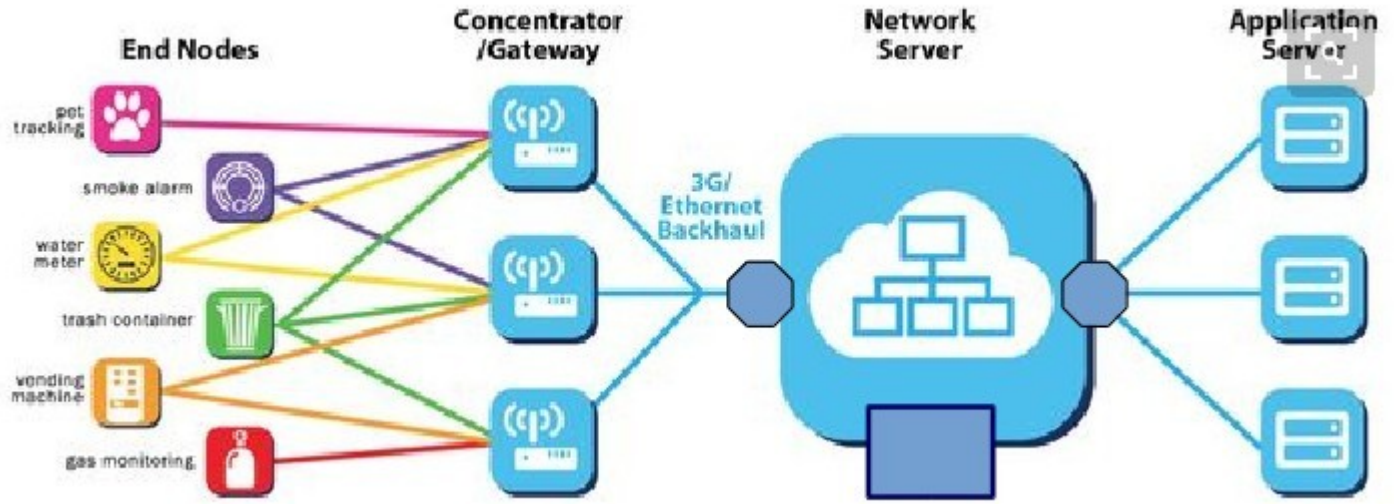
Internet of Things

Connecting Thing to the Internet



Thing:
Sensors/Actuators + MCU

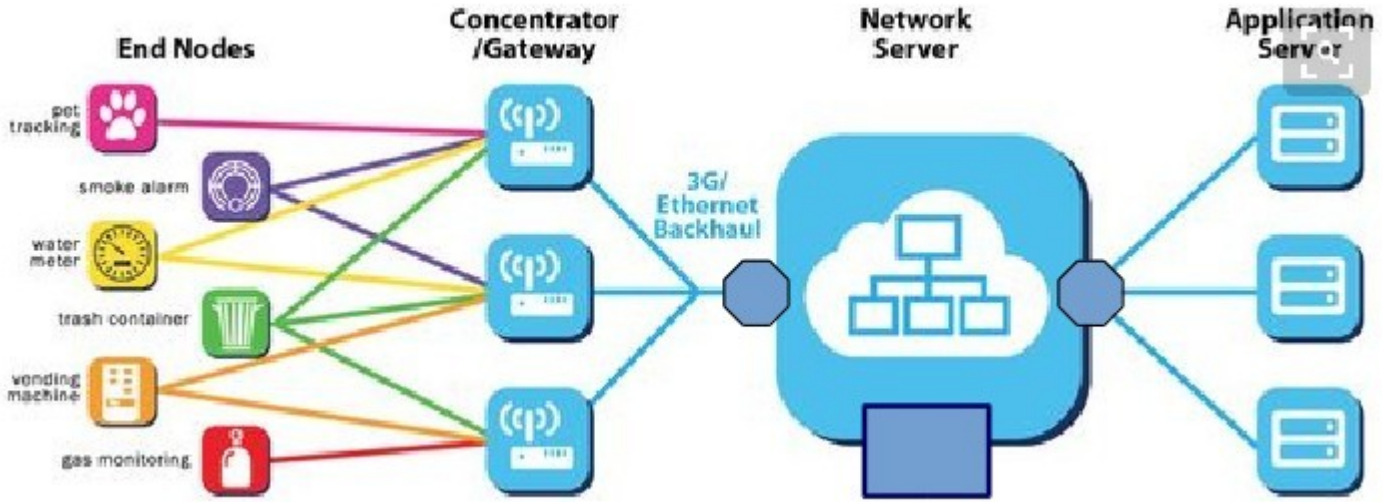
Concentrator/Gateway
SBC

Servers:
ThingSpeak.com/
BakoBox.fr

Clients:
WEB/Android
Laptops/Smartphones

Internet of Things

Connecting Thing to the Internet



Middle Range/Long Range RF links
434+ MHz
868+ MHz

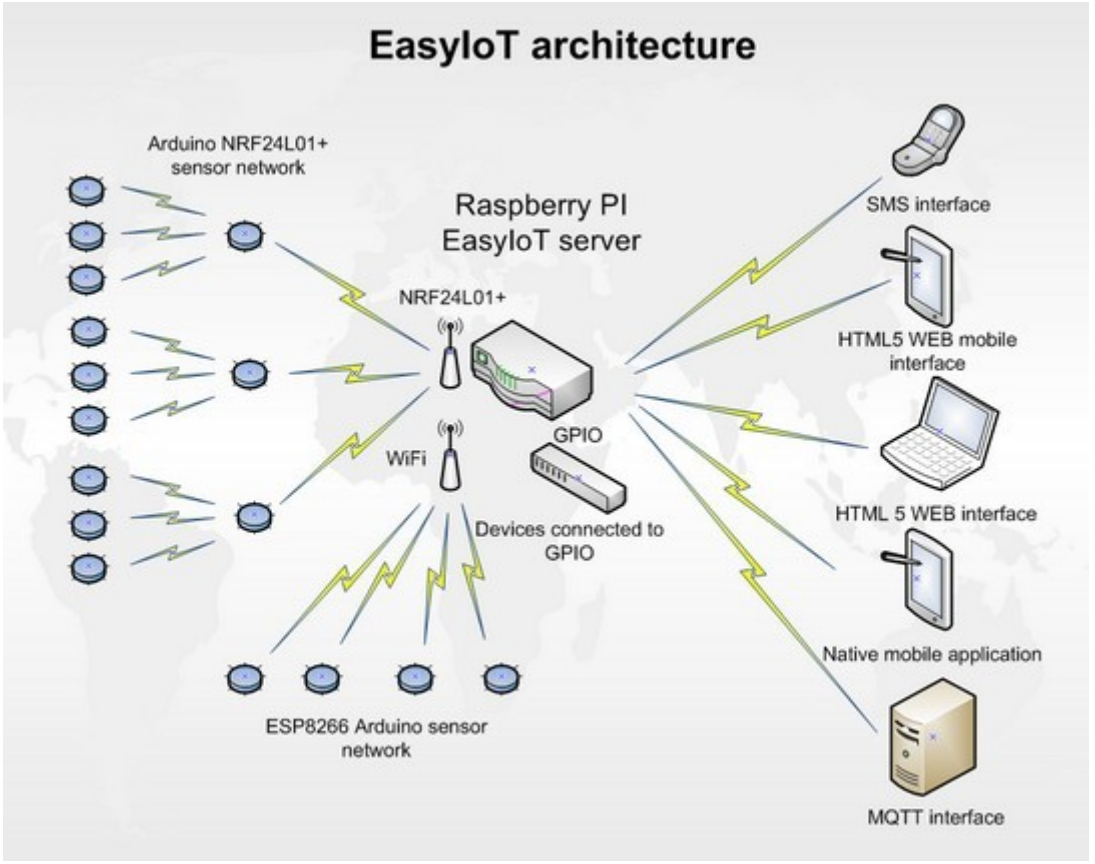
Ethernet/WiFi to Internet links

Internet links

Internet to Ethernet/WiFi links

Internet of Things

IoT servers – EasyIoT

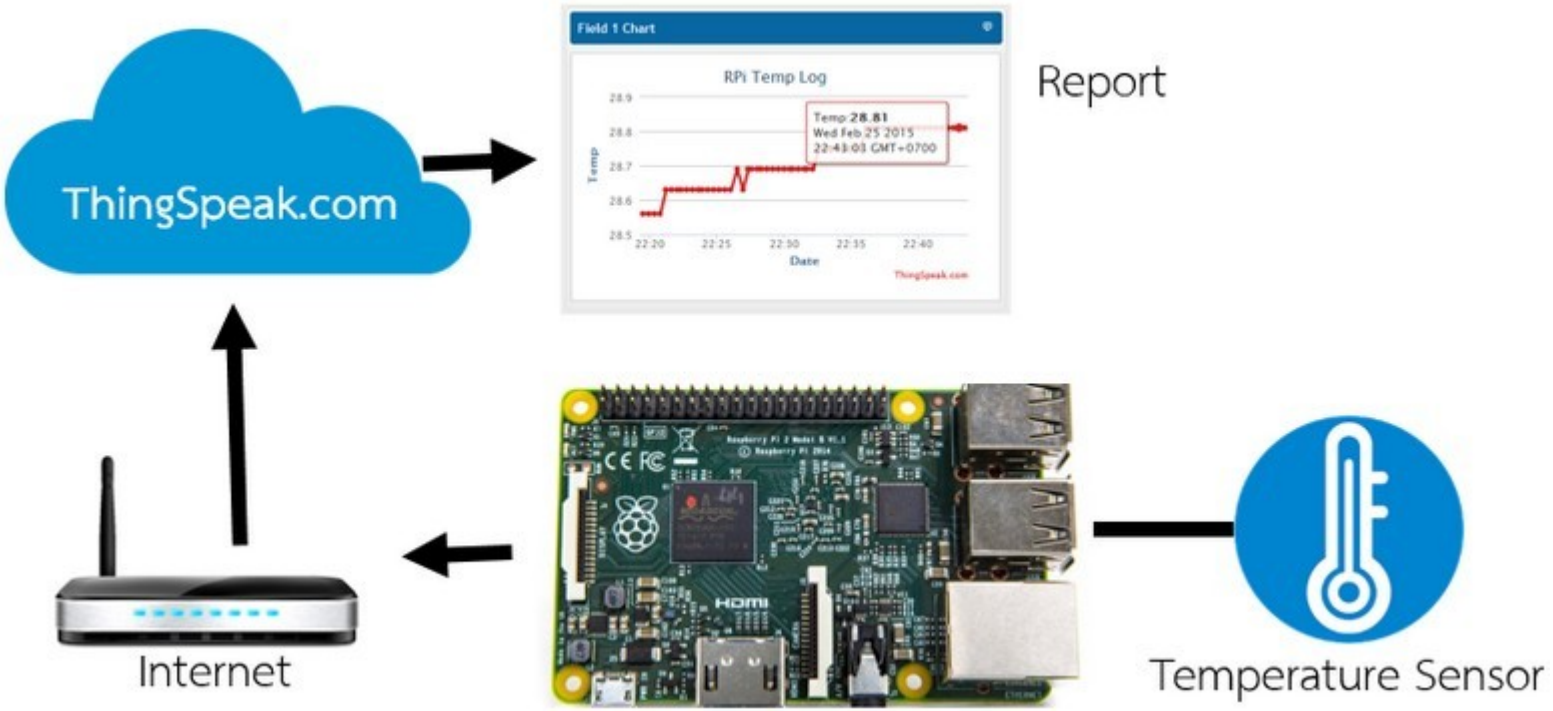


IoT servers

Internet of Things

IoT servers – ThingSpeak.com

MathWorks®



The open IoT platform with MATLAB analytics

Internet of Things

IoT servers – ThingSpeak.com

ThingSpeak™ Channels Apps Community Support

My Channels

New Channel

Name	Created
bakobox Private Public Settings API Keys Data Import / Export	2016-03-05
opiz Private Public Settings API Keys Data Import / Export	2016-06-21
wemos Private Public Settings API Keys Data Import / Export	2016-11-03

Internet of Things

Private View | Public View | Channel Settings | API Keys | Data Import /

Channel Settings

Percentage complete	50%
Channel ID	96085
Name	bakobox
Description	DHT11
Field 1	Temperature ☑
Field 2	Humidity ☑
Field 3	Luminosity ☑
Field 4	☐
Field 5	☐
Field 6	☐
Field 7	☐
Field 8	☐
Metadata	

Internet of Things

bakobox

Channel ID: **96085**
Author: bakobox
Access: Public

DHT11

Private View | Public View | Channel Settings | API Keys | Data Import / E

Write API Key

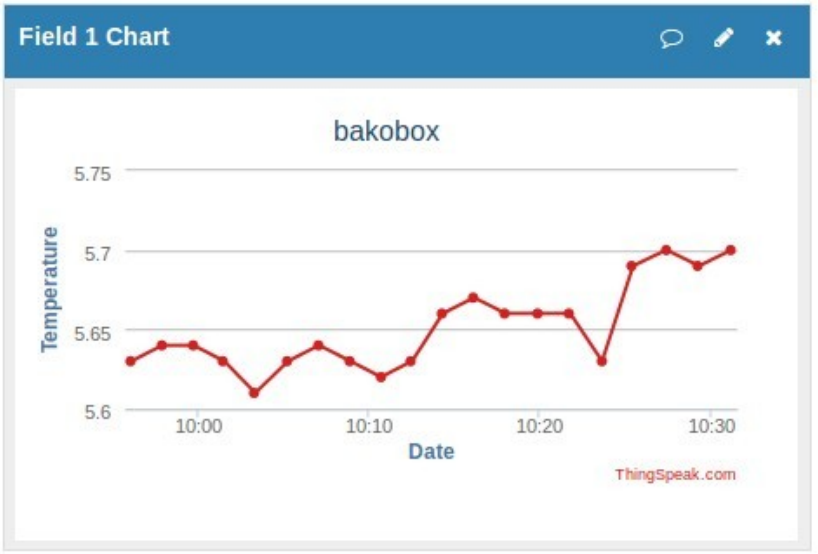
Key: P18HVHSXTYAARD83

Generate New Write API Key

Read API Keys

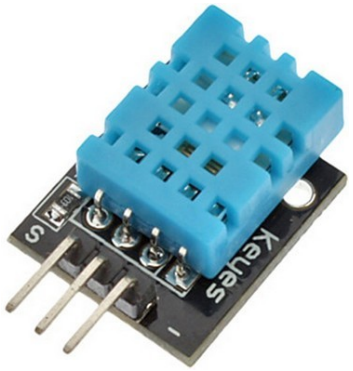
Key: 9DYDN6XM1G5V87BS

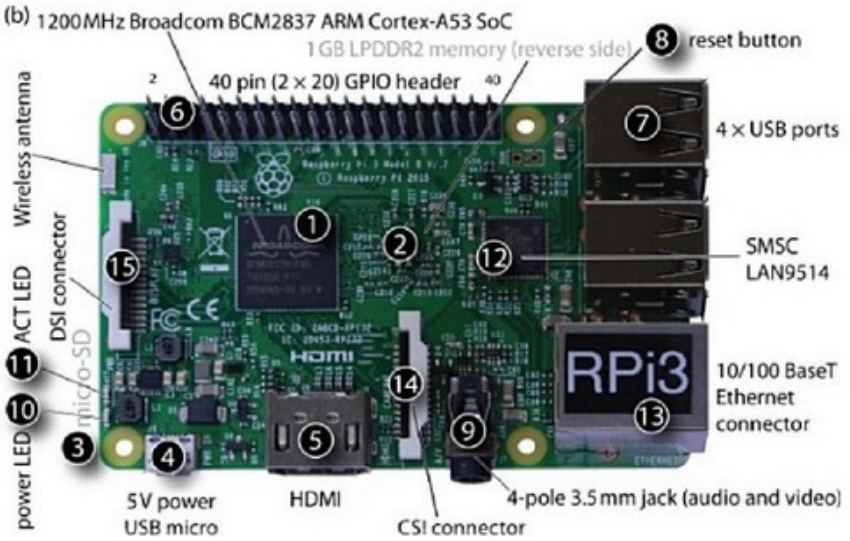
Internet of Things

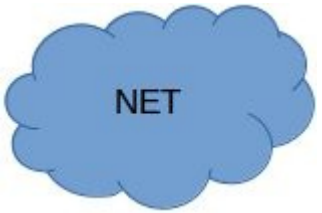


Internet of Things

Basic IoT architecture







proxy

172.19.1.12:3128

Internet of Things

Preparing HTTP request

```
int tssend(char *key,char *field,char *value)
{
int port=3128;  // proxy port
char *tsip="172.19.1.12"; // proxy IP
int sd,i;
struct sockaddr_in sa;
char response[3000+1];  // + 1 is for null
int lsa=16;
char request[100]; char filename[100];

sprintf(filename,"http://184.106.153.149/update?api_key=%s&field%s=
%s",key,field,value);

sprintf(request,"GET %s HTTP/1.1\r\nHost: %s\r\n\r\n",filename,tsip);
printf("%s\n\n",request);
```

request

Internet of Things

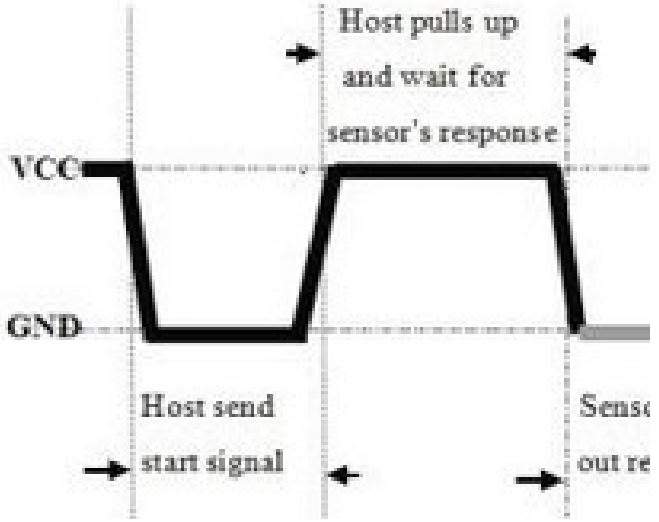
Sending GET request

```
..
sa.sin_family= AF_INET;
sa.sin_port = htons(port);
sa.sin_addr.s_addr=inet_addr(tsip);
sd=socket(AF_INET,SOCK_STREAM,0);

connect(sd,(struct sockaddr *)&sa,lsa);
if(send(sd,request,strlen(request),0) < strlen(request))
{ printf("Send Error!!\n");return(1); }

memset(response,0x00,3000);
if(recv(sd,response,3000,0)==0)
{ printf("Recv Error!!\n");return(2); }
close(sd); // we dont need it any more
if(strlen(response)>522) return 0; else return 1;
}
```

Internet of Things

DHT11/22 serial protocol

https://github.com/dmeziere/rpi-dht/blob/master/util/dht11.c



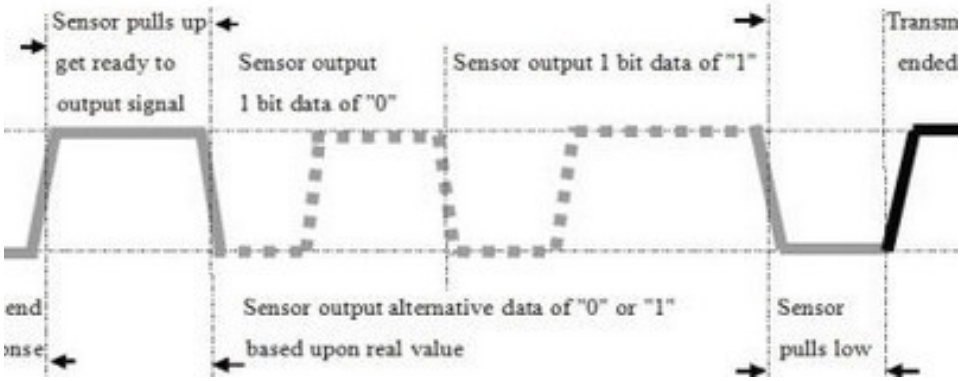

```
#include <wiringPi.h> #include <stdio.h>
#include <stdlib.h> #include <stdint.h>
#define MAXTIMINGS 85
#define DHTPIN     7
int dht11_dat[5] = { 0, 0, 0, 0, 0 };

void read_dht11_dat()
{
    uint8_t laststate  = HIGH;
    uint8_t counter = 0;
    uint8_t j= 0, i;
    float   f;
    dht11_dat[0]=dht11_dat[1]=dht11_dat[2]=dht11_dat[3]=dht11_dat[4]=0;
    pinMode(DHTPIN,OUTPUT);
    digitalWrite(DHTPIN,LOW);  // init transaction
    delay(18);
    digitalWrite(DHTPIN,HIGH); // start transaction
    delayMicroseconds(40);
    pinMode( DHTPIN, INPUT );  // wait for data
..
```

Internet of Things

DHT11/22 serial protocol

```
  for ( i = 0; i < MAXTIMINGS; i++ )
  {
       counter = 0;
       while(digitalRead(DHTPIN) == laststate )
       {
            counter++; delayMicroseconds(1);
            if ( counter == 255 ){break;}
       }
       laststate = digitalRead(DHTPIN);
       if (counter == 255) break;
       if ( (i >= 4) && (i%2 == 0) )
       {
            dht11_dat[j/8] <<= 1;
            if ( counter>16) dht11_dat[j/8] |= 1;
            j++;
       }
  }
  // check for 40 bits : 5*8 bits
  if ((j>=40) &&
(dht11_dat[4]==((dht11_dat[0]+dht11_dat[1]+dht11_dat[2]+dht11_dat[3])&0xFF)))
  {
       printf( "Humidity = %d.%d %% Temperature = %d.%d C\n",
            dht11_dat[0],dht11_dat[1],dht11_dat[2],dht11_dat[3]);
  } else  { printf( "Data not good, skip\n" );}
}
```

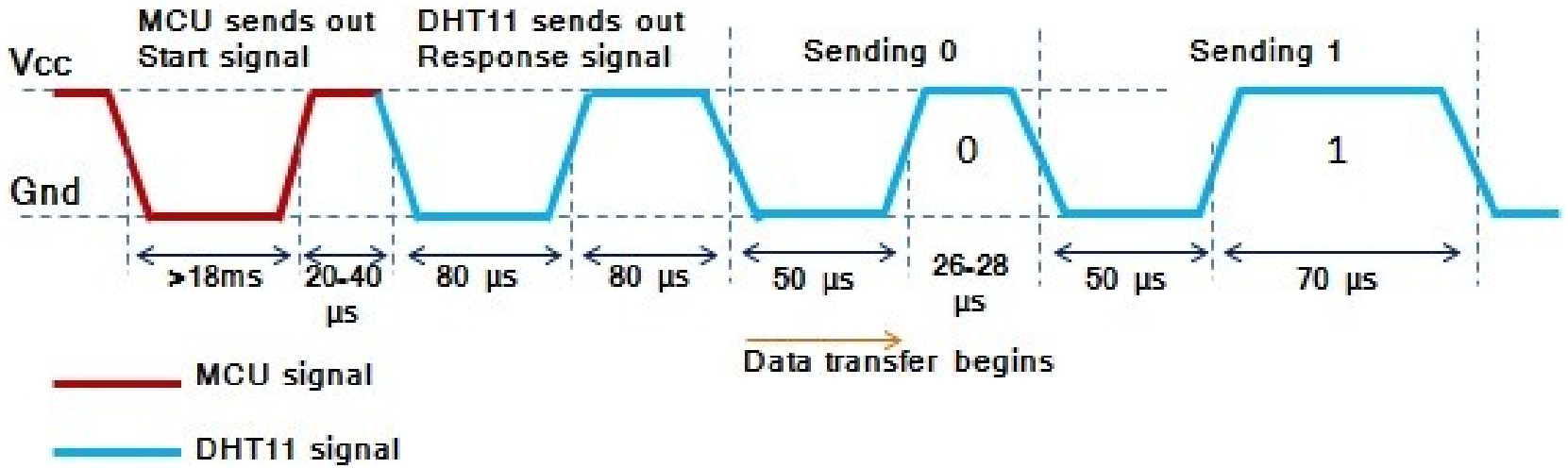
```
uint8_t laststate  = HIGH;
```



A one corresponds to 70 microseconds and a zero corresponds to 26 microseconds.

Internet of Things

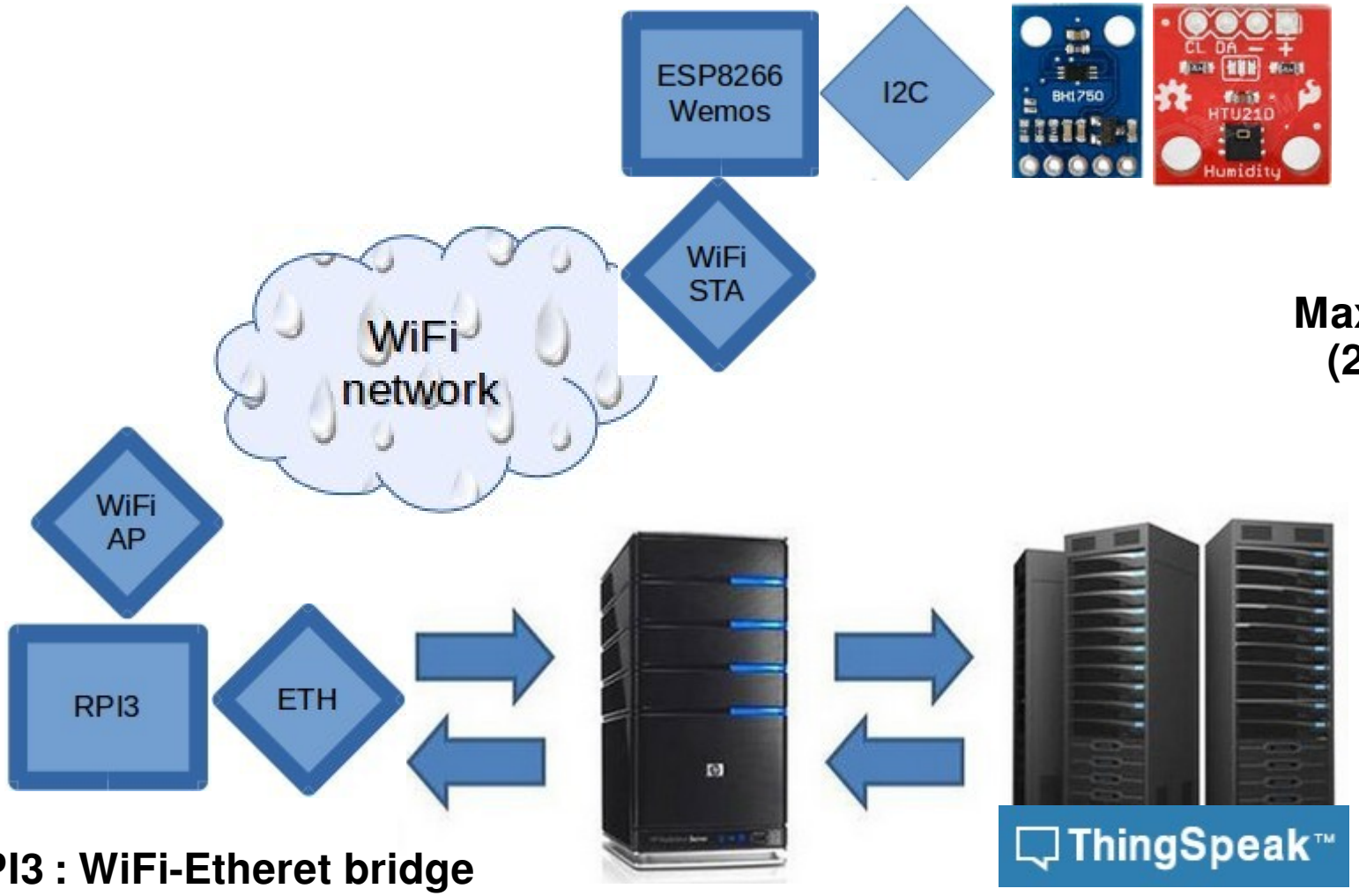
DHT11/22 data frame




```
int main(void)
{
    printf("Raspberry Pi wiringPi DHT11 Temperature test program\n");
    if (wiringPiSetup() == -1) exit(1);
    while (1)
    {
        read_dht11_dat(); delay(1000);
    }
    return(0);
}
```

Internet of Things

IoT terminal and WiFi-Ethernet bridge



Max : 20-50 m
(2.4+ GHz)

RPI3 : WiFi-Etheret bridge

Internet of Things

```
#include <ESP8266WiFi.h>
#include <Wire.h>
#include "Adafruit_HTU21DF.h"
#include <BH1750.h>
Adafruit_HTU21DF htu = Adafruit_HTU21DF();
BH1750 lightMeter;
WiFiClient  client;
//char thingSpeakAddress[] = "api.thingspeak.com";

char proxy[]="172.19.1.12";
int proxy_port=3128;
unsigned long myChannelNumber = 178529;
String  myWriteAPIKey = "77QWEFDSMCU3TMGU";
float temperature=0.0;
float tem,humidity;
const int updateThingSpeakInterval = 16 * 1000;
long lastConnectionTime = 0;
boolean lastConnected = false;
int failedCounter = 0;
```

Internet of Things

```
void updateThingSpeak(String tsData)
{
 // if (client.connect(thingSpeakAddress, 80))
 if (client.connect(proxy, proxy_port))
  {
  client.print("GET http://api.thingspeak.com/update?key=77QWEFDSMCU3TMGU&"+tsData+"
HTTP/1.1\n");
  client.print("Host: api.thingspeak.com\r\n");
  client.print("Connection: close\r\n");
  client.print("\n\n");
    if (client.connected())
    { Serial.println("Connecting to ThingSpeak..."); Serial.println();
      failedCounter = 0;}
    else
    { failedCounter++;
      Serial.println("Connection TS failed ("+String(failedCounter,DEC)+")");
      Serial.println(); }
  }
  else
  { failedCounter++;
    Serial.println("Connection to TS Failed ("+String(failedCounter,DEC)+")");
    Serial.println();
    lastConnectionTime = millis(); }
}
```

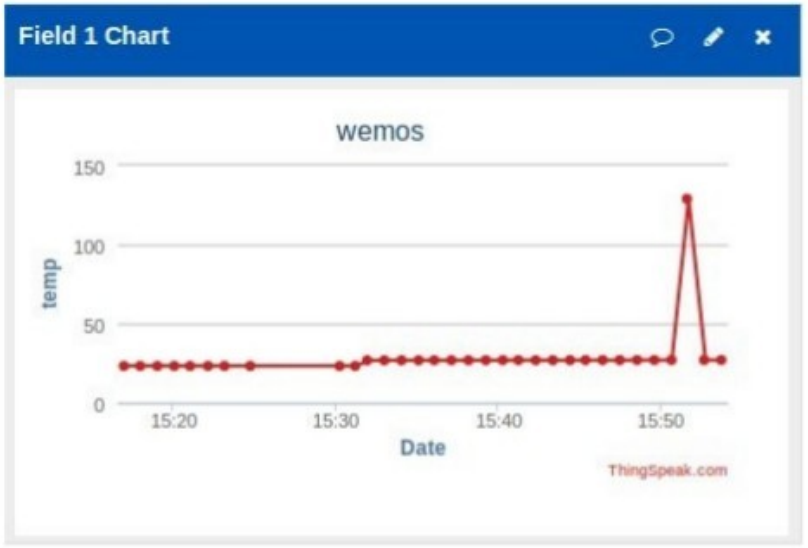
Internet of Things

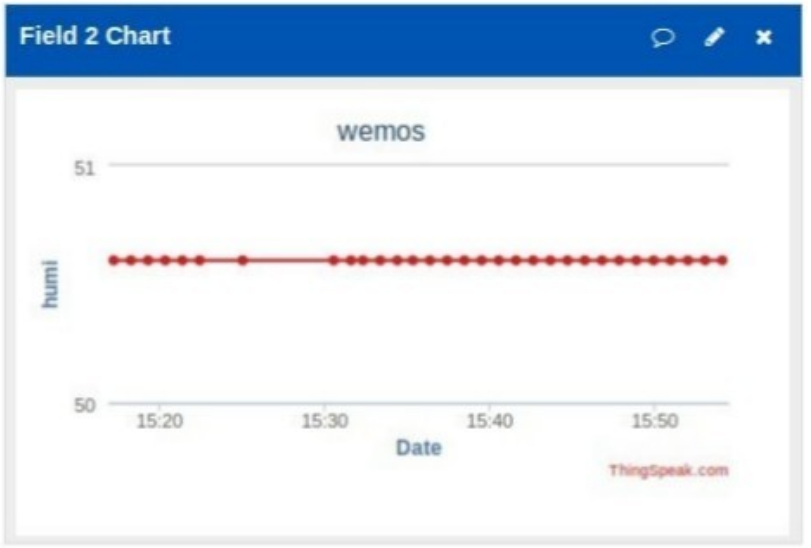
```
void setup()
{
  Serial.begin(9600);
  WiFi.begin("smtr.AP3", "");  // WiFi channe 3
    while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  IPAddress ip = WiFi.localIP();
  Serial.print("IP Address: ");
  Serial.println(ip);
  Wire.begin();
  Serial.println("BH1750 and HTU21D test");
   lightMeter.begin();
  if (!htu.begin()) {
    Serial.println("Couldn't find sensor!");
    while (1);
    }
}
```

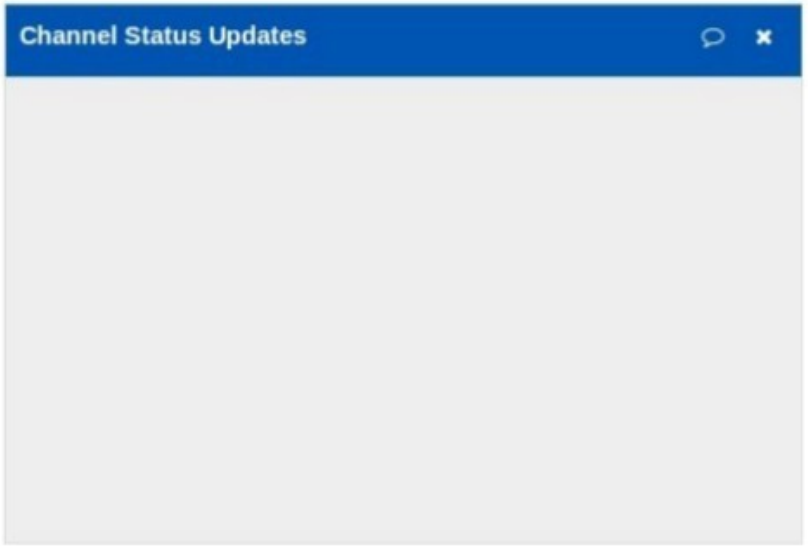
Internet of Things

```
void loop()
{
  uint16_t luminosity=lightMeter.readLightLevel();//uint16_t luminosity= 65;
  Serial.print("Light: ");
  Serial.print(luminosity);
  Serial.println(" lx");
  delay(1000);
  tem = htu.readTemperature(); //tem=23.2;
  Serial.print("Temp: "); Serial.print(tem);
  temperature=(float)tem;
  Serial.print("\t\tHum: ");
  humidity=htu.readHumidity();  // humidity=50.6;
  Serial.println(humidity);
  delay(1000);
  String temp = String(temperature, DEC);
  String lumi = String(luminosity, DEC);
  String humi = String(humidity, DEC);
  Serial.println("update");
  updateThingSpeak("field1="+temp); delay(20000);
  updateThingSpeak("field2="+humi); delay(20000);
  updateThingSpeak("field3="+lumi); delay(20000);
}
```

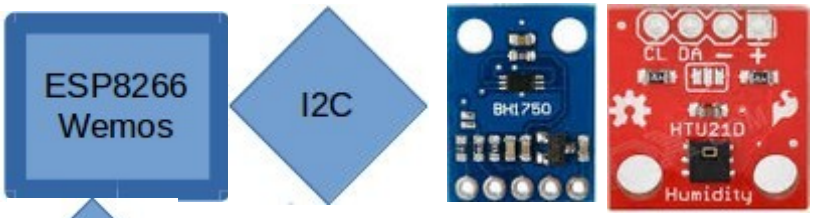
Internet of Things

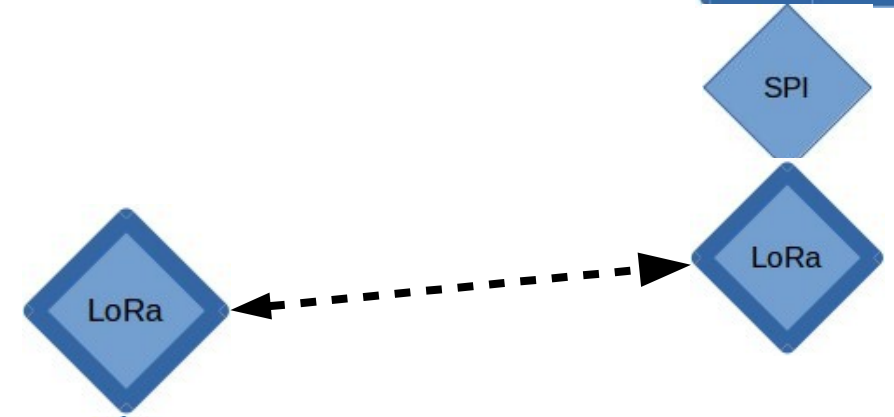






Internet of Things





max : 1000 – 2000 m
(434+ MHz, 868+MHz)

RPI3 : Lora-Ethernet Gateway

Internet of Things

```
#include <SPI.h>
#include <RH_RF95.h>
#include <Wire.h>
#include "Adafruit_HTU21DF.h"
#include <BH1750.h>
Adafruit_HTU21DF thsens = Adafruit_HTU21DF();
BH1750 lsens;

RH_RF95 rf95(D4,D8);   // nodeMCU V1

//RH_RF95 rf95(D0,D8); // Wemos Gateway

//RH_RF95 rf95(D8, D2); //Wemos + module

uint8_t data[16];

float t,h;
int l;   // sensor data
```

Internet of Things

```
int getdata()
{
  uint16_t lux = lsens.readLightLevel();
  l = (int) lux;
  Serial.print("Light: ");
  Serial.print(lux);
  Serial.println(" lx");
  delay(1000);
  t = thsens.readTemperature();
  delay(300);
  h = thsens.readHumidity();
  Serial.print("Temp: "); Serial.print(t);
  Serial.print("\t\tHum: "); Serial.println(h);
  delay(1000);
}
```

l – integer
f,h – floating point

Internet of Things

```
int senddata()
{
  int i=0;
  char st[8],sh[8],sl[8];  // 21.6445.6500024  - 5 5 5 : frame structure
  int tint,tdec;
  int hint,hdec;
  tint= (int) t; hint=(int)h;
  tdec = (int)(100.0*(t - (int)t)); hdec=(int)(100.0*(h -int(h)));
  sprintf(st,"%02d.%02d",tint,tdec);sprintf(sh,"%02d.
%02d",hint,hdec);sprintf(sl,"%05d",l);
  for(i=0;i<5;i++) data[i]=(uint8_t)st[i];
  for(i=0;i<5;i++) data[i+5]=(uint8_t)sh[i];
  for(i=0;i<5;i++) data[i+10]=(uint8_t)sl[i];
  Serial.println(st);
  Serial.println(sh);
  Serial.println(sl);
  rf95.send(data, sizeof(data));
  rf95.waitPacketSent();
  Serial.println("Sent a reply");
}
```

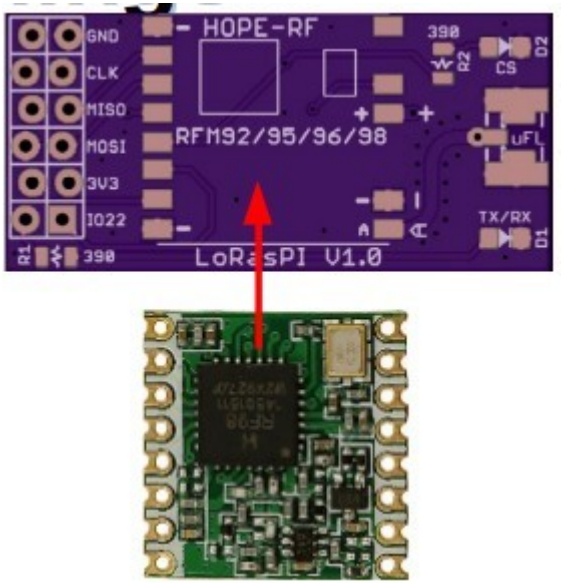
Internet of Things

```
void setup()
{
  Serial.begin(9600);
// sensors init (I2C bus)
  Wire.begin();
  lsens.begin();
  if (!thsens.begin()) {
    Serial.println("Couldn't find HTU21DF sensor!");
  }
// LoRa modem init (SPI bus)
  if (!rf95.init())
    Serial.println("init failed");
  else
    Serial.println("init ok");
}
```

Internet of Things

```
void loop()
{
  if (rf95.available())
  {
    // Should be a message for us now
    uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
    uint8_t len = sizeof(buf);
    if (rf95.recv(buf, &len))
    {
      Serial.print("got request for data ! ");
      Serial.println((char*)buf);
    getdata();
    senddata();
    }
    else
    {
      Serial.println("recv failed");
    }
  }
}
```

Internet of Things



RPI Lora SPI – small shield

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#include <poll.h>
#include <RH_RF95.h>
//RH_RF95 rf95(8,4); // simple module RFM95
// RH_RF95 rf95(8,4); // Lora HAT
RH_RF95 rf95(8,25);   // LoraSpiShield -small shield
//RH_RF95 rf95; // chistera Pi and simple RFM95 module
/* The address of the node which is 10 by default */
uint8_t devnum = 10;
uint8_t msg[32];  // {10, 0};
int mode=1; // default mode
float freq=868.0; // default frequency to be changed for 434.0 modules
uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
uint8_t len = sizeof(buf);
int i=0;
char mbuf[32];
struct pollfd mypoll = { STDIN_FILENO, POLLIN|POLLPRI };
```

Internet of Things

```
void setup(int sc, char *sa[])
{
    wiringPiSetupGpio();
    if (!rf95.init())
    {fprintf(stderr, "Init failed\n");exit(1);}
    else printf("Init ok\n");

    printf("Available modes with default frequency\n");
        printf("Mode 1 (default):BW=125KHz,CR=45,SF=128 £requency:%f\n",freq);
        printf("Mode 2: BW=31.25KHz,CR=48,SF=512 £requency:%f\n",freq);
        printf("Mode 3: BW=125KHz,CR=48,SF=4096 £requency:%f\n",freq);
        printf("Mode 4: BW=500KHz,CR=45,SF=128 £requency:%f\n",freq);
        printf("Mode 5: BW=125KHz,CR=48,SF=1024 £requency:%f\n",freq);
        printf("Usage:\n sudo ./bakolora mode_number frequency\n");

    if(sc==1) { mode=1; freq=868.0; }
    if(sc==2) { mode=atoi(sa[1]); freq=868.0; }
    if(sc==3) { mode=atoi(sa[1]); freq=atof(sa[2]); }
    if(mode==1)
        {
        rf95.setModemConfig(RH_RF95::Bw125Cr45Sf128);
        printf("Mode 1: BW=125KHz, CR=45, SF=128 -frequency:%f\n",freq);
        }
..
```

Internet of Things

```
..
if(mode==5)
        {
        rf95.setModemConfig(RH_RF95::Bw125Cr48Sf1024);
        printf("Mode 5: BW=125KHz, CR=48, SF=1024 -frequency:%f\n",freq);
        }
rf95.setFrequency(freq);
if(freq>=433.0 && freq<435.0) rf95.setTxPower(14);  //RFM98 (RM98)
if(freq>=868.0 && freq<=870.0) rf95.setTxPower(23); //RFM95 (RM95)
if(freq<433.0 || (freq>435.0 && freq<868.0) || freq>870)
        {
        printf("wrong frequency\n");
        printf("must be between 433.0 and 435.0 or 868.0 and 870.0 !\n");
        exit(1); }
}
```

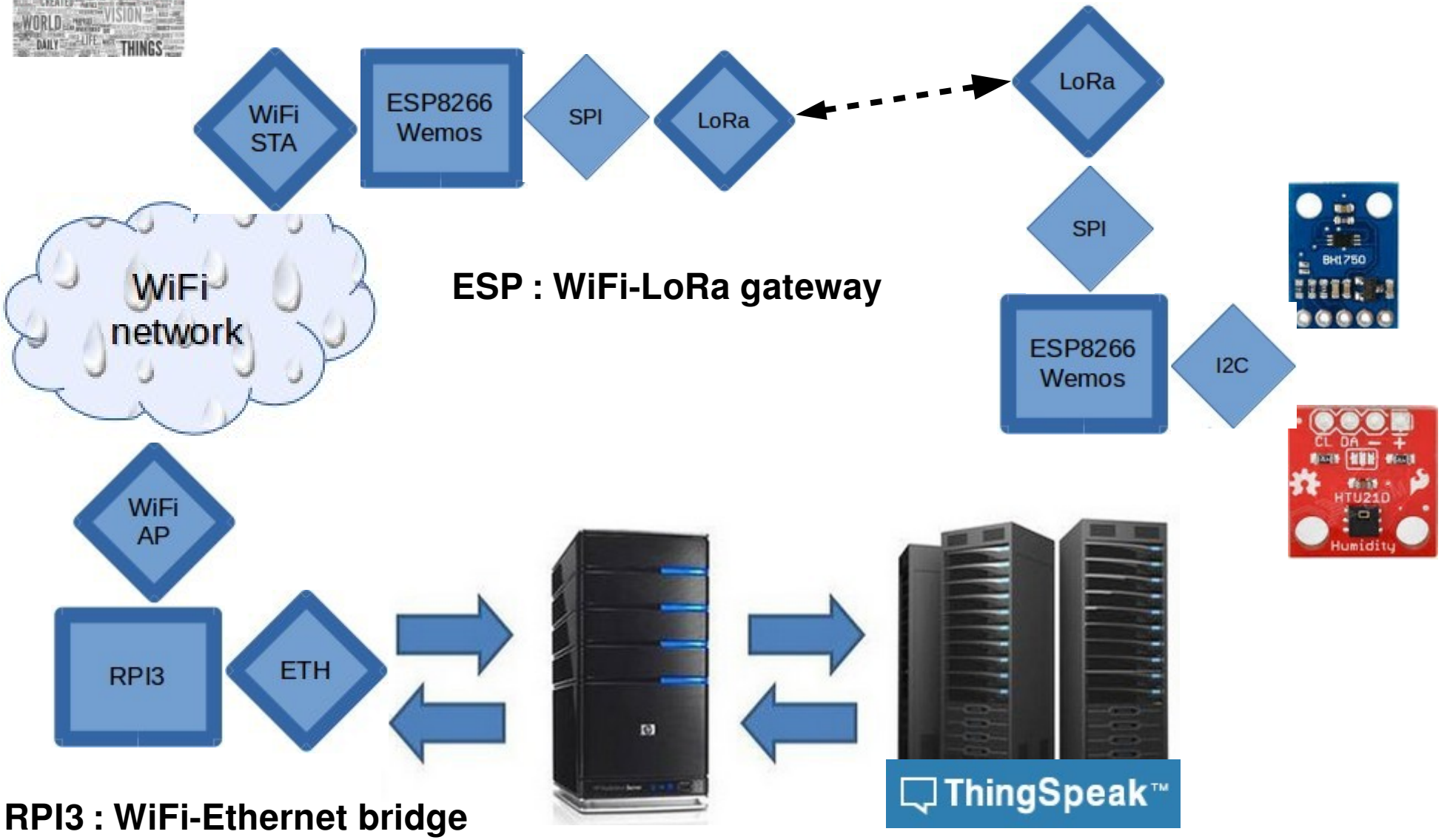
Internet of Things

```
void loop()
{
while(1)   // sending a message and receiving the replay message
  {
    msg[0] = devnum;
    strcpy(mbuf+1, "request_to_send_data");
    for(i=0;i<strlen(mbuf)+1;i++) msg[i+1]=(uint8_t)mbuf[i];
    rf95.send(msg, sizeof(msg));
    rf95.waitPacketSent();
    printf("Send!\n");
    usleep(10);
  if( rf95.waitAvailableTimeout(3000) )
  {
  memset(buf,0x00,len);
  if (rf95.recv(buf, &len))
    { printf("got reply RSSI= %d\n", rf95.lastRssi());
      printf("%s\n", buf); }
 // buf contains the data to be sent by
 // tssend(char *key,char *field,char *value) – 5 characters ex 45.76 or 00065
  else printf("rcv failed\n");
  }
  else printf("no reply\n"); sleep(3);
  }
}
```

Internet of Things

```
int main(int argc, char **argv)
{
if(argc==2) devnum=atoi(argv[1]);
setup(argc,argv);
loop();
return EXIT_SUCCESS;
}
```

Internet of Things



ESP : WiFi-LoRa gateway

RPI3 : WiFi-Ethernet bridge

Internet of Things

In this architecture RPI3 operates (once more) as a transparrent bridge.

To build a Wemos gateway we need to complete and/or modify the **`loop()`** function to receive the data from LoRa modem :

`recvdata()`

This function will get the data from LoRa modem to be sent via the Wifi interface with the HTTP requests.

The request will be forwarded to the Ethernet Interface and sent to the ThingSpeak server (via the proxy!)

Remark :
Each value to send to a channel field is a chain of 5 characters, ex : 23.43 or 00054

Summary

- Global IoT architectures with IoT servers
- `ThingSpeak.com` server – channels and fields
- A basic IoT architecture with a sensor and RPI3 to `ThingSpeak.com`
- A simple IoT architecture with one terminal and RPI3 bridge to `ThingSpeak.com`
- An IoT architecture with one terminal , LoRa connection and RPI3 gateway to `ThingSpeak.com`
- An IoT architecture with a terminal, LoRa connection , LoRa to WiFi gateway and an RPI3 bridge to `ThingSpeak.com`