МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ КАФЕДРА ПРОГРАМУВАННЯ ТА МАТЕМАТИКИ

До захисту допускається

В.о. завідувач кафедри

_____Лифар В.О.

«____»____2021 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломної роботи

бакалавр

(освітньо-кваліфікаційний рівень)

НА ТЕМУ:

Комп’ютерна система допомоги водію

Керівник роботи:	Захожай О.І.
(підпис)	(ініціали, прізвище)
Студент:	Устовицький Д.А.
(підпис)	(ініціали, прізвище)
Група:	КІ-17

Сєвєродонецьк 2021

ЛИСТ ПОГОДЖЕННЯ І ОЦІНЮВАННЯ

дипломної роботи студента гр. КІ-17 Устовицький Д.А.

Науковий керівник

Доцент, к.т.н. _____ Захожай О.І.

Оцінка наукового керівника: _____

Рецензент: _____

ПІБ, місто роботи, посада

Оцінка рецензента: _____

Кінцева оцінка за результатами захисту:

_____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра Програмування та математики
Освітньо-кваліфікаційний рівень бакалавр
Напрям підготовки 6.050102 «Комп'ютерна інженерія»

(шифр і назва)

Спеціальність

(шифр і назва)

ЗАТВЕРДЖУЮ:

В.о. завідувач кафедри

В.О. Лифар

«_____» _____ 20____ р.

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ
Устовицького Дениса Андрійовича

(прізвище, ім'я, по батькові)

1. Тема роботи: Система допомоги автомобілісту
керівник проекту (роботи) Лифар В.О. кандидат технічних наук, доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від " " 2021 р. №
2. Строк подання студентом роботи 10 червня 2021

3. Вихідні дані до роботи : Коректне відображення показань датчиків автомобіля

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): Аналітичний огляд, розробка апаратного забезпечення, розробкапрограмного забезпечення. Висновки

---

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завданнявидав	Завданняприйняв

Дата видачі завдання 22 березня 2021 року

Керівник

_____

(підпис)

Завдання прийняв до виконання _____

(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Складання плану роботи	22.03.20 - 24.03.20	
2	Аналіз літератури	24.03.20 – 29.03.20	
3	Вивчення і підбирання матеріалу	29.03.20 – 20.04.20	
4	Написання розділів	20.04.20 – 25.05.20	

5	Оформлення пояснювальної записки	25.05.20 – 28.05.20	
6	Оформлення графічного Матеріалу	28.05.20 – 03.06.20	
7	Підготовка доповіді і слайдів для Презентації	03.06.20 – 10.06.20	

Студент _____ _____
(Підпис) (прізвище та ініціали)

Науковий керівник _____ _____
(Підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка включає 40 сторінок, 20рисунків, 2 таблиць, 10джерел за переліком посилань.

Ключові слова:

Мета роботи: вивід більш зрозумілих показань стандартних датчиків автомобіля.

Результат проектування : була спроектована автоматична система виводу показань : рівня палива, температури охолоджувальної рідини, температури салону. Також показання відкриття дверей.

Додаток Proteus 8 використовується для візуалізації та перевірки програмного забезпечення. Atmel studio 6.2 використовується для написання коду та прошивки мікро контролера.

Ця ідея відтворена фізично і буде показана в дії.

ЗМІСТ

ВИЗНАЧЕННЯ, ПОЗНАЧЕННЯ І СКОРОЧЕННЯ

У цій пояснювальній записці застосовують такі терміни з відповідними визначеннями:

МК- Atmega8 мікроконтролер

АЦП- аналого цифровий перетворювач який встоин в МК

ДТ- ds18b20 датчик температури салону

ДРП- датчик рівня палива

ДТОР- датчик температури охолоджувальної рідини

ДД- датчики хола на відкриття дверей

ЛСД- LCD1602 для виведення показань

.

ВСТУП

Стандартні показання датчиків автомобіля реалізовані за допомогою схематичних позначок та стрілки яка указує на приблизні показання.

У 60-х роках автомобілі були обладнані датчиками тиску масла, рівня палива, температури охолоджуючої рідини. Їх виходи були підключені до стрілочним або ламповим індикаторами на щитку приладів.

У 70-х роках автомобільні компанії почали боротися за зменшення кількості токсичних викидів з глушника автомобіля — потрібні були додаткові датчики для управління силовою установкою, які необхідні для забезпечення нормальної роботи електронного запалювання, системи вприскування палива, трикомпонентного нейтралізатора, для точного завдання співвідношення повітря/паливо в робочій суміші, для зменшення токсичності вихлопних газів.

У 80-х роках почали приділяти більше уваги безпеці водія і пасажирів — з’явилися антиблокувальна система гальмування (ABS) і повітряні мішки безпеки.

Із за неточних даних які отримує водій може статися значна поломка . Або незачинені двері можуть стати причиною ДТП.

В рамках цієї роботи прийнято рішення зробити автоматичний вивід даних на невеликий дисплей. Завдяки сучасним технологіям перетворення аналогових сигналів з датчиків в цифровий та їх відображення на дисплей буде всього один мікро контролер.

Дана система позволить виводити фактично будь які дані з датчиків автомобіля і не тільки, які потрібні саме вам.

1.АНАЛІТИЧНИЙ ОГЛЯД

1.1 Формулювання актуальності, мети і завдань проекту

У сучасних автомобілів є маса можливостей для налаштування всіх систем для коректної роботи але у більшості автомобілів стоять показники датчиків з минулого віку. Аналогові показники дивлячись на які можливо тільки приблизно розуміти їх показання.

Але в наш час вивід зрозумілих та коректних показань не є проблемою. Так як на всіх сучасних автомобілях є мозок на якому зчитуються та перероблюються дані. Незважаючи на це все одно на більшості автомобілів немає таких заводських рішень. Або мають тільки іномарки з розширеною комплектацією. Завдяки розвитку сучасних технологій в сфері мікро електроніки є можливість створити автономну універсальну систему для виведення показань зі штатних датчиків. Замість коливань стрілки зрозумілі для сприйняття цифри з позначками що це за показання. Така система допоможе більш ретельно спостерігати за своїм автомобілем та уникнути несподіваних ситуацій , наприклад закінчиться паливо чи відкриються погано замкнуті двері на повороті, стукне двигун із за перевищення температурного режиму і так далі. Таку систему можливо буде доповнити різними додатковими датчиками які знадобляться в процесі експлуатації такої системи.

Відштовхуючись від вищесказаного, розробка функціональної адаптивної та автоматичної системи виведення даних з датчиків автомобіля допоможе більшості сумлінних автомобілістів більш ретельно та точно слідкувати за своїм автомобілем .

Завданням цієї роботи є: виготовлення працюючу макетну універсальну автоматичну систему виводу даних з шатних датчиків автомобіля на мікроконтролері.

Для досягнення цієї мети поставлені і виконані подальші завдання:

1. Огляд наявних рішень
2. Розробка схеми пристрою
3. Вибір потрібних компонент

4. Завдання методу роботи
5. Створення програми роботи пристрою
6. Виготовлення макету пристрою
7. Налагодження програми та макета

1.2 Аналіз початкових даних і відомих рішень

Сьогодні існує багато рішень цієї проблеми як заводські бортові комп’ютери які підключаються до мозку автомобіля. Такі прилади мають багато функцій та налаштувань. Також є індивідуальний вивід з датчика який заміняється в шатне місце.

Бортовий комп'ютер (маршрутний комп'ютер, бортовий комп'ютер) - невелике автомобільне пристрій, який зчитує, обробляє і виводить на дисплей корисну інформацію, серед якої: витрата палива (середній, миттєвий), його залишок в баку (з точністю до літра); статистика (середня швидкість, пройдену відстань, витрачене паливо); температура повітря в салоні і за бортом; неполадки автомобіля і багато інших параметрів. Купити бортовий комп'ютер хоча б заради перерахованих вище параметрів варто як в стареньку інжекторних машину двадцятирічної давності, так і в новий недавно придбаний автомобіль.

Бортовий комп'ютер також створений для виведення і подальшого аналізу даних про всі агрегатах автомобіля. Автомобільні комп'ютери попереджають і миттєво повідомляють про знайдені неполадки або проблеми. При експлуатації будь-якого сучасного автомобіля надійна і безвідмовна робота неможлива без точного системного контролю всіх процесів. Бортовий комп'ютер призначається саме для таких цілей.

Є два основних типи маршрутних комп'ютерів - модельні бортові комп'ютери і універсальні мультисистемні бортові комп'ютери. Модельні бортові комп'ютери працюють з протоколами певних моделей автомобіля і мають конструктивні обмеження. Універсальні мультисистемні бортові комп'ютери підтримують більш широкий список автомобілів, від простих інжекторних авто, до більш складних дизельних двигунів з турбонаддувом.

Відмінність модельного і універсального бортових комп'ютерів також полягає і в дизайні. Зазвичай, модельний буде виглядати краще на штатному місці конкретного автомобіля, але при заміні автомобіля доведеться попрощатися з вашим «помічником» і доглядати нового «друга». З універсальним маршрутним комп'ютером ситуація простіше, - після оновлення програмного забезпечення його можна буде легко встановити в ваше нове авто. У будь-якому випадку, спочатку слід з'ясувати яка саме модель бортового комп'ютера підходить для вашого автомобіля.

Рисунок1.1:бортові компьютери

Маршрутний комп'ютер сповіщає водія про основні параметри поїздки -

швидкість, частота обороту двигуна і його температура, решту обсягу палива, температуру повітря в салоні і відкритого повітря, напруга електросистеми і багато чого іншого. З іншого боку, частково ці дані можна отримати по штатних датчиків автомобіля, але покупка бортового комп'ютера не втрачає своєї актуальності, а навпаки. Хоча б тому, що дані маршрутного комп'ютера відображаються точніше. А ще він може коригувати вихідні дані для обчислень після заміни різних комплектуючих і систем автомобіля. Купити маршрутний комп'ютер в Києві, Харкові, Одесі можна на 130.com.ua з доставкою по Україні. Якісні і дорогі моделі бортових комп'ютерів, крім тривіальних показань датчика, допоможуть з плануванням вашого маршруту, підберуть оптимальну швидкість на певній ділянці шляху, підрахують кілометри згідно залишку палива в баку. Бортовий комп'ютер проводить перевірку всіх вузлів і систем автомобіля і при виявленні «нестандартної» ситуації блискавично сповістить водія про виявлену проблему або поломки. Маршрутний (бортовий) комп'ютер «зчитує» всі наявні датчики автомобіля. Після тестування початкових параметрів, він робить аналіз продуктивності автомобіля і надає детальний звіт водієві. У комп'ютерах дорого класу зазвичай використовується рідкокристалічний дисплей, а в більш простих комп'ютерах - цифровий (3-х або 4-х-розрядний).

Індивідуальні електроні датчики виводу даних розроблені таким способом що для підключення в шатні датчики

Рисунок 1.2:індивідуальний електронний датчик

Вони пристосованфі для виводу тільки одного датчика .

2.ПРОЕКТНИЙ РОЗДІЛ

2.1 Розробка електричної схеми і вибір необхідних компонентів

В даному проекті був вибраний мікроконтролер Atmega8 на якому буде построенна вся система.

Мікроконтролер ATmega8 виконаний за технологією CMOS, 8-розрядний, мікропотребляющій, заснований на AVR-архітектурі RISC. Виконуючи одну повноцінну інструкцію за один такт, ATmega8 досягає продуктивності 1 MIPS на МГц, дозволяючи досягти оптимального співвідношення продуктивності до споживаної енергії

Мікроконтролер та його распіновка вказана на рисунку 2.1

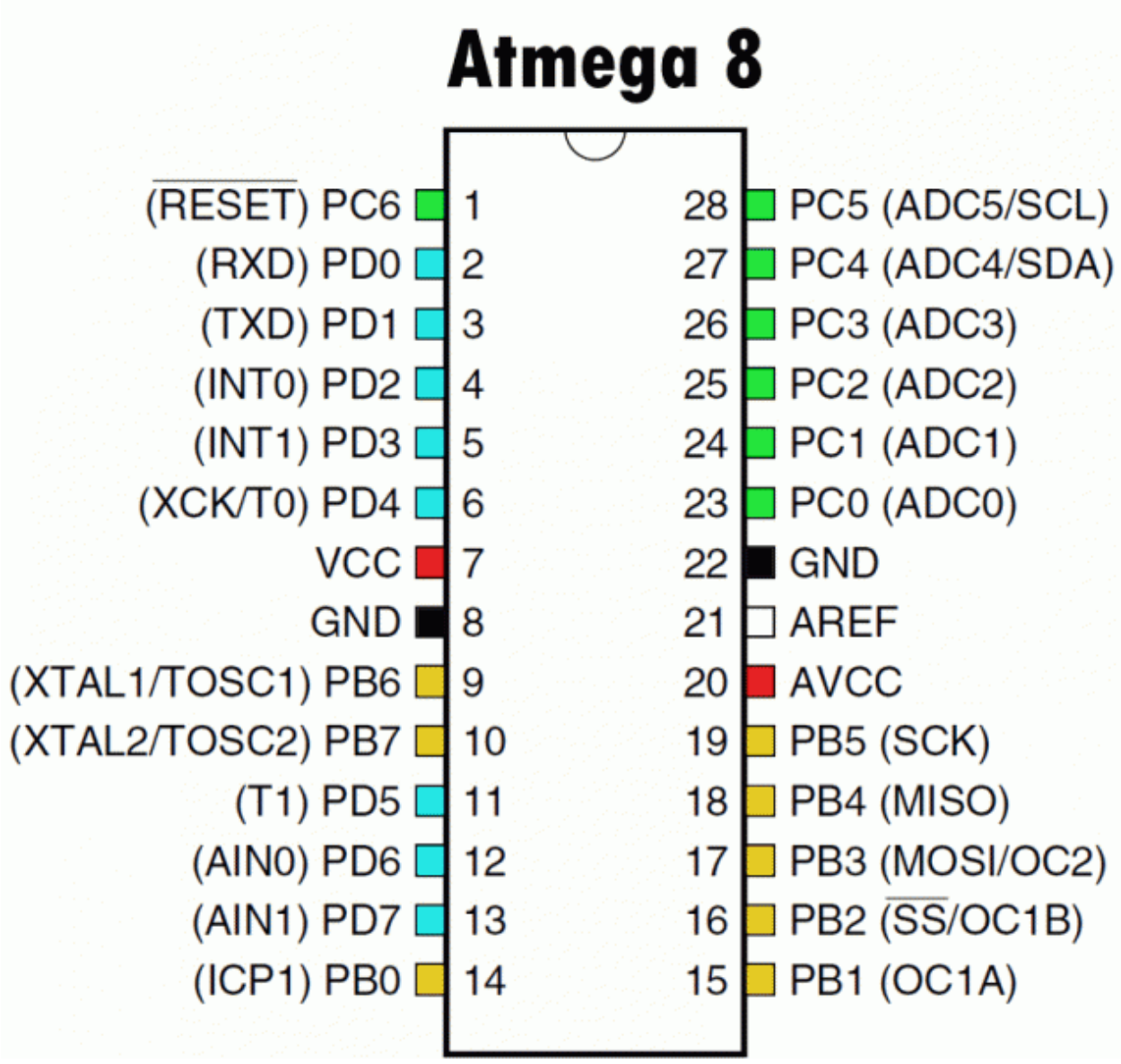


Рисунок 2.1: Atmega8 його распіновка (корпус PDIP)

Технічні характеристики :

Програмна пам’ять - 8кБ

Флеш пам’ять - 512 байт

Оперативна пам’ять (ОЗУ) - 1кБ

Регистри спільного призначення - 32

Два 8-ми розрядних таймера

Таймер реального часу

Три канали ШИМ

Шість АЦП , 10-ти розрядні

I2C(TWI)

USART

SPI

Сторожовий таймер

Аналоговий компаратор

Можливість обробки внутрішніх та зовнішніх переривань

Напруга від 4,5В до 5,5 В

Тактова частота від 0 до 16 МГц

Потужність- 3.6mA

Робоча температура выд -40 до 85С

Роспіновка ATmega8

Таблиця 1: призначення портів D

№	**Назва**	**Опис**
1	PD0, RxD	вхід USART
2	PD1, TxD	вихід USART
3	PD2 ,INT0	зовнішнє прикривання 0 канала
4	PD3, INT1	зовнішнє прикривання 1 канал
5	PD4,XKC, T0	зовнішній такт для USART, зовнішній вхід таймера0
6	PD5 - /*/ , T1	зовнішній вхід таймера1

7	PD6, AIN0	вхід аналогового компоратора канал0
8	PD7, AIN1-	вхід аналогового компоратора канал1

Порт С:

№	Назва	Опис
1	PC0, ADC0	аналоговий вхід канал0
2	PC1, ADC1	аналоговий вхід канал0
3	PC2, ADC2	аналоговий вхід канал0
4	PC3, ADC3	аналоговий вхід канал0
5	PC4, ADC4	аналоговий вхід канал0
6	PC5, ADC5	SDA дву проводний послідовний інтерфейс (канал даних)
7	PC6, ADC6	RESET- зовнішній сброс

Порт В:

№	Назва	Опис
1	PB0, ICP1	захват входу1
2	PB1, OC1A	вихід зрівняння/ШИМ 1А
3	PB2, OC1B	вихід зрівняння/ШИМ 1В
4	PB3, OC2	вхід зрівняння/ШИМ2, MOSI
5	PB4	MISO
6	PB5, SCK-	тактовий вхід
7	PB6, XTAL1	тактовий вхід резонатора
8	PB7, XTAL2	тактовий вхід в випадку роботи встроєного резонатору

Вивод для питання мікроконтролеру

№	Назва	Опис
1	Vcc	вхід напруги
2	Gnd	заземлення
3	AVcc	вхід напругу для модуля АЦП
4	ARef	вхід опорної напруги для АЦП

Датчик температури DS18B20

DS18B20 це цифровий вімірювач температури, з дозволила превращение 9 - 12 розрядів и функцією тривожної сигналу контролю за температурою. Параметри контролю могут буті задані користувачем и збережені в енергонезалежній пам'яті датчика. DS18B20 обмінюється данімі з мікро контролером за однопровідною Лінії зв'язку, вікорістовуючі протокол інтерфейсу 1-Wire. Харчування датчик может отрімуваті безпосередно від Лінії Даних, без использование зовнішнього джерела. В цьом режімі харчування датчика походити від ЕНЕРГІЇ, запасеної на паразітної ємності. Діапазон вимірювання температура стає від -55 до +125 ° С. Для діапазону від -10 до +85 ° С похібка НЕ перевіщує 0,5 ° С. У кожної мікросхеми DS18B20 є Унікальний серійній код довжина 64 розряду, Який дозволяє декількома датчикам підключатіся на одну Загальну лінію зв'язку. Тобто через один порт мікроконтролера можна обмінюватіся данімі з декількома датчиками, розподіленімі на значній відстані. Режим Вкрай Зручний для использование в системах екологічного контролю, моніторингу температури в будівлях, Вузли устаткування.

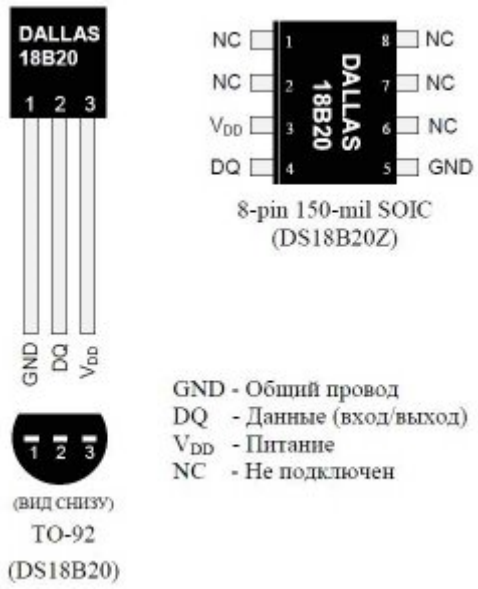


Рисунок 2.2: датчик температури DS18B20 та роспіновка

Для однопровідного інтерфейсу 1-Wire досить одного порту зв'язку з контролером.

Кожен пристрій має унікальний серійний код довжиною 64 розряду.

Можливість підключення декількох датчиків через одну лінію зв'язку.

Немає необхідності в зовнішніх компонентах.

Можливість отримувати харчування безпосередньо від лінії зв'язку. Напруга харчування в межах 3,0 В ... 5,5 В.

Діапазон вимірювання температури -55 ... +125 ° С.

Похибка не перевищує 0,5 ° С в діапазоні -10 ... +85 ° С.

Дозвіл перетворення 9 ... 12 біт. Здається користувачем.

Час вимірювання, не перевищує 750 мс, при максимально можливому дозволі 12 біт.

Можливість програмування параметрів тривожного сигналу.

Тривожний сигнал передає дані про адресу датчика, у якого температури вийшла за задані межі.

Вкрай широкі області застосування.

Датчик хола A3144

Датчик Холла - датчик, що працює на ефекті Холла (виникнення поперечної різниці потенціалів при приміщенні провідника з постійним струмом в магнітне поле). Датчик Холла A3144 - цифровий датчик (на виході тільки 2 стану - LOW і HIGH), що ідентифікує наявність магнітного поля, в безпосередній близькості від датчика.

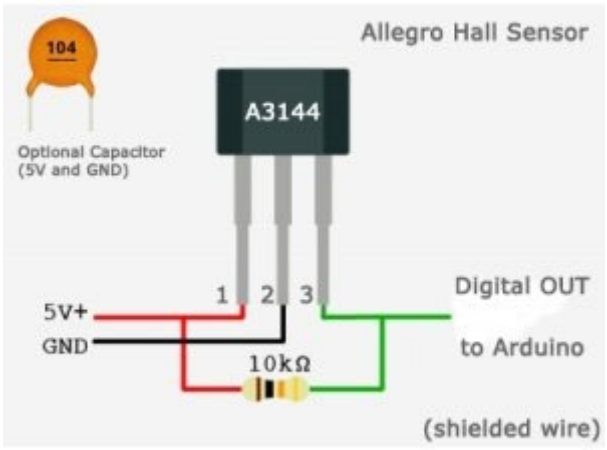


Рисунок 2.3: Датчик хола A3144, розпіновка та підключення

LCD 1602

Рідкокристалічний дисплей (Liquid Crystal Display) скорочено LCD побудований на технології рідких кристалів. При проектуванні електронні пристрої, нам потрібно недорогий пристрій для відображення інформації і другий не менш важливий фактор наявності готових бібліотек для Arduino. З усіх доступних LCD дисплеїв на ринку, найбільш часто використовуваною є LCD 1602A, який може відображати ASCII символу в 2 рядки (16 знаків в 1 рядку) кожен символ в вигляді матриці 5x7 пікселів. Зображене на рисунку 2,4

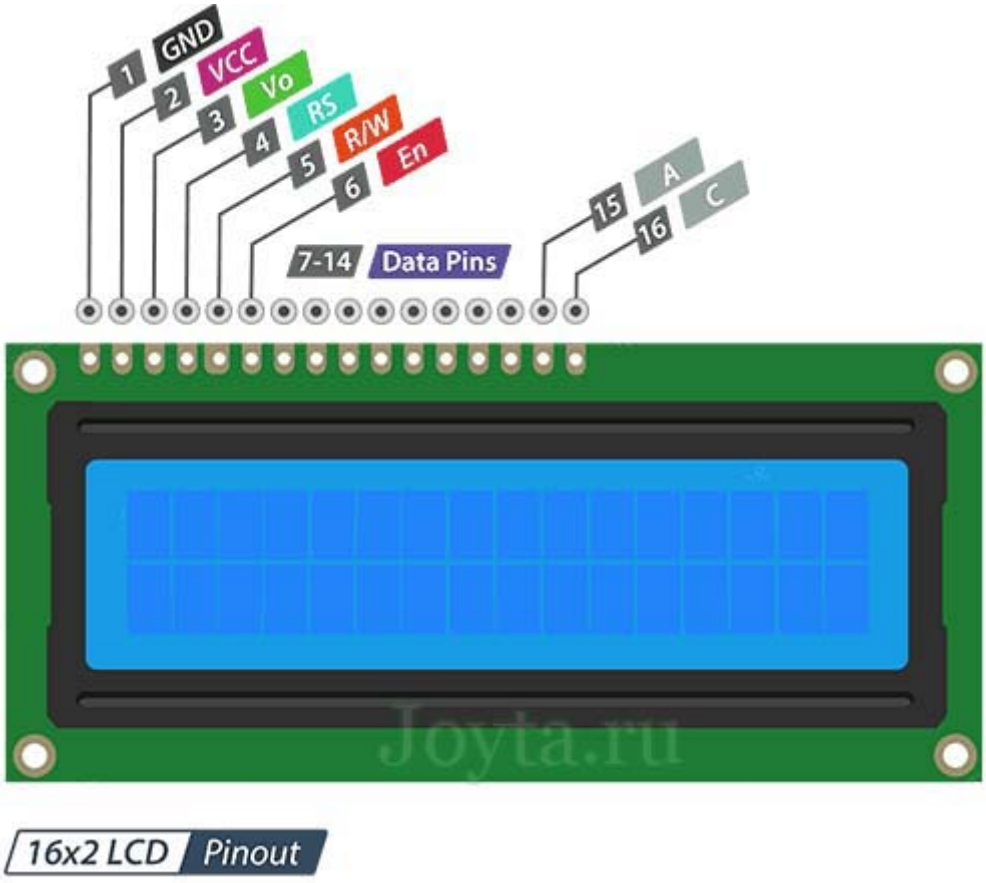


Рисунок 2,4: LCD 1602 та розпіновка

Обозначення пінів

► VSS: «-» харчування модуля

► VDD: «+» харчування модуля

► VO: Висновок управління контрастом

► RS: Вибір регістру

► R/W: Вибір режиму запису або читання (при підключенні до землі, встановлюється режим запису)

► En: Строб по спаду

► DB0-DB3: Біти інтерфейсу

► DB4-DB7: Біти інтерфейсу

► A: «+» харчування підсвічування

► C: «-» харчування підсвічування

Технічні характеристики

► Напруга живлення: 5 В

► Розмір дисплея: 2.6 дюйма

► Тип дисплея: 2 рядки по 16 символів

► Колір підсвічування: синій

► Колір символів: білий

►Габарітние: 80мм х 35мм х 11мм

Знижувач напруги AMS1117

Використовуємо плату на основі мікросхеми AMS1117 так як це буде біль надійним цей вузол. Зображений на рисунку 2,5

Мікросхема харчування AMS1117-5.0 SOT-223 відноситься до типу лінійних стабілізаторів напруги (ЛСН), мікросхема призначена для автоматичного, постійної підтримки стабільного напруги. Залежно від типів стабілізаторів, їх можна використовувати для регулювання одного або декількох напруг, змінного або постійного струму. Стабілізатори напруги набули широкого застосування в повсякденному житті споживача. Однією з таких сфер є використання в блоках живлення комп'ютерів, де вони стабілізують напругу постійного струму, що використовується процесором і іншими елементами. Мікросхеми стабілізаторів напруги діляться на два класи: лінійні стабілізатори і імпульсні стабілізатори. Лінійні стабілізатори - пристрої, які працюють в своїй лінійної області. На вхід лінійного стабілізатора подається вхідний, нестабільна напруга, а при виході генерується стабільний. Лінійні стабілізатори прості і не вимагають великої кількості додаткових електронних компонентів.2

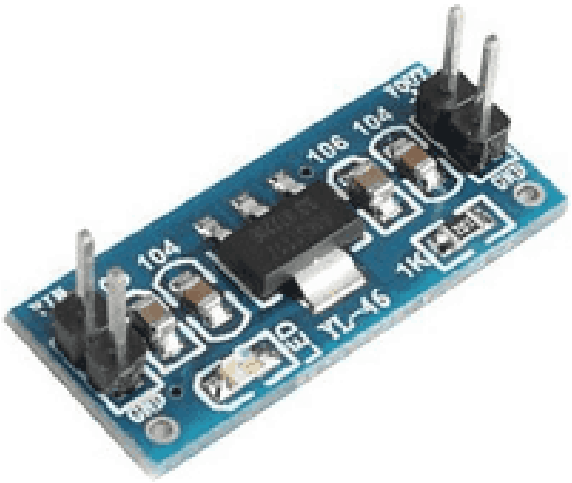


Рисунок 2,5: плата перетворення напруги на мікросхемі ASM1117

Технічні характеристики

Корпус SOT-223

серія AMS1117

Напруга на виході 4,8-5,1 В

Вхідна напруга 1,5-15 У

Струм власного споживання 5 мА

Максимальний вихідний струм 800 мА

Придушення нестабільності джерела живлення (PSRR) 68 дБ

Допустиме падіння напруги вхід-вихід 1,1 В

Робоча температура від -40 до 125 ° С

Розміри (ВхДхШ) 1,8 х 6,5 х 3,5 мм

Вага 280 мг

Активний Зумер

Звуковипромінювач електромеханічний є реле з парою нормально замкнутих контактів (у вихідному положенні контакти реле замкнуті). Котушка реле і джерело струму включаються в електричне коло послідовно.

Коливання якоря реле викликають коливання повітря - виходить звук, що нагадує дзижчання.

В середньому, звуковипромінювач релейного типу має напрацювання на відмову не більше 5000 годин. Зображенно на рисунку 2,6

Рисунок 2,6;зображення зумеру

Диаметр: 12 мм.

Длина: 9,5.

Входное напряжение: 3.5-5.5V.

Ток: <25 мА.

Частота: 2300 +/- 500 Гц.

Розробка конструкції пристрою

Для розробки проекту була вибрана програма Proteus8. У якій можливо реалізувати роботу мікропроцесора та датчиків автомобіля та додаткових датчиків .

Розглянемо підключення мікро контролера та всіх датчиків.

Підключення lcd 1602 к ATmega 8 Рисунок 2.10

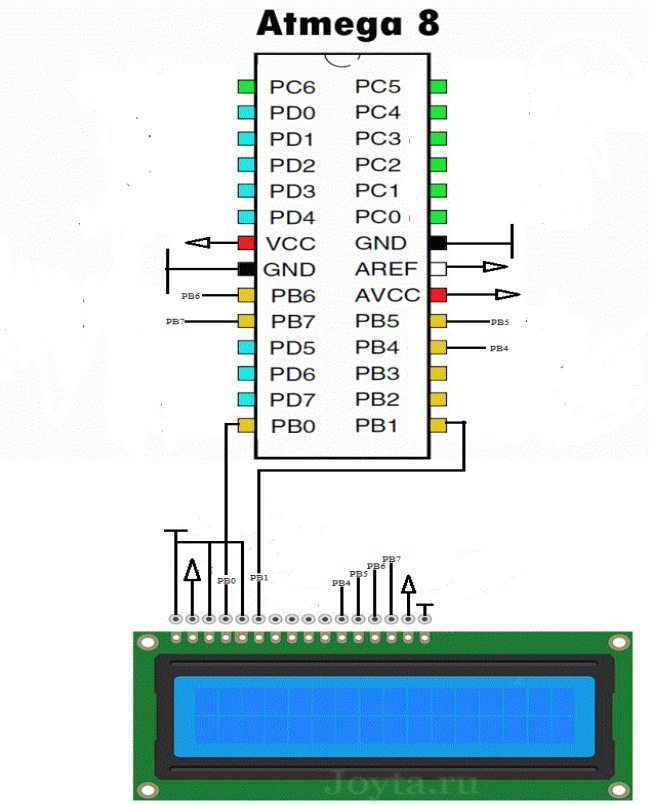


Рисунок 2.10:підключення LCD1602 до Atmega8 по 4 бітній шині

Такий тип підключення був обраний із за збереження портів , також відмова від підстроювального резистора для максимальної відображення.

Розглянемо підключення датчикі температури та рівня палива. Рисунок 2,11

Для підключення штатних датчиків ми повині знизити напругу з 12 В до 5В для цього ми підводимо до збуджуючої напруги датчиків 5В замість 12В . та на виході з датчиків підключаймо посилювач напруги LM2909

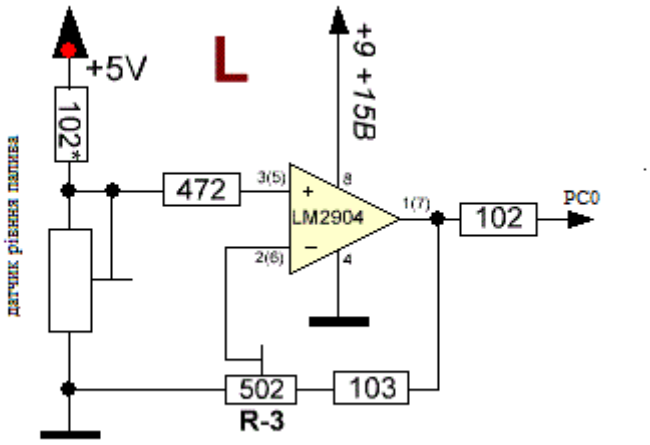


Рисунок 2,11: Підключення штатних аналогових датчиків.

Аналогічним способом підключаємо штатний датчик температури. Порти для підключення ДРП (PC0), ДТОР (PC1).

Підключення датчику температури DS18B20 проводиться по шині 1Wey. Та провід даних підключається к живленню за допомогою підтягуючого резистора на 4.7КОм. Рисунок 2,12

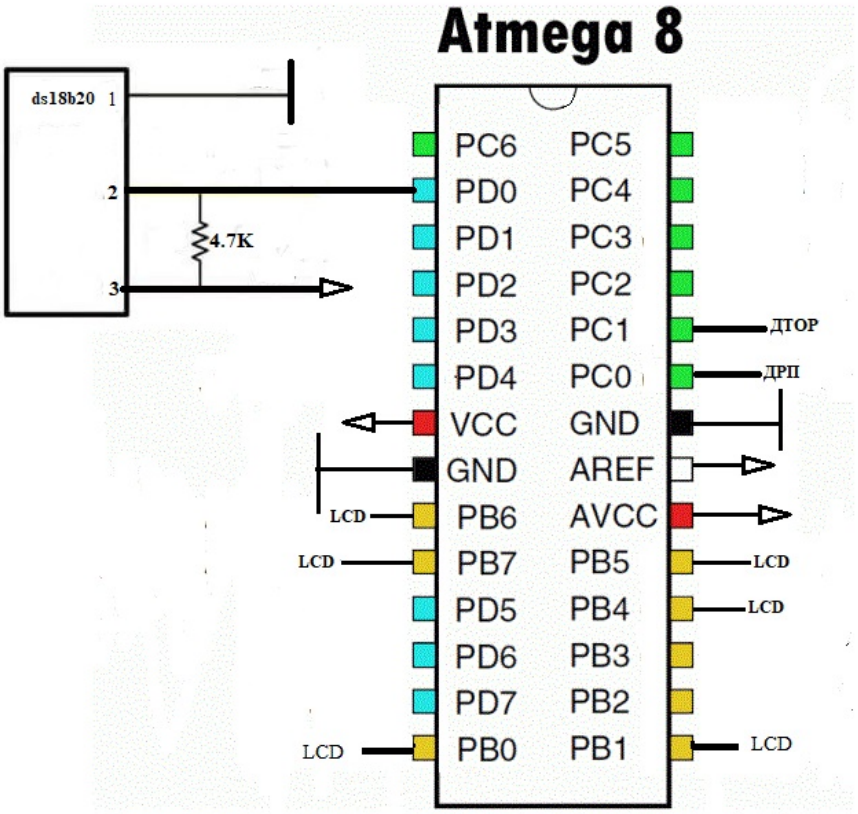


Рисунок 2,12: Підключення датчику температури ds18b20

Підключення датчику холу порозводиться також до контакту виводу за допомогою підтягуючого резистору на 10Ком. Рисунок2,13.

Датчики Хола виводять тільки 1\0 для кожної двері встановлюється свій датчик. Та підключаються до портів з PD4 до PD7.

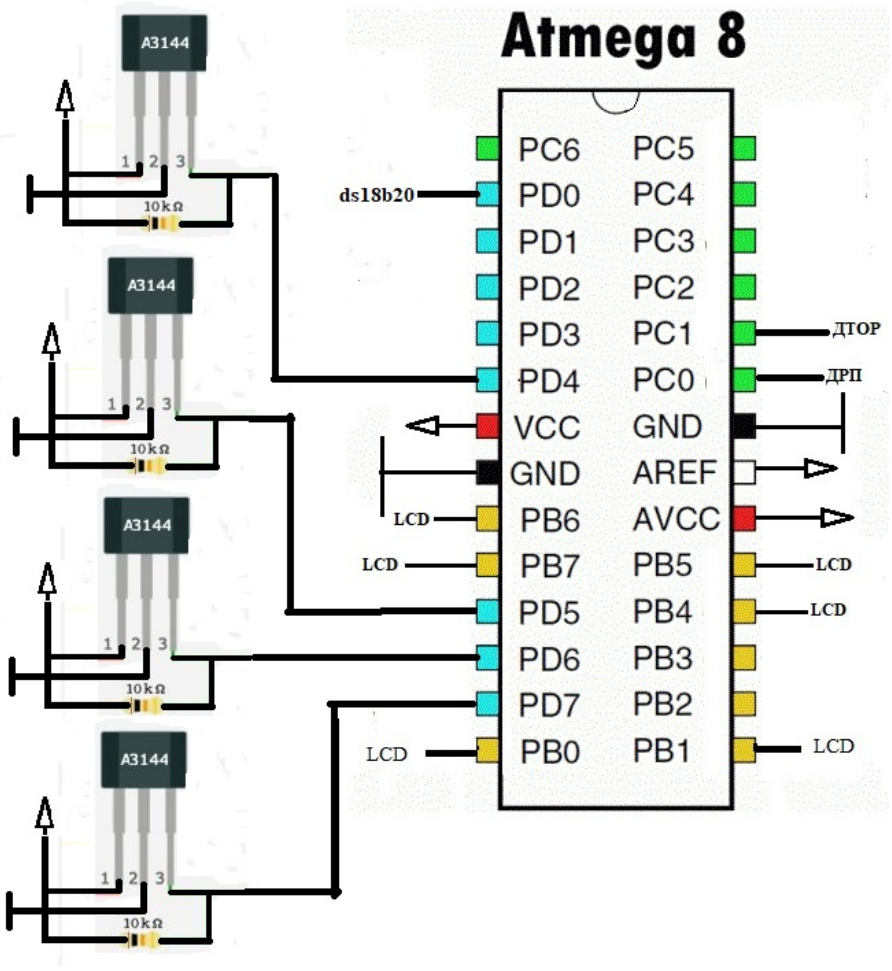


Рисунок 2,13: Підключення датчиків хола

Підключення зумеру проводиться на пряму через контакт напруги. Він буде сигналізувати про відчинення дверей та про низький рівень палива. Підключений к порту PD1 Рисунок 2,14

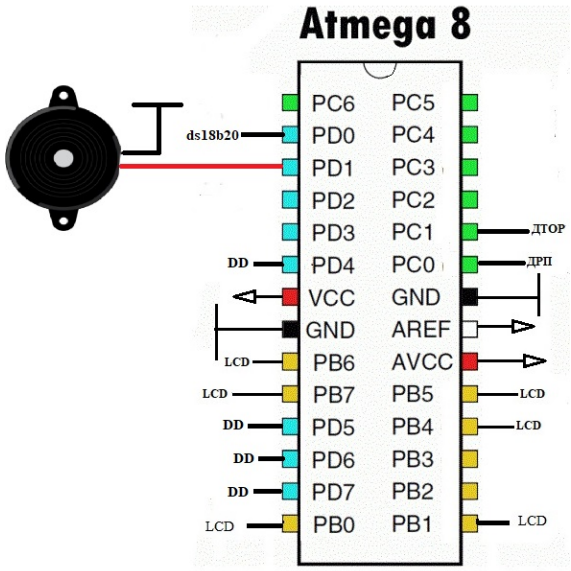


Рисунок 2,14: Підключення спікеру

Підключення плати пониження напруги на базі мікросхемі ASM1117. Проводиться пряме підключення від живлення акамулятора до мікросхеми рисунок 2,15.

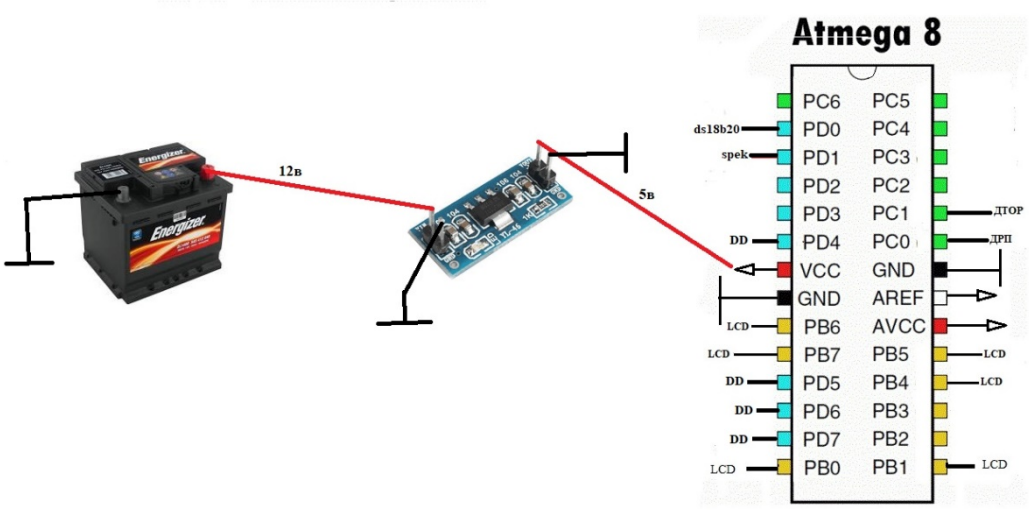


Рисунок 2,15: підключення понижувача напруги.

Схематична система в зборі окрім підключення пинижувача напруги розроблена в програмі Proteus. Зображена на рисунку 2,16

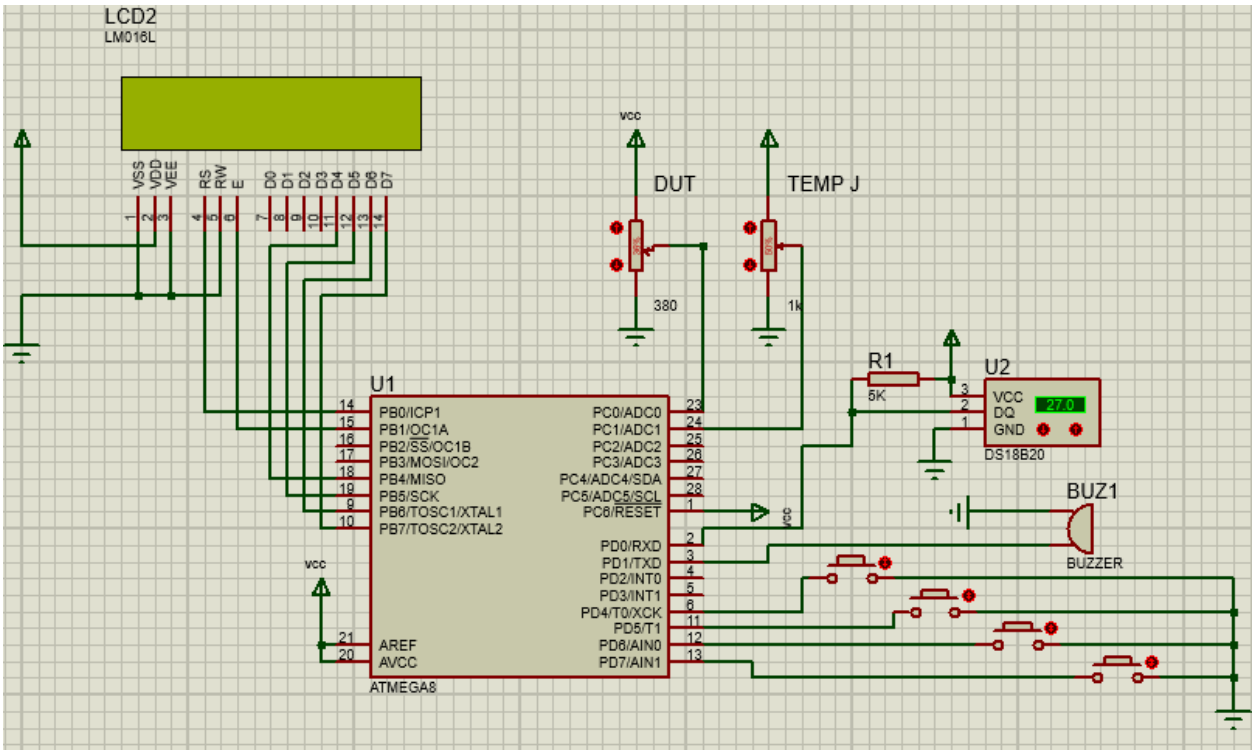


Рисунок 2,16: підключення всіх датчиків

Розробка програмної частини пристрою

Програмна частина була написана на Atmel Studio , на язику С. Були написані бібліотеки для кожного з датчиків та лсд дисплею(деякі частини коду запозичені). Бібліотеки.

Для того щоб почати розробляти програму треба спочатку задатися базисним методом роботи пристрою, потрібним для усвідомлення дій виконання різних ситуацій(малюнок 2.17).

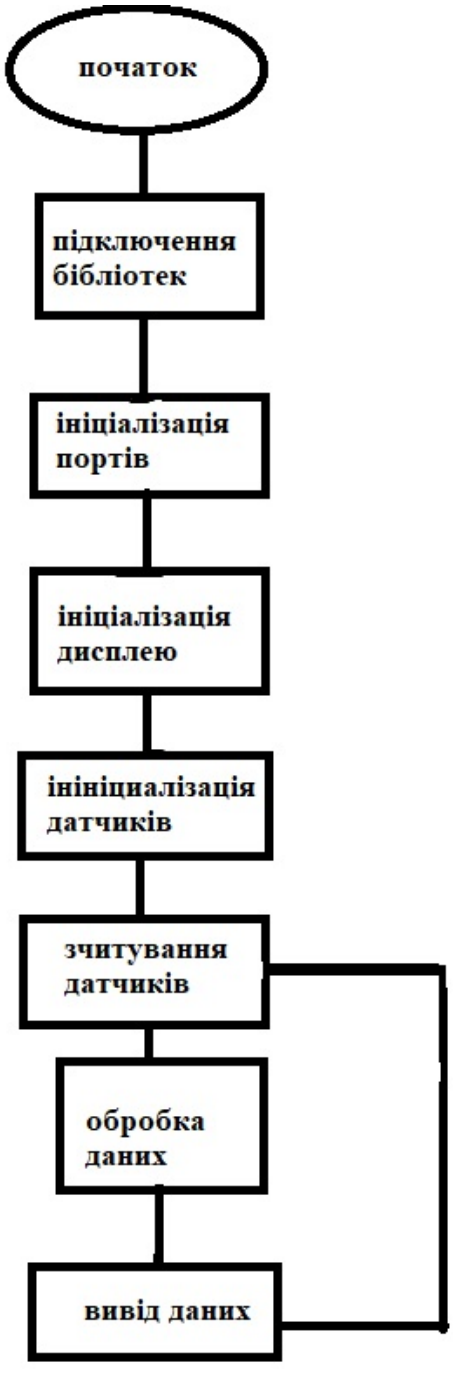


Рисунок 2,17: метод роботи пистрою

Програмна частина створенна на язику крограмування С. В програмі для програмування мікроконтролерів AVR Atmel studio. Яка була створена компанією AVR для розробки та прошивки програмного коду для їх мікро контролерів.

Для програмування були використані штатні бібліотеки такі як : <avr/io.h>,<avr/portpins.h>,<util/delay.h>,<avr/interrupt.h>.

Та були розроблені бібліотеки :"Lcd.h","DS18B20.h","ACD.h","Door.h"

Практична реалізація проекту

Виготовлення системи

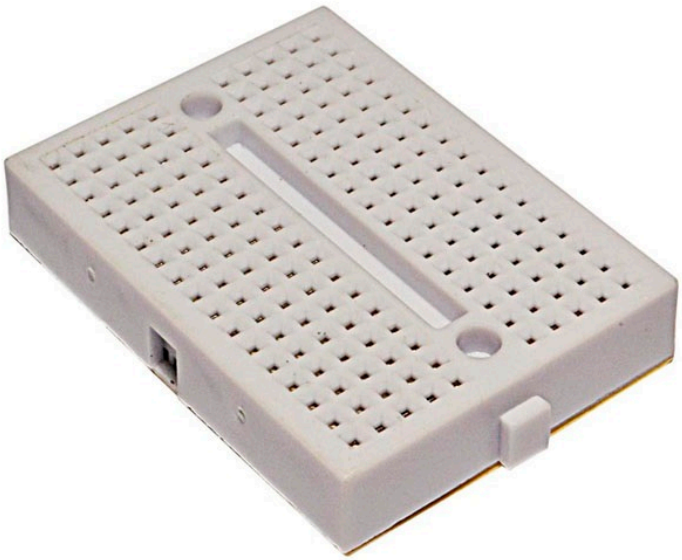
Для підключення різних модулів до мікроконтролера Atmega існує декілька способів:

1) використати спеціальну макетну плату, на якій зручно розташовувати різні елементи схеми, а також сполучати їх дротами між собою і мікроконтролером. Яку можна побачити на малюнку 3.1

2) Використати дроти мама-мама якими можна зручніше з'єднати елементи між собою оскільки вони займають менше місця і мають гнучкішу структуру що дозволяє зручно розташувати їх усередині корпусу які можна побачити на малюнку 3.2

3)Також використовуємо провідні проводи з січенням 0.7мм для з’єднання датчиків хола. Рисунок 3,3

4)для зєднання зі штатними датчиками використовуємо стандартні для підключення клеми мама. Рисунок 3,4



Малюнок 3.1-макетна плата

Малюнок 3.2- дроти мама-мама

Рисунок 3,3- провід 0.7мм

Рисунок 3,4 – клема мама.

Перевірка і відладка програмної частини пристрою

В процесі створення практичної моделі були труднощі по підк4люченю штатних датчиків. та розташуванням пристрою в торпеді автомобіля. Для вирішення цих питань було прийняте рішення помісти мікроконтролер у

бардачок. Кріплення датчиків хола та магнітів було вирішено приклеяти на супер клей (зрозуміле колхозне рішення). Lcd дисплей поставленно поверх основної приборної панелі.

Програмні трудноші з налаштування які виникли були усунуті за допомогою зміни деяких параметрів

Висновок

У такій випускній кваліфікаційній роботі здійснена розробка, створення робочої програми, виготовлення, відладка і працюючого макету облаштування контролю доступу в приміщення на мікроконтролерном управлінні.

Розроблена система дозволяє спостерігати за показаннями датчиків та обробки інформації з них і вивід коректних даних на дисплей.

На цьому прикладі можна зробити дадаткові розширення для цієї системи та обладнати його більш зручним виводом іформації .

Система розроблена для карбюраторних двигунів прошлого сторіча . однак може бути обладненна и на нові автомобілі з інжектором .

Список використаної літератури

1. https://www.rlocman.ru/datasheet/data.html?di=505479&/ATmega8 (електроний ресурс)
2. https://sxem.org/2-vse-stati/30-avto-skhemy/index.php?option=com_content&view=article&id=78&catid=24&Itemid=162(електроний ресурс)
3. https://www.waveshare.com/datasheet/LCD_en_PDF/LCD1602.pdf(електроний ресурс)
4. https://static.chipdip.ru/lib/029/DOC001029248.pdf(електроний ресурс)
5. https://narodstream.ru/avr-urok-20-podklyuchaem-datchik-temperatury-ds18b20-chast-2/(електроний ресурс)
6. https://micro-pi.ru/датчик-ds18b20-к-atmega8-lcd-hd44780/(електроний ресурс)

Додаток А

Лістинг програми облаштування

```
/*
 * DS18B20.c
 */
#include "DS18B20.h"
#include "main.h"
//функция определения датчика на шине
char dt_testdevice(void) //dt - digital termomether | определим, есть ли устройство на шине
{
	char stektemp=SREG;// сохраним значение стека
	cli(); //запрещаем прерывание
	char dt;
	DDRTEMP |= 1<<BITTEMP; //притягиваем шину
	_delay_us(485); //задержка как минимум на 480 микросекунд
	DDRTEMP &= ~(1<<BITTEMP); //отпускаем шину
	_delay_us(65); //задержка как максимум на 60 микросекунд
	if ((PINTEMP & (1<<BITTEMP))==0)//проверяем, ответит ли устройство
	{
		dt=1;//устройство есть
	}
	else dt=0;//устройства нет
	SREG = stektemp;// вернем значение стека
	_delay_us(420); //задержка как минимум на 480 микросекунд, но хватит и 420, тк это с учетом времени прошедших команд
	return dt; //вернем результат
}
```

```
//функция записи бита на устройство
void dt_sendbit(char bt)
{
	char stektemp=SREG;// сохраним значение стека
	cli(); //запрещаем прерывание
	DDRTEMP |= 1<<BITTEMP; //притягиваем шину
	_delay_us(2); //задержка как минимум на 2 микросекунды
	if(bt)
	DDRTEMP &= ~(1<<BITTEMP); //отпускаем шину
	_delay_us(65); //задержка как минимум на 60 микросекунд
	DDRTEMP &= ~(1<<BITTEMP); //отпускаем шину
	SREG = stektemp;// вернем значение стека
}
//функция записи байта на устройство
void dt_sendbyte(unsigned char bt)
{
	char i;
	for(i=0;i<8;i++)//посылаем отдельно каждый бит на устройство
	{
		if((bt & (1<<i)) == 1<<i)//посылаем 1
		dt_sendbit(1);
		else //посылаем 0
		dt_sendbit(0);
	}
}

//функция чтения бита с устройства
char dt_readbit(void)
{
	char stektemp=SREG;// сохраним значение стека
```

```
	cli(); //запрещаем прерывание
	char bt; //переменная хранения бита
	DDRTEMP |= 1<<BITTEMP; //притягиваем шину
	_delay_us(2); //задержка как минимум на 2 микросекунды
	DDRTEMP &= ~(1<<BITTEMP); //отпускаем шину
	_delay_us(13);
	bt = (PINTEMP & (1<<BITTEMP))>>BITTEMP; //читаем бит
	_delay_us(45);
	SREG = stektemp;// вернем значение стека
	return bt; //вернем результат
}

//функция чтения байта с устройства
unsigned char dt_readbyte(void)
{
	char c=0;
	char i;
	for(i=0;i<8;i++)
	c|=dt_readbit()<<i; //читаем бит
	return c;
}

//функция преобразования показаний датчика в температуру
int dt_check(void)
{
	unsigned char bt;//переменная для считывания байта
	unsigned int tt=0;
	if(dt_testdevice()==1) //если устройство нашлось
	{
		dt_sendbyte(NOID); //пропустить идентификацию, тк у нас только
```

```
одно устройство на шине
                dt_sendbyte(T_CONVERT); //измеряем температуру
                _delay_ms(750); //в 12битном режиме преобразования - 750 милисекунд
                dt_testdevice(); //снова используем  те же манипуляции с шиной что и при проверке ее присутствия
                dt_sendbyte(NOID); //пропустить идентификацию, тк у нас только одно устройство на шине
                dt_sendbyte(READ_DATA); //даем команду на чтение данных с устройства
                bt = dt_readbyte(); //читаем младший бит
                tt = dt_readbyte(); //читаем старший бит MS
                tt = (tt<<8)|bt;//сдвигаем старший влево, младший пишем на его место, тем самым получаем общий результат
        }
        return tt;

}

char converttemp (unsigned int tt)

{
        char t = tt>>4;//сдвиг и отсечение части старшего байта
        return t;
}
```

```
/*
 * DS18B20.h
 */
#ifndef DS18B20_H_
#define DS18B20_H_
//=========================================
#include "main.h"
//=========================================
#define NOID 0xCC //Пропустить идентификацию
#define T_CONVERT  0x44 //Код измерения температуры
#define READ_DATA 0xBE //Передача байтов ведущему



#define PORTTEMP PORTD
#define DDRTEMP DDRD
#define PINTEMP PIND
#define BITTEMP 0

int dt_check(void); //функция преобразования показаний датчика в температуру
char converttemp (unsigned int tt); //преобразование температуры в единицы

#endif /* DS18B20_H_ */
```

```
/*
 * Door.c
 */
#include "main.h"
#include "Door.h"
#include "Lcd.h"
//=======================================инициализация портов на прием=========================================
void Door_init (void)
{
	DDRD = 0x00;
	PIND = 0xf0;
}

//=============================================цикл для проверки дверей================================
void Door_Open (void)
{
	char door = 0;
	door = PORTD;//переносим значение с портов на которых двери

	switch(door)//сравниваем для определения какова дверь открыта
	{
		case 0xf0://все закрыты
		{//начальные позиции дверей
			Lcd_setcursor(0,14);
			Lcd_Data(0x7c); //PL
			Lcd_Data(0x7c);   //PP
			Lcd_setcursor(1,14);
			Lcd_Data(0x7c); //ZL
```

```
        Lcd_Data(0x7c);   //ZP
        break;//выводи начальные позиции
}
case 0xe0://открыта передня правая дальше как PP
{
        Lcd_setcursor(0,14);
        Lcd_Data(0x7c); //PL
        Lcd_Data(0x7e);   //PP
        Lcd_setcursor(1,14);
        Lcd_Data(0x7c); //ZL
        Lcd_Data(0x7c);   //ZP
        break;
}
case 0xd0://открыта передня левая дальше как PL
{
        Lcd_setcursor(0,14);
        Lcd_Data(0x7f); //PL
        Lcd_Data(0x7c);   //PP
        Lcd_setcursor(1,14);
        Lcd_Data(0x7c); //ZL
        Lcd_Data(0x7c);   //ZP
        break;
}
case 0xc0:// PP PL
{
        Lcd_setcursor(0,14);
        Lcd_Data(0x7f); //PL
        Lcd_Data(0x7e);   //PP
        Lcd_setcursor(1,14);
        Lcd_Data(0x7c); //ZL
```

```
        Lcd_Data(0x7c);   //ZP
        break;
}
case 0xb0:// открыта задня правая ZP
{
        Lcd_setcursor(0,14);
        Lcd_Data(0x7c); //PL
        Lcd_Data(0x7c);   //PP
        Lcd_setcursor(1,14);
        Lcd_Data(0x7c); //ZL
        Lcd_Data(0x7e);   //ZP
        break;
}
case 0xa0:// PP ZP
{
        Lcd_setcursor(0,14);
        Lcd_Data(0x7c); //PL
        Lcd_Data(0x7e);   //PP
        Lcd_setcursor(1,14);
        Lcd_Data(0x7c); //ZL
        Lcd_Data(0x7e);   //ZP
        break;
}
case 0x90:// PL ZP
{
        Lcd_setcursor(0,14);
        Lcd_Data(0x7f); //PL
        Lcd_Data(0x7c);   //PP
        Lcd_setcursor(1,14);
        Lcd_Data(0x7c); //ZL
```

```
            Lcd_Data(0x7e);   //ZP
            break;
    }
    case 0x80:// PP PL ZP
    {
            Lcd_setcursor(0,14);
            Lcd_Data(0x7f); //PL
            Lcd_Data(0x7e);   //PP
            Lcd_setcursor(1,14);
            Lcd_Data(0x7c); //ZL
            Lcd_Data(0x7e);   //ZP
            break;
    }
    case 0x70://открыта задняя лева дальше ZL
    {
            Lcd_setcursor(0,14);
            Lcd_Data(0x7c); //PL
            Lcd_Data(0x7c);   //PP
            Lcd_setcursor(1,14);
            Lcd_Data(0x7f); //ZL
            Lcd_Data(0x7c);   //ZP
            break;
    }
    case 0x60:// PP ZL
    {
            Lcd_setcursor(0,14);
            Lcd_Data(0x7c); //PL
            Lcd_Data(0x7e);   //PP
            Lcd_setcursor(1,14);
            Lcd_Data(0x7f); //ZL
```

```
            Lcd_Data(0x7c);   //ZP
            break;
    }
    case 0x50:// PL ZL
    {
            Lcd_setcursor(0,14);
            Lcd_Data(0x7f); //PL
            Lcd_Data(0x7c);   //PP
            Lcd_setcursor(1,14);
            Lcd_Data(0x7f); //ZL
            Lcd_Data(0x7c);   //ZP
            break;
    }
    case 0x40:// PP PL ZL
    {
            Lcd_setcursor(0,14);
            Lcd_Data(0x7f); //PL
            Lcd_Data(0x7e);   //PP
            Lcd_setcursor(1,14);
            Lcd_Data(0x7f); //ZL
            Lcd_Data(0x7c);   //ZP
            break;
    }
    case 0x30:// ZP ZL
    {
            Lcd_setcursor(0,14);
            Lcd_Data(0x7c); //PL
            Lcd_Data(0x7c);   //PP
            Lcd_setcursor(1,14);
            Lcd_Data(0x7f); //ZL
```

```
            Lcd_Data(0x7e);   //ZP
            break;
    }
    case 0x20:// PP ZP ZL
    {
            Lcd_setcursor(0,14);
            Lcd_Data(0x7c); //PL
            Lcd_Data(0x7e);   //PP
            Lcd_setcursor(1,14);
            Lcd_Data(0x7f); //ZL
            Lcd_Data(0x7e);   //ZP
            break;
    }
    case 0x10:// PL ZP ZL
    {
            Lcd_setcursor(0,14);
            Lcd_Data(0x7f); //PL
            Lcd_Data(0x7c);   //PP
            Lcd_setcursor(1,14);
            Lcd_Data(0x7f); //ZL
            Lcd_Data(0x7e);   //ZP
            break;
    }
    case 0x00:// PP PL ZP ZL
    {
            Lcd_setcursor(0,14);
            Lcd_Data(0x7f); //PL
            Lcd_Data(0x7e);   //PP
            Lcd_setcursor(1,14);
            Lcd_Data(0x7f); //ZL
```

```
                Lcd_Data(0x7e);   //ZP
                break;
            }
        }

}
```

```
/*файл для функций датчиков двери
 */


//=================================================================
=======
#ifndef DOOR_H_
#define DOOR_H_
//=================================================================
=======
#include "main.h"
//=================================================================
=======
void Door_init (void);
void Door_Open (void);




#endif /* DOOR_H_ *
```

```
/*
 * Lcd.c
 */
#include "main.h"
#include "Lcd.h"
//=============================================инициализация портов Lcd==============================
void Lcd_PortInit (void)
{
	DDRB = 0xff;
	PORTB = 0x00;
}
//============================передача половины байта (младшая тетрада) на lcd============================
void Lcd_SetHalfBit(unsigned char d)
{
	d<<=4;
	e1;
	_delay_us(50);
	PORTB &=0b00001111;
	PORTB |=d;
	e0;
	_delay_us(50);
}
//=====================================передача полного байта=======================================
void Lcd_SetByte(unsigned char d,unsigned char mood)
{
	if (mood==0)//Определям команда или даные в отсылаются
	{
```

```
        rs0;
    }
    else
    {
        rs1;
    }

    unsigned char bd=0;//вводим переменую
    bd = d>>4;//записываем в нее весь байт //сдвигаем старшую тетраду в младшую

    Lcd_SetHalfBit(bd);//выводим младшую и старшую тетраду
    Lcd_SetHalfBit(d);
}
//=========================================задаем работу lcd=====================================
void Lcd_Init (void)
{
    //включение 4 битного режима
    _delay_ms(15);
    Lcd_SetHalfBit(0b00000011);
    _delay_ms(4);
    Lcd_SetHalfBit(0b00000011);
    _delay_us(100);
    Lcd_SetHalfBit(0b00000011);
    _delay_ms(1);
    Lcd_SetHalfBit(0b00000010);
    _delay_ms(1);
    Lcd_SetByte(0b00101000,0);//4 bit и 2 линии
    _delay_ms(1);
```

```
        Lcd_SetByte(0b00001100,0);//включае дисплей и курсор
        _delay_ms(1);
        Lcd_SetByte(0b00000110,0);//передвижение курсора



}
//==================================выбор положения курсора====================
void Lcd_setcursor(unsigned char x, unsigned char y)
{
        char adres=0 ;
        adres= (0x40*x+y)|0b10000000;
        Lcd_SetByte(adres,0);

}
//==================================вывод команды==============================
void Lcd_Comand (unsigned char c)
{
        Lcd_SetByte(c,0);
}
//==================================вывод даных================================
void Lcd_Data (unsigned char d)
{
        Lcd_SetByte(d,1);

}
//==================================начальний вид
```

панели====================

```
void Lcd_panel(void)
{
        Lcd_setcursor(0,0);
        Lcd_Data('t');
        Lcd_Data('s');
        Lcd_setcursor(1,0);
        Lcd_Data('t');
        Lcd_Data('j');
        Lcd_setcursor(0,6);
        Lcd_Data(0xdf);
        Lcd_Data('C');
        Lcd_Data(0x7c);
        Lcd_Data('F');
        Lcd_Data('u');
        Lcd_Data('e');
        Lcd_Data('l');
        Lcd_Data(0x7c);
        Lcd_setcursor(1,6);
        Lcd_Data(0xdf);
        Lcd_Data('C');
        Lcd_Data(0x7c);
        Lcd_setcursor(1,13);
        Lcd_Data(0x7c);
}
```

```
/*файл для функций Lcd дисплея
 */
//=============================================================
=======
#ifndef LCD_H_
#define LCD_H_
//=============================================================
=======
#include "main.h"
//==============================константы=======================
===============
#define e1          PORTB|=0b00000010; //инициируем сигнал записи
#define e0          PORTB&=0b11111101; //сбрасуем сигнал записи
#define rs1         PORTB|=0b00000001; //инициируем сигнал команды
#define rs0         PORTB&=0b11111110; //сбрасуем сигнал команды
//============================функции=========================
==============
void Lcd_PortInit (void);
void Lcd_Init (void);
void Lcd_setcursor(unsigned char x, unsigned char y);
void Lcd_Data (unsigned char d);
void Lcd_panel(void);
#endif /* LCD_H_ */
```

```
/*
 * ACD.c
 */

#include "main.h"
//--------------------инициализация АЦП---------------------
void ADC_DUT_init(void)
{
	ADCSRA = 0b10000111; //включаем АЦП и выбераем делитель
	ADMUX = 0b00100000; //выбераем сравнение с внешнего источника , режим старшего бита , выбор порта PC0
}
//----------------------начало работы---------------------------------

int ADC_start(void)
{
	ADCSRA |= (1<<ADSC);//запуснаем компорацию
	while ((ADCSRA &(1<<ADSC)));
	return ADCH;
}
//--------------------------вывод коректных даных------------------------------
void ADC_DUT_Data(void)
{
	unsigned char data_adc = 0;
	Lcd_setcursor(1,9);
	float d;
	data_adc = ADC_start();
	d = (float) data_adc/6.5;
	Lcd_Data((unsigned char) d/10 + 0x30);
	Lcd_Data((unsigned char) d%10 + 0x30);
```

```
        Lcd_Data(',');
        Lcd_Data((unsigned char) (d*10)%10 + 0x30);

}
//==================t
void ADC_T_init(void)
{
        ADCSRA = 0b10000111; //включаем АЦП и выбераем делитель
        ADMUX = 0b00100001; //выбераем сравнение с внешнего источника , режим старшего бита , выбор порта PC0
}

//--------------------------вывод коректных даных------------------------------
void ADC_T_Data(void)
{
        unsigned char data_adc = 0;
        Lcd_setcursor(1,9);
        float d;
        data_adc = ADC_start();
        d = (float) data_adc/1,8;
        Lcd_Data((unsigned char) d/100 + 0x30);
        Lcd_Data((unsigned char) d/10 + 0x30);
        Lcd_Data((unsigned char) d%10 + 0x30);

}
```

```
/*файл для функций АЦП
 */


#ifndef ACD_H_
#define ACD_H_

#include "Main.h"
#include "LCD.h"
//=========================================================
void ADC_DUT_Data(void);
int ADC_start(void);
void ADC_DUT_init(void);
void ADC_T_Data(void);
void ADC_T_init(void);

#endif /* ACD_H_ */
```

```
/*файл для подключения всех библиотек
 */

//=================================================================
=======
#ifndef MAIN_H_
#define MAIN_H_
//============================встроеные
библиотеки=====================================
#define F_CPU 8000000
#include <avr/io.h>
#include <avr/portpins.h>
#include <avr/interrupt.h>
#include <util/delay.h>
//============================пользовательские
библиотеки=====================================
#include "Lcd.h"
#include "DS18B20.h"
#include "ACD.h"
#include "Door.h"

#endif /* MAIN_H_ */
```

```c
#include "main.h"

int main(void)
{
	unsigned int tt =0 ;

	Door_init();
	Lcd_PortInit();
	Lcd_Init();
	Lcd_panel();
	while(1)
   {
	Door_Open();
	ADC_DUT_init();
	ADC_start();
	ADC_DUT_Data();
	ADC_T_init();
	ADC_start();
	ADC_T_Data();
	tt = converttemp(dt_check());
	Lcd_Data(tt/100+0x30);
	Lcd_Data(tt/10+0x30);
	Lcd_Data(tt%10+0x30);


	}
}
```