Снятие показаний с погодного датчика Oregon THN132N (Погодная станция Oregon Scientific Bar339P)

Краткое описание работы

1. Считываем показание с датчика
2. Формируем строку для передачи на сервер вида:
`deviceid=EA4C108122234030&channel=1&temperature=22&battery=90`, где:

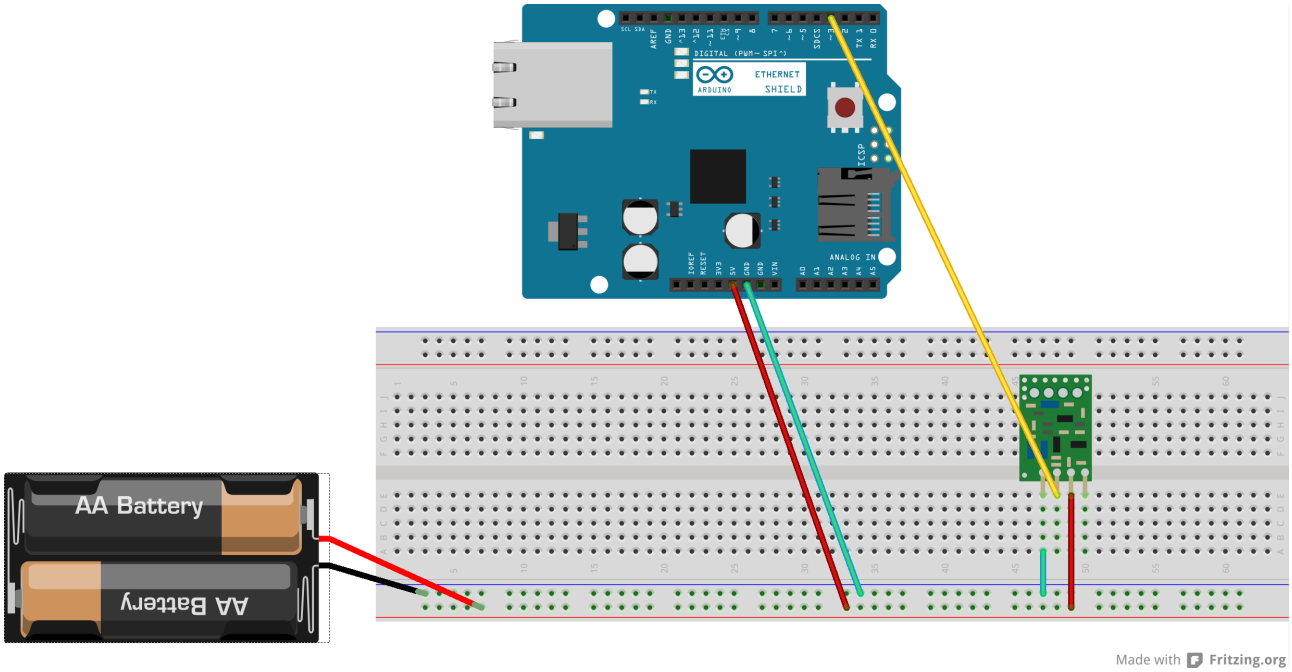
Название параметра	Описание
deviceid	ID температурного датчика
channel	канал на котором настроен датчик
temperature	температура
battery	заряд батареи

3. Отправляем данные в БД сайта.
4. Отображаем данные на сайте.

Исходные компоненты

1. Погодная станция с температурным датчиком (433Mhz) - http://i-on.ru/video/oregon-scientific-bar339p
2. Радио приёмник-передатчик (RF-kit) - http://www.seeedstudio.com/depot/433mhz-rf-link-kit-p-127.html
3. Arduino Uno
4. Arduino Ethernet Shield - http://www.seeedstudio.com/depot/arduino-eth-shield-rev3-with-poe-module-p-1070.html?cPath=19_17
5. BreadBoard - http://www.seeedstudio.com/depot/basic-bread-board-16555-cm-p-4.html?cPath=44_45

Подключение



Скетч

```
// Oregon V2 decoder modfied     - Olivier Lebrun
// Oregon V2 decoder added       - Dominique Pierre
// New code to decode OOK signals from weather sensors, etc.
// 2010-04-11 <jcw@equi4.com> http://opensource.org/licenses/mit-license.php
// $Id: ookDecoder.pde 5331 2010-04-17 10:45:17Z jcw $

#include <SPI.h>
#include <util.h>
#include <Ethernet.h>

// Ethernet options
byte mac[] = {  0x90, 0xA2, 0x0D, 0x0D, 0xA2, 0xCC };          // Arduino Mac
IPAddress ip(192,168,1,6);                                     // Arduino IP
IPAddress gateway(192,168,1,1);                                // WAN GateWay
IPAddress subnet(255,255,255,0);                               // Subnet

char pageName[]   = "/handlers/SetSensorTemperature.ashx";     // Function to fill temperature
data
char serverName[] = "www.silvergate.ru";                       // Server
int serverPort    = 80;                                        // Port
// insure params is big enough to hold your variables
char params[255];

EthernetClient client;
boolean lastConnected = false;                                 // state of the connection last
time through the main loop

class DecodeOOK
{
   protected:
      byte total_bits, bits, flip, state, pos, data[25];

      virtual char decode(word width) = 0;

   public:

      enum { UNKNOWN, T0, T1, T2, T3, OK, DONE };
```

```
DecodeOOK () { resetDecoder(); }

bool nextPulse(word width)
{
   if (state != DONE)
      switch (decode(width))
      {
         case -1: resetDecoder(); break;
         case 1:  done(); break;
      }
   return isDone();
}

bool isDone () const { return state == DONE; }

const byte* getData (byte& count) const
{
   count = pos;
   return data;
}

void resetDecoder()
{
   total_bits = bits = pos = flip = 0;
   state = UNKNOWN;
}

// add one bit to the packet data buffer
virtual void gotBit (char value)
{
   total_bits++;
   byte *ptr = data + pos;
   *ptr = (*ptr >> 1) | (value << 7);

   if (++bits >= 8)
   {
      bits = 0;
      if (++pos >= sizeof data)
      {
         resetDecoder();
         return;
      }
   }
   state = OK;
}

// store a bit using Manchester encoding
void manchester(char value)
{
   flip ^= value; // manchester code, long pulse flips the bit
   gotBit(flip);
}

// move bits to the front so that all the bits are aligned to the end
void alignTail(byte max =0)
{
   // align bits
   if (bits != 0)
   {
      data[pos] >>= 8 - bits;
      for (byte i = 0; i < pos; ++i)
         data[i] = (data[i] >> bits) | (data[i+1] << (8 - bits));
      bits = 0;
   }

   // optionally shift bytes down if there are too many of 'em
   if (max > 0 && pos > max)
   {
      byte n = pos - max;
```

```
            pos = max;
            for (byte i = 0; i < pos; ++i)
               data[i] = data[i+n];
         }
      }

      void reverseBits()
      {
         for (byte i = 0; i < pos; ++i)
         {
            byte b = data[i];
            for (byte j = 0; j < 8; ++j)
            {
               data[i] = (data[i] << 1) | (b & 1);
               b >>= 1;
            }
         }
      }

      void reverseNibbles()
      {
         for (byte i = 0; i < pos; ++i)
            data[i] = (data[i] << 4) | (data[i] >> 4);
      }

      void done()
      {
         while (bits)
            gotBit(0); // padding

         state = DONE;
      }
};

class OregonDecoderV2 : public DecodeOOK
{
   public:

      OregonDecoderV2() {}

      // add one bit to the packet data buffer
      virtual void gotBit (char value)
      {
         if(!(total_bits & 0x01))
         {
            data[pos] = (data[pos] >> 1) | (value ? 0x80 : 00);
         }

         total_bits++;
         pos = total_bits >> 4;

         if (pos >= sizeof data)
         {
            if (Serial.available())
               Serial.println("sizeof data");

            resetDecoder();
            return;
         }
         state = OK;
      }

      virtual char decode(word width)
      {
         if (200 <= width && width < 1200)
         {
            byte w = width >= 700;

            switch (state)
```

```
            {
               case UNKNOWN:
                  if (w != 0)
                  {
                     // Long pulse
                     ++flip;
                  }
                  else if (w == 0 && 24 <= flip)
                  {
                     // Short pulse, start bit
                     flip = 0;
                     state = T0;
                  }
                  else
                  {
                     // Reset decoder
                     return -1;
                  }
                  break;
               case OK:
                  if (w == 0)
                  {
                     // Short pulse
                     state = T0;
                  }
                  else
                  {
                     // Long pulse
                     manchester(1);
                  }
                  break;
               case T0:
                  if (w == 0)
                  {
                     // Second short pulse
                     manchester(0);
                  }
                  else
                  {
                     // Reset decoder
                     return -1;
                  }
               break;
            }
         }
         else if (width >= 2500  && pos >= 8)
         {
            return 1;
         }
         else
         {
            return -1;
         }
         return 0;
      }
};

OregonDecoderV2 orscV2;

volatile word pulse;

void ext_int_1(void)
{
   static word last;
   // determine the pulse length in microseconds, for either polarity
   pulse = micros() - last;
   last += pulse;
}
```

```
float temperature(const byte* data)
{
   int sign   = (data[6]&0x8) ? -1 : 1;
   float temp = ((data[5]&0xF0) >> 4)*10 + (data[5]&0xF) + (float)(((data[4]&0xF0) >> 4) / 10.0);
   return sign * temp;
}

byte humidity(const byte* data)
{
   return (data[7]&0xF) * 10 + ((data[6]&0xF0) >> 4);
}

// Ne retourne qu'un apercu de l'etat de la baterie : 10 = faible
byte battery(const byte* data)
{
   return (data[4] & 0x4) ? 10 : 90;
}

byte channel(const byte* data)
{
   byte channel;
   switch (data[2])
   {
      case 0x10:
         channel = 1;
         break;
      case 0x20:
         channel = 2;
         break;
      case 0x40:
         channel = 3;
         break;
   }

   return channel;
}

String reportSerial(const char* s, class DecodeOOK& decoder)
{
   byte pos;
   const byte* data = decoder.getData(pos);

   if (Serial.available())
   {
      Serial.print(s);
      Serial.print(' ');
   }

   String vTemperatureData = "";

   for (byte i = 0; i < pos; ++i)
   {
      if (Serial.available())
      {
         Serial.print(data[i] >> 4, HEX);
         Serial.print(data[i] & 0x0F, HEX);
      }
   }

   // Outside/Water Temp : THN132N,...
   if(data[0] == 0xEA && data[1] == 0x4C)
   {
      float tempC = temperature(data);

      if (Serial.available())
      {
         Serial.print("[THN132N,...] Id:");
         Serial.print(data[3], HEX);
         Serial.print(", Channel:");
```

```
         Serial.print(channel(data));
         Serial.print(", temp:");
         Serial.print(tempC);
         Serial.print(", bat:");
         Serial.print(battery(data));
         Serial.println();
      }

      char temp[5];
      dtostrf(tempC,5,1,temp);
      //char * vTempData;
      //dtostrf(temperature(data), 5, 2, vTempData);
      sprintf(params,"deviceid=%s&channel=%i&temperature=%s&battery=
%i","EA4C1083322340A3",channel(data),temp,battery(data));
      vTemperatureData = params;

      if (Serial.available())
         Serial.println("vTemperatureData: " + vTemperatureData);

   }
   else if(data[0] == 0x1A && data[1] == 0x2D)
   {
      // Inside Temp-Hygro : THGR228N,...
      if (Serial.available())
      {
         Serial.print("[THGR228N,...] Id:");
         Serial.print(data[3], HEX);
         Serial.print(" ,Channel:");
         Serial.print(channel(data));
         Serial.print(" ,temp:");
         Serial.print(temperature(data));
         Serial.print(" ,hum:");
         Serial.print(humidity(data));
         Serial.print(" ,bat:");
         Serial.print(battery(data));
         Serial.println();
      }
   }

   decoder.resetDecoder();
   return vTemperatureData;
}

byte sendData
  (char*  domainBuffer,
   int    thisPort,
   char*  page,
   String thisData)
{
   if (client.connected()) client.stop();

   int inChar;
   char outBuf[64];

   if (Serial.available())
   {
      Serial.print(F("connecting..."));
      Serial.println(domainBuffer);
   }

   if (client.connect(serverName,serverPort))
   {
      if (Serial.available())
         Serial.println(F("connected"));

      // send the header
      //String data = "deviceid=EA4C108122234030&channel=1&temperature=22&battery=90";
      String data = thisData;
      if (Serial.available())
```

```
         Serial.println("Sending to Server: ");

      client.println("POST /handlers/SetSensorTemperature.ashx HTTP/1.1");

      if (Serial.available())
         Serial.print("POST /handlers/SetSensorTemperature.ashx HTTP/1.1");

      client.println("Host: www.silvergate.ru");
      client.println("Content-Type: application/x-www-form-urlencoded");
      client.println("Connection: close");
      client.println("User-Agent: Arduino/1.0");
      client.print("Content-Length: ");
      client.println(data.length());
      client.println();
      client.print(data);
      client.println();

      if (Serial.available())
         Serial.println("data uploaded");
   }
   else
   {
      if (Serial.available())
         Serial.println(F("connection failed"));
   }
}

void setup ()
{
   Serial.begin(115200);
   delay(500);

   // start the Ethernet connection:
   if (Ethernet.begin(mac) == 0)
   {
      if (Serial.available())
         Serial.println("Failed to configure Ethernet using DHCP");
      // no point in carrying on, so do nothing forevermore:
      // try to congifure using IP address instead of DHCP:
      Ethernet.begin(mac, ip);
   }
   // give the Ethernet shield a second to initialize:
   delay(2000);

   attachInterrupt(1, ext_int_1, CHANGE);
}

void PostTemperatureDataToServer(String vData)
{
   sendData(serverName,serverPort,pageName,vData);

   // if there's no net connection, but there was one last time
   // through the loop, then stop the client:
   if (!client.connected() && lastConnected)
   {
      if (Serial.available())
      {
         Serial.println();
         Serial.println("disconnecting.");
      }

      client.stop();
   }

   lastConnected = client.connected();
}

void loop()
{
```

```
   static int i = 0;
   cli();
   word p = pulse;

   pulse = 0;
   sei();

   if (p != 0)
   {
      if (orscV2.nextPulse(p))
      {
         String vData = reportSerial("OSV2", orscV2);
         if (vData != "")
         {
            PostTemperatureDataToServer(vData);
         }
      }
   }
}
```

Отображение на сайте (Не MajorDoMo)

Gadget.html

```
<!DOCTYPE html>
<html>
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
      <script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
      <script type="text/javascript" src="main.js"></script>
      <style>
         body
         {
            width  : 240px;
            height : 150px;
            background:#333;
            color: #333;
         }
      </style>
      <script type="text/javascript">
         var t;
```

```
         var timer_is_on = 0;
         function initialize()
         {
            if (!timer_is_on)
            {
               timer_is_on = 1;
               GetData();
            }
         }
      </script>
   </head>
   <body onload="initialize()">
      <div style="margin-top:1em;" class="notify_block_div">
         <div class="notify_block" style="border-bottom: 1px solid #D8D8D8;border-radius: 8px 8px 8px 8px;position: relative;z-index: 10;">
            <div class="notify_block_title" style="background-color: #F28000;border-radius: 8px 8px 0 0;color: #FFFFFF;font-weight: bold;padding: 5px;text-align:center;">Температура на улице</div>
            <div class="notify_block_content" style="background-color: #fff;border-radius: 0 0 8px 8px;margin:0;padding: 0.5em;">
               <div style="text-align:center;font-size:140%;"><span id="WeatherTemp"></span> &deg;C</div>
               <div style="font-size:70%">Дата снятия показания: <span id="WeatherDate"></span><br />Заряд батареи: <span id="Battery">%</div>
            </div>
         </div>
      </div>
   </body>
</html>
```

main.js

```
function GetData()
{
   now = new Date();
   $('#CurrentDate').html(now);
   var WeatherUrl = "/Handlers/GetLastTempSensorDataByID.ashx";
   $.ajax({
           type: 'GET',
           url: WeatherUrl,
           success: function (data)
           {
               processData(data);
           }
   });

   function processData(data)
```

```
    {
        var sensor = data.split("|");
        $('#WeatherDate').html(sensor[0]);
        $('#WeatherTemp').html(sensor[1]);
        $('#Battery').html(sensor[2]);
    }

    t = setTimeout("GetData()", 100000);
};
```