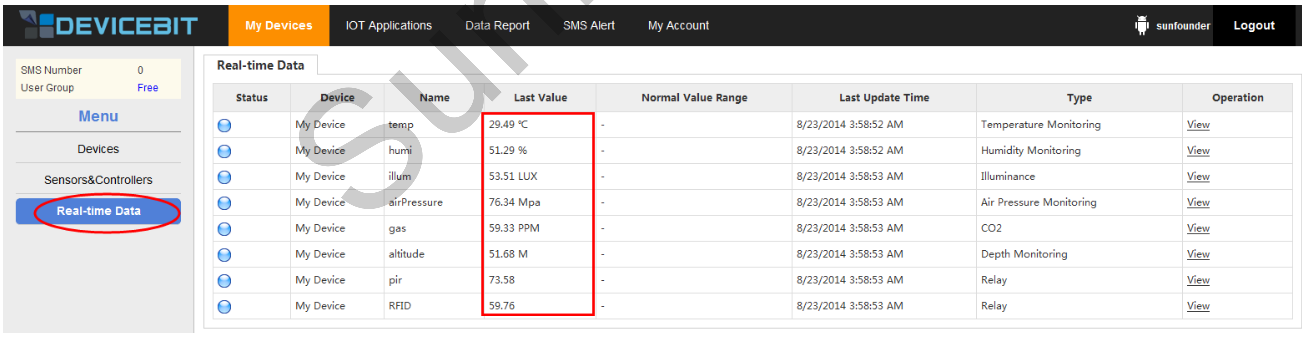


```
pi@raspberrypi /home/smartHome_for_Rpi/RF24-master/librf24-rpi/librf24/examples $ sudo ./pingtest
SPI device       = /dev/spidev0.0
SPI speed        = 8000000
CE GPIO  = 8
STATUS           = 0x0e RX_DR=0 TX_DS=0 MAX_RT=0 RX_P_NO=7 TX_FULL=0
RX_ADDR_P0-1     = 0xf0f0f0f0e1 0xf0f0f0f0d2
RX_ADDR_P2-5     = 0xc3 0xc4 0xc5 0xc6
TX_ADDR          = 0xf0f0f0f0e1
RX_PW_P0-6       = 0x20 0x20 0x00 0x00 0x00 0x00
EN_AA            = 0x3f
EN_RXADDR        = 0x03
RF_CH            = 0x6e
RF_SETUP         = 0x05
CONFIG           = 0x0f
DYNPD/FEATURE    = 0x3f 0x04
Data Rate        = 1MBPS
Model            = nRF24L01+
CRC Length       = 16 bits
PA Power         = PA_HIGH
Sensor_Rx_Buf : 31 56 39 23 65 37 82 63 84 33 39 52 69 38
Starting write data to file---------->
---------------------------------------------Write data to file success !
Start upload data to deviceBit ------------>
{"Successful":true,"Message":"Successful. "}---------------Update data to deviceBit platform success !

Sensor_Rx_Buf : 24 60 70 32 63 56 83 59 35 44 52 58 59 36
Starting write data to file---------->
---------------------------------------------Write data to file success !
Start upload data to deviceBit ------------>
{"Successful":true,"Message":"Successful. "}---------------Update data to deviceBit platform success !
```

Note:

The sensor data printed on the screen is not actual sensor data, but generated data by random number generating functions. This is for debugging without data acquisition node.

Enter your DeviceBit background, click **Real-time Data**, and you will see the latest uploaded sensor data as shown below:

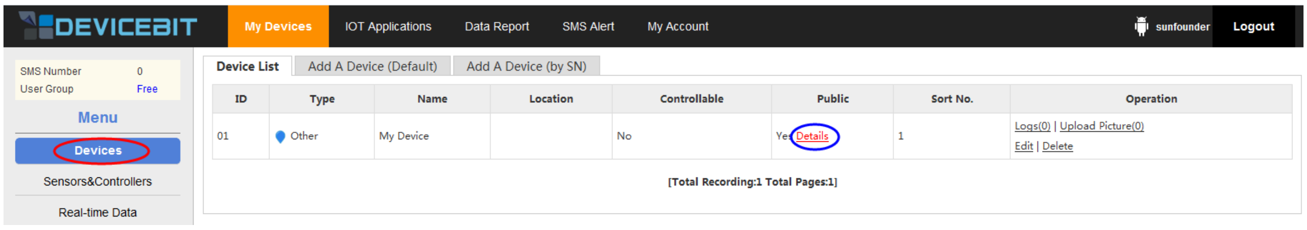


SMS Number 0
User Group Free

Real-time Data

Status	Device	Name	Last Value	Normal Value Range	Last Update Time	Type	Operation
●	My Device	temp	29.49 ℃	-	8/23/2014 3:58:52 AM	Temperature Monitoring	View
●	My Device	humi	51.29 %	-	8/23/2014 3:58:52 AM	Humidity Monitoring	View
●	My Device	illum	53.51 LUX	-	8/23/2014 3:58:53 AM	Illuminance	View
●	My Device	airPressure	76.34 Mpa	-	8/23/2014 3:58:53 AM	Air Pressure Monitoring	View
●	My Device	gas	59.33 PPM	-	8/23/2014 3:58:53 AM	CO2	View
●	My Device	altitude	51.68 M	-	8/23/2014 3:58:53 AM	Depth Monitoring	View
●	My Device	pir	73.58	-	8/23/2014 3:58:53 AM	Relay	View
●	My Device	RFID	59.76	-	8/23/2014 3:58:53 AM	Relay	View

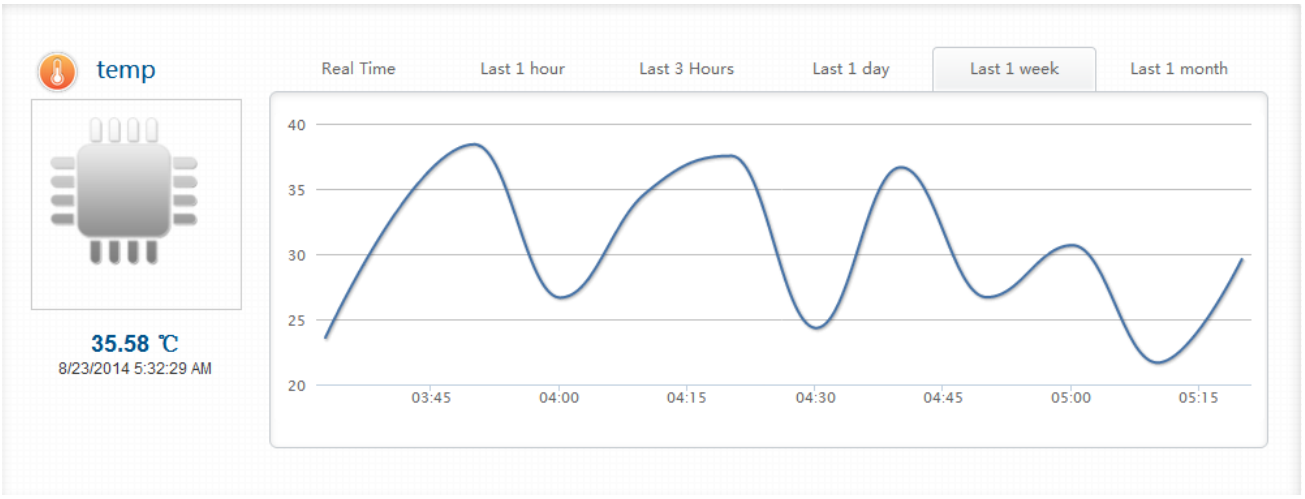
Click **Devices** > **Details**, you will see data form the sensor changes in the form of curve, as shown in the following figure:



SMS Number 0
User Group Free

ID	Type	Name	Location	Controllable	Public	Sort No.	Operation
01	Other	My Device		No	Yes Details	1	Logs(0) | Upload Picture(0) Edit | Delete

[Total Recording:1 Total Pages:1]



Key Subfunction Explanation:

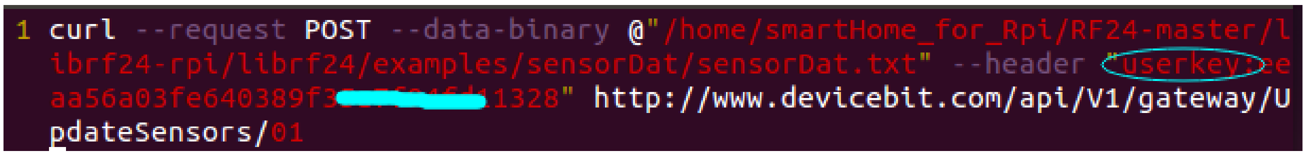
The ch_dat( ) function is used to generate random numbers, simulate received data, and store the data to Sensor_Rx_Buf array. The function is realized by calling rand( ) and srand( ) function. This enables you to debug program completely separated from the data acquisition node, so you can find problems more easily.

The radio.read(Sensor_Rx_Buf , BUFSZ) function is used to read the sensor data received by nRF24L01+, and store the data to Sensor_Rx_Buf array.

The w_sensorDat2file( ) function is used to write the received data from the sensor into sensorDat.txt file in data format specified by the DeviceBit platform.

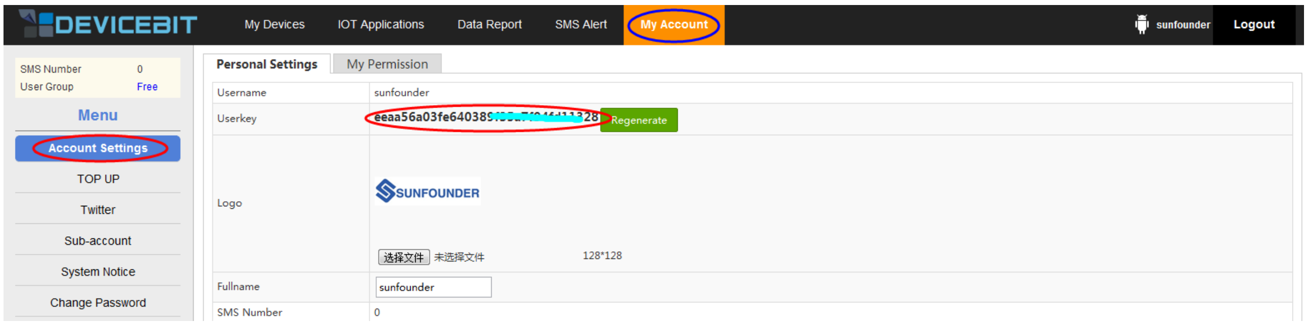
The system("/home/smartHome_for_Rpi/RF24-master/librf24-rpi/librf24/examples/deviceBit/upload.sh") function is used to call upload.sh script and upload sensorDat.txt file to DeviceBit, which has realized package uploading of various sensor data.

upload.sh Script Introduction:



```
curl --request POST --data-binary @"/home/smartHome_for_Rpi/RF24-master/librf24-rpi/librf24/examples/sensorDat/sensorDat.txt" --header "userkey:ee
aa56a03fe640389f3[redacted]1328" http://www.devicebit.com/api/V1/gateway/UpdateSensors/01
```

This script enables you to upload the sensorDat.txt file to the DeviceBit platform. The string POST in the script refers to sending the data (located in /home/smartHome_for_Rpi/RF24-master/librf24-rpi/librf24/examples/sensorDat/sensorDat.txt) to remote server. The userkey is unique, equivalent to a user's password. You must change it into a userkey corresponding with your DeviceBit account, or you cannot upload your data successfully. Please refer to the following diagram to determine your own userkey:

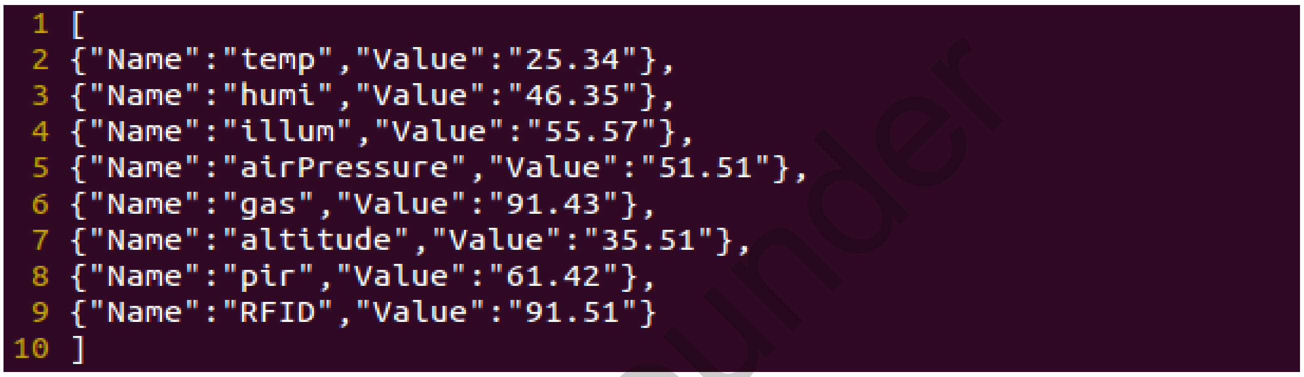


SMS Number	0
User Group	Free

Username	sunfounder
Userkey	eeaa56a03fe64038[redacted]28
Logo	SUNFOUNDER 128*128
Fullname	sunfounder
SMS Number	0

The end of the script "01" is a gat+-eway number, which generally does not need to be modified. The first device gateway that you create on DeviceBit is typically 01.

sensorDat.txt Introduction:

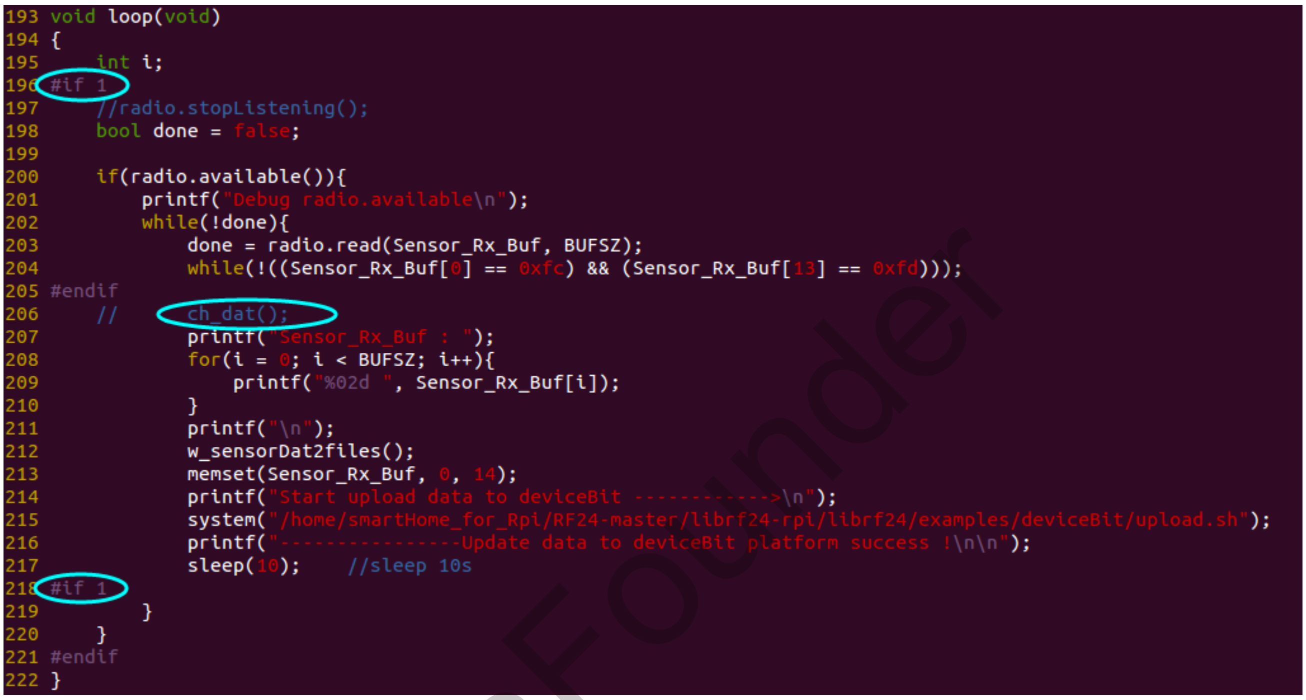

```
[
{"Name":"temp","Value":"25.34"},
{"Name":"humi","Value":"46.35"},
{"Name":"illum","Value":"55.57"},
{"Name":"airPressure","Value":"51.51"},
{"Name":"gas","Value":"91.43"},
{"Name":"altitude","Value":"35.51"},
{"Name":"pir","Value":"61.42"},
{"Name":"RFID","Value":"91.51"}
]
```

The content in this file is written by *pingtest.cpp*. Each pair of **{ }** contains one sensor data. The format of this file is specified by DeviceBit platform. You must write data strictly according to this format. Otherwise the DeviceBit platform will not admit your sensor data.

System Joint Debugging

First, download an application for Data Acquisition Node.

Second, open data processing and relay node program (/home/smartHome_for_Rpi/RF24-master/librf24-rpi/librf24/examples/pingtest.cpp), mask comments about nRF24L01 in the program (refer to the following diagram, change 0 on line 196 and 218 to 1 and mask line 206).



```
193 void loop(void)
194 {
195     int i;
196 #if 1
197     //radio.stopListening();
198     bool done = false;
199 
200     if(radio.available()){
201         printf("Debug radio.available\n");
202         while(!done){
203             done = radio.read(Sensor_Rx_Buf, BUFSZ);
204             while(!((Sensor_Rx_Buf[0] == 0xfc) && (Sensor_Rx_Buf[13] == 0xfd)));
205 #endif
206     //      ch_dat();
207             printf("Sensor_Rx_Buf : ");
208             for(i = 0; i < BUFSZ; i++){
209                 printf("%02d ", Sensor_Rx_Buf[i]);
210             }
211             printf("\n");
212             w_sensorDat2files();
213             memset(Sensor_Rx_Buf, 0, 14);
214             printf("Start upload data to deviceBit ------------>\n");
215             system("/home/smartHome_for_Rpi/RF24-master/librf24-rpi/librf24/examples/deviceBit/upload.sh");
216             printf("----------------Update data to deviceBit platform success !\n\n");
217             sleep(10);    //sleep 10s
218 #if 1
219         }
220     }
221 #endif
222 }
```

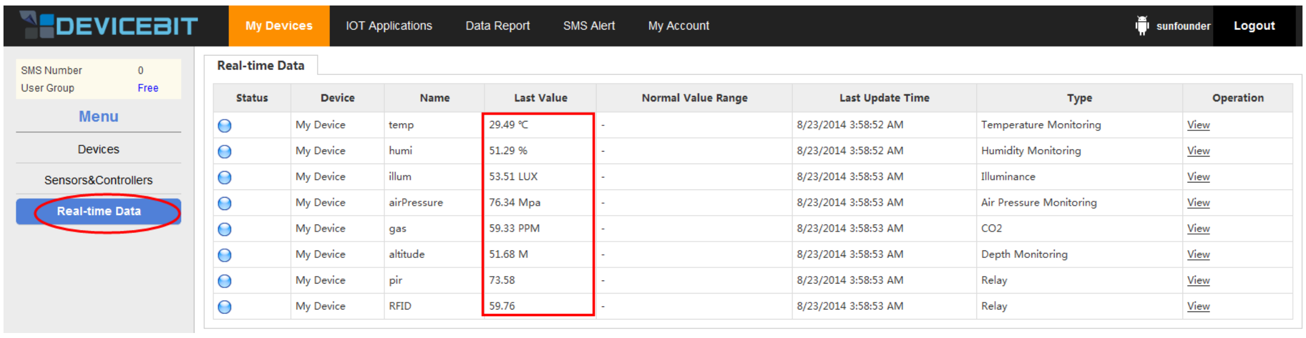
Then recompile the program.

```
$ sudo make clean
$ sudo make
```

Run the program:

```
$ sudo ./pingtest
```

Third, login your DeviceBit platform background management interface, then click **Real-time Data**, now you should be able to see data in the background of DeviceBit platform updating about once every 10 seconds.

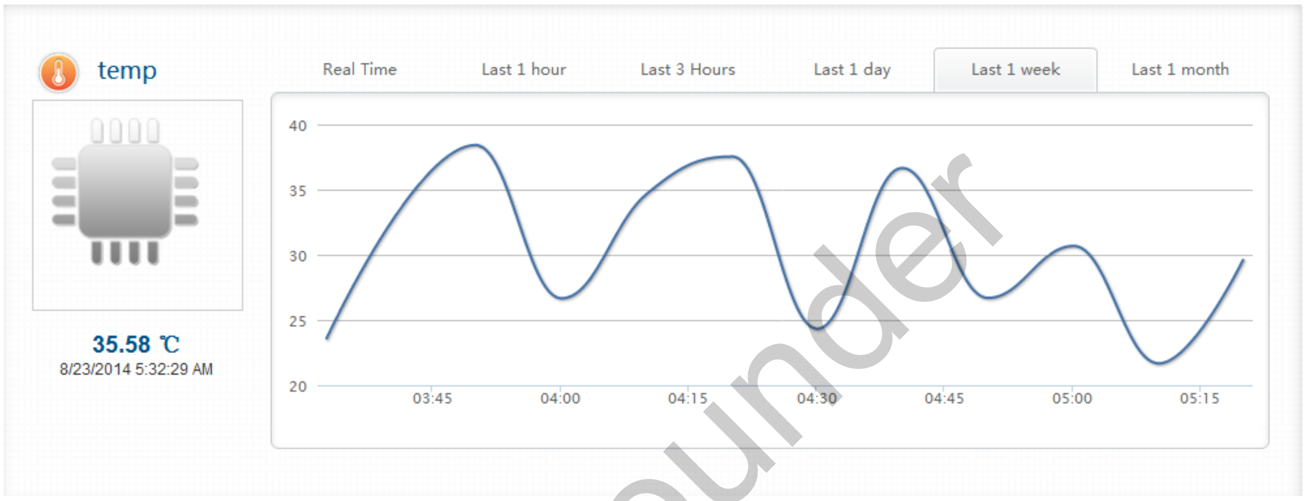


SMS Number	0
User Group	Free

Real-time Data

Status	Device	Name	Last Value	Normal Value Range	Last Update Time	Type	Operation
●	My Device	temp	29.49 ℃	-	8/23/2014 3:58:52 AM	Temperature Monitoring	View
●	My Device	humi	51.29 %	-	8/23/2014 3:58:52 AM	Humidity Monitoring	View
●	My Device	illum	53.51 LUX	-	8/23/2014 3:58:53 AM	Illuminance	View
●	My Device	airPressure	76.34 Mpa	-	8/23/2014 3:58:53 AM	Air Pressure Monitoring	View
●	My Device	gas	59.33 PPM	-	8/23/2014 3:58:53 AM	CO2	View
●	My Device	altitude	51.68 M	-	8/23/2014 3:58:53 AM	Depth Monitoring	View
●	My Device	pir	73.58	-	8/23/2014 3:58:53 AM	Relay	View
●	My Device	RFID	59.76	-	8/23/2014 3:58:53 AM	Relay	View

Click **Devices** > **Details**, you will see the following data curve.



Expansion

The DeviceBit platform also support users to send sensor data to designated email address regularly and to your own Twitter and Facebook. For more information about these contents, please refer to related routines on www.devicebit.com.

Summary

Through learning SunFounder Smart Home Kit, I believe you have made your own simple Smart Home system, experienced the fun of DIY, learnt about the open source platform Arduino and Raspberry Pi and how to use the IoT platform Devicebit. I hope you can make more funny stuff based on these platforms. Have fun!

Postscript

If you have any questions, please send an email to support@sunfounder.com. We will reply to you ASAP!

For more tutorials, please visit our website www.sunfounder.com.

If you've created something you would like to share, you're welcome to post on our website.

Thanks!

SunFounder Technical Support Team