МІНІСТЕРСТВО ВНУТРІШНІХ СПРАВ УКРАЇНИ
ЛЬВІВСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ ВНУТРІШНІХ СПРАВ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ УПРАВЛІННЯ, ПСИХОЛОГІЇ ТА БЕЗПЕКИ

Кафедра інформаційних технологій

РОЗРОБЛЕННЯ НА ПЛАТФОРМІ ARDUINO СИСТЕМИ МОНІТОРИНГУ ФІЗИКО-ГЕОГРАФІЧНИХ ПАРАМЕТРІВ В ХОДІ ПІДГОТОВКИ ДО СТРІЛЬБИ

Кваліфікаційна робота
здобувача вищої освіти
4 курсу денної форми навчання
Максима ПОЦЮРКА

Науковий керівник:
доцент, кандидат технічних наук
Ігор ФАРМАГА

Рецензент:

_____
вчене звання, науковий ступінь

_____
(Ім'я ПРІЗВИЩЕ рецензента)

Кваліфікаційна робота допущена до захисту
«___» _____ 2025 р., протокол № ____

Завідувач кафедри інформаційних технологій
_____Ігор ФЕДЧАК_
(підпис)

Львів
2025

СПИСОК УМОВНИХ СКОРОЧЕНЬ

ПК – персональний комп’ютер;

МК – мікроконтролер;

АЗ – апаратне забезпечення;

ПЗ – програмне забезпечення;

МП – мікроконтролерний пристрій;

САПР – система автоматизованого проектування;

AVR – сімейство 8 - розрядних мікроконтролерів фірми Atmel;

Arduino Mega 2560 – апаратно-програмна платформа на МК ATmega2560;

Arduino IDE – інтегроване середовище розробки ПЗ для плат Arduino;

FLASH – різновид напівпровідникової технології пам’яті, яка електричним методом перепрограмовується (EEPROM);

EEPROM – постійна пам’ять, яка стирається і перепрограмовується електричним методом;

ПЗП - постійний запам’ятовуючий пристрій, використовується для зберігання масиву незмінних даних;

ОЗП (RAM) – оперативна пам’ять або оперативний запам’ятовуючий пристрій;

РКД (LCD) – рідкокристалічний дисплей;

1-Wire – двонапрямлена шина зв’язку для пристроїв з низькою швидкістю передачі даних (зазвичай 15,4 Кбіт/с, макс. 125 Кбіт/с). Розроблена корпорацією Dallas Semiconductor.

SPI – послідовний синхронний стандарт передачі даних в режимі повного дуплексу, призначений для забезпечення простого і недорогого високошвидкісного сполучення мікроконтролерів і периферії;

I^2C – послідовна шина даних для зв’язку інтегральних схем, що використовує дві двонаправлені лінії зв’язку (SDA і SCL).

UART – універсальний асинхронний приймач/передавач.

АНОТАЦІЯ

Поцюрко М., Фармага І. (керівник). Розроблення на платформі Arduino системи моніторингу фізико-географічних параметрів в ході підготовки до стрільби. Бакалаврська кваліфікаційна робота. – Львівський державний університет внутрішніх справ, Львів, 2025.

У дипломній роботі розроблено апаратне та програмне забезпечення мікроконтролерного пристрою для моніторингу основних параметрів погоди. Пристрій моніторить такі метеопараметри як: температура, відносна вологість повітря і атмосферний тиск, та виводить виміряні значення метеорологічних величин на символьний рідкокристалічний дисплей. Пристрій має меню налаштування одиниць виводу на РК-дисплей температури і тиску. Меню виводиться на РК - дисплей при натисненні кнопки MENU_BTN. За допомогою кнопок S_PLUS_BTN і S_MINUS_BTN виконується перехід між пунктами меню. Температура може відображатися в ºC, ºF або ºK, а атмосферний тиск в мбар, мм. рт. ст., дюймах рт. ст. або гПа.

Апаратне забезпечення пристрою для моніторингу основних параметрів поголи розроблено на МК ATmega328P-PU платформи Arduino Uno R3 та цифровому давачі тиску BMP180, температури LM35, вологості DHT11 та символьному рідкокристалічному дисплеї на базі контролера HD44780.

Розроблено електричну принципову схему та модель мікроконтролерного пристрою для моніторингу основних параметрів погоди в системі автоматизованого проектування Proteus. Розроблено алгоритм роботи та програмне забезпечення пристрою в середовищі Arduino IDE з використанням мови С. Змодельовано роботу пристрою в емуляторі Proteus ISIS.

Зібрано прототип мікроконтролерного пристрою для моніторингу основних параметрів погоди на безпаєчній макетній платі та протестовано його роботу.

Ключові слова: апаратне забезпечення, мікроконтролерний пристрій моніторингу основних параметрів погоди, метеодані, цифровий давач тиску і температури BMP180, цифровий давач вологості і температури DHT11, аналоговий давач температури LM35, рідкокристалічний дисплей HD44780, мікроконтролер ATmega328P-PU, комплекс програм для автоматизованого проектування електронних пристроїв Proteus, вбудоване програмне забезпечення, мова програмування C, середовище розробки Arduino IDE

.

ABSTRACT

Potsyurko M., Farmaha I. (supervisor). Development of a system for monitoring physical and geographical parameters during preparation for shooting on the Arduino platform. Bachelor's thesis. – Lviv State University of Internal Affairs, Lviv, 2025.

In the diploma thesis, hardware and software of the microcontroller device for monitoring the main weather parameters have been developed. The device monitors such meteorological parameters of the environment as: temperature, relative humidity and atmospheric pressure, and displays the measured values of meteorological values on a symbolic liquid crystal display. The device has a menu for setting the output units of temperature and pressure on the LCD. The menu is displayed on the LCD by pressing the MENU_BTN button. For navigation between the menu items the S_PLUS_BTN and S_MINUS_BTN buttons are used. The temperature can be displayed in ºC, ºF or ºK, and atmospheric pressure in mbar, mmHg, inHg or hPa.

The hardware of the device for monitoring the main weather parameters is developed on the ATmega328P-PU microconroller and using the digital pressure sensor BMP180, temperature sensor LM35, humidity sensor DHT11 and the alpha-numeric LCD based on the HD44780 controller.

The electronic circuit and the model of the microcontroller device for monitoring the main weather parameters have been created using Proteus Design Suite. The operation algorithm and software of the device in Arduino IDE using the language C have been developed. The operation of the microcontroller device for monitoring the main weather parameters in Proteus ISIS has been simulated.

The prototype of the microcontroller device for monitoring the main weather parameters has been built on a solderless breadboard and its operation has been tested.

Key words: hardware, microcontroller device for monitoring the main weather parameters, meteorological data, digital pressure and temperature sensor

BMP180, digital relative humidity and temperature sensor DHT11, analog temperature sensor LM35, alpha-numeric LCD HD44780, ATmega328P-PU microcontroller, software for electronic design automation Proteus, embedded software, embedded C, Arduino IDE programming environment for Arduino platform.

ЗМІСТ

ВСТУП

Погода відображає поточний стан атмосфери в конкретному місці в певний момент. Для повного опису атмосферних умов необхідно кількісно визначити її **елементи**. До ключових погодних параметрів, або **метеопараметрів**, належать температура, відносна вологість повітря, атмосферний тиск, швидкість і напрямок вітру, а також хмарність та опади. У метеорології дані, зібрані за допомогою різних приладів, слугують основою для прогнозування погодних явищ. З давніх-давен людина прагнула зрозуміти причини різноманітних погодних умов, навчитися їх виявляти і, можливо, навіть передбачати погоду на найближчий час. Це спонукало до розробки різноманітних методів та засобів для вимірювання та моніторингу погодних елементів.

Сучасний рівень розвитку електроніки дозволяє створювати доступні **вимірювальні системи на базі мікроконтролерів** та різноманітних датчиків. Отже, розробка мікроконтролерних пристроїв для моніторингу метеопараметрів є вельми актуальною [10].

Для апаратного забезпечення мікроконтролерного пристрою, призначеного для моніторингу основних параметрів погоди, запропоновано використовувати **платформу Arduino Uno R3** з мікроконтролером ATmega328P-PU.

Сьогодні **мікроконтролери** широко застосовуються в багатьох сучасних пристроях та обладнанні. Їхня найважливіша особливість полягає в тому, що вони значно спрощують і здешевлюють розробку різноманітних пристроїв. Мікроконтролер може керувати різними компонентами та отримувати від них дані, мінімізуючи потребу в додаткових елементах,

оскільки значна частина периферії вже інтегрована в його кристал. Використання мікроконтролерів дозволяє зменшити габарити конструкції та знизити енергоспоживання пристрою.

Мета роботи полягає у розробці апаратного та програмного забезпечення мікроконтролерного пристрою для моніторингу основних параметрів погоди. Цей пристрій повинен відстежувати метеорологічні показники, зберігати їх значення в енергонезалежній пам'яті та виводити отримані дані на шістнадцятисимвольний дворядковий рідкокристалічний дисплей.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- **Спроєктувати апаратне забезпечення** системи моніторингу фізико-географічних параметрів на базі мікроконтролера ATmega328P-PU платформи Arduino Uno R3, цифрових датчиків тиску BMP180 та вологості DHT11, аналогового датчика температури LM35, а також алфавітно-цифрового рідкокристалічного дисплея HD44780.
- **Створити модель** мікроконтролерної системи моніторингу фізико-географічних параметрів у САПР Proteus.
- **Розробити програмне забезпечення** для взаємодії з датчиками, РКД, а також для збору та обробки даних для МК ATmega328P-PU платформи Arduino Uno R3 мовою С у середовищі Arduino IDE.

Об'єкт дослідження – проєктування цифрового пристрою для моніторингу фізико-географічних параметрів. **Предмет дослідження** – моделі та засоби проєктування системи моніторингу фізико-географічних параметрів на мікроконтролері ATmega328P-PU платформи Arduino Uno R3, цифрових датчиках тиску BMP180 та вологості DHT11, аналоговому датчику температури LM35, алфавітно-цифровому рідкокристалічному дисплеї

HD44780. **Методи досліджень**. У процесі виконання роботи було застосовано низку загальновідомих методів пізнання, що дозволили повною мірою досягти поставленої мети та вирішити завдання дослідження. Зокрема, **бібліометричний метод** дозволив опрацювати опубліковані наукові праці та осягнути динаміку розвитку даної проблеми. За допомогою різних **філософських методів** (діалектичного, феноменологічного, синергетичного, аксіологічного) було здійснено всебічний аналіз поглядів на предмет дослідження. Використання значної кількості **загальнонаукових методів** (структурного, індуктивного, дедуктивного, аналізу, синтезу, моделювання) сприяло виконанню визначених вище теоретичних та практичних завдань дослідження.

Структура роботи. Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. Основний текст роботи займає 55 сторінок, містить 21 рисунок, 2 таблиці, 3 додатки та 26 бібліографічних джерел. Загальний обсяг роботи становить 100 сторінок.

РОЗДІЛ І.
АНАЛІЗ ТА ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Основні фізико-географічні параметри та їх вимірювання

Фізико-географічні параметри, також відомі як **метеопараметри**, охоплюють такі показники, як температура, атмосферний тиск, вологість повітря, швидкість та напрямок вітру, ступінь захмареності, кількість опадів.

Атмосферні явища являють собою фізичні процеси, що супроводжуються помітними якісними змінами стану атмосфери. До них належать, наприклад, дощ, сніг, іній, веселка, гроза.

Погода визначається як сукупність метеорологічних показників та атмосферних явищ у конкретний момент або протягом певного часового проміжку в заданій місцевості. **Клімат** же є багаторічним режимом погодних умов у певному географічному регіоні.

Температура — це величина, що характеризує тепловий стан тіла, а також ступінь його нагрітості. Температура повітря в різних місцях атмосфери нашої планети постійно змінюється. В один і той самий час вона може бути різною в різних куточках Землі. На поверхні Землі температурний діапазон досить широкий: від значень трохи нижче +60 °С (у тропічних регіонах) до приблизно -90 °С (на Антарктиді). Зі збільшенням висоти температура повітря також змінюється, причому ці зміни відбуваються по-різному в різних слоях атмосфери та за різних умов. У середньому температура спочатку знижується до висоти 10-15 км, потім зростає до 50-60 км, а потім знову спадає, і так далі. У системі СІ температура повітря вимірюється в **градусах міжнародної температурної шкали**, відомої як **шкала Цельсія** (°С), де за нуль прийнято температуру танення льоду, а температура кипіння води становить 100 °С. Окрім шкали Цельсія, широке розповсюдження має **абсолютна шкала температури — шкала Кельвіна**. Нуль цієї шкали означає повне припинення молекулярного руху, тобто

найнижчу можливу температуру, яка за шкалою Цельсія становить -273,15 °C. Одиницею абсолютної шкали є Кельвін. За абсолютною шкалою температура може бути лише додатною, тобто завжди вищою за абсолютний нуль. У формулах абсолютна температура позначається як T, а температура за Цельсієм — як t.

Для переведення температури з градусів Цельсія в Кельвіни використовується наступна формула: T=t+273.15

Існує ще одна температурна шкала, що застосовується, зокрема, у США — **шкала Фаренгейта**, запропонована Г. Фаренгейтом у 1724 році. Один градус за шкалою Фаренгейта (1 °F) дорівнює 1/180 різниці температур кипіння води та танення льоду. Точка танення льоду відповідає +32 °F. Зв'язок температури за Фаренгейтом (F) з температурою за Цельсієм (t °C) виражається наступною формулою: F=1.8·t+32

Отже, градус Фаренгейта є майже вдвічі меншим за градус стоградусної шкали. Нульові точки цих шкал не збігаються: 0 °F відповідає температурі -17,8 °C за шкалою Цельсія.

Тиск визначається як гідростатична сила тиску повітря, що припадає на одиницю площі. **Атмосферний тиск** вимірюється вагою стовпа повітря, розташованого вище, на одиницю горизонтальної поверхні. Загальна маса атмосфери, якою вона тисне на поверхню Землі, становить 5,15·1015 тонн. З часів Торрічеллі (XVII століття) тиск повітря вимірювали висотою ртутного стовпа в міліметрах або дюймах. Однак, коли почали впроваджувати різні розрахункові методи для аналізу та прогнозу стану атмосфери, виявилося, що лінійна міра — міліметри — не пов'язана з фізичною сутністю тиску як сили і є вкрай незручною. Тому в 20-х роках норвезький метеоролог В. Беркенс запропонував нову одиницю для вимірювання атмосферного тиску — **мілібар (мбар)**. Мілібар — це одиниця атмосферного тиску, що дорівнює 1000 дин на 1 см2 (де 1 дин — це сила, яка надає масі в 1 г прискорення руху в 1 см/с2). У мілібарах нормальний тиск (середній тиск на рівні моря на широті 45° при температурі повітря 0 °C) становить 1013,25 мбар або 760 мм

рт. ст. Стандартним тиском вважається тиск 1000 гПа або 750 мм рт. ст. У системі одиниць СІ тиск вимірюється в **паскалях (Па)**. Паскаль — це тиск, який створює сила в 1 Н, рівномірно розподілена по площі в 1 м2 (100 Па=1 гПа). Один гектопаскаль чисельно дорівнює одному мілібару. Одиницями вимірювання тиску є **гектопаскаль (гПа)**, **мілібар (мбар)**, **міліметр ртутного стовпчика (мм рт. ст.)**. Один гектопаскаль дорівнює 100 паскалям або одному мілібару. Один міліметр ртутного стовпа дорівнює 4/3 (приблизно 1,333) гектопаскаля. Один гектопаскаль дорівнює 3/4 (0,75) міліметрам ртутного стовпчика.

Вологість повітря

Вологість повітря — це показник вмісту водяної пари в атмосфері. Її можна оцінити за допомогою **абсолютної** та **відносної вологості**. Повітря вважається **насиченим вологою**, якщо за певної температури воно вже не може поглинати додаткову водяну пару. У такому випадку, навіть найменше охолодження призведе до конденсації та виділення крапельок води у вигляді роси, туману чи хмар. Натомість, **сухе повітря** має здатність до подальшого поглинання вологи.

Важливо розуміти, що тепліше повітря може утримувати значно більше водяної пари. Наприклад, при -20 °С повітря містить не більше 1 г/м3 води, тоді як при +10 °С — приблизно 9 г/м3, а при +20 °С — близько 17 г/м3. Через цю залежність, навіть якщо відносна вологість у тундрі висока, а в степу низька, їхня абсолютна вологість може бути однаковою через різницю в температурах. Визначення вологості повітря має вирішальне значення не лише для прогнозування погоди та вивчення клімату, а й для численних технічних сфер, таких як збереження книг та музейних експонатів, лікування легеневих захворювань і, особливо, при застосуванні штучного зрошення.

Відомо, що людський організм на 80-90% складається з води, тому рівень вологості в атмосфері суттєво впливає на життя людини. Вміст вологи в повітрі безпосередньо впливає на **самопочуття людини**. Відхилення цього параметра від оптимальних значень може непомітно й поступово знизити

імунітет, погіршити стан шкіри та підвищити втомлюваність. Натомість, нормальний рівень вологості у діапазоні 45-65% є корисним для.

Вітер

Вітер – це рух повітряних мас відносно поверхні Землі. Його характеризує **вектор швидкості**, який, як відомо, визначається як абсолютною величиною, так і напрямком. **Напрямок вітру** відповідає напрямку цього вектора швидкості. Зазвичай його визначають як азимут точки, звідки дме вітер, відраховуючи його від Півночі за годинниковою стрілкою (через Схід).

Швидкість вітру вимірюють у **метрах за секунду (м/с)**. В авіаційній сфері швидкість вітру частіше подають у **км/год**, а на флоті – у **вузлах** (морських милях за годину). Щоб перевести швидкість вітру з м/с у вузли, треба помножити значення в м/с на два.

Швидкість вітру також оцінюється за **шкалою Бофорта**, яка є дванадцятибальною. Вона прийнята за стандарт Всесвітньою метеорологічною організацією і використовується для приблизного визначення швидкості вітру за його видимим впливом на наземні об'єкти або за ступенем хвилювання на відкритому морі. Шкалу розробив англійський адмірал Френсіс Бофорт у 1806 році, а з 1874 року її було ухвалено для використання в міжнародній синоптичній практиці. У 1955 році, для більш точного розрізнення ураганних вітрів різної сили, Бюро погоди США розширило шкалу до 17 балів.

Табл.1.1. Шкала Бофорта

Бал за шкалою Бофорта	Опис	Швидкість вітру (м/с)
0	Штиль	< 0.3
1	Тихий вітер	0.3 - 1.5
2	Легкий вітер	1.6 - 3.3
...	...	...
12	Ураган	> 32.7

Швидкість вітру біля земної поверхні вимірюють за допомогою **анемометрів** різних конструкцій або **флюгера Вільда**. Найпоширенішими є анемометри з обертовими приймальними частинами, як-от **чашковий анемометр** або **млиновий анемометр**. Ці прилади обертаються з різною швидкістю, залежно від тиску вітру.

Анемометр був винайдений ще в XVII столітті англійським вченим Робертом Гуком. Його назва походить від двох давньогрецьких слів: "анемос" (вітер) і "метрон" (вимірюю). Обертання вертушки анемометра дозволяло обчислювати поточну швидкість вітру в метрах за секунду. Знаючи напрямок і швидкість вітру, можна прогнозувати зміни в погоді на найближчий час.

Рис.1.1. Цифровий портативний анемометр AR856 з можливістю підключення до ПК через USB інтерфейс

Флюгер Вільда — це комбінований метеорологічний прилад, призначений для одночасного вимірювання як напрямку, так і швидкості вітру. Його конструкція поєднує елементи традиційного флюгера та

анемометра. Швидкість вітру можна визначити за швидкістю обертання вертушки або за відхиленням спеціальної дошки.

Існують також інші типи вітровимірювальних приладів, засновані на різних принципах:

- **Манометричні прилади** (наприклад, трубка Піто) використовують різницю тиску.
- **Термоанемометри** вимірюють швидкість вітру за величиною охолодження нагрітого тіла.

Крім того, розроблено низку самописних приладів: **анемографи** (для запису швидкості вітру) та **анеморумбографи** (для запису як швидкості, так і напрямку вітру). На наземних метеорологічних станціях прилади для вимірювання вітру встановлюються на висоті 10-12 метрів над земною поверхнею. Виміряний таким чином вітер називають **вітром біля земної поверхні**.

Хмарність

Хмарність — це сукупність хмар, що спостерігається в певній точці (або на певній території) у конкретний момент чи період часу. Вона є одним із ключових факторів, що визначають погодні умови та клімат. Завдяки своєму екрануючому ефекту хмарність запобігає як надмірному охолодженню земної поверхні через власне теплове випромінювання, так і її надмірному нагріванню сонячним випромінюванням. Таким чином, хмари зменшують сезонні та добові коливання температури повітря.

Кількість хмар відображає ступінь покриття неба хмарами і вимірюється за 10-бальною шкалою або у відсотках покриття неба. Ця 10-бальна шкала була прийнята на I Морській Міжнародній Метеорологічній Конференції, яка відбулася в Брюсселі у 1853 році. На метеорологічних станціях під час спостережень фіксується як загальна кількість хмар, так і кількість хмар нижнього ярусу. Ці показники зазвичай записуються в щоденниках погоди через дріб, наприклад, 10/4.

В авіаційній метеорології використовується простіша для візуального спостереження **8-октантна шкала**.

1.3. Автоматизовані метеостанції: Принцип роботи та сучасні розробки

Сучасні **автоматизовані метеостанції** є важливим інструментом для безперервного моніторингу погодних умов. Їхнє основне призначення — збір та реєстрація метеорологічних даних за допомогою **датчиків** без безпосереднього втручання людини. Зібрані відомості можуть або зберігатися на місці, або передаватися на центральний комп'ютер через різні канали зв'язку. Саме **комунікаційні можливості** є ключовим елементом таких систем.

Нині на ринку представлено безліч комерційних рішень для автоматизованого метеорологічного моніторингу. Окрім того, науково-дослідні установи у сферах метеорології та інженерної геології активно розробляють власні метеостанції. Вимірювання можуть здійснюватися з різною частотою — від періодичних до щогодинних або навіть частіших інтервалів. Сучасні автоматизовані метеостанції досягли високого рівня розвитку. Проте, точність спостережень та обсяг даних, які вони можуть надавати, часто обмежуються їхньою вартістю, що переважно залежить від ціни високоточних вимірювальних приладів та обладнання для бездротової передачі даних.

Існуючі мікроконтролерні рішення для метеомоніторингу

Глибокий аналіз доступної літератури та інтернет-джерел [1-20] виявив широкий спектр **мікроконтролерних систем**, призначених для вимірювання та моніторингу метеопараметрів. Багато з цих систем інтегрують **GSM-модулі**, що дозволяє їм дистанційно передавати дані в реальному часі.

Наведемо кілька прикладів таких розробок:

- **Метеостанція з радіочастотним модулем**: Дослідники Ісвант і Гельман Мухамед з Індонезійського університету (кафедра електротехніки) створили метеостанцію, що використовує **мікроконтролер AT89S51** та датчики для вимірювання швидкості

вітру, температури й вологості. Мікроконтролер обробляє отримані дані, виводячи швидкість вітру в м/с, температуру в °C та відносну вологість (RH%) на РК-дисплей. Їхня система складається з двох окремих модулів: передавального та приймального.

- **Система моніторингу на базі Zigbee**: В Індії Абдул Азіз та професор Навін Шрівастава розробили систему моніторингу метеопараметрів, використовуючи **бездротовий модуль Tarang Zigbee** та **мікроконтролер AVR**. Ця система відстежує температуру, вологість та концентрацію вуглекислого газу в атмосфері. Вибір Zigbee обґрунтований його достатньою швидкістю та здатністю працювати без мережі Wi-Fi. Система складається з двох блоків: пристрою для збору даних з датчиків та пристрою для обробки та управління. Перший блок бездротово передає зібрані дані на приймальний пристрій, який підключений до ПК. На комп'ютері встановлено спеціально розроблене програмне забезпечення для подальшої обробки, візуалізації, аналізу та збереження отриманих метеорологічних даних у файлах.
- **Мікроконтролерні метеостанції в журналі "Elektronika Praktyczna"**: У цьому виданні представлено розробки двох мікроконтролерних систем для вимірювання метеопараметрів — для використання в приміщенні та на вулиці [13, 14, 15, 16, 17, 18, 19, 20]. Внутрішня метеостанція побудована на **мікроконтролері PIC18F2560** та використовує датчики температури й вологості SHT75, а також датчик тиску MPX4115. Метеодані відображаються на графічному РК-дисплеї. Зовнішня метеостанція, призначена для вимірювання температури, розроблена на **мікроконтролері PIC16F628**, цифровому датчику температури DS1620 та мікросхемі MAX202 для узгодження рівнів TTTL/RS232, що дозволяє підключення до ПК.

Візуалізація розробок цифрових метеопристроїв

На рисунках 1.2 – 1.5 представлені різні приклади розробок цифрових пристроїв для вимірювання та моніторингу параметрів навколишнього

середовища. Ці пристрої, зазвичай, побудовані на базі мікроконтролерів і оснащені РК-дисплеями для відображення інформації.

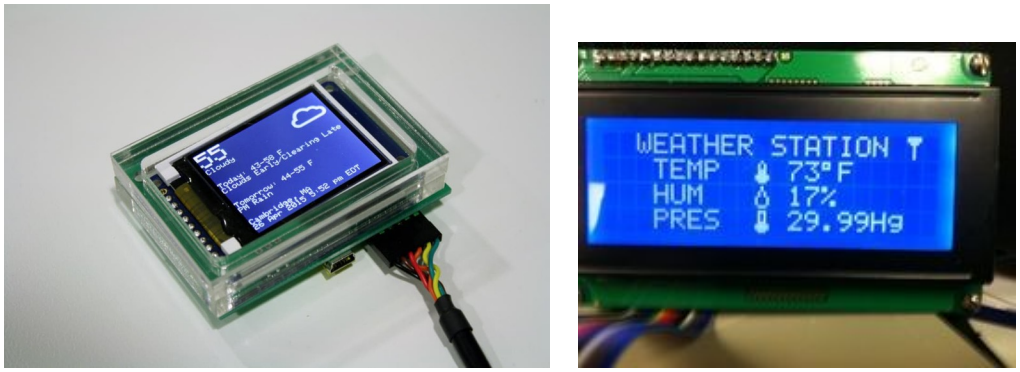


Рис. 1.2. Пристрої для виміру і спостережень за метеопараметрами.

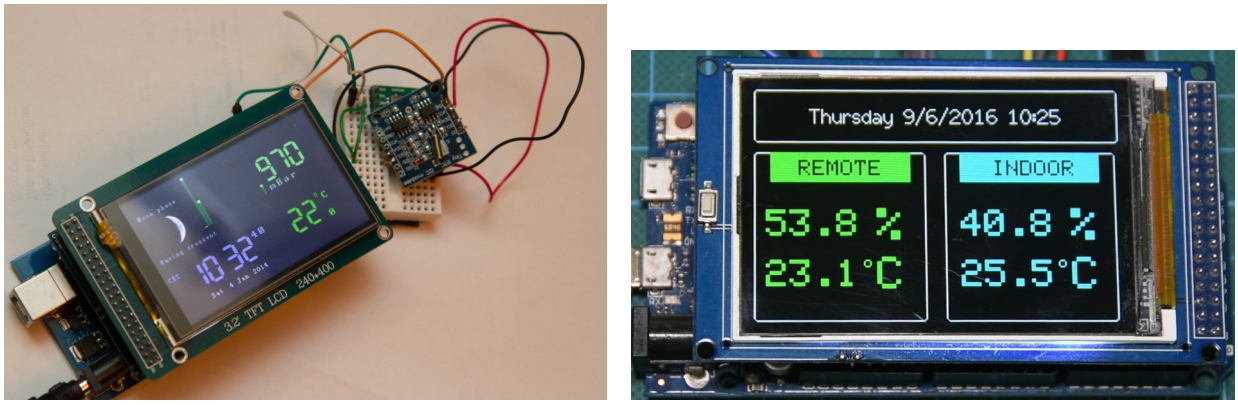


Рис. 1.3. Мікроконтролерні пристрої моніторингу метеопараметрів.

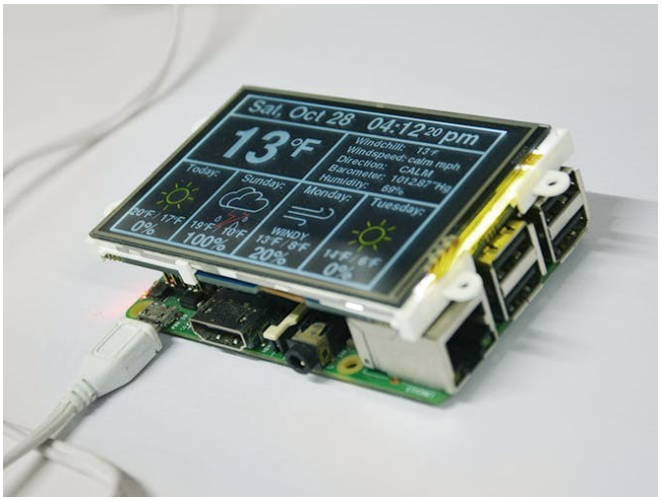


Рис. 1.4. Метеостанції на одноплатному комп'ютері Raspberry Pi.

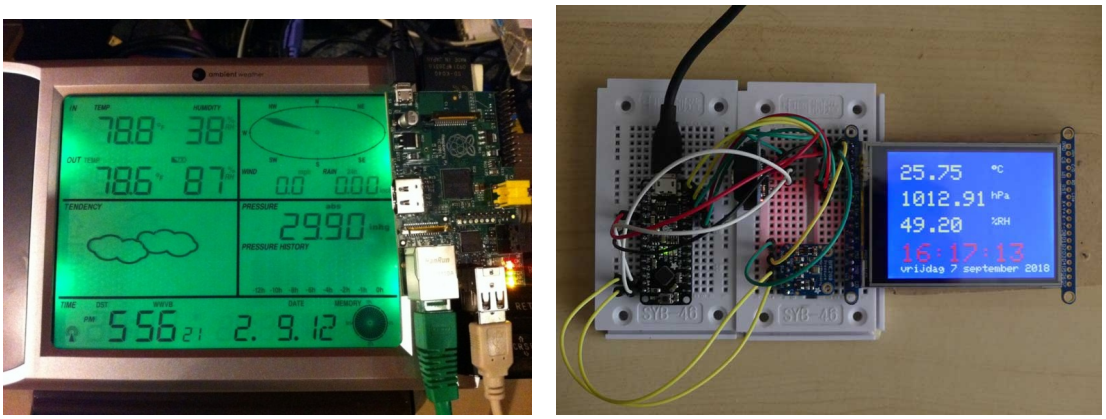


Рис. 1.5. Raspberry Pi дозволяє інтегрувати різноманітні датчики для вимірювання метеопараметрів. Інформація з цих датчиків обробляється і відображається на графічному кольоровому РК-дисплеї.

РОЗДІЛ ІІ.
РОЗРОБЛЕННЯ ТЕХНІЧНИХ І ПРОГРАМНИХ ЗАСОБІВ СИСТЕМИ МОНІТОРИНГУ ФІЗИКО-ГЕОГРАФІЧНИХ ПАРАМЕТРІВ

2.1. Proteus: Комплексний інструмент для проектування та моделювання мікропроцесорних систем

Proteus — це потужний програмний комплекс від британської компанії Labcenter Electronics, призначений для **автоматизованого проектування електронних схем**, зокрема тих, що базуються на **мікроконтролерах** різних сімейств. Ця система відрізняється своїми можливостями **схемотехнічного моделювання**, яке реалізується за допомогою віртуальних моделей електронних компонентів.

Ключові можливості Proteus

Однією з головних переваг Proteus є його здатність моделювати роботу різноманітних програмованих пристроїв, таких як **мікропроцесори, мікроконтролери, цифрові сигнальні процесори (DSP)** тощо. Користувачі можуть створювати **електричні принципові схеми**, виконувати їх **моделювання** за допомогою віртуальних приладів та проводити **налагодження програмного коду** (прошивки) для мікроконтролерів.

Пакет Proteus також включає функції для **проектування та моделювання друкованих плат (PCB)**, що доповнюється зручною 3D-візуалізацією. У Proteus можна моделювати роботу широкого спектра мікроконтролерів, включаючи сімейства **ARM7, 8051, PIC, AVR, Motorola**.

Бібліотеки компонентів та сумісність

Внутрішня бібліотека компонентів Proteus містить обширні довідкові дані та підтримує популярні мікроконтролери, такі як **8051, PIC, HC11, AVR, ARM7/LPC2000**, а також інші поширені мікроконтролери та

мікропроцесори. Крім того, Proteus містить понад 6000 цифрових та аналогових моделей різноманітних пристроїв.

Програмне забезпечення Proteus сумісне з більшістю компіляторів та асемблерів, що спрощує процес розробки. Воно забезпечує досить достовірне моделювання та налагодження складних пристроїв, які можуть містити декілька мікроконтролерів різних сімейств. Важливо пам'ятати, що жодне моделювання електронних схем не може абсолютно точно відтворити роботу реального пристрою. Однак для загального налагодження будь-якого алгоритму роботи мікроконтролера цього цілком достатньо.

Proteus також має великий набір електронних елементів схем, а ті моделі, яких немає, можуть бути створені користувачем самостійно. Передбачена підтримка **SPICE-моделей**, які часто надаються виробниками електронних компонентів. Це дозволяє завантажувати SPICE-моделі від виробників та додавати їх до відповідного корпусу, якщо якийсь компонент не програмується безпосередньо.

Основні модулі Proteus

Програма Proteus складається з двох ключових модулів:

1. **ISIS** — це редактор електронних схем, призначений для створення схем та подальшої імітації їхньої роботи.
2. **ARES** — це редактор друкованих плат, який включає автотрасувальник Electra, вбудований редактор бібліотек та автоматичну систему розміщення компонентів на платі. Крім того, ARES може генерувати тривимірну модель друкованої плати.

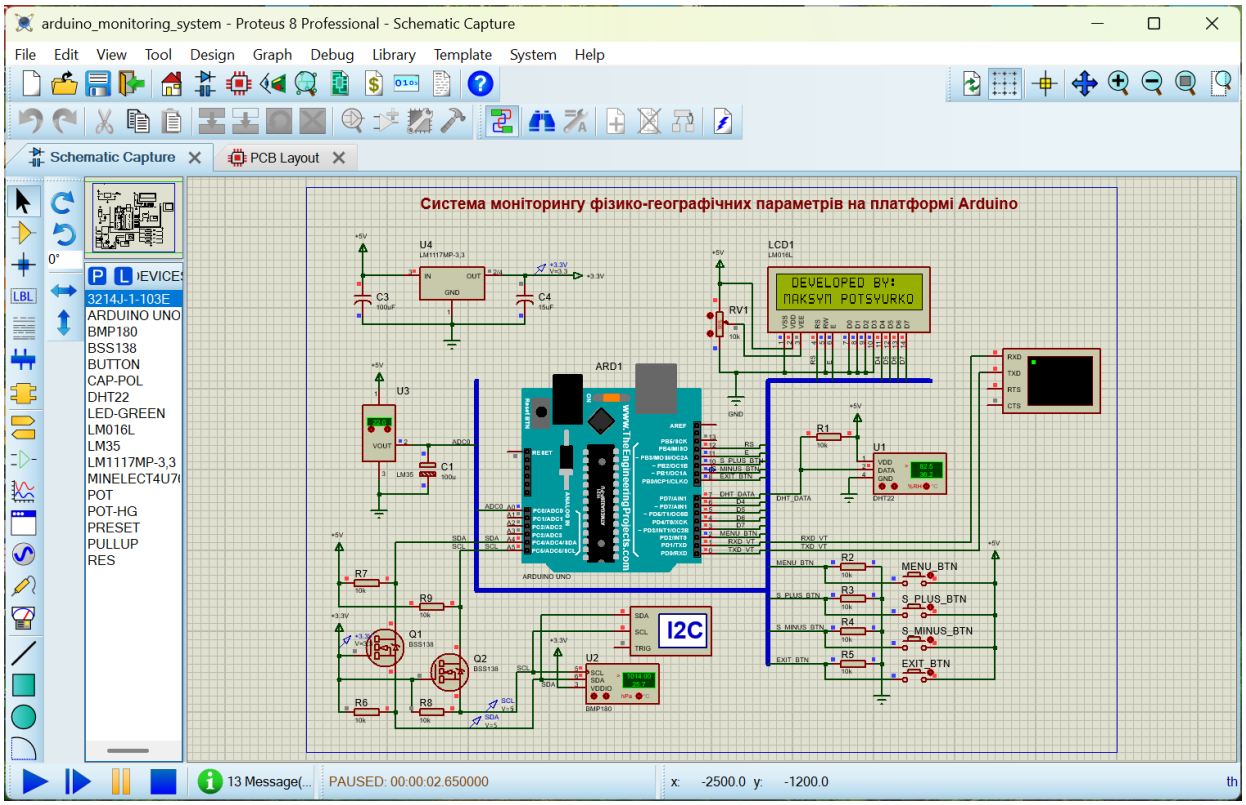


Рис.2.1. Зображення інтерфейсу середовища Proteus ISIS

2.2. Розроблення програмного забезпечення – аналіз середовища

Розробка програмного забезпечення для використання на платформі Arduino здійснюється в середовищі розробки **Arduino IDE**. Це середовище дозволяє писати, компілювати та завантажувати програми безпосередньо в пам'ять **мікроконтролера**, встановленого на сумісних з Arduino платах. В основі Arduino IDE лежить мова, яка є варіацією **C++**, доповненою простими функціями для керування вводом/виводом. Arduino IDE можна використовувати на системах Windows, Mac OS та Linux [7]. Останню версію, Arduino 1.8.9, можна завантажити з офіційного сайту за адресою http://arduino.cc/en/Main/Software.

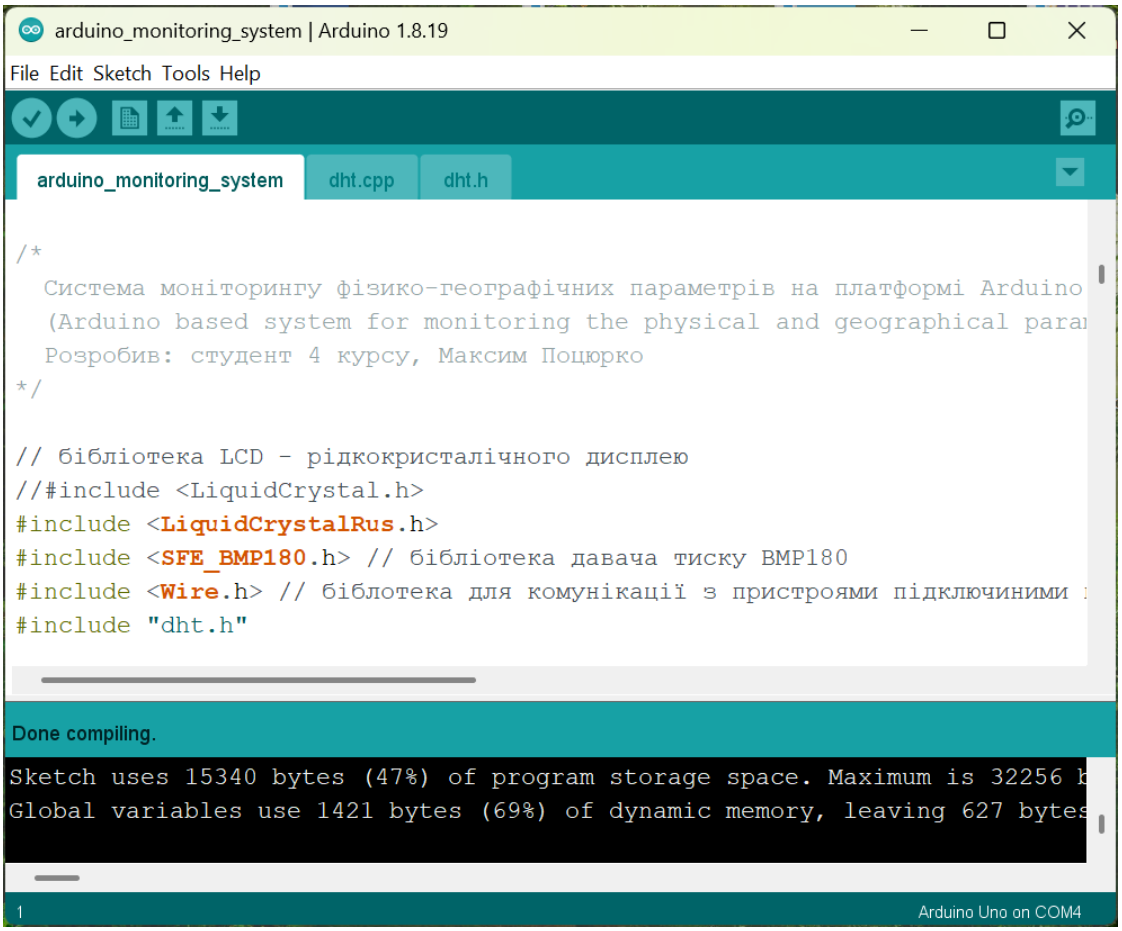


Рис. 2.2 Зображення середовища розробки ПЗ під платформу Arduino — Arduino IDE

Налаштування та компоненти Arduino IDE

Середовище розроблення Arduino IDE включає:

- **Редактор програмного коду**: Тут ви пишете свій код.
- **Область повідомлень**: Відображає пояснення та інформацію про помилки під час збереження або експорту проєкту.
- **Вікно виводу тексту**: Показує повідомлення від Arduino, включаючи повні звіти про помилки та іншу важливу інформацію.
- **Панель інструментів**: Містить кнопки для часто використовуваних команд, таких як перевірка та завантаження програми, створення, відкриття та збереження скетчів, а також відкриття монітора послідовного порту.
- **Декілька меню**: Для доступу до різних функцій та налаштувань.

Програма, написана в Arduino IDE, називається **скетчем**. Редактор тексту підтримує **кольорову підсвітку коду** для зручності.

Бібліотеки та їх використання

Для розширення функціональності скетчів можна використовувати **бібліотеки**. Це спеціально оформлений програмний код, який реалізує певні можливості та може бути підключений до вашого проєкту. Існує безліч спеціалізованих бібліотек, які спрощують вирішення складних завдань, приховуючи деталі програмно-апаратної реалізації.

Arduino IDE постачається зі стандартним набором бібліотек (Serial, EEPROM, SPI, Wire тощо), які знаходяться у підпапці `libraries` в каталозі встановлення Arduino. Додаткові бібліотеки можна завантажити з різних джерел. Щоб встановити нову бібліотеку, її папку слід скопіювати до каталогу стандартних бібліотек. У каталозі кожної бібліотеки зазвичай знаходяться файли `.cpp` та `.h`. Багато бібліотек також супроводжуються прикладами застосування у папці `examples`.

Якщо бібліотека встановлена коректно, вона з'явиться в меню **Sketch | Import Library**. Вибір бібліотеки з меню автоматично додасть до вашого коду рядок:

```
#include <назва_бібліотеки.h>
```

Ця директива підключає заголовний файл, що містить опис об'єктів, функцій та констант бібліотеки, які тепер можна використовувати у вашому проєкті. Arduino IDE компілюватиме ваш скетч разом із зазначеною бібліотекою.

Завантаження скетчу та монітор послідовного порту

Перед завантаженням скетчу у пам’ять платформи необхідно налаштувати параметри в меню **Сервіс | Плата (Tools | Board)** та **Сервіс | Послідовний порт (Tools | Serial Port)**. Плати Arduino сьогодні автоматично перезавантажуються перед завантаженням коду, тоді як на старих моделях

може знадобитися натискання кнопки перезавантаження. Під час процесу завантаження у більшості випадків будуть блимати світлодіоди RX і TX.

Для завантаження скетчу застосовуємо **завантажувач (bootloader)** Arduino, який вже записаний в МК плати. Він дозволяє завантажувати програмний код без застосування додаткових апаратних засобів. Роботу завантажувача можна розпізнати через блимання світлодіода на виводі D13.

Serial Monitor показує дані, що передаються на платформу Arduino (через USB-інтерфейс або послідовну шину). Для відправлення даних на плату необхідно ввести текст у відповідне поле та натиснути кнопку **Відіслати (Send)** або клавішу `<Enter>`. Також слід вибрати швидкість передачі зі списку, яка має відповідати значенню `Serial.begin()` у вашому скетчі. Зверніть увагу, що при підключенні монітора послідовного порту на операційних системах Mac або Linux плата Arduino перезавантажиться і скетч почне виконуватися спочатку.

Структура програми на Arduino

Програми для платформи Arduino створюються на мові, яку часто називають **Wiring**. Хоча окремої мови чи компілятора Wiring не існує, насправді код, написаний у цьому стилі, перетворюється на програму на **C/C++**, а потім компілюється за допомогою **AVR-GCC**. По суті, використовується спеціалізований варіант C/C++ для МК AVR.

Програма для Arduino обов'язково містить дві функції: **`setup()`** та **`loop()`**. Перед функцією `setup()` зазвичай оголошуються змінні та підключаються бібліотеки.

- функція **setup()**: виконується лише раз після ввімкнення живлення або перезавантаження плати Arduino. Вона застосовується для ініціалізації

змінних, налаштування режимів роботи портів та виконання інших підготовчих дій для основного циклу програми. Навіть якщо вона не виконує жодних дій, ця функція повинна бути в програмі.

- **loop()**: Ця функція виконується циклічно, виконуючи команди, описані в ній. Вона є основним робочим циклом програми і постійно повторюється.

РОЗДІЛ ІІІ.
ТЕХНІЧНЕ ОБЛАДНАННЯ СИСТЕМИ МОНІТОРИНГУ ФІЗИКО-ГЕОГРАФІЧНИХ ПАРАМЕТРІВ

3.1 Вибір електронних компонентів для системи моніторингу фізико-географічних параметрів

Для розробки апаратної частини мікроконтролерної системи моніторингу фізико-географічних параметрів були обрані наступні електронні компоненти:

- **Платформа Arduino Uno R3** з мікроконтролером ATmega328P-PU.
- **Символьний рідкокристалічний дисплей** на контролері HD44780.
- **Цифровий датчик температури та тиску BMP180**.
- **Цифровий датчик температури та вологості DHT11**.
- **Аналоговий датчик температури LM35**.
- **Узгоджувач рівнів** 5 В – 3,3 В.
- **Клавіатура** (з чотирьох кнопок).

Огляд Платформи Arduino

Сучасна мікроелектронна індустрія пропонує безліч різноманітних контролерів, що відрізняються за ціною, розмірами, потужністю та надійністю. Серед них **Arduino** виділяється як один із найдоступніших та найпростіших для освоєння варіантів.

Arduino – це **апаратно-програмна платформа**, призначена для створення простих систем автоматики та робототехніки. Її програмна частина включає **безкоштовне інтегроване середовище розробки (IDE)**, яке дозволяє писати програми, компілювати їх та завантажувати (прошивати) в мікроконтролер Arduino. Апаратна частина складається з різних моделей **друкованих плат**, таких як Arduino Uno R3, Arduino Pro Mini, Arduino Mega2560, Arduino Nano та інші, що виробляються як офіційною компанією, так і іншими

виробниками. **Відкрита архітектура** системи Arduino дозволяє вільно копіювати або доповнювати продукцію Arduino.

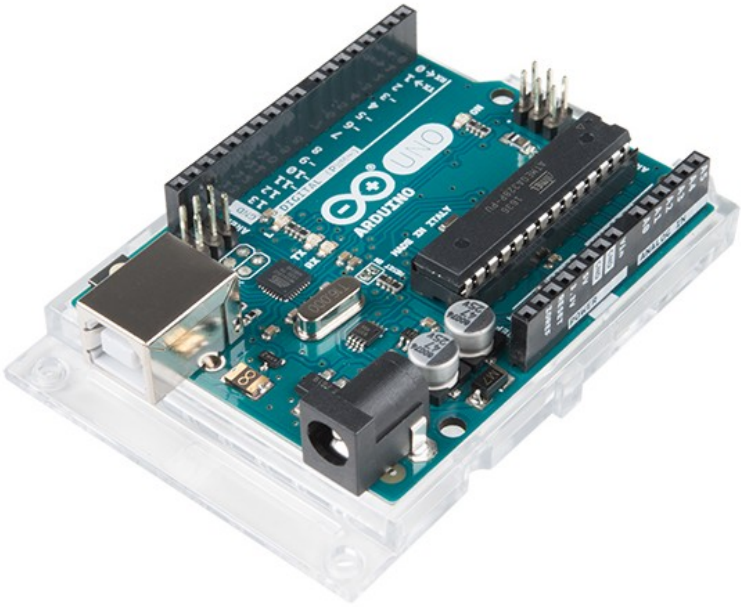


Рис.3.1. Зображення плати Arduino Uno R3

Характеристики Arduino Uno R3

Arduino Uno базується на мікроконтролері **ATmega328** і має чотирнадцять цифрових входів/виходів, 6 з них можуть застосовуватися як **виходи з широтно-імпульсною модуляцією**. Вона також оснащена шістьма аналоговими входами, **кварцовим генератором на 16 МГц**, USB-роз'ємом, роз'ємом живлення, роз'ємом ICSP та кнопкою перезавантаження. Для початку треба з’єднати платформу з ПК USB-кабелем, чи заживити від AC/DC-адаптера чи батареї. На відміну від попередніх версій плат, які використовували мікроконтролер FTDI USB для зв'язку через USB, нова Arduino Uno використовує мікроконтролер **ATmega8U2**.

Табл.3.1. Технічні характеристики плати Arduino Uno

Характеристика	**Значення**
Мікроконтролер	ATmega328
Робоча напруга	5 В
Вхідна напруга (рекомендована)	7-12 В

Вхідна напруга (гранична)	6-20 В
Цифрові входи/виходи	14 (з них 6 підтримують ШІМ)
Аналогові входи	6
Максимальний струм на вхід/вихід	40 мА
Максимальний струм для виводу 3,3 В	50 мА
Flash-пам'ять	32 КБ (ATmega328), з яких 0,5 КБ зарезервовано під завантажувач
Оперативна пам'ять (ОЗП)	2 КБ (ATmega328)
EEPROM	1 КБ (ATmega328)
Тактова частота	16 МГц

Живлення Arduino Uno

Плата Arduino Uno може живитися від **зовнішнього джерела** або через **USB-порт**. Переключення джерела живлення відбувається автоматично. Зовнішнє живлення може подаватися від **акумуляторної батареї або** блока живлення, який підключається через роз'єм 2,1 мм з плюсовим центральним виводом. Проводи від батареї необхідно під'єднати до виводів Gnd і Vin роз'єму живлення.

Платформа здатна працювати при напрузі 6 - 20 В. Однак, якщо напруга нижче 7 В, це може призвести до нестабільної роботи плати. При напрузі понад 12 В є ризик перегріву регулятора напруги та пошкодження плати. **Рекомендований діапазон напруги 7 - 12 В.**

Виводи Живлення

- **VIN**: Цей вхід використовується для живлення від зовнішнього джерела, у разі якщо 5 В не надходить від USB-порту.
- **5V**: Регульоване джерело напруги, що живить мікроконтролер та інші компоненти на платі. Живлення може надходити від виводу VIN через регулятор напруги, від USB-порту або іншого зовнішнього регульованого джерела 5 В.
- **3V3**: На цьому виводі формується напруга 3,3 В за допомогою вбудованого регулятора плати. Максимальний струм, який можна споживати з цього виводу, становить 50 мА.

- **GND**: Виводи заземлення.

Пам'ять в Arduino

Мікроконтролери сімейств ATmega168, ATmega328, ATmega1280 та ATmega2560, що використовуються на платформах Arduino, мають три типи пам'яті:

- **Флеш-пам'ять**: Призначена для зберігання скетчів (програм).
- **ОЗП (статична оперативна пам'ять, SRAM)**: Використовується для зберігання та обробки змінних під час виконання програми. Це енергозалежна пам'ять, тобто дані втрачаються при відключенні живлення.
- **EEPROM (енергонезалежна пам'ять)**: Застосовується для постійного зберігання інформації, яка не повинна бути втрачена при вимкненні живлення.

Флеш-пам'ять та EEPROM є **енергонезалежними** видами пам'яті, що означає збереження даних при відключенні живлення.

Мікроконтролер ATmega328 на платі Arduino Uno має 32 КБ **flesh-пам'яті** (з яких 0,5 КБ відведено для завантажувача), 2 КБ **ОЗП (SRAM)** та 1 КБ (1024 байти) **енергонезалежної пам'яті EEPROM**, доступ до якої здійснюється за допомогою бібліотеки EEPROM.

Виводи Плати Arduino Uno

Кожен із 14 цифрових виводів Arduino Uno можна налаштувати як **вхід** або **вихід** за допомогою функцій pinMode(), digitalWrite() та digitalRead(). Ці виводи працюють при напрузі 5 В, мають внутрішній підтягуючий резистор (за замовчуванням вимкнений) з опором 20-50 кОм і можуть пропускати струм до 40 мА. Деякі виводи потрібні для певних потреб:

- **Послідовна шина (UART)**: Виводи 0 (RX) та 1 (TX) використовуються для прийому (RX) та передачі (TX) даних TTL. Вони підключені до відповідних виводів мікросхеми ATmega8U2, яка забезпечує USB-to-TTL зв'язок.

- **Зовнішні переривання**: Виводи 2 та 3 можна налаштувати для виклику переривань при низькому значенні сигналу, по фронту (висхідному або спадному) або при зміні значення. Детальніше про це можна дізнатися в описі функції attachInterrupt().
- **ШІМ (PWM)**: Виводи 3, 5, 6, 9, 10 та 11 підтримують ШІМ з 8-бітною роздільною здатністю, використовуючи функцію analogWrite().
- **SPI (Serial Peripheral Interface)**: Виводи 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK) використовуються для SPI-зв'язку, для чого застосовується бібліотека SPI.
- **LED**: Вивід 13 підключений до вбудованого світлодіода. Він світиться, коли на виводі встановлено високий потенціал.

Платформа Uno має 6 аналогових входів (A0..A5), кожен з 10-бітною роздільною здатністю (тобто здатний розрізняти 1024 різних значення). За замовчуванням ці виводи вимірюють напругу до 5 В відносно землі. Однак, верхню межу вимірювання можна змінити за допомогою виводу AREF (джерело опорної напруги) та функції analogReference(). Деякі аналогові виводи також мають додаткові функції:

- **I2C (TWI)**: Виводи 4 (SDA) та 5 (SCL) використовуються для I2C-зв'язку (Two Wire Interface), для роботи з яким інтегрована бібліотека Wire.

Додаткові виводи платформи:

- **AREF**: Опорна напруга для аналогових входів. Використовується в поєднанні з функцією analogReference().
- **Reset**: Низький логічний рівень на цьому виводі призводить до перезавантаження МК. Цей вивід часто використовується для підключення кнопки перезавантаження на платах розширення, що відключає кнопку на платі Arduino.

Комунікаційні Можливості

Платформа Arduino Uno оснащена кількома інтерфейсами для обміну даними з комп'ютером, іншими платами Arduino або мікроконтролерами. Мікроконтролер ATmega328 підтримує послідовний інтерфейс **UART TTL (5 В)**, реалізований на виводах 0 (RX) та 1 (TX). Вбудована мікросхема ATmega8U2 маршрутизує цей інтерфейс через USB, дозволяючи програмам на комп'ютері взаємодіяти з платою через **віртуальний COM-порт**. Прошивка ATmega8U2 використовує стандартні USB COM-драйвери, тому додаткові драйвери зазвичай не потрібні, хоча для Windows може знадобитися файл ArduinoUNO.inf.

Монітор послідовного порту (Serial Monitor) в Arduino IDE дозволяє пересилати текстові дані між комп’ютером та платою Arduino. Світлодіоди RX і TX на платі будуть блимати під час передачі даних через мікросхему FTDI або USB-з'єднання (але не при використанні послідовної передачі через виводи 0 і 1). Використовуючи бібліотеку **SoftwareSerial**, можна реалізувати запрограмовану передачу даних послідовно через будь-який цифровий вивід плати. Крім того, МК ATmega328 підтримує інтерфейси **SPI** та **I2C (TWI)**. Arduino IDE включає бібліотеку Wire для зручної роботи з шиною I2C.

Струмовий Захист USB-Порту

Arduino Uno має вбудований **самовідновлювальний автоматичний запобіжник** для захисту USB-порта комп'ютера від коротких замикань та значних струмів. Хоча більшість комп'ютерів мають власний захист, цей запобіжник забезпечує додатковий захист. Він спрацьовує, якщо струм через USB-порт перевищує 500 мА, розмикаючи ланцюг до відновлення нормальних значень струму.

Фізичні Характеристики

Друкована плата Uno має довжину 6,9 см та ширину 5,3 см. Роз'єм USB та роз'єм живлення виступають за ці розміри. Чотири монтажні отвори на платі дозволяють надійно закріпити її на поверхні. Відстань між цифровими

виводами 7 і 8 становить 0,4 см, тоді як між іншими виводами вона дорівнює 0,25 см.

Опис Цифрового Датчика Тиску та Температури BMP180

BMP180 — це інтегрований **п'єзорезистивний датчик тиску та температури**, виготовлений за мікромашинною технологією (Рис. 3.2). Кожен датчик проходить індивідуальне калібрування на заводі. Завдяки вбудованому термометру та калібрувальним коефіцієнтам, що зберігаються безпосередньо в датчику, він забезпечує високу абсолютну точність, раніше недосяжну для пристроїв такого класу. Датчик здатний реєструвати зміни висоти з точністю до 0,2-0,3 метра. Модуль BMP180 може бути використаний у погодних станціях, датчиках вертикальної швидкості для літальних апаратів та висотомірах, крім цього дозволяє контролювати тиск в медичних приладах та системах вентиляції.

Рис.3.2. Зображення цифрового датчик тиску та температури BMP180 та модулів з цим датчиком

Цифровий датчик тиску та температури BMP180

BMP180 є сучасним, функціонально сумісним наступником моделі BMP085, представляючи нове покоління високоточних цифрових датчиків тиску, розроблених для широкого спектру застосувань. Цей датчик відрізняється **низьким енергоспоживанням** та оптимізований для інтеграції в **мобільні пристрої**, КПК, GPS-навігатори та обладнання для зовнішнього використання. Завдяки низькому висотному шуму (всього 0,25 м при

швидкому часі перетворення), BMP180 демонструє **відмінні експлуатаційні характеристики**.

Технологія та Особливості

Підключення BMP180 до мікроконтролера спрощується завдяки його **інтерфейсу I2C**. Датчик базується на **п'єзорезистивній технології**, що забезпечує високу **електромагнітну сумісність**, точність, лінійність та довготривалу стабільність.

Компанія Robert Bosch, світовий лідер у виробництві автомобільних датчиків тиску з більш ніж 400 мільйонами виготовлених сенсорів, продовжує розвивати мікро-машинні (мікро-механічні) датчики тиску, і BMP180 є яскравим представником цього нового покоління.

Принцип Роботи та Калібрування

BMP180 розроблений для прямого підключення до мікроконтролера мобільного пристрою через шину I2C. Для отримання точних показань, необроблені дані тиску та температури мають бути скомпенсовані з використанням **калібрувальних даних**, що зберігаються у внутрішній пам'яті **E2PROM** датчика BMP180.

Структурно BMP180 складається з п'єзорезистивного елемента, аналого-цифрового перетворювача, модуля управління пам'яттю E2PROM та послідовного інтерфейсу I2C. Датчик передає необроблені значення тиску (UP, 16-19 біт) та температури (UT, 16 біт). Пам'ять E2PROM зберігає 176 біт індивідуальних калібрувальних даних, які використовуються для компенсації зміщення, температурної залежності та інших параметрів датчика, забезпечуючи високу точність вимірювань.

На **Рис. 3.3** представлена схема підключення датчика BMP180 до мікроконтролера по шині I2C.

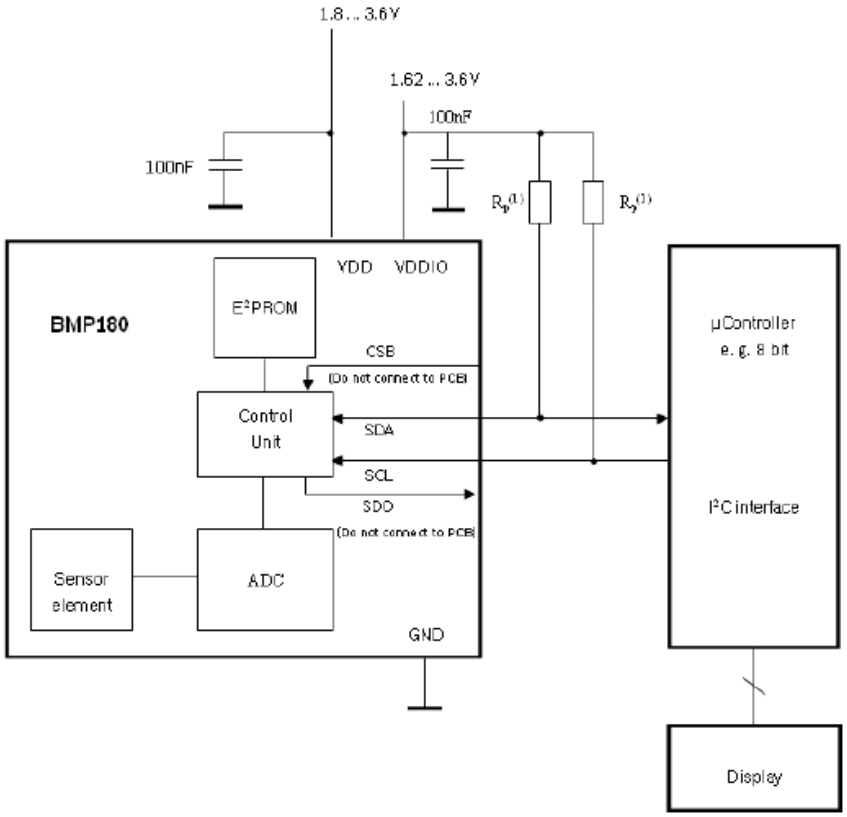


Рис.3.3. Схема підключення датчика BMP180 до мікроконтролера по шині I^2C

Схема підключення датчика BMP180 до мікроконтролера та узгодження рівнів

Для коректної роботи шини **I2C**, що використовується для зв'язку з датчиком BMP180, необхідні **підтягуючі резистори** (Rp). Їх номінал зазвичай варіюється від 2,2 кОм до 10 кОм, при цьому типовим значенням є 4,7 кОм.

На **Рис. 3.4** представлена схема, яка демонструє **узгодження рівнів напруги** між датчиком BMP180 та мікроконтролером AVR. Це узгодження є важливим, оскільки BMP180, ймовірно, працює з логічними рівнями 3,3 В, тоді як мікроконтролери AVR на платах Arduino (як-от ATmega328P-PU) зазвичай використовують логічні рівні 5 В. Схема узгодження забезпечує безпечну та ефективну взаємодію між компонентами з різними робочими напругами.

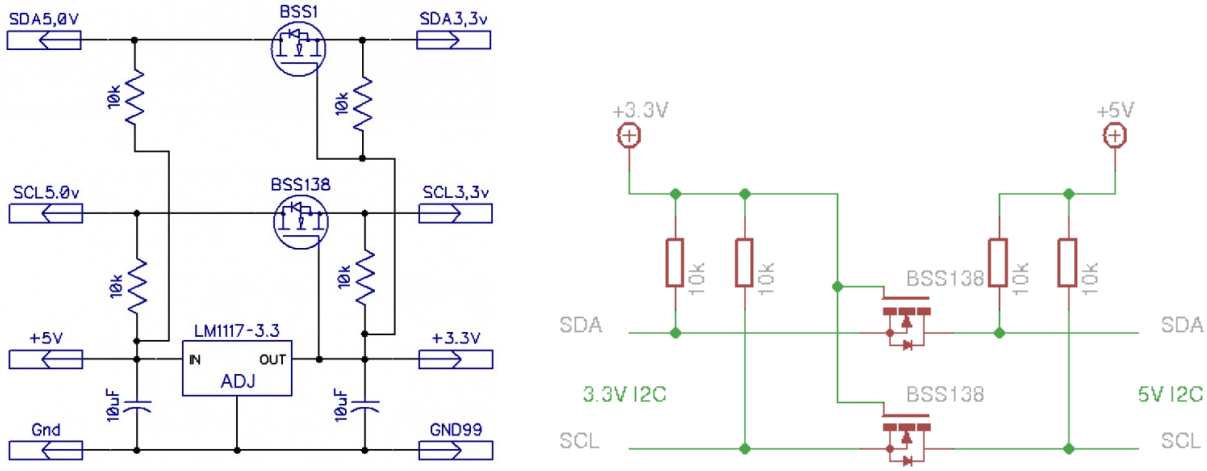


Рис.3.4. Схема погодження рівнів напруги від 5 до 3,3 В для підєднання BMP180 до мікроконтролера

Алгоритм вимірювання тиску та температури за допомогою BMP180

Для початку вимірювання тиску та температури, мікроконтролер надсилає на датчик BMP180 спеціальну стартову послідовність (як показано на Рис. 3.6). Після завершення процесу перетворення, що займає певний час, отримані значення — UP (для тиску) або UT (для температури) — можуть бути зчитані через інтерфейс I2C.

Щоб отримати точні значення температури в градусах Цельсія (°C) та тиску в гектопаскалях (гПа), необхідно застосувати калібрувальні дані. Ці унікальні для кожного датчика константи зберігаються у внутрішній пам'яті BMP180 E2PROM і повинні бути зчитані мікроконтролером під час програмної ініціалізації.

Для задач динамічного вимірювання, де потрібна висока швидкість збору даних, частоту вибірки можна збільшити до 128 вибірок за секунду (це стандартний режим). У таких випадках немає потреби вимірювати температуру при кожній вибірці тиску. Достатньо виміряти температуру лише один раз на секунду і використовувати це значення для всіх вимірювань тиску протягом цього ж періоду.

Рис.3.5. Вимірювання за допомогою BMP180

Калібрувальні Коефіцієнти Датчика BMP180

Для досягнення високої точності вимірювань, кожен датчик BMP180 містить у своїй **E2PROM-пам'яті** унікальні **калібрувальні коефіцієнти**. Ці дані, обсягом 176 біт, організовані в 11 16-бітних словах і є індивідуальними для кожного сенсорного модуля.

Перед першим обчисленням температури та тиску, **керуючий пристрій** (мікроконтролер, який взаємодіє з датчиком) повинен **зчитати ці калібрувальні дані** з E2PROM через інтерфейс I2C під час програмної ініціалізації. Для перевірки цілісності даних можна переконатися, що жодне зі зчитаних слів не має значення 0 або 0xFFFF.

Алгоритм Обчислення Тиску та Температури

На **Рис. 3.6** візуально представлено **алгоритм вимірювання тиску та температури** за допомогою датчика BMP180. Цей алгоритм включає етапи зчитування необроблених даних (UP для тиску, UT для температури) та їх подальшу компенсацію за допомогою отриманих калібрувальних

коефіцієнтів для перетворення на точні значення в градусах Цельсія та гектопаскалях.

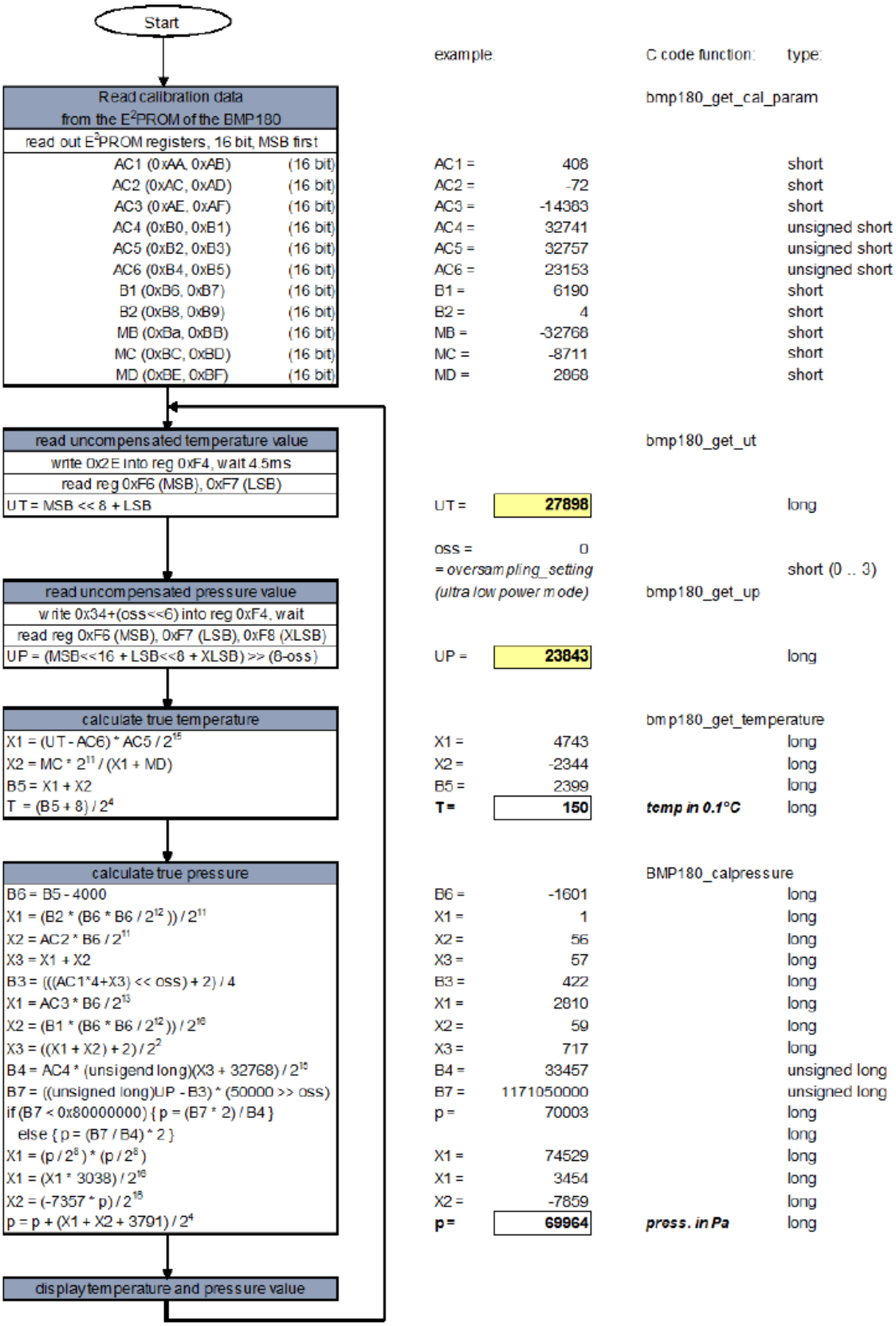


Рис.3.6. Процедура вимірів P і T

Налаштування режиму вимірювання та розрахунок висоти

Датчик BMP180 дозволяє обирати різні **режими вимірювання** — від ультраекономного до ультрависокої роздільної здатності. Це здійснюється за допомогою змінної oversampling_setting, якій можна присвоїти значення від 0 до 3 у програмному коді. Таким чином, можна налаштувати баланс між точністю вимірювань і енергоспоживанням.

Після обробки даних, датчик надає результати з високою деталізацією: **тиск** обчислюється з кроком 1 Па (що відповідає 0,01 гПа або 0,01 мБар), а **температура** — з кроком 0,1 °C.

Обчислення абсолютної висоти

Знаючи виміряний **тиск p** та **тиск на рівні моря p0** (наприклад, 1013 гПа, що є стандартним атмосферним тиском), можна розрахувати **висоту в метрах** за допомогою **міжнародної барометричної формули**:

$$altitude = 44330 * \left(1 - \left(\frac{p}{p0}\right)^{\frac{1}{5,255}}\right) \quad (3.1)$$

Ця формула дозволяє перетворити показання тиску на відповідну абсолютну висоту.

На основі формули бачимо, що зміні тиску на 1 гПа відповідає зміні висоти приблизно на 8,43 метра над рівнем моря. Цей зв'язок є ключовим для перетворення барометричних даних на показники висоти.

Опис цифрового датчика вологості та температури DHT11

DHT11 — це доступний і простий у використанні цифровий датчик вологості та температури (Рис. 3.9). Він ідеально підходить для проектів, де потрібне отримання цих параметрів у цифровому вигляді.

Принцип його роботи базується на комбінації двох основних вимірювальних елементів:

- Ємнісний датчик вологості: Вимірює відносну вологість повітря, фіксуючи зміни електричної ємності спеціального полімеру, що реагує на вміст водяної пари.
- Термістор: Використовується для визначення температури. Термістор — це тип резистора, опір якого змінюється залежно від температури (зазвичай зменшується при її зростанні).

Всі виміряні дані обробляються вбудованим в DHT11 мікроконтролером, який перетворює аналогові показання з ємнісного датчика та термістора в калібрований цифровий сигнал. Цей цифровий сигнал потім передається по одній інформаційній шині, що значно спрощує підключення датчика до мікроконтролерів, таких як Arduino.

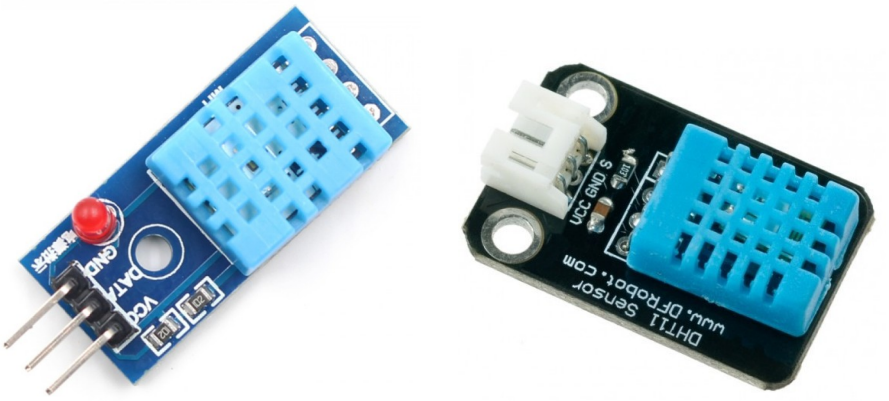


Рис.3.7. DHT11

Розроблена програмно-технічна система, окрім вище наведеного, використовує:

аналоговий даватчик температури LM35;

алфавітно цифровий рідкокристалічний дисплей 16×2.

3.2. Проєктування апаратного середовища системи моніторингу фізико-географічних параметрів у САПР Proteus

Для створення **мікроконтролерної системи моніторингу фізико-географічних параметрів**, здатної відстежувати температуру, відносну вологість повітря та атмосферний тиск, було розроблено її апаратну частину в системі автоматизованого проектування (САПР) **Proteus**.

На **Рис. 3.15** представлена структурна схема апаратного забезпечення цієї мікроконтролерної системи. В її основі лежить **платформа Arduino Uno R3** з інтегрованим **мікроконтролером ATmega328P-PU**. До мікроконтролера Arduino Uno підключені наступні датчики:

- **Датчик тиску BMP180**
- **Датчик температури LM35**

- **Датчик вологості DHT11**

Мікроконтролер Arduino Uno відповідає за **зчитування та обробку метеорологічних даних**, отриманих від цих датчиків. Останні виміряні значення метеовеличин потім виводяться на **алфавітно-цифровий 16x2 рідкокристалічний дисплей HD44780** для зручного відображення користувачу.

Рис.3.8. Структурна схема системи моніторингу фізико-географічних параметрів

Проектування апаратного забезпечення пристрою в Proteus

На **Рис. 3.9** представлено розроблене апаратне забезпечення системи моніторингу в САПР Proteus. Розгляньмо, як підключені ключові компоненти до плати Arduino Uno:

- **Датчик вологості та температури DHT11**: Вивід даних DATA цього цифрового датчика під'єднано до **піна 7 (PD7 = DHT_DATA)** плати Arduino Uno.
- **Датчик тиску та температури BMP180**: Ця мікросхема підключена до Arduino Uno за допомогою шини **I2C**. Виводи A4 (SDA) та A5 (SCL) мікроконтролера Arduino Uno використовуються для цього з'єднання.

- **Датчик температури LM35**: Аналоговий вихід цього датчика підключено до **нульового каналу аналого-цифрового перетворювача (ADC0), тобто до виводу A0**, плати Arduino Uno.
- **Рідкокристалічний дисплей HD44780 (16x2)**: Цей дисплей під'єднано до портів B та D Arduino Uno наступним чином:
    - Вивід RS РКД з'єднано з **піном 12 (PB4)** Arduino Uno.
    - Вивід E РКД підключено до **піна 11 (PB3)** Arduino Uno.
    - Виводи даних D4...D7 РКД з'єднано з **пінами 6...3** плати Arduino Uno відповідно.
- **Клавіатура (4 кнопки)**:
    - Кнопка MENU_BTN підключена до **піна 2 (PD2)**. Її натискання генерує нульове переривання (INT0), що викликає відповідну функцію обробки в програмі.
    - Кнопка S_PLUS_BTN (вибрати/збільшити) з'єднана з **піном 10 (PB2)**.
    - Кнопка S_MINUS_BTN (вибрати/зменшити) підключена до **піна 9 (PB1)**.
    - Кнопка EXIT (для виходу з меню в робочий режим) приєднана до **піна 8 (PB0)** плати Arduino Uno.

Така конфігурація дозволяє мікроконтролеру Arduino Uno зчитувати дані з різних датчиків, виводити їх на дисплей та взаємодіяти з користувачем через кнопки.

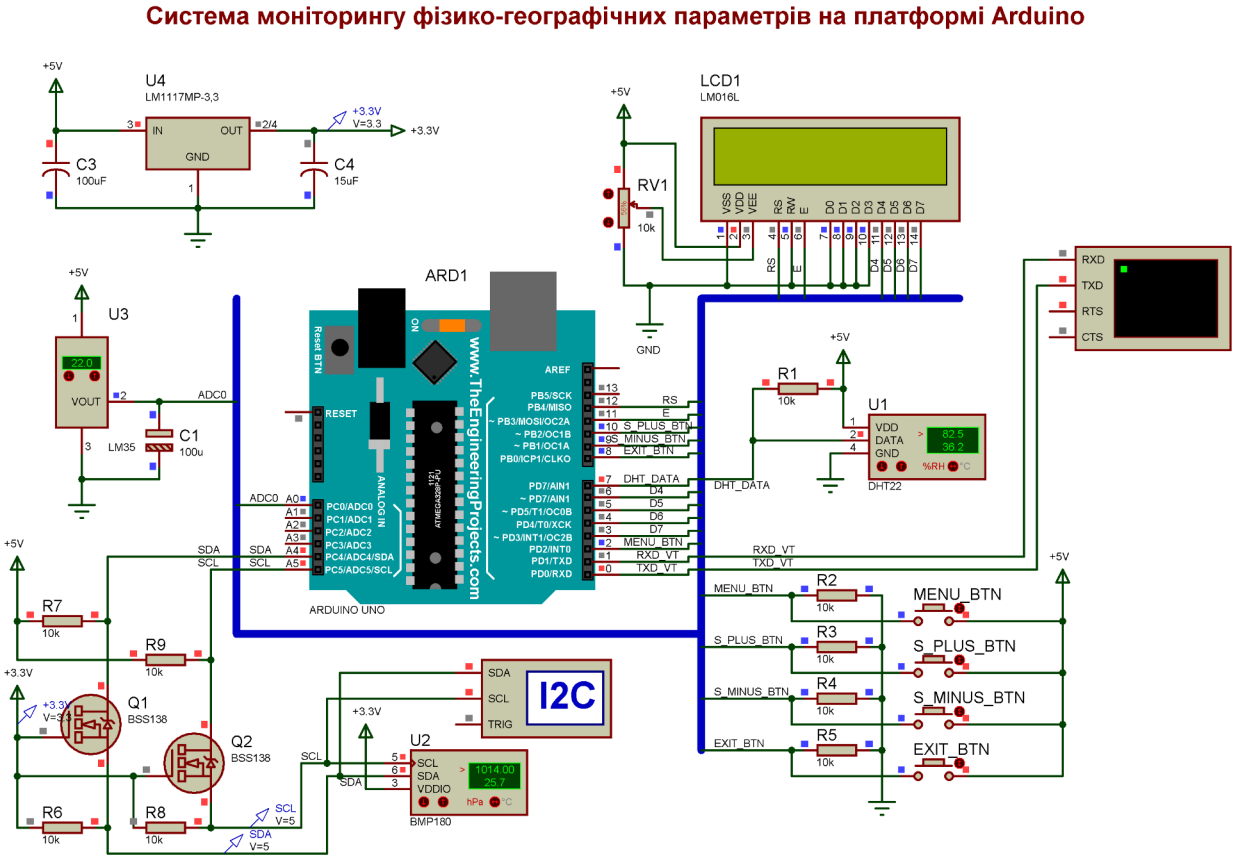


Рис.3.9 Система моніторингу фізико-географічних параметрів спроєктоване в САПР Proteus

РОЗДІЛ IV.
РОЗРОБЛЕННЯ АЛГОРИТМІВ ТА ПРОГРАМ СИСТЕМИ МОНІТОРИНГУ ФІЗИКО-ГЕОГРАФІЧНИХ ПАРАМЕТРІВ

4.1. Алгоритмізація моніторингу фізико-географічних параметрів

Наведемо алгоритмічну процедуру роботи мікроконтролерної системи моніторингу фізико-географічних параметрів.

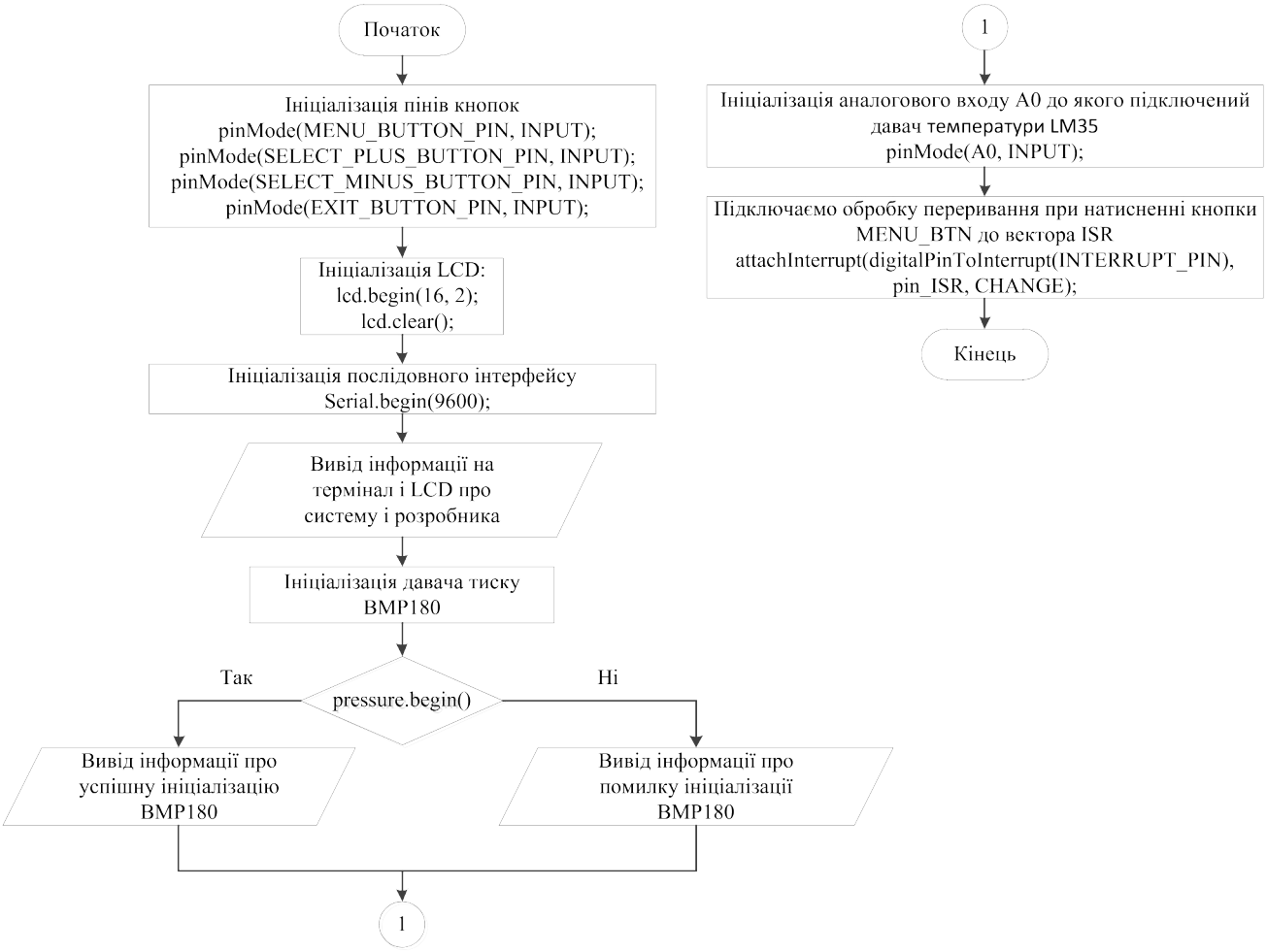


Рис. 4.1. Алгоритмічна процедура моніторингу фізико-географічних параметрів

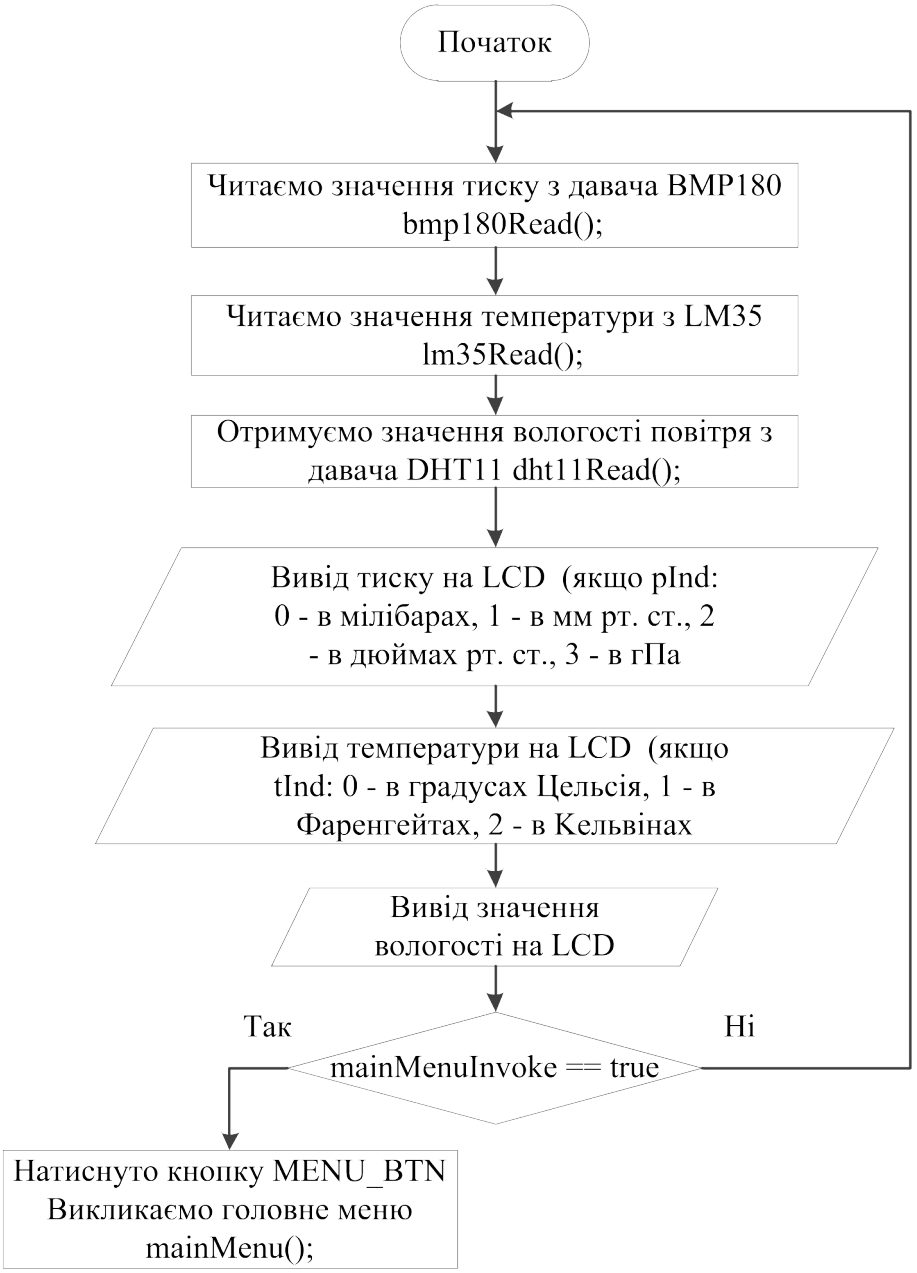


Рис.4.2. Мікроконтролерна система моніторингу фізико-географічних параметрів – головне меню

Після подачі живлення та активації мікроконтролерної системи моніторингу фізико-географічних параметрів, пристрій починає виконувати програму, завантажену у його флеш-пам'ять.

Початкова ініціалізація та інформаційне повідомлення

Насамперед, програма ініціалізує піни, до яких підключені кнопки керування — MENU_BTN, S_PLUS_BTN, S_MINUS_BTN та EXIT_BTN — встановлюючи їх як входи:

```
C++
// ініціалізація пінів кнопок як входи:
pinMode(MENU_BUTTON_PIN, INPUT);
pinMode(SELECT_PLUS_BUTTON_PIN, INPUT);
pinMode(SELECT_MINUS_BUTTON_PIN, INPUT);
pinMode(EXIT_BUTTON_PIN, INPUT);
```

Далі відбувається ініціалізація РК-дисплея та послідовного інтерфейсу. Система виводить на екран та в послідовний порт інформацію про сам пристрій та його розробника:

```
C++
// встановлення кількості стовпців і рядків LCD:
lcd.begin(16, 2);
lcd.clear();
// ініціалізація послідовного інтерфейсу
Serial.begin(9600);
Serial.println(F("Arduino based system"));
Serial.println(F("for monitoring the physical and geographical parameters"));
Serial.println(F("Developed by: Maksym Potsyurko"));
lcd.setCursor(3,0);
lcd.print("MONITORING");
lcd.setCursor(2,1);
lcd.print("SYSTEM, 2025");
delay(2000);
lcd.clear();
lcd.setCursor(1,0);
lcd.print("DEVELOPED BY:");
lcd.setCursor(0,1);
lcd.print("MAKSYM POTSYURKO");
delay(2000);
```

Ініціалізація датчиків та налаштування переривань

Наступним кроком є ініціалізація датчиків, підключених до плати Arduino Uno. Особлива увага приділяється датчику тиску BMP180. Залежно від успішності ініціалізації, на послідовний порт виводиться відповідне повідомлення — про успішне підключення або про помилку.

Також налаштовується функція, яка буде викликатися при виникненні зовнішнього переривання INT0. Це переривання генерується натисканням кнопки MENU_BTN, підключеної до піна INT0. Переривання спрацьовуватиме щоразу, коли змінюється стан цього піна (режим CHANGE), що дозволяє фіксувати як натискання, так і відпускання кнопки.

```
C++
void pin_ISR() {
  //digitalWrite(ledPin, buttonState); // Цей рядок закоментований або не використовується
  // Перевірка, чи натиснута кнопка. Якщо так, то buttonState HIGH:
      if (digitalRead(MENU_BUTTON_PIN) == HIGH) // або isButtonPressed(MENU_BUTTON_PIN)
  {
          detachInterrupt(digitalPinToInterrupt(INTERRUPT_PIN)); // Відключення переривання для уникнення багаторазових спрацьовувань
    mainMenuInvoke = true; // Встановлення прапора для входу в головне меню
  }
}
```

Цикл вимірювань та виведення даних

Далі починається основний цикл програми, де відбувається безперервне вимірювання метеорологічних величин та обробка їхніх значень.

Для зчитування даних з датчиків використовуються такі функції:

BMP180 (тиск і температура):

pressure.getPressure(bmp180Pressure) для отримання значення тиску.

pressure.getTemperature(bmp180Temperature) для отримання значення температури.

LM35 (температура):

Функція val = analogRead(A0) оцифровує аналоговий сигнал температури з датчика LM35, підключеного до каналу ADC0 аналого-цифрового перетворювача (вивід A0).

Оцифровані дані val перетворюються на значення температури в градусах Цельсія за допомогою формули: lm35Temp = (val / 1023.0) * 5.0 * 1000 / 10;.

DHT11 (вологість):

Функція dht11Read() зчитує дані вологості та температури з датчика DHT11 за допомогою chk = DHT.read11(DHT11_PIN). Після зчитування доступні значення DHT.temperature та DHT.humidity.

Отримані значення потім виводяться на РК-дисплей. Температура з BMP180 виводиться за допомогою lcd.print(bmp180Temperature, 1).

Відображення тиску та температури з різних датчиків

Виведення значення тиску залежить від змінної pInd, яка визначає одиниці вимірювання:

```
C++
switch (pInd)
{
  case 0:
    lcd.print(bmp180Pressure,0);
    lcd.print(" мбар");
    break;
  case 1:
                lcd.print(bmp180Pressure*0.0295333727*25.399999705,0);    //
Переведення в мм рт. ст.
    lcd.print(" мм рт");
    break;
```

```
  case 2:
     lcd.print(bmp180Pressure*0.0295333727,1); // Переведення в дюйми рт. ст.
    lcd.print(" дюйм");
    break;
  case 3:
    lcd.print(bmp180Pressure,0);
    lcd.print(" гПа");
    break;
 }
```

Температура, отримана з датчика LM35 (lm35Temp), виводиться залежно від значення змінної tInd, яка визначає бажані одиниці вимірювання:

```
C++
switch (tInd)
{
  case 0:
    lcd.print(lm35Temp,1);
    lcd.print(char(223)); // Символ градуса
    lcd.print("C ");
    break;
  case 1:
    lcd.print((9.0/5.0)*lm35Temp+32.0,1); // Переведення в Фаренгейти
    lcd.print(char(223)); // Символ градуса
    lcd.print("F ");
    break;
  case 2:
    lcd.print(lm35Temp+274.15,1); // Переведення в Кельвіни
    //lcd.print(char(223)); // Символ градуса зазвичай не використовується для Кельвінів
    lcd.print(" K ");
```

```
    break;
}
```

Далі виводиться значення вологості, отриманої з датчика DHT11:

```
C++
lcd.print(dht11Humidity, 1);
lcd.print("% RH");
```

У циклі вимірювання програма постійно відстежує натискання кнопки MENU_BTN. При її активації система переходить у головне меню, де користувач може обирати різні параметри налаштування. Це реалізується за допомогою перевірки прапора mainMenuInvoke, який встановлюється у функції обробки переривання:

```
C++
if (mainMenuInvoke) {
  mainMenu(); // Виклик функції головного меню
}
```

4.2. Програма оброблення інформації від датчика тиску і температури BMP180

Бібліотека bmp180 для роботи з датчиком тиску та температури

Для взаємодії з цифровим датчиком тиску та температури **BMP180** розроблено спеціальну бібліотеку на мові C під назвою bmp180. Вона містить низку функцій, які спрощують роботу з датчиком, дозволяючи зчитувати необроблені дані, калібрувати їх та перетворювати на фізичні величини.

Ось опис основних функцій, що входять до складу цієї бібліотеки:

- **void bmp180Calibration(int16_t BMP180_calibration_int16_t[], int16_t BMP180_calibration_uint16_t[], uint8_t* errorcode)** Ця функція відповідає за **зчитування калібрувальних коефіцієнтів** з пам'яті датчика BMP180. Ці коефіцієнти є унікальними для кожного сенсора і використовуються для точної компенсації вимірювань температури та

тиску. Результат операції (успіх або помилка) повертається через вказівник errorcode.

- **uint16_t bmp180ReadShort(uint8_t address, uint8_t* errorcode)** Ця функція дозволяє **зчитувати 16-бітне значення** з певного регістра датчика за заданою адресою. Це базова функція для доступу до внутрішньої пам'яті та регістрів BMP180.
- **int32_t bmp180ReadTemp(uint8_t* error_code)** Функція призначена для **зчитування необробленого значення температури** безпосередньо з датчика BMP180. Помилка операції також відслідковується через error_code.
- **int32_t bmp180ReadPressure(uint8_t* errorcode)** Ця функція виконує **зчитування необробленого значення тиску** з датчика BMP180. Як і для температури, статус помилки повертається через errorcode.
- **void bmp180Convert(int16_t BMP180_calibration_int16_t[], int16_t BMP180_calibration_uint16_t[], int32_t* temperature, int32_t* pressure, uint8_t* error_code)** Це ключова функція, яка використовує зчитані калібрувальні коефіцієнти та необроблені дані для **перетворення їх у фізичні одиниці**: температуру в градусах Цельсія (°C) та тиск у гектопаскалях (гПа). Результати записуються у вказівники temperature та pressure.
- **int32_t bmp180CalcAltitude(int32_t pressure)** Ця функція дозволяє **обчислити висоту над рівнем моря** на основі виміряного значення тиску, використовуючи відповідну барометричну формулу.

Ці функції надають розробникам зручний інтерфейс для інтеграції датчика BMP180 у свої мікроконтролерні проекти, забезпечуючи точний збір та обробку даних про температуру та тиск.

4.3. Програма обробки інформації про температуру з аналового пристрою LM35

Функція lm35Read() для аналогового датчика температури LM35

Для роботи з аналоговим датчиком температури LM35 [18] була розроблена функція lm35Read(). Вона відповідає за зчитування аналогового сигналу з датчика та перетворення його на значення температури в градусах Цельсія. Ось її реалізація:

C++

```
void lm35Read() {
  int val;
  // Зчитуємо аналоговий сигнал з піна A0.
  // Отримане значення знаходиться в діапазоні від 0 до 1023 (для 10-бітного АЦП).
  val = analogRead(A0);

  // Обчислюємо температуру в градусах Цельсія за формулою.
  // (val / 1023.0) - нормалізація значення до діапазону 0-1.
  // * 5.0 - множимо на опорну напругу АЦП (зазвичай 5В).
  // * 1000 - перетворюємо вольти в мілівольти (LM35 видає 10мВ на градус Цельсія).
  // / 10 - ділимо на 10 мВ/°C, щоб отримати температуру в градусах Цельсія.
  lm35Temp = (val / 1023.0) * 5.0 * 1000 / 10;

  Serial.println(); // Новий рядок для кращого форматування виводу
  Serial.println(F("*** Давач температури LM35 ***"));
   //Serial.println(F("*** Temperature sensor LM35 ***")); // Закоментована англійська версія
  Serial.print("температура: "); // Виводимо текстовий опис
  //Serial.print("temperature: "); // Закоментована англійська версія
   Serial.print(lm35Temp); // Виводимо обчислене значення температури на термінал
  Serial.println(F(" градусів C"));
  // Serial.println(F(" degC")); // Закоментована англійська версія
```

}

Ця функція спочатку отримує сире аналогове значення з датчика LM35 через пін A0. Потім це значення перетворюється у температуру в градусах Цельсія, враховуючи, що LM35 видає 10 мВ на кожен градус Цельсія. Результат виводиться у послідовний порт для налагодження або моніторингу.

4.4. Програма отримання та обробки інформації про вологість і температуру від DHT11

Бібліотека dht11 для роботи з цифровим датчиком вологості та температури
Для ефективної взаємодії з цифровим датчиком вологості та температури DHT11 [15, 16, 17] розроблено спеціальну бібліотеку під назвою dht11. Вона надає набір функцій, які спрощують процес ініціалізації датчика та зчитування його показань.

Ось перелік ключових функцій, що входять до складу цієї бібліотеки:

void dht_init(struct dht11 *dht, uint8_t pin)
Ця функція призначена для ініціалізації датчика DHT11. Вона приймає вказівник на структуру dht11 для зберігання внутрішніх даних датчика та номер піна (виводу) мікроконтролера, до якого підключено датчик.

uint8_t dht_read_temp(struct dht11 *dht, float *temp)
Функція для зчитування значення температури з датчика DHT11. Вона повертає статус операції (наприклад, успіх або помилка) і записує отримане значення температури у змінну, на яку вказує *temp.

uint8_t dht_read_hum(struct dht11 *dht, float *hum)

Ця функція дозволяє зчитувати значення вологості з датчика DHT11. Вона також повертає статус операції та записує отримане значення вологості у змінну, на яку вказує *hum.

uint8_t dht_read_data(struct dht11 *dht, float *temp, float *hum)

Це комплексна функція для одночасного зчитування як температури, так і вологості з датчика DHT11. Вона повертає статус операції, а отримані значення температури та вологості записуються у відповідні змінні за вказівниками *temp та *hum.

Ці функції дозволяють легко інтегрувати DHT11 у ваш проект, забезпечуючи надійне отримання даних про навколишнє середовище.

4.5. Програма оброблення алгоритму роботи мікроконтролерної системи моніторингу фізико-географічних параметрів

У поданому фрагменті коду void loop() знаходиться основний цикл роботи системи моніторингу фізико-географічних параметрів. Цей цикл виконується нескінченно, забезпечуючи безперервний збір даних, їх обробку та відображення.

Огляд Функції loop()

Функція loop() виконує такі ключові дії:

1. **Зчитування даних з датчиків**:
    - **BMP180 (тиск і температура)**: Викликається функція bmp180Read() для отримання показників тиску та температури з датчика BMP180.
    - **LM35 (температура)**: Викликається функція lm35Read() для зчитування температури з аналогового датчика LM35.
    - **DHT11 (вологість і температура)**: Викликається функція dht11Read() для отримання даних вологості та температури з цифрового датчика DHT11.

2. **Відображення даних на РК-дисплеї**:
    - **Температура з BMP180**: Першим рядком на дисплеї (позиція (0,0)) виводиться температура, отримана з датчика BMP180, у градусах Цельсія (з одним знаком після коми).

C++
```
lcd.setCursor(0,0);
lcd.print(bmp180Temperature,1);
lcd.print(char(223)); // Вивід символу градуса
lcd.print("C ");
```

    - **Тиск з BMP180**: Далі в тому ж першому рядку виводиться значення атмосферного тиску, також отримане з BMP180. Одиниці вимірювання (мБар, мм рт. ст., дюйми рт. ст. або гПа) обираються залежно від значення змінної pInd за допомогою оператора switch:

C++
```
switch (pInd) {
  case 0:
    lcd.print(bmp180Pressure,0); lcd.print(" мбар"); break;
  case 1:
    lcd.print(bmp180Pressure*0.0295333727*25.399999705,0); lcd.print(" мм рт");
break;
  case 2:
    lcd.print(bmp180Pressure*0.0295333727,1); lcd.print(" дюйм"); break;
  case 3:
    lcd.print(bmp180Pressure,0); lcd.print(" гПа"); break;
}
```

    - **Температура з LM35**: У другому рядку дисплея (позиція (0,1)) відображається температура з датчика LM35. Одиниці вимірювання (Цельсій, Фаренгейт або Кельвін) залежать від значення змінної tInd:

```
C++
lcd.setCursor(0,1);
switch (tInd) {
  case 0:
    lcd.print(lm35Temp,1); lcd.print(char(223)); lcd.print("C "); break;
  case 1:
    lcd.print((9.0/5.0)*lm35Temp+32.0,1); lcd.print(char(223)); lcd.print("F ");
break;
  case 2:
    lcd.print(lm35Temp+274.15,1); lcd.print(" K "); break;
}
```

- **Вологість з DHT11**: У другому рядку, поруч із температурою від LM35, виводиться значення відносної вологості, отримане з датчика DHT11:

```
C++
lcd.print(dht11Humidity, 1); lcd.print("% RH");
```

3. **Обробка входу в головне меню**:
   - Програма перевіряє значення прапора mainMenuInvoke. Якщо цей прапор істинний (що встановлюється під час спрацьовування зовнішнього переривання від кнопки MENU_BTN), то викликається функція mainMenu(), яка відповідає за навігацію та налаштування в головному меню.
4. **Затримка**:
   - В кінці кожного циклу передбачена затримка в 1000 мілісекунд (1 секунда) за допомогою функції delay(1000). Це регулює частоту оновлення даних на дисплеї та зчитування показань датчиків, забезпечуючи стабільну роботу системи.

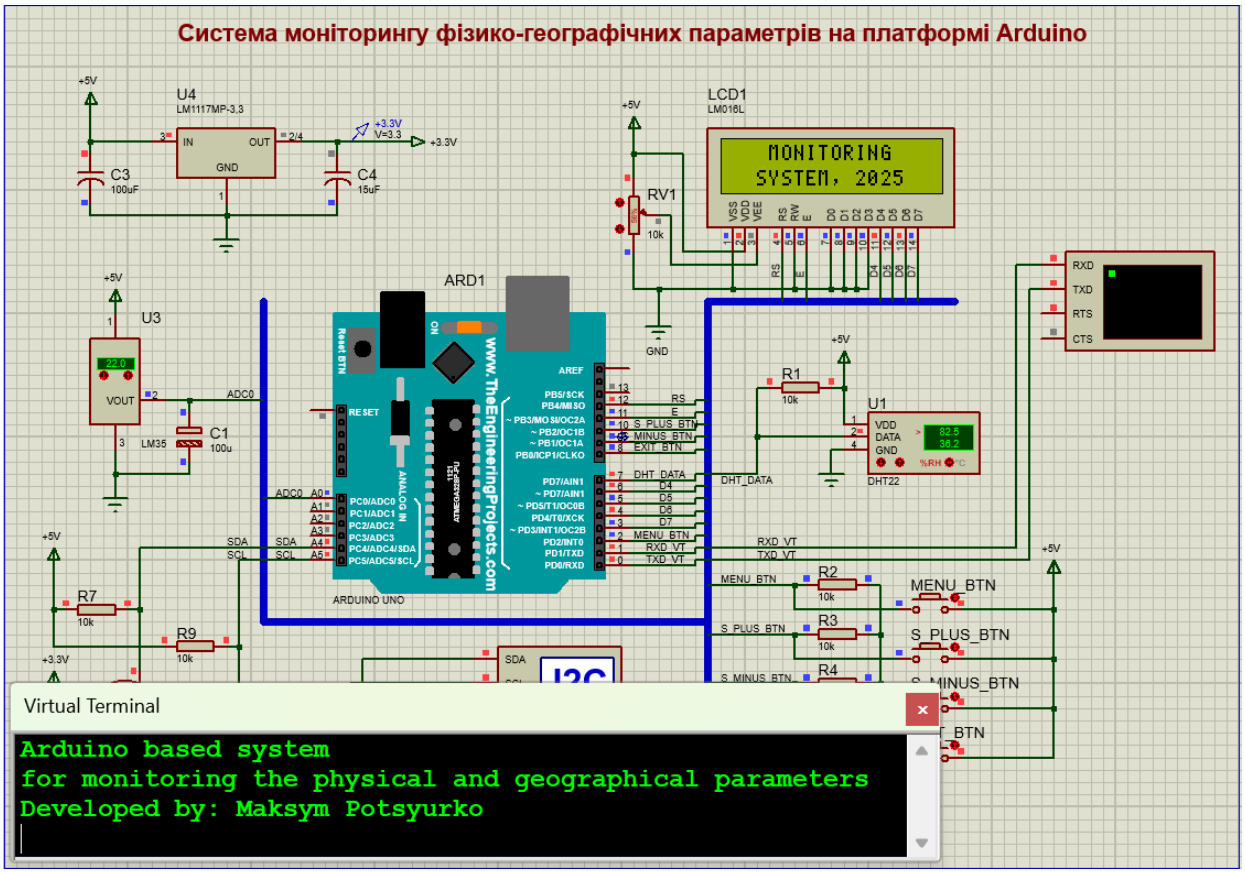
Таким чином, функція loop() забезпечує безперервний моніторинг погодних параметрів, їх обробку та відображення на РК-дисплеї, а також дозволяє користувачеві взаємодіяти з системою через меню.

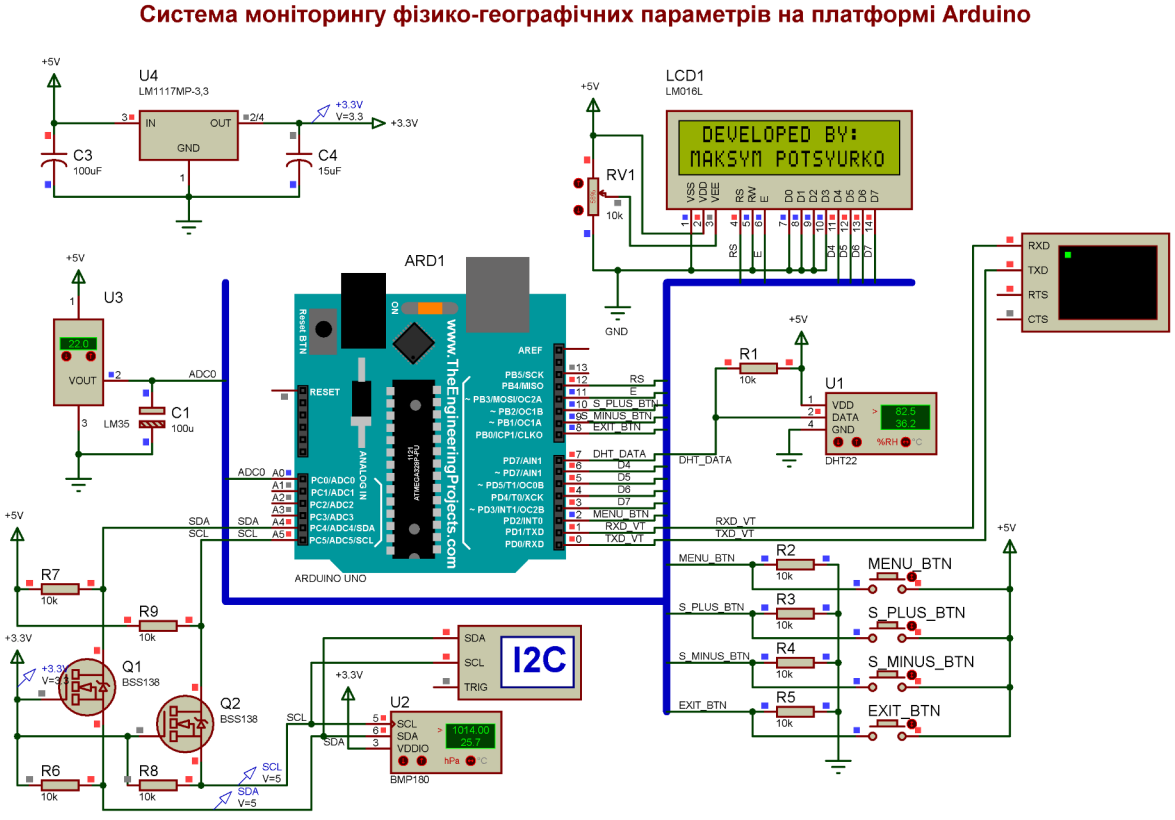
4.6. Робота мікроконтролерної системи моніторингу фізико-географічних параметрів в Proteus ISIS

Після завершення розробки програмного забезпечення, воно компілюється та лінкується у файл прошивки, призначений для мікроконтролера платформи Arduino. Щоб перевірити коректність роботи розробленого ПЗ та змоделювати функціонування пристрою, цей файл прошивки завантажується у віртуальну модель пристрою, створену в Proteus ISIS.

Proteus є потужним інструментом, що дозволяє моделювати та налагоджувати доволі складні пристрої. Його можливості охоплюють симуляцію систем, які можуть містити одночасно декілька мікроконтролерів, навіть різних сімейств.

На рисунках, представлених нижче, а також у детальному звіті в Додатку 3, наведено приклади моделювання мікроконтролерної системи моніторингу фізико-географічних параметрів у середовищі Proteus. Ці візуалізації демонструють, як апаратна частина та програмне забезпечення взаємодіють у віртуальному середовищі, дозволяючи всебічно протестувати систему перед її реальним впровадженням.


Система моніторингу фізико-географічних параметрів на платформі Arduino
MONITORING
SYSTEM, 2025
Virtual Terminal
Arduino based system
for monitoring the physical and geographical parameters
Developed by: Maksym Potsyurko


Система моніторингу фізико-географічних параметрів на платформі Arduino
DEVELOPED BY:
MAKSYM POTSYURKO

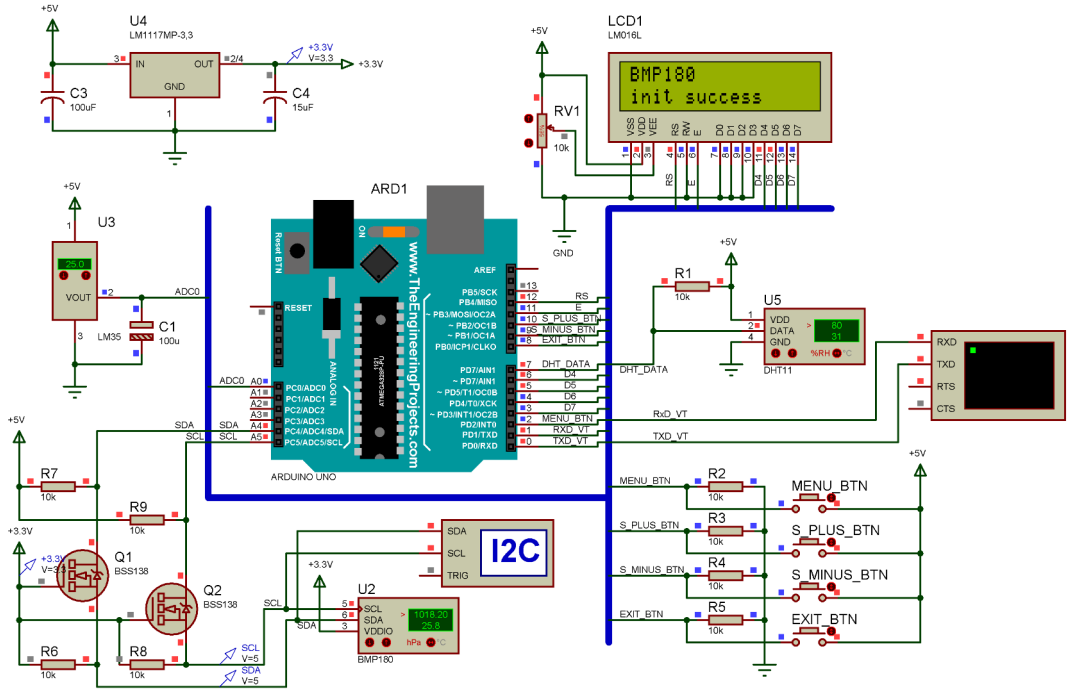


Рис.4.3. Система моніторингу фізико-географічних параметрів в Proteus ISIS

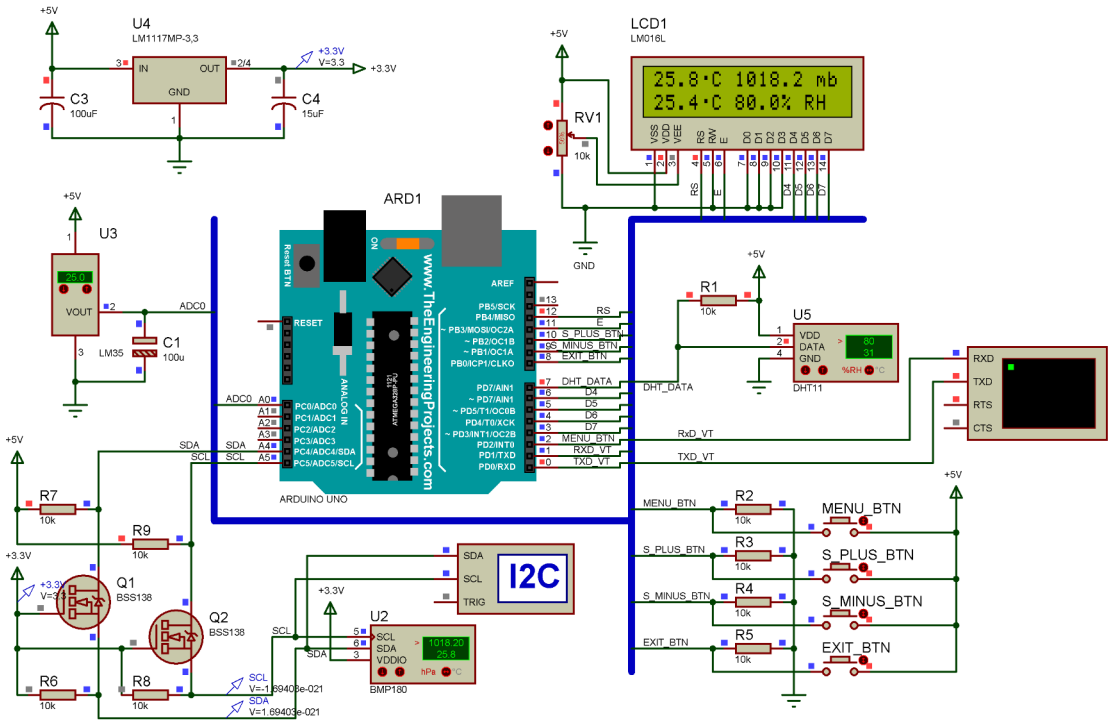
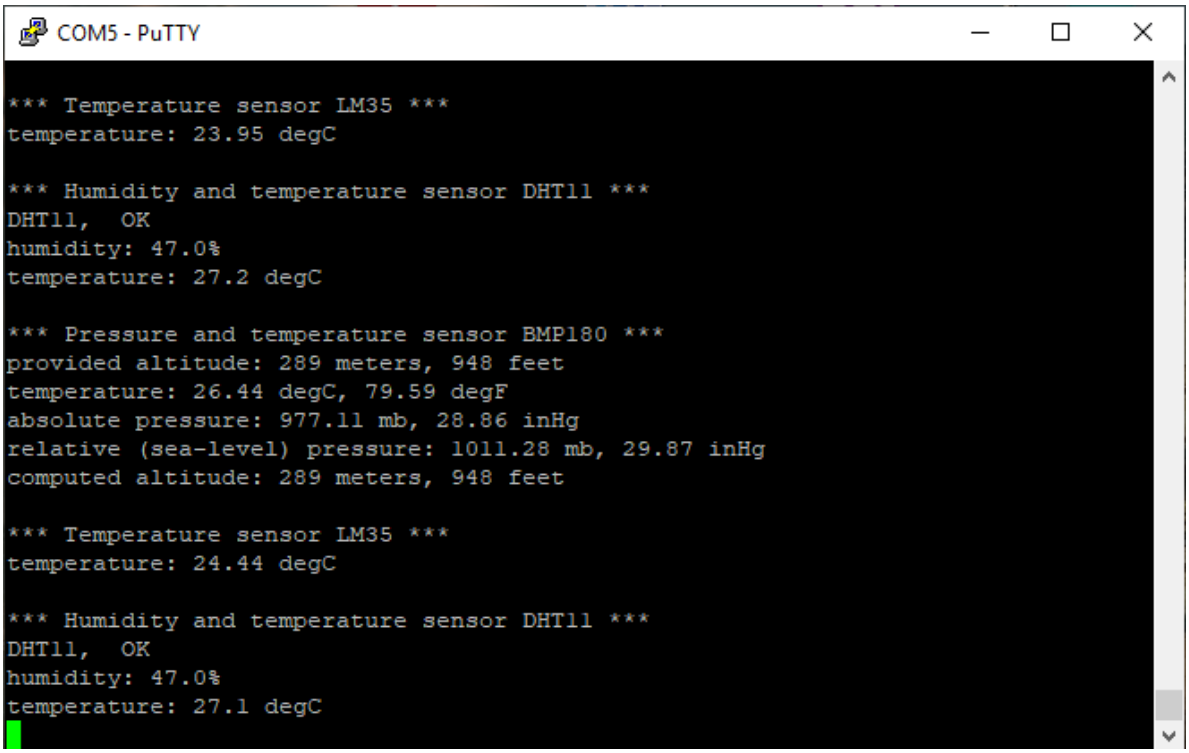


Рис.4.4. Система моніторингу фізико-географічних параметрів в емуляторі Proteus ISIS



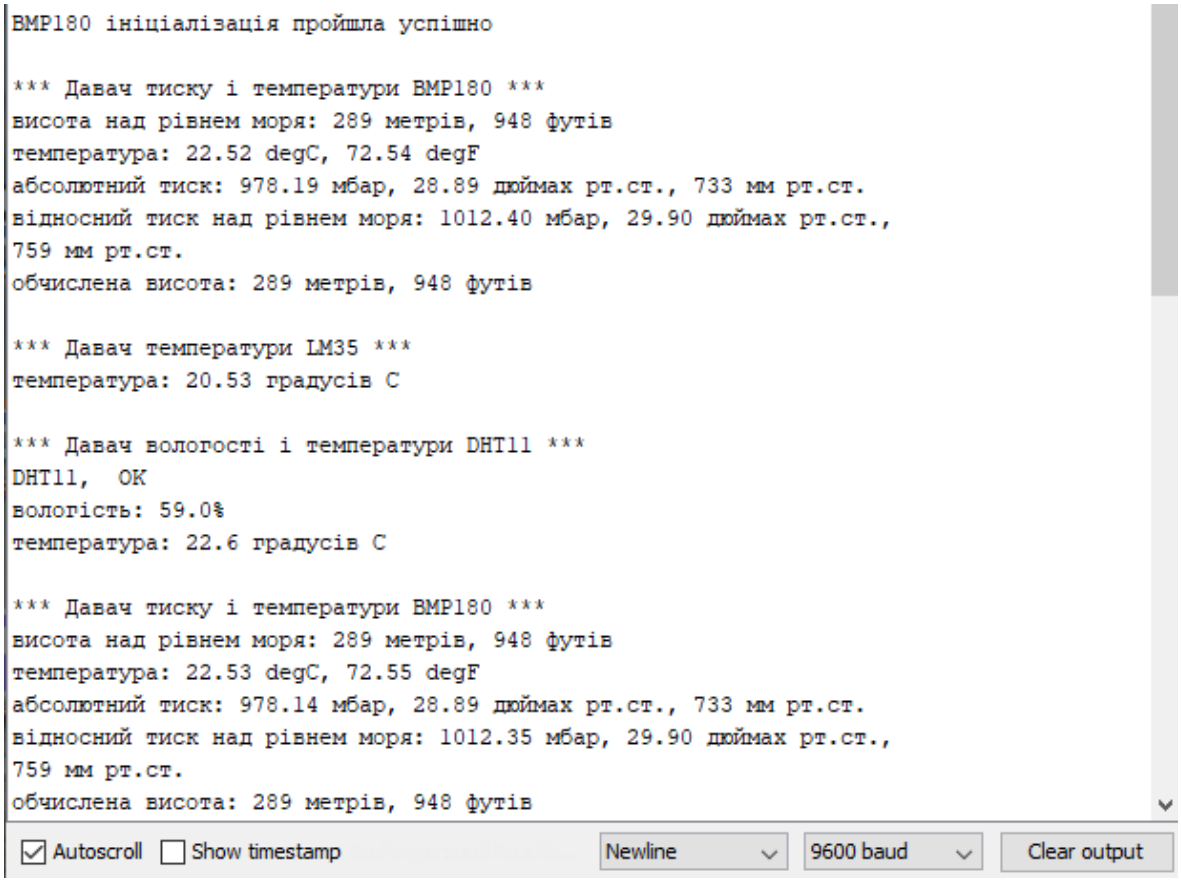


Рис.4.5. Іінформація на моніторі у послідовноу інтерфейсі

ВИСНОВКИ

У цій кваліфікаційній роботі було успішно розроблено як **апаратне, так і програмне забезпечення мікроконтролерної системи моніторингу фізико-географічних параметрів**. Створений пристрій здатний вимірювати **температуру, відносну вологість повітря та атмосферний тиск**, а також відображати отримані метеорологічні дані на **символьному РК-дисплеї 16x2**. Додатково, мікроконтролерний пристрій оснащений **меню налаштувань**, що дозволяє користувачу взаємодіяти з ним.

Деталі Реалізації

Система моніторингу фізико-географічних параметрів базується на:

- **Платформі Arduino Uno R3** з мікроконтролером **ATmega328P-PU**.
- **Аналоговому датчику температури LM35**.
- **Цифровому датчику тиску BMP180**.
- **Цифровому датчику вологості DHT11**.
- **Символьному рідкокристалічному дисплеї на контролері HD44780**.

Етапи Розробки та Тестування

У процесі роботи було виконано наступні ключові етапи:

- Розроблено **електричну принципову схему** та **модель мікроконтролерної системи** в середовищі **Proteus VSM**.
- Створено **алгоритм роботи** та **програмне забезпечення** для системи моніторингу в середовищі **Arduino IDE**, використовуючи мову програмування **C**.
- Розроблено спеціалізовані **програмні модулі** для взаємодії з датчиками BMP180, DHT11, LM35, а також модуль для виведення інформації на РК-дисплей HD44780.
- Виконано **моделювання роботи** мікроконтролерної системи в емуляторі **Proteus ISIS**.

- Зібрано **прототип** мікроконтролерної системи на безпаєчній макетній платі та проведено його **тестування**.

Результати та Висновки

Результати моделювання та фізичного тестування мікроконтролерної системи моніторингу фізико-географічних параметрів підтвердили **коректність роботи** розробленого апаратного та програмного забезпечення. Це свідчить про успішне досягнення поставлених цілей дипломної роботи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Jonathan Bartlett. Electronics for Beginners: A Practical Introduction to Schematics, Circuits, and Microcontrollers. – Published by Apress, 1st ed., 2020. – p. 537.
2. Massimo Banzi, Michael Shiloh. Getting Started With Arduino: The Open Source Electronics Prototyping Platform. – Published by Make Community, LLC; 4th ed., 2022. – p. 285.
3. Jody Culkin, Eric Hagan. Learn Electronics with Arduino: An Illustrated Beginner's Guide to Physical Computing (Make: Technology on Your Time). – Published by Make Community, LLC; 1st ed., 2017. – p. 374.
4. Jonathan Oxer and Hugh Blemings. Practical Arduino: Cool Projects for Open Source Hardware. - Springer-Verlag New York, 2009. – p. 445.
5. Massimo Banzi. Getting Started with Arduino. – 2nd Edition. O'Really Media Inc., 2011. – p. 130.
6. Brian Evans. Beginning Arduino Programming. Writing Code for the Most Popular Microcontroller Board in the World. – Springer Science + Business Media, 2011. – p. 271.
7. Michael Margolis. Arduino Cookbook, Second Edition. – Published by O'Reilly Media Inc., 2012. – p. 724.
8. Enrique Ramos Melgar and Ciriaco Castro Diez with Przemek Jawosrki. Arduino and Kinect Projects. Design, Build, Blow Their Minds. – Springer Sceince+Business Media New York, 2012. – p. 411.
9. Brian Huang and Derek Runberg. The Arduino Inventor's Guide. – SparkFun Electronics, 2017. – p. 472.
10. Emily Gertz & Patrick Di Justo. Environmental Monitoring with Arduino. 32. Building Simple Devices to Collect Data About the World Around Us. – Published by O'Really Media Inc., 2012 – p. 96.

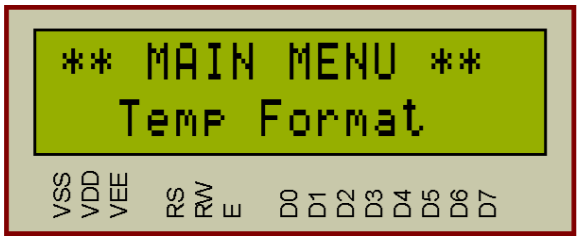
11. Sarful Hassan. Arduino Programming Essentials: A Practical Handbook for Makers, Engineers, and Students. – Independently published, 2024. – p. 308.
12. Середовище розробки Arduino IDE. Електронний ресурс: https://www.arduino.cc/en/Main/Software.
13. Електронний ресурс: https://www.bosch-sensortec.com/bst/products/all_products/bmp180, http://www.santy.cz/data/items/179_44.pdf.
14. Електронний ресурс datasheet по давачу тиску BMP180: https://cdn-shop.adafruit.com/datasheets/BST-BMP180-DS000-09.pdf.
15. Електронний ресурс по давачу вологості DHT11: https://www.mouser.com/ds/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf.
16. Електронний ресурс по давачу вологості DHT22: https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/SEN0137_Web.pdf.
17. Електронний ресурс: http://www.produktinfo.conrad.com/datenblaetter/1600000-1699999/001616244-an-01-en-TEMP__SENS__MOD__DHT_22_ST1173.pdf.
18. Електронний ресурс datasheet по давачу температури LM35: http://www.ti.com/lit/ds/symlink/lm35.pdf.
19. Електронний ресурс по давачу температури LM35: http://www.avr-tutorials.com/projects/atmega16-microcontroller-digital-lcd-thermometer. https://circuitdigest.com/microcontroller-projects/avr-microcontroller-lm35-temperature-sensor-based-digital-thermometer.
20. Електронний ресурс по давачу температури LM35: https://www.electronicwings.com/avr-atmega/lm35-temperature-sensor-interfacing-with-atmega1632.

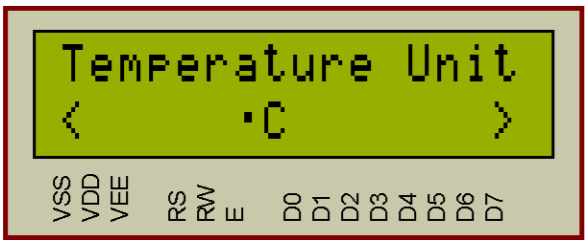
21. Sarful Hassan, Mehadi Hasan Maruf. Arduino Programming for Absolute Beginners: Master Arduino from Scratch with Easy-to-Follow Projects and Examples. – Published by Kindle Edition, 2024. – p. 478.
22. Norman Dunbar. Arduino Software Internals: A Complete Guide to How Your Arduino Language and Hardware Work Together (Maker Innovations Series). – Published by Apress (Kindle Edition), 2nd ed., 2024. – p. 717.
23. Benjamin Spahic. Arduino Without Prior Knowledge: Create your own first project within 7 days (Become an Engineer Without Prior Knowledge). – Kindle Edition, 2020. – p. 113.
24. Jack Purdum. Beginning C for Arduino, Second Edition: Learn C Programming for the Arduino. – Published by Apress; 2nd ed., 2015. – p. 414.
25. Simon Monk. Programming Arduino Next Steps: Going Further with Sketches, Second Edition. – Published by McGraw Hill TAB; 2nd ed., 2018. – p. 320.
26. Blum Richard. Arduino Programming in 24 Hours, Sams Teach Yourself. – Sams Publishing; 1st ed., 2014. – p. 434.

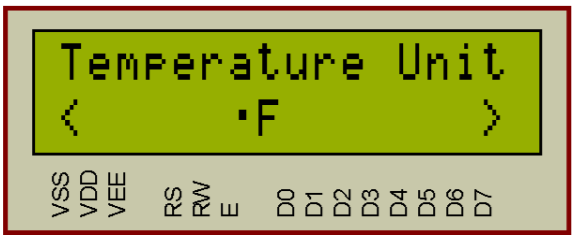
ДОДАТКИ

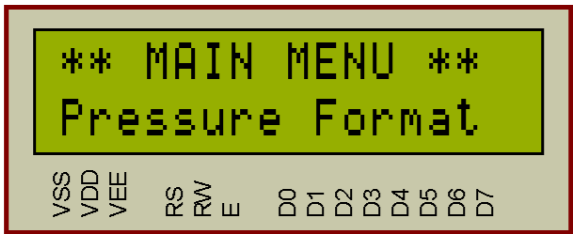
Додаток 1

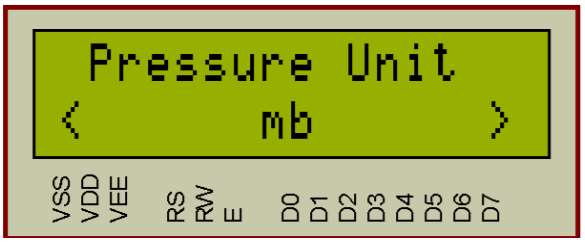
Результати моделювання та аналіз результатів роботи системи моніторингу фізико-географічних параметрів (робочий режим, меню налаштувань, встановлення формату представлення метеовеличин)

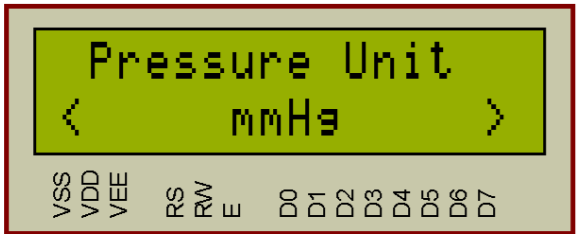


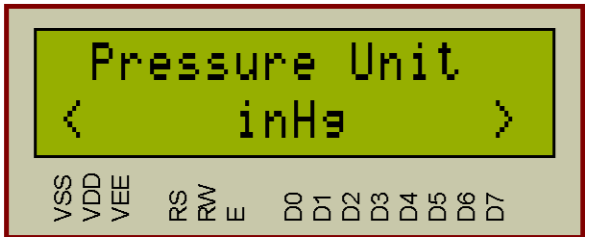


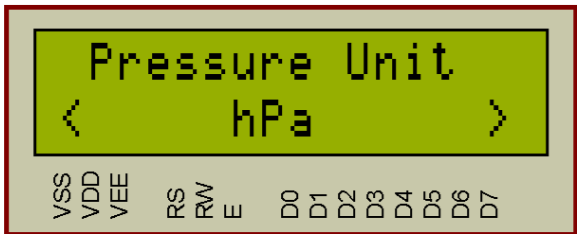


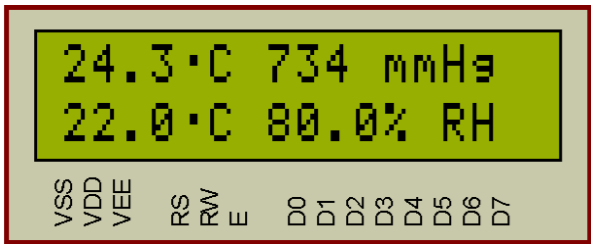


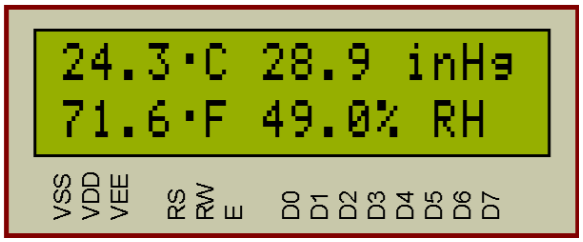






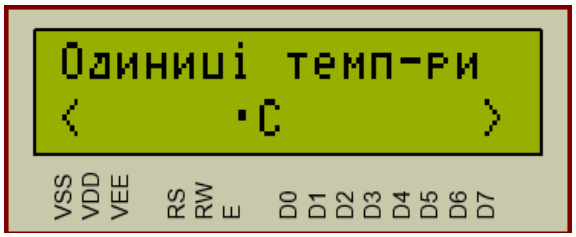


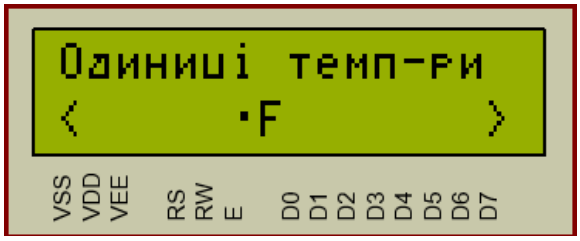


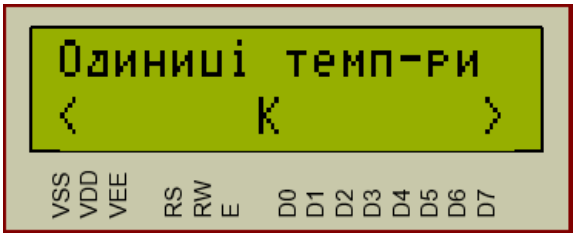

24.3·C 28.9 inHg
71.6·F 49.0% RH
VSS VDD VEE RS RW E D0 D1 D2 D3 D4 D5 D6 D7


** MAIN MENU **
Exit
VSS VDD VEE RS RW E D0 D1 D2 D3 D4 D5 D6 D7

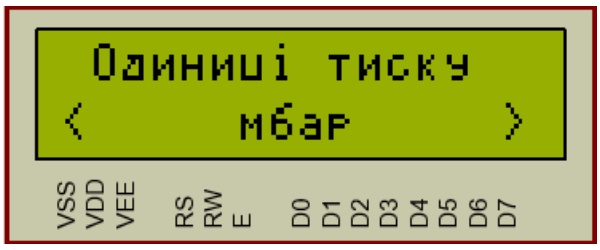

* ГОЛОВНЕ МЕНЮ *
Формат темп-ри
VSS VDD VEE RS RW E D0 D1 D2 D3 D4 D5 D6 D7

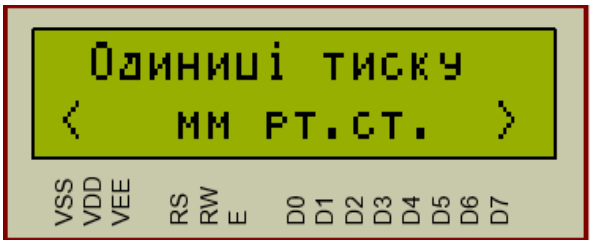

Одиниці темп-ри
< ·C >
VSS VDD VEE RS RW E D0 D1 D2 D3 D4 D5 D6 D7

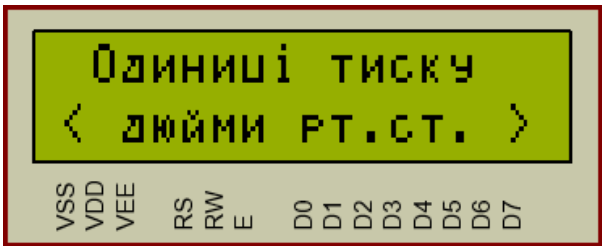

Одиниці темп-ри
< ·F >
VSS VDD VEE RS RW E D0 D1 D2 D3 D4 D5 D6 D7

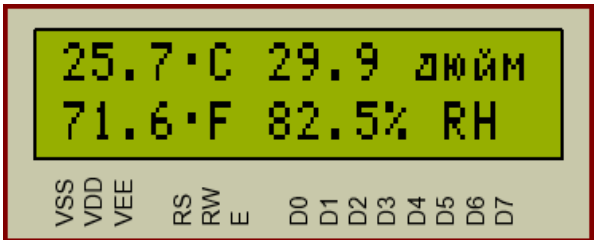

Одиниці темп-ри
< K >
VSS VDD VEE RS RW E D0 D1 D2 D3 D4 D5 D6 D7

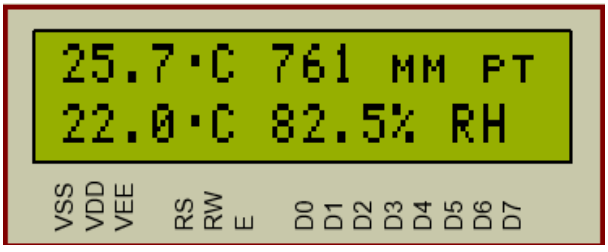

* ГОЛОВНЕ МЕНЮ *
Формат тиску
VSS VDD VEE RS RW E D0 D1 D2 D3 D4 D5 D6 D7

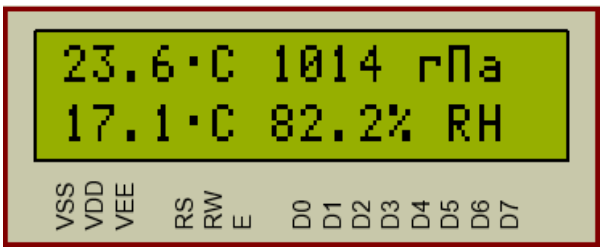

Одиниці тиску
< мбар >
VSS VDD VEE RS RW E D0 D1 D2 D3 D4 D5 D6 D7


Одиниці тиску
< мм рт.ст. >
VSS VDD VEE RS RW E D0 D1 D2 D3 D4 D5 D6 D7

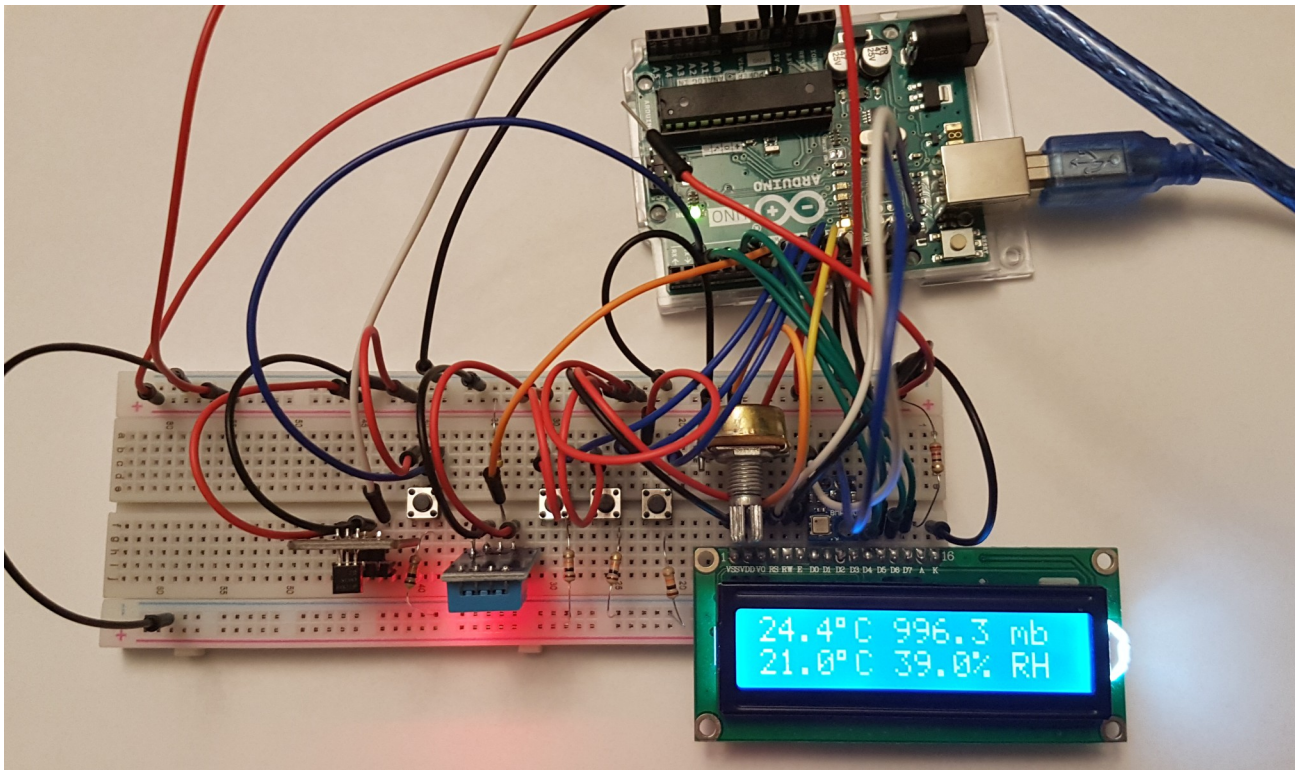

Одиниці тиску
< дюйми рт.ст. >
VSS VDD VEE RS RW E D0 D1 D2 D3 D4 D5 D6 D7

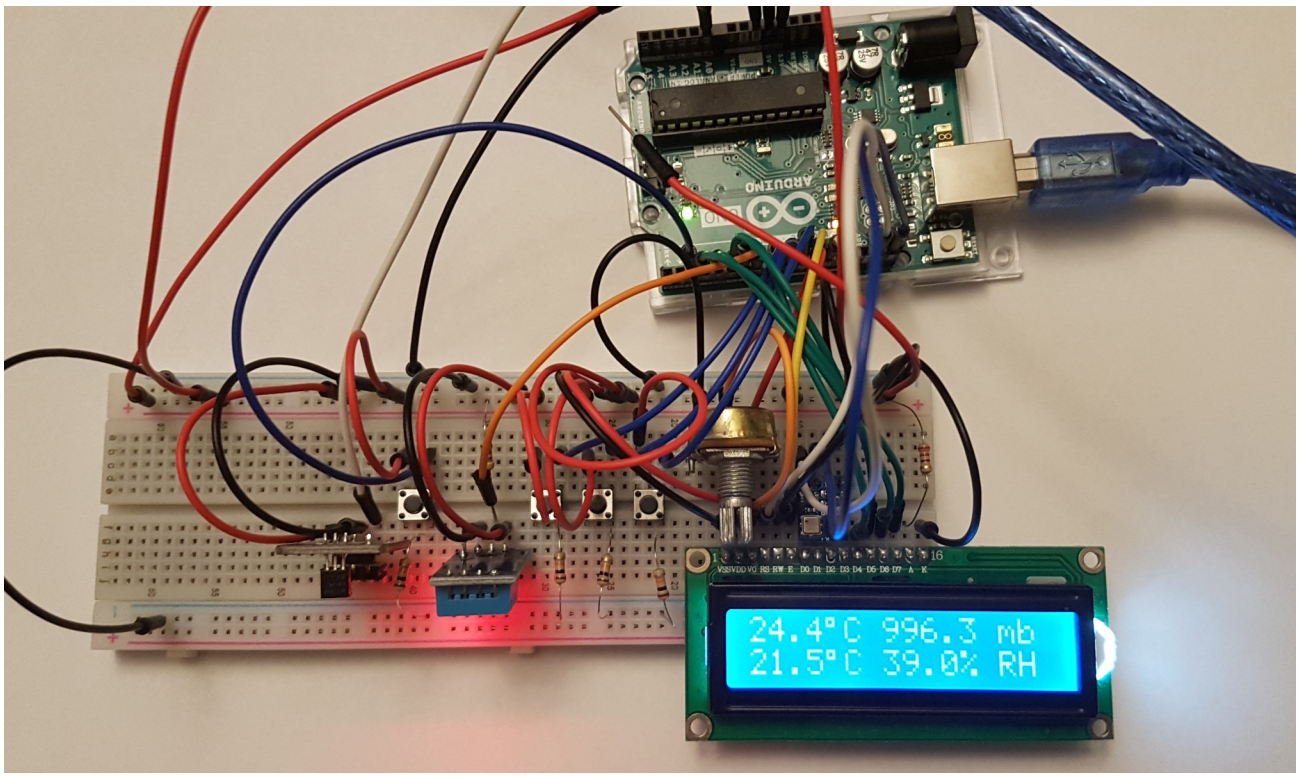

25.7·C 29.9 дюйм
71.6·F 82.5% RH
VSS VDD VEE RS RW E D0 D1 D2 D3 D4 D5 D6 D7

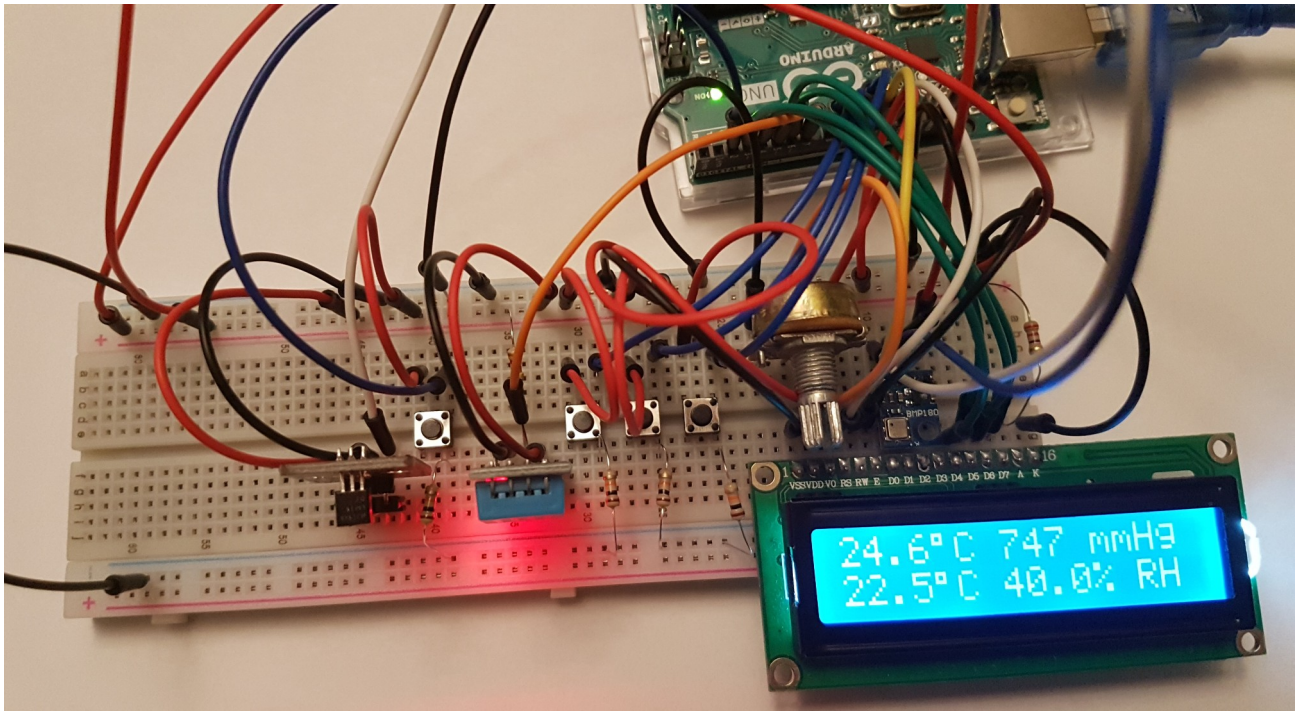

25.7·C 761 мм рт
22.0·C 82.5% RH
VSS VDD VEE RS RW E D0 D1 D2 D3 D4 D5 D6 D7

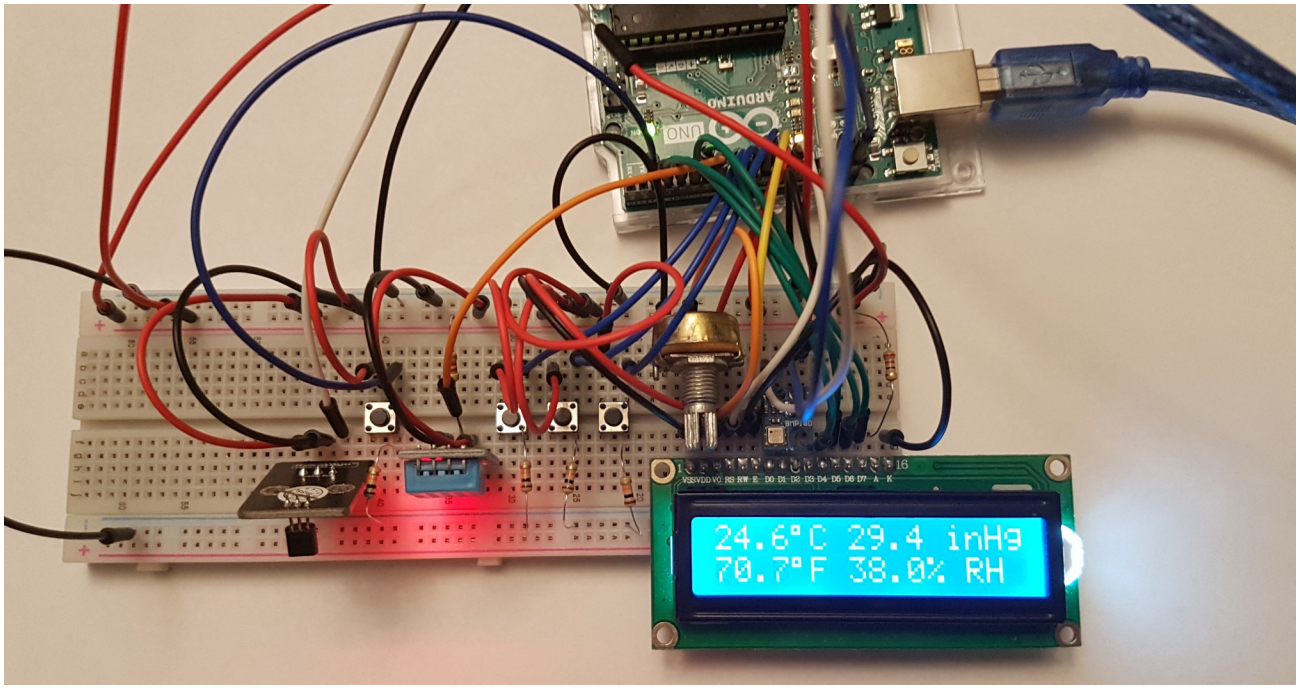

23.6·C 1014 гПа
17.1·C 82.2% RH
VSS VDD VEE RS RW E D0 D1 D2 D3 D4 D5 D6 D7


* ГОЛОВНЕ МЕНЮ *
Вихід
VSS VDD VEE RS RW E D0 D1 D2 D3 D4 D5 D6 D7


24.4°C 996.3 mb
21.0°C 39.0% RH


24.4°C 996.3 mb
21.5°C 39.0% RH


24.6°C 747 mmHg
22.5°C 40.0% RH


24.6°C 29.4 inHg
70.7°F 38.0% RH

Додаток 2

Головний програмний модуль системи моніторингу фізико-географічних параметрів

```
/*
  Мікроконтролерна система моніторингу фізико-географічних параметрів
  (Microcontroller system for monitoring the physical and geographical parameters)
  Розробив: студент 4 курсу, Максим Поцюрко
*/

// бібліотека LCD - рідкокристалічного дисплею
//#include <LiquidCrystal.h>
//#include <LiquidCrystal_1602_RUS.h>
#include <LiquidCrystalRus.h>
#include <SFE_BMP180.h> // бібліотека давача тиску BMP180
#include <Wire.h> // бібліотека для комунікації з пристроями підключиними по шині
I2C/TWI
#include "dht.h"

#define ALTITUDE 289.0 // Середня висота Львова над рівнем моря в метрах

SFE_BMP180 pressure;
double bmp180Temperature, bmp180Pressure;

#define DHT22_PIN 7
dht DHT;
float dht11Temperature, dht11Humidity;

struct
{
    uint32_t total;
    uint32_t ok;
    uint32_t crc_error;
    uint32_t time_out;
    uint32_t connect;
    uint32_t ack_l;
    uint32_t ack_h;
    uint32_t unknown;
} stat = { 0,0,0,0,0,0,0,0};

float lm35Temp;
int pInd =0, tInd = 0;

// ініціалізація бібліотеки номерами інтерфейсних пінів
//LiquidCrystal lcd(12, 11, 6, 5, 4, 3);
//LiquidCrystal_1602_RUS lcd(12, 11, 6, 5, 4, 3);
LiquidCrystalRus lcd(12, 11, 6, 5, 4, 3);

// constants won't change. They're used here to set pin numbers:
/*const int menuButtonPin = 13;     // the number of the pushbutton pin
const int selectPlusButtonPin = 10;
const int selectMinusButtonPin = 9;
const int exitButtonPin = 8;*/

#define MENU_BUTTON_PIN 2
#define SELECT_PLUS_BUTTON_PIN 10
#define SELECT_MINUS_BUTTON_PIN 9
#define EXIT_BUTTON_PIN 8
```

```
#define INTERRUPT_PIN 2

volatile bool mainMenuInvoke = false;

//int buttonState;             // the current reading from the input pin
//int lastButtonState = LOW;   // the previous reading from the input pin

// the following variables are unsigned longs because the time, measured in
// milliseconds, will quickly become a bigger number than can be stored in an int.
//unsigned long lastDebounceTime = 0;  // останній час, коли вихідний пін був
переключений
unsigned long debounceDelay = 50;    // час деренчання; збільшити якщо вихід
деренчить (шумить)

#define NUMBUTTONS 4
unsigned long lastDebounceTime[NUMBUTTONS] = {0, 0, 0, 0};

int lastButtonState[NUMBUTTONS] = {LOW, LOW, LOW, LOW};
int buttonState[NUMBUTTONS];
int buttonsArray[NUMBUTTONS] = {MENU_BUTTON_PIN, SELECT_PLUS_BUTTON_PIN,
SELECT_MINUS_BUTTON_PIN, EXIT_BUTTON_PIN};

// button array pos
int buttonToPos(uint8_t BUTTON)
{
   for (int i=0; i<NUMBUTTONS; i++) {
     if (buttonsArray[i] == BUTTON) {
       return i;
     }
   }
   return -1;
}

bool isButtonPressed(int buttonPin)
{
  // читаємо пін кнопки, якщо кнопка натиснута буде високим, якщо ні буде низьким
  int value = digitalRead(buttonPin);
  int pos = buttonToPos(buttonPin);

  /*  випалку якщо зчитане значення не рівне останьому стану кнопки, то встановити
останній час деренчання (брязкотіння) у поточний час в мілісекундках,
  що пройшоа з початку викоання програми millis() */
  if (value != lastButtonState[pos])  {
    lastDebounceTime[pos] = millis();
  }
  // перевірити різницю між поточним часом і останнім зареєстрованим часом натиснення
кнопки, якшо він більший встановленого часу затримки, то означає що натиснута кнопка
  if ((millis()-lastDebounceTime[pos]) > debounceDelay) {
    if (value != buttonState[pos]) {
      buttonState[pos] = value;
    if (buttonState[pos] == HIGH) {
         return true;
      }
    }
  }
  //зберегти зчитане значення стан кнопки. наступного ращу в циклі зчитаний стан
кнопки буде вже як lastButtonState
  lastButtonState[pos] = value;
  return false;
}
```

```
void setPressureUnit()
{
  String pressureUnits[] = {"    мбар    ", "  мм рт.ст. ", "дюйми рт.ст.", "    гПа
"}; //{"мбар", "мм рт.ст.", "дюйми рт.ст.", "гПа"}; // {" mb ", "mmHg", "inHg", "hPa
"};
  int x[] = {6, 3, 2, 6};
  const char pressureUnitsMenuTitle[] = "Одиниці тиску"; //" Pressure Unit ";
  int ind = 0, p_ind = -1;

    lcd.clear();

  while(1)
  {

  //if (p_ind != ind)
 // {

    lcd.setCursor(1,0);
    lcd.print(pressureUnitsMenuTitle);
    lcd.setCursor(0,1);
    lcd.print("<");
    lcd.setCursor(2,1);
    lcd.print(pressureUnits[ind]);
    lcd.setCursor(15,1);
    lcd.print(">");
  //  p_ind = ind;
  //}

    if (isButtonPressed(SELECT_PLUS_BUTTON_PIN))
      if (ind == 3) ind = 0;
      else ind++;

    if (isButtonPressed(SELECT_MINUS_BUTTON_PIN))
      if (ind == 0) ind = 3;
      else ind--;

    if(isButtonPressed(MENU_BUTTON_PIN))
    {
      pInd = ind;
      return;
    }
    if (isButtonPressed(EXIT_BUTTON_PIN))
      return;
  }
}

void setTemperatureUnit()
{
  const char *temperatureUnits[] = {"C", "F", "K"};
  const char degree[] = {'\xdf', '\xdf', ' '};
  const char temperatureUnitsMenuTitle[] = "Одиниці темп-ри";
  int ind = 0;

  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print(temperatureUnitsMenuTitle);
  lcd.setCursor(0,1);
  lcd.print("<");
  lcd.setCursor(15,1);
  lcd.print(">");
```

```
  while(1)
  {
    lcd.setCursor(6,1);
    lcd.print(degree[ind]);
    lcd.print(temperatureUnits[ind]);

    if (isButtonPressed(SELECT_PLUS_BUTTON_PIN))
      if (ind == 2) ind = 0;
      else ind++;

    if (isButtonPressed(SELECT_MINUS_BUTTON_PIN))
      if (ind == 0) ind = 2;
      else ind--;

    if(isButtonPressed(MENU_BUTTON_PIN))
    {
      tInd = ind;
      return;
    }
    if (isButtonPressed(EXIT_BUTTON_PIN))
      return;
  }
}

void mainMenu()
{
  char *mainMenuItems[] = {" Формат темп-ри ", "  Формат тиску  ", "     Вихід
"}; //{"  Temp Format  ", "Pressure Format ", "      Exit      "}; // {" Формат темп-
ри ", "  Формат тиску  ", "     Вихід      "};
  String menuTitle = "* ГОЛОВНЕ МЕНЮ *"; //= "** MAIN MENU **";
  int i = 0;
  bool lcdPrint = false;

  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print(menuTitle);
  //delay(300);
  lcd.setCursor(0,1);
  lcd.print(mainMenuItems[i]);

  while(1)
  {
   if (lcdPrint)
   {
     lcd.setCursor(0,0);
     lcd.print(menuTitle);
     lcd.setCursor(0,1);
     lcd.print(mainMenuItems[i]);
     lcdPrint = !lcdPrint;
  }
    if (isButtonPressed(SELECT_PLUS_BUTTON_PIN))
    {
      if(i < 2) i++;
      else i = 0;
      lcdPrint = !lcdPrint;
    }
    if (isButtonPressed(SELECT_MINUS_BUTTON_PIN))
    {
      if (i > 0) i--;
      else i = 2;
```

```
      lcdPrint = !lcdPrint;
    }
    if(isButtonPressed(MENU_BUTTON_PIN))
    {
      switch(i)
      {
        case 0: setTemperatureUnit(); break;
        case 1: setPressureUnit(); break;
        case 2: return;
        default: break;
      }
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print(menuTitle);
      lcd.setCursor(0,1);
      lcd.print(mainMenuItems[i]);
    }
    if (isButtonPressed(EXIT_BUTTON_PIN))
    {
      lcd.clear();
      mainMenuInvoke = 0;
      attachInterrupt(digitalPinToInterrupt(INTERRUPT_PIN), pin_ISR, CHANGE);
      return;
    }
  }
}

void setup()
{
  // ініціалізація пінів кнопок як входи:
  pinMode(MENU_BUTTON_PIN, INPUT);
  pinMode(SELECT_PLUS_BUTTON_PIN, INPUT);
  pinMode(SELECT_MINUS_BUTTON_PIN, INPUT);
  pinMode(EXIT_BUTTON_PIN, INPUT);
  // встановлення кількості стовпців і рядків LCD:
  lcd.begin(16, 2);
  lcd.clear();
  // ініціалізація послідовного інтерфейсу
  Serial.begin(9600);
  Serial.println(F("*** Мікроконтролера система моніторингу фізико-географічних
параметрів ***"));
  Serial.println(F("Розробив: студент 4 курсу, Максим Поцюрко"));
  /*lcd.print("  *** WPMS *** ");
  lcd.setCursor(1,1);
  lcd.print("Maksym Potsyurko");*/
  lcd.setCursor(4,0);
  lcd.print(F("СИСТЕМА"));
  lcd.setCursor(0,1);
  lcd.print(F("МЕТЕОМОНІТОРИНГУ"));
  delay(2000);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print(F("2025, Розробив:"));
  lcd.setCursor(0,1);
  lcd.print(F("Максим Поцюрко"));
  delay(2000);
  // ініціалізація давача тиску BMP180 (важливо отримати калібровочні значення
збережені в пристрої).
  if (pressure.begin())
  {
    Serial.println(F("BMP180 ініціалізація пройшла успішно"));
```

```
    //Serial.println(F("BMP180 init success"));
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("BMP180 ініц-я");
    lcd.setCursor(0,1);
    lcd.print("пройшла успішно");
    delay(100);
    //lcd.clear();
  }
  else
  {
    // щось пішло не так, зазвичай проблеми з підключенням
    Serial.println(F("BMP180 ініціалізація пройшла невдало"));
    //Serial.println(F("BMP180 init fail"));
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print(F("BMP180 ініц-я"));
    lcd.setCursor(0,1);
    lcd.print(F("пройшла невдало"));
    while(1); // Pause forever.
  }
  pinMode(A0, INPUT); // сенсор LM35 підключаємо до аналогового входу A0
  // Attach an interrupt to the ISR vector
  attachInterrupt(digitalPinToInterrupt(INTERRUPT_PIN), pin_ISR, CHANGE);
}

void dht11Read()
{
    // читаємо дані темератури і вологості з  dht
    Serial.println();
    Serial.println(F("*** Давач вологості і температури DHT11 ***"));
    //Serial.println(F("*** Humidity and temperature sensor DHT11 ***"));
    Serial.print("DHT22, \t");
    uint32_t start = micros();
    int chk = DHT.read22(DHT22_PIN);
    uint32_t stop = micros();

    stat.total++;
    switch (chk)
    {
    case DHTLIB_OK:
        stat.ok++;
        Serial.println(F("OK"));
        break;
    case DHTLIB_ERROR_CHECKSUM:
        stat.crc_error++;
        //Serial.println(F("Checksum error"));
        Serial.println(F("Помилка контрольної суми"));
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("DHT22");
        lcd.setCursor(0,1);
        lcd.print(F("Помилка кон.суми"));
        //lcd.print(F("Checksum error"));
        break;
    case DHTLIB_ERROR_TIMEOUT:
        stat.time_out++;
        Serial.println(F("Час вийшов"));
        //Serial.println(F("Time out error"));
        lcd.clear();
        lcd.setCursor(0,0);
```

```
        lcd.print("DHT11");
        lcd.setCursor(0,1);
        lcd.print(F("Час вийшов"));
        //lcd.print(F("Time out error"));
        break;
    default:
        stat.unknown++;
        Serial.println(F("Невідома помилка"));
        //Serial.println(F("Unknown error"));
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("DHT11");
        lcd.setCursor(0,1);
        lcd.print(F("Невідома помилка"));
        //lcd.print(F("Unknown error"));
        break;
    }
    // виводимо дані
    Serial.print(F("вологість: "));
    //Serial.print(F("humidity: "));
    Serial.print(DHT.humidity,1);
    Serial.println("%");
    Serial.print("температура: ");
    //Serial.print(F("temperature: "));
    Serial.print(DHT.temperature,1);
    Serial.println(F(" градусів С"));
    //Serial.println(F(" degC"));
    //Serial.print(stop - start);
    //Serial.println();
    dht11Temperature = DHT.temperature; dht11Humidity = DHT.humidity;
    //lcd.print(DHT.humidity, 1);
    //lcd.print(F("% "));
    //lcd.setCursor(0,1);
    //lcd.print(DHT.temperature, 1);
    //lcd.print(char(223));
    //lcd.print(F("C "));

    if (stat.total % 20 == 0)
    {
        Serial.println("\nTOT\tOK\tCRC\tTO\tUNK");
        Serial.print(stat.total);
        Serial.print("\t");
        Serial.print(stat.ok);
        Serial.print("\t");
        Serial.print(stat.crc_error);
        Serial.print("\t");
        Serial.print(stat.time_out);
        Serial.print("\t");
        Serial.print(stat.connect);
        Serial.print("\t");
        Serial.print(stat.ack_l);
        Serial.print("\t");
        Serial.print(stat.ack_h);
        Serial.print("\t");
        Serial.print(stat.unknown);
        Serial.println("\n");
    }
   delay(20);
}

void bmp180Read()
```

```
{
  char status;
  double p0, a;
  int press_mmHg;

  // Loop here getting pressure readings every 10 seconds.
  // If you want sea-level-compensated pressure, as used in weather reports,
  // you will need to know the altitude at which your measurements are taken.
  // We're using a constant called ALTITUDE in this sketch:
  Serial.println();
  Serial.println(F("*** Давач тиску і температури BMP180 ***"));
  //Serial.println(F("*** Pressure and temperature sensor BMP180 ***"));
  Serial.print(F("висота над рівнем моря: "));
  //Serial.print(F("provided altitude: "));
  Serial.print(ALTITUDE,0);
  Serial.print(F(" метрів, "));
  //Serial.print(" meters, ");
  Serial.print(ALTITUDE*3.28084,0);
  Serial.println(F(" футів"));
  //Serial.println(" feet");
  // If you want to measure altitude, and not pressure, you will instead need
  // to provide a known baseline pressure. This is shown at the end of the sketch.
  // You must first get a temperature measurement to perform a pressure reading.
  // Start a temperature measurement:
  // If request is successful, the number of ms to wait is returned.
  // If request is unsuccessful, 0 is returned.
  status = pressure.startTemperature();
  if (status != 0)
  {
    // чекаємо коли завершиться вимірювання:
    delay(status);
    // зчитуємо виміряну температуру:
    // виміряне значення зберігається в змінній bmp180Temperature.
    // функція повертає 1 при успішному виконанні і 0 якщо невдача.

    status = pressure.getTemperature(bmp180Temperature);
    if (status != 0)
    {
      // вивід виміряної температури :
      Serial.print(F("температура: "));
      //Serial.print(F("temperature: "));
      Serial.print(bmp180Temperature, 2);
      Serial.print(" degC, ");
      Serial.print((9.0/5.0)*bmp180Temperature+32.0,2);
      Serial.println(" degF");
      //lcd.setCursor(0,0);
      //lcd.print(T, 1);
      //lcd.print(char(223));
      // старт вимірювання тиску:
      // The parameter is the oversampling setting, від 0 до 3 (найбільша роздільна
здатність, найдовше очікування).
      // якшо запит успішний, число мс очікування буде повернено.
      // якщо запит невдалий, повертається 0.
      status = pressure.startPressure(3);
      if (status != 0)
      {
        // очікуємо на завершення вимірювання:
        delay(status);
        // Retrieve the completed pressure measurement:
        // Note that the measurement is stored in the variable P.
```

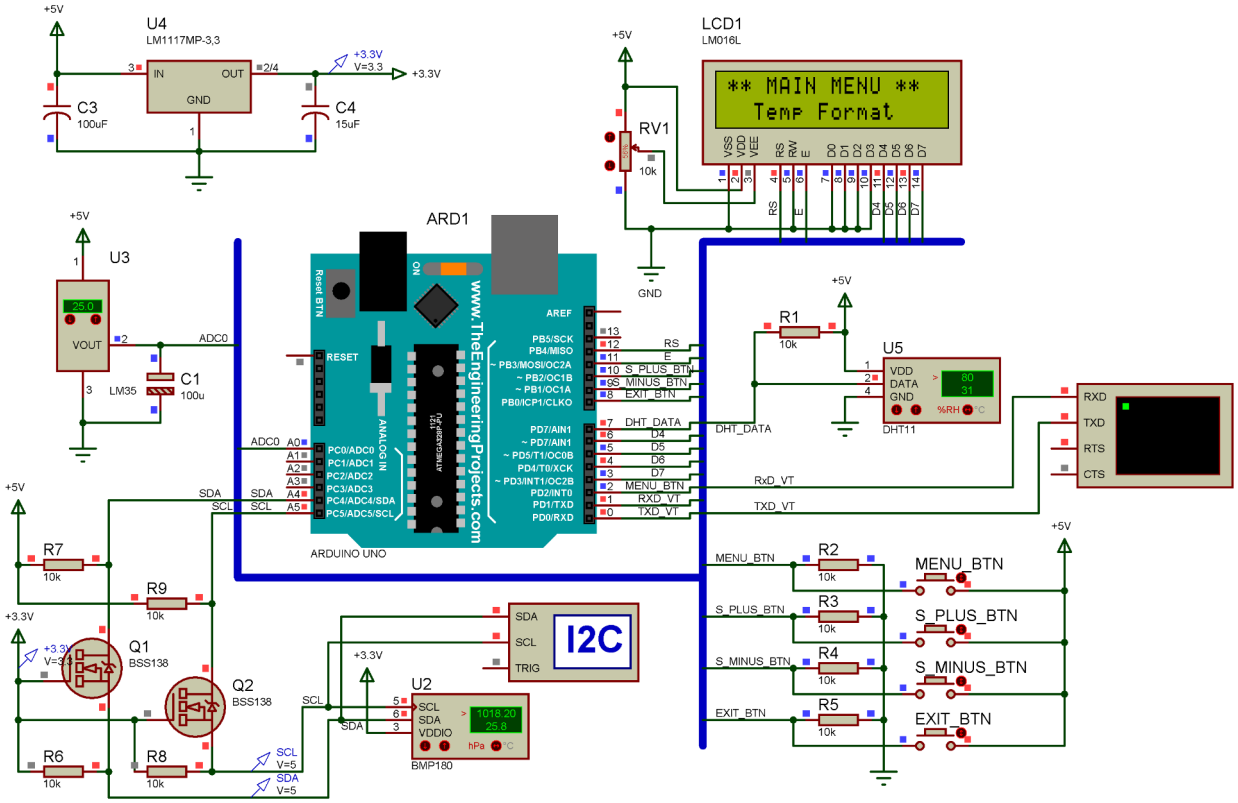
```
        // Note also that the function requires the previous temperature measurement
(T).
        // (If temperature is stable, you can do one temperature measurement for a
number of pressure measurements.)
        // Function returns 1 if successful, 0 if failure.
        status = pressure.getPressure(bmp180Pressure, bmp180Temperature);
        if (status != 0)
        {
          // виводимо виміряне значення тиску:
          Serial.print(F("абсолютний тиск: "));
          //Serial.print(F("absolute pressure: "));
          Serial.print(bmp180Pressure,2);
          Serial.print(F(" мбар, "));
          //Serial.print(F(" mb, "));
          Serial.print(bmp180Pressure*0.0295333727,2);
          Serial.print(F(" дюймах рт.ст., "));
          //Serial.println(F(" inHg"));
          press_mmHg = bmp180Pressure*0.0295333727*25.399999705; // 25.4
          Serial.print(press_mmHg);
          Serial.println(F(" мм рт.ст."));
          //lcd.print(press_mmHg);
          //lcd.print(" mmHg");

          // Сенсор тиску повертає абсолютний тиск, який змінюється з висотою.
          // Щоь усунути ефект висоти, потрібно використати функцію sealevel та
поточну висоту.
          // Це значення зазвичай використовується в погодніх прогнозах
(метеорологічних повідомленнях).
          // Parameters: P = абсолютний тиск в mb, ALTITUDE = поточна висота в m.
          // Result: p0 = тиск скомпенсований над рівнем моря в mb
          p0 = pressure.sealevel(bmp180Pressure,ALTITUDE); // у Львові середня висота
над рівнем моря 289 метра
          Serial.print(F("відносний тиск над рівнем моря: "));
          //Serial.print(F("relative (sea-level) pressure: "));
          Serial.print(p0,2);
          Serial.print(F(" мбар, "));
          //Serial.print(F(" mb, "));
          Serial.print(p0*0.0295333727,2);
          Serial.println(F(" дюймах рт.ст., "));
          //Serial.println(F(" inHg"));
          press_mmHg = p0*0.0295333727*25.4;
          Serial.print(press_mmHg);
          Serial.println(F(" мм рт.ст."));
          /*lcd.print(press_mmHg);
          lcd.print(" mmHg");*/

          // On the other hand, if you want to determine your altitude from the
pressure reading,
          // use the altitude function along with a baseline pressure (sea-level or
other).
          // Parameters: bmp180Pressure = absolute pressure in mb, p0 = baseline
pressure in mb.
          // Result: a = altitude in m.
          a = pressure.altitude(bmp180Pressure,p0);
          Serial.print(F("обчислена висота: "));
          //Serial.print(F("computed altitude: "));
          Serial.print(a,0);
          Serial.print(F(" метрів, "));
          //Serial.print(F(" meters, "));
          Serial.print(a*3.28084,0);
          Serial.println(F(" футів"));
```

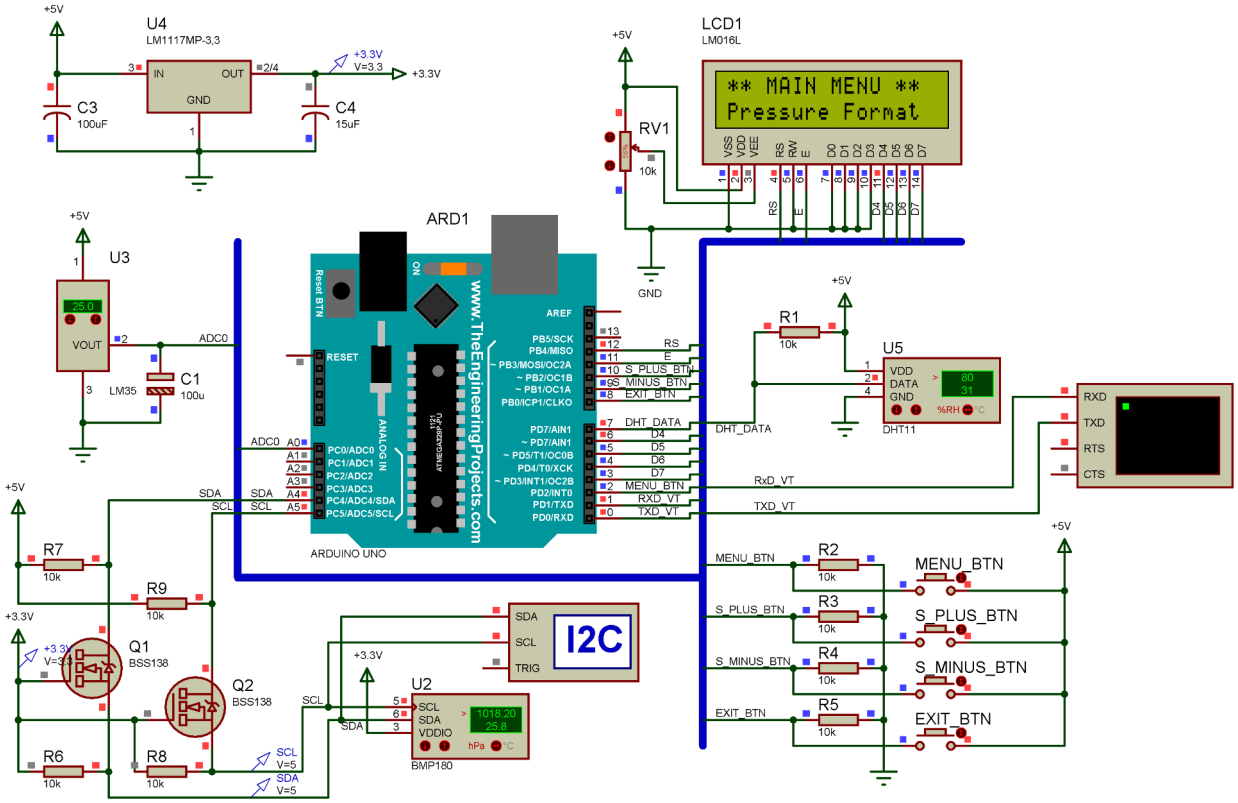
```
          //Serial.println(F(" feet"));
        }
        else Serial.println(F("помилка отримання замірів тиску\n"));
//Serial.println(F("error retrieving pressure measurement\n"));
      }
      else Serial.println(F("помилка початку вимірювання тиску\n"));
//Serial.println(F("error starting pressure measurement\n"));
    }
    else Serial.println(F("помилка отримання замірів температури\n"));
//Serial.println(F("error retrieving temperature measurement\n"));
  }
  else Serial.println(F("помилка початку вимірювання температури\n"));
//Serial.println(F("error starting temperature measurement\n"));
}

void lm35Read()
{
  int val;
  val = analogRead(A0); // змінна знаходиться в інтервалі 0 - 1023
  lm35Temp = (val/1023.0)*5.0*1000/10; // формула обчислення температури в градусах
Цельсія
  Serial.println();
  Serial.println(F("*** Давач температури LM35 ***"));
  //Serial.println(F("*** Temperature sensor LM35 ***"));
  Serial.print("температура: ");
  //Serial.print("temperature: ");
  Serial.print(lm35Temp); // виводимо значення температури на термінал
  Serial.println(F(" градусів С"));
  //Serial.println(F(" degC"));
  /*lcd.setCursor(0,1);
  lcd.print(lm35Temp,1);
  lcd.print(char(223));  */
}

void loop()
{
  // BMP180
  bmp180Read();
  // LM35
  lm35Read();
  // DHT11
  dht11Read();
  lcd.setCursor(0,0);
  lcd.print(bmp180Temperature,1);
  lcd.print(char(223));
  lcd.print("C ");
  switch (pInd)
  {
    case 0:
       lcd.print(bmp180Pressure,0);
       lcd.print(" мбар");
       break;
    case 1:
       lcd.print(bmp180Pressure*0.0295333727*25.399999705,0);
       lcd.print(" мм рт");
       break;
    case 2:
       lcd.print(bmp180Pressure*0.0295333727,1);
       lcd.print(" дюйм");
       break;
     case 3:
```

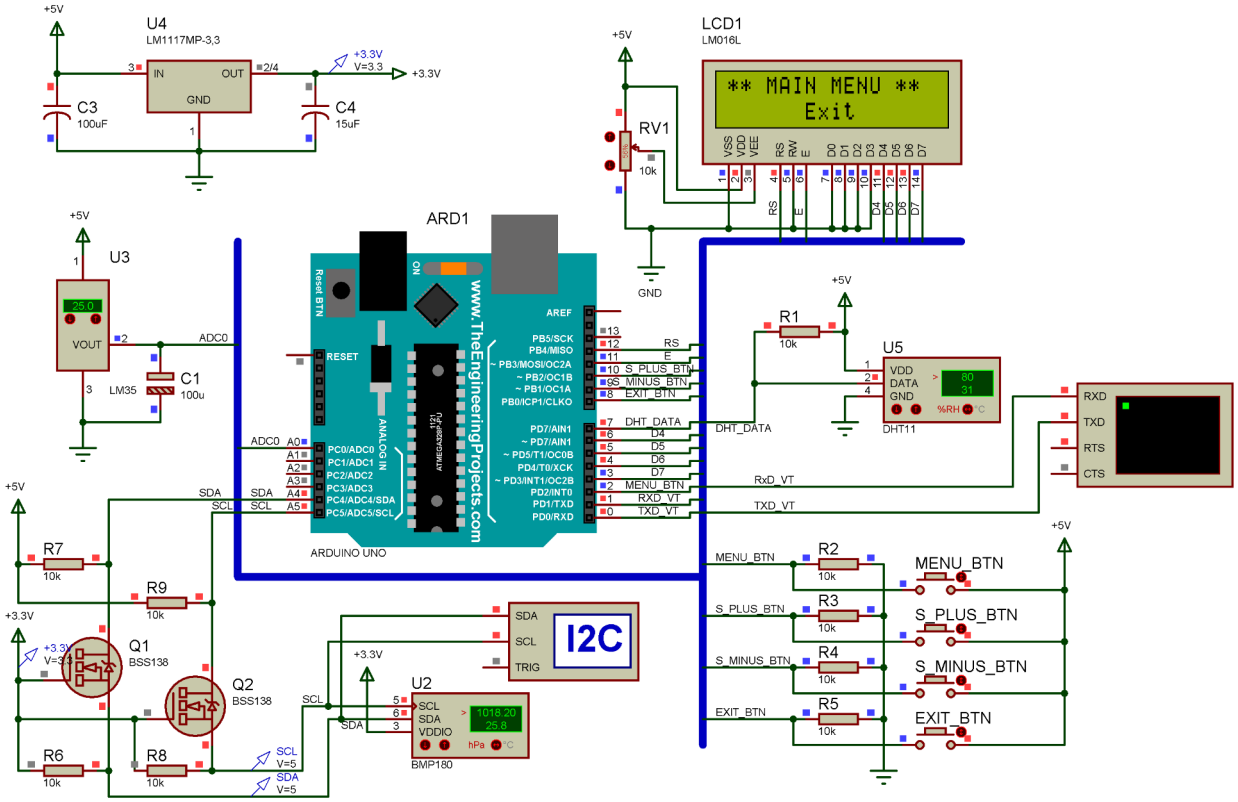
```
      lcd.print(bmp180Pressure,0);
      lcd.print(" гПа");
      break;
  }
  //lm35Read();
  lcd.setCursor(0,1);
  switch (tInd)
  {
    case 0:
      lcd.print(lm35Temp,1);
      lcd.print(char(223));
      lcd.print("C ");
      break;
    case 1:
      lcd.print((9.0/5.0)*lm35Temp+32.0,1);
      lcd.print(char(223));
      lcd.print("F ");
      break;
    case 2:
      lcd.print(lm35Temp+274.15,1);
      //lcd.print(char(223));
      lcd.print(" K ");
      break;
  }
  /*lcd.setCursor(0,1);
  lcd.print(lm35Temp,1);
  lcd.print(char(223));  */
  // DHT22
  //dht22Read();
  lcd.print(dht11Humidity, 1);
  lcd.print("% RH");
  // read the state of the pushbutton value:
 if (mainMenuInvoke)
  mainMenu();
  //lcd.clear();
 delay(1000);  // Pause for 500 ms
}

void pin_ISR() {
  /*buttonState = digitalRead(buttonPin);
  digitalWrite(ledPin, buttonState);.*/
  // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
  if (digitalRead(MENU_BUTTON_PIN) == HIGH) //isButtonPressed(MENU_BUTTON_PIN))
  {
     detachInterrupt(digitalPinToInterrupt(INTERRUPT_PIN));
     mainMenuInvoke = true;
  }
}
```

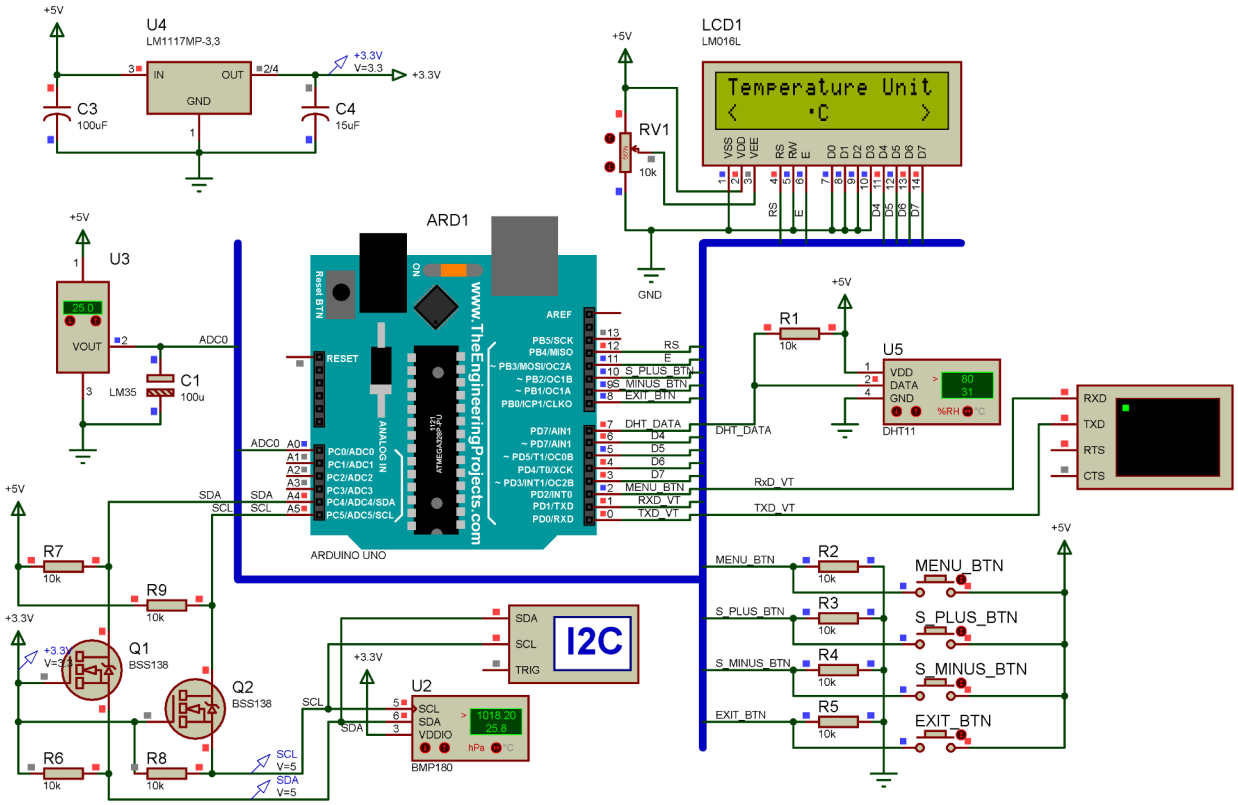
Додаток 3

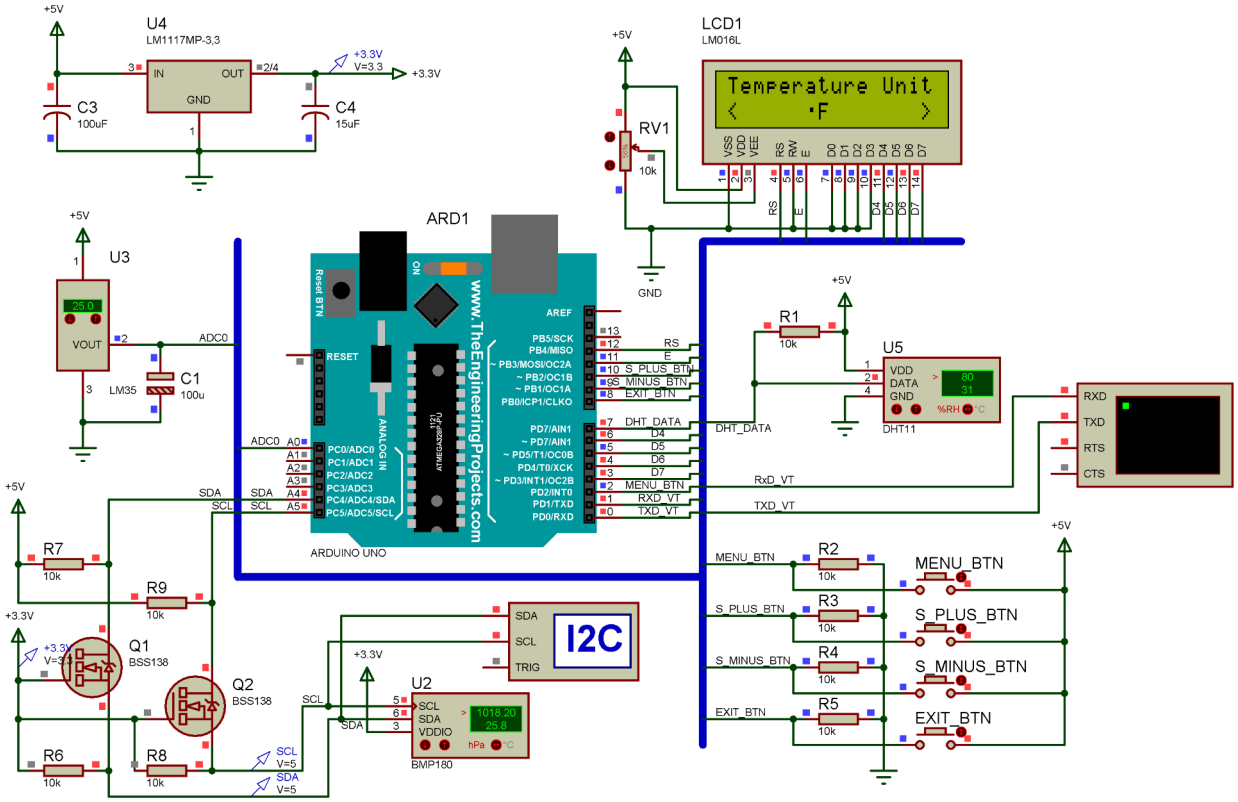
Моделювання роботи системи моніторингу фізико-географічних параметрів

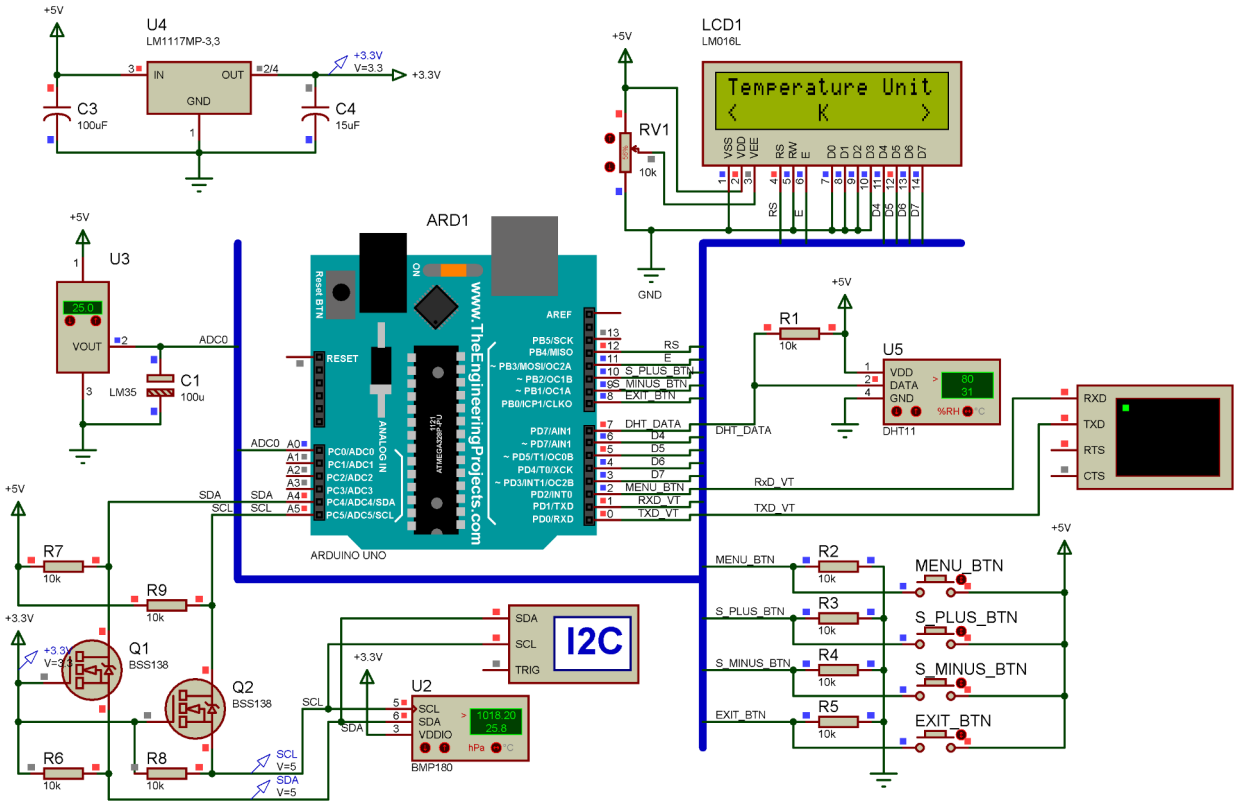


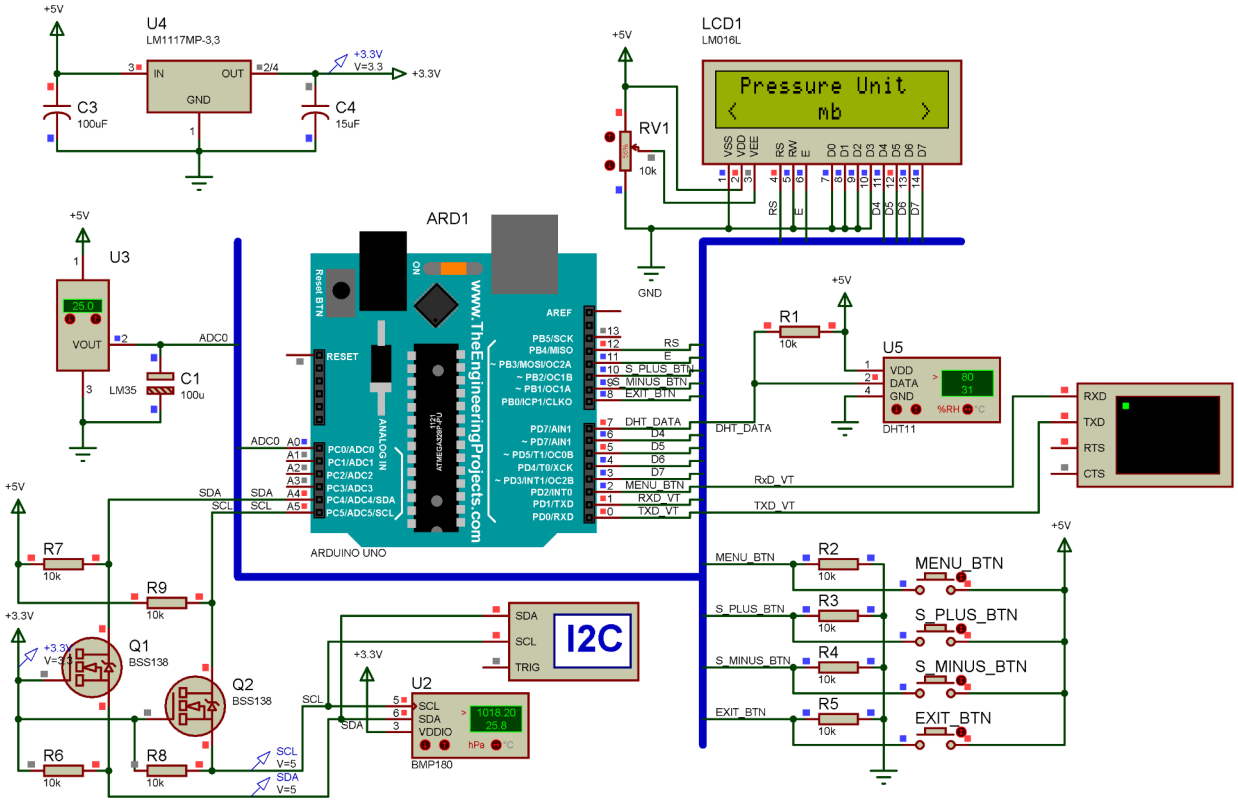


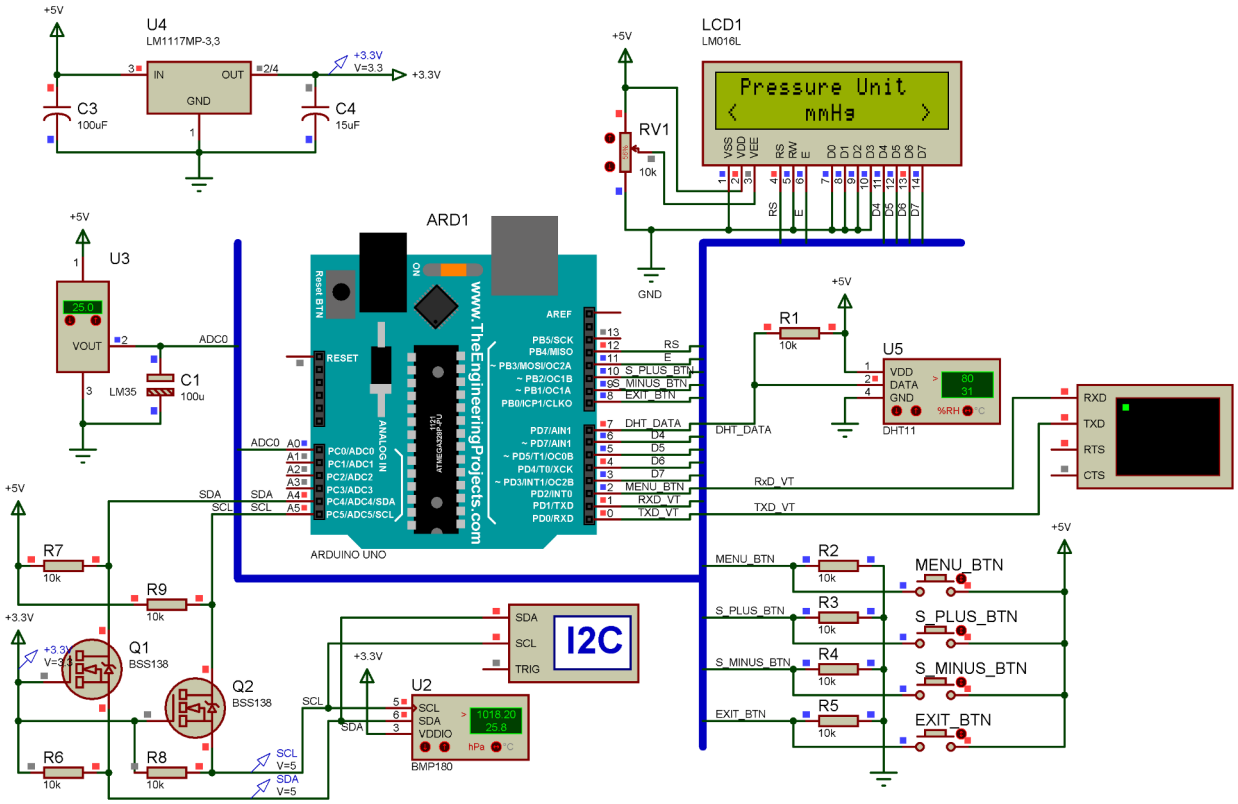

U4
LM1117MP-3,3
IN
OUT
GND
C3
100uF
C4
15uF
+3.3V
LCD1
LM016L
** MAIN MENU **
Exit
RV1
10k
ARD1
www.TheEngineeringProjects.com
ARDUINO UNO
U3
LM35
C1
100u
ADC0
U5
DHT11
DHT_DATA
R1
10k
RXD
TXD
RTS
CTS
RxD_VT
TXD_VT
MENU_BTN
S_PLUS_BTN
S_MINUS_BTN
EXIT_BTN
R2
R3
R4
R5
R6
R7
R8
R9
Q1
BSS138
Q2
I2C
SDA
SCL
TRIG
U2
BMP180

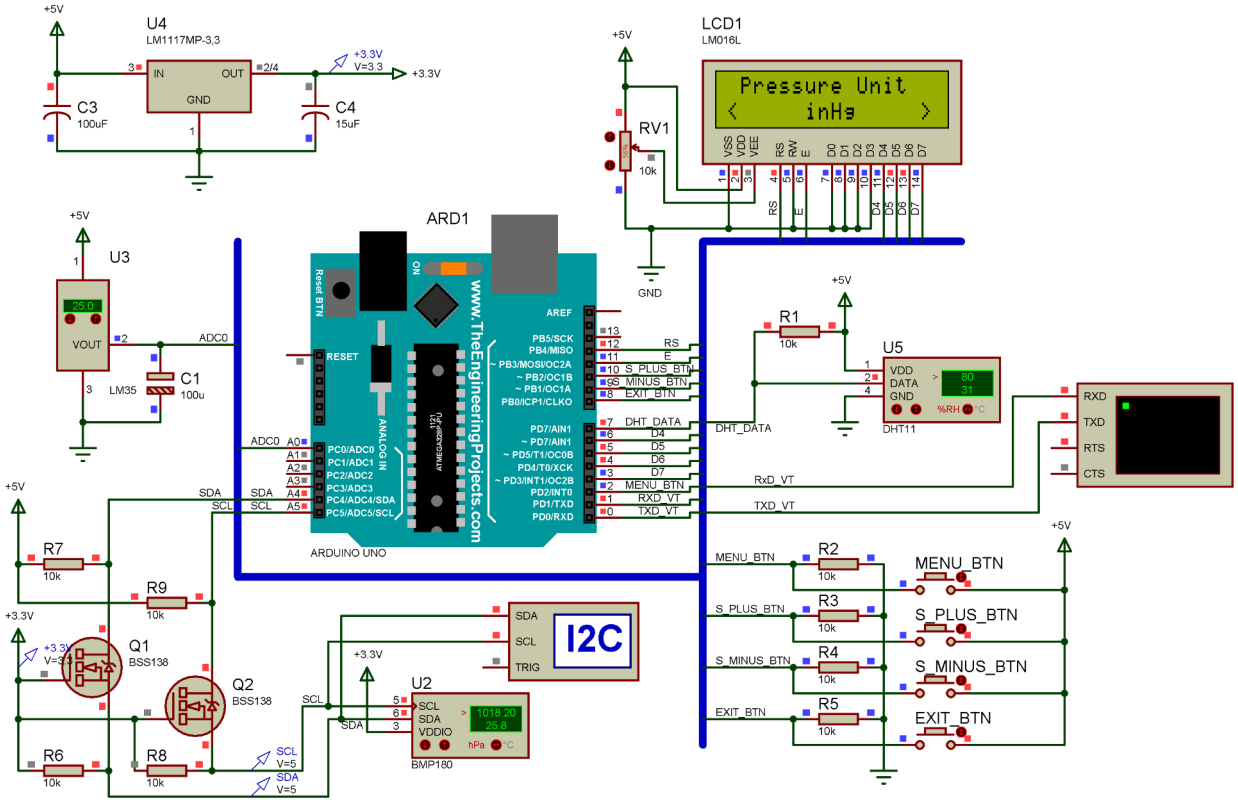

U4
LM1117MP-3,3
IN
OUT
GND
C3
100uF
C4
15uF
+3.3V
LCD1
LM016L
Temperature Unit
< ·C >
RV1
10k
ARD1
www.TheEngineeringProjects.com
ARDUINO UNO
U3
LM35
C1
100u
ADC0
U5
DHT11
DHT_DATA
R1
10k
RXD
TXD
RTS
CTS
RxD_VT
TXD_VT
MENU_BTN
S_PLUS_BTN
S_MINUS_BTN
EXIT_BTN
R2
R3
R4
R5
R6
R7
R8
R9
Q1
BSS138
Q2
I2C
SDA
SCL
TRIG
U2
BMP180

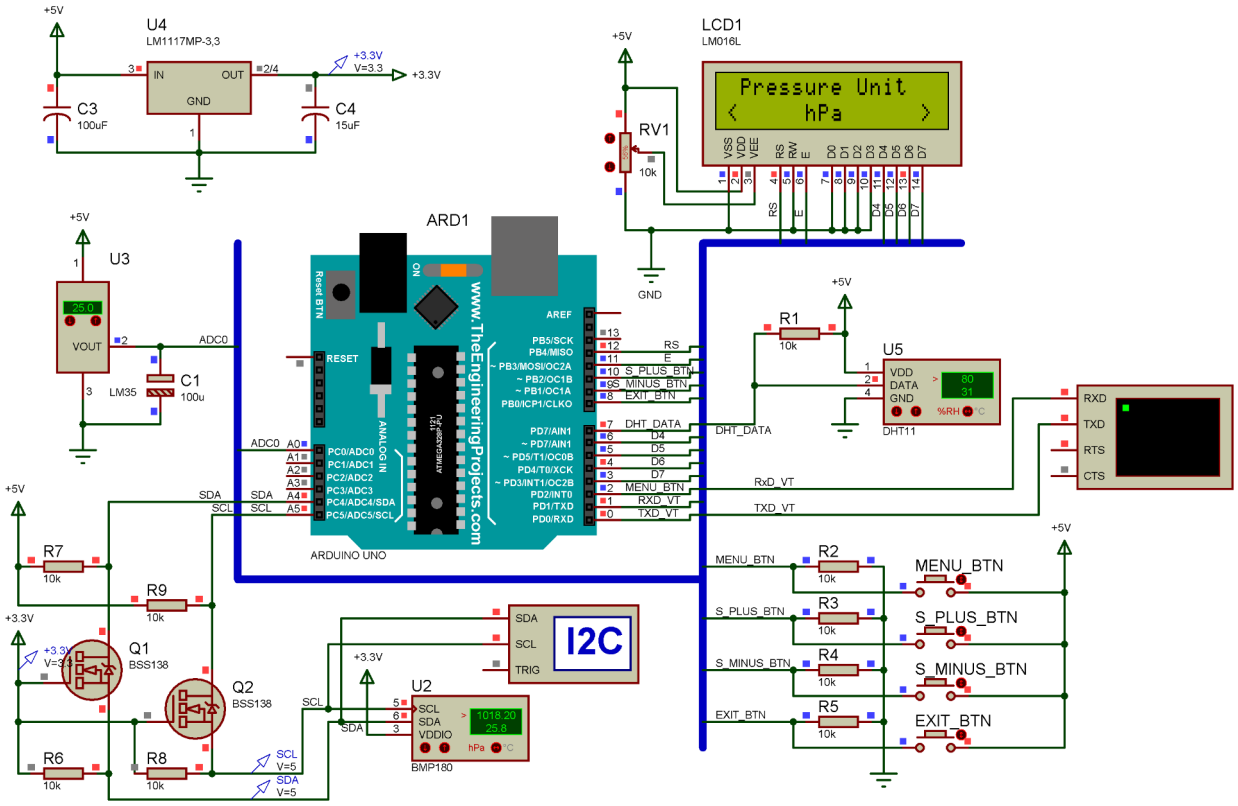

U4
LM1117MP-3,3
C3
100uF
C4
15uF
LCD1
LM016L
Temperature Unit
·F
RV1
10k
ARD1
www.TheEngineeringProjects.com
ARDUINO UNO
U3
LM35
C1
100u
ADC0
R1
10k
U5
DHT11
DHT_DATA
RXD
TXD
RTS
CTS
RxD_VT
TXD_VT
MENU_BTN
S_PLUS_BTN
S_MINUS_BTN
EXIT_BTN
R2
R3
R4
R5
R6
R7
R8
R9
Q1
Q2
BSS138
U2
BMP180
I2C
SDA
SCL
TRIG

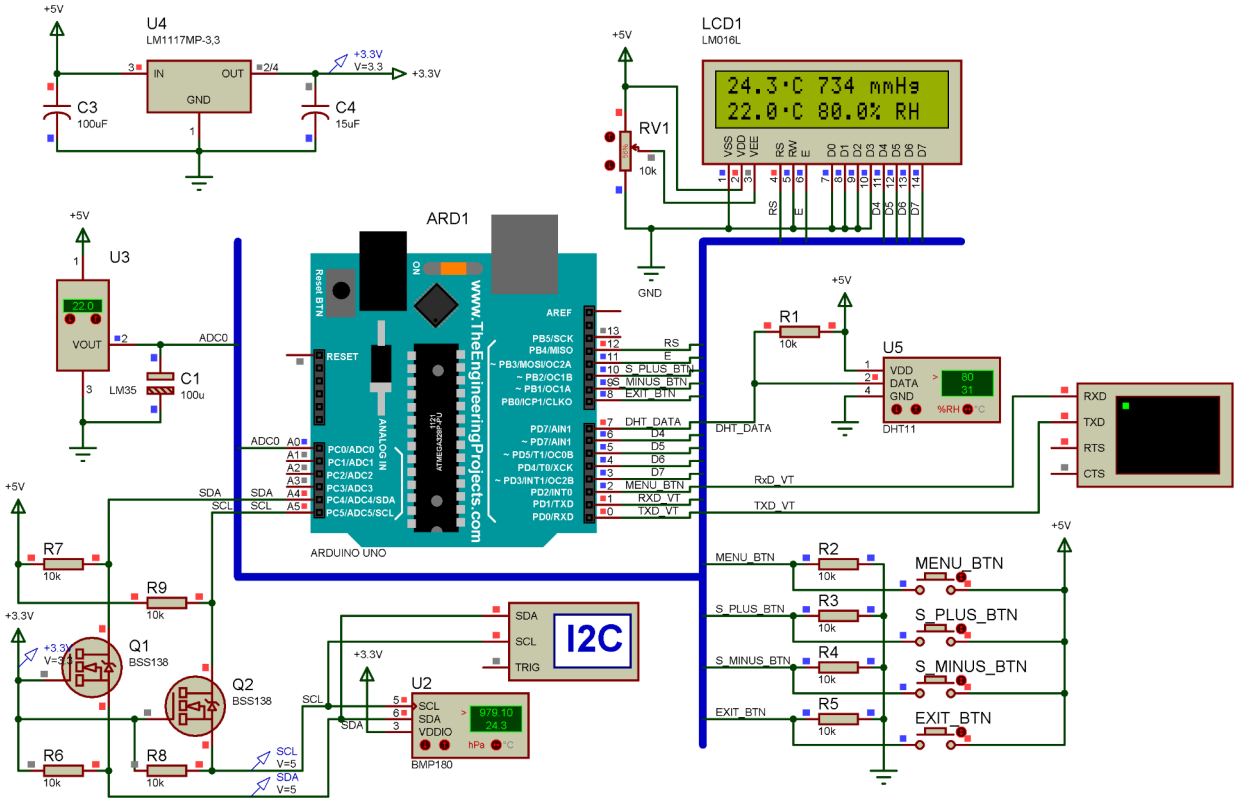

U4
LM1117MP-3,3
C3
100uF
C4
15uF
LCD1
LM016L
Temperature Unit
K
RV1
10k
ARD1
www.TheEngineeringProjects.com
ARDUINO UNO
U3
LM35
C1
100u
ADC0
R1
10k
U5
DHT11
DHT_DATA
RXD
TXD
RTS
CTS
RxD_VT
TXD_VT
MENU_BTN
S_PLUS_BTN
S_MINUS_BTN
EXIT_BTN
R2
R3
R4
R5
R6
R7
R8
R9
Q1
Q2
BSS138
U2
BMP180
I2C
SDA
SCL
TRIG

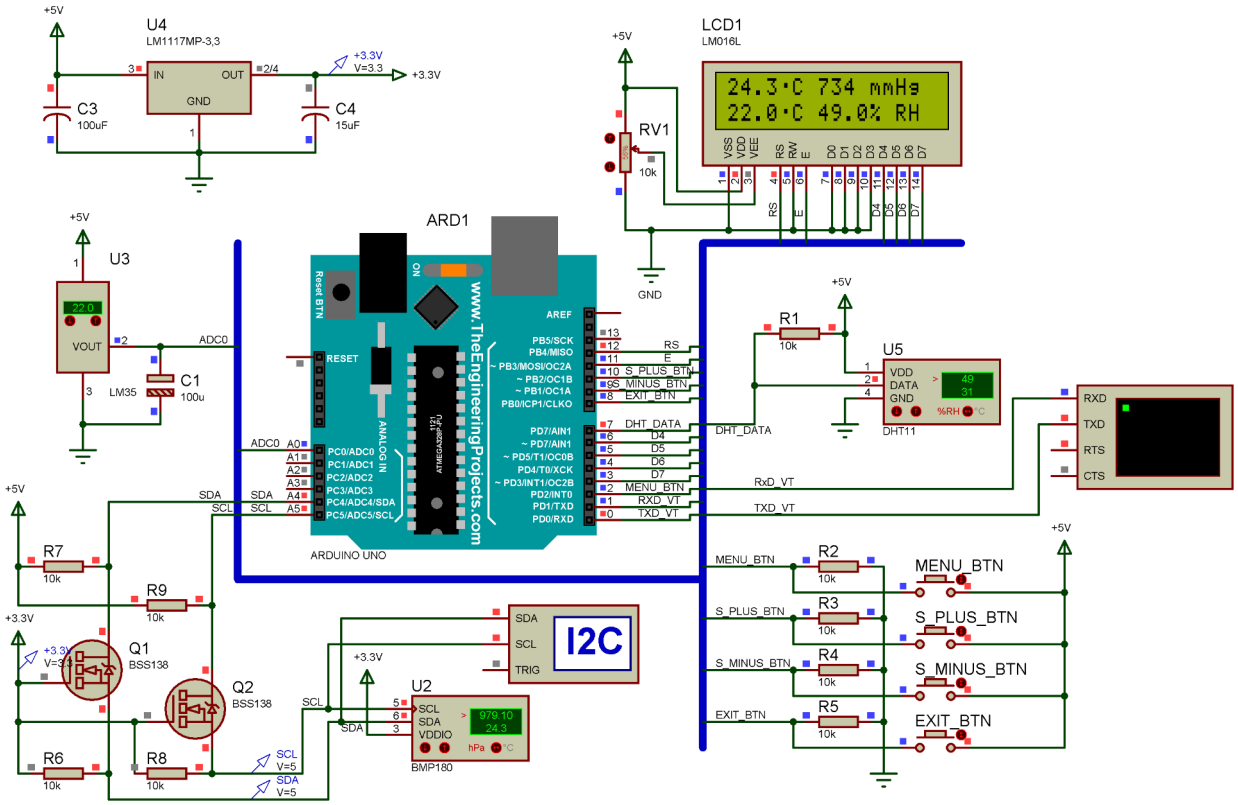

U4
LM1117MP-3,3
C3
100uF
C4
15uF
LCD1
LM016L
Pressure Unit
mb
RV1
10k
ARD1
www.TheEngineeringProjects.com
ARDUINO UNO
U3
LM35
C1
100u
ADC0
U5
DHT11
DHT_DATA
R1
10k
RXD
TXD
RTS
CTS
RxD_VT
TXD_VT
MENU_BTN
S_PLUS_BTN
S_MINUS_BTN
EXIT_BTN
R2
R3
R4
R5
R7
R9
R6
R8
Q1
Q2
BSS138
I2C
SDA
SCL
TRIG
U2
BMP180
+5V
+3.3V
GND

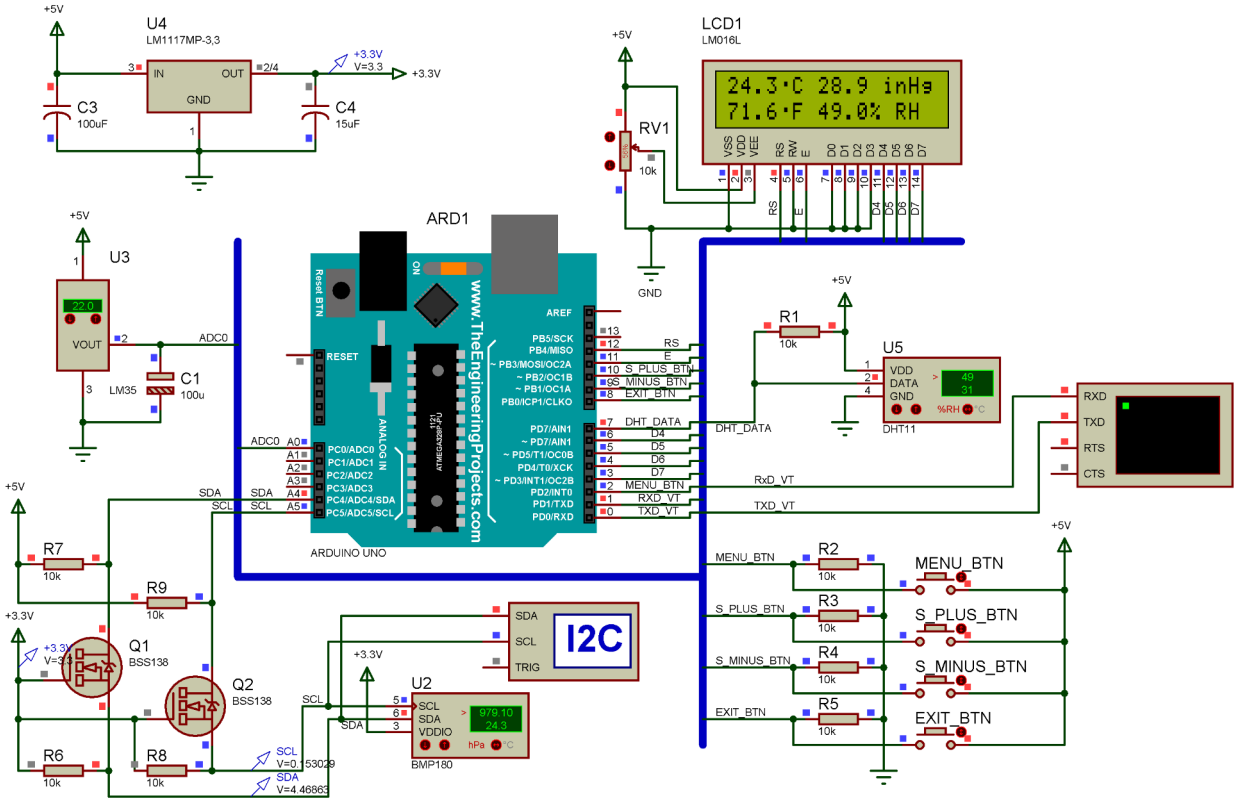

U4
LM1117MP-3,3
C3
100uF
C4
15uF
LCD1
LM016L
Pressure Unit
mmHg
RV1
10k
ARD1
www.TheEngineeringProjects.com
ARDUINO UNO
U3
LM35
C1
100u
ADC0
U5
DHT11
DHT_DATA
R1
10k
RXD
TXD
RTS
CTS
RxD_VT
TXD_VT
MENU_BTN
S_PLUS_BTN
S_MINUS_BTN
EXIT_BTN
R2
R3
R4
R5
R7
R9
R6
R8
Q1
Q2
BSS138
I2C
SDA
SCL
TRIG
U2
BMP180
+5V
+3.3V
GND

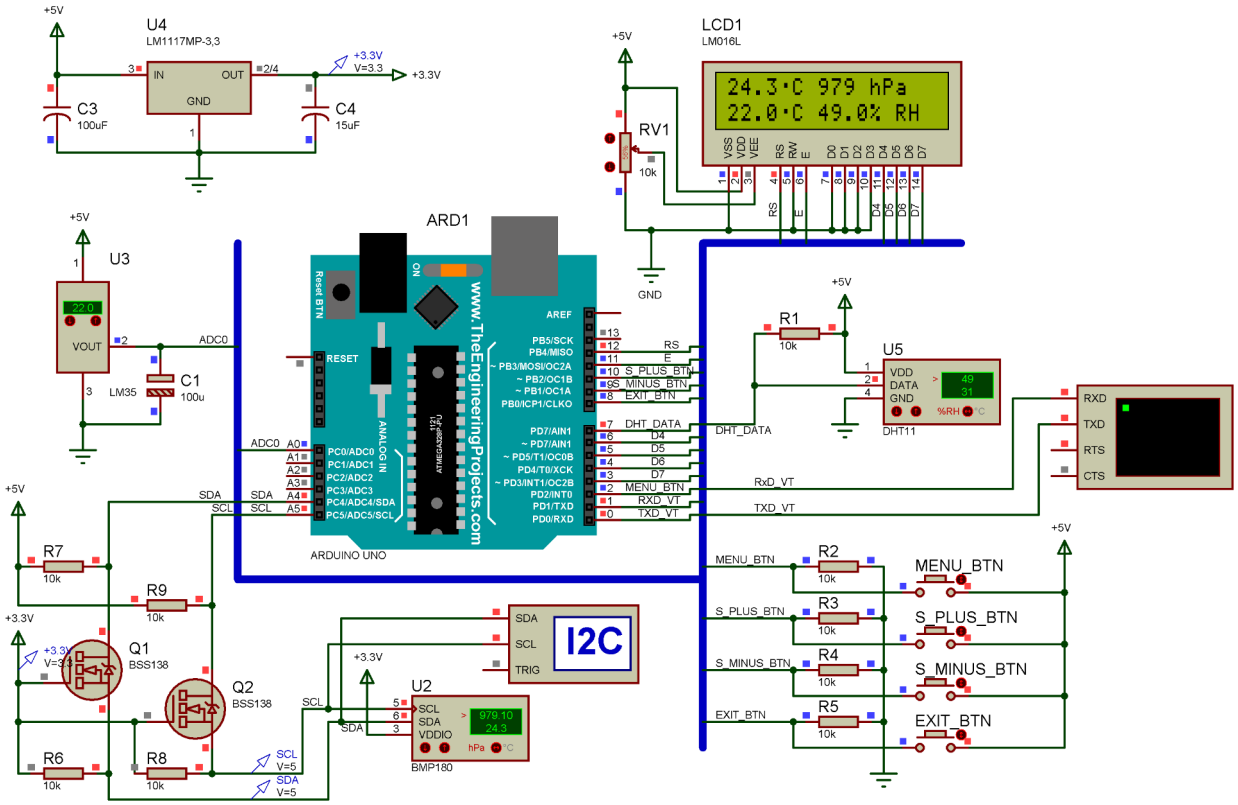

U4
LM1117MP-3,3
IN
OUT
GND
C3
100uF
C4
15uF
+3.3V
V=3.3
LCD1
LM016L
Pressure Unit
inHg
RV1
10k
ARD1
U3
LM35
C1
100u
ADC0
ARDUINO UNO
www.TheEngineeringProjects.com
RESET
ANALOG IN
AREF
S_PLUS_BTN
S_MINUS_BTN
EXIT_BTN
MENU_BTN
DHT_DATA
RXD_VT
TXD_VT
R1
U5
DHT11
RXD
TXD
RTS
CTS
R2
R3
R4
R5
R6
R7
R8
R9
Q1
Q2
BSS138
I2C
SDA
SCL
TRIG
U2
BMP180
VDDIO
hPa

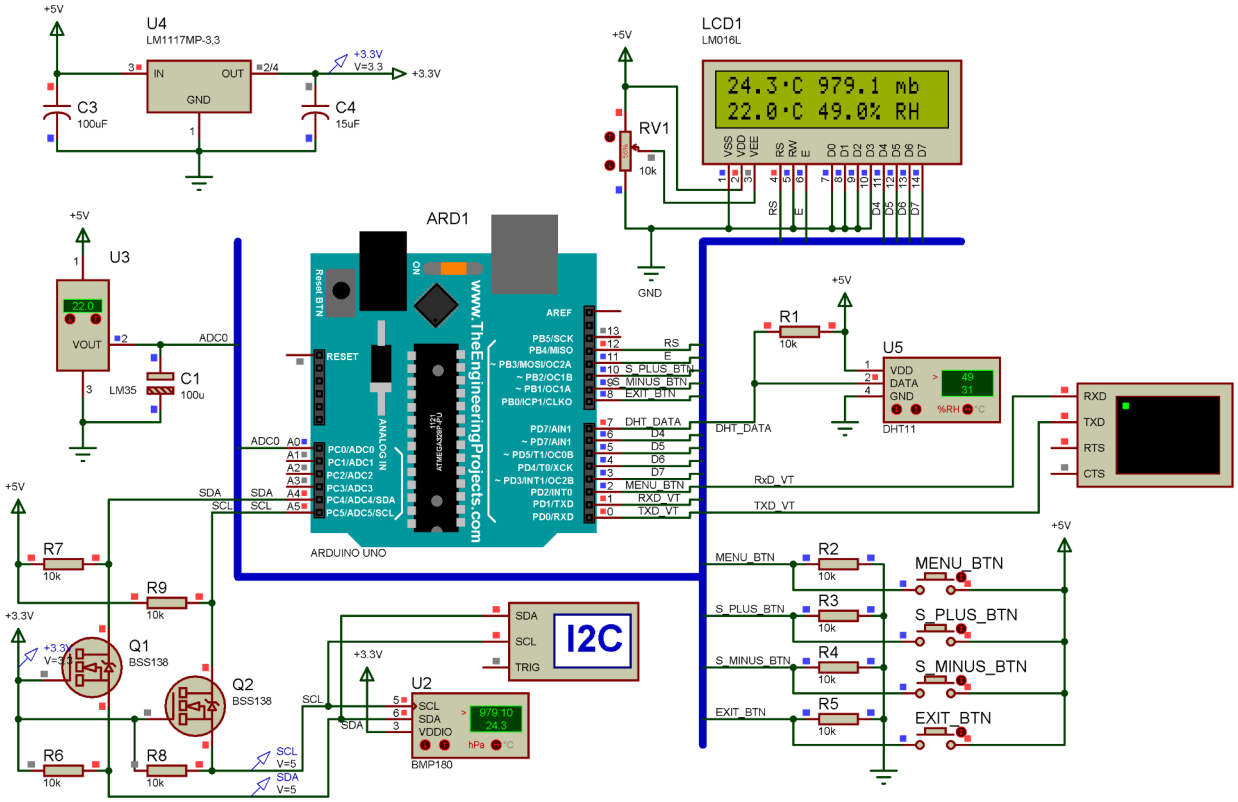

U4
LM1117MP-3,3
IN
OUT
GND
C3
100uF
C4
15uF
+3.3V
V=3.3
LCD1
LM016L
Pressure Unit
hPa
RV1
10k
ARD1
U3
LM35
C1
100u
ADC0
ARDUINO UNO
www.TheEngineeringProjects.com
RESET
ANALOG IN
AREF
S_PLUS_BTN
S_MINUS_BTN
EXIT_BTN
MENU_BTN
DHT_DATA
RXD_VT
TXD_VT
R1
U5
DHT11
RXD
TXD
RTS
CTS
R2
R3
R4
R5
R6
R7
R8
R9
Q1
Q2
BSS138
I2C
SDA
SCL
TRIG
U2
BMP180
VDDIO
hPa

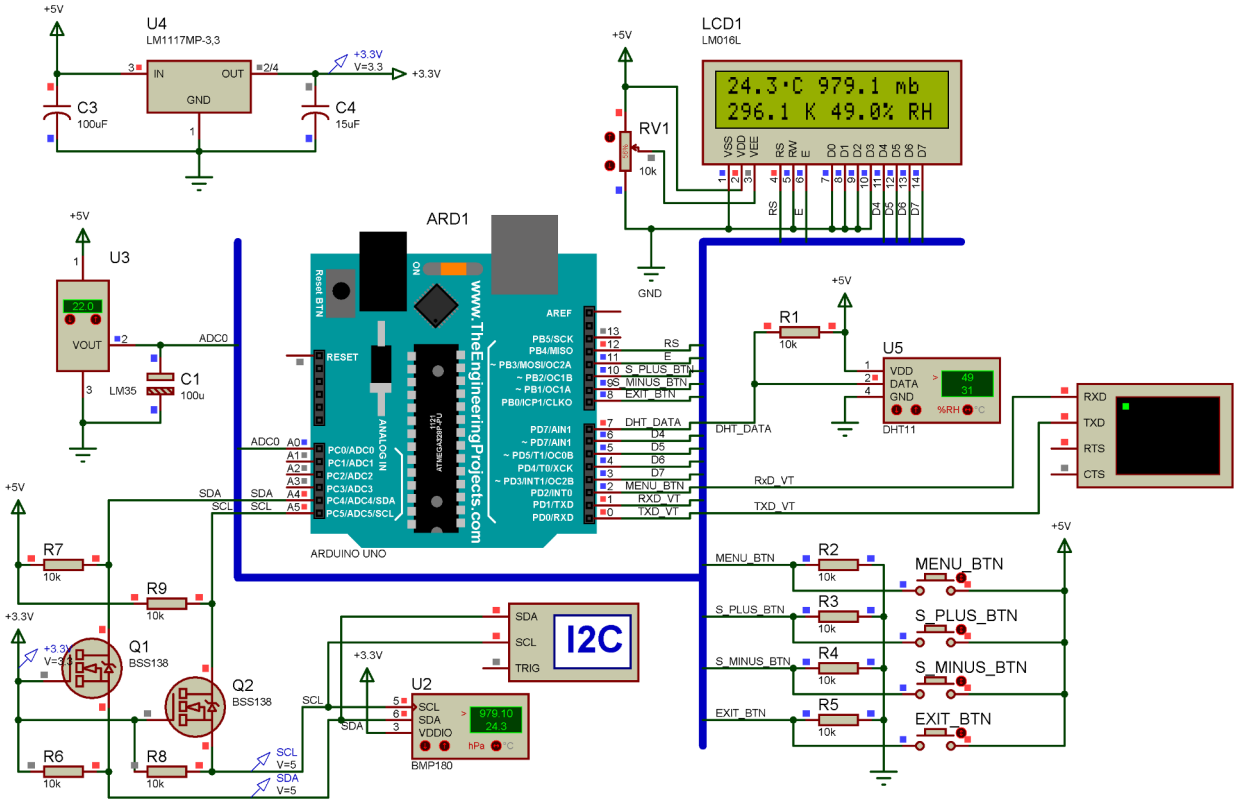

U4
LM1117MP-3,3
IN
OUT
GND
C3
100uF
C4
15uF
+3.3V
V=3.3
LCD1
LM016L
24.3°C 734 mmHg
22.0°C 80.0% RH
RV1
10k
ARD1
www.TheEngineeringProjects.com
ARDUINO UNO
U3
LM35
C1
100u
ADC0
U5
DHT11
DHT_DATA
R1
10k
RS
E
D4
D5
D6
D7
S_PLUS_BTN
S_MINUS_BTN
EXIT_BTN
MENU_BTN
RXD_VT
TXD_VT
RXD
TXD
RTS
CTS
R2
R3
R4
R5
R7
R8
R9
R6
Q1
Q2
BSS138
I2C
SDA
SCL
TRIG
U2
BMP180
979.10
24.3
hPa
°C
%RH

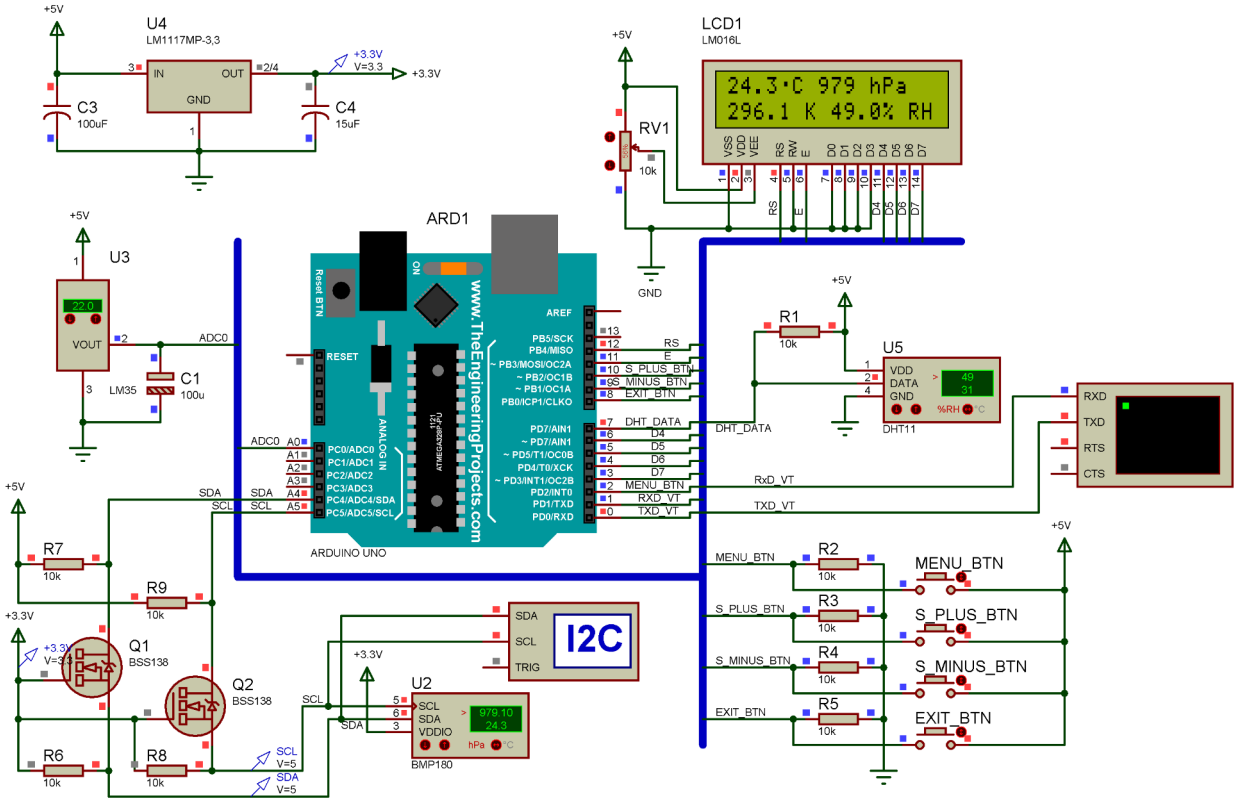

U4
LM1117MP-3,3
IN
OUT
GND
C3
100uF
C4
15uF
+3.3V
V=3.3
LCD1
LM016L
24.3°C 734 mmHg
22.0°C 49.0% RH
RV1
10k
ARD1
www.TheEngineeringProjects.com
ARDUINO UNO
U3
LM35
C1
100u
ADC0
U5
DHT11
DHT_DATA
R1
10k
RS
E
D4
D5
D6
D7
S_PLUS_BTN
S_MINUS_BTN
EXIT_BTN
MENU_BTN
RXD_VT
TXD_VT
RXD
TXD
RTS
CTS
R2
R3
R4
R5
R7
R8
R9
R6
Q1
Q2
BSS138
I2C
SDA
SCL
TRIG
U2
BMP180
979.10
24.3
hPa
°C
%RH

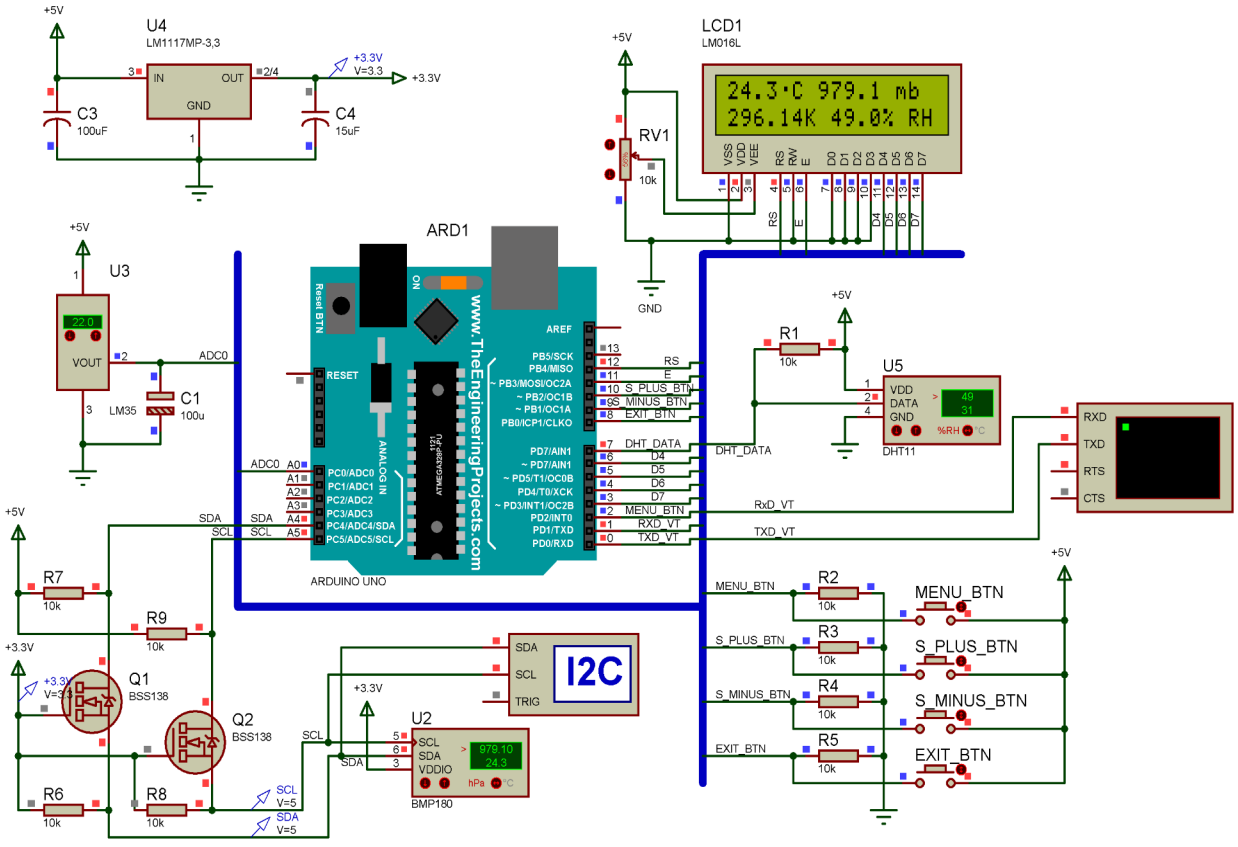

U4
LM1117MP-3,3
IN
OUT
GND
C3
100uF
C4
15uF
+3.3V
LCD1
LM016L
24.3°C 28.9 inHg
71.6°F 49.0% RH
RV1
10k
ARD1
www.TheEngineeringProjects.com
ARDUINO UNO
U3
LM35
C1
100u
ADC0
U5
DHT11
DHT_DATA
RXD
TXD
RTS
CTS
RxD_VT
TXD_VT
MENU_BTN
S_PLUS_BTN
S_MINUS_BTN
EXIT_BTN
R1
R2
R3
R4
R5
R6
R7
R8
R9
Q1
Q2
BSS138
I2C
SDA
SCL
TRIG
U2
BMP180
979.10
24.3
hPa


U4
LM1117MP-3,3
IN
OUT
GND
C3
100uF
C4
15uF
+3.3V
LCD1
LM016L
24.3°C 979 hPa
22.0°C 49.0% RH
RV1
10k
ARD1
www.TheEngineeringProjects.com
ARDUINO UNO
U3
LM35
C1
100u
ADC0
U5
DHT11
DHT_DATA
RXD
TXD
RTS
CTS
RxD_VT
TXD_VT
MENU_BTN
S_PLUS_BTN
S_MINUS_BTN
EXIT_BTN
R1
R2
R3
R4
R5
R6
R7
R8
R9
Q1
Q2
BSS138
I2C
SDA
SCL
TRIG
U2
BMP180
979.10
24.3
hPa


U4
LM1117MP-3,3
IN
OUT
GND
C3
100uF
C4
15uF
+3.3V
LCD1
LM016L
24.3°C 979.1 mb
22.0°C 49.0% RH
RV1
10k
ARD1
www.TheEngineeringProjects.com
ARDUINO UNO
U3
LM35
C1
100u
ADC0
U5
DHT11
DHT_DATA
R1
10k
RXD
TXD
RTS
CTS
RxD_VT
TXD_VT
MENU_BTN
S_PLUS_BTN
S_MINUS_BTN
EXIT_BTN
R2
R3
R4
R5
R7
R9
R6
R8
Q1
Q2
BSS138
I2C
SDA
SCL
TRIG
U2
BMP180


U4
LM1117MP-3,3
IN
OUT
GND
C3
100uF
C4
15uF
+3.3V
LCD1
LM016L
24.3°C 979.1 mb
296.1 K 49.0% RH
RV1
10k
ARD1
www.TheEngineeringProjects.com
ARDUINO UNO
U3
LM35
C1
100u
ADC0
U5
DHT11
DHT_DATA
R1
10k
RXD
TXD
RTS
CTS
RxD_VT
TXD_VT
MENU_BTN
S_PLUS_BTN
S_MINUS_BTN
EXIT_BTN
R2
R3
R4
R5
R7
R9
R6
R8
Q1
Q2
BSS138
I2C
SDA
SCL
TRIG
U2
BMP180


U4
LM1117MP-3,3
C3
100uF
C4
15uF
+3.3V
LCD1
LM016L
24.3°C 979 hPa
296.1 K 49.0% RH
RV1
10k
ARD1
www.TheEngineeringProjects.com
ARDUINO UNO
U3
LM35
C1
100u
ADC0
R1
U5
DHT11
DHT_DATA
RXD
TXD
RTS
CTS
RxD_VT
TXD_VT
MENU_BTN
S_PLUS_BTN
S_MINUS_BTN
EXIT_BTN
R2
R3
R4
R5
R6
R7
R8
R9
Q1
Q2
BSS138
I2C
SDA
SCL
TRIG
U2
BMP180
979.10
24.3
hPa


U4
LM1117MP-3,3
C3
100uF
C4
15uF
+3.3V
LCD1
LM016L
24.3°C 979.1 mb
296.14K 49.0% RH
RV1
10k
ARD1
www.TheEngineeringProjects.com
ARDUINO UNO
U3
LM35
C1
100u
ADC0
R1
U5
DHT11
DHT_DATA
RXD
TXD
RTS
CTS
RxD_VT
TXD_VT
MENU_BTN
S_PLUS_BTN
S_MINUS_BTN
EXIT_BTN
R2
R3
R4
R5
R6
R7
R8
R9
Q1
Q2
BSS138
I2C
SDA
SCL
TRIG
U2
BMP180
979.10
24.3
hPa

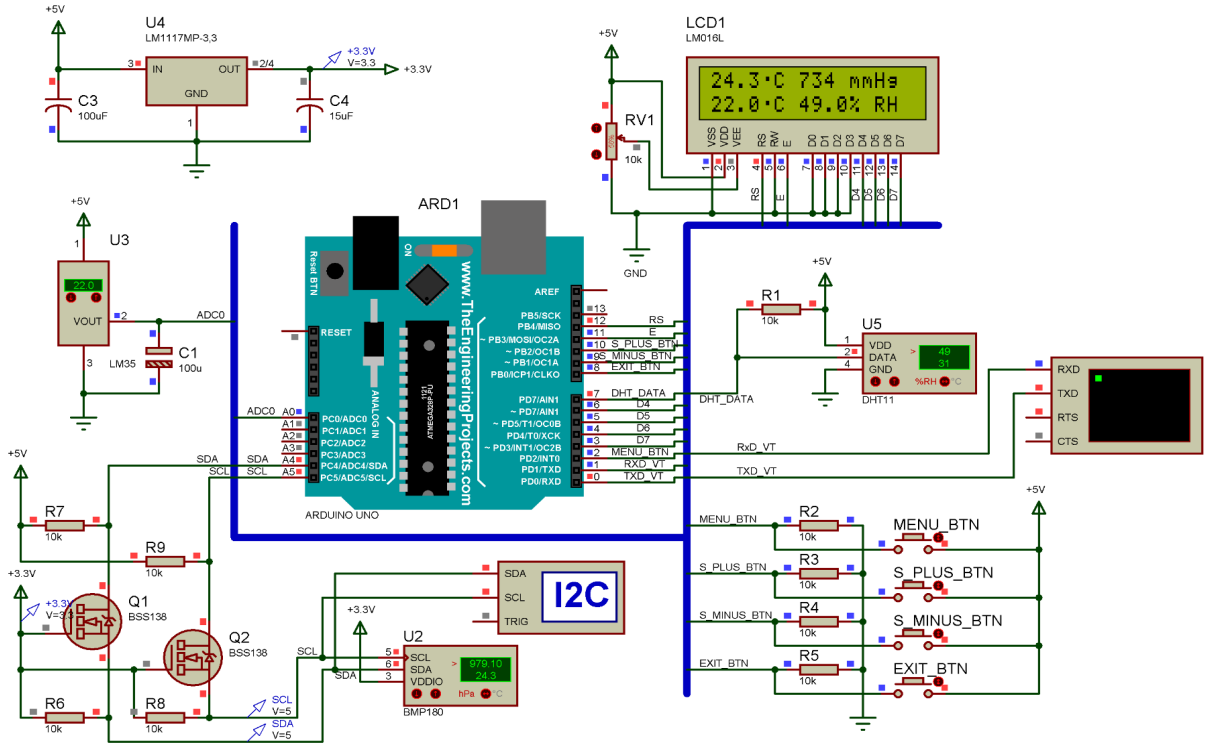
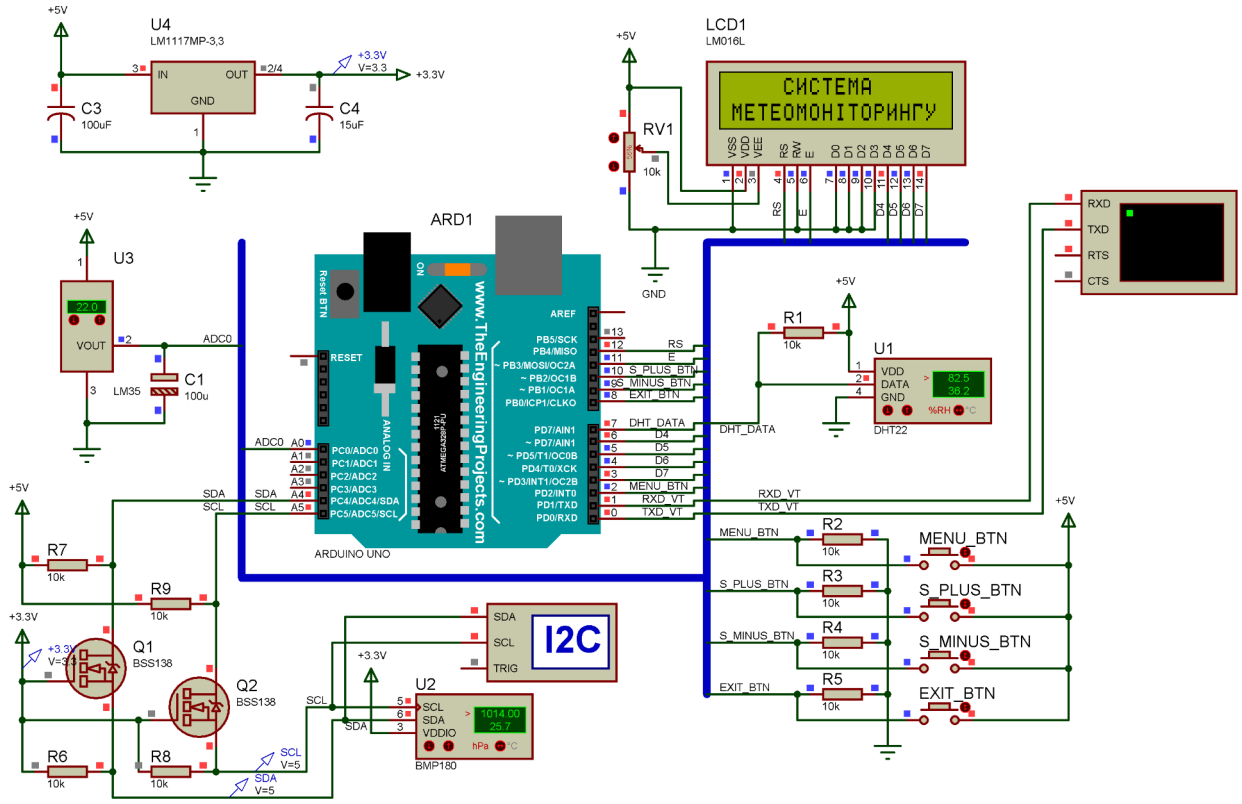
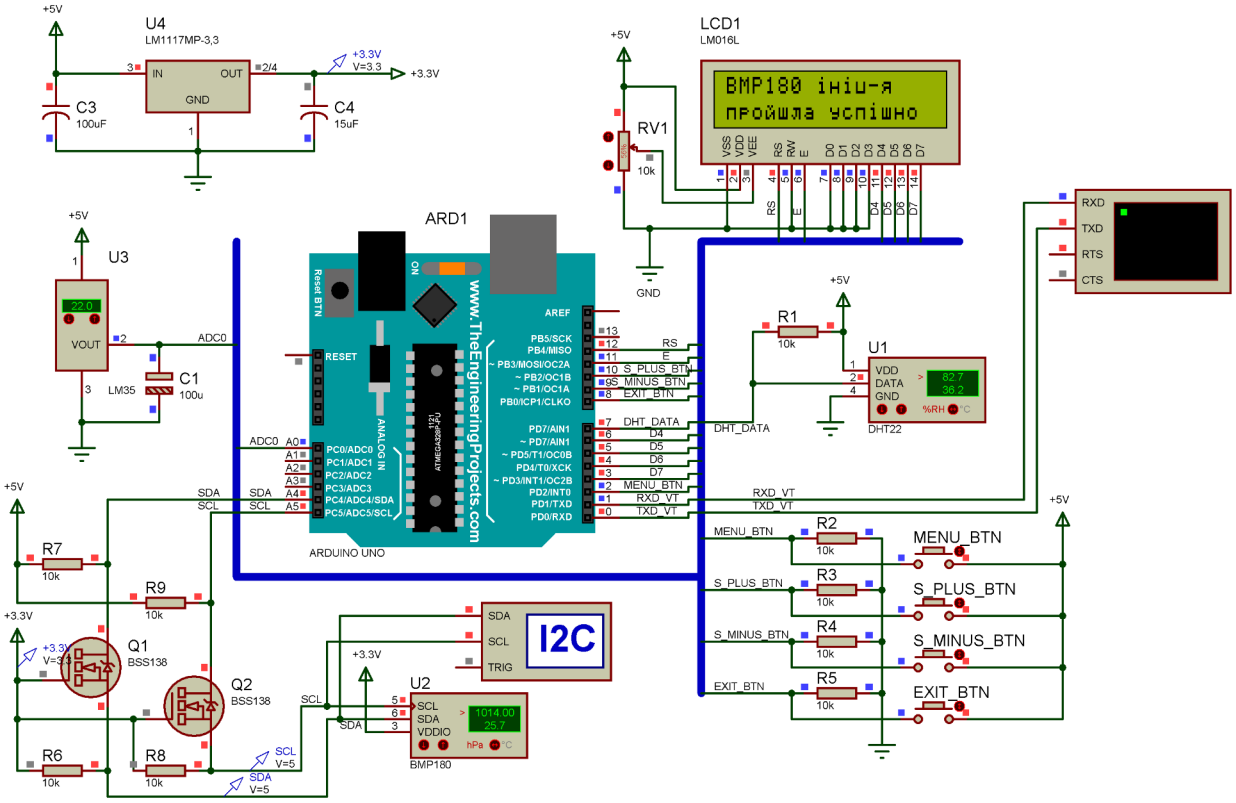
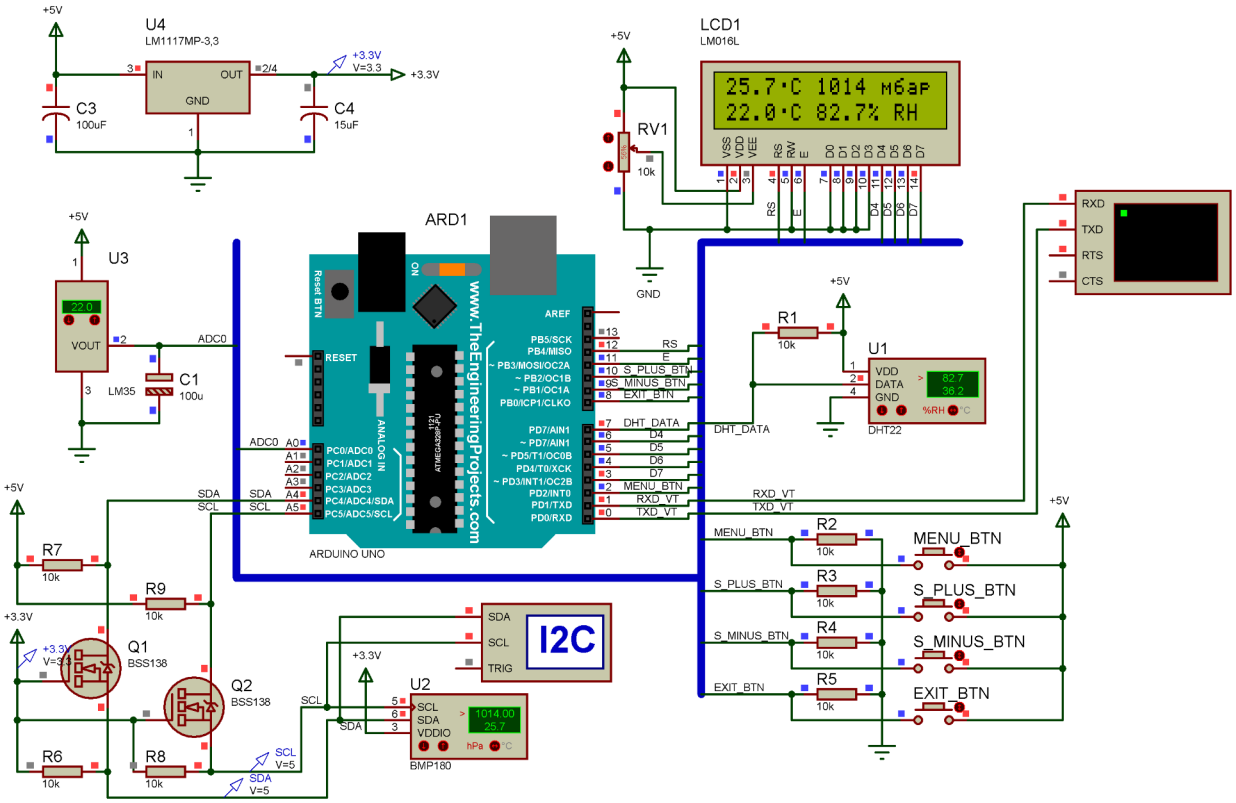
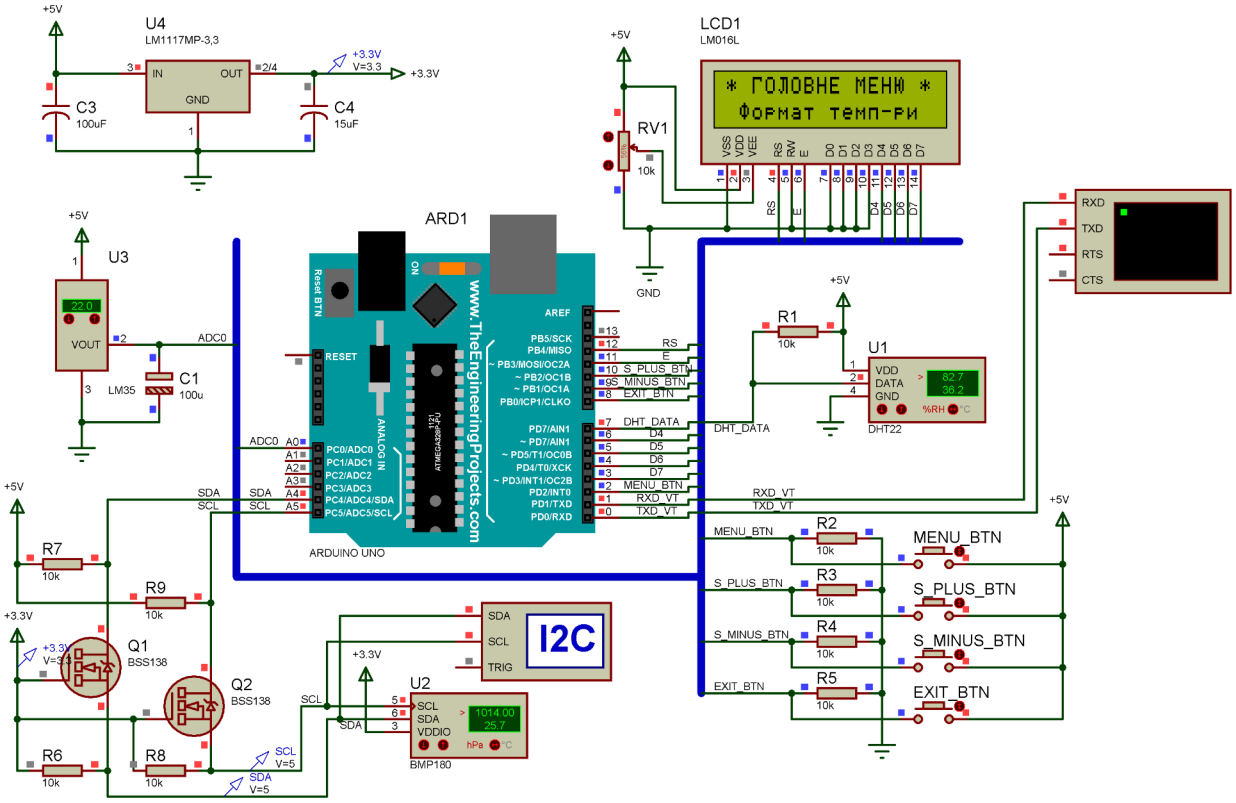


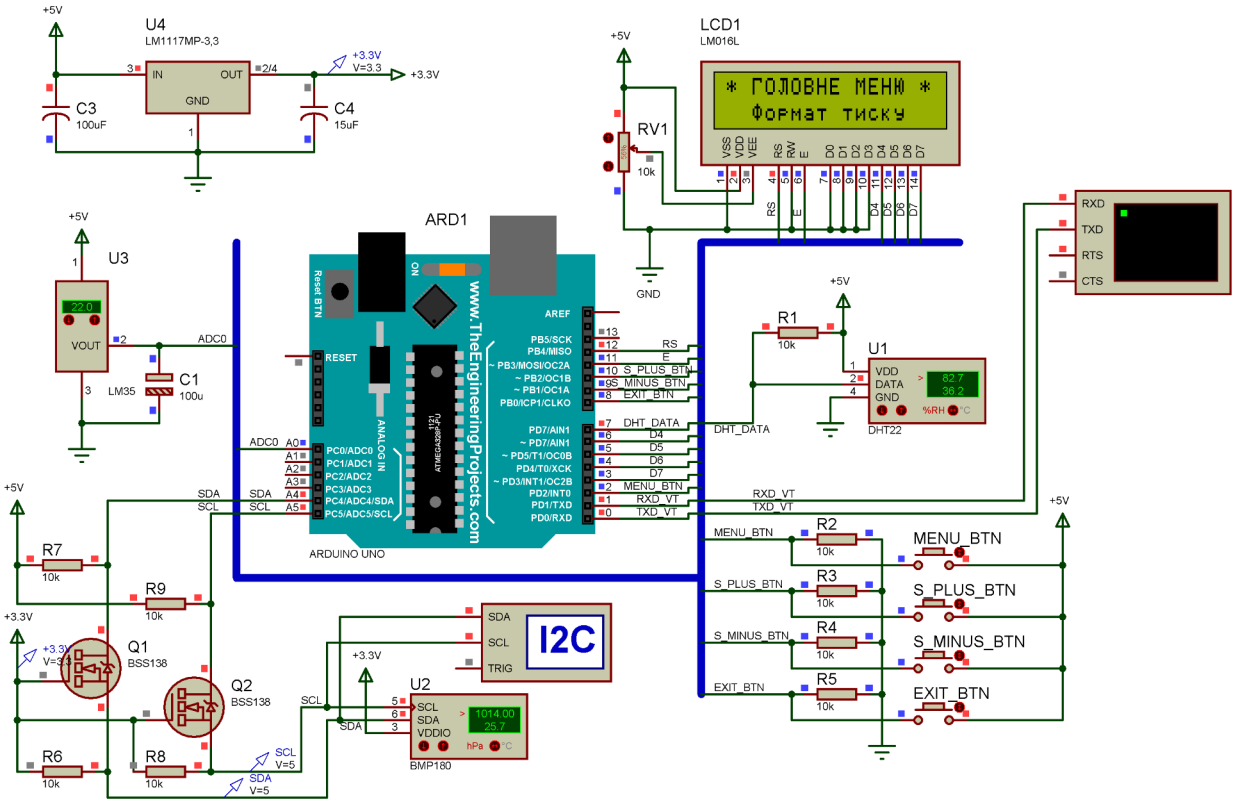
Рис.4.5. Моделювання роботи системи моніторингу фізико-географічних параметрів

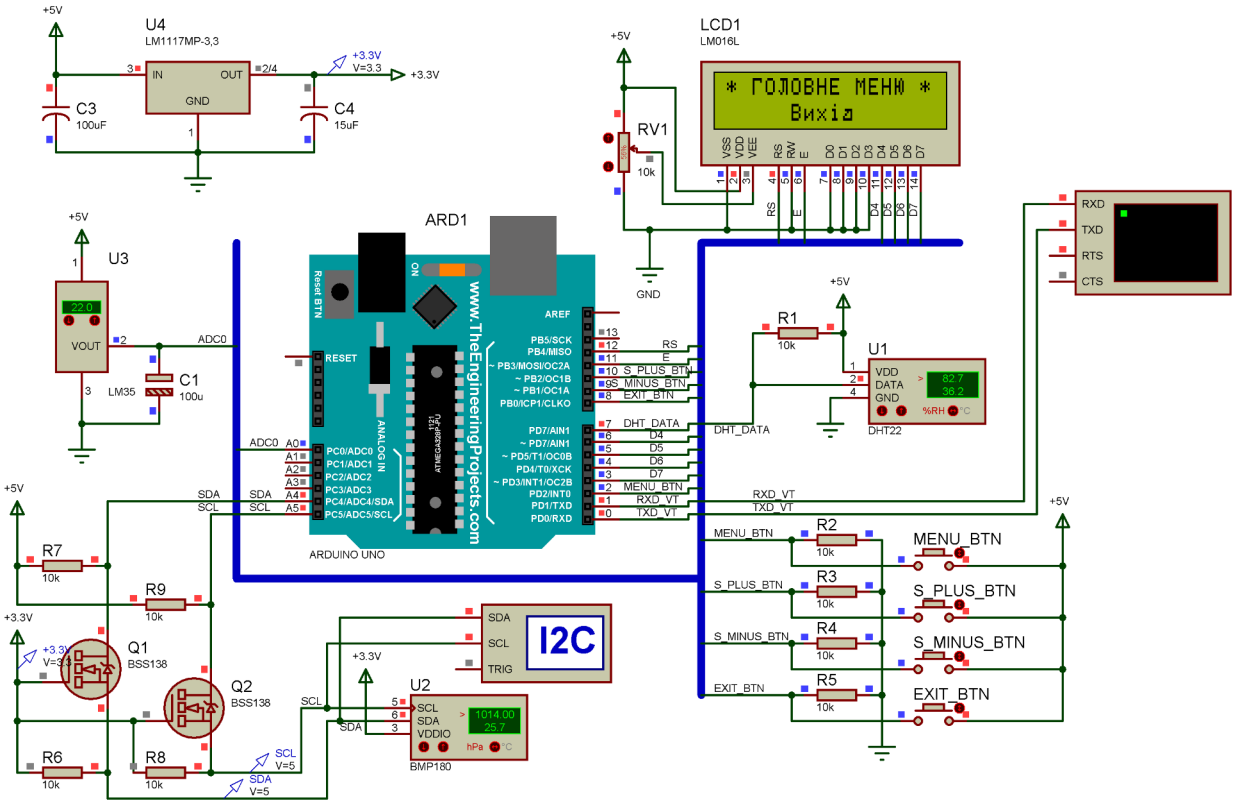


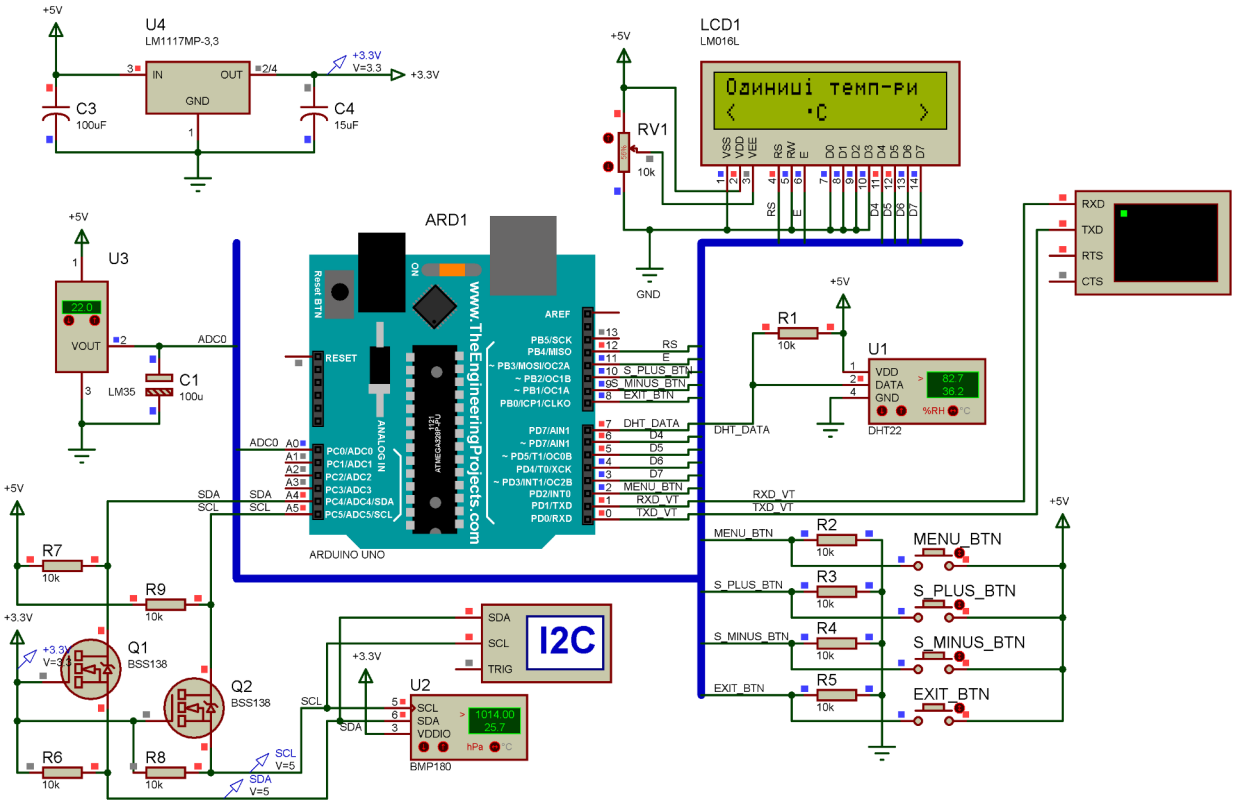

U4
LM1117MP-3,3
C3
100uF
C4
15uF
+3.3V
LCD1
LM016L
BMP180 ініц-я
пройшла успішно
RV1
10k
ARD1
www.TheEngineeringProjects.com
ARDUINO UNO
U3
LM35
C1
100u
ADC0
U1
DHT22
DHT_DATA
R1
10k
RXD
TXD
RTS
CTS
R2
R3
R4
R5
MENU_BTN
S_PLUS_BTN
S_MINUS_BTN
EXIT_BTN
R7
R9
R6
R8
Q1
BSS138
Q2
I2C
U2
BMP180
SDA
SCL

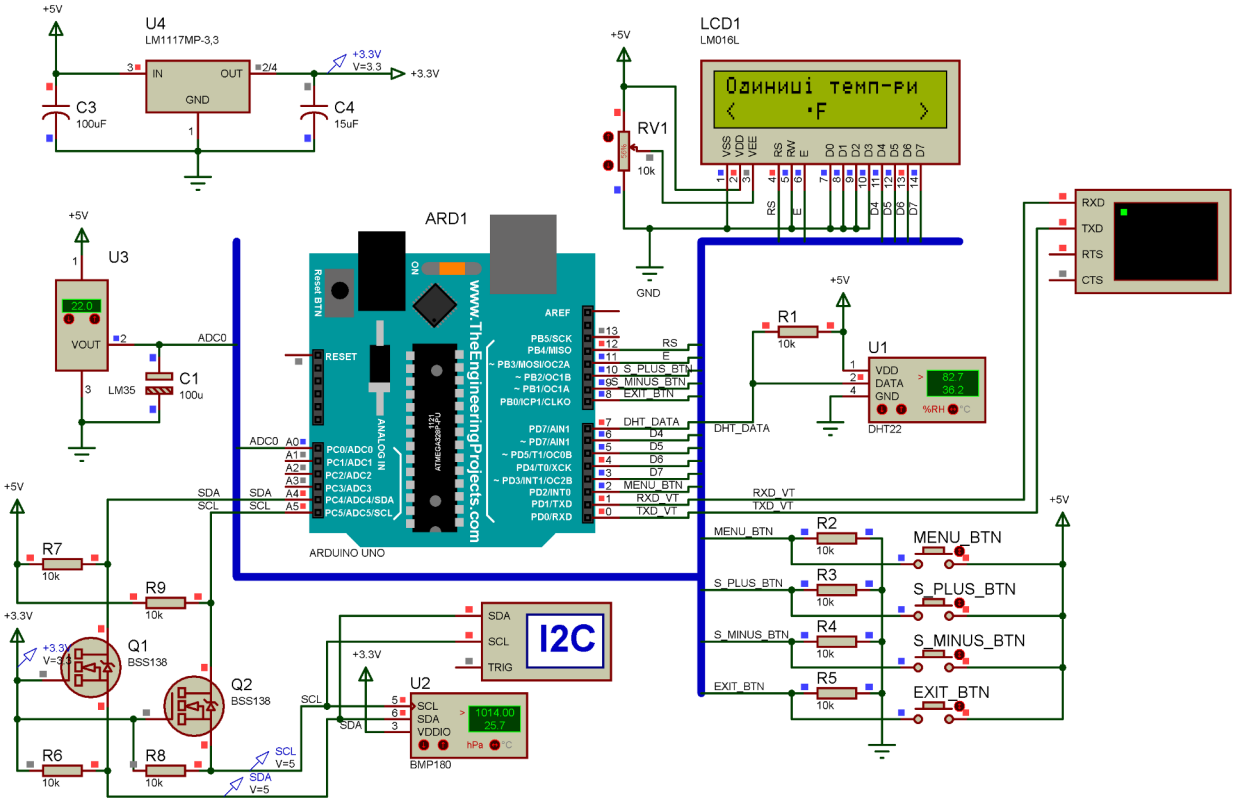

U4
LM1117MP-3,3
C3
100uF
C4
15uF
+3.3V
LCD1
LM016L
25.7°C 1014 мбар
22.0°C 82.7% RH
RV1
10k
ARD1
www.TheEngineeringProjects.com
ARDUINO UNO
U3
LM35
C1
100u
ADC0
U1
DHT22
DHT_DATA
R1
10k
RXD
TXD
RTS
CTS
R2
R3
R4
R5
MENU_BTN
S_PLUS_BTN
S_MINUS_BTN
EXIT_BTN
R7
R9
R6
R8
Q1
BSS138
Q2
I2C
U2
BMP180
SDA
SCL

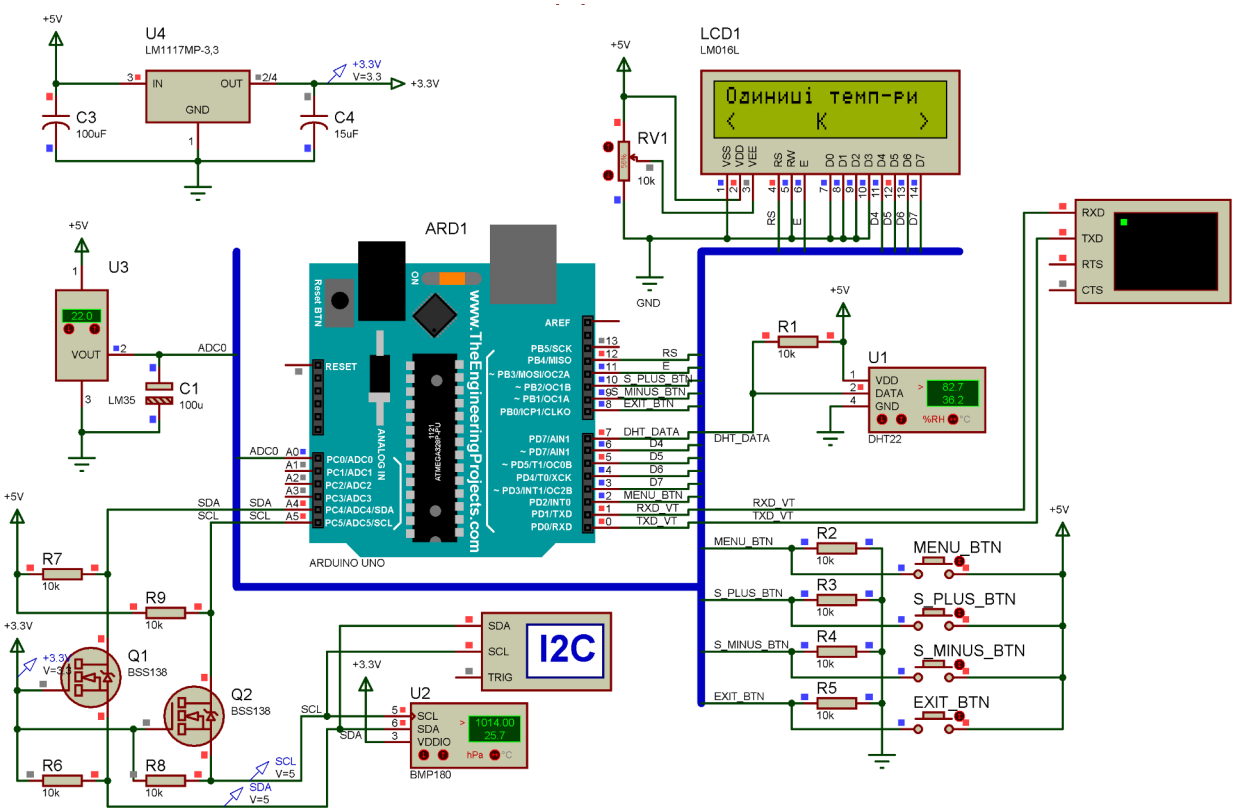

U4
LM1117MP-3,3
C3
100uF
C4
15uF
+3.3V
LCD1
LM016L
* ГОЛОВНЕ МЕНЮ *
Формат темп-ри
RV1
10k
ARD1
ARDUINO UNO
www.TheEngineeringProjects.com
U3
LM35
C1
100u
ADC0
R1
U1
DHT22
DHT_DATA
RXD
TXD
RTS
CTS
R7
R9
R2
R3
R4
R5
MENU_BTN
S_PLUS_BTN
S_MINUS_BTN
EXIT_BTN
Q1
BSS138
Q2
R6
R8
I2C
U2
BMP180
SDA
SCL

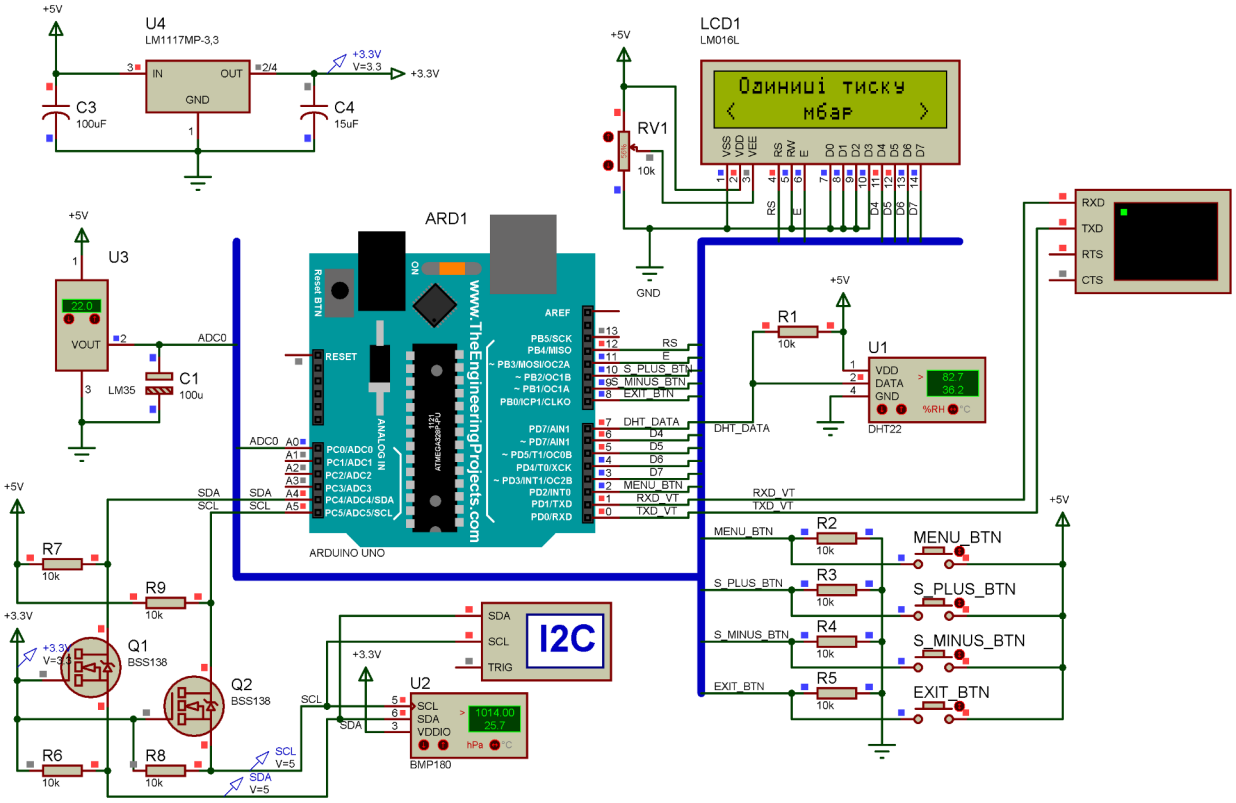

U4
LM1117MP-3,3
C3
100uF
C4
15uF
+3.3V
LCD1
LM016L
* ГОЛОВНЕ МЕНЮ *
Формат тиску
RV1
10k
ARD1
ARDUINO UNO
www.TheEngineeringProjects.com
U3
LM35
C1
100u
ADC0
R1
U1
DHT22
DHT_DATA
RXD
TXD
RTS
CTS
R7
R9
R2
R3
R4
R5
MENU_BTN
S_PLUS_BTN
S_MINUS_BTN
EXIT_BTN
Q1
BSS138
Q2
R6
R8
I2C
U2
BMP180
SDA
SCL

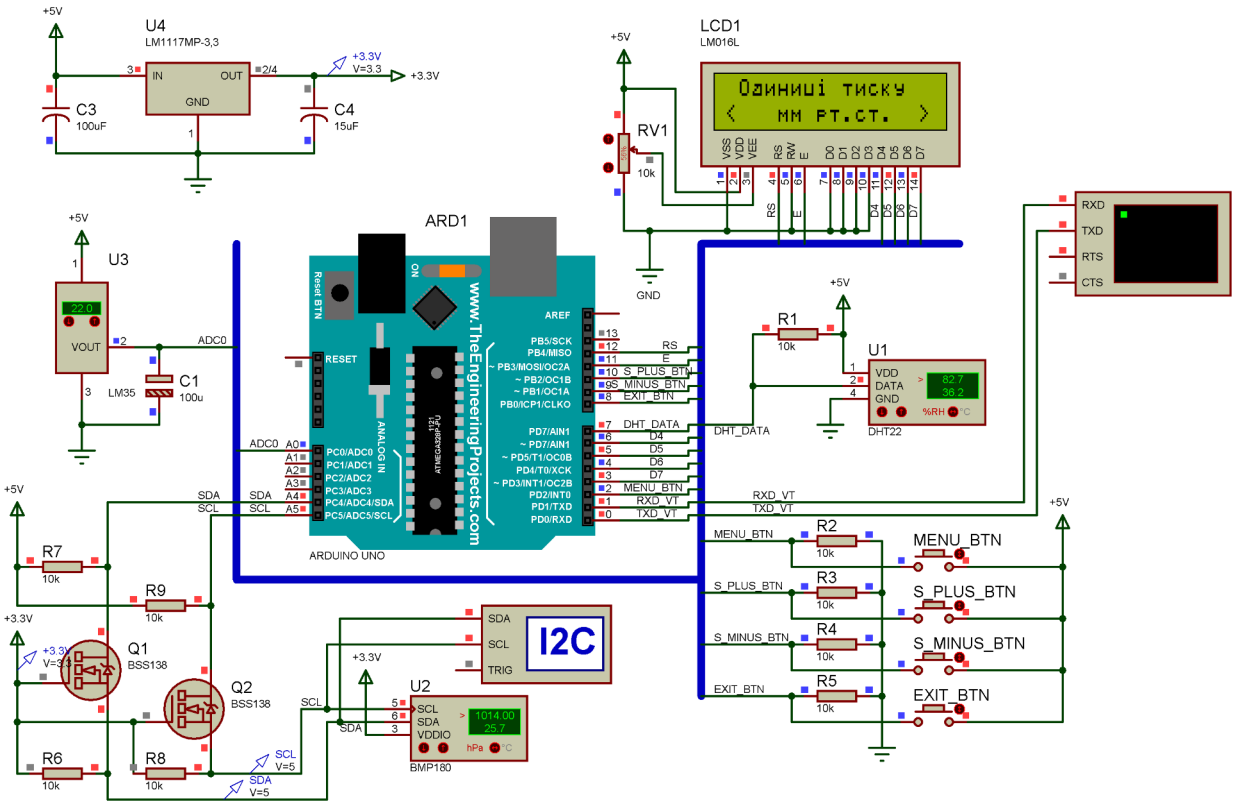

U4
LM1117MP-3,3
IN
OUT
GND
C3
100uF
C4
15uF
+3.3V
V=3.3
LCD1
LM016L
* ГОЛОВНЕ МЕНЮ *
Вихід
RV1
10k
ARD1
ARDUINO UNO
www.TheEngineeringProjects.com
U3
VOUT
LM35
C1
100u
ADC0
U1
DHT22
DHT_DATA
R1
10k
MENU_BTN
S_PLUS_BTN
S_MINUS_BTN
EXIT_BTN
R2
R3
R4
R5
RXD_VT
TXD_VT
RXD
TXD
RTS
CTS
R7
R9
R6
R8
Q1
Q2
BSS138
I2C
SDA
SCL
TRIG
U2
BMP180
hPa
°C

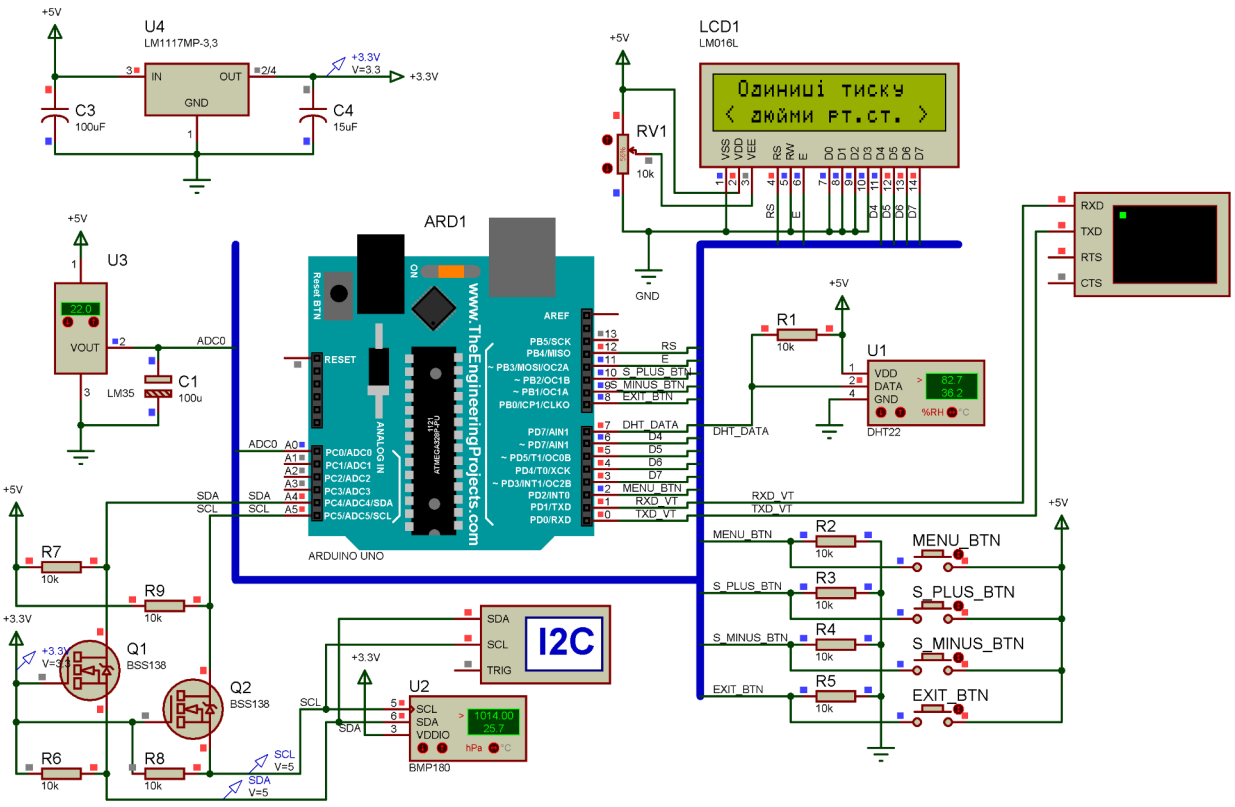

U4
LM1117MP-3,3
IN
OUT
GND
C3
100uF
C4
15uF
+3.3V
V=3.3
LCD1
LM016L
Одиниці темп-ри
< °C >
RV1
10k
ARD1
ARDUINO UNO
www.TheEngineeringProjects.com
U3
VOUT
LM35
C1
100u
ADC0
U1
DHT22
DHT_DATA
R1
10k
MENU_BTN
S_PLUS_BTN
S_MINUS_BTN
EXIT_BTN
R2
R3
R4
R5
RXD_VT
TXD_VT
RXD
TXD
RTS
CTS
R7
R9
R6
R8
Q1
Q2
BSS138
I2C
SDA
SCL
TRIG
U2
BMP180
hPa
°C

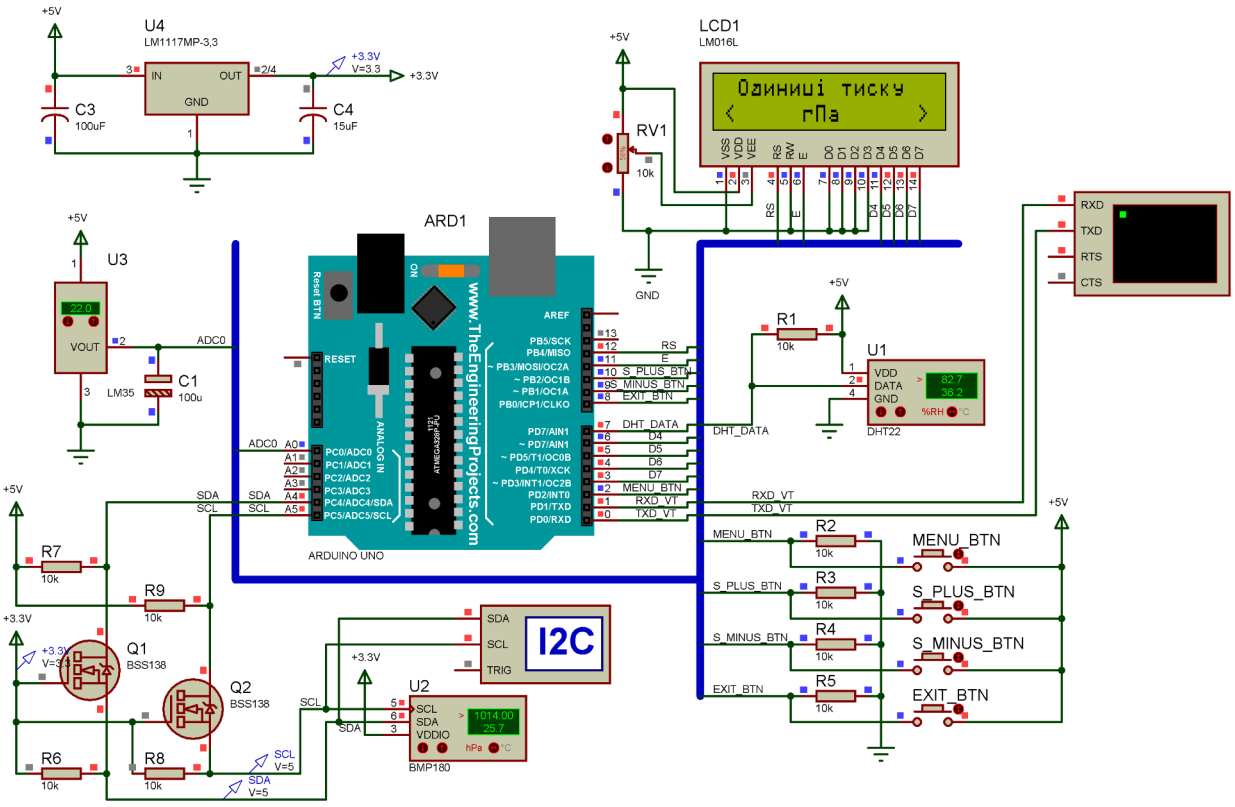

U4
LM1117MP-3,3
IN
OUT
GND
C3
100uF
C4
15uF
+3.3V
V=3.3
LCD1
LM016L
Одиниі темп-ри
< °F >
RV1
10k
ARD1
ARDUINO UNO
www.TheEngineeringProjects.com
RESET
ANALOG IN
U3
LM35
VOUT
C1
100u
ADC0
R1
U1
DHT22
DHT_DATA
RXD
TXD
RTS
CTS
MENU_BTN
S_PLUS_BTN
S_MINUS_BTN
EXIT_BTN
R2
R3
R4
R5
R7
R8
R9
R6
Q1
Q2
BSS138
SDA
SCL
I2C
TRIG
U2
BMP180
RXD_VT
TXD_VT

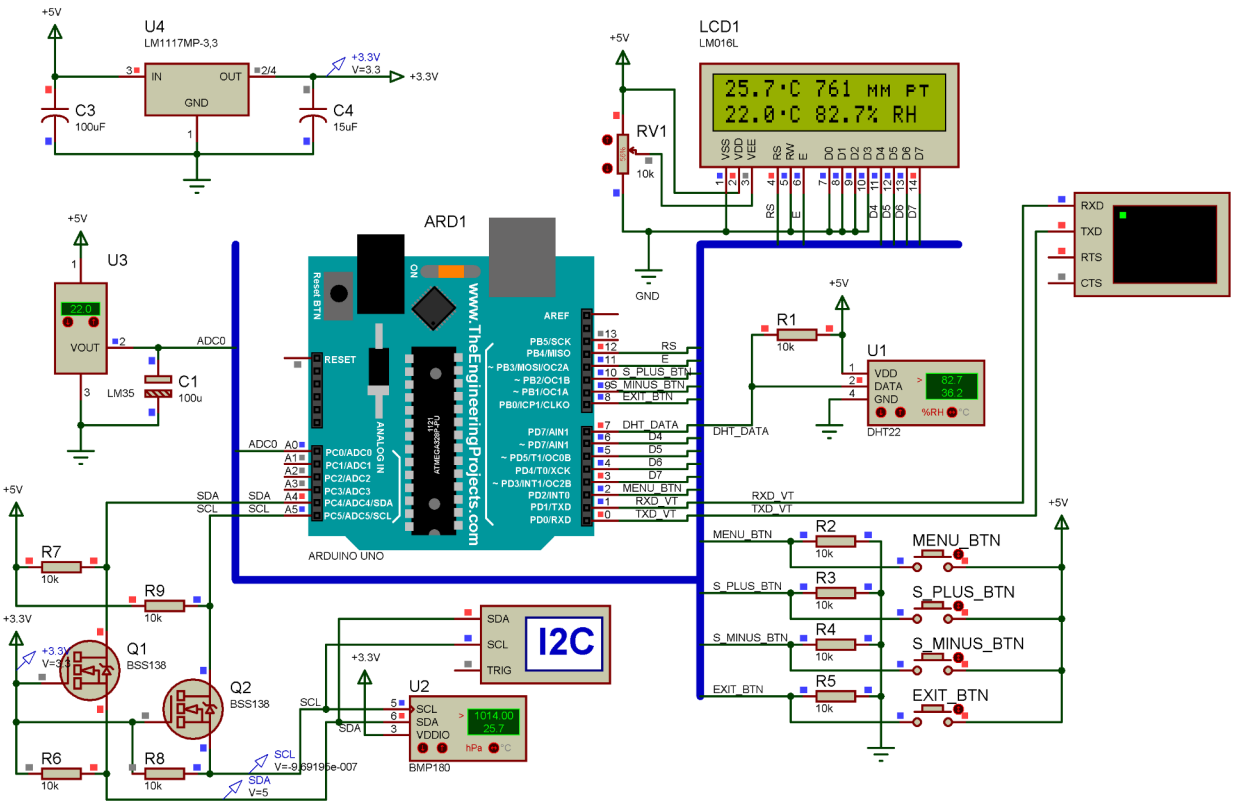

U4
LM1117MP-3,3
IN
OUT
GND
C3
100uF
C4
15uF
+3.3V
V=3.3
LCD1
LM016L
Одиниі темп-ри
< K >
RV1
10k
ARD1
ARDUINO UNO
www.TheEngineeringProjects.com
RESET
ANALOG IN
U3
LM35
VOUT
C1
100u
ADC0
R1
U1
DHT22
DHT_DATA
RXD
TXD
RTS
CTS
MENU_BTN
S_PLUS_BTN
S_MINUS_BTN
EXIT_BTN
R2
R3
R4
R5
R7
R8
R9
R6
Q1
Q2
BSS138
SDA
SCL
I2C
TRIG
U2
BMP180
RXD_VT
TXD_VT

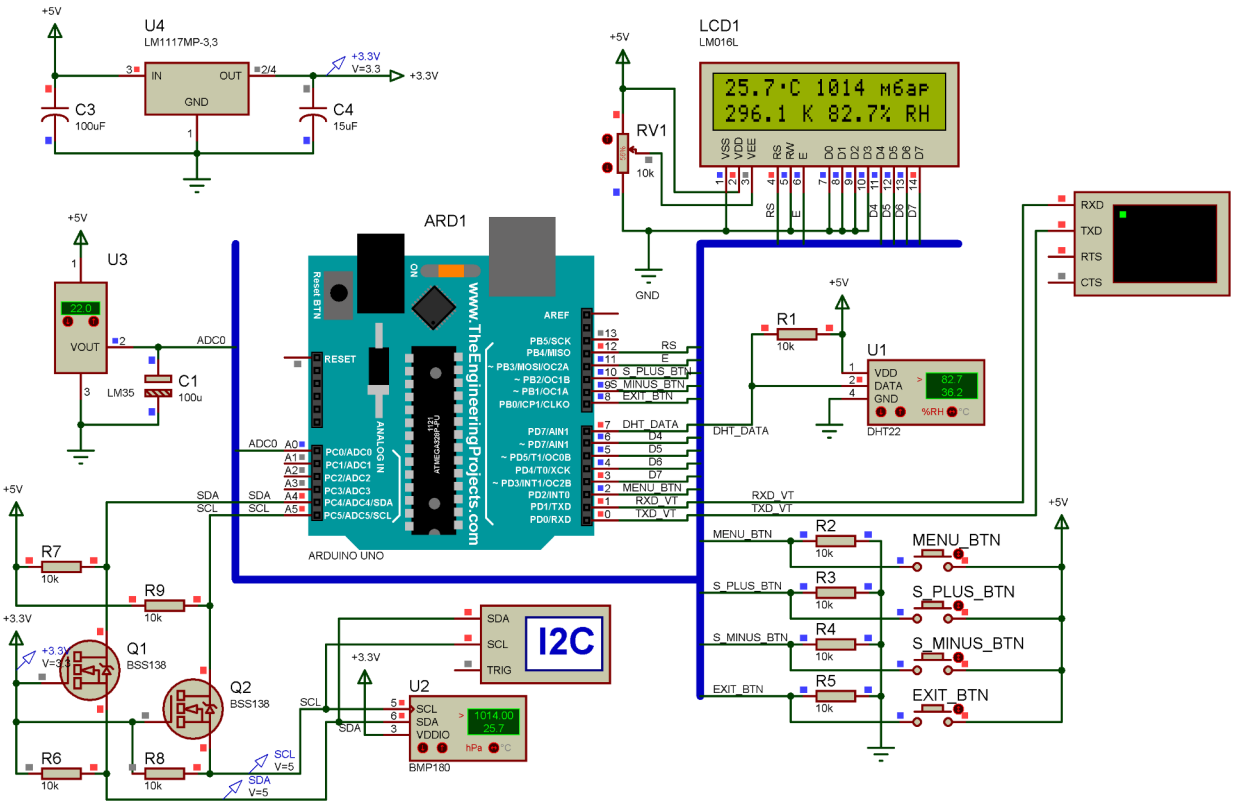

U4
LM1117MP-3,3
C3
100uF
C4
15uF
+3.3V
LCD1
LM016L
Одиниці тиску
< мбар >
RV1
10k
ARD1
ARDUINO UNO
www.TheEngineeringProjects.com
U3
LM35
C1
100u
ADC0
R1
10k
U1
DHT22
DHT_DATA
RXD
TXD
RTS
CTS
RXD_VT
TXD_VT
MENU_BTN
S_PLUS_BTN
S_MINUS_BTN
EXIT_BTN
R2
R3
R4
R5
R7
R8
R9
Q1
Q2
BSS138
SDA
SCL
I2C
TRIG
U2
BMP180
V=5

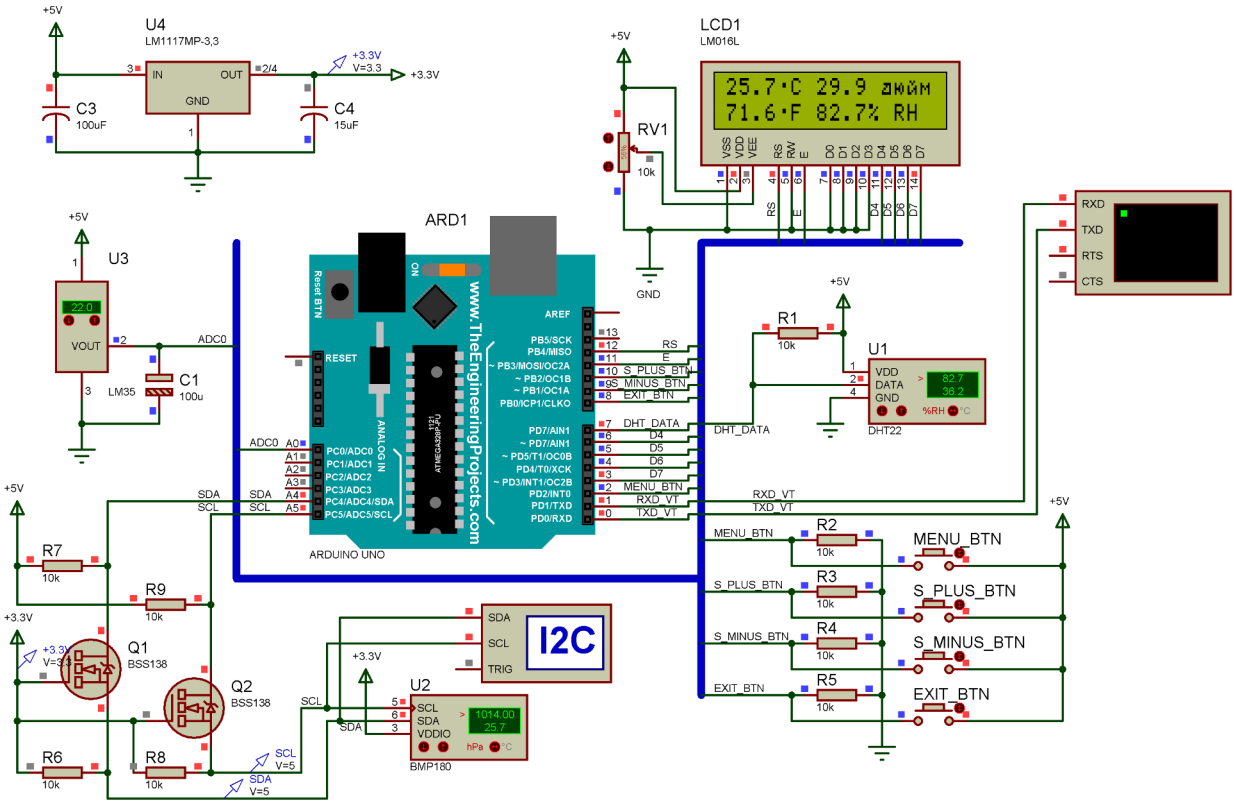

U4
LM1117MP-3,3
C3
100uF
C4
15uF
+3.3V
LCD1
LM016L
Одиниці тиску
< мм рт.ст. >
RV1
10k
ARD1
ARDUINO UNO
www.TheEngineeringProjects.com
U3
LM35
C1
100u
ADC0
R1
10k
U1
DHT22
DHT_DATA
RXD
TXD
RTS
CTS
RXD_VT
TXD_VT
MENU_BTN
S_PLUS_BTN
S_MINUS_BTN
EXIT_BTN
R2
R3
R4
R5
R7
R8
R9
Q1
Q2
BSS138
SDA
SCL
I2C
TRIG
U2
BMP180
V=5

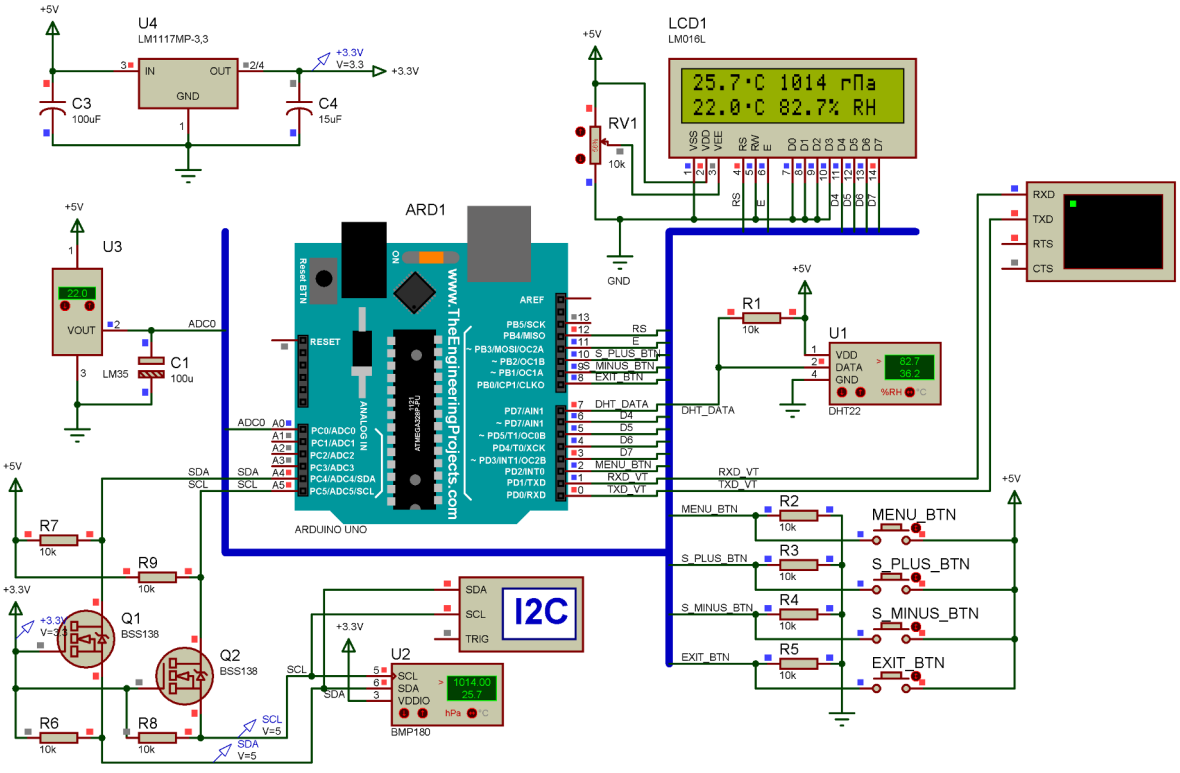

U4
LM1117MP-3,3
C3
100uF
C4
15uF
+3.3V
LCD1
LM016L
Одиниці тиску
< дюйми рт.ст. >
RV1
10k
ARD1
www.TheEngineeringProjects.com
ARDUINO UNO
U3
LM35
C1
100u
ADC0
R1
U1
DHT22
DHT_DATA
RXD_VT
TXD_VT
RXD
TXD
RTS
CTS
R7
R9
R6
R8
Q1
BSS138
Q2
I2C
U2
BMP180
MENU_BTN
S_PLUS_BTN
S_MINUS_BTN
EXIT_BTN
R2
R3
R4
R5


U4
LM1117MP-3,3
C3
100uF
C4
15uF
+3.3V
LCD1
LM016L
Одиниці тиску
< гПа >
RV1
10k
ARD1
www.TheEngineeringProjects.com
ARDUINO UNO
U3
LM35
C1
100u
ADC0
R1
U1
DHT22
DHT_DATA
RXD_VT
TXD_VT
RXD
TXD
RTS
CTS
R7
R9
R6
R8
Q1
BSS138
Q2
I2C
U2
BMP180
MENU_BTN
S_PLUS_BTN
S_MINUS_BTN
EXIT_BTN
R2
R3
R4
R5


U4
LM1117MP-3,3
C3
100uF
C4
15uF
LCD1
LM016L
25.7°C 761 мм рт
22.0°C 82.7% RH
RV1
10k
ARD1
U3
LM35
C1
100u
ADC0
ARDUINO UNO
www.TheEngineeringProjects.com
R1
10k
U1
DHT22
DHT_DATA
RXD_VT
TXD_VT
MENU_BTN
S_PLUS_BTN
S_MINUS_BTN
EXIT_BTN
R2
R3
R4
R5
R7
R9
R6
R8
Q1
BSS138
Q2
BSS138
I2C
U2
BMP180
SDA
SCL
RXD
TXD
RTS
CTS


U4
LM1117MP-3,3
C3
100uF
C4
15uF
LCD1
LM016L
25.7°C 1014 мбар
296.1 K 82.7% RH
RV1
10k
ARD1
U3
LM35
C1
100u
ADC0
ARDUINO UNO
www.TheEngineeringProjects.com
R1
10k
U1
DHT22
DHT_DATA
RXD_VT
TXD_VT
MENU_BTN
S_PLUS_BTN
S_MINUS_BTN
EXIT_BTN
R2
R3
R4
R5
R7
R9
R6
R8
Q1
BSS138
Q2
BSS138
I2C
U2
BMP180
SDA
SCL
RXD
TXD
RTS
CTS


U4
LM1117MP-3,3
C3
100uF
C4
15uF
LCD1
LM016L
25.7°C 29.9 гПа
71.6°F 82.7% RH
RV1
10k
ARD1
www.TheEngineeringProjects.com
ARDUINO UNO
U3
LM35
C1
100u
ADC0
R1
10k
U1
DHT22
DHT_DATA
RXD
TXD
RTS
CTS
R7
10k
R9
10k
Q1
BSS138
Q2
BSS138
R6
10k
R8
10k
U2
BMP180
I2C
SDA
SCL
TRIG
MENU_BTN
S_PLUS_BTN
S_MINUS_BTN
EXIT_BTN
R2
R3
R4
R5
RXD_VT
TXD_VT


U4
LM1117MP-3,3
C3
100uF
C4
15uF
LCD1
LM016L
25.7°C 1014 гПа
22.0°C 82.7% RH
RV1
10k
ARD1
www.TheEngineeringProjects.com
ARDUINO UNO
U3
LM35
C1
100u
ADC0
R1
10k
U1
DHT22
DHT_DATA
RXD
TXD
RTS
CTS
R7
10k
R9
10k
Q1
BSS138
Q2
BSS138
R6
10k
R8
10k
U2
BMP180
I2C
SDA
SCL
TRIG
MENU_BTN
S_PLUS_BTN
S_MINUS_BTN
EXIT_BTN
R2
R3
R4
R5
RXD_VT
TXD_VT