МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет електроніки та інформаційних технологій

Кафедра електроніки,
загальної та прикладної фізики

Кваліфікаційна робота бакалавра

МІКРОКОНТРОЛЕРИ ATTINY В АВТОНОМНИХ СИСТЕМАХ КЕРУВАННЯ

Студент гр. ЕП -61 В.В. Сімагін

Науковий керівник,
старший викладач К.В. Тищенко

Завідувач кафедри ЕЗПФ
д-р фіз.-мат. наук, професор І. Ю. Проценко

Суми 2020

РЕФЕРАТ

Об’єктом дослідження бакалаврської роботи є використання сімейства мікроконтролерів ATtiny для розробки системи контролю системою освітлення у громадських місцях.

Мета роботи полягає у вивченні сучасних програмних та апаратних засобів для побудови вимірювальної системи.

Актуальність роботи полягає у тому, що необхідно вдосконалювати автоматизацію різноманітних систем, таких як освітлення, особливо у громадських місцях. Це відбувається завдяки оптимізації та правильному підборі та налагодженні апаратної та програмної частини для заощадження електроенергії та інших витрат.

Під час виконання роботи використовувалися аналогові та цифрові датчики, мікроконтролер ATtiny85, персональний комп'ютер з інстальованим програмним забезпеченням Arduino Uno для плати.

У результаті виконання роботи було спроектовано і побудовано систему контролю освітлення на базі мікроконтролера ATtiny85. Розробка складалася із апаратної програмної частини. Система є придатної для промислової реалізації.

Робота викладена на 33 сторінках, у тому числі містить 12 рисунків, список цитованої літератури із 13 джерел.

КЛЮЧОВІ СЛОВА: мікроконтролер, attiny85, arduino, датчик, світлочутливий, фотоелемент, п’єзоелемент, освітлення.

ЗМІСТ

ВСТУП

При вирішенні будь-якої задачі управління здійснюється обробка інформації на рівні спеціаліста з можливим залученням засобів комп’ютерної обробки. Інформаційне забезпечення повинне забезпечити ефективність обміну інформацією між керівництвом і об’єктом управління. В склад інформаційного забезпечення звичайно включають дані, які характеризують різнобічну діяльність підприємств, нормативні та законодавчі акти, що впливають на процеси господарювання, засоби їх формалізованого опису, програмні засоби ведення і підтримки баз даних.

Як і у всіх інших нових технологій, існують деякі обмеження та проблеми, пов'язані з широким розповсюдженням її. Пристрої, підключені до IoT(інтернет речей), вразливі до обмежень джерел живлення, наприклад, оскільки мікросхеми, призначені для зв'язку з іншими "речами" в системі IoT, використовують енергію для встановлення та підтвердження з'єднань або для передачі та прийому даних.

Сьогодні впроваджується все більше досконалих автономних систем. Ми реалізовуємо технології, які самостійно будують складні відносини навколишнього світу із наданих даних, авто ідентифікації об’єктів та даних з різноманітних сенсорів. Це необхідно для отримання точного цифрового представлення дійсності для реалізації поставлених задач. Впроваджуються системи, здатні аналізувати подальший розвиток подій навколишньої дійсності, тобто які самостійно планують та реалізують завдання, не вимагаючи при цьому ніякої допомоги зі сторони, їх наділяють когнітивними здібностями людини, які в свою чергу, дозволяють їм працювати повністю автономно.

Метою роботи є огляд апаратних та програмних рішень автоматизації систем керування на базі мікроконтролерів, а також розробка системи.

Результати кваліфікаційної роботи були представлені і обговорені на Міжнародній науково-технічній конференції «Фізика, електроніка, електротехніка ФЕЕ-2020».

РОЗДІЛ 1. ОГЛЯД МІКРОКОНТРОЛЕРНОЇ ПЛАТФОРМИ

1.1. Історія створення, розвитку мікроконтролерів ATtiny та архітектури

1.1.1. Історія створення архітектури AVR.

Ідея створення та розробки нової архітектури належить двом студентам Норвезького університету науки та технології Альфу Богену та Вегарду Воллену. Вони вирішили запропонувати свою ідею випускати новий 8-бітний мікроконтролер, забезпечити його Flash-пам'яттю для програм на одному кристалі з обчислювальним ядром американській корпорації Atmel, яка була спеціалізована на виготовленні чіпів з Flash-пам’яттю. Корпорація ухвалила їдею студентів, проінвестувала розробку і в 1996 році був виготовлений тестовий мікроконтролер AT90S1200, а в другій половині 1997 року корпорація Atmel приступила до серійного виробництва нового сімейства мікроконтролерів. Нове ядро було запатентовано і отримало назву **AVR** . Існує кілька трактувань даної абревіатури. Хтось стверджує, що це **A**dvanced **V**irtual **R**ISC, інші вважають, що не обійшлося тут без A lf Egil Bogen V egard Wollan R ISC.

1.1.2. Структура архітектури та система команд

Мікроконтролери AVR основані на гарвардській архітектурі (Рисунок 1.1) (програма і дані знаходяться в різних адресних просторах) та систему команд, близьку до ідеології RISC.

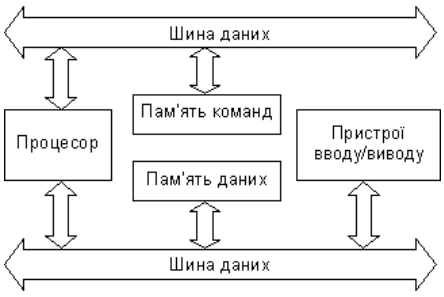


Рисунок 1.1 – Структура гарвардської архітектури. Адаптовано з роботи [1]

Система команд мікроконтролерів AVR дуже розвинена і налічує в різних моделях від 90 до 133 різних інструкцій.

Більшість команд займає лише 1 осередок пам'яті (16 біт).

Більшість команд виконується за 1 такт.

Всі команди мікроконтролерів AVR можна розбити на кілька груп [1]:

- команди логічних операцій;
- команди арифметичних операцій і команди зсуву;
- команди операції з битами;
- команди пересилання даних;
- команди передачі управління;
- команди управління системою.

Управління периферійними пристроями здійснюється через адресний простір даних. Для зручності існують «скорочені команди» IN / OUT.

Сімейство мікроконтролерів налічує в собі 3 стандартні сімейства:

tiny AVR (**ATtiny** xxx):

- флеш-память до 16 Кб; SRAM до 512 б; EEPROM до 512 б;
- число ліній введення-виведення 4-18 (загальна кількість висновків 6-32);
- обмежений набір периферійних пристроїв.

Mega AVR (**ATmega** xxx):

- флеш-память до 256 Кб; SRAM до 16 Кб; EEPROM до 4 Кб;
- число ліній введення-виведення 23-86 (загальна кількість висновків 28-100);
- апаратний помножувач;
- розширена система команд і периферійних пристроїв.

XMEGA AVR (**ATxmega** xxx):

- флеш-память до 384 Кб; SRAM до 32 Кб; EEPROM до 4 Кб;
- чотирьохканальний DMA -контролер;
- інноваційна система обробки подій.

Мікроконтролери ATtiny є наймолодшими в лінійці AVR: в них найменше число ліній введенняя-виведення, менший об’єм пам’яті та обмежений набір периферійних пристроїв в порівнянні з мікроконтролерами mega/XMEGA, проте це перекривається меншою коштовністю і малими розмірами. Більше того, tiny МК мають таку ж саму продуктивність, що і старші моделі сімейства Mega. Вони стануть кращим вибором для конструювання пристроїв, в яких нема необхідності у великій кількості периферійних пристроях, де розмір і ціна мають велике значення.

Мікроконтролери tinyAVR оптимізовані для систем, в яких потрібна висока продуктивність, ефективне використання енергії, простота застосування і компактність. Всі МК tinyAVR мають загальну архітектуру і сумісні з іншими пристроями AVR. Оскільки в ці мікроконтролери вбудовані АЦП, пам'ять EEPROM і детектор зниження напруги живлення, для проектування систем не потрібні додаткові зовнішні компоненти. Мікроконтролери tinyAVR мають флеш-пам'ять і вбудований інструмент налагодження, що прискорює і робить безпечним внутрісхемне оновлення програмного забезпечення, знижує вартість цього процесу і веде до скорочення часу виведення готового виробу на ринок. Вони унікальним чином поєднують в собі мініатюрність, обчислювальну потужність, високу продуктивність аналогової частини і інтеграцію на системному рівні.

За допомогою даних контролерів є багато проектів, серед яких розумний будинок (автоматизація пристроїв за певним алгоритмом та використанням різноманітних датчиків, які виконують задані операції), додавши до мікроконтролера датчик температури та елемент живлення, сконструювати пристрій, який буде протягом певного часу вимірювати температуру та надсилати дані на смартфон та багато інших, їх використовують також у портативних навігаторах, іграшках, спортивних гаджетах, побутовій техніці, інтелектуальних датчиках та ін.

1.2. Огляд контролера ATtiny85 та платформ на його основі

Системи простої автоматизації, зазвичай, потребують невеликих обчислювальних потужностей, і повинні бути компактними. Звичайно, існують сумісні з Ардуіно плати, але вони для багатьох проектів будуть надмірними. Не дивно, що появу ATtiny85 зустріли з натхненням все, хто займається створенням мініатюрних пристроїв.

Мікроконтролер ATtiny85 цілком функціональний, покажемо його характеристики і можливості:

- для програмного коду передбачено 8 КБ пам'яті;
- для виконуваного коду (ОЗУ, SRAM, RAM) зарезервовано 512Б;
- пам'ять даних (EEPROM) 512 Б
- наявність 6 цифрових пинов, в реальності краще використовувати 5 (залишивши 1 під RESET);
- два виходи PWM і 4 ADC (дозвіл 10-bit);
- частота від 1 до 20 МГц.

ATtiny85 пропонується в корпусах: MLF (WQFN) 20M1 20dip8, PDIP 8P3 8, SOIC (208mil) 8S2 8.

ATtiny випускається в двох корпусах (Рисунок 1.2), в залежності від модифікації джерело живлення може бути від 1,8 до 5,5В. В економічному режимі прилад споживає від 0,1 мкА, в мікроконтролері реалізовано апаратне переривання. ATtiny85 в порівнянні з іншими мікроконтроллерами цього сімейства має максимальну пам'ять, що робить його більш затребуваним на ринку.

Мікроконтролер ATtiny85 на перший погляд може здатися більш дешевої заміною Arduino. Він також здатний тривалий час працювати в компактних пристроях від однієї батарейки. Однак, на відміну від Ардуіно, контролер ATtiny85 спочатку не може підключатися до комп'ютера напряму, для цього був потрібен програматор.

Рисунок 1.2 – Digispark Attiny85 з варіантами портів USB тип А (а) та MiniUSB (б). Адаптовано з роботи [6]

У Digistump запропонували програмувати ATtiny85 через USB, для цього потрібно мінімальну апаратну обв'язку, спеціальний бутлоадер і плагін для Arduino IDE. Цей мінімальний набір засобів прекрасно працює, однак в мікроконтролері повністю відсутній захист. Якщо переплутати полярність при підключенні без USB-хаба, то роз'єм комп'ютера може згоріти. Ще один мінус – плата працює не з усіма сумісними USB пристроями.

Digistump пропонує свою розробку безкоштовно, не дивлячись на те, що плата стає своєрідним еталоном схемотехніки. В результаті з'явилося чимало аналогів Digispark, що розрізняються часом тільки портами USB. Серед цього розмаїття найбільш раціональної є плата з портом мікро-USB. Вона укомплектована ATtiny85 20 SU в корпусі SOIC8. Цей мікроконтролер має максимальну частоту в 20 МГц і працює при напрузі від 2,7 до 5,5 В. При розмірі плати 18х22 мм її укомплектували 9 пінами.

Завдяки наявності стабілізатора, мікроконтролер можна живити двома шляхами: через пін напругою 5В і через стабілізатор 10 В. При цьому розробники плати стверджують, що останній зможе витримати напругу і в 35В, але гарантій в тривалості роботи при таких значеннях немає ніяких. Переплутати піни на

платі ATtiny85 20 SU складно навіть новачкові, так як вони з лицьового боку підписані цифрою, а зі зворотного боку виконуваних функцій і протоколу.

1.3. Початок роботи з ATtiny85 в середовищі програмування Arduino IDE

Перш за все, нам необхідно встановити драйвери для Attiny85 Digispark. Система зробить установку і настройку драйвера автоматично. Після цього необхідно встановити спеціальний плагін в Arduino IDE.

Для установки плагіна (Рисунок 1.3) потрібно перейти в «Файл» - «Налаштування» і вписати в полі «Додаткові посилання для менеджера плат» наступний рядок: http://digistump.com/package_digistump_index.json, після чого натиснути кнопку «застосувати». Відбудеться встановлення модулів, який дозволить нам встановити плату в систему.

Потім необхідно перейти в «Меню» – «Інструменти» – «Менеджер плат», після чого знайти там Digistump AVR Boards і установити.Після завершення установки в менеджері плат варто вибрати плату Digispark (Default - 16,5mhz), яку радять для початківців схемотехніки. При роботі з ATtiny85 20 SU немає необхідності підключати мікроконтролер до комп'ютера до завантаження прошивки [2].

При створенні прототипів на базі плати ATtiny85 20 SU слід враховувати її відмінності від Arduino. Почнемо з першого піна (PIN 1): він в ATtiny85 використовується для необхідного при прошивці сигналу Reset і при перенесенні коду його не варто використовувати. Якщо цього не вдалося уникнути, то приготуйтеся користуватися складним програматором. Наступні піни, що вимагають до себе особливого ставлення – 3 і 4 (PIN 3, PIN 4). До PIN 3 підключається резистор 1,5 К, що в результаті забезпечує більш високі значення, ніж 0. Третій пін, поряд з четвертим, застосовуються для підключення порту USB. Якщо планується використовувати ці піни для налагодження периферії, то перед завантаженням прошивки їх потрібно відключити.

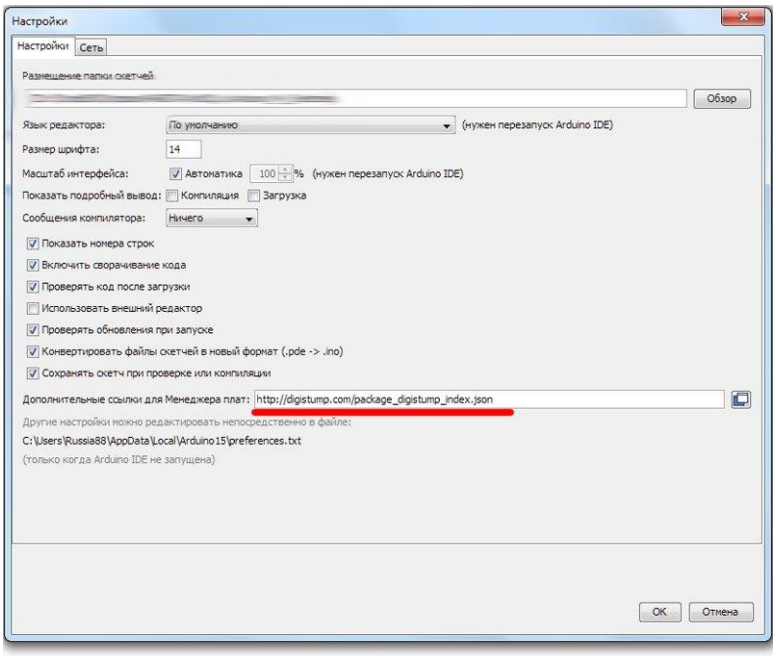


Рисунок 1.3 – Налаштування Arduino IDE для роботи з ATtiny85. Адаптовано з роботи [2]

Плата ATtiny85 20 SU, на відміну від Arduino, підтримує не всі команди і бібліотеки, і її пам'ять в 8К відрізняється від стандартної версії в 6К. При розробці пристроїв необхідно пам'ятати про наступні крайніх значеннях: мінімальне споживання 13,38мА, максимальне близько 30мА.

1.4. Приклади практичного застосування ATtiny85

Сигналізатор відкритих дверей. Одним з найбільш нагальних прикладів застосування пристрою на базі ATtiny85 є сигналізатор дверного замка. Будь-який з жителів міської квартири стикався з проблемою не закриття для вхідних дверей. Для уникнення неприємних наслідків, що можуть виникнути внаслідок незачинених дверей, досить зібрати невеликий пристрій на базі ATtiny85. Для цього крім мікроконтролера знадобиться мікроперемикач, п'єзокерамічна пищалка і батарейка. Для роботи пристрою необхідно написати простий код, який буде включати звук через 3-5 хвилин після початку роботи

мікроконтролера. Для включення контролера необхідний мікроперемикач, який буде взаємодіяти з ригелем дверного замка. Якщо ригель не тисне на вмикач, то пристрій спрацьовує через заданий тайм аут. Звук попередить, що замок вхідних дверей не закритий.

При включенні пристрою (при відкритті замка) лунає короткий писк, що попереджає про те, що сигналізатор працює, а його батарейка все ще генерує енергію. Сигналізатор відкритих дверей має компактні габарити, легко поміщається в дверній коробці.

Автоматичне підсвічування гардероба. У гардеробі постійно не вистачає світла, тому що автоматично включається освітлювальний прилад не завадить нікому. Найкраще, якщо він буде самостійно оцінювати освітленість і включатися без втручання господаря будинку.Крім мікроконтролера ATtiny85 для створення подібного пристрою знадобитися датчик світла, батарейний відсік, передавач на 433 МГц і датчик руху.

Світильник з мікро контролером розміщується на одній з полиць гардероба, він займає мало місця, але неймовірно функціональний. ATtiny85 починає роботу після сигналу від датчика руху. Включившись, він оцінює рівень освітленості гардероба і при нестачі світла включає світильник. При відсутності руху ATtiny85 вимикає світильник, тайм аут можна налаштувати на будь-який час, оптимальним варіантом є 1 хвилина. Це дозволяє економити енергію, яка в сплячому режимі споживається не менше 60 мкА. В процесі роботи пристрій споживає 8-9 мА.

Датчик контролю протікання. Звичайно, подібних систем чимало у вільному доступі, але більшість з них спрацьовує вже при затопленні. У ряді випадку сигнал від такого датчика виявляється запізнілим. Теоретично, при протіканні повинна швидко збільшуватися вологість, так як ситуація розвивається в невеликому за обсягом приміщенні. За основу приладу з цієї причини був узятий популярний датчик вологості і температури DHT-11.

Мінусом даного пристрою, як датчика протікання, є поріг спрацьовування, оскільки вологість в ньому протягом доби може коливатися в значних межах без

будь-яких протікань.Подібний результат дослідів засмутив, але не сильно, адже пристрій може прекрасно працювати як метеодатчик, передаючи дані про динаміку рівня вологості.

1.5. Керування завантаженням із використанням контролера ATtiny85

Останнім часом активно просуваються пристрої розумного будинку, однак, однією з основних проблем комутації силових навантажень в таких пристроях є збереження кнопкового або іншого управління [3]. Такі рішення тягнуть за собою ситуацію із надмірним використанням доступних виводів мікроконтролера, що обмежує як область їх використання, так і кількість підключених датчиків та елементів керування.

1.5.1. Управління навантаженням 220 В за допомогою симистора і мікроконтролера

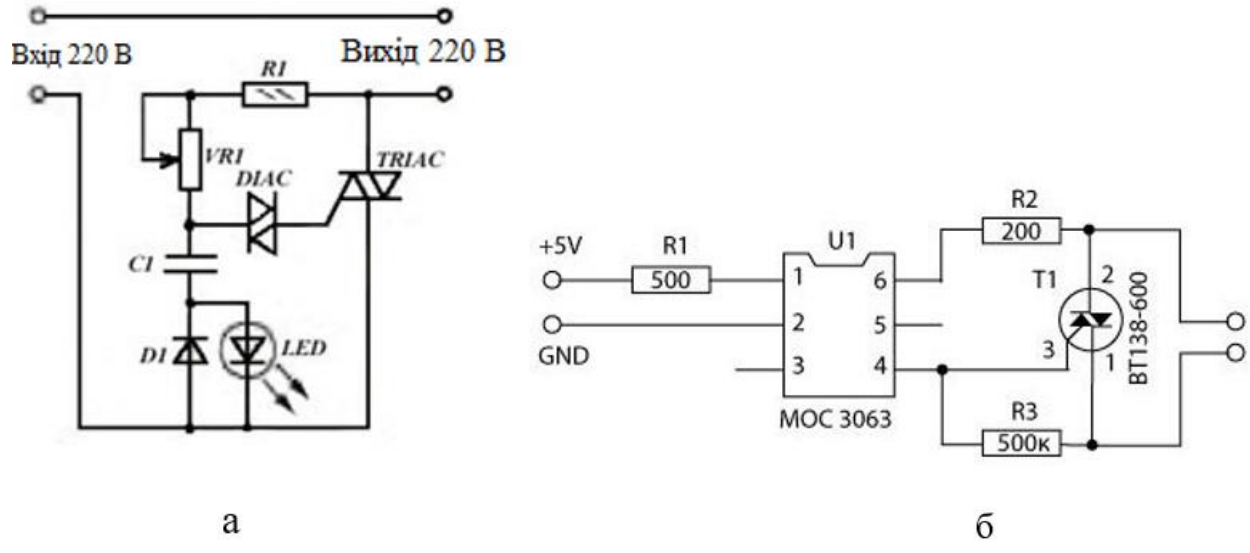
Для системи «Розумний будинок» основним завданням є управління побутовими приладами з керуючого пристрою будь то мікроконтролер типу ATTiny, Ардуіно, мікрокомп'ютера типу Raspberry PI або будь-який інший. Але зробити цього напряму не вийде, давайте розберемося як управляти навантаженням 220 В.

Для управління ланцюгами змінного струму засобів мікроконтролера недостатньо з двох причин:

1. На виході мікроконтролера формується сигнал постійної напруги.
2. Струм через пін мікроконтролера зазвичай обмежений величиною в 20-40 мА.

Ми маємо два варіанти комутації за допомогою реле або за допомогою симистора. Симистор може бути замінений двома включеними зустрічно-паралельно тиристорами (це і є внутрішня структура симистора). Давайте докладніше розглянемо це.

Тиристор працює наступним чином: коли до тиристора прикладена напруга в прямому зміщенні (плюс до анода, а мінус до катода) струм через нього проходити не буде, поки ви не подасте керуючий імпульс на керуючий електрод. На відміну від транзистора тиристор є напівкерованих напівпровідникових ключем. Це означає, що при знятті керуючого сигналу струм через тиристор продовжить протікати, тобто він залишиться відкритим. Щоб він закрився потрібно перервати струм в ланцюзі або змінити полярність прикладеної напруги. Це означає, що при утриманні позитивного імпульсу на керуючому електроді потрібно тиристор в колі змінного струму буде пропускати тільки позитивну півхвилю. Симистор може пропускати струм в обох напрямках, але тому що він складається з двох тиристорів підключених назустріч один одному. Керуючі імпульси по полярності для кожного з внутрішніх тиристорів повинні відповідати полярності відповідної напівхвилі, тільки при виконанні такої умови через симистор буде протікати змінний струм. На практиці така схема реалізована в симісторного регулятора потужності (Рисунок 1.4 а).



а б

Рисунок 1.4 – Принципова схема симісторного регулятора потужності (а) та оптосимісторного драйвера (б). Адаптовано з роботи [7]

Оскільки мікроконтролер видає сигнал тільки однієї полярності, для того щоб його узгодити потрібно використовувати драйвер побудований на оптосімісторі (Рисунок 1.4 б). Сигнал включає внутрішній світлодіод оптопари, вона відкриває симистор, який і подає керуючий сигнал на силовий симистор Т1. Як оптодрайвер може бути використана оптична мікросхема MOC3063 і подібні, наприклад, MOC3041.

Схема може бути реалізована і без оптодрайвера, де узгодження організовано через діодний міст (Рисунок 1.5), але в ній, на відміну від попереднього варіанту немає гальванічної розв'язки. Це означає, що при стрибку напруги міст може пробити і висока напруга потрапить на вивод мікроконтролера, а це може привести до виходу його з ладу.

При включенні / виключенні потужної навантаження, особливо індуктивного характеру, типу двигунів і електромагнітів виникають сплески напруги, тому паралельно всім напівпровідникових приладів потрібно встановлювати снаберний RC ланцюг.

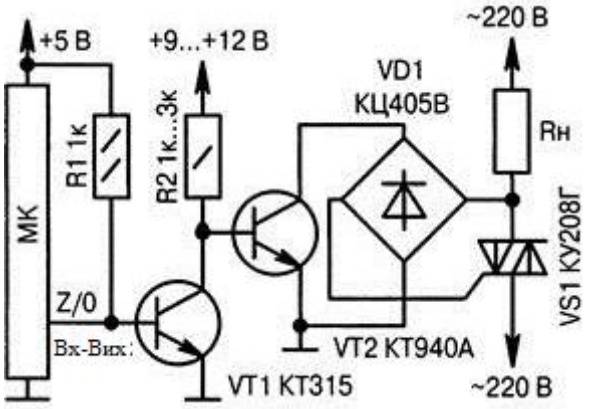


Рисунок 1.5 – Принципова схема симісторного драйвера на діодному мості. Адаптовано з роботи [6]

1.5.2. Управління навантаженням за допомогою реле

Для управління реле з мікроконтролера потрібно використовувати додатковий транзистор для посилення струму та діод, який потрібен для гасіння

сплесків ЕРС самоіндукції в індуктивності, це потрібно щоб транзистор не вийшов з ладу від високої напруги.

Таку схему можна зібрати самому, або придбати готовий модуль або цілий Шилд з реле для Ардуіно Схема підключення навантаження на напрузі 220 В до контролера через реле показана на рисунку 1.6.

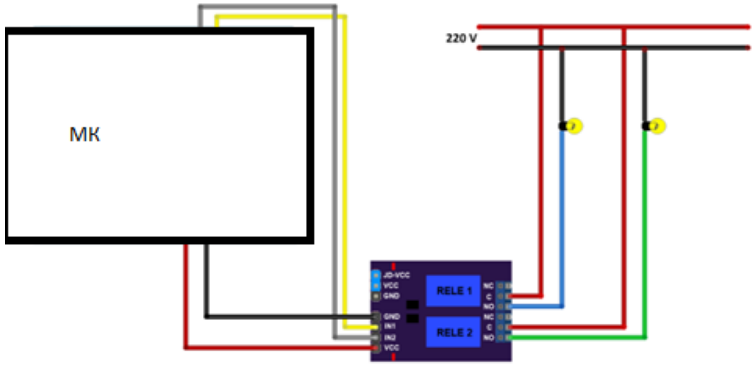


Рисунок 1.6 – Принципова схема підключення реле до мікроконтролера. Адаптовано з роботи [3]

Головне завдання - забезпечити потрібні напругу і струм для управління симистором або реле і гальванічна розв'язка ланцюгів управління і силовий ланцюга змінного струму.

Крім безпеки для мікроконтролера, таким чином, ви підстраховує себе, щоб при обслуговуванні не одержати електротравму.

Описані вище схеми можна використовувати і для управління потужними пускателями і контакторами. Сімістори і реле в такому випадку виступають в ролі проміжного підсилювача і узгоджувача сигналів. На потужних комутаційних приладах великі струми управління котушкою і залежать безпосередньо від потужності контактора або пускача.

1.5.3. Управління навантаженням за допомогою Mosfet транзисторів

Mosfet або МОП-транзистор це потужний засіб для управління навантаженням. Такі транзистори бувають N і P типів (Рисунок 1.7), N-канальні забезпечують більші можливості і потужності по керуванню силовими навантаженнями.

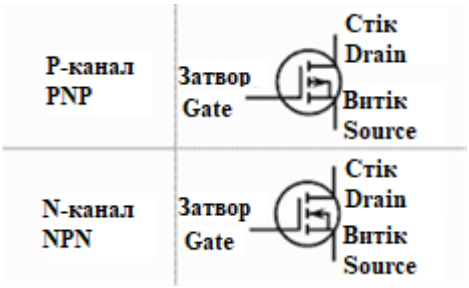


Рисунок 1.7 – Типи Mosfet транзисторів. Адаптовано з роботи [9]

При підключенні транзистора до мікроконтролері у режимі ключа (Рисунок 1.8) ми отримуємо схему, аналогічну за механікою роботи із реле, але з відсутністю механічного перемикання. Програмно така схема керується TTL-сумісним сигналом і проста в реалізації.

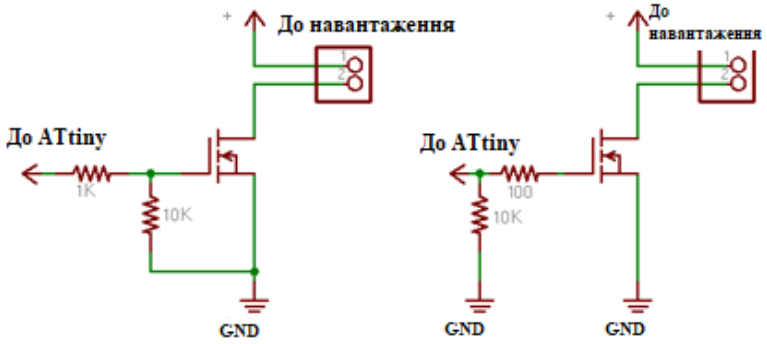


Рисунок 1.8 – Підключення Mosfet транзисторів до мікроконтролера у режимі ключа. Адаптовано з роботи [9]

Якщо виникає необхідність й плавно вмикати / вимикати силове навантаження, або вмикати його не на всю потужність, а тільки на половину, можна з вивода мікроконтролера керувати шімом, а між затвором і витоком включити ще конденсатор близько 300 микрофарад. Це потрібно щоб відкрити

мосфети на половину. Однак це підійде тільки для малопотужного навантаження, томущо напіввідкритий мосфет має великий внутрішній опір і як наслідок буде дуже сильно грітись.

Також слід мати на увазі, що затвор (gate) у польового транзистора має певну ємність і є в якійсь мірі конденсатором. Так що в момент перемикання через затвор проходять великі струми, які може не витримати контролер. Для цього потрібен резистор між затвором і піном мікроконтролера.

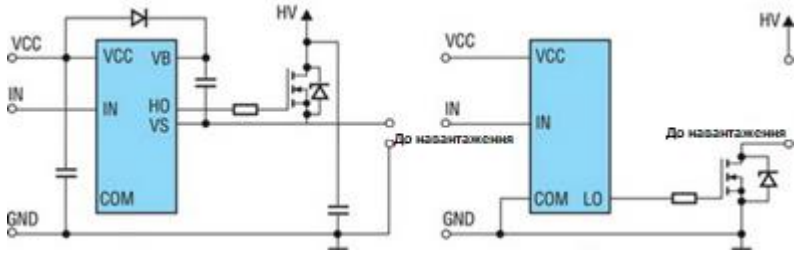


Рисунок 1.9 – Варіанти драйверів для підключення навантаження через Mosfet транзистор. Адаптовано з роботи [9]

Підключення Mosfet до мікроконтролера зазвичай здійснюється із використанням спеціальних драйверів (Рисунок 1.9). Суть в тому, що драйвер потрібен якраз для узгодження п'яти вольт з виводів мікроконтролерів з рівнями, необхідними для управління затворами мосфетів.

У разі якщо, із якихось параметрів присутній ШІМ контролер не задовільняє потреби у керуванні обраним пристроєм, то слід використовувати зовнішній ШІМ драйвер, краще високопродуктивний, наприклад типу TC4420.

1.6. Керування двигунами постійного струму за допомогою мікроконтролерів

1.6.1. Керування мініатюрними двигунами

Управління маленьким двигуном може бути може здійснюватися досить просто. Якщо двигун досить маленький і малопотужний, то він може бути безпосередньо з'єднаний з виводом контролера, і просто змінюючи рівень сигналу, що управляє від логічної одиниці до нуля ми зможемо його контролювати. Це не є стандартним способом підключення двигунів до мікроконтролерів, а є лише моделюванням можливості керування та програмних прийомів.

Для прикладу, підключимо мініатюрний вібродвигун (наприклад від мобільного телефону) до нашого контролера. У вібродвигуна є два дроти живлення. Необхідно з’єднати один його провід з нульовим виводом (GND) контролера, не має значення який із двох виводів це буде. Далі потрібно підключити резистор опором 220 Ом між дискретним виходом контролера і непідключеним проводом двигуна. Підключення резистора обмежить струм і гарантує нам цілісність і захист контролера, так як вони не проектувалися для прямого контролю електродвигунами без перетворювачів.

Текст програми, який скетч запустить моторчик на 1 секунду, і зупинить його на такий же час, написаної в середовищі Arduino IDE матиме наступний вигляд:

```
int motorPin = 2; // Резервуємо номер дискретного виходу
void setup () {
    pinMode (motorPin, OUTPUT); //Позначаємо другий
    дискретний вивод як вихід
}
void loop () {
    digitalWrite (motorPin, HIGH); // Включаємо двигун
    delay (1000); // Чекаємо 1000 мс (1 с)
    digitalWrite (motorPin, LOW); // Вимикаємо двигун
    delay (1000); // Чекаємо 1000 мс (1 с)
}
```

Розглянемо як працює описаний вище код. Кожного разу, коли програма буде подавати логічну одиницю на наш вихід, струм буде протікати через резистор, через двигун (М), і на землю. Якщо М дійсно малопотужний, він почне обертатися, якщо це стандартний двигун постійного струму, або почне вібрувати, якщо це вібродвигун. Резистор дуже важливий для цієї схеми. Кожен дискретний вихід більшості мікроконтролерів розрахований на струм до 40 мА, при чому рекомендується не перевищувати 20 мА. Вибране значення резистора 220 Ом обмежить струм до 22 мА, і тому, що М включений з ним послідовно, струм буде навіть менше. Кілька двигунів можуть бути підключені на різні цифрові виводи плати контролера. Наприклад, виходи 2, 3, і 4 можуть незалежно управляти різними трьома електродвигунами. Кожен дискретний вивод може керувати окремим двигуном.

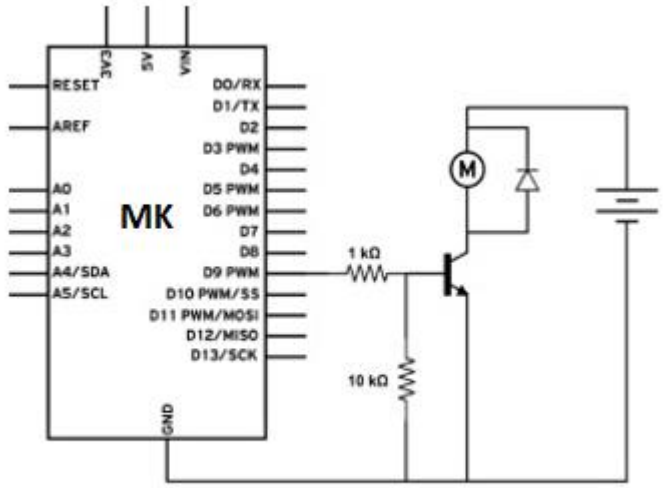
Кожен електродвигун постійного струму є котушкою індуктивності. Коли ми знімаємо з нього струм, або коли ми обертаємо М вручну, він буде генерувати зворотна напруга. Що може підпалити підключений до нього електронний компонент. Щоб уникнути цього, ми можемо підключити діод між дискретним виходом і виводом живлення 5В. Всякий раз, коли М буде віддавати паразитну зворотню напругу, діод буде з'єднувати його з плюсом живлення.

1.6.2. Керування електродвигуном за допомогою транзисторів

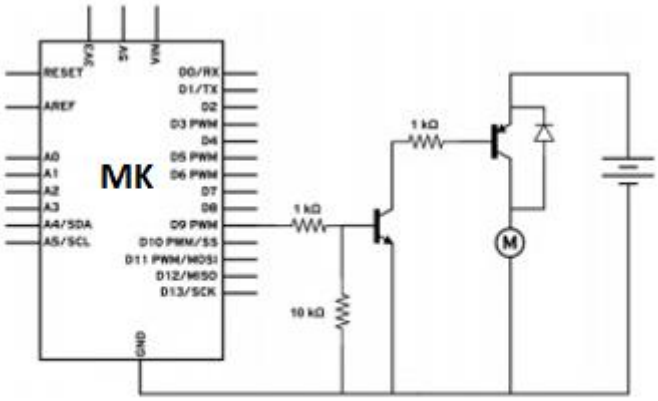
Ми звичайно можемо управляти мініатюрним електродвигуном безпосередньо підключивши його до виходу контролера, однак, дискретний вихід не зможе живити двигуни, які споживають більше 40 мА. Вихід полягає у використанні простого підсилювального пристрою, транзистора, щоб мати можливість керувати електродвигунами постійного струму будь-якої потужності. Розглянемо на прикладі, як управляти великими електродвигунами, використовуючи два транзистора npn і pnp структури. Для npn транзистора схема управління матиме вигляд, як показано на рисунку 1.10 а.

Схема буде працювати наступним чином. Коли ми подаємо логічну одиницю на вихід контролера, струм проходить від виводу через базу транзистора NPN, що змушує струм проходити і через інші дві ніжки транзистора. Коли ми виставляємо нуль на виході, струм не йде через базу і не буде проходити через інші дві ноги.

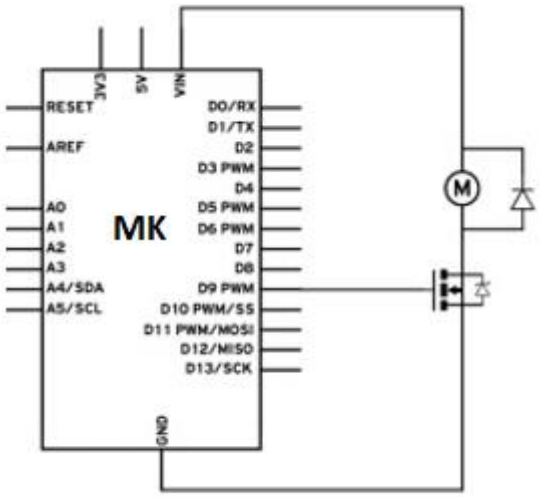
Транзистори цікаві в тому, що з дуже малим струмом бази, ми можемо контролювати дуже великий струм через колектор до емітера. Звичайний коефіцієнт посилення для транзистора складає близько 200. Це означає, що для струму бази 1 мА, транзистор через колектор до емітера пропустить 200 мА. Важливим компонентом схеми є діод, про який не варто забувати. Як вже було сказано вище, двигун має індуктивну складову, яка може генерувати великі сплески напруги, небезпечні для транзистора. Діод гарантує, що всі паразитні струми від двигуна загасяться на ньому, а не на транзисторі [4].



а



б



в

Рисунок 1.10 – Схеми управління двигуном постійного струму з використанням npn (а), pnp (б) та Mosfet (в) транзистора. Адаптовано з роботи [5, 9]

База транзистора дуже чутлива. Навіть дотик її пальцем може провернути електродвигун. Щоб уникнути небажаних шумів і непередбачуваного запуску двигунів використовується підключення підтягуючого резистора близько 10 кОм на базі.

Логіку роботи схеми з pnp транзистором (Рисунок 1.10 б) зрозуміти дещо складніше. Вона використовує той же принцип, але в зворотному напрямку. Струм тече від бази до цифрового виводу контролера. Інша важлива відмінність в тому, що pnp-транзистор встановлений між плюсом джерела живлення і контрольованим нами навантаженням. Навантаження ж, в даному випадку, це двигун, буде підключено між колектором pnp-транзистора і землею.

Ключовий момент на замітку розробникам ще й в тому, що при використанні транзисторів pnp з Arduino максимальна напруга на емітер 5 В, і при цьому ми на двигун не зможемо подати більше, ніж 5 В. Якщо використовувати зовнішнє джерело живлення для живлення двигунів з великим напруженням ніж 5В, на базі з'явиться потенціал вище п'яти вольт і Arduino підгорить. Одне з можливих рішень показане якраз і використане в схемі на рисунку 1.10 б.

Більш простим та зручним у використанні та такі, що можуть забезпечити набагато більшу потужність управління є Mosfet транзистори. Функціонально їх використовують точно так же, як і звичайні транзистори. При подачі напруги на накоїв, струм буде проходити з витоку на стік в разі N-канального МОП-транзистора. Схема керування двигуном за допомогою такого транзистора показана на рисунку 1.11 в. Програмно керування двигуном за допомогою Mosfet транзистора досить просте і описане в пункті 1.9 даної роботи.

Двигун, реле, та інші розглянуті в роботі пристрої це не все, чим ми можна управляти через транзистор. Будь-який вид навантаження постійного струму може управлятися таким чином. Світлодіоди, лампочки або інші споживачі, навіть інший мікроконтролер може бути підключений подібним чином.

РОЗДІЛ 2. РОЗРОБКА СХЕМИ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ АВТОНОМНОЇ СИСТЕМИ КОНТРОЛЮ ОСВІТЛЕНОСТІ

2.1. Розробка принципової схеми системи контролю освітленості

У рамках даної роботи було розроблено схему та програмне забезпечення, які дозволяють на базі платформи Digispark, заснованій на мікроконтролері ATTiny 85, побудувати дешеву систему, яка контролює систему освітлення у громадських місцях. Для здешевлення готового приладу і підвищення його ремонтопридатності було вирішено взяти складові частини системи у вигляді модулів розширення, сумісних з платформою Arduino.

До складу системи увійшли датчик освітлення – фоторезисторний модуль (Рисунок 2.1 а); п’єзоелектричний датчик руху (Рисунок 2.1 б); мікроконтролерний модуль Digispark (Рисунок 2.1 в) та релейний модуль із оптичною розв’язкою (Рисунок 2.1 г). Реалізація зміни порогового рівня спрацювання здійснювалась за допомогою двох апаратних кнопок у межах 10 кроків.

Із огляду на поставлені задачі було розроблено схему приладу (Рисунок 2.2). Оскільки усі модулі живляться від напруги 5 В, було вирішено підключити їх паралельно до двох шин живлення контролера – 5V та GND. До аналогового входу A0 (P5 у номенклатурі Digispark) був підключений фоторезистор, сигнал із якого буде зчитуватись із роздільною здатністю 10 біт, тобто прийматиме 1024 можливих рівні. З огляду на те, що п’єзоелектричний датчик руху має вихід із TTL-сумісним сигналом, порт контролера P4 було запрограмовано як цифровий вхід. Оскільки реле є керуючим модулем, і повинно приймати на свій вхід керуючий сигнал, його вхід було під’єднано до порту P3, який програмно було переведено у режим цифрового виходу.

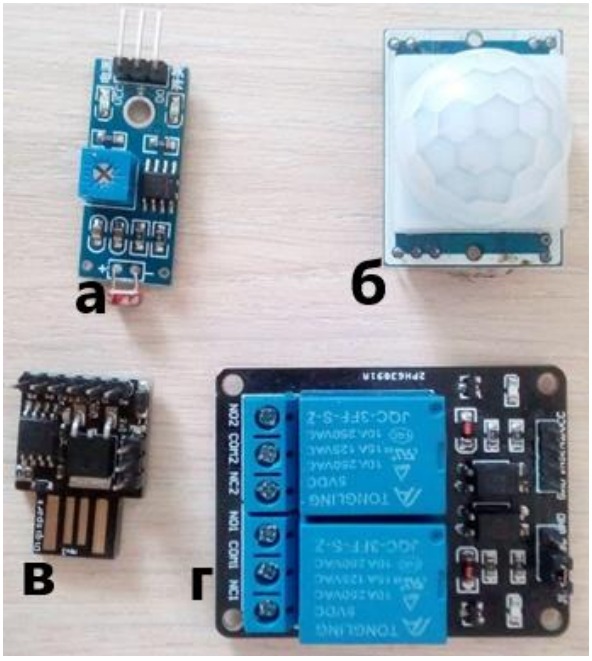


Рисунок 2.1 – Модулі розширення, використані при розробці автоматизованої системи: а - фоторезисторний модуль, б - п’єзоелектричний датчик руху, в - мікроконтролерний модуль Digispark, г - релейний модуль із оптичною розв’язкою.

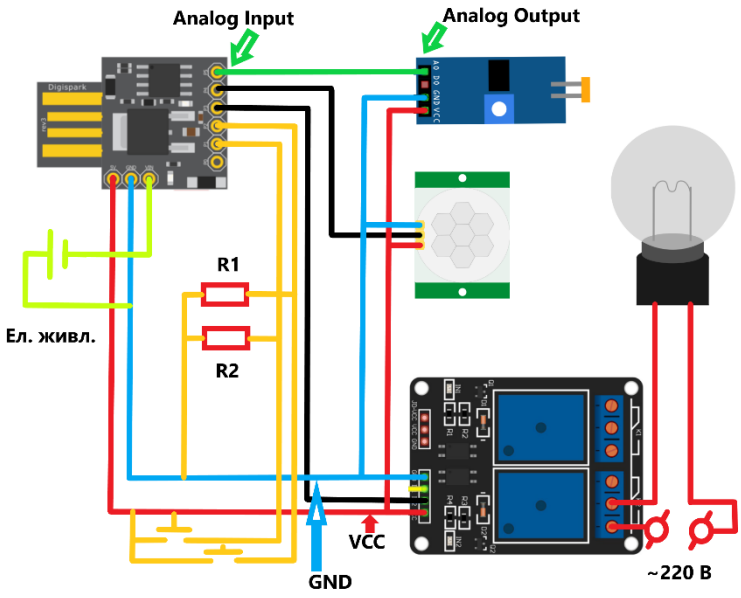


Рисунок 2.2 – Схема системи контролю освітленості

До портів контролера P1 і P2 було підключено апаратні кнопки, які здійснюють зміну порогового рівня спрацювання у межах 10 кроків. Для попередження хибних спрацювань кнопок, внаслідок можливих наводок у провідниках, у схемі використано два підтягуючих резистора R1 та R2 номіналом 10 кОм. Вони дозволяють встановити перманентний низький сигнал на портах P1 і P2 у момент, коли конопки не натиснуті, у разі натискання останніх, порт замикається на 5В, і встановлюється сигнал високого логічного рівня.

У результаті розробки апаратної та програмної складової системи було розроблено прилад на базі модулів і компонентів, що випускаються серійно із мінімальними затратами на зборку. Розроблена система може досить довго працювати автономно (близько 1 місяця) від 3-х батарейок типу AA. Концепція може бути розвинута для реалізації інших проектів, а автономність підвищена за рахунок використання сонячних перетворювачів енергії.

2.2. Розробка програми для мікроконтролерного модуля Digispark

Перейдемо до розробки програмного забезпечення для керування роботою розробленої нами схеми приладу. Написання коду програми здійснювалось нами у середовищі програмування Arduino IDE, з додатковими програмними модулями для систем на базі мікроконтролерів Attiny. Такий підхід був обраний, зважаючи на те, що програмування здійснюється на високорівневій мові C++, синтаксис якої зрозумілий, і не потребує заглиблення у особливості роботи апаратної складової мікроконтролера.

Програма в середовищі програмування Arduino IDE структурно ділиться на декілька блоків:

- описання директив препроцесора;
- оголошення констант та глобальних змінних;
- блок ініціалізації обладнання (setup);
- цикл основної програми;

– окремо описані функції.

Опишемо вміст кожного із цих блоків, в контексті нашої програми, та приведемо їх вміст. В блоці описання директив препроцесора зазначаються директиви `#define` та `#include`, перша із котрих служить для описання текстових підмін у коді, у нашій програмі вона не використовується. Друга директива служить для підключення бібліотек до проекту, у нашому випадку це `#include <TinyPinChange.h>`, яка призначена для нумерації виводів контролера, так, як це прийнято у середовищі розробки Arduino IDE.

Наступним етапом у програмі є описання глобальних змінних та констант, у нашій програмі даний розділ має наступний вигляд:

```
bool buttonMinus = false;
bool buttonPlus = false;
float riven = 0;
bool ruh = false;
```

Тут ми бачимо оголошення трьох змінних логічного типу `bool` та однієї – `float` (число з плаваючою крапкою). Змінні `buttonMinus` та `buttonPlus` використовуються у програмі для тимчасового запам’ятовування стану фізичних кнопок регулювання рівня освітленості при якому відбувається спрацювання тригера увімкнення освітлення у процесі програмної компенсації дребежання контактів. Змінна `riven` зберігає програмно задане значення рівня освітленості. Більше глобальних змінних у програмі не описано, оскільки вони займають багато апаратних ресурсів, і використовувати їх потрібно лише за крайньої необхідності, у решті випадків потрібно застосовувати локальні змінні.

Блок Setup нашої програми представлений наступним кодом:

```
void setup() {
  pinMode (1, INPUT);
  pinMode (2, INPUT);
  pinMode (3, OUTPUT);
  pinMode (4, INPUT);
```

```
    float tmpRiven = EEPROM_read(0);
    if (tmpRiven >= 0 && tmpRiven <= 9)
      riven = tmpRiven;
    else riven = 0.0 ;
  }
```

Функцією `pinMode` ми можемо виконати попередню ініціалізацію дискретного порту, тобто встановити його як вхіж або вихід. У нашій програмі виводи № 1, 2 та 4 ініціалізовані входом, оскільки до них підключені фізичні кнопки та датчик руху. Далі іде блок зчитування числового значення програмно заданого порогового рівня освітленості із нульової комірки енергонезалежної пам’яті EEPROM, після чого перевіряється, чи знаходиться зчитане значення у заданих межах, тобто чи не відбулась помилка зчитування і встановлює значення за замовчуванням у разі потреби (така ситуація може скластись, наприклад, при першому включенні приладу, коли дані ще жодного разу не були записані в цю область пам’яті). Сама процедура зчитування описана у окремій функції, яка розміщена за межами основного циклу та блоку налаштування і має наступний вид:

```
  float EEPROM_read(int addr) {
    byte raw[4];
    for (byte i = 0; i < 4; i++) raw[i] = EEPROM.read(addr
+ i);
    float &num = (float&)raw;
    return num;
  }
```

Вона описано окремою структурою, вхідним параметром якої є адреса комірки в області пам’яті, а вихідним – зчитане числове значення. Всередині функції реалізоване по-байтне зчитування числа в циклі, та збір його в єдину цілу структурну одиницю даних типу `float`.

Наступним етапом виконання програми є головний цикл `Void`. Він має єдину точку входу, і не має жодної точки виходу, а також не повертає ніяких

значень, оскільки його виконання не переривається до фізичного вимкнення мікроконтролера (зняття живлення). У даному циклі описані наступні операції:

– зчитування значень із датчиків;
– зчитування натискань кнопок;
– прийняття рішень для виконання.

Перший пункт реалізовано наступною програмною послідовністю:

```
bool ruh = digitalRead(3);
float rivenDatchik = analogRead(A0) / 102.4;
```

де ми створюємо локальні змінні типу `bool` та `float`, і за допомогою функцій дискретного та аналогового введення присвоюємо їм зчитані значення фізичних параметрів із датчиків.

Зчитування натискання кнопок «+» та «–» реалізовано однаковим чином і має такий вигляд:

```
//кнопка мінус та запис в EEPROM
  if (digitalRead(2)) {
    if (drebezg(2) && !buttonMinus) {
      riven--;
      if (riven < 0) {
        riven = 0;
      }
      EEPROM_write(0, riven);
      buttonMinus = true;
    }
    else if (!digitalRead(2))buttonMinus = false;
  }

  //кнопка плюс та запис в EEPROM
  if (digitalRead(1)) {
    if (drebezg(1) && !buttonPlus) {
      riven++;
```

```
        if (riven > 9) {
          riven = 9;
        }
        EEPROM_write(0, riven);
        buttonPlus = true;
      }
      else if (!digitalRead(1))buttonPlus = false;
    }
```

У кожній із операцій зчитування передбачено запис до енергонезалежної пам’яті зміненого числового значення порогового рівня освітленості, котре буде зчитано при наступному запуску контролера. Дана операція також описана окремою функцією і має вид:

```
void EEPROM_write(int addr, float num) {
  byte raw[4];
  (float&)raw = num;
  for (byte i = 0; i < 4; i++) EEPROM.write(addr + i, raw[i]);
}
```

Логіка роботи даної послідовності операцій обернена до описаної вище процедури зчитування, тобто спочатку ми розбиваємо число на байти, після чого у циклі записуємо їх до комірок енергонезалежної пам’яті.

Також у коді зчитування натискання кнопки використовується ще одна написана нами зовнішня функція `drebezg`, яка необхідна для запобігання хибних реакцій при натисканні кнопки, її код показаний нижче:

```
bool drebezg (int pin) {
  bool state = digitalRead(pin);
  delay(20);
  if (digitalRead(pin) == state) {
    return true;
  }
```

```
    else {
      return false;
    }
  }
```

Справа у тому, що фізичні кнопки не є ідеальним ключем, і при замиканні їхніх контактів може зчитатися декілька імпульсів, які будуть розцінені програмою, як окремі натискання і система працюватиме некоректно. Такий ефект є негативним явищем, і називається дребежання контактів. Існує ряд способів його позбутися, найбільш надійними є апаратні способи, але вони ускладнюють схему приладу, внаслідок додавання до неї додаткових деталей, менш надійним, але досить дієвим є програмний метод компенсації, який нами і був застосований. При такому підході відбувається наступна логічна послідовність подій: фіксується натискання кнопки, після чого програма чекає деякий час, і знову зчитує стан кнопки, якщо вона має такий же стан, як і при першому зчитуванні, то вважається що кнопка натиснута, і виконується обробка дій, запрограмованих на натискання. Час, який програма очікує між двома послідовними зчитуваннями стану кнопки залежить від конкретного фізичного елементу, і для кожної кнопки підбирається експериментально, так, щоб забезпечилось повне замикання контактів. Для кнопок, що використовувались нами було підібрано час 20 мілісекунд, який був взятий із запасом, оскільки система працювала коректно із затримкою 15 мс.

В результаті розробки програмної та апаратної частини системи, нами було отримано прилад, який може автоматично вмикати освітлення в місцях загального користування при таких параметрах: фіксується рух, і освітленість, виміряна датчиком, менше певного порогового рівня, який ми можемо змінювати кнопками у межах 10 кроків.

ВИСНОВКИ

1. Розглянуто та проаналізовано історію створення, розвитку мікроконтролерів ATtiny, їх архітектури та системи команд. Здійснено огляд технічних характеристик контролера ATtiny85 та платформ на його основі.

2. Так як ATtiny85 є універсальним контролером, який можна запрограмувати для конкретної задачі та перетворити в робочий пристрій, для його програмування використали середовище програмування Arduino IDE. Перед початком програмування в даному середовищі програмування, було встановлено необхідні для його роботи драйвери та плагіни, спираючись на літературу, було враховано усі нюанси написання скетчів.

3. Розглянуто приклади практичного застосування ATtiny85, також проаналізували варіанти керування навантаженням за допомогою симистора, мікроконтролера, реле та Mosfet транзисторів.

4. Розроблено принципову схему системи контролю системою освітлення у громадських місцях, за якою буде сконструйовано прилад на базі модулів і компонентів, що випускаються серійно із мінімальними затратами на зборку та з практичною автономністю.

5. Розроблено програмну та апаратну частини системи, нами було отримано прилад, який може автоматично вмикати освітлення в місцях загального користування при таких параметрах: фіксується рух, і освітленість, виміряна датчиком, менше певного порогового рівня, який ми можемо змінювати кнопками у межах 10 кроків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. Рябенький В.М., Ушкаренко О.О., Буряк В.С. Схемотехніка електронних пристроїв та систем: мікропроцесорна техніка. Том 3. - Миколаїв: Іліон, 2012. - 446 с.

2. Бєлов А.В. Мікроконтролери AVR. Від азів програмування до створення практичних пристроїв. - Санкт-Петербург: Наука і Техніка, 2016. – 544 ст.

3. B. Afzal, M. Umair, E. Ahmed. Enabling IoT platforms for social IoT applications: Vision, feature mapping, and challenges, Future Generation Computer Systems, 2017. – 685 p.

4. L. Pintilie, T. Pop, I. Gros, A Iuoras Department of Electrical Machines and Drives. - Cluj-Napoca, Romania: Technical University of Cluj-Napoca, 2019. – 124 p.

5. Q. Liu, X. Yang, L. Deng. An IBeacon-Based Location System for Smart Home Control // Sensors. – 2018. – V. 18. – P. 1897–1910.

6. Микроконтролери AVR. Електронний ресурс. Режим доступу: http://www.studmed.info/docs/document3940/content. Дата доступу: 14.04.2020.

7. Z. A. Jabbar, R.S. Kawitkar. Implementation of Smart Home Control by Using Low Cost Arduino & Android Design // International Journal of Advanced Research in Computer and Communication Engineering. – 2016. – V. 5, № 2. – P. 248–256.

8. Atmel AVR 8-bit and 32-bit Microcontrollers. Електронний ресурс. Режим доступу: http://www.atmel.com/products/microcontrollers/avr/ Дата доступу: 12.04.2020

9. Бессекерский В.А., Попов Е.П. Теория систем автоматического управления / В.А. Бесссекерский, Е.П. Попов – Издание 4-е, переработанное и дополненное Санкт-Петербург, «Профессия», 2007. - 752с.

10. Широтно-Імпульсна Модуляція (ШІМ, PWM). Електронний ресурс. Режим доступу: http://articles.greenchip.com.ua/1-1-20-1.html Дата доступу 15.04.2020

11. Саймон Монк Программируем Arduino. Профессиональная работа со скетчами. - Санкт-Петербург: Питер, 2017. – 98с.

12. Макаров С. Л. Arduino Uno и Raspberry Pi 3. От схемотехники к интернету вещей. - Москва: ДМК Пресс, 2019. - 202с.

13. Джереми Блум Изучаем Arduino: инструменты и методы технического волшебства. - Санкт-Петербург: БХВ-Петербург, 2018. – 336с.