**1 Лабораторна робота №2**

**РОЗРОБКА ГРАНИЧНИХ ПРИСТРОЇВ “ІНТЕРНЕТУ РЕЧЕЙ” В СЕРЕДОВИЩІ МОДЕЛЮВАННЯ PROTEUS**

**Мета роботи:** Ознайомитися із функціональними можливостями середовища віртуального моделювання Proteus. Ознайомитися із віртуальними моделями пристроїв “Інтернету речей”. Навчитися створювати схеми IoT-пристроїв і виконувати їх симуляцію. Отримати практичні навички відлагодження програм, написаних для мікроконтролерів AVR в середовищі Proteus.

**1 КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ**

Розробником середовища віртуального моделювання Proteus є компанія Labcenter Electronics (Великобританія). Сайт розробника - http://www.labcenter.co.uk/. Відмінність цього середовища від аналогічних за призначенням (наприклад, Electronics Workbench, Multisim Ultiboard, MicroCap, Tina) полягає у розвиненій системі симуляції (інтерактивного налагодження як в режимі реального часу, так і покроково) для різних сімейств мікроконтролерів: Intel 8051, AVR і PIC (компанії Microchip), ARM (різних виробників) та ін.

Proteus має в своєму складі велику кількість як бібліотек окремих компонентів, так і периферійних пристроїв: світлодіодних і рідкокристалічних індикаторів, датчиків температури, годинників реального часу (RTC), інтерактивних елементів вводу/виводу (кнопок, перемикачів, віртуальних портів і віртуальних вимірювальних приладів), інтерактивних графічних об’єктів, які не завжди присутні в інших подібних програмах.

Середовище Proteus VSM складається із двох самостійних частин: ISIS і ARES. Основним призначенням підсистеми ARES є робота із проектування, трасування та виготовлення друкованих плат (PCB – Printed Circuit Board). Підсистема ISIS, що має безпосередній зв'язок із ARES, призначена для створення електричних принципових схем, написання та редагування коду програмованих пристроїв, а також моделювання і відлагодження їх роботи.

У спрощеному вигляді роботу в середовищі моделювання Proteus ISIS можна представити наступним алгоритмом:

1. вибір необхідної елементної бази, створення схеми електричної принципової пристрою або системи, які розробляються;
2. налаштування необхідних початкових параметрів компонентів для їх функціонування;
3. розміщення необхідних віртуальних пристроїв (вимірювання та/або візуалізації), налаштування режимів їх роботи;
4. написання та відлагодження програм для мікроконтролерів;
5. симуляція або проведення спеціалізованого аналізу.

Більш детальне ознайомлення із можливостями середовища моделювання Proteus слід починати із ознайомлення із проектами-прикладами. Зокрема, в Proteus VSM реалізовано підтримку наборів інструментів для роботи із платформою Arduino. Це дозволяє розробляти прототипи проектів для Arduino безпосередньо в самому середовищі Proteus, тобто, редактор коду є вбудованим і не вимагає використання, наприклад, Arduino IDE. Така функція проекту є дуже корисною, оскільки проекти можна легко створити для плат Arduino різних версій.

**2 ХІД ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ**

Завдання для лабораторної роботи зображено в таблиці 2.1.

Таблиця 2.1 – Завдання до лабораторної роботи №2

|  |  |  |
| --- | --- | --- |
| № варіанту (бригади) | Конфігурація пристрою (компонент / кількість) | Примітки |
| 4 | BME280/1 | Кожні 10 с ведучий МК видає команду до веденого на вимірювання трьох параметрів.  Результати вимірювання датчиком, приєднаним до веденого МК, передаються ведучому.  Ведучий здійснює розрахунок середнього геометричного кожного із параметрів окремо і раз на хвилину виводить результати у віртуальний термінал |

Код програми наведено в лістингу 2.2

Контролер ARD1 – ведучий. Контролер ARD2 – ведений.

Лістинг 2.1 – Код програми ведучого контролера

|  |
| --- |
| #include<SPI.h>  void setup (void){  Serial.begin(115200);      SPI.begin();  SPI.setClockDivider(SPI\_CLOCK\_DIV8);  digitalWrite(SS,HIGH);  delay(100);  }  void loop(void){      double mTemperature=1, mAltitude = 1, mPressure = 1;  for(int i=0; i < 6; i++){  digitalWrite(SS, LOW);  SPI.transfer(1);  delay(30);  mTemperature = mTemperature \* (int)map(SPI.transfer(2), 0, 200, -100, 100) / 100.0;  delay(30);  mPressure = mPressure \* (int)map(SPI.transfer(3), 0, 254, 633, 887) / 100.0;  delay(30);  mAltitude = mAltitude \* (int)SPI.transfer(3) / 100.0;  delay(9910);  }    Serial.print("Temperature ");  Serial.println(pow(mTemperature, 1.0/6)\*100);  Serial.print("Pressure in mm Hg ");  Serial.println(pow(mPressure, 1.0/6)\*100);  Serial.print("Altitude ");  Serial.println(pow(mAltitude, 1.0/6)\*100);  } |

Лістинг 2.2 - Код програми веденого контролера

|  |
| --- |
| #include<SPI.h>  #include <Wire.h>  #include <Adafruit\_BMP280.h>  volatile boolean received;  volatile byte Slavereceived;  Adafruit\_BMP280 bmp;  void setup(){  Serial.begin(115200);    pinMode(MISO,OUTPUT);  SPCR |= \_BV(SPE);  received = false;  SPI.attachInterrupt();  if(!bmp.begin(0x77)) {  Serial.println(F("Could not find a valid BMP280 sensor,check wiring!"));  }  }  ISR (SPI\_STC\_vect) {  Slavereceived = SPDR;  received = true;  }  void loop(){  if(received){  if(Slavereceived==1){  SPDR = (byte)map((int)bmp.readTemperature(), -100, 100, 0, 200);  }  if(Slavereceived==2){  SPDR = (byte)map((int)(bmp.readPressure() \* 0.00750063755419211), 633, 887, 0, 254);  }  if(Slavereceived==3){  SPDR = (byte)bmp.readAltitude(1013.25);  }  received = false;  }  delay(10);  } |

На рисунку 2.1 зображено схему пристрою.

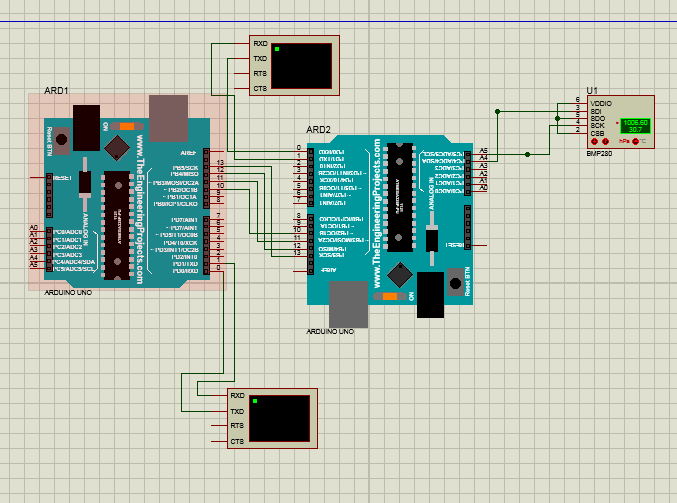


Рисунок 2.1 – Схема пристрою

На рисунку 2.2 та 2.3 зображено роботу пристрою.



Рисунок 2.2 – Початок роботи пристрою



Рисунок 2.3 – Отримання даних з веденого до ведучого та виведення середнього геометричного значення

Особливості функціонування Proteus VSM:

* можливість створення та симуляції складних електронних схем з різними типами компонентів, включаючи мікроконтролери, сенсори та інші пристрої;
* наявність бібліотек компонентів, які містять багато моделей компонентів, що відповідають різним стандартам;
* зручний інтерфейс користувача, який дозволяє легко налаштовувати та керувати симуляцією електронної схеми;
* можливість відстежування та аналізу поведінки електронної схеми під час симуляції, включаючи відлагодження коду мікроконтролера та аналіз результатів симуляції;
* наявність функцій інспектора компонентів, що дозволяє користувачеві перевірити параметри та стан компонентів в режимі реального часу.

**ВИСНОВКИ**

Ознайомився із функціональними можливостями середовища віртуального моделювання Proteus. Ознайомився із віртуальними моделями пристроїв “Інтернету речей”. Навчився створювати схеми IoT-пристроїв і виконувати їх симуляцію. Отримав практичні навички відлагодження програм, написаних для мікроконтролерів AVR в середовищі Proteus.