**ДОДАТОК 3**

**Код мікроконтролера блоку обробки інформації**

#include <SPI.h>

#include <Ethernet.h>

#include <RF24.h>

#include <SoftwareSerial.h>

// define SPI pins for arduino uno

#define useSoftSPI true

#define CE 5

#define CSN 6

#define SCK 7

#define MOSI 8

#define MISO 9

// define recieve structures

struct SensorData

{

int id;

float value;

};

// init ethernet module

byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};

IPAddress ip(192, 168, 1, 177);

EthernetServer server(80);

// init other modules

RF24 radio(CE, CSN);

SoftwareSerial SIM800(A1, A0);

/\*\*

\* Method for setup arduino

\*/

void setup()

{

// open serial port

Serial.begin(9600);

delay(500);

// setup modules

setupServer();

setupRadio();

}

/\*\*

\* Method for setup radio module

\*/

void setupRadio()

{

// set default props

const int channel = 12;

const uint64\_t pipe = 0xFFFFFFFFFFLL;

// setup radio

radio.begin();

radio.setChannel(channel);

radio.setDataRate(RF24\_250KBPS);

radio.setPALevel(RF24\_PA\_MIN);

radio.openReadingPipe(0, pipe);

radio.startListening();

}

/\*\*

\* Method for looping arduino proccess

\*/

void loop()

{

// define need to send flag

bool needToSend = false;

// handle radio data receiving

SensorData light = {0, 0};

SensorData temperature = {0, 0};

SensorData recievePackage[] = {light, temperature};

if (radio.available())

{

// read received data

radio.read(&recievePackage, sizeof(recievePackage));

// handle light

light = recievePackage[0];

Serial.print("Light: ");

Serial.print(light.value);

Serial.println(" lx");

// handle temperature

temperature = recievePackage[1];

Serial.print("Temperature: ");

Serial.print(temperature.value);

Serial.println(" C");

Serial.println("Radio data was completely received.\n");

// set need to send flag

needToSend = true;

}

if (needToSend)

{

String sensorsDataJSON = converSensorsDataToJSON(recievePackage);

makeRequest(sensorsDataJSON);

}

}

/\*\*

\* Helper for waiting GSM-module response

\*/

void waitResponse()

{

while (SIM800.available())

{

Serial.write(SIM800.read());

}

}

/\*\*

\* Helper for making POST request via GSM-module

\*/

void makeRequest(String data)

{

SIM800.begin(9600);

SIM800.println("AT");

waitResponse();

delay(1000);

SIM800.println("AT+CSTT=\"internet\",\"\",\"\"");

delay(1000);

waitResponse();

SIM800.println("AT+CIICR");

delay(3000);

waitResponse();

SIM800.println("AT+CIFSR");

delay(2000);

waitResponse();

SIM800.println("AT+CIPSPRT=0");

delay(3000);

waitResponse();

SIM800.println("AT+CIPSTART=\"tcp\",\"solar-monitor.herokuapp.com\",\"80\"");

delay(3000);

waitResponse();

SIM800.println("AT+CIPSEND");

delay(3000);

waitResponse();

SIM800.println("POST /sensors/data HTTP/1.1");

delay(100);

waitResponse();

SIM800.println("Content-Type: application/json");

delay(100);

waitResponse();

SIM800.print("Content-Length: ");

SIM800.println(data.length());

delay(100);

waitResponse();

SIM800.println("Connection: close");

delay(100);

waitResponse();

SIM800.println("Host: solar-monitor.herokuapp.com");

delay(100);

waitResponse();

SIM800.println();

delay(100);

waitResponse();

SIM800.println(data);

delay(100);

waitResponse();

SIM800.println();

delay(100);

waitResponse();

SIM800.println((char)26);

delay(10000);

SIM800.println("AT+CIPCLOSE");

delay(200);

waitResponse();

}

/\*\*

\* Helper for converting sensors data to JSON

\*/

String converSensorsDataToJSON(SensorData sensorsData[2])

{

int sensorsDataLength = 2;

String data = "{ \"data\": [";

for (int i = 0; i < sensorsDataLength; i++)

{

SensorData sensorData = sensorsData[i];

data += "{ \"id\": ";

data += sensorData.id;

data += ", \"value\": ";

data += sensorData.value;

data += " }";

if (i != sensorsDataLength - 1)

{

data += ", ";

}

}

data += "] }";

return data;

}