Henrietta BAN[1]

Opiekun naukowy: Vitaly GERASIMOV[2], Alexander MOLNAR[3]

SYSTEM MONITORINGU IoT Z ZASTOSOWANIEM MODUŁU WEMOS D1

Streszczenie: W niniejszej pracy przedstawiono ideę zastosowania modułu VEMOS D1 jako urządzenie IoT do prowadzenia monitoring środowiska. Zaproponowany system składa się z kilku części. Jadna z nich zbiera informację z czujników stanu środowiska oraz zapisuje dane w module SD. Takie rozwiązanie umożliwia przesyłanie wiadomosci o statusie czujników oraz innych parametrach systemu za pomocą protokołu SMTP, z możliwością przechowywania danych lokalnie lub innych rodzajach pamięci.

Słowa kluczowe: IoT, czujniki, protokół SMTP

DEVELOPMENT OF AN IoT MONITORING SYSTEM BASED ON THE WEMOS D1 MODULE

Summary: The results of using the VEMOS D1 module as an IoT device for monitoring the environment have been presented in this article. The proposed system consists of several parts, one of which collects information from environmental sensors and records data on SD module. The development provides for sending a message about the status of sensors and other aspects of the system via the SMTP protocol on local and remove storage.

Keywords: IoT system, sensors, SMTP

[1]PhD studet, Department of Semiconductor Physics, Uzhhorod National University, Ukraine,
[2]Mukachevo State University, Faculty of Economics, Management and Engineering, vitgerv@gmail.com
[3]Department of Semiconductor Physics, Uzhhorod National University, Ukraine

1. Introduction

The topic of IoT technologies is receiving a lot of attention today in fact they are the technologies of the future information society. Modular development tools for various devices such as Arduino, TI, Raspberry Pi and the like are very suitable for creating various types of devices from the field of IoT. The advantages of such modular systems are affordability in terms of price, multifunctionality, the ability to use ready-made libraries for working with external sensors(others modules), etc. In this respect, ESP8266 modules are out of competition ofthe above parameters [1]. The authors have set the task of creating a monitoring system for environmental control using high-quality and reliable sensors (temperature, humidity, CO_2) as well as available Wi-Fi modules for transferring data to external storage (cloud server) with simultaneous synchronous storage of information on the local SD module. Also, the condition of the project did not include the use of paid cloud services, which significantly attracted the interest of part of future users to this monitoring system. Despite the seeming simplicity of the project's implementation, the authors have faced a number of problems that were solved in one way or another (Fig 1).

a) *b)* *c)*

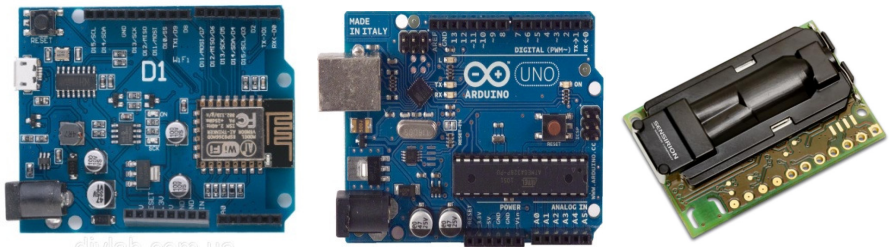

Figure 1. a) Module Wemos D1 for Wi-Fi connection [6]; b) Arduino UNO [7]; c) SCD30 Sensor Module for control CO2, humidity and temperature [8]

2. Implementation

First of all, WEMOS D1 was chosen as a Wi-FI module it is the main one in this design. This module is a relatively new, at the same time cheap and effective development tool based on Wi-Fi communication. The advantage is that it uses the widely popular ESP8266 circuit [2, 3], as well as a large number of ports for in/out data. This one is the main advantage in relation to cheaper modules, for example ESP8266-01,03 etc. In addition, we are pleased with the good quality of the module, power supply via standard 5(12) volts, ease of programming via USB. Since the data transmission was planned using wireless technologies, where the probability of receiving-transmitting failure is especially high, a regular module on ARDUINO UNO [4] with an integrated SCD30 air Sensor was provided to collect information from the sensors. With the ARDUINO module, information based on sensor data is transmitted to WEMOS and from this point on to a remote cloud storage. Such an organization allows to significantly increase the stability of work, since the

implementation of the procedure for collecting data from sensors, recording on SD card in one module and transferring to Wi-Fi can lead to sagging of the device on the whole due to complex software execution.

3. Results and discussion

For date, these two independent parts of the project work autonomously and stably, without any comments. We would like to note that the remote storage actually failed to be implemented on one of the popular services, so it was not possible to programmatically implement authentication through the Arduino environment. We think this situation is connected with the security of access protocols for such services, so they are not published open sources. However, it was possible to send and store information through the SMTP protocol, which is actually access to e-mail. On the Internet resource, an example of sending a message through a GMAIL mailbox forESP8266 module was taken [5]. Changes have been made in the program code corresponding to the parameters of the local network: access passwords and, of course, the mailbox address on Google. In addition to this service, it is possible to connect to a number of postal services, for example *Yandex*. The only moments in the implementation of this Google connection was that the system informed user for a dubious connection and warned about the possibility of unauthorized access to mail. In the settings of Google mail, we managed to disable this alarm and continue to use the mail. Since, under the terms of the project, data are sent relatively rarely - once a day, this does not lead to an overflow of notifications in the mail box, and moreover, it makes it possible for the module to send an alarm message, for example, when there is no power on the modules or the critical state of the sensor indicator or its being out of service.

The program code for sending messages to the SMTP protocol is presented below.

```
#include <ESP8266WiFi.h>
#include "Mail.h"
const char* const staSSID = "netis_3";
const char* const staPass = "222222222";
const char* const smtpHost[] = {"smtp.gmail.com" };
const uint16_t smtpPort = 465;
const char* const smtpUser[] = { "vitgerv@gmail.com" };
const char* const smtpPass[] = { "Vov1212345" };
const char* const mailTo = "vitge@mail.ru";
const char* const mailSubject = "test to mail";
void setup() {
  Serial.begin(115200);
  Serial.println();
  Serial.print(F("Connecting to \""));
  Serial.print(staSSID);
  Serial.print('"');
  WiFi.begin(staSSID, staPass);
  while (WiFi.status() != WL_CONNECTED) {
```

```
    delay(500);
    Serial.print('.');
  }
  Serial.print(' ');
  Serial.println(WiFi.localIP());
  delay(10000);
  for (int8_t i = 0; i < 3; ++i) {
    Serial.println();
    if    (sendMail(smtpHost[i],    smtpPort,    smtpUser[i],
smtpPass[i], mailTo, mailSubject, F("p1\r\np2\r\np3"))) {
      Serial.print(F("Mail sended through "));
      Serial.println(smtpHost[i]);
    } else {
      Serial.print(F("Error sending mail through "));
      Serial.print(smtpHost[i]);
      Serial.println('!');
    }
  }
  WiFi.disconnect();
}
void loop() {
}
```

Fragment of code for collecting and recording data from the sensor:

```
// lib for  RTC
#include <Wire.h>
#include <Time.h>
#include <DS1307RTC.h>
#include "SparkFun_SCD30_Arduino_Library.h"
SCD30 airSensor;
// lib for SD SD
#include <SD.h>
const int chipSelect = 10;
File myFile;
String sfilename;
char filename[20];
tmElements_t tm;
//string for record
String record="";
void setup()
{
Serial.begin(9600);
Serial.print("Initializing SD card...");
  // see if the card is present and can be initialized:
  if (!SD.begin(chipSelect))
  {
```

```
    Serial.println("Card failed, or not present");
    // don't do anything more:
    while (1);
  }
Serial.println("card initialized.");
  airSensor.begin(); //This will cause readings to occur every
two seconds
  airSensor.setMeasurementInterval(4);  //Change  number  of
seconds between measurements: 2 to 1800 (30 minutes)
  //start level of preasure
  airSensor.setAltitudeCompensation(164); //Set  altitude  of
the sensor in m
  // CO is 24.65inHg or 834.74mBar
  airSensor.setAmbientPressure(835);    //Current    ambient
pressure in mBar: 700 to 1200
}
void loop()
......
```

4. Conclusions

This development of an IoT monitoring system based on the WEMOS D1 module allows you to receive data from sensors and send it to a remote and local storage. The one was used as Gmail mail service.
Receiving data from sensors is implemented on a relatively slow module Arduino Uno.
This approach reduces the likelihood sagging of a system by separating the sending and collecting functions in one device.

REFERENCES

1. MAGESH JAYAKUMAR: The internet of things with esp8266 Hands on approach: Get started with Arduino IDE and ESP8266, CreateSpace Independent Publishing Platform, 2017.
2. SCHWARTZ M.: Internet of Things with ESP8266, Packt Publishing, 2016.
3. CATALIN BATRINU: ESP8266 Home Automation Projects: Leverage the power of this tiny WiFi chip to build exciting smart home projects, Packt Publishing, 2017.
4. NICHOLAS S.: Arduino Programming: A Comprehensive Beginner's Guide to Learn the Realms of Arduino from A-Z, Independently published, 2020.
5. Web page: *https://randomnerdtutorials.com/esp32-send-email-smtp-server-arduino-ide/*, 22.10.2020.

6. Web page: *https://www.arduiner.com/prodotto/wemos-d1-wifi-esp8266-80-160mhz-placa-de-desarrollo-flash-de-4mb-compatible-arduino-uno/*, 22.10.2020.
7. Web page: *https://www.arduino.cc/*, 22.10.2020.
8. Web page: *https://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/9.5_CO2/Sensirion_CO2_Sensors_SCD30_Datasheet.pdf*, 22.10.2020.