МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

Присвячується 60-річчю
кафедри систем управління
літальними апаратами

Л. О. Краснов, К. Ю. Дергачов, О. В. Плахотний, І. О. Пявка

ОСНОВИ ПОБУДОВИ СУЧАСНИХ МОБІЛЬНИХ СИСТЕМ ТЕХНІЧНОГО ЗОРУ

Частина 3
ЛАБОРАТОРНІ РОБОТИ

Навчальний посібник

Харків «ХАІ» 2019

УДК 004.942(075.8)
К78

Колектив авторів:
Л. О. Краснов, К. Ю. Дергачов, О. В. Плахотний, І. О. Пявка

Рецензенти: канд. техн. наук Ю. О. Кузнєцов,
канд. техн. наук С. М. Флерко

Краснов, Л. О.

К78 Основи побудови сучасних мобільних систем технічного зору [Текст] : навч. посіб. У 3 ч. Ч. 3. Лабораторні роботи / Л. О. Краснов, К. Ю. Дергачов, О. В. Плахотний, І. О. Пявка. – Харків : Нац. аерокосм. ун-т ім. М. Є. Жуковського «Харків. авіац. ін-т», 2019. – 72 с.

ISBN 978-966-662-659-5

Подано матеріали для практичного вивчення методів будування сучасних систем технічного зору. Розглянуто питання застосування мови програмування Python разом з бібліотекою для оброблення зображень OpenCV на базі мікрокомп'ютерів Raspberry Pi з метою вирішення завдань оброблення даних і керування процесом відеоспостереження. Наведено приклади практичної роботи з реальними зображеннями й даними відеокамер.

Для студентів, що навчаються за напрямами підготовки «Авіоніка», «Аеронавігація» і «Системна інженерія» спеціальностей «Системи управління літальними апаратами і комплексами», «Комп`ютеризовані системи управління і автоматика», «Аеронавігаційні системи і системи аеронавігаційного обслуговування».

Іл. 43. Бібліогр.: 14 назв

УДК 004.942(075.8)

ISBN 978-966-662-659-5

ПЕРЕДМОВА

Створення сучасних мобільних систем технічного зору потребує від розробників уже на етапі ескізного проектування вирішення цілого комплексу питань щодо обгрунтування й вибору апаратної платформи, комп'ютера й засобів програмування.

Ці завдання було досить детально розглянуто в першій частині пропонованого посібника. На сучасному етапі оптимальним з огляду на ефективність роботи, мобільність і вартість системи є використання одноплатної комп'ютерної платформи ***Raspberry Pi*** з базовою операційною системою ***Raspbian*** і мови програмування ***Python***.

У другій частині навчального посібника було детально обговоррено питання використання основних ресурсів оброблення зображень і відео в системах технічного зору. Розглянуто програмування мовою ***Python*** основних алгоритмів оброблення зображень з використанням бібліотеки ***Pillow*** (Python Imaging Library). Велику увагу приділено підімкненню й роботі в програмі ***Python*** бібліотеки оброблення зображень ***OpenCV*** (Open Source Computer Vision Library). Використання в'язки ***Python*** і бібліотеки ***OpenCV*** дає змогу проводити оброблення відеоінформації в реальному масштабі часу із застосуванням найсучасніших алгоритмів оброблення відеоданих.

Третя частина навчального посібника містить теоретичні й методичні матеріали для виконання циклу лабораторних робіт з використання мікрокомп'ютерів ***Raspberry Pi*** як засобів керування сучасними мобільними системами технічного зору з використанням апаратних портів уведення/виведення ***GPIO*** (General Purpose Input/output). Особливо корисною є технологія створення систем дистанційного керування відеосенсорами з допомогою сервоприводів з можливістю передання зображень у віддалений центр збору й оброблення відеоінформації. Ці завдання успішно вирішуються завдяки доступу до портів ***GPIO*** через веб-сервер-інтерфейс, що дає змогу контролювати стан і керувати всіма портами ***GPIO*** локально або віддалено з браузера або будь-якого застосування.

Викладені матеріали допоможуть набути достатніх практичних навичок і, безперечно, будуть корисними читачеві при самостійній роботі в цій області.

ВСТУП

У першій частині цього навчального посібника при аналізі можливих шляхів будування систем технічного зору було показано, що як апаратну платформу доцільно використовувати одноплатний комп'ютер ***Raspberry Pi.*** Таке рішення є оптимальним з огляду на прийнятні массо-габарітні показники, високу обчислювальну продуктивність, сумісність з широко використовуваними стандартними операційними системами й мовами програмування. Однак однією з дуже корисних і привабливих особливостей комп'ютера ***Raspberry Pi*** є наявність на платі апаратних портів уведення/виведення ***GPIO*** (General Purpose Input/output, входи/виходи загального призначення). Це дає змогу використовувати його в різних робототехнічних проектах і проектах зі створення систем технічного зору. На базі цих портів реалізовано інтерфейси ***UART***, консольний порт, ***SPI*** (Serial Peripheral Interface, послідовний периферійний інтерфейс) і $\boldsymbol{I^2c}$ (Inter-integrated Circuit − послідовна шина даних для зв'язку інтегральних схем). Виводи ***UART***, ***SPI*** і $\boldsymbol{I^2c}$ у разі необхідності можна настроїти як звичайні порти введення/виведення.

Тому для виконання пропонованих лабораторних робіт необхідно вивчити базові методи керування зовнішніми пристроями з допомогою виводів ***GPIO*** комп'ютера ***Raspberry Pi*** і вирішити кілька практичних завдань для закріплення отриманих навичок.

Конструктивнє виконання Raspberry Pi

Зовнішній вигляд плати комп'ютера показано на рис. В.1.

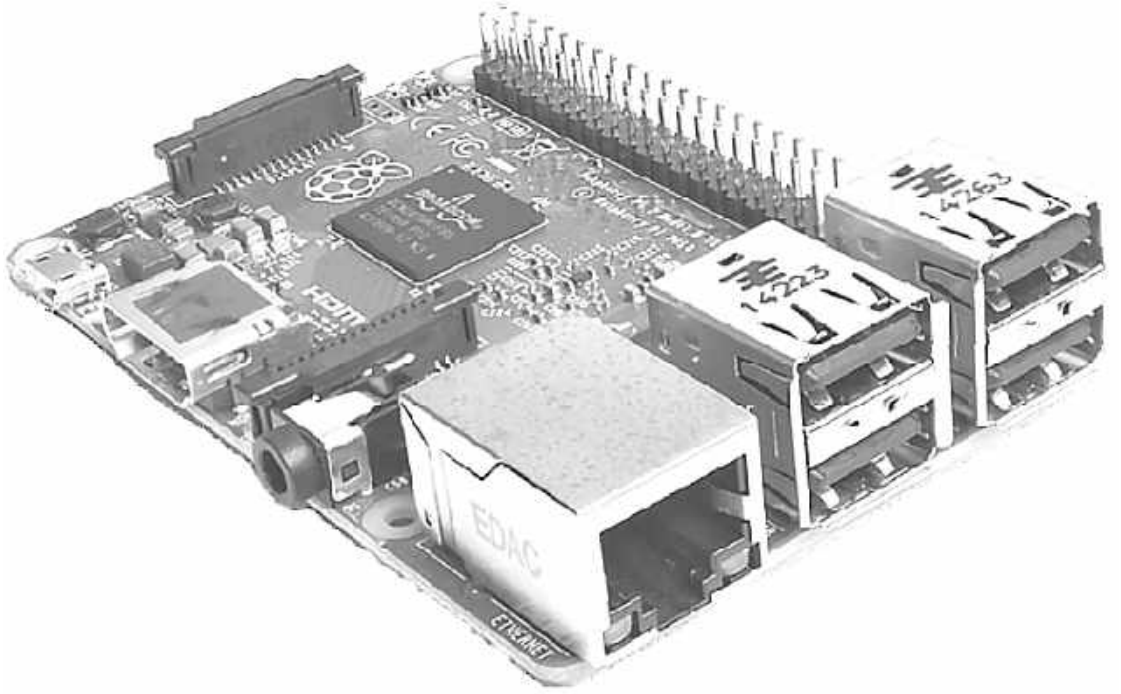


Рис. В.1

Апаратним портом уведення/виведення ***GPIO*** є 26- або 40-вивідний рознім, що містить загальні порти введення/виведення, інтерфейси UART, I^2C, SPI, I^2S, лінії 3.3 V, 5 V, GND, GPCLK. Розміщення цього рознімуна платі Raspberry Pi 2 (модель B) показано на рис. B.2.

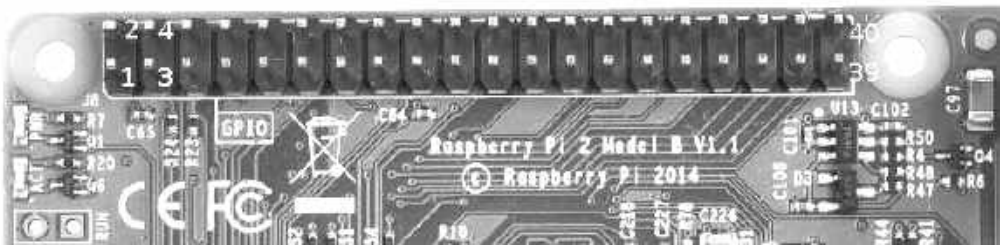


Рис. B.2

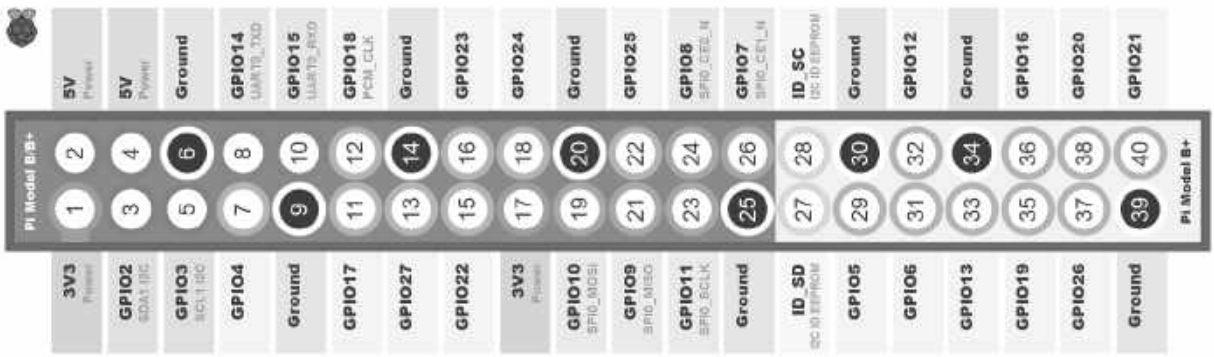
Оскільки порти ***GPIO*** (General Purpose Input/output) на ***Raspberry Pi*** не підписано, наводимо роздрук цоколівки (розпіновки) цього рознімуна (рис. B.3). Це завжди необхідно пам'ятати при створенні проектів з керування зовнішніми пристроями.

3.3V	1	2	5V
I2C0 SDA	3	4	DNC
I2C0 SCL	5	6	GROUND
GPIO4	7	8	UART TXD
DNC	9	10	UART RXD
GPIO 17	11	12	GPIO 18
GPIO 21	13	14	DNC
GPIO 22	15	16	GPIO 23
DNC	17	18	GPIO 24
SP10 MOSI	19	20	DNC
SP10 MISO	21	22	GPIO 25
SP10 SCLK	23	24	SP10 CE0 N
DNC	25	26	SP10 CE1 N

Модель A − GPIO типу 1 (26 pin)

3.3V	1	2	5V
I2C1 SDA	3	4	5V
I2C1 SCL	5	6	GROUND
GPIO4	7	8	UART TXD
GROUND		10	UART RXD
GPIO 17	11	12	GPIO 18
GPIO 27	13	14	GROUND
GPIO 22	15	16	GPIO 23
3.3V	17	18	GPIO 24
SP10 MOSI	19	20	GROUND
SP10 MISO	21	22	GPIO 25
SP10 SCLK	23	24	SP10 CE0 N
GROUND	25	26	SP10 CE1 N

Модель B − GPIO типу 2 (26 pin)



Модель B − ***GPIO*** (40 pin)

Рис. B.3

Призначення виводів ***GPIO*** (так звану розпіновку) показано на рис. В.3. У першій версії плати виводи 4, 9, 14, 17, 20, 25 позначено як DNC (Do Not Connect), і під'єднувати до них що-небудь не можна, оскільки плата може згоріти. На нових версіях плати є розведеними, але не розпаяними, ще чотири ***GPIO*** мають додаткові виводи I^2C і I^2S (Inter-integrated Circuit, послідовна шина даних для зв'язку інтегральних схем). Зазначимо, що інтерфейс ***GPIO*** має 40 пінів, перші 26 контактів є ідентичними моделі A, що забезпечує 100-відсоткову зворотну сумісність для проектів.

Запобіжні заходи під час роботи з інтерфейсом GPIO

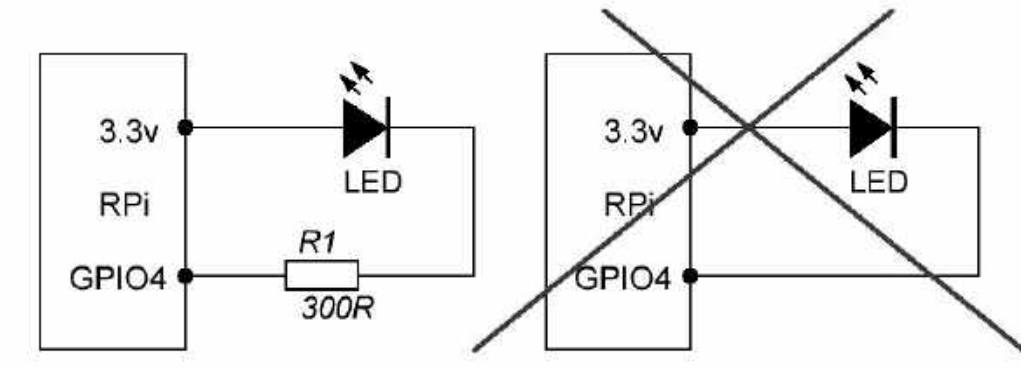
Під час роботи з портами ***GPIO*** слід пам'ятати про деякі їх особливості й дотримуватися певних запобіжних заходів, щоб не пошкодити ***Raspberry Pi***. Перелічимо основні з них:

- максимальний сумарний струм обох виводів 3,3 V становить 50 mA, і ці виводи можна використовувати для живлення зовнішніх пристроїв лише в тому випадку, якщо їх споживаний струм є меншим за 50 mA;
- максимальний сумарний струм обох виводів 5 V становить 300 mA, і ці виводи також можна використовувати для живлення зовнішніх пристроїв лише в тому випадку, якщо їх споживаний струм є меншим за 300 mA;
- на ***GPIO*** не можна подавати напругу понад 3,3 V; цифрові виводи ***GPIO*** мають однакові напруги 0…3,3 V і не є сумісними з традиційними рівнями напруги 0…5 V. Якщо подати на вивід ***GPIO*** логічну одиницю, що становить 5 V (а не 3,3 V), то цей вивід може вийти з ладу;
- виводи GPIO14 і GPIO15 за замовчуванням виконують альтернативну функцію і є виводами UART (RXD і TXD), тому після вмикання на них установиться високий рівень напруги 3,3 V, проте програмно їх можна переконфігурувати на звичайні виводи. Усі останні виводи ***GPIO*** після вмикання ***Raspberry Pi*** виконують основну функцію і працюють як звичайні цифрові;
- усі виводи ***GPIO***, що настроюються (окрім GPIO0 (SDA) і GPIO1 (SCL)), за замовчуванням є входами, і тому мають високий вхідний опір, при цьому підтягнення логічного рівня в них не ввімкнено, так що після ввімкнення ***Raspberry Pi*** напруга на них може «плавати»;
- виводи GPIO0 (SDA) і GPIO1 (SCL) за замовчуванням «підтягнуто» до живлення, тому після ввімкнення ***Raspberry Pi*** на них є напруга логічної одиниці (3,3 V);
- сигнал на будь-якому з цифрових виводів може бути джерелом зовнішнього переривання.

Слід пам'ятати, що ***GPIO*** – це виводи, безпосередньо підімкнені до процесора ***Raspberry Pi***, які є інструментом для взаємодії з ним. Тому необережне поводження з ***GPIO*** може призвести до необоротних наслідків для процесора.

При проектуванні пристроїв, у яких використовується велика кількість контактів ***GPIO,*** необхідно обов'язково робити розв'язку через додаткові буферні схеми, перетворювачі рівня напруги або електронні ключі для того, щоб забезпечити комп'ютер від можливого пошкодження (вигорання) буферів усередині чіпа.

Наведемо простий приклад – підімкнення світлодіода до одного з портів ***GPIO***. На рис. В.4 показано два варіанти його підімкнення: *а* – правильний, *б* – неправильний.

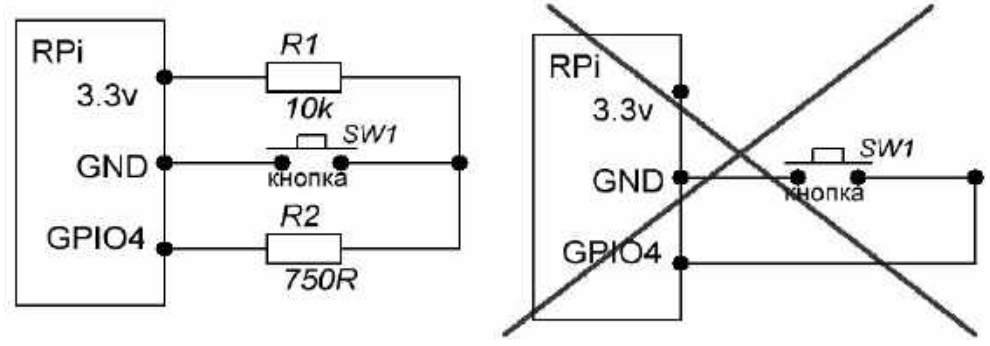


а *б*

Рис. В.4

Як видно з рисунка, при правильному підімкненні є обмежувальний (гасильний) резистор, який лімітує струм через світлодіод і контакт GPIO4. При прямому підімкненні без резистора на світлодіод надходить напруга 3,3 V, що є явно більше від норми для світлодіодів (2–3 V). Пряма напруга такої величини може спричинити досить великий струм у ланцюзі (50 mA), що може призвести до вигорання світлодіода і виходу з ладу як окремого буфера ***GPIO***, так і процесора в цілому.

Розглянемо ще один приклад – підімкнення кнопки до ***GPIO*** з використанням опорної напруги її обмежувальних резисторів (рис. В.5, *а*).



а *б*

Рис. В.5

У разі прямого підімкнення кнопки (рис. В.5, *б*) виникає ймовірність вигорання використовуваного виходу, оскільки під час ініціалізації системи або неправильного установлення режимів порту (конфігурованого як OUTPUT) на використовуваний контакт може надійти напруга +3,3 V, що, у свою чергу, при замиканні кнопки спричинить коротке замикання порту із землею GND.

Якщо до кожного контакту ***GPIO*** підімкнути світлодіод, то можна їх загальну точку з'єднати із землею. У цьому випадку світлодіод світитиметься, якщо на контакті буде високий (+3,3 V) рівень напруги. Для обмеження струму до 15 mA послідовно світлодіоду підімкнемо резистор.

Контакти з напругою 5 V підімкнено до основної шини живлення 5 V плати (після стабілізатора), оскільки велике навантаження може вплинути на стабільність роботи платформи в цілому, а також вивести з ладу внутрішній стабілізатор напруги +5 V. Використовувати їх слід обережно, оскільки допустимий струм не повинен перевищувати 1 А (без застосування самої плати): 1 А…700 мА (струм, що споживає плата моделі В становить 300 mA).

Насправді, струм, який споживає ***Raspberry Pi***, багато в чому залежить від завантаженості мікропроцесора й відеопроцесора (графічна оболонка потребує набагато більше ресурсів, ніж консольний режим), а також від кількості підімкнених пристроїв до ***USB***-портів і їх застосування. Значення загального споживаного струму може варіюватися в межах від 250 mA до 1 А.

Raspberry Pi, як і будь-яку іншу чутливу електроніку, може пошкодити статична електрика. Перед роботою бажано прибрати статичний заряд, що накопичився на тілі, зняти із себе шерстяний одяг.

Перед паянням модулів і плат бажано відімкнути їх від розніму ***GPIO***. Монтаж рекомендується виконувати паяльником з напругою 36 V (або нижче) з використанням знижувального трансформатора 220 / 36 V тощо.

Необхідно дотримуватися таких заходів безпеки:

- перед роботою зняти із себе статичний заряд;
- виконувати паяння електроніки з відімкненим рознімом ***GPIO***;
- не перенавантажувати піни 3,3 V (50 mA) і не закорочувати їх безпосередньо з іншими пінами;
- не перенавантажувати піни 5 V (300…500 mA) і не закорочувати їх безпосередньо з іншими пінами;
- виключити протікання струму величиною понад 15 mA через піни ***GPIO***;
- виключити попадання напруги понад 3,3 V на піни ***GPIO***;
- підмикати безпосередньо до ***Raspberry Pi*** не більше трьох світлодіодів (до 15 mA кожен);
- уважно перевірити правильність підімкнення перед поданням живлення на схему і комп'ютер.

Розрахунок резисторів, що обмежують струм, для світлодіодів

Зазвичай для експериментів зі світлодіодами буде досить гасильного резистора на 270…600 Ом. При більшому опорі світлодіод світитиметься тьмяніше, а при меншому – яскравіше.

Для точнішого розрахунку опору гасильного резистора (при живленні від лінії 3,3 V) використаємо формулу

$$R = \frac{3.3V - U_д}{I_д},$$

де R – опір гасильного резистора; $U_д$ – робоча напруга світлодіода; $I_д$ – робочий струм світлодіода.

Наведемо типові значення напруги для світлодіодів різного кольору:

- червоний – 1,8...2 V;
- жовтий і зелений – 2...2,4 V;
- білий і синій (яскраві) – 3...3,5 V.

Робочий струм світлодіодів залежно від типу може коливатися в межах 10…25 мА. Для точніших розрахунків краще уточнити параметри конкретного світлодіода в довіднику. Для експериментів, що проводяться, струм 10–15 мА буде достатнім і безпечним. Наприклад, для світлодіода АЛ307 червоного кольору розрахуємо гасильний резистор при струмі 10 мА: R = (3,3 − 2) / 0,01 = 130 Ом.

При опорі гасильного резистора 130 Ом і живильній напрузі 3,3 V світлодіод світитиме досить яскраво. Можна поекспериментувати з величиною опору. Наймовірніше, при опорі 300 Ом яскравість свічення буде цілком достатньою для експериментів з ***GPIO*** в ***Raspberry Pi*** і перевантаження порту не буде.

Автоматизована система для створення схем експерименту

Для успішного проведення досліджень зазвичай необхідно використовувати нескладні засоби автоматизації, що дають змогу прискорити процес формування супровідної документації. У цьому випадку необхідно скористатися пакетом ***Fritzing***. Це − програмне забезпечення з відкритим кодом для віртуального моделювання електричних ланцюгів, схем і електронного устаткування. Програмний пакет ***Fritzing*** можна використовувати для створення прототипу схеми на макетній платі, а також автоматичної генерації принципових схем і друкованих плат.

Установка Fritzing. Цей пакет є доступним для вільного скачування на офіційному сайті http:/fritzing.org. Для встановлення необхідно вибрати свою операційну систему і далі дотримуватись інструкцій на сторінці. Спочатку в пакеті є велика кількість бібліотек різних компонентів. Якщо немає необхідного компонента, то можна встановити нову бібліотеку або додати окремий компонент.

Початок роботи. Відкривши ***Fritzing,*** користувач побачить вікно (рис. В.6).

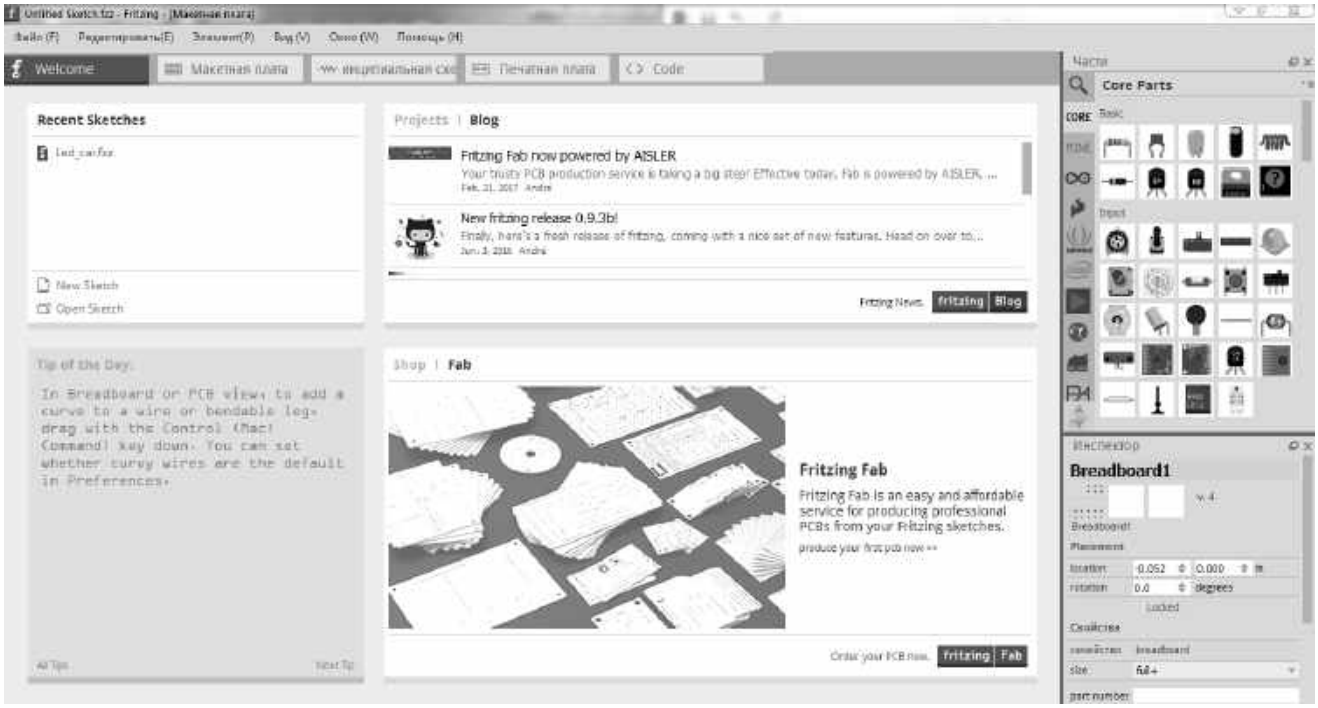


Рис. В.6

Поряд з вкладкою привітання 

знаходяться вкладки проектування і створення макетної плати, принципової схеми й друкованої плати. Після переходу на вкладку виникне вікно 

(рис. В.7).

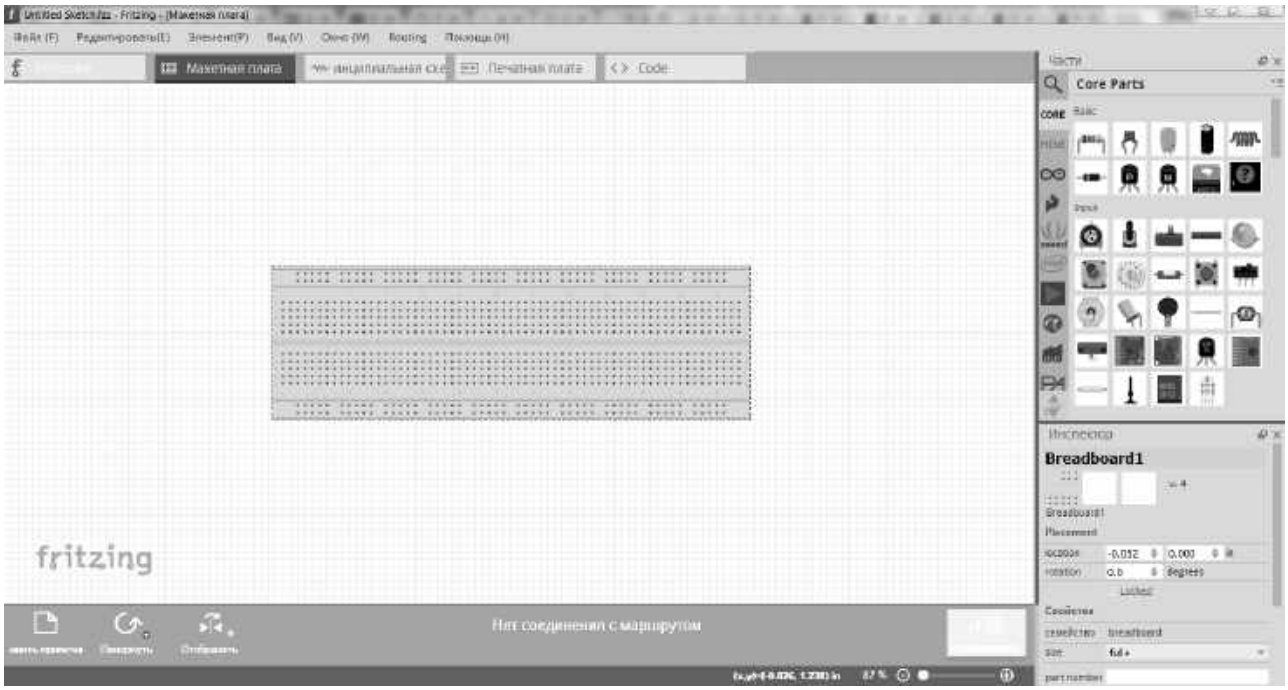


Рис. В.7

У правій частині екрану знаходиться панель інструментів зі всіма елементами й опціями (рис. В.8, *а*). Якщо вибраний компонент необхідно настроїти, то в нижній частині панелі інструментів відображують параметри для настроюванняцього компонента (рис. В.8, *б*).

а *б*

Рис. В.8

Складання схеми для виконання експерименту

Розглянемо створення схеми експерименту на прикладі схеми керування роботою світлодіода з допомогою кнопки. Обмеження струму світлодіода здійснюється резистором. Для того щоб скласти цю схему, виберемо й помістимо резистор на робочу область. Перемістимо резистор на макетну плату так, щоб кожен вивід потрапив на окремий стовпець на платі. Коли компонент підімкнеться до тієї чи іншої колонки, весь стовпець стане ясно-зеленим, як показано на рис. В.9. Зелена лінія вкаже на електричне з'єднання між отворами.

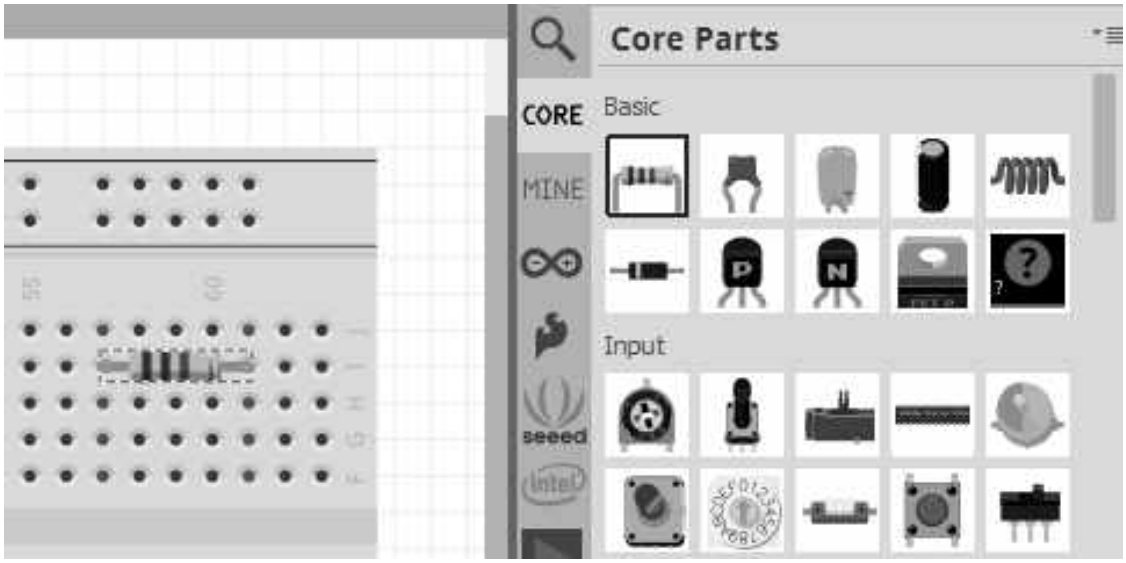


Рис. В.9

Параметри виділеного елемента (опір, допуск, відстань між виводами) можна настроїти в нижній частині панелі інструментів (рис. В.10). Слід зазначити, що відстань між виводами задається в мілах (mil), 1 mil – це 1/1000 дюйма.

Рис. В.10

Потім поміщаємо світлодіод на платі поряд з резистором, як показано на рис. В.11. Поки резистор і світлодіод не підімкнено до джерела живлення або один до одного. Додамо також кнопку на полі і з допомогою пошуку елементів знайдемо й помістимо ***Raspberry Pi*** поряд з макетною платою (розмір макетної плати заздалегідь змінено для зручності роботи). Після проведених дій усі елементи додано і вікно програми має вигляд, який показано на рис. В.11.

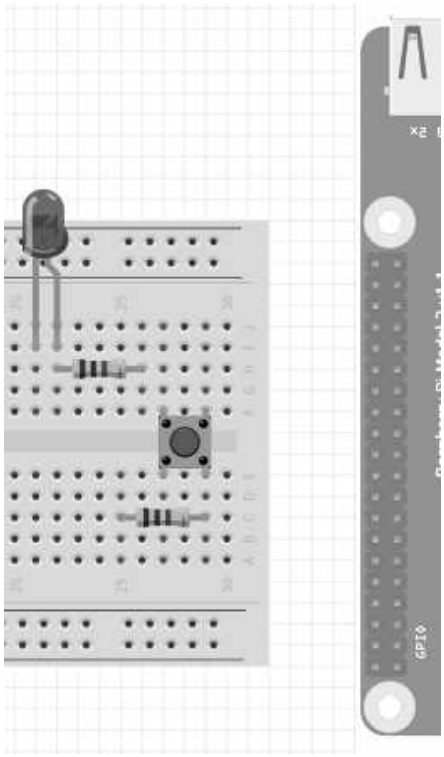


Рис. В.11

Поки всі елементи просто розташовано на макетній платі. Для їх підімкнення необхідно навести курсор миші на отвір на макетній платі, яке при цьому зафарбується в синій колір. Це означає, що можна починати вводити провід. Вибравши отвір на макетній платі і не відпускаючи лівої кнопки миші, провести другий кінець проводу в необхідний отвір. Підімкнемо від'ємний вивід світлодіода до верхнього ряду контактів на макетній платі, а додатний вивід світлодіода з'єднаємо з резистором. Верхній ряд контактів візьмемо за землю і до нього ж підімкнемо кнопку і контакт ***GND*** на ***GPIO***. Інші виводи резисторів і другий вивід кнопки підімкнемо до двох різних пінів ***GPIO*** для видачі керувального сигналу. Необхідно також з'єднати кнопку з піном ***GPIO*** +5V. Після підімкнення всіх елементів робоче поле набуде вигляду, як показано на рис. В.12.

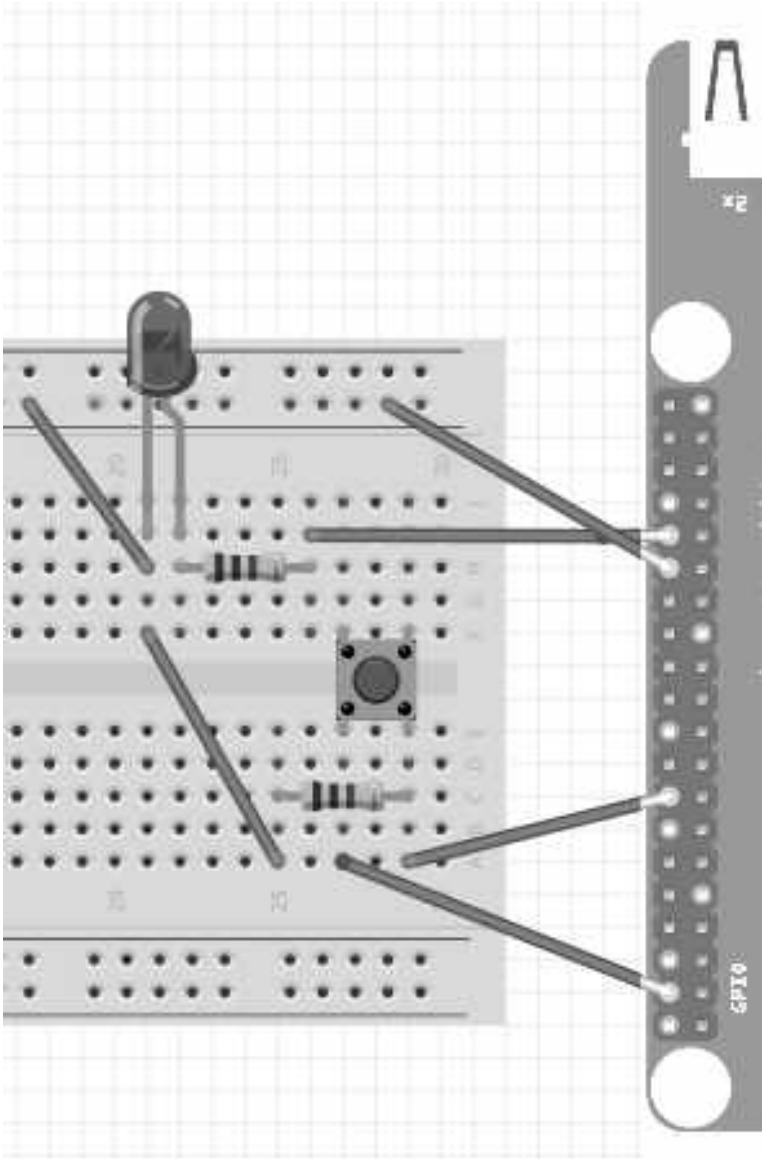


Рис. В.12

Щоб схема була зручною для читання, позбавимося зайвих сполучних ліній, розмістимо елементи так, щоб проводи не перехрещувались. Після проведених дій схема може мати вигляд, як на рис. В.13.

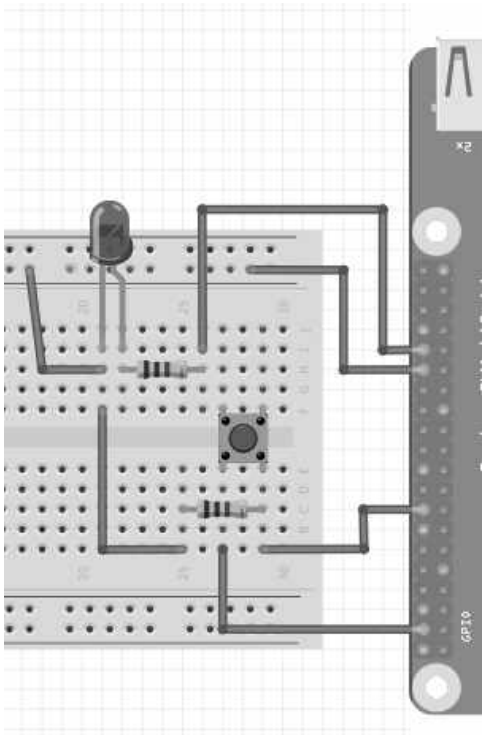


Рис. В.13

Заключним етапом буде збереження проекту або експорт зображення в зручному форматі.

Формування електричної принципової схеми

На основі вже створеної схеми з'єднання побудуємо принципову схему. Для цього необхідно переключитися на відповідну вкладку у відкритому проекті. ***Fritzing*** уже зробив всі необхідні з'єднання, слід лише звести це все до легкого для читання вигляду. Виберемо в меню ***Fritzing*** **Вид → Подогнать окно**, для центрування і масштабування схеми автоматично на робочому полі (рис. В.14).

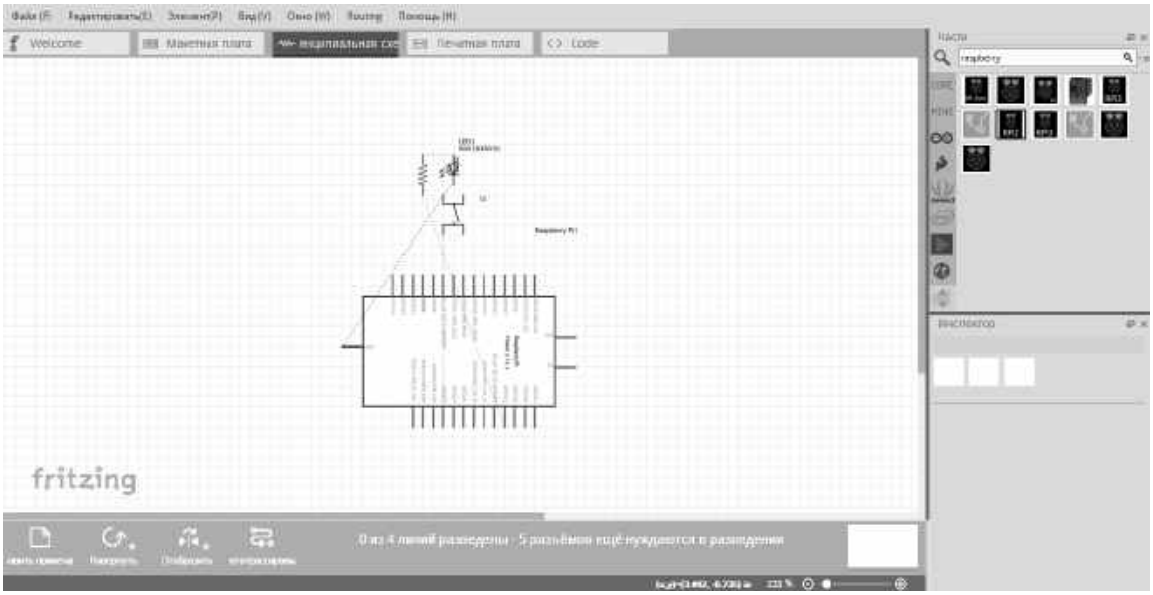


Рис. В.14

Перетягуючи й повертаючи (Ctrl + r) компоненти, необхідно добитися того, щоб провідники не перетиналися або кількість перетинів була мінімальною (рис. В.15).

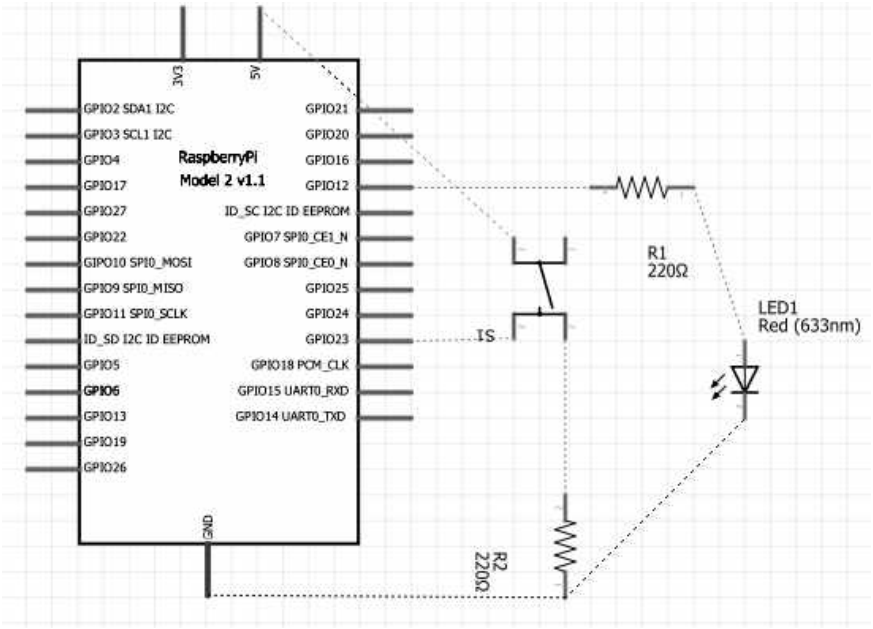


Рис. В.15

Потім у нижньому лівому куті екрану ***Fritzing*** вибираємо ***Автотрассировка*** (рис. В.16) або ж самостійно розводимо лінії з'єднання. Тут також можна побачити, скільки рознімів необхідно підімкнути.

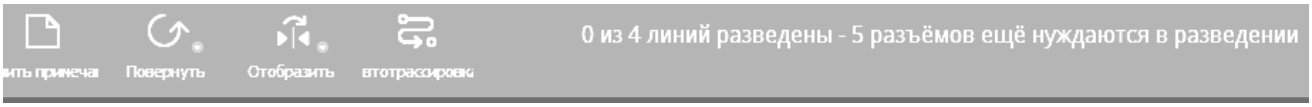


Рис. В.16

Після цього принципова схема набуде нормального вигляду (рис. В.17).

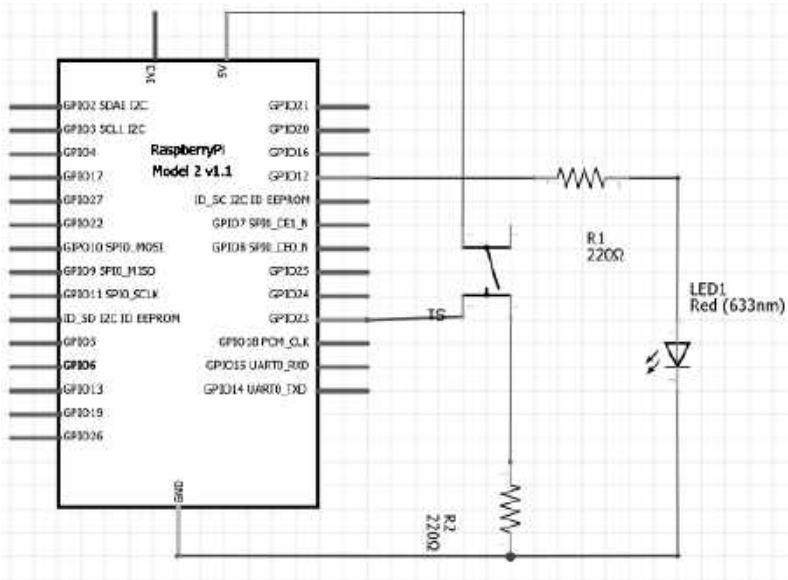


Рис. В.17

За необхідності можна перейти на вкладку «***Печатная плата***», де також автоматично додано елементи і зв'язки між ними. Після редагування проекту є можливість експортувати готовий макет і використовувати його для виготовлення реальної друкованої плати.

Керування GPIO при використанні мови Python

Для роботи з ***GPIO*** мовою ***Python*** потрібна спеціальна бібліотека ***Rpi.GPIO***. У новому дистрибутиві ***Raspbian*** її вже встановлено, а якщо дистрибутив застарів, то для установлення бібліотеки ***Rpi.GPIO*** слід виконати команду

```
sudo apt-get install python-rpi.gpio
```

Для користування цією бібліотекою необхідно в програму мовою ***Python*** додати рядок імпорту бібліотеки `RPi.GPIO`:

```
Import RPi.GPIO as GPIO
```

Під час підготовки програми можна вибрати один із двох способів нумерації портів ***GPIO***. Перший – `GPIO.BOARD` – використовує систему нумерації портів на платі ***Raspberry Pi***. Перевага цієї системи нумерації полягає в тому, що устаткування працюватиме завжди, незалежно від номера ревізії плати, тобто перемонтувати свій рознім або змінити наявний код не доведеться. Друга система нумерації – `GPIO.BCM` (номери `BCM`). Це нижчий рівень роботи – з прямим зверненням до номерів каналів на процесорі (`SOC`) `Broadcom`. Вибір способу нумерації визначається відповідними командами:

```
GPIO.setmode(GPIO.BOARD)
GPIO.setmode(GPIO.BCM)
```

Режим роботи контакту на вхід або вихід установлюють такі команди:

```
GPIO.setup(channel, GPIO.IN)
GPIO.setup(channel, GPIO.OUT)
```

Якщо вхідний канал ні до чого не підключено, то його значення може «плавати». Початкове «підтягування» виводу до живлення або «землі» установлюють команди

```
GPIO.setup(channel, GPIO.IN, GPIO.PUD_UP)
GPIO.setup(channel, GPIO.IN, GPIO.PUD_DOWN)
```

Для виходів `OUT` можна встановити початкове значення `0` або `1`:

```
GPIO.setup(channel, GPIO.OUT, GPIO.LOW)
GPIO.setup(channel, GPIO.OUT, GPIO.HIGH)
```

Для читання значення контакту ***GPIO***, настроєного як вхід `IN`, призначено таку команду:

```
GPIO.input(channel)
```

Значення контакту, настроєного як вихід OUT, установлює команда

```
GPIO.output(channel, state)
```

Наведемо приклад використання команд роботи з ***GPIO***:

```
import RPi.GPIO as GPIO       # підключаємо бібліотеку
GPIO.setmode(GPIO.BCM)  # встановлюємо режим нумерації
GPIO.setup(7, GPIO.OUT) # конфігуруємо GPIO 7 як вихід
GPIO.setup(8, GPIO.IN)  # конфігуруємо GPIO 8 як вхід
GPIO.output(7, True)    # виводимо на GPIO 7 логічну "1"
(3.3 V)
GPIO.output(7, False)   # виводимо на GPIO 7 логічний "0"
signal = GPIO.input(8)  # прочитуємо з GPIO 8 у змінну
signal
GPIO.cleanup()          # завершуємо роботу з GPIO
```

Бібліотека Rpi.GPIO дає змогу використовувати контакти ***GPIO*** як виходи ШІМ (сигналів широко-імпульсної модуляції). Стисло опишемо основні команди для формування керувальних дій на базі ШІМ, а потім зупинимося на цьому детальніше. Для створення екземпляра ШІМ призначено команду

```
p = GPIO.PWM(channel, frequency)
```

де `frequency` – частота, Гц;

для старту ШІМ на контакті –

```
p.start(dc)
```

де dc – робочий цикл ШІМ (0,0…100,0);

для змінення частоти сигналу –

```
p.ChangeFrequency(freq)
```

де `freq` – частота сигналу, Гц;

для змінення робочого циклу ШІМ –

```
p.ChangeDutyCycle(dc) # where 0.0 <= dc <= 100.0
```

для зупинення видачі сигналу ШІМ на контакті –

```
p.stop()
```

Наведемо приклад плавного вмикання/вимикання світлодіода, підімкненого до контакту:

```
import time

import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
```

```
GPIO.setup(12, GPIO.OUT)

p = GPIO.PWM(12, 50) # контакт 12 частота 50 Гц
p.start(0)
try:
    while 1:
        for dc in range(0, 101, 5):
            p.ChangeDutyCycle(dc)
            time.sleep(0.1)
        for dc in range(100, -1, -5):
            p.ChangeDutyCycle(dc)
             time.sleep(0.1)
   except KeyboardInterrupt:
    pass
p.stop()
GPIO.cleanup()
```

Функція очікування події – змінення стану на вході IN – має такий вигляд:

```
GPIO.wait_for_edge(channel, GPIO.RISING)
GPIO.wait_for_edge(channel, GPIO.FALLING)
GPIO.wait_for_edge(channel, GPIO.BOTH)
```

Ця функція перериває виконання програми до змінення стану на вході ***GPIO***. На відміну від неї функція add_event_detected() виконується в циклі програми, і щоб дізнатися про змінення стану на контакті ***GPIO***, необхідно в циклі перевіряти настання події event_detected():

```
GPIO.add_event_detect(channel, GPIO.RISING
.................
if GPIO.event_detected(channel):
    print('Button pressed')
```

Кожен контакт ***GPIO*** можна настроїти на роботу в режимі переривання, у цьому випадку при настанні події на контакті керування передається функція оброблення переривання. Ця функція працює в окремому потоці, не перериваючи виконання основної програми:

```
def my_callback(channel):
    print('обробка переривання!')
GPIO.add_event_detect(channel, GPIO.RISING,
callback=my_callback)
```

Після завершення будь-якої програми рекомендується очистити всі ресурси, які використовувалися. Для такого очищення в кінці скрипту необхідно передбачити команду

```
GPIO.cleanup()
```

Для скидання одного контакту передбачено команду

`GPIO.cleanup(channel)`.

Склад і структура універсального лабораторного стенду

Для проведення циклу лабораторних робіт з тематики, пов'язаної з дослідженням характеристик і властивостей систем технічного зору, було створено універсальний лабораторний макет, який можна модифікувати й додатково укомплектувати для проведення конкретної роботи.

Зовнішній вигляд базового комплекту лабораторного стенду показано на рис. В.18. Центральним ядром системи є одноплатний комп'ютер ***Raspberry Pi***. З метою дотримання вимог техніки безпеки його поміщено в стандартний пластиковий корпус з вирізами для комутаційних рознімів. Лабораторний стенд в базовій комплектації складається з таких елементів:

- мікрокомп'ютер ***Raspberry Pi*** модель 3;
- пластиковий корпус для ***Raberry Pi 2***;
- SD-карта для завантаження ОС 32 Gb;
- монітор і кабель з рознімами HDMI;
- USB-клавіатура;
- USB-мишка;
- кабель живлення або акумулятор micro-USB;
- безпайна макетна плата з комплектом з’єднувальних проводів.

Рис. В.18

Лабораторна робота № 1

РОБОТА З ІНТЕРФЕЙСОМ GPIO КОМП'ЮТЕРА RASPBERRY PI

Мета роботи – вивчення принципів роботи і використання апаратних портів одноплатного комп'ютера ***Raspberry Pi*** для вирішення різних завдань у системах технічного зору й робототехніці. Набуття практичних навичок роботи в цій області шляхом проведення експериментальних досліджень з керування станом світлодіода з допомогою апаратних портів ***GPIO*** одноплатного комп'ютера ***Raspberry Pi***, що є характерним для формування разових команд у системах.

Схема керування роботою світлодіода

Для проведення експерименту необхідно мати такі комплектувальні елементи:

- світлодіод;
- кнопка;
- резистор на 220 Ом;
- три з’єднувальні проводи «мама-папа»;
- два з’єднувальні проводи «папа-папа».

Схему можна досить просто скласти із застосуванням як навісного монтажу, так і безпайної макетної плати (рис. 1.1).

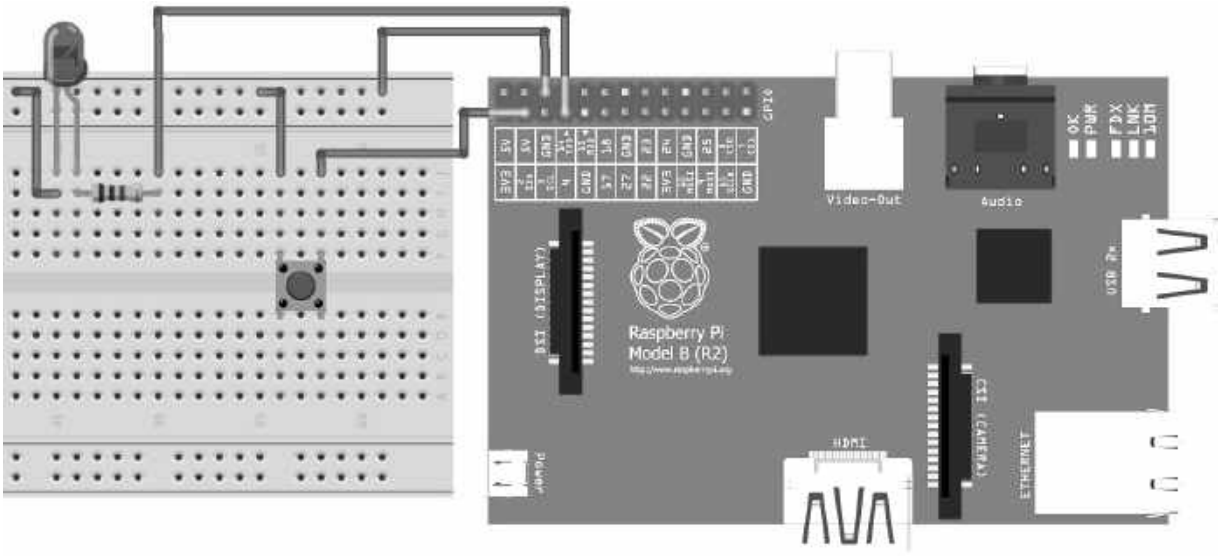


Рис. 1.1

Керування роботою світлодіода на Raspberry Pi

Для вирішення цього завдання можливими є такі варіанти роботи:

- керування світлодіодом з консолі;

- автономна робота світлодіода за заданою програмою;
- керування світлодіодом з допомогою кнопки;
- керування світлодіодом з клавіатури.

Розглянемо ці приклади детальніше.

Керування світлодіодом на Raspberry Pi з консолі. У цьому випадку заходимо в `Lxterminal` і набираємо команду

```
sudo python
```

Після цього замість імені користувача на початку рядка має відобразитися

```
>>>
```

Далі необхідно набрати такі рядки:

```
>>> import RPi.GPIO as GPIO  # імпорт бібліотеки
>>> GPIO.setmode(GPIO.BOARD) # "увімкнення" GPIO
>>> GPIO.setup(7, GPIO.OUT)  # оголошення 7-го піна як ви-
    ходу
```

Потім для ввімкнення світлодіода можна використати команду

```
>>> GPIO.output(7, 1),
```

а для вимкнення – команду

```
>>> GPIO(output(7, 0)
```

Після роботи з ***GPIO*** бажано обов'язково виконати команду

```
>>> GPIO.cleanup()
```

Цей приклад відображає принципову можливість керування роботою схеми з допомогою портів ***GPIO***, однак є багато незручностей для користувача.

Автономна робота світлодіода за заданою програмою. Для автономної роботи світлодіода необхідно написати й запустити програму. Для цього відкриємо встановлену наперед програму `IDLE 3` і в меню `File` натиснемо `New`. У вікні, що відкриється, можна писати програму. Напишемо такий код:

```
import RPi.GPIO as GPIO      # імпорт бібліотеки для роботи
                             з GPIO
import time                  # імпорт бібліотеки для очіку-
                             вання команди
GPIO.setmode(GPIO.BOARD)    # "запуск" GPIO
GPIO.setup(7, GPIO.OUT)     # оголошення 7-го порту як ви-
                              ходу
while True:                # нескінчений цикл
____GPIO.output(7, 1)      # увімкнення світлодіода
____time.sleep(1)          # очікування – 1 секунда
```

```
____GPIO.output(7,0)        # вимкнення світлодіода
____time.sleep(1)           # очікування – 1 секунда
```

Програму необхідно зберегти в папці /home/pi. Запустити програму з Lxterminal можна з допомогою команди

```
sudo python program_name.py
```

Нагадаємо, що в програмі необхідно дотримуватися відступів (кожен по чотири пропуски, або ж одній табуляції, наприклад: після "else" рядок починається з трьох табуляцій, або 12 пропусків, а перед "while True" – з однієї табуляції, або чотирьох пропусків). Це є важливим і необхідним для правильної роботи програми мовою ***Python***.

Керування світлодіодом з допомогою зовнішньої кнопки. У схемі керування передбачено два випадки:

1 – коли кнопка натиснута, світлодіод горить;

2 – коли кнопка відтиснута, світлодіод не горить.

Для реалізації заданого сценарію підімкнемо кнопку до порту 5. Для керування необхідна така програма:

```
import RPi.GPIO as GPIO          # імпорт бібліотеки GPIO
GPIO.setmode(GPIO.BOARD)         # "включення GPIO"
GPIO.setup(7, GPIO.OUT)          # оголошення 7-го порту як
                                   виходу
GPIO.setup(3, GPIO.IN)           # оголошення 3-го порту як
                                   входу
while True:                      # нескінченний цикл
____if GPIO.input(3) == False: # якщо кнопка натиснута, то
________GPIO.output(7, 1)        # вмикаємо світлодіод,
____else:                        # інакше –
________GPIO.output(7, 0)        # вимикаємо
```

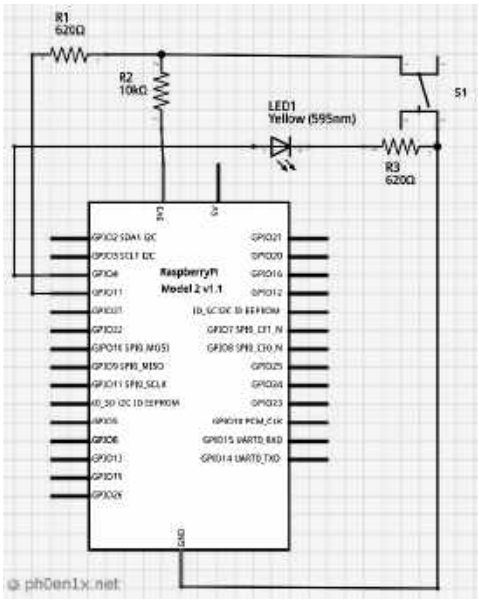
Керування світлодіодом з клавіатури. Напишемо ще один програмний код, який зминюватиме стан світлодіода при виникненні порожнього рядка і закінчуватимется при виникненні іншого рядка:

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setup(7, GPIO.OUT)
while True:
____str = input("Enter - увікмення, інше - вихід");
____if str != "":
________break
____else:
________GPIO.output(7, 1)
____str = input("Enter - вимкнення, інше - вихід");
____if str != "":
________break
```

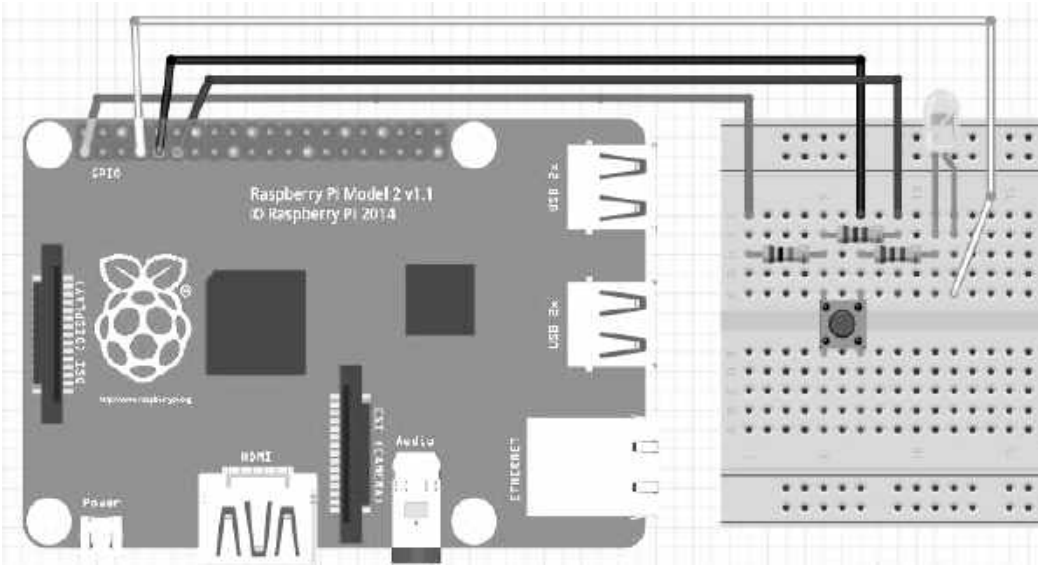
```
____else:
________GPIO.output(7, 0)
GPIO.cleanup()
```

Ускладнений варіант керування світлодіодом з допомогою кнопки. Реалізуємо програму, за якою світлодіод буде мерехтіти з частотою два рази за секунду (2 Гц), а при натисканні кнопки частота мерехтіння збільшуватиметься приблизно в п'ять разів (10 Гц). Такий приклад є цікавішим, ніж просте натискання кнопки для засвічування світлодіода.

Електричну й монтажну схеми підімкнення діода й кнопки зображено на рис. 1.2.



а



б

Рис. 1.2

Створимо на ***Python*** програмний код для цього варіанта керування свіченням діода:

```
# Підмикаємо необхідні бібліотеки (для затримки за часом і GPIO)
import time
import RPi.GPIO as GPIO
# Установлюємо номери пінів GPIO, з якими працюватимемо
LED = 4
KEY = 17
# Робимо скидання станів портів (усі порти конфігуруються на вхід - INPUT)
GPIO.cleanup()
# Режим нумерації пінів - за назвою (не за порядковим номером на рознімі)
GPIO.setmode(GPIO.BCM)
# Конфігуруємо пін LED на вивід (OUTPUT)
GPIO.setup(LED, GPIO.OUT)
# Установлюємо низький рівень (0) на піні LEDT)
GPIO.output(LED, GPIO.LOW)
# Конфігуруємо пін KEY на введення (INPUT)
GPIO.setup(KEY, GPIO.IN)
# Виводимо на екран текст-привітання
print 'Hello! Blink...blink...'
#  Перевіряємо на переривання програми з клавіатури (Ctrl+c)
try:
    # Вічний цикл
    while True:
        # Якщо кнопку натиснуто (на піні KEY низький рівень напруги – 0 V), то
        if GPIO.input(KEY) == False:
            # Установлюємо затримку 0,1 с і виводимо повідомлення:
            timeout = 0.1
            print 'Key pressed.'
        else:
            # інакше – затримка 0,5 с
            timeout = 0.5
        # Засвітимо світлодіод, підімкнутий до піна LED
        GPIO.output(LED, GPIO.HIGH)
           # Зачекаємо (виконаємо задану вище затримку)
        time.sleep(timeout)
```

```
        # Погасимо світлодіод, підімкнутий до піна LED
        GPIO.output(LED, GPIO.LOW)
        time.sleep(timeout)
# При натисканні комбінації клавіш Ctrl+c - скидання пінів і завершення роботи
except KeyboardInterrupt:
    GPIO.cleanup()
```

Завдання для самостійної роботи

Виконати завдання з керування роботою світлодіода з використанням портів ***GPIO*** комп'ютера ***Raspberry Pi***.

Для цього необхідно:

- скласти схему керування світлодіодом відповідно до рекомендацій;
- використовуючи всі варіанти програмних кодів мовою ***Python,*** провести експерименти з керування роботою світлодіода.

Звіт про виконання лабораторної роботи повинен містити:

- схеми керування електричні принципові й схеми монтажні;
- тексти програмних кодів;
- висновки за результатами проведених досліджень.

Контрольні запитання

1. Для чого призначено апаратні порти введення/виведення ***GPIO*** в комп'ютері ***Raspberry Pi***?
2. Який конструктивний вигляд має рознім ***GPIO*** на платі комп'ютера? Якою є його цоколівка (розпіновка) ?
3. Якіх запобіжних заходів необхідно дотримуватися під час роботи з інтерфейсом ***GPIO***?
4. Якою є здатність навантаження входів/виходів портів ***GPIO*** і як їх захистити від перевантаження й виходу з ладу?
5. Як для роботи з ***GPIO*** мовою ***Python*** установити бібліотеки ***Rpi.GPIO***? Опишіть основні команди настроєння керувальних входів (`INPUT`) і виходів (`OUTPUT`) мовою ***Python***.
6. Які команди для портів введення/виведення ***GPIO*** використовуються під час формування керувальної програми з використанням широтно-імпульсної модуляції (***ШІМ***)?
7. Як керувати роботою одночасно двох або більшої кількості світлодіодів?
8. Як збільшення кількості керувальних команд може вплинути на безпеку роботи портів ***GPIO*** комп'ютера ***Raspberry Pi***?

Лабораторна робота № 2

КЕРУВАННЯ СЕРВОПРИВОДАМИ НА RASPBERRY PI

Мета роботи – вивчення й використання апаратних портів одноплатного комп'ютера ***Raspberry Pi*** для вирішення завдань керування сервоприводами в системах технічного зору й робототехніці. Набуття практичних навичок роботи в цій області.

Стислі теоретичні відомості

У робототехніці й системах технічного зору, що вирішуються з допомогою комп'ютера ***Raspberry Pi***, одним із практично корисних завдань є будування широтно-імпульсних модуляторів (***ШІМ***) і керування з їх допомогою сервоприводами. Це завдання можна вирішити кількома способами – програмно і з використанням апаратних ресурсів комп'ютера ***Raspberry Pi***. Розглянемо його детальніше.

Широтно-імпульсна модуляція (ШІМ) (англ. pulse-width modulation – PWM) – це імпульсний сигнал постійної частоти і змінної шпаруватості, тобто відношення тривалості імпульсу до періоду його дотримання. З допомогою завдання шпаруватості можна змінювати середнє значення напруги на виході модулятора ***ШІМ***.

На рис. 2.1 показано епюри напруги на виході модулятора ***ШІМ*** при різних значеннях глибини модуляції.

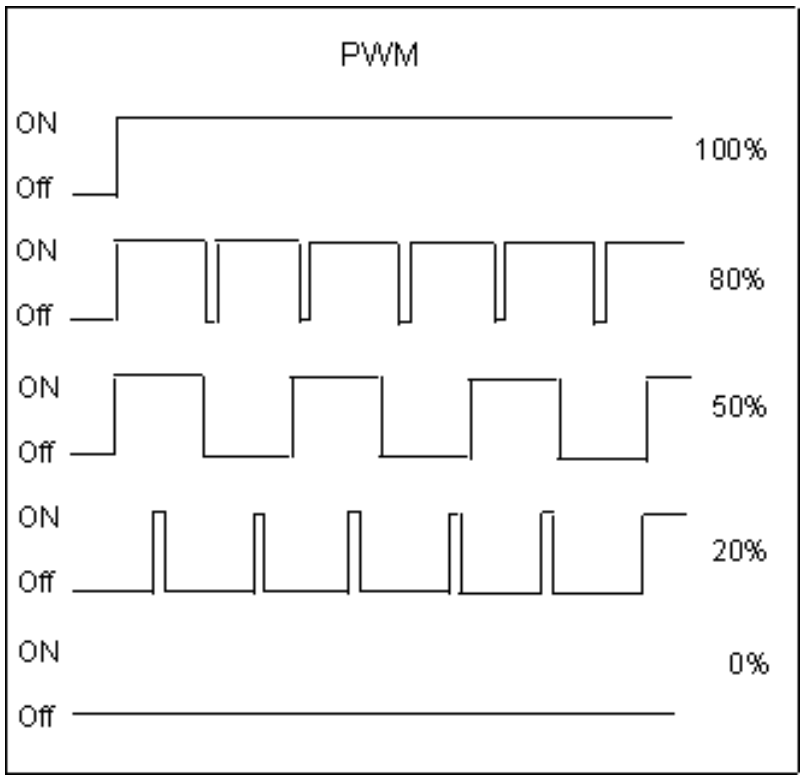


Рис. 2.1

Програмна реалізація ШІМ

Поставимо завдання плавного змінення яскравості свічення світлодіода. Для цього підімкнемо світлодіод до порту GPIO23 через обмежувальний для струму резистор, як показано на схемі (рис. 2.2).

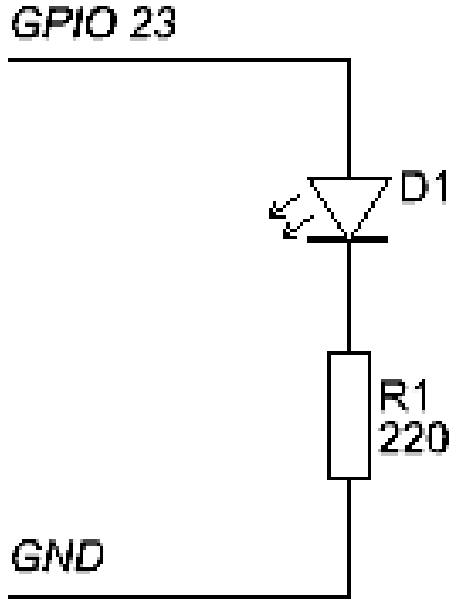


Рис. 2.2

Нагадаємо, що бібліотека Rpi.GPIO дає змогу використовувати контакти ***GPIO*** як виходи ***ШІМ*** (сигналів широтно-імпульсної модуляції).

Для створення екземпляра ***ШІМ*** призначено команду

```
p = GPIO.PWM(channel, frequency)
```

де frequency – частота, Гц.
Для старту ***ШІМ*** на контакті використовується команда

```
p.start(dc)
```

де dc – робочий цикл ***ШІМ*** (0,0…100,0).
для змінення частоти сигналу –

```
p.ChangeFrequency(freq)
```

де freq – частота сигналу, Гц;
для змінення робочого циклу ***ШІМ*** –

```
p.ChangeDutyCycle(dc) # where 0.0 <= dc <= 100.0
```

для припинення видачі сигналу ***ШІМ*** на контакті –

```
p.stop()
```

Напишемо програмний код для вирішення поставленого завдання і присвоїмо йому ім'я **pwm_soft.py**:

```
import time
import RPi.GPIO as GPIO
```

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(23, GPIO.OUT)
p = GPIO.PWM(23, 50)          # channel=23 frequency=50Hz
p.start(0)
try:
    while 1:
        for dc in range(0, 101, 5):
            p.ChangeDutyCycle(dc)
            time.sleep(0.1)
        for dc in range(100, -1, -5):
            p.ChangeDutyCycle(dc)
            time.sleep(0.1)
except KeyboardInterrupt:
    pass
p.stop()
GPIO.cleanup()
```

Після запуску цієї програми світлодіод плавно засвітиться, а потім так само плавно погасне.

Програмна реалізація ***ШІМ*** дає змогу сформувати сигнал ***ШІМ*** на будь-якому виводі. У цьому прикладі було використано `Rpi.GPIO` для програмної генерації ***ШІМ***-сигналу. Це означає, що витрачаються обчислювальні ресурси мікрокомп'ютера. Якщо мікрокомп'ютер буде виконувати одночасно й інші завдання, то сигнал ***ШІМ*** спотвориться і не буде стабільним. Це не принципово, якщо ***ШІМ*** застосовується для керування яскравістю світлодіода. Однак це неприйнятно при формуванні керувального сигналу. Наприклад, під час керування сервоприводами програмна реалізація ***ШІМ*** не може стабільно утримувати їх у заданому положенні. Проте цей недолік можна легко подолати, оскільки ***Raspberry Pi*** дає технічну можливість використовувати для генерації ***ШІМ*** апаратний ресурс.

Генерація ШІМ-сигналу з використанням апаратних ресурсів

Проект `Wiringpi` – це бібліотека, яка містить утиліти для простого доступу до ***GPIO*** і дає змогу настроїти апаратні модулі для спеціальних виходів ***ШІМ***. Установимо `wiringpi`:

```
sudo apt-get install git-core
git clone git://git.drogon.net/wiringPi
cd wiringPi
./build
cd ..
```

Як і в попередньому випадку, підімкнемо світлодіод до порту `GPIO18` (рис. 2.3).

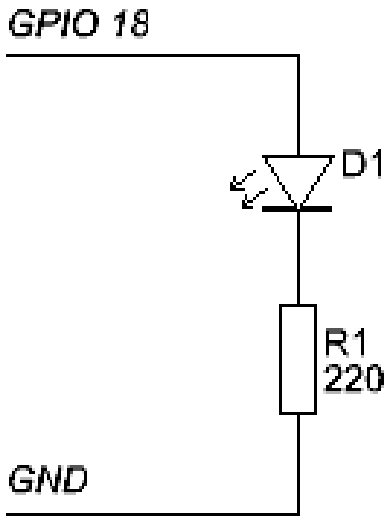


Рис. 2.3

Перший вихід ***ШІМ*** виведено на GPIO18. Виконаємо команди для формування на GPIO18 сигналу ***ШІМ***. Настроїмо перший канал PWM(GPIO18):

```
gpio mode 1 pwm
```

Задамо шпаруватість від 0 до 1024:

```
gpio pwm 1 500
```

Світлодіод має світитися напіврозжарено. Для експерименту з ***ШІМ*** можна задати такі значення:

```
gpio pwm 1 10
gpio pwm 1 1023
```

Вимкнемо ***ШІМ***:

```
gpio unexport 1
```

або

```
gpio unexportall
```

Генерація апаратного сигналу ШІМ на Python

Щоб використовувати ***ШІМ*** у ***Python***, установимо ***Wiringpi-python***:

```
sudo apt-get install python-dev python-setuptools
git clone https://github.com/WiringPi/WiringPi-Python
cd WiringPi-Python
git submodule update --init
python setup.py install
cd ..
```

Після цього напишемо програмний код для вирішення поставленого завдання і присвоїмо йому ім'я **pwm.py**:

```
import time
import wiringpi
# GPIO pin 12 = BCM pin 18 = wiringpi pin 1
led_pin = 1
wiringpi.wiringPiSetup()
wiringpi.pinMode(led_pin, 2)
wiringpi.pwmWrite(led_pin, 0)
def led(led_value):
wiringpi.pwmWrite(led_pin, led_value)
led(0)
while 1:
        for dc in range(0, 1023, 5):
                led(dc)
                time.sleep(0.01)
        for dc in range(1024, 0, -5):
                led(dc)
                time.sleep(0.01)
```

Написаний програмний код запустимо з допомогою командного рядка, використовуючи команду `sudo`, яка дає змогу виконувати скрипт з правами адміністратора. Це є необхідним для виконання команди `wiringpi.wiringPiSetup()`.

Після запуску цієї програми світлодіод плавно засвітиться, а потим так само плавно погасне. Апаратна реалізація ***ШІМ*** забезпечує стабільніший результат. На жаль, апаратний вихід у ***Raspberry Pi*** лише один. Тому у випадках, коли необхідно керувати декількома пристроями, наприклад сервоприводами, зазвичай використовують зовнішній контролер ***PWM*** (рис. 2.4).

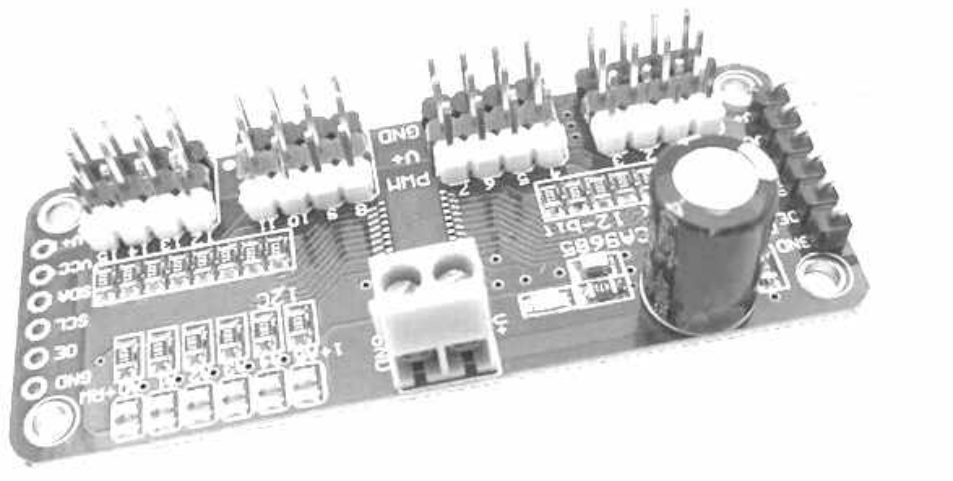
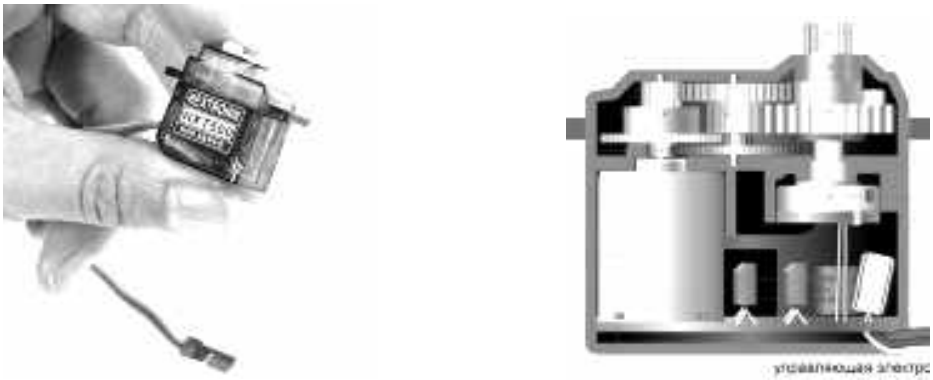


Рис. 2.4

Керування сервоприводом

Загальні відомості про сервоприводи. У більшости сервоприводів для роботи використовують три проводи: провід для живлення, зазвичай 4.8 або 6 V, загальний (земля) і сигнальний. Керувальний сигнал передає інформацію про необхідне положення вихідного вала. Вал пов'язаний з потенціометром, який визначає його положення. Контролер у сервоприводі за опором потенціометра й значеннням керувального сигналу визначає, у який бік слід обертати мотор, щоб отримати необхідне положення вихідного вала. Чим вище напруга живлення сервоприводу, тим швидше він працює і більший момент розвиває. Зовнішній вигляд і кінематичну схему сервоприводу показано на рис. 2.5.



а *б*

Рис. 2.5

Характеристики сервоприводів

Розмір і маса. Існують такі розміри: мікро, міні, стандарт і гігант. У межах кожного класу розміри можуть трохи різнитися. Середні розміри сервоприводів для довідки:

- мікро: 24 × 12 × 24 мм, маса 5…10 г;
- міні: 30 × 15 × 35 мм, маса 25 г;
- стандарт: 40 × 20 × 37 мм, маса: 50…60 г.

Швидкість. Швидкість сервоприводу залежить від часу повороту його качалки на кут 60^0 при напругах живлення 4,8 і 6 V. Наприклад, сервопривід з параметром 0,22 с/60° при напрузі живлення 4,8 V повертає вал на 60° за 0,22 с. Це не так швидко, як може здатися. Час переміщення найбільш швидких сервоприводів становить від 0,06 до 0,09 с.

Кут повороту вала сервоприводів може бути 60^0, 90^0 або 180^0. Кут повороту обмежується електронікою і механічно. Існують сервоприводи без обмеження, тобто такі, що обертаються на 360^0. Якщо є сервопривід з робочим діапазоном 60^0, то розширити його можна, лише змінивши конструкцію сервомашинки. Інколи можна збільшити діапазон спеціально спотворивши керувальний сигнал. Однак це є нестандартним і ненадійним способом.

Момент на валу сервоприводу вимірюється за масою вантажу в кілограмах, яку сервопривід може утримувати нерухомо на качалці з плечем 1 см. Указують дві цифри для напруг живлення 4,8 і 6 V. Наприклад, якщо вказано, що сервопривід розвиває зусилля 10 кг/см, то це означає, що на качалці завдовжки 1 см сервопривід може розвинути зусилля 10 кг, перш ніж зупиниться. Для качалки завдовжки 2 см такий сервопривід зможе розвинути зусилля 5 кг, а завдовжки 5 см – 20 кг.

Цифрові й аналогові сервоприводи механічно не відрізняються один від одного. У них ті самі корпуси, мотори, шестерінки й навіть потенціометри. Різниця полягає в способі керування мотором. Цифрові сервоприводи є точнішими і їх час реакції завжди є меншим. Однак вони споживають більше енергії, ніж аналогові сервоприводи. Керувальний сигнал для аналогових і цифрових сервоприводів є однаковим.

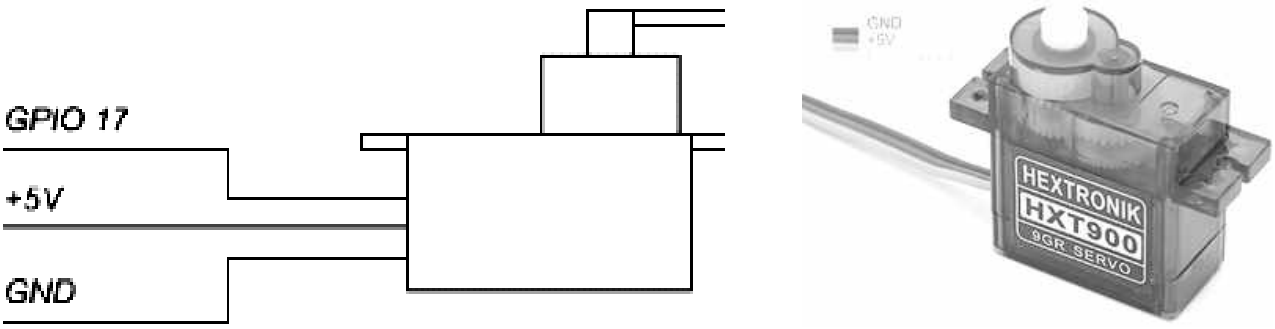
Керувальний сигнал є імпульсами змінної ширини (рис. 2.6). Імпульси повторюються з постійною частотою (зазвичай 50 Гц). Положення сервоприводу визначається шириною імпульсу. Для типового сервоприводу, що використовується в керованих по радіо моделях, тривалість імпульсу 1500 мкс означає, що сервопривід має зайняти середнє положення. Збільшення або зменшення довжини імпульсу змусить сервопривід обернутися за ходом або проти ходу годинникової стрілки відповідно.

Таким чином, для керування сервоприводом необхідно формувати ***ШІМ*** з частотою 50 Гц. При цьому для положення «0» тривалість імпульсу має становити 1 мс, а для положення «максимум» – 2 мс, середнє положення – 1,5 мс.

Рис. 2.6

Зазвичай напруга живлення сервоприводів становить 5 V. Малопотужний сервопривід можна живити від ***Raspberry Pi***. Однак якщо привід споживає великий струм, або необхідно підімкнути кілька сервомашинок, то краще не навантажувати ***Raspberry Pi*** і використовувати окреме джерело живлення.

Схема підімкнення сервоприводу до порту `GPIO 17` і зовнішній вигляд сервоприводу HEXTRONIC Hxt900 показано на рис. 2.7. Зверніть увагу на колірну схему проводів для підімкнення сервоприводу. Колірний код зменшує ймовірність неправильного підімкнення.



а *б*

Рис. 2.7

Керування сервоприводом з допомогою програмно сформованої ШІМ

Спочатку сформуємо ***ШІМ*** для керування сервоприводом з допомогою програми. Для цього створимо програмний код з ім'ям **`servo.py`**:

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setup(17,GPIO.OUT)
p=GPIO.PWM(17,50)
p.start(7.5)
try:
        while True:
                p.ChangeDutyCycle(7.5)
                print "Left"
                time.sleep(1)
                p.ChangeDutyCycle(12.5)
                print "Center"
                time.sleep(1)
                p.ChangeDutyCycle(2.5)
                print "Right"
                time.sleep(1)
```

```
except KeyboardInterrupt:
        p.stop()
        GPIO.cleanup()
```

Функції `print` у цьому коді свідомо зроблено надлишковими. Наявність цих функцій усуває описану раніше проблему нестабільності програмно сформованої ***ШІМ***. Сервомашинка при цьому може не фіксуватися в заданому положенні й сіпатися. Якщо видалити інструкції `print`, то проблема зменшиться або взагалі зникне.

Керування сервоприводом з допомогою ШІМ, сформованої через DMA

DMA (від англ. direct memory access) – прямий доступ до пам'яті.

Установлюємо `RPIO`:

```
apt-get install python-setuptools
easy_install -U RPIO
```

Створюємо сріпт **`servo_dma.py`**:

```
import time
from RPIO import PWM
servo = PWM.Servo()
# Set servo on GPIO17 to 900.s (0.9ms)
servo.set_servo(17, 900)
# Set servo on GPIO17 to 2000.s (2.0ms)
#servo.set_servo(17, 2000)
try:
        while True:
                servo.set_servo(17, 750)
                print "Left"
                time.sleep(1)
                servo.set_servo(17, 1500)
                print "Center"
                time.sleep(1)
                print "Right"
                servo.set_servo(17, 2500)
                time.sleep(1)
except KeyboardInterrupt:
        # Clear servo on GPIO17
        servo.stop_servo(17)
```

Тепер сервопривід працює стабільно. Це є найбільш простим і надійним способом для керування сервоприводами.

Завдання для самостійної роботи

Виконати завдання з керування роботою сервоприводів з використанням портів ***GPIO*** комп'ютера ***Raspberry Pi***. Для цього необхідно:

- зібрати схему керування яскравістю світлодіода відповідно до рекомендацій і перевірити роботу схеми для різних варіантів формування ***ШІМ*** (програмної й апаратної);
- підімкнути до шини ***GPIO*** комп'ютера ***Raspberry Pi*** сервопривід згідно з рис. 2.6 і експериментально перевірити якість роботи при різних способах керування (програмно сформованої ***ШІМ*** і ***ШІМ***, сформованої через ***DMA***).

Звіт про виконання лабораторної роботи повинен містити:

- схеми керування електричні принципові й схеми монтажні;
- тексти програмних кодів, що використовувались у цій роботі;
- виводи за результатами проведених досліджень.

Контрольні запитання

1. Для чого призначено апаратні порти введення/виведення ***GPIO*** в комп'ютері ***Raspberry Pi***?
2. Якій конструктивний вигляд має рознім ***GPIO*** на платі комп'ютера? Якою є його цоколівка (розпіновка)?
3. Якою є здатність навантаження входів/виходів портів ***GPIO?*** Як їх захистити від перевантаження й виходу з ладу?
4. Поясніть принцип будування сигналів широтно-імпульсної модуляції (***ШІМ***).
5. Як здійснюється програмна реалізація ***ШІМ***?
6. Як здійснюється генерація ***ШІМ***-сигналу з використанням апаратних ресурсів комп'ютера ***Raspberry Pi***?
7. Наведіть приклад керування яскравістю свічення діода з допомогою модулятора ***ШІМ***.
8. Як здійснити генерацію апаратного сигналу ***ШІМ*** на ***Python***?
9. Якими є принципи роботи сервоприводів у системах керування? Наведіть основні характеристики сервоприводів.
10. Опишіть основні команди настроєння входів (INPUT) і виходів (0UTPUT) при написанні програми мовою ***Python***.
11. Які команди для портів введення/виведення ***GPIO*** використовуються при формуванні керувальної програми з використанням широтно-імпульсної модуляції (***ШІМ***)?
12. Поясніть значення змінних у командах p.ChangeDutyCycle; wiringpi.pwmSetClock; wiringpi.pwmSetRange при керуванні сервоприводом.

Лабораторна робота № 3

КЕРУВАННЯ ДЕКІЛЬКОМА СЕРВОПРИВОДАМИ З ДОПОМОГОЮ ШІМ-КОНТРОЛЕРА RASPBERRY PI

Мета роботи − вивчити й уміти використовувати апаратні засоби одноплатного комп'ютера ***Raspberry Pi*** для вирішення завдань керування кількома сервоприводами. Набуття практичних навичок роботи з ***ШІМ***-контролером і створення інтерфейсу програми.

Стислі теоретичні відомості

Поворот коліс робота або відхилення керувальних поверхонь коптера потребують окремого сервоприводу й відповідно можливості незалежного керування. Тому створення складних рухомих керованих об'єктів завжди супроводжується установленням кількох сервоприводів на борту. Керувати такою кількістю об'єктів з допомогою програмно-генерованої ***ШІМ*** недоцільно, оскільки використовується значна частина обчислювальних ресурсів мікрокомп'ютера. Як наслідок, роботу сервоприводів буде порушено або зовсім не буде можливості керувати ними. Ця проблема успішно вирішується шляхом підімкнення багатоканального зовнішнього ***ШІМ***-контролера. Прикладом такого пристрою є 16-канальний 12-bit PWM/Servo-модуль з ***I^2C***-інтерфейсом на ***PCA9685*** (рис. 3.1).

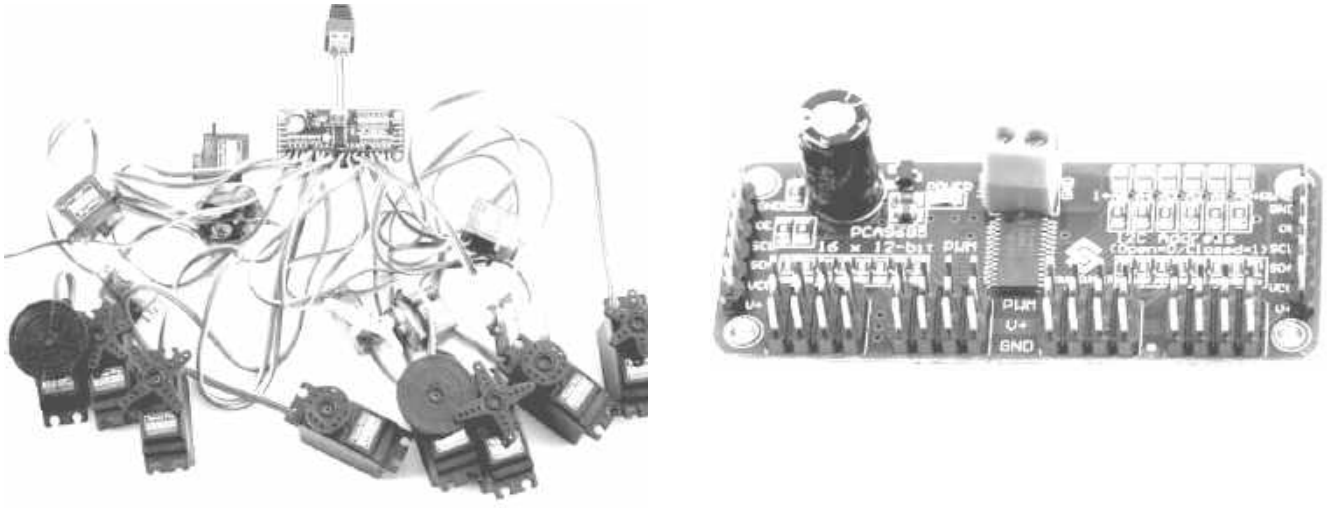


Рис. 3.1

PCA9685 − це 16-канальний 12-розрядний контролер. Частота ***ШІМ*** настроюється в межах від 24 до 1526 Гц.

Контролер керується з допомогою шини ***I^2c***. На цій платі є дві групи рознімів для шини з двох боків, що дає змогу підмикати послідовно кілька плат або інші пристрої до шини ***I^2c***. На платі є перемички, з допомогою яких можна встановити адресу пристрою, що є відмінною від стандартної. Тому якщо 16 каналів недостатньо, то можна послідовно ввімкнути кілька таких плат, установивши перемичками на кожній свою адресу.

Живлення контролера і виходів ***ШІМ***-каналів розділено і може становити від 3 до 5 В. Для цих каналів допускається максимальна напруга 6 В. Живлення для ***ШІМ***-каналів можна подавати на контакти (***V +***) або через клему. На платі є місце для фільтрувального конденсатора. При великих навантаженнях живлення може бути нестабільним, що негативно позначається на роботі керованих пристроїв. Щоб уникнути цього, рекомендується живити сервоприводи від окремого джерела постійного струму, наприклад від акумуляторних батарей. Схему підімкнення трьох сервоприводів до ***Raspberry Pi*** через ***ШІМ***-контролер показано на рис. 3.2.

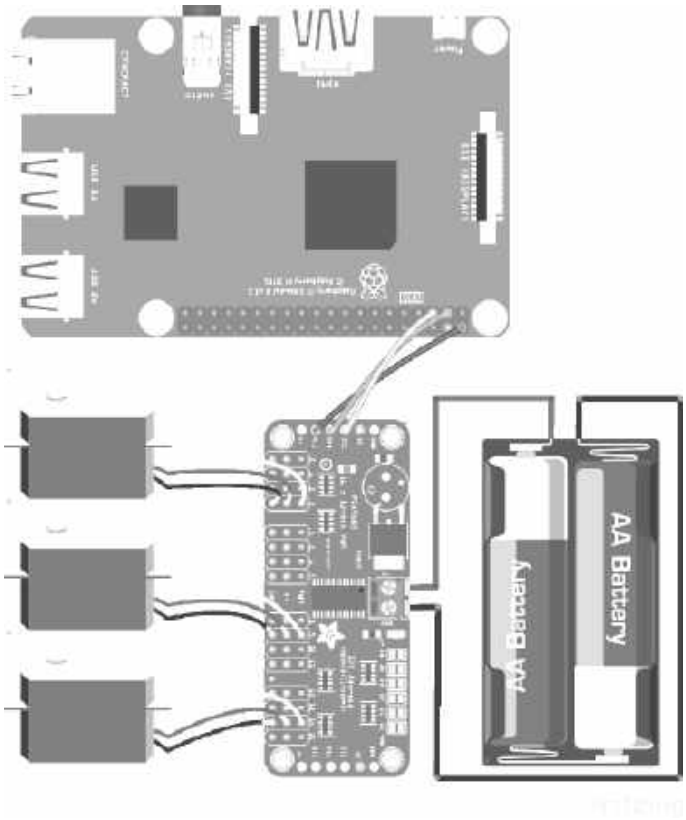


Рис. 3.2

Наявність 3-пінових рознімів дає змогу легко підімкнути 16 сервоприводів до плати, а для підімкнення ***ШІМ***-контролера до комп'ютера ***Raspberry Pi*** досить з'єднати виходи ***Vcc*** і підімкнути контакти I^2C.

I^2C (англ. Inter-Integrated Circuit) – послідовна асиметрична шина для зв'язку між інтегральними схемами всередині електронних приладів, у якій використовуються дві двонаправлені лінії зв'язку (SDA і SCL). Її застосовують для з'єднання низькошвидкісних периферійних компонентів з процесорами й микроконтролерами (наприклад, на материнських платах, у вбудованих системах, мобільних телефонах).

В I^2C використовуються дві двонаправлені лінії, «підтягнуті» до напруги живлення, які керуються через відкритий колектор або відкритий стік − послідовну лінію даних (SDA, англ. Serial DAta) і послідовну лінію тактування (SCL, англ. Serial CLock). Стандартними напругами є +5 або +3,3 В, однак допускаються й інші. Класична адресація містить 7-бітовий адресний

простір з 16 зарезервованими адресами. Це означає, що розробники мають до 112 вільних адрес для підімкнення периферії на одну шину. Основний режим роботи − 100 кбіт/с; режим роботи зі зменшеною швидкістю − 10 кбіт/с. Важливо також, що в стандарті допускається припинення тактування для роботи з повільними пристроями.

Програмна реалізація керування багатьма сервоприводами

Перш ніж розпочинати написання програми для керування сервоприводами, необхідно завантажити файл бібліотеки роботи з модулем ***PCA9685***, потім створити директорію, у якій будуть знаходитися файли бібліотеки й програми керування сервоприводами (рис. 3.3).

Рис. 3.3

Наведемо код, який дає змогу керувати одним сервоприводом, підімкненим до ***ШІМ***-контролера:

```
import PCA9685 as servo
import time                    # Імпорт необхідних модулів

MinPulse = 200
MaxPulse = 700

pwm = servo.PWM()              # ініціалізація сервоприводів

pwm.frequency = 60

                               # Установлення значень ШІМ
                               для задання кута повороту сер-
                               воприводу
Current_PWM=input(“Set PWM”)
pwm.write(14, 0, Current_PWM)
```

Створення інтерфейсу для керування сервоприводом

Під час виконання завдання необхідно написати інтерфейс для спрощеної роботи із сервоприводами, для чого можна використовувати бібліотеку ***Tkinter***.

Tkinter − це кросплатформена бібліотека для розроблення графічного інтерфейсу мовою ***Python*** (починаючи з ***Python 3.0*** її перейменовано в ***tkinter***). ***Tkinte***r входить до складу стандартних дистрибутивів

Python. Для того щоб почати працювати з цією бібліотекою, підімкнемо її з допомогою команди `from tkinter import *`. Унаслідок цього отримаємо такий код для керування одним сервоприводом з допомогою елемента `scrollbar`:

```
from tkinter import *
import PCA9685 as servo
import time
MinPulse = 200
MaxPulse = 700
pwm = servo.PWM()              # ініціалізація сервоприводів
pwm.frequency = 60

class App:
    def __init__(self, master):
        frame = Frame(master)
        frame.pack()
        scale = Scale(frame, from_=0, to=180,
# Задається діапазон змінення значень «повзунка»
              orient=HORIZONTAL, command=self.update)
        scale.grid(row=0)

    def update(self, angle):
        Current_PWM = 2.8*float(angle) + 200
# 2.8 – коефіцієнт перетворення значення кута на ШІМ-сигнал
        pwm.write(14, 0, Current_PWM)
root = Tk()
root.wm_title('Servo Control')
app = App(root)
root.geometry("300x150+10+10")
root.mainloop()
```

Результатом виконання цього коду буде керування одним сервоприводом. Інтерфейс програми зображено на рис. 3.4.

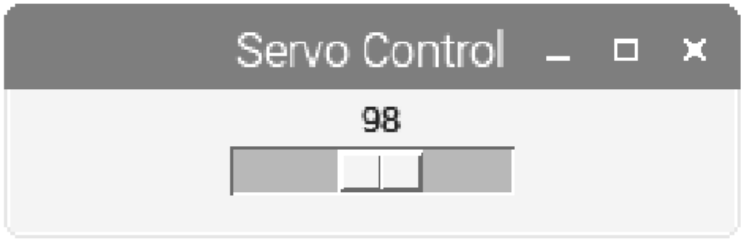


Рис. 3.4

Завдання для самостійної роботи

Реалізувати керування трьома сервоприводами з використанням ***ШІМ***-контролера. Для цього необхідно:

- підімкнути сервоприводи до відповідних виводів ***ШІМ***-контролера;
- використовуючи рекомендації і запропоновані команди, видавати три різних ***ШІМ***-сигнали керування сервоприводом; проаналізувати роботу об'єктів;
- написати інтерфейс для керування трьома сервоприводами, задаючи кутове положення.

Звіт про виконання лабораторної роботи повинен містити:

- схеми керування електричні принципові;
- тексти програмних кодів, які використовуються в цій роботі;
- скріншот інтерфейсу програми;
- вирази для розрахунку коефіцієнтів перетворення змін ***ШІМ***-сигналу на кутове положення вала сервоприводу
- висновки за результатами проведених досліджень.

Контрольні запитання

1. Назвіть недоліки керування сервоприводами, підімкненими безпосередньо до мікрокомп'ютера?
2. Опишіть призначення й принцип роботи зовнішнього ***ШІМ***-контролера?
3. Навіщо використовується шина ***I²C***? Як нею керують?
4. Як здійснюється програмна й апаратна реалізація ***ШІМ***-сигналу? У чому полягають переваги й недоліки кожної з них?
5. Як впливають характеристики окремого сервоприводу на роботу з ним?

Лабораторна робота № 4

КЕРУВАННЯ ПОРТАМИ GPIO ЧЕРЕЗ ВЕБ-ІНТЕРФЕЙС

Мета роботи – вивчення й використання методів керування портами ***GPIO*** одноплатного комп'ютера ***Raspberry Pi*** з допомогою веб-сервера-інтерфейсу. Набуття практичних навичок роботи в цій області.

Стислі теоретичні відомості

При використанні систем технічного зору й робототехнічних систем різного призначення постійно виникають проблеми дистанційного керування роботою окремих пристроїв. Це, наприклад, завдання керування положенням веб-камери з допомогою сервоприводів у системах віддаленого відеоспостереження. Такі завдання потребують вирішення при створенні систем «Розумного будинку» – віддалений контроль температури, тиску, керування системою електроживлення та ін. Для цього зазвичай використовують доступ до портів ***GPIO*** одноплатного комп'ютера ***Raspberry Pi*** з допомогою фреймворка ***Webiopi***. Це дає змогу віддалено керувати всіма портами ***GPIO***. Вивчимо завдання детальніше на всіх етапах її практичного вирішення.

Доступ до портів GPIO через веб-інтерфейс. Для доступу до портів ***GPIO*** через веб-інтерфейс скористаємося ***Webiopi*** – фреймворком, що дає змогу контролювати стан і керувати всіма портами ***GPIO*** локально або віддалено з браузера або будь-якого застосування.

Доступ до портів GPIO через веб-інтерфейс

Webiopi є пакетом програм, спеціально розробленим для ***Raspberry Pi*** для віддаленого керування пристроями. Спільно з ***Raspberry P***і він реалізує технологію ***Internet of Things*** (інтернет речей) і дає змогу створювати різні застосування, призначені для користувача. Основні можливості цього фреймворка відповідно до його опису на офіційному сайті http://webiopi.trouch.com зводяться до таких:

- убудований web-сервер, реалізований мовою ***Python***;
- убудована підтримка більш ніж 30 пристроїв з інтерфейсами ***UART***, ***SPI***, ***I²C***, ***1***-wire;
- бібліотеки ***Javascript/html*** для створення web-інтерфейсу;
- бібліотеки ***Python/Java*** для створення додатків під ***Android***;
- підтримка протоколу ***COAP***, призначеного для керування й взаємодії між простими електронними пристроями через мережу.

Webiopi має відкритий код, який користувач може змінити. Це дає змогу збільшити кількість завдань для вирішення. Для настроєння пакета під конкретне завдання слід змінити файл конфігурації. Наприклад, у цей файл записують ***GPIO***-контакти (pins), до яких підімкнено пристрої. Якщо використовуються датчики, то їх також заносять у конфігураційний файл. Проте в деяких випадках необхідно також увімкнути і драйвер пристрою.

Розглянемо установку ***Webiopi***. З версією ***Webiopi 0.7.1*** і вище стабільно працює ***Raspberry Pi 2***, який має 40-піновий порт ***GPIO***. Для установки фреймворка необхідно перейти на офіційний сайт (http://webiopi.trouch.com) і скачати архів клієнта ***Webiopi***. Далі в терміналі по черзі вписати такі команди:

```
$ tar xvzf Webiopi-x.y.z.tar.gz
$ cd Webiopi-x.y.z
$ sudo ./setup.sh                    (x.y.z – версія клієнта)
```

Після завершення установлення необхідно активувати автозапуск ***Webiopi,*** який потрібен для того, щоб кожного разу після ввімкнення ***Raspberry Pi*** не виконувати запуск додатка вручну. Для цього виконаємо таку команду:

```
$ sudo update-rc.d webiopi defaults
```

Після цього необхідно перезапустити ***Raspberry Pi****:*

```
$ sudo reboot
```

Після перезавантаження ***Raspberry Pi*** перевіримо роботу ***Webiopi***, для чого з будь-якого комп'ютера в тій же локальній мережі в браузері наберемо IP-адресу з портом підімкнення 8000:

`http:// xxx.xx.xxx.xx:8000`

Замість символів «x» необхідно вписати IP-адресу, яку можна дізнатися, увівши в терміналі таку команду:

`$ ifconfig`

Далі отримаємо список усіх мережних підімкнень. Знайдемо розділ підкімкнення `eth0`, у другому рядку буде вказано IP-адресу inet addr: `192.168.1.81` (рис. 4.1).

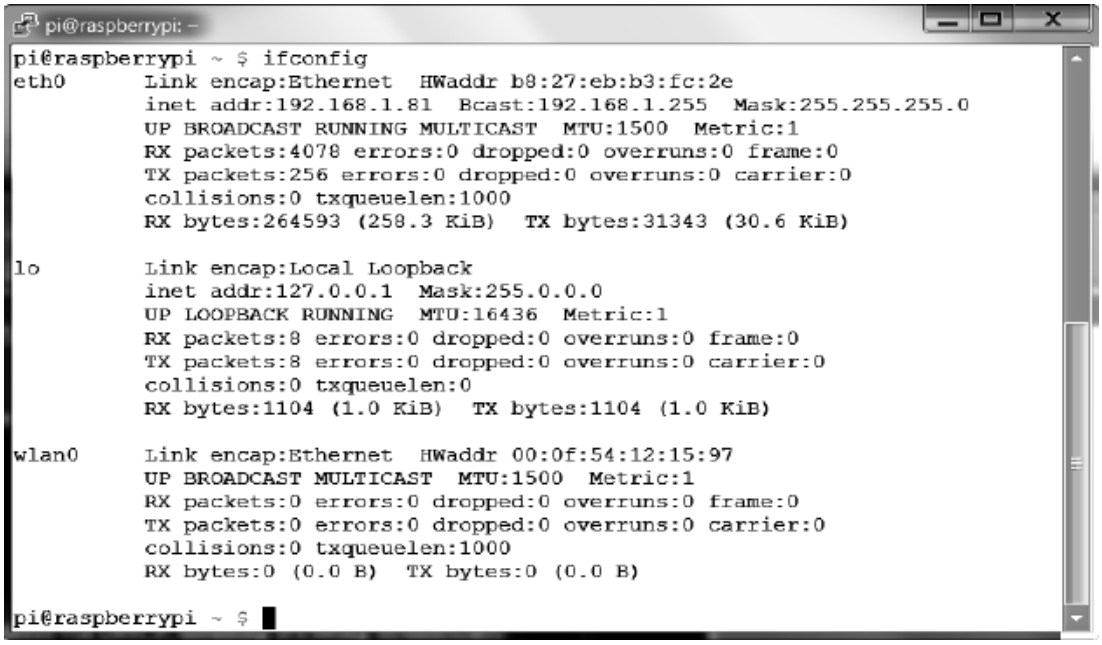


Рис. 4.1

Для доступу до ***Webiopi*** необхідно у формі, що відкрилася, увести логін і пароль. За замовчуванням логін – «`webiopi`», пароль – «`raspberry`». Їх, як і номер порту, потім можна буде змінити, але про це поговоримо окремо, а для наших експериментів скористаємося стандартними параметрами (рис. 4.2).

Рис. 4.2

Після змінення пароля сервер необхідно перезавантажити:

```
sudo /etc/init.d/webiopi restart
```

Для зняття захисту при вході в ***Webiopi*** необхідно або ввести порожній пароль, або видалити файл /`etc/webiopi/passwd`.

Отже, якщо при установленні ***Webiopi*** уведено правильні логін і пароль, то в браузері відкриється сторінка, показана на рис. 4.3.

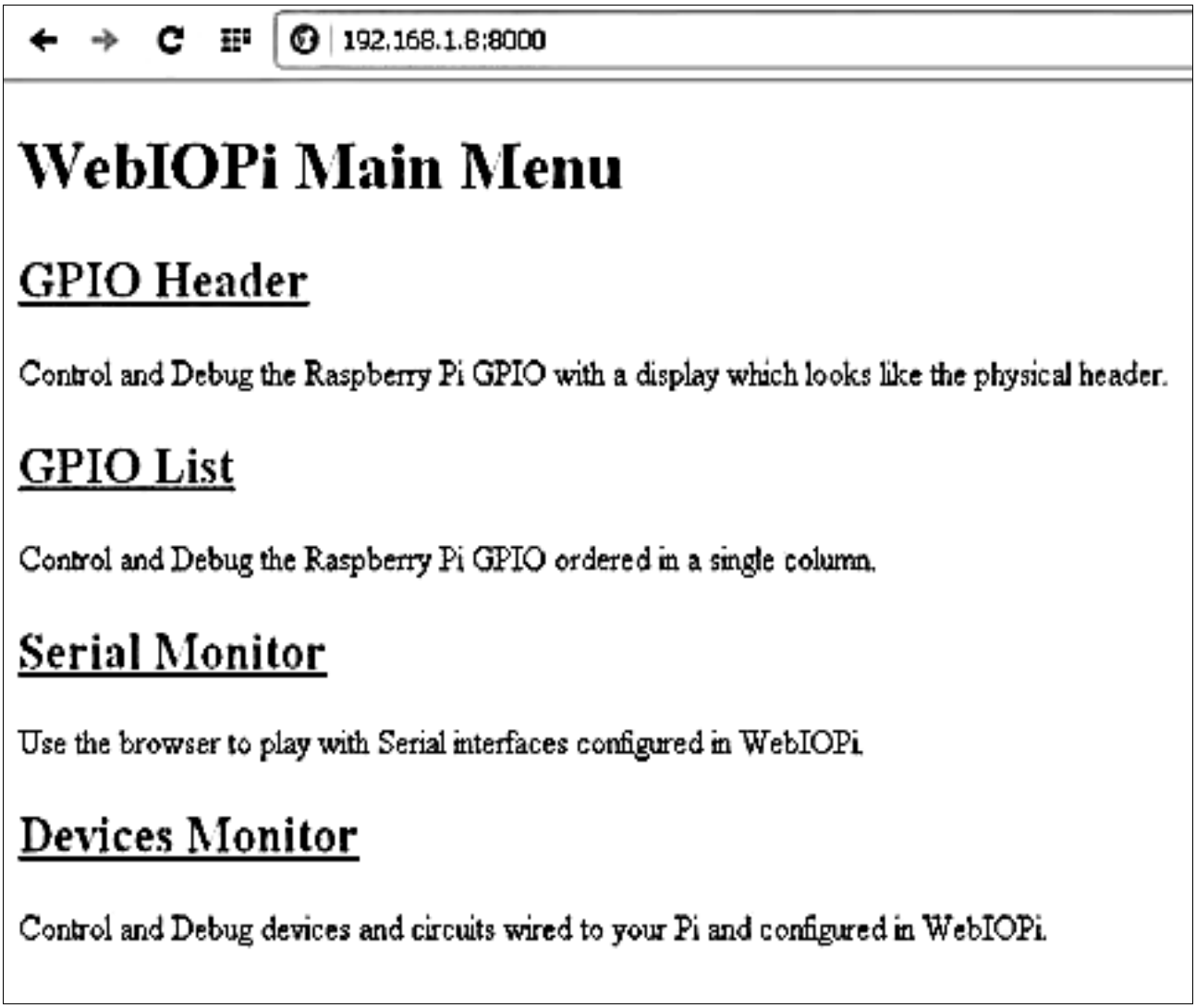


Рис. 4.3

На цій сторінці є чотири пункти меню:

- ***GPIO Header*** і ***GPIO List*** відкривають два варіанти графічних інтерфейсів роботи з портами введення/виведення;
- ***Serial Monitor*** – термінал для роботи з послідовним портом ***UART***;
- ***Devise Monitor*** – підімкнені до ***GPIO*** датчики.

Виберємо в меню пункт ***GPIO Header*** і потрапимо на сторінку з графічним інтерфейсом керування портами (рис. 4.4).

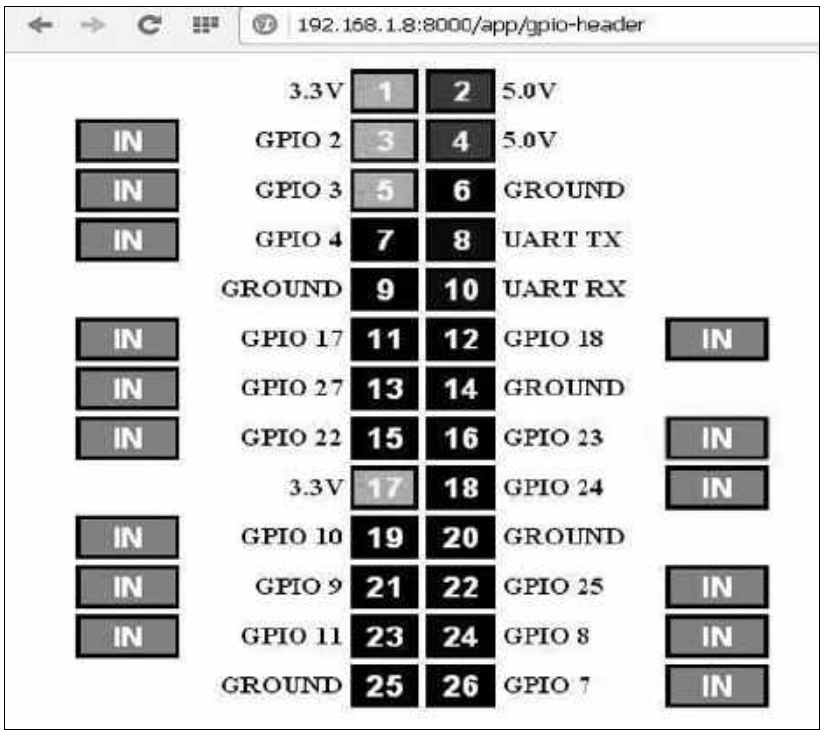


Рис. 4.4

Унаслідок виконаних дій, якщо все правильно підімкнено, відкриється доступ до стандартного web-інтерфейсу, з якого можна керувати всіма 40 пінами, змінюючи вхід і вихід кожного з них.

Настроєння Webiopi

Настроїти сервер ***Webiopi*** можна шляхом внесення змін до файла його конфігурації `/etc/webiopi/config`. Синтаксис цього файла є таким самим, як і синтаксис інших `INI`-файлів, тобто має кілька розділів, що містять пари «ключ = значення».

Блок [HTTP] дає змогу ввімкнути або вимкнути HTTP, а також змінити значення порту. У цьому блоці можна змінити місце розташування файла `passwd`, домашньої теки й назву індексного файла `HTML`:

```
[HTTP]
enabled = true
port = 8000
passwd-file = /etc/webiopi/passwd
doc-root = /home/pi/webiopi/examples/scripts/macros
welcome-file = index.html
```

Блок [COAP] дає змогу ввімкнути або вимкнути сервер COAP, а також змінити значення порту:

```
[COAP]
enabled = true
port = 5683
multicast = true
```

Блок [GPIO] дає змогу встановити призначені для користувача настройки і значення для портів ***GPIO*** при запуску ***Webiopi***. Наприклад:

```
[GPIO]
21 = IN
23 = OUT 0
24 = OUT 0
25 = OUT 1
```

Блок [~GPIO] дає змогу встановити призначені для користувача настройки і значення для портів ***GPIO*** при перезавантаженні ***Webiopi***. Наприклад:

```
[~GPIO]
21 = IN
23 = IN
24 = IN
25 = OUT 0
```

Блок [SCRIPTS] визначає список скриптів, що виконуються під час запуску ***Webiopi***. Наприклад:

```
[SCRIPTS]
Myscript =
/home/pi/webiopi/examples/scripts/macros/script.py
```

Блок [REST] дає змогу з допомогою REST API обмежити доступ Get/post до деяких портів:

```
[REST]
gpio-export = 21, 23, 24, 25
gpio-post-value = false
gpio-post-function = false
device-mapping = false
```

Блок [DEVICES] дає змогу підімкнути пристрої, які підтримує ***Webiopi***, до конкретних портів ***GPIO*** (список пристроїв, що підтримують ***Webiopi***, буде розглянуто далі):

```
usb0 = Serial device:ttyUSB0 baudrate:9600
adc = MCP3008
dac = MCP4922 chip:1
gpio0 = MCP23017
gpio1 = MCP23017 slave:0x21
gpio2 = MCP23017 slave:0x22
pwm0 = PCA9685
pwm1 = PCA9685 slave:0x41
```

Блок [ROUTES] визначає список маршрутів переадресації. Це дає змогу під час використання REST API приховати в адресному рядку зна-

чення порту доступу або інше призначення, тобто надати адресам зручного вигляду:

```
/bedroom/light = /GPIO/25/value
/bedroom/temperature = /devices/temp2/sensor/temperature/c
```

Формування власного інтерфейсу

Ця процедура починається зі створення папки проекту, у якій розміщуватимуться html-файл і файл скрипту ***Python*** (`script.py`). Наступним кроком буде змінення файла конфігурації ***Webiopi***, щоб фреймворк звертався до наших файлів, а не до стандартних. Для цього в терміналі необхідно ввести таку команду:

```
$ sudo webiopi /etc/webiopi/config
```

У терміналі відкриється файл конфігурацій (рис. 4.5), що містить такі розділи:

- HTTP
- SCRIPTS

У розділі HTTP необхідно додати шлях до `html`-файла свого проекту, який набуде приблизно такого вигляду:

```
[HTTP]
```

- `enabled = true`
- `port = 8000`
- `passwd-file = /etc/webiopi/passwd`
- `doc-root = /home/pi/webiopi/examples/scripts/macros`
- `doc-root = /home/pi/myproject/`
- `welcome-file = index.html`

Наступною дією буде вказання файла скрипту ***Python*** нашого проекту. Для цього в розділі ***SCRIPTS*** необхідно внести змінення, додавши в розділ такий рядок:

```
myscript = /home/pi/myproject/script.py
```

Рис. 4.5

Наступним кроком буде створення файлів `index.html` і `script.py` у папці проекту. Файл `index.html` відповідає за зовнішній вигляд нашого інтерфейсу і зв'язок інтерфейсу з web-сервером. Файл `script.py` є з'єднувальною частиною програмного й апаратного сегментів, у якому описується все, що відбуватиметься під час тієї або іншої дії в інтерфейсі.

У html-файлі використовується стандартна розмітка веб-сторінки з підімкненням стилів і скриптів. Взаємозв'язок інтерфейсу здійснюється з допомога скриптів `Javascript` або `Jquery`. Оскільки використання бібліотеки ***Javascript*** істотно підвищує ефективність роботи зі створення інтерфейсу, наведемо детальніші відомості про неї.

Бібліотека Javascript

Webiopi містить `http`-сервер, що забезпечує як `html`-ресурси, так і інтерфейс `REST API` для керування виводами ***GPIO***. Під час запуску сервера браузер спочатку завантажує `html`-файл з увімкненою `javascript`-бібліотекою `webiopi.js`, що містить бібліотеку `jquery` для асинхронних викликів до `REST API`. Цей метод є дуже ефективним, тому що не потребує для оновлення даних оновлення сторінки. Розширити можливості ***Webiopi*** можна шляхом завантаження створеного з використанням ***arduino***-подібного синтаксису призначеного для користувача сценарію ***Python***, що містить функції настроєння виводів ***GPIO*** (рис. 4.6).

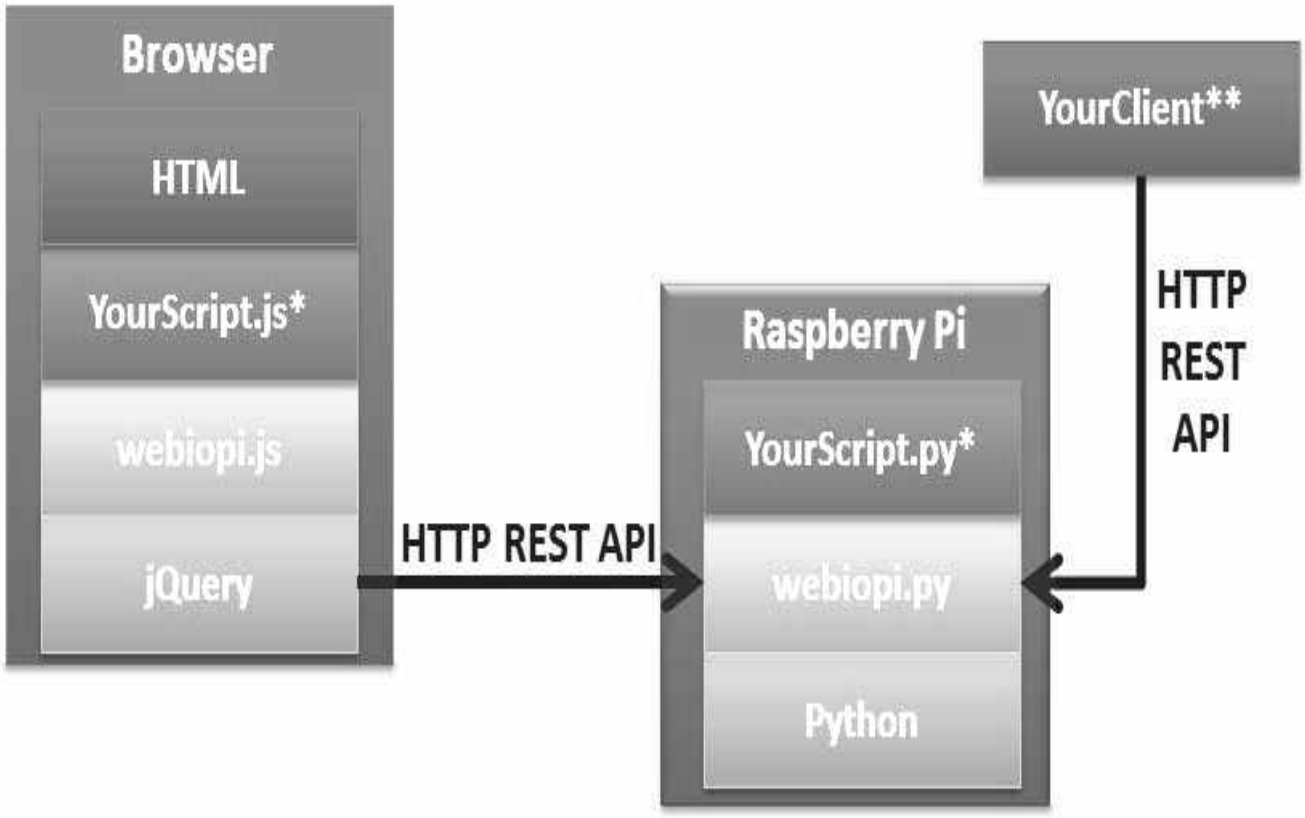


Рис. 4.6

Для підімкнення бібліотеки webiopi.js на сторінці HTML у блоці [HEADER] поміщаємо такий код:

```
<script type="text/javascript" src="/webiopi.js"></script>
```

Функції бібліотеки webiopi.js

Примітка: тут і далі webiopi() – повертаний об'єкт ***WebIopi***.

Функція Webiopi.ready

Функція WebIOPi.ready()реєструє функцію зворотного виклику при завантаженні бібліотеки webiopi.js.
Синтаксис: WebIOPi.ready(callback)
Параметр: callback – функція зворотного виклику.

Функція Webipi.setFunction

Функція WebIOPi.setFunction()установлює призначення виводу ***GPIO***.
Синтаксис:

- WebIOPi.setFunction(gpio, func)
- WebIOPi.setFunction(gpio, func, callback)

Параметри:

- Gpio – номер виводу ***GPIO***;
- func – призначення виводу:
  - IN – вивід, конфігурований як вхід;
  - OUT – вивід, конфігурований як вихід;
  - PWM – вивід, конфігурований як ***ШІМ***-вихід.
- callback – функція зворотного виклику.

Функція Webiopi.digitalWrite

Функція Webiopi.digitalWrite()установлює цифрове значення виводу ***GPIO***, конфігурованого як вихід (OUT).
Синтаксис:

- Webiopi.digitalWrite(gpio, value)
- Webiopi.digitalWrite(gpio, value, callback)

Параметри:

- gpio – номер виводу GPIO;
- value – значення для виводу (0 или 1);
- callback – функція зворотного виклику.

Функція Webiopi.digitalRead

Функція Webiopi.digitalRead()набуває значення з виводу ***GPIO***, конфігурованого як вхід (IN).

Синтаксис:

- `Webiopi.digitalRead(gpio);`
- `Webiopi.digitalRead(gpio, callback).`

Параметри:

- `gpio` — номер виводу ***GPIO***;
- `callback` — функція зворотного виклику;
- повертане значення: `value (0 або 1)`.

Функція Webiopi.toggleValue

Функція `Webiopi.toggleValue()` перемикає значення на виводі ***GPIO*** на протилежне.

Синтаксис: `Webiopi. toggleValue(gpio)`.

Параметр: `gpio` – номер виводу ***GPIO***.

Функція WebIopi.callMacro

Функція `Webiopi.callMacro()` виконує макрофункцію на сервері ***Webiopi***. Макрос може оголошуватися в скрипті ***Python***, який запускається під час старту ***WebIopi***.

Синтаксис:

- `Webiopi.callMacro(macro)`
- `Webiopi.callMacro(macro, args)`
- `Webiopi.callMacro(macro, args, callback)`

Параметри:

- `macro` – найменування макрофункції;
- `args` – масив аргументів, що передаються для макрофункції;
- `callback` – функція зворотного виклику.

Функція WebIopi.outputSequence

Функція `Webiopi.outputSequence()` відправляє послідовність бітів на вихід ***GPIO***.

Синтаксис:

- `Webiopi.outputSequence(gpio, period, sequence)`
- `Webiopi.outputSequence(gpio, period, sequence, callback)`

Параметри:

- `gpio` – вивід ***GPIO***;
- `period` – час відправлення одного біта, мс;
- `sequence` – послідовність бітів;
- `callback` – функція зворотного виклику.

Приклад:

```
var sequence = "01010100110011001100101010";
// відправка послідовності на gpio 7 з періодом 100 мсек
```

```
webiopi().outputSequence(7, 100, sequence,
sequenceCallback);
```

Функція Webiopi.pulse

Функція `Webiopi.pulse()`відправляє імпульс на вихід ***GPIO***.
Синтаксис:

- `Webiopi.pulse(gpio);`
- `Webiopi.pulse(gpio, callback).`

Параметри:

- `gpio` – вивід ***GPIO***;
- `callback` – функція зворотного виклику.

Функція Webiopi.pulseRatio

Функція `Webiopi.pulseRatio()`відправляє ***ШІМ***-сигнали на вихід ***GPIO***.
Синтаксис:

- `Webiopi.pulseRatio (gpio, ratio);`
- `Webiopi.pulseRatio (gpio, ratio, callback).`

Параметри:

- `gpio` – вивід ***GPIO***;
- `ratio` – значення для сігналу ШІМ (0,0…1,0);
- `callback` – функція зворотного виклику.

Функція Webiopi.pulseAngle

Функція `Webiopi.pulseAngle()`відправляє ШІМ-сигнали на вихід ***GPIO***. Ця функція є зручною під час керування сервоприводами для установки кута повороту робочого органу від –45° до +45° .
Синтаксис:

- `Webiopi.pulseAngle (gpio, angle);`
- `WebIOPi.pulseAngle(gpio, angle, callback).`

Параметри:

- `gpio` – вивід ***GPIO***;
- `angle` – значення сигналу ***ШІМ*** (от –45° до +45°);
- `callback` – функція зворотного виклику.

Функція Webiopi.createButton

Функція `WebIOPi.createButton()`повертає об'єкт – просту кнопку.
Синтаксис:

- `Webiopi.createButton(id, label);`
- `Webiopi.createButton(id, label, mousedown);`
- `Webiopi.createButton(id, label, mousedown, mouseup).`

Параметри:

- `id` – ідентифікатор кнопки;
- `label` – напис на кнопці;
- `mousedown` – функція, що викликається під час події `mousedown` (натиснення по кнопці);
- `mouseup` – функція, що викликається під час події `mouseup` (відпуск кнопки).

ПРИМІТКА. Події `mousedown` і `mouseup` в основному використовуються, коли кнопку натискують, переміщують, а потім мишку відпускають. Щоб кнопка з'явилася на сторінці, її необхідно додати, наприклад, так:

```
button = webiopi().createButton("bt1","button1",fun_down, fun_up);
$("#div1").append(button);
```

Функція Webiopi.createFunctionButton

Функція `Webiopi.createFunctionButton()`створює об'єкт – кнопку, при натисненні на яку змінюється призначення виводу ***GPIO***.

Синтаксис: `Webiopi.createFunctionButton(gpio)`

Параметр: `gpio` – вивід, призначення якого змінюється `(IN, OUT)`, при цьому на кнопці виводиться відповідний напис `(IN, OUT)`.

Функція Webiopi.createGPIOButton

Функція `Webiopi.createGPIOButton()`створює об'єкт – кнопку, при натисненні на яку змінюється значення (0 або 1) виводу ***GPIO***.

Синтаксис: `Webiopi.createGPIOButton(label, gpio).`

Параметри:

- `label` – напис на кнопці;
- `gpio` – вивід, призначення якого змінюється (0, 1), при цьому на кнопці виводиться відповідний напис (0, 1).

Функція Webiopi.createMacroButton

Функція Webiopi.createMacroButton() створює об'єкт – кнопку, при натисненні на яку виконується макрофункція на сервері (макрофункції прописано в ***python***-скріпті, що запускається під час старту `webiopi`).

Синтаксис: `Webiopi.createMacroButton(id, label, macro, args)`.

Параметри:

- `id` – ідентифікатор кнопки;
- `label` – напис на кнопці;
- `macro` – назва макрофункції на сервері;
- `args` – список параметрів, що передаються макрофункції.

Функція Webiopi.createSequenceButton

Функція `Webiopi.createSequenceButton()`створює об'єкт – кнопку, при натисненні на яку на виводі ***GPIO*** видправляється послідовність бітів.
Синтаксис: `Webiopi.createSequenceButton(id, label, gpio, period, sequence)`
Параметри:

- `id` – ідентифікатор кнопки;
- `label` – напис на кнопці;
- `gpio` – вивід для відправлення послідовності бітів;
- `period` – час відправлення одного біта, мс;
- `sequence` – послідовність бітів у вигляді рядка.

Приклад:

```
button = webiopi().createSequenceButton("but1", "label1",
25, 100,
"01010100110011001100101010");
$("#div1").append(button);
```

Функція Webiopi.createRatioSlider

Функція `Webiopi.createRatioSlider()`створює об'єкт – шкалу (рис. 1.5), положенням покажчика на якій регулюється значення ***ШІМ***-сигналу на виводі ***GPIO***.
Синтаксис: `WebIOPi.createRatioSlider(gpio, ratio).`
Параметри:

- `gpio` – вивід для відправлення послідовності бітів;
- `ratio` – початкове значення для ***ШІМ*** (0,0…1,0).

Приклад:

```
button = webiopi().createRatioSlider(24, 1.0);
$("#div1").append(button);
```

Функція Webiopi.createAngleSlider

Функція `Webiopi.createAngleSlider()`створює об'єкт – шкалу, положенням покажчика якої регулюється значення ***ШІМ***-сигналу, що подається на вивід ***GPIO*** під час керування сервоприводом для значень повороту кута робочого органу від –45° до +45° .
Синтаксис: `Webiopi.createAngleSlider(gpio, angle).`
Параметри:

- `gpio` – вивід для відправлення послідовності бітів;
- `angle` – початкове значення кута для сигналу ***ШІМ*** (від –45° до +45°).

Функція Webiopi.setLabel

Функція `Webiopi.setLabel()`змінює напис на кнопці.

Синтаксис: `Webiopi.setLabel(id, label)`
Параметри:

- `id` – ідентифікатор кнопки;
- `lable` – новий напис для кнопки.

Приклад коду для формування елементів керування Webiopi

У цьому прикладі розглянемо підімкнення світлодіода до ***Raspberry Pi 3*** (рис. 4.7) з подальшим керуванням з допомогою елементів ***Webiopi***.

Наведемо лістинг коду для створення об'єктів ***Webiopi***.

```
<html>
<head>
<meta http-equiv="Content-Type" cotent="text/html; char-
set=UTF-8">
<meta name="viewport" content = "height = device-height,
width = 420, user-scalable = no" />
<title>WebIOPi | Demo</title>
<script type="text/javascript" src="/webiopi.js"></script>
<script type="text/javascript">
webiopi().ready(function() {
    var content, button;
    content = $("#content");
    //створює кнопку необхідної функції
    button = webiopi().createFunctionButton(25);
    //додає кнопку в HTML-размітку в контейнері “content”
    content.append(button);
    button = webiopi().createGPIOButton(7, "SWITCH");
    content.append(button);
    button = webiopi().createSequenceButton("sos", "S.O.S
1", 25, 100,
    "01010100110011001100101010");
    content.append(button);
    button = webiopi().createMacroButton("macro","Print
Time","PrintTime");
    content.append(button);
    button = webiopi().createAngleSlider(23, 30);
    content.append(button);
    button = webiopi().createRatioSlider(24, 0.5);
    content.append(button);
    webiopi().refreshGPIO(true);
});
</head>
<body>
<div id="content" align="center"></div>
```

```
</body>
</html>
```

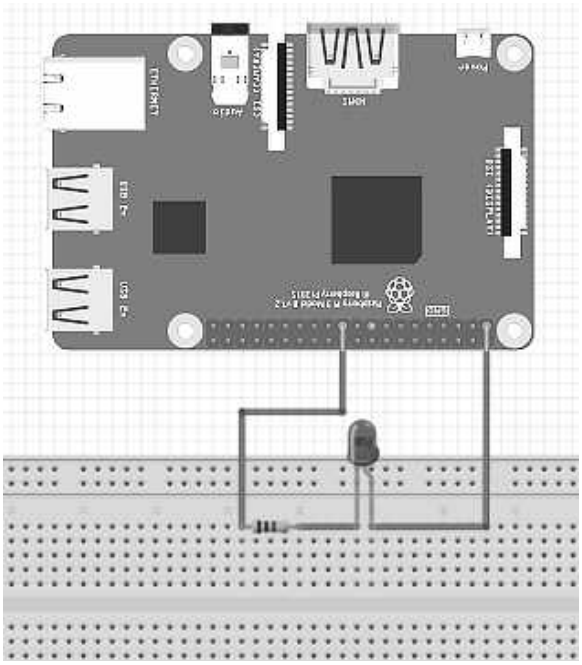


Рис. 4.7

На рис. 4.8 показано вигляд веб-сторінки для формування елементів керування ***Webiopi***, для чого було використано ***Javascript***-бібліотеку ***webiopi.js***.

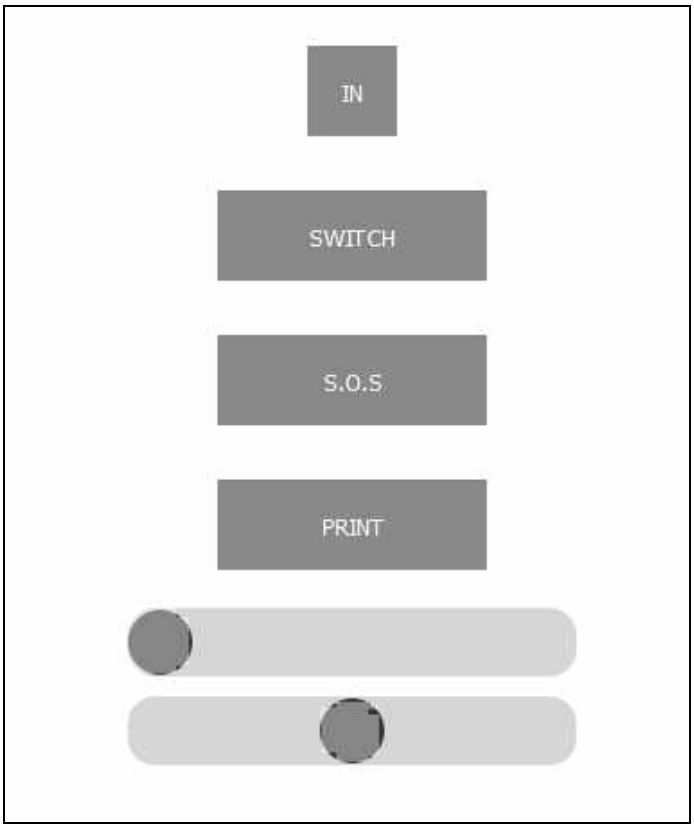


Рис. 4.8

Унаслідок синтезу було створено такі керувальні елементи веб-сервера-інтерфейсу:

- **IN** – реалізує функцію `createfunctionbutton()`, що створює об'єкт – кнопку, при натисненні на яку змінюється призначення виводу ***GPIO***;
- **SWITCH** – реалізує функцію `creategpiobutton()`, що створює об'єкт – кнопку, при натисненні на яку змінюється значення (0 або 1) виводу ***GPIO***;
- **SOS** – реалізує функцію `createsequencebutton()`, що створює об'єкт – кнопку, при натисненні на яку на вивід ***GPIO*** відправляється послідовність бітів;
- **Print** – реалізує функцію `createmacrobutton()`, що створює об'єкт – кнопку, при натисненні на яку виконується макрофункція на сервері (макрофункції прописуються в python-скріпті, що запускається під час старту ***Webiopi***);
- **Шкала 1** – реалізує функцію `createratioslider()`, що створює об'єкт – шкалу, положенням покажчика на якій регулюється значення ***ШИМ***-сигналу на виводі ***GPIO***.
- **Шкала 2** – реалізує функцію `createangleslider()`, створює об'єкт – шкалу, положенням покажчика якої регулюється значення ***ШІМ***-сигналу, що подається на вивід ***GPIO*** під час керування сервоприводом для значень повороту кута робочого органу від –45° до +45° .

Кожен створений об'єкт дає змогу керувати іншим об'єктом (вмикати/вимикати, вмикати з певною частотою, а також керувати з допомогою ***ШІМ***-сигналу).

Завдання для самостійної роботи

Виконати завдання зі створення веб-сервера-інтерфейсу для дистанційного керування роботою портів ***GPIO*** комп'ютера ***Raspberry Pi***. Для цього необхідно:

- зібрати схему керування яскравістю світлодіода відповідно до рекомендацій і перевірити роботу схеми для різних варіантів формування ***ШІМ*** (програмного й апаратного);
- підімкнути до шини ***GPIO*** комп'ютера ***Raspberry Pi*** сервопривід згідно з рис. 2.7 і експериментально перевірити якість роботи для різних способів керування (програмно сформованого ***ШІМ*** і ***ШІМ***, сформованого через ***DMA***).

Звіт про виконання лабораторної роботі повинен містити:

- схеми керування електричні принципові й схеми монтажні;
- тексти програмних кодів, які використовуються в цій роботі;
- висновки за результатами проведених досліджень.

Контрольні запитання

1. Для чого призначено апаратні порти введення/виведення ***GPIO*** в комп'ютері ***Raspberry Pi***?
2. Який конструктивний вигляд має рознім ***GPIO*** на платі комп'ютера, якою є його цоколівка (терморегулятор)?
3. Як здійснюється доступ до портів ***GPIO*** через веб-інтерфейс?
4. Що таке фреймворк ***WebIOPi***? Як його установлюють і настроюють?
5. Як установлюють фреймворк Webiopi на ОС ***Raspbian***?
6. Як змінити пароль ***Webiopi***?
7. Як використовувати ***Javascript***-бібліотеку ***webiopi.js*** для настроювання веб-інтерфейсу?
8. Наведіть приклад керування яскравістю світіння діода з допомогою модулятора ***ШІМ***.
9. Як здійснити генерацію апаратного сигналу ***ШІМ*** на ***Python***?
10. Якими принципи роботи сервоприводів в системах керування? Наведіть основні характеристики сервоприводів.
11. Поясніть значення змінних у командах `p.ChangeDutyCycle`; `wiringpi.pwmSetClock`; `wiringpi.pwmSetRange` під час керування сервоприводом.

Лабораторна робота № 5

ДИСТАНЦІЙНЕ КЕРУВАННЯ РОБОТОЮ ВЕБ-КАМЕРИ

Мета роботи − вивчення й використання методів керування портами ***GPIO*** одноплатного комп'ютера ***Raspberry Pi*** з допомогою веб-інтерфейсу на прикладі дистанційного керування роботою веб-камери з допомогою сервоприводів.

Схеми експериментальних досліджень

Для практичного закріплення навичок дистанційного керування різними пристроями з допомогою веб-інтерфейсу вивчимо методи дистанційного керування веб-камерою з допомогою сервоприводів.

Для створення самостійного проекту з використанням фреймворка ***Webiopi*** необхідно реалізувати такі його компоненти:

- веб-інтерфейс − HTML-сторінка з використанням бібліотеки ***webiopi.js***;
- серверний файл мовою ***Python***, що запускається під час старту ***Webiopi*** для реалізації функціоналу веб-інтерфейсу;
- файл конфігурації з установленням початкових значень портів.

Розглянемо проект керування камерою, закріпленою на підвісі з трьома ступенями свободи, забезпеченому сервоприводами. Таке керування

дає змогу повертати камеру, а також робити знімки й відправляти їх на електронну пошту.

Для реалізації цього проекту потрібні такі деталі й комплектування:

- USB-камера для комп'ютера ***Raspberry Pi*** (рис. 5.1);
- підвіс для камери і три сервоприводи EMAX з металевим редуктором (рис. 5.2);
- додатковий блок живлення 5 V;

Для керування сервоприводами передбачено три виводи ***GPIO*** (***GPIO22***, ***GPIO23*** і ***GPIO24***).

Електричну схему проекту показано на рис. 5.3. Щоб запобігти перевантаженню комп'ютера ***Raspberry Pi***, для сервоприводів призначено окреме джерело живлення з підвищеною здатністю до навантаження.

Спочатку настроїмо у файлі конфігурацій `config`, який знаходиться в папці `/etc/webiopi/`, необхідні параметри: шлях до домашньої папки і шлях до файла, що виконується при запуску `webiopi`:

```
doc-root = /home/pi/webiopi/examples/servo-camera
script = /home/pi/webiopi/examples/servo-camera/camera.py
```

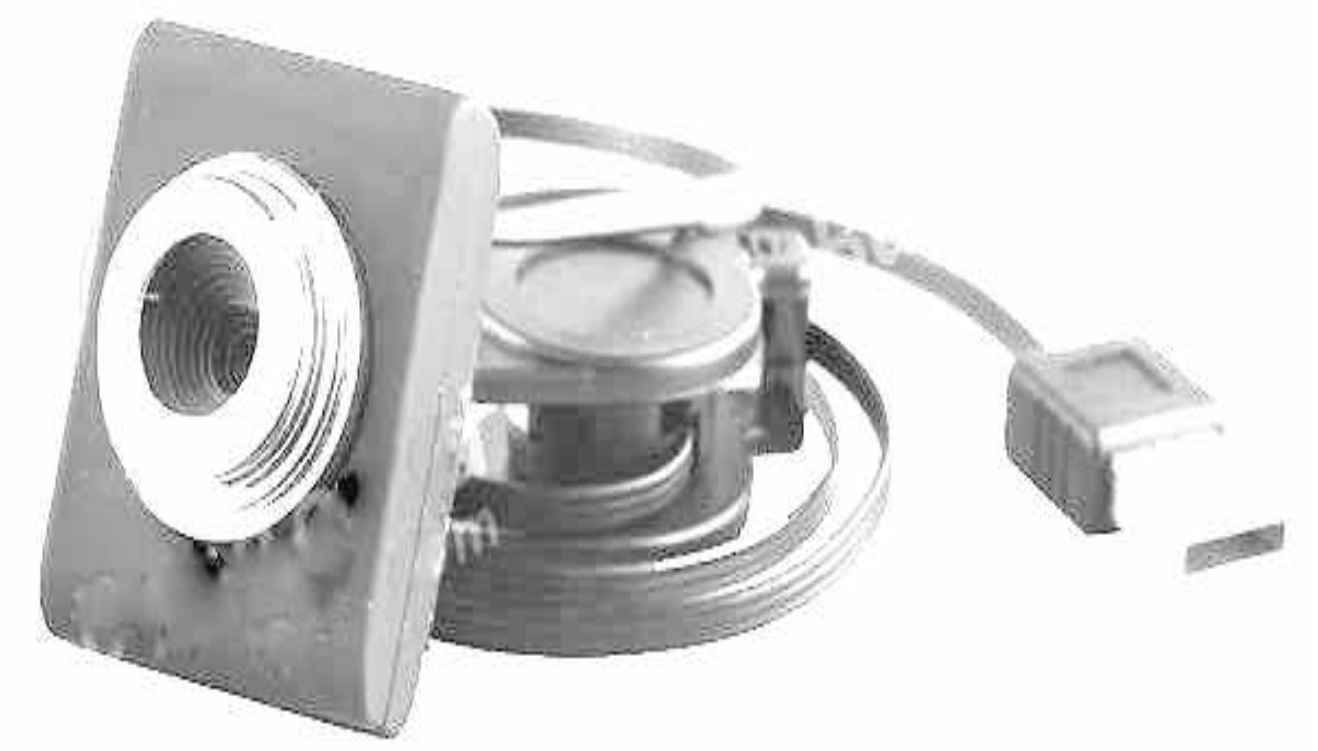


Рис. 5.1

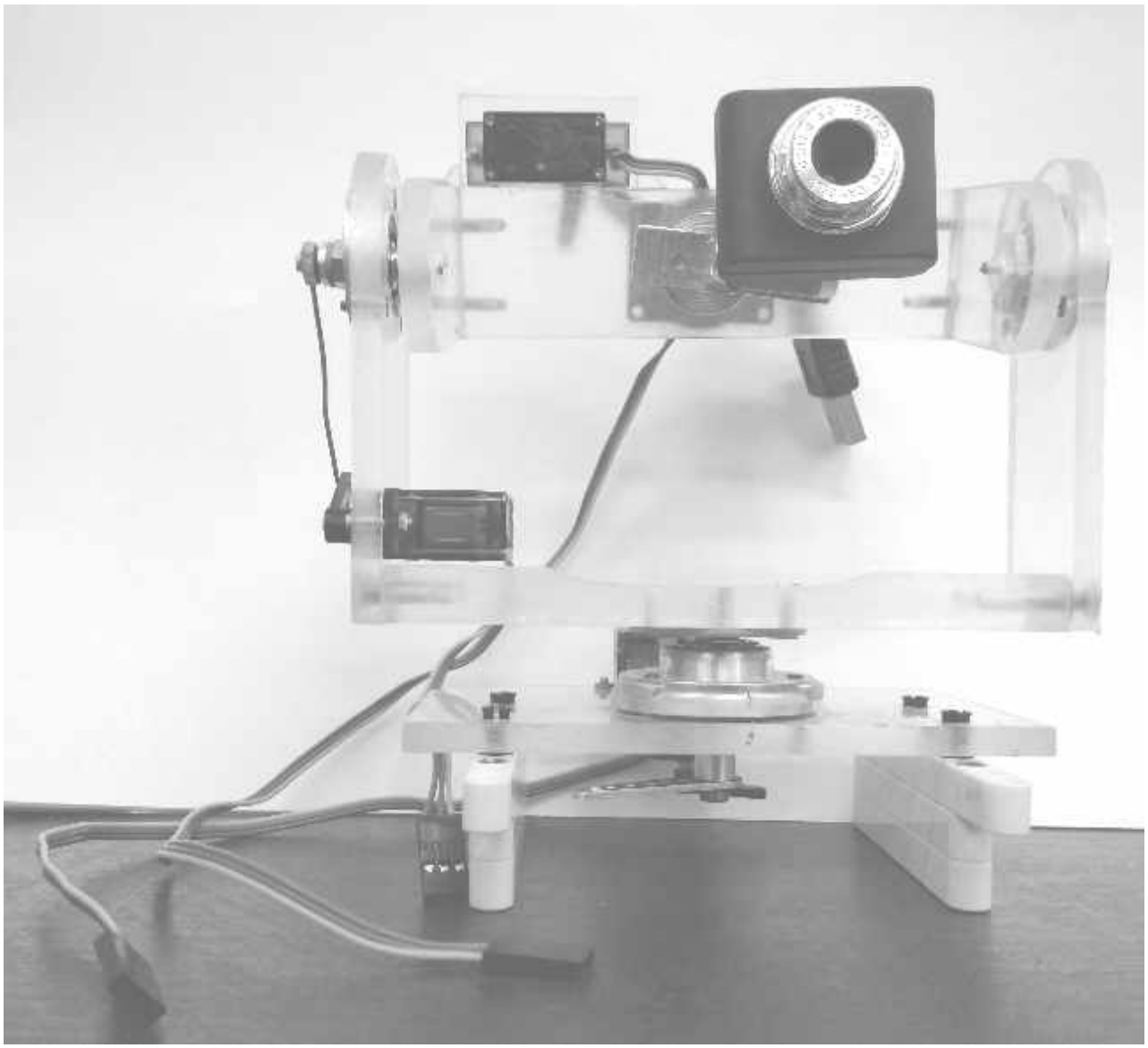


Рис. 5.2

Спочатку настроїмо відеотрансляцію з камери. Для цього встановимо й настроїмо додаток ***MJPG-Streamer***. Наведемо команди терміналу для установлення й настроєння програми.

Припустимо, поточним каталогом є `/ home / pi`. Створимо новий каталог:

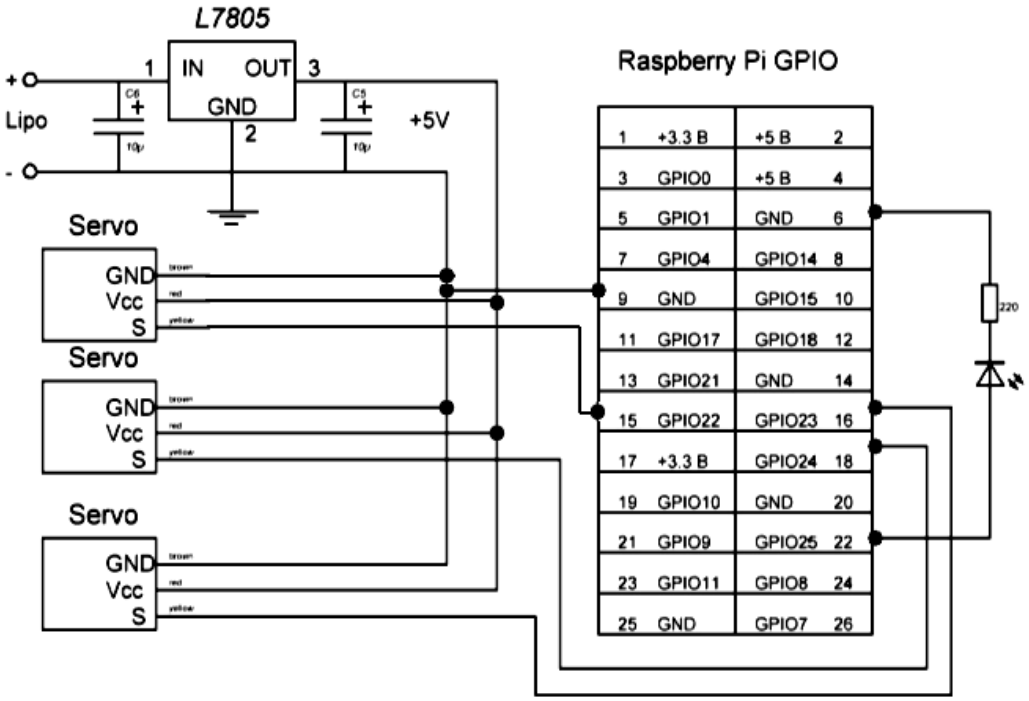
```
sudo mkdir /opt/mjpg-streamer
```

Установимо бібліотеку `libjpeg62-dev`:

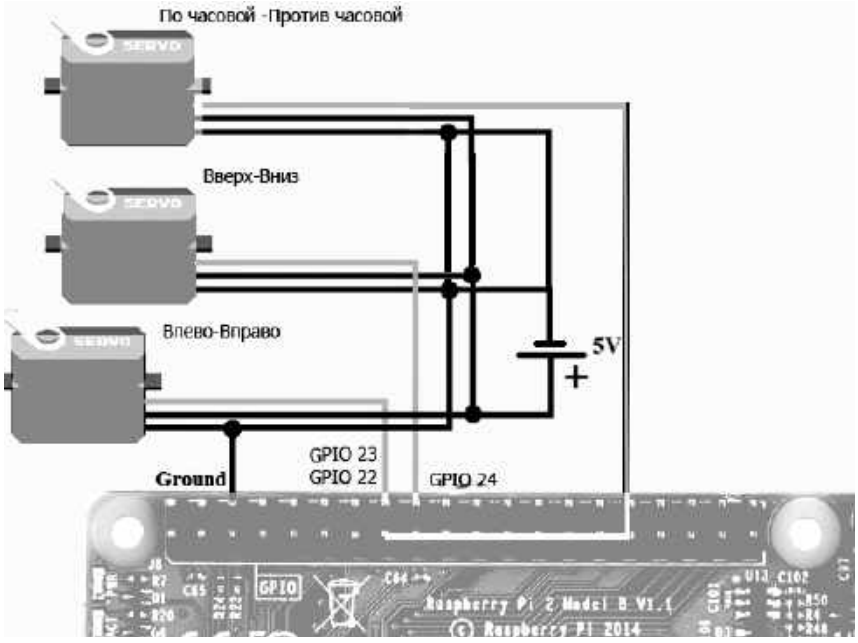
```
sudo apt-get install libjpeg62-dev
```

Установимо `cmake`:

```
sudo apt-get install cmake
```



а



б

Рис. 5.3

Завантажимо `mjpg-streamer` з плагіном `raspicam`:

```
git clone https://github.com/jacksonliam/mjpg-streamer.git
~/mjpg-streamer
```

Зайдемо в директорію:

```
cd ~/mjpg-streamer/mjpg-streamer-experimental
```

Виконаємо компіляцію

```
make clean all
```

Перемістимо файли й видалимо робочу директорію:

```
sudo mv ~/mjpg-streamer/mjpg-streamer-experimental
/opt/mjpg-streamer
```

```
sudo rm -rf ~/mjpg-streamerСоздаем файл /opt/mjpg-
streamer/start_stream.sh для старта MJPG-Streamer:
```

Далі створимо два файли для запуску й припинення трансляції відео.

Створимо файл /opt/mjpg-streamer/start_stream.sh для старту MJPG-Streamer:

```
#!/bin/bash
if pgrep mjpg_streamer > /dev/null
then
echo "mjpg_streamer already running"
else
LD_LIBRARY_PATH=/opt/mjpg-streamer/ /opt/mjpg-
streamer/mjpg_streamer -i "input_raspicam.so -fps 10 -q 50
-x 640 -y 480" -o "output_http.so -p 8080 -w /opt/mjpg-
streamer/www" > /dev/null 2>&1&
echo "mjpg_streamer started"
fi
```

Створмо файл /opt/mjpg-streamer/stop_stream.sh для припинення MJPG-Streamer:

```
#!/bin/bash
if pgrep mjpg_streamer
then
kill $(pgrep mjpg_streamer) > /dev/null 2>&1
echo "mjpg_streamer stopped"
else
echo "mjpg_streamer not running"
fi
```

Таким чином, для запуску трансляції необхідно в терміналі записати команду

```
/opt/mjpg-streamer/start_stream.sh
```

для припинення – команду

```
/opt/mjpg-streamer/stop_stream.sh
```

Керування сервоприводами реалізується з допомогою програмних ***ШІМ***-сигналів. Для цього необхідно написати скрипт мовою ***Python***, у якому записати призначення контактів і макроси. Уміст файла camera.py має такий виглад:

```
# Imports
import webiopi
import datetime
GPIO = webiopi.GPIO
SERVO1=22 #Сервопривід 1
SERVO2=23 #Сервопривід 2
SERVO3=24 #Сервопривід 3
SERRL = 0 # Кут повороту “Рискання”
STE = 0 # Крок кута повороту
SERUD = 0 # Кут повороту “Тангаж”
SERCWCCW = 0 # Кут повороту “Крен”
# Функція настроєння автоматичного виклику при запуску
WebIOPi
def setup():
# установити GPIO, що використовує інформацію для виведен-
ня
    GPIO.setFunction(SERVO1, GPIO.PWM)
    GPIO.setFunction(SERVO2, GPIO.PWM)
    GPIO.setFunction(SERVO3, GPIO.PWM)
    GPIO.pwmWriteAngle(SERVO1, 0)
    GPIO.pwmWriteAngle(SERVO2, 0)
    GPIO.pwmWriteAngle(SERVO3, 0)
    webiopi.sleep(0.25)
    GPIO.setFunction(SERVO1, GPIO.OUT)
    GPIO.setFunction(SERVO2, GPIO.OUT)
    GPIO.setFunction(SERVO3, GPIO.OUT)
# Функція destroy викликається при завершенні роботи WebI-
OPi
def destroy():
   GPIO.digitalWrite(SERVO1, GPIO.OUT)
   GPIO.digitalWrite(SERVO2, GPIO.OUT)
@webiopi.macro
def steps():
   global SERRL,STE
   return "%d" % SERRL
@webiopi.macro
def steps1():
   global SERUD,STE
   return "%d" % SERUD
@webiopi.macro
```

```
def steps2():
   global SERCWCCW,STE
   return "%d" % SERCWCCW
@webiopi.macro
def gets():
   global SERRL
   return "%d" % STE
#--------------------Рискання --------------
#----Уліво----
@webiopi.macro
def lef(on):
   global SERRL,STE
   STE = int(on)
   GPIO.setFunction(SERVO1, GPIO.PWM)
   SERRL=SERRL+STE
   GPIO.pwmWriteAngle(SERVO1, SERRL)
   webiopi.sleep(0.25)
   if (abs(SERRL)>=90):
       SERRL=90
       GPIO.pwmWriteAngle(SERVO1, SERRL)
       webiopi.sleep(0.25)
       GPIO.setFunction(SERVO1, GPIO.OUT)
   return gets()
#----Управо----
@webiopi.macro
def ri(on):
   global SERRL,STE
   STE = int(on)
   GPIO.setFunction(SERVO1, GPIO.PWM)
   SERRL=SERRL-STE
   GPIO.pwmWriteAngle(SERVO1, SERRL)
   webiopi.sleep(0.25)
   if (abs(SERRL)>=90):
       SERRL=-90
       GPIO.pwmWriteAngle(SERVO1, SERRL)
       webiopi.sleep(0.25)
       GPIO.setFunction(SERVO1, GPIO.OUT)
   return gets()
#--------------------ТАНГАЖ-------------
@webiopi.macro
```

```
#----Униз----
def downs(on):
   global SERUD,STE
   STE = int(on)
   GPIO.setFunction(SERVO2, GPIO.PWM)
   SERUD=SERUD+STE
   GPIO.pwmWriteAngle(SERVO2, SERUD)
   webiopi.sleep(0.25)
   if (abs(SERUD)>=90):
       SERUD=90
       GPIO.pwmWriteAngle(SERVO2, SERUD)
       webiopi.sleep(0.25)
       GPIO.setFunction(SERVO2, GPIO.OUT)
   return gets()
#----Уверх----
@webiopi.macro
def ups(on):
   global SERUD,STE
   STE = int(on)
   GPIO.setFunction(SERVO2, GPIO.PWM)
   SERUD=SERUD-STE
   GPIO.pwmWriteAngle(SERVO2, SERUD)
   webiopi.sleep(0.25)
   if (abs(SERUD)>=90):
       SERUD=-90
       GPIO.pwmWriteAngle(SERVO2, SERUD)
       webiopi.sleep(0.25)
       GPIO.setFunction(SERVO2, GPIO.OUT)
   return gets()
#--------------------КРЕН-------------
#----За ходом годинникової стрілки----
@webiopi.macro
def cws(on):
   global SERCWCCW,STE
   STE = int(on)
   GPIO.setFunction(SERVO3, GPIO.PWM)
   SERCWCCW=SERCWCCW+STE
   GPIO.pwmWriteAngle(SERVO3, SERCWCCW)
   webiopi.sleep(0.25)
   if (abs(SERCWCCW)>=90):
```

```
        SERCWCCW=90
        GPIO.pwmWriteAngle(SERVO3, SERCWCCW)
        webiopi.sleep(0.25)
        GPIO.setFunction(SERVO3, GPIO.OUT)
    return gets()
#----Проти ходу годинникової стрілки ----
@webiopi.macro
def ccws(on):
    global SERCWCCW,STE
    STE = int(on)
    GPIO.setFunction(SERVO3, GPIO.PWM)
    SERUD=SERUD-STE
    GPIO.pwmWriteAngle(SERVO3, SERCWCCW)
    webiopi.sleep(0.25)
    if (abs(SERCWCCW)>=90):
        SERCWCCW=-90
        GPIO.pwmWriteAngle(SERVO3, SERCWCCW)
        webiopi.sleep(0.25)
        GPIO.setFunction(SERVO3, GPIO.OUT)
    return gets()
```

Керувальні контакти сервоприводів підімкнено до виводів 15 (***GPIO22***), 16 (***GPIO23***) і 18 (***GPIO24***)

Далі необхідно написати веб-інтерфейс, з допомогою якого будуть керуватися сервоприводи шляхом виклику макросів з файла ***Python***. Код файла цього веб-інтерфейсу є таким:

```
DOCTYPE    html    PUBLIC    "-//W3C//DTD    HTML    4.01
<!Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
    <meta name="viewport" content = "width = 420, user-
scalable = no" />
    <title>WebIOPi | Devices Monitor</title>
<link rel="stylesheet" href="style.css">
<script type="text/javascript">
setInterval ("callMacro_steps()",500);{}
function callMacro_steps(){
webiopi().callMacro("steps", [], macro_steps_Callback);}
function macro_steps_Callback(macro, args, data) {
```

```
$("#inf_risk").text("Right Left: "+data);}
setInterval ("callMacro_steps1()", 500);{}
function callMacro_steps1(){webiopi().callMacro("steps1",
[], macro_steps1_Callback);}
function macro_steps1_Callback(macro, args, data) {
$("#inf_tang").text("Up Down: "+data);}
setInterval ("callMacro_steps2()", 500);{}
function callMacro_steps2(){
webiopi().callMacro("steps2", [], macro_steps1_Callback);}
function macro_steps2_Callback(macro, args, data) {
$("#inf_kren").text("CW CCW: "+data);}
 webiopi().ready(function() {
        //Наступна функція оброблятиме дані, отримані з
набору/отримує макрос
        var update = function(macro, args, response) {
                    var servo=response;
                    //Наступні рядки використовують
функції jQuery
                    $("#inputOn").val(servo);
        }
        // Відразу ж виклик отримує макрос для оновлення
призначеного для користувача інтерфейсу з поточними зна-
ченнями
        // «gets» належить до імені макроса
                  // [] - порожній масив, тому що макрос не
приймає жодних аргументів
                  // update - це функція зворотного викли-
ку, визначена вище
                webiopi().callMacro("gets", [], update);
        // Створення кнопки для lef макроса
     //----------РИСКАННЯ
        var sendButton1 = webio-
pi().createButton("sendButton1", "Уліво", function() {
           // Аргументи, відправлені макросу
                    var servo = $("#inputOn").val();
            // Виклик макроса
                    webiopi().callMacro("lef", servo, up-
date);
                });
        // Додайте кнопку в полі керування з допомогою
функції jQuery
```

```
                $("#btn_risk").append(sendButton1);
        // Створення кнопки для ri макроса
        var sendButton2 = webio-
pi().createButton("sendButton2", "Управо",function(){
           // Аргументи, відправлені макросу
                    var servo = $("#inputOn").val();
            // Виклик макроса
                    webiopi().callMacro("ri", servo, up-
date);
                });
        // Додайте кнопку в полі керування з допомогою
функції jQuery
                $("#btn_risk").append(sendButton2);
//-------------ТАНГАЖ
        var sendButton3 = webi-
opi().createButton("sendButton3", "Уверх", function() {
                    var servo = $("#inputOn").val();
                    webiopi().callMacro("downs", servo,
update);
                });
                $("#btn_tang").append(sendButton3);
        var sendButton4 = webi-
opi().createButton("sendButton4", "Униз", function() {
                    var servo = $("#inputOn").val();
                    webiopi().callMacro("ups", servo, up-
date);
                });
                $("#btn_tang").append(sendButton4);
//-------------КРЕН
        var sendButton5 = webi-
opi().createButton("sendButton5", "CWs", function() {
                    var servo = $("#inputOn").val();
                    webiopi().callMacro("cws", servo, up-
date);
                });
                $("#kren").append(sendButton5);
        var sendButton6 =webi-
opi().createButton("sendButton6", "CCWs", function() {
                    var servo = $("#inputOn").val();
                 webiopi().callMacro("ccws", servo, up-
date);
                });
                $("#kren").append(sendButton6);
        webiopi().refreshGPIO(true);
```

```
        });
    </script>
</head>
<body>
<div class="header">
    <div class="nav">
        <h1 id="kaf">Кафедра систем управління літальними
аппаратами</h1>
         <h1 id="free_space"> </h1>
        <h1 id="inst">Національний аерокосмічний університет <br> им. М.Є. Жуковського</h1>
    </div>
</div>
<div id="content">
    <div id="angles">
            </div>
    <input type="text" maxlength="3" size="3" id="inputOn"
placeholder="Задайте кут">
    <div class="risk">
        <h1>Рискання:</h1>
        <span id="inf_risk"></span>
        <div id="btn_risk"> </div>
    </div>
    <div class="tang">
        <h1>Тангаж:</h1>
        <span id="inf_tang"></span>
        <div id="btn_tang"> </div>
    </div>
    <div class="kren">
        <h1>Крен:</h1>
        <span id="inf_kren"></span>
        <div id="btn_kren"></div>
    </div>
     <img src="" id="" alt="">
</div>
</body>
</html>
```

Веб-інтерфейс має досить простий вигляд: шість кнопок (по дві на кожній осі), поле введення кута, на який необхідно повернути камеру, а також виведений поточний попередній кут, на який у певний момент повернуто камеру (рис. 5.4).

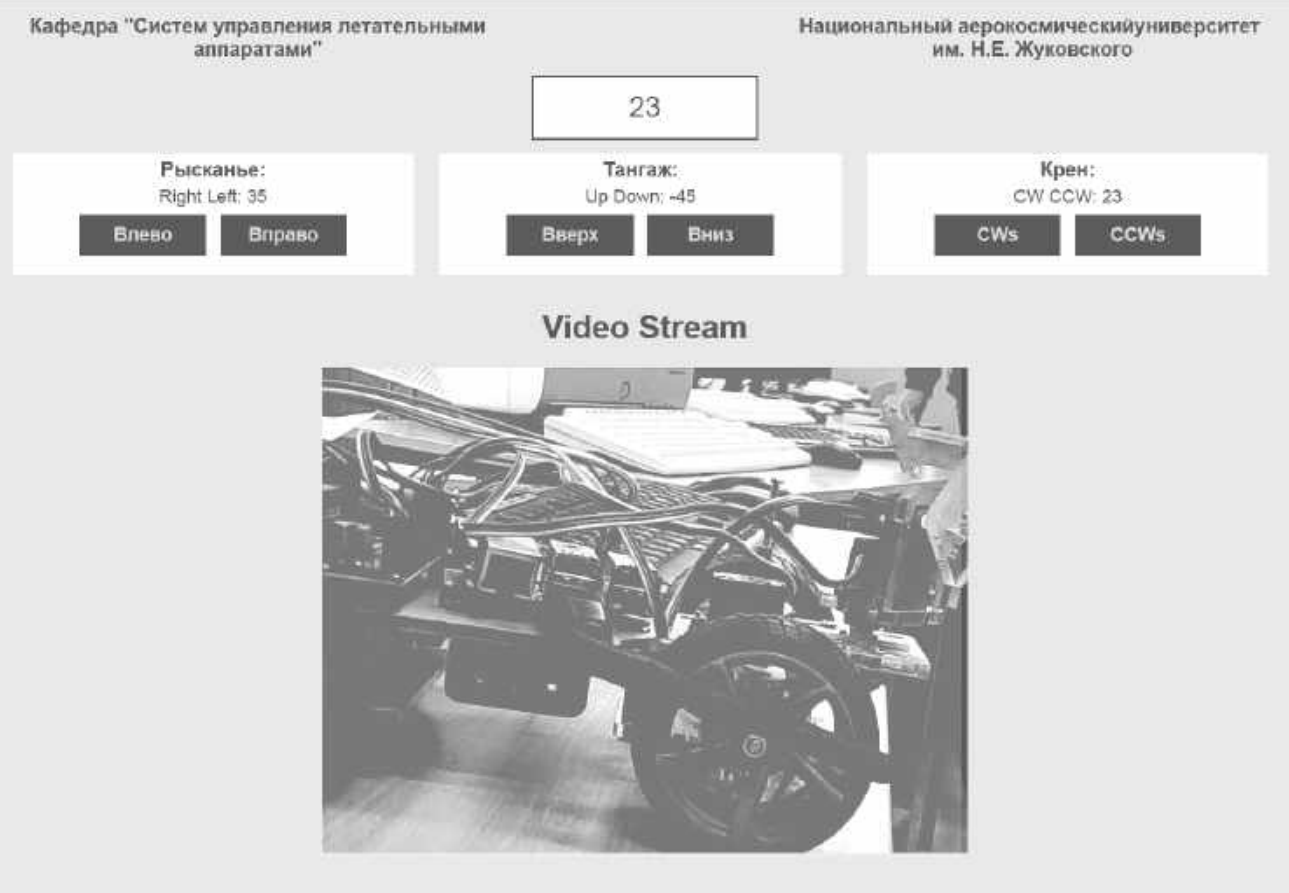


Рис. 5.4

Завдання для самостійної роботи

Виконати завдання з дистанційного керування роботою відеокамери з допомогою сервоприводів і портів ***GPIO*** комп'ютера ***Raspberry Pi***. Для цього необхідно:

- скласти схему керування відповідно до рекомендацій і схеми принципової електричної (див. рис. 5.3). Окрім цього необхідно перевірити правильність підімкнення сервоприводів до підвісів камери;
- перевірити коректність роботи програмного забезпечення проекту разом з апаратними засобами.

Звіт про виконання лабораторної роботи має містити:

- схеми керування електричні принципові й схеми монтажні;
- тексти програмних кодів, які використовуються в цій роботі;
- виводи за результатами проведених досліджень.

Контрольні запитання

1. Як здійснюється доступ до портів ***GPIO*** через веб-інтерфейс?
2. Опишіть методику установлення фреймворка ***Webiopi*** на ***ОС Raspbian*** і призначеного для користувача пароля.

3. Яких запобіжних заходів необхідно дотримуватися під час роботі з інтерфейсом ***GPIO****?*

4. У чому полягає настроєння ***Webiopi****?* Перелічіть основні пункти настроєння.

5. Для чого призначено бібліотеку `Javascript` при створенні веб-інтерфейсу?

6. Наведіть приклади основних функцій бібліотеки `webiopi.js`.

7. Якими є принципи роботи сервоприводів у системах керування? Наведіть основні характеристики сервоприводів.

8. Опишіть основні команди настроєння входів (`INPUT`) і виходів (`OUTPUT`) керувальної програми мовою ***Python***.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. WebIOPi − The Raspberry Pi Internet of Things Framework [Electronic resource]. − Mode of access : http://webiopi.trouch.com/, 2016.
2. Internet of Things for Everyone [Electronic resource]. − Mode of access : https://www.weaved.com/ , 2016.
3. Комплексная система домашней автоматизации на Raspberry Pi [Электронный ресурс]. − Режим доступа: http://electromost.com/ , 2014.
4. Мясищев, А. А. Интернет электророзетка на основе мини компьютера Raspberry Pi и фреймворка WebIOPi. Практика для студентов [Электронный ресурс]. – Режим доступа : https://sites.google.com/site/webstm32/internet_rozetka, 2016.
5. Tutorial : Framework basis [Electronic resource]. − Mode of access : https://code.google.com/p/webiopi/wiki/Tutorial_Basis
6. Raspberry pi underclock, from 200 to 50MHz [Electronic resource]. − Mode of access : http://monkeyplush.blogspot.de/2012/09/ raspberry-pi-underclock.html
7. Reducing power consumption of a raspberry Pi [Electronic resource]. − Mode of access : http://www.bitwizard.nl/wiki/index.php?title=Reducing_power_consumption_of_a_raspberry_Pi
8. How Much Less Power does the Raspberry Pi B+ use than the old model B? [Electronic resource]. − Mode of access : http://raspi.tv/2014/how-much-less-power-does-the-raspberry-pi-b-use-than-the-old-model-b
9. Webbiopi. Installation [Electronic resource]. − Mode of access : https://code.google.com/p/webiopi/wiki/INSTALL
10. Dr. Bob Davidov. Компьютерные средства систем управления. Raspberry Pi [Электронный ресурс] – Режим доступа : http//portalnp.ru/2013/12/1691
11. Dr. Bob Davidov. Подключение периферии к среде разработки систем управления МатЛАБ [Электронный ресурс]. – Режим доступа : http://portalnp.ru/2014/03/1783
12. Dr. Bob Davidov. Импорт и экспорт МатЛАБ данных через Raspberry Pi [Электронный ресурс]. – Режим доступа :http://portalnp.ru/2014/04/1858
13. Dr. Bob Davidov. Построение RT системы управления на базе компьютера Raspberry Pi [Электронный ресурс]. – Режим доступа : http://portalnp.ru/2014/04/1867
14. Dr. Bob Davidov. Компьютерные технологии управления в технических системах http[Электронный ресурс]. – Режим доступа : //portalnp.ru/author/bobdavidov

ЗМІСТ

Навчальне видання

Краснов Леонід Олександрович
Дергачов Костянтин Юрійович
Плахотний Олександр Вікторович
Пявка Ірина Олександрівна

ОСНОВИ ПОБУДОВИ СУЧАСНИХ МОБІЛЬНИХ СИСТЕМ ТЕХНІЧНОГО ЗОРУ

Частина 3

ЛАБОРАТОРНІ РОБОТИ

Редактор О. Ф. Серьожкіна

Зв. план, 2019
Підписано до друку 12.04.2019
Формат 60х84 1/16. Папір офс. № 2. Офс. друк
Ум. друк. арк. 4. Обл.-вид. арк. 4,5. Наклад 100 пр.
Замовлення 95. Ціна вільна

Видавець і виготовлювач
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»
61070, Харків-70, вул. Чкалова, 17
http://www.khai.edu
Видавничий центр «ХАІ»
61070, Харків-70, вул. Чкалова, 17
izdat@khai.edu

Свідоцтво про внесення суб`єкта видавничої справи
до Державного реєстру видавців, виготовлювачів і розповсюджувачів
видавничої продукції сер. ДК № 391 від 30.03.2001