**HOME AUTOMATION USING ARDUINO WIFI**

We frequently forget to turn off lights, televisions, air conditioners, fans, and other appliances in our daily lives, resulting in unnecessary waste of electricity and, of course, money, both of which are extremely valuable to us.

All I need is a modem that is connected to a wireless router or a wireless gateway that combines a modem and a wireless router into one device. I can access my appliances from anywhere thanks to connection.

The automation system will be able to be operated remotely via a Pocket PC with a Windows Mobile-based application, as well as from a central host PC and the Internet.

The system will also detect an accidental gas leak or a low water level and send an SMS notification to the user.

**Technical requirements**

Microcontroller ATmega328P

Operating Voltage 5V

Input Voltage (recommended) 7-12V

Input Voltage (limit) 6-20V

Digital I/O Pins 14 (of which 6 provide PWM output)

PWM Digital I/O Pins 6

Analog Input Pins 6

DC Current per I/O Pin 20 mA

DC Current for 3.3V Pin 50 mA

Flash Memory 32 KB (ATmega328P)

of which 0.5 KB used by bootloader

SRAM 2 KB (ATmega328P)

EEPROM 1 KB (ATmega328P)

Clock Speed 16 MHz

Length 68.6 mm

Width 53.4 mm

Weight 25 g.

#include "SoftwareSerial.h"

#define DEBUG true

SoftwareSerial esp8266(2,3); // make RX Arduino line is pin 2, make TX Arduino

line is pin 3.

// This means that you need to connect the TX

line from the esp to the Arduino's pin 2

23

// and the RX line from the esp to the Arduino's pin

3

void setup()

{

Serial.begin(9600);

esp8266.begin(9600); // your esp's baud rate might be different

pinMode(4,OUTPUT);

digitalWrite(4,LOW);

pinMode(5,OUTPUT);

digitalWrite(5,LOW);

pinMode(6,OUTPUT);

digitalWrite(6,LOW);

pinMode(7,OUTPUT);

digitalWrite(7,LOW);

24

pinMode(13,OUTPUT);

digitalWrite(13,LOW);

sendData("AT+RST\r\n",2000,DEBUG); // reset module

sendData("AT+CWMODE=2\r\n",1000,DEBUG); // configure as access point

sendData("AT+CIFSR\r\n",1000,DEBUG); // get ip address

sendData("AT+CIPMUX=1\r\n",1000,DEBUG); // configure for multiple

connections

sendData("AT+CIPSERVER=1,80\r\n",1000,DEBUG); // turn on server on port 80

}

void loop()

{

if(esp8266.available()) // check if the esp is sending a message

{

if(esp8266.find("+IPD,"))

{

delay(1000); // wait for the serial buffer to fill up (read all the

serial data)

// get the connection id so that we can then disconnect

25

int connectionId = esp8266.read()-48; // subtract 48 because the read()

function returns

// the ASCII decimal value and 0

(the first decimal number) starts at 48

esp8266.find("pin="); // advance cursor to "pin="

int pinNumber = (esp8266.read()-48)\*10; // get first number i.e. if the

pin 13 then the 1st number is 1, then multiply to get 10

pinNumber += (esp8266.read()-48); // get second number, i.e. if the pin

number is 13 then the 2nd number is 3, then add to the first number

switch (pinNumber)

{

case 1://switch 1 on

digitalWrite(4,HIGH);

break;

case 2://switch 2 on

digitalWrite(5,HIGH);

26

break;

case 3://switch 3 on

digitalWrite(6,HIGH);

break;

case 4://switch 4 on

digitalWrite(7,HIGH);

break;

case 5://led on

digitalWrite(13,HIGH);

break;

case 6://switch 1 off

digitalWrite(4,LOW);

break;

case 7://switch 2 off

27

digitalWrite(5,LOW);

break;

case 8://switch 3 off

digitalWrite(6,LOW);

break;

case 9://switch 4 off

digitalWrite(7,LOW);

break;

case 10://led off

digitalWrite(13,LOW);

break;

default:

break;

}

//digitalWrite(pinNumber, !digitalRead(pinNumber)); // toggle pin

28

// make close command

String closeCommand = "AT+CIPCLOSE=";

closeCommand+=connectionId; // append connection id

closeCommand+="\r\n";

sendData(closeCommand,1000,DEBUG); // close connection

}

}

}

/\*

\* Name: sendData

\* Description: Function used to send data to ESP8266.

\* Params: command - the data/command to send; timeout - the time to wait for a

response; debug - print to Serial window?(true = yes, false = no)

\* Returns: The response from the esp8266 (if there is a reponse)

\*/

String sendData(String command, const int timeout, boolean debug)

{

29

String response = "";

esp8266.print(command); // send the read character to the esp8266

long int time = millis();

while( (time+timeout) > millis())

{

while(esp8266.available())

{

if(debug)

{

Serial.print(response);

return response;

}