Спочатку команда, потім механізм

Як це працює

Дві окремі перевірки

Спершу перевіряємо, чи програма видає правильні команди. Потім — чи механізм може виконати їх без упору. Важіль у повітрі показує логіку, але ще не доводить, що серво підніме кришку.

Петля несе кришку

Кришка спирається на власну петлю, а серво рухає її через тягу. Довжина тяги й точки кріплення впливають на потрібне зусилля та кути. Команди 60 і 120 поки лише дві цілі для вільного серво; готовій механіці можуть знадобитися інші перевірені значення.

Зроби

1. Підготуй Uno, серво, кнопку, HC-SR04, макетну плату та джерело 5 V.
2. Почни без кришки і тяги.
3. Повтори центральне положення серво.

Очікуваний результат

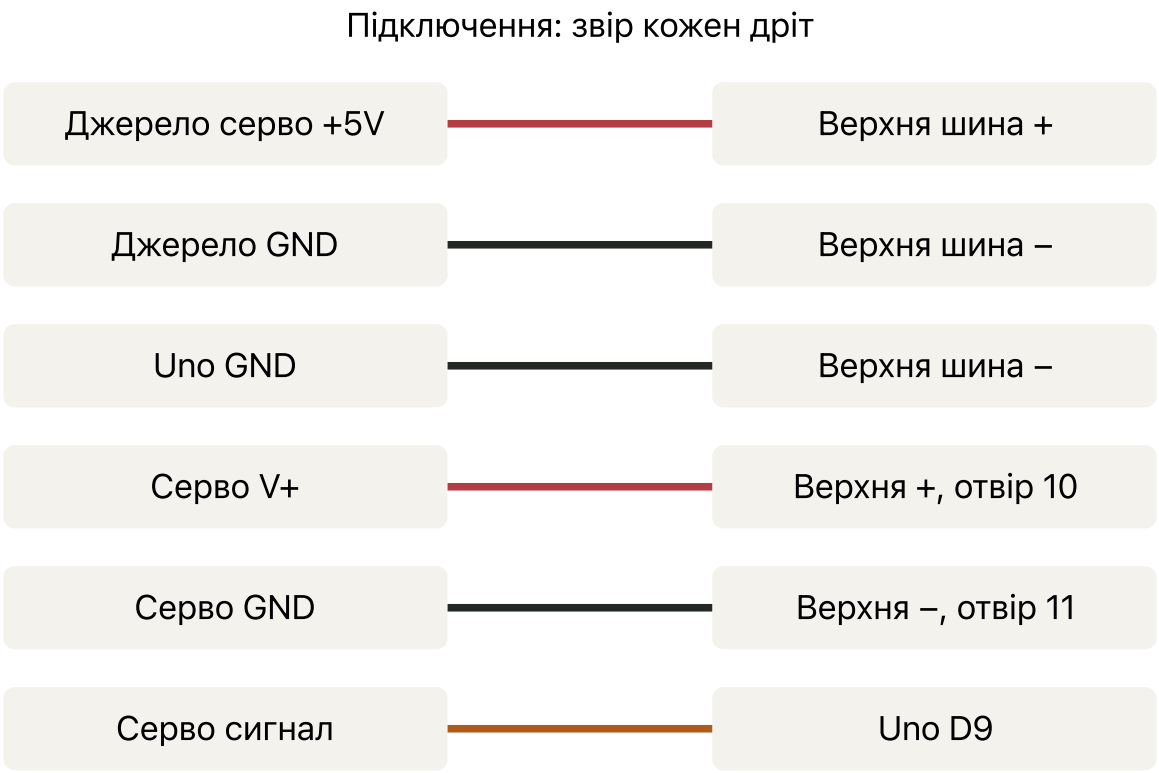
Електроніку можна дослідити до виготовлення кріплення.

Що можна змінити

Для фізичної кришки потрібні перевірена петля, легка кришка і кріплення. До їх готовності демонструй положення важелем.

У Wokwi: відкрий новий Arduino Uno, перенеси diagram.json, sketch/sketch.ino та libraries.txt з ZIP потрібного кроку у відповідні вкладки, натисни ▶. На справжній платі спочатку перевір підключення з учителем. Кожен код замінює весь попередній скетч.

Підключення справжнього комплекту



Два плюси розділені, GND спільний

На макетній платі верхня + — тільки джерело серво, нижня + — тільки Uno. Об’єднуємо мінусові шини. Перевір розриви шин; за потреби з’єднай половини тієї самої шини. У Wokwi верхня + живиться від Uno лише як спрощення. На справжній платі цей дріт заміни на +5 V окремого джерела; його мінус додай на верхню –.

Крок 1: код · 1

```
#include <Arduino.h>
#include <Servo.h>
const byte SERVO_PIN = 9;
const int ANGLE = 90; // Початковий кут без стріли
Servo arm;

void setup() {
  arm.write(ANGLE); // Задаємо початкове положення
  arm.attach(SERVO_PIN); // Сигнал керування через D9
}

void loop() {
}
```

Відкривай кнопкою

Як це працює

Чому натискання означає LOW

INPUT_PULLUP вмикає внутрішню підтяжку входу до високого рівня. Поки кнопка розімкнена, читаємо HIGH. Натискання з'єднує `D2` із `GND` — читаємо LOW. Тому умова натискання перевіряє саме LOW.

Простеж рішення програми

На кожному проході loop програма читає кнопку. Натиснута — надсилає OPEN_ANGLE, відпущена — CLOSED_ANGLE. Тут немає запам'ятовування натискання: відкриту команду отримуємо лише поки тримаємо кнопку.

Зроби

1. Вимкни живлення.
2. Підключи кнопку між `D2` і `GND`; у Wokwi це 1.l та 2.l.
3. Завантаж код.
4. Натисни і відпусти кнопку.

Очікуваний результат

Натиснута — команда OPEN_ANGLE; відпущена — CLOSED_ANGLE. Перевіряємо без тяги.

Що можна змінити

Кути: тільки перевірений діапазон 60–120. Кнопку тримай натиснутою для відкритого положення.

Підключення справжнього комплекту



Два плюси розділені, GND спільний

На макетній платі верхня + — тільки джерело серво, нижня + — тільки Uno. Об’єднуємо мінусові шини. Перевір розриви шин; за потреби з’єднай половини тієї самої шини. У Wokwi верхня + живиться від Uno лише як спрощення. На справжній платі цей дріт заміни на +5 V окремого джерела; його мінус додай на верхню –.

Крок 2: код · 1

```
#include <Arduino.h>
#include <Servo.h>
const byte SERVO_PIN = 9;
const byte BUTTON_PIN = 2;
const int CLOSED_ANGLE = 60;
const int OPEN_ANGLE = 120;
Servo arm;

void setup() {
  pinMode(BUTTON_PIN, INPUT_PULLUP);
  arm.write(CLOSED_ANGLE);
  arm.attach(SERVO_PIN);
}

void loop() {
  bool pressed = digitalRead(BUTTON_PIN) == LOW;

  if (pressed) {
    arm.write(OPEN_ANGLE);
  } else {
    arm.write(CLOSED_ANGLE);
  }

  delay(20);
}
```

Додай датчик замість кнопки

Як це працює

Датчик замінює рішення людини

Кнопка давала відповідь «натиснуто чи ні». HC-SR04 повертає час відлуння, з якого програма обчислює відстань. Тепер умова відкривання — надійний показ у зоні від 2 до 15 см.

Чому поки немає закривання

Спочатку перевіряємо одну реакцію: наближення відкриває. Віддалення ще не дає зворотної команди. Якщо відлуння не отримано, ми не знаємо, чи справді немає предмета: причина може бути в його поверхні, напрямку або підключенні.

Зроби

1. Вимкни живлення.
2. Від'єднай кнопку.
3. Підключи HC-SR04 за таблицею: TRIG `D7`, ECHO `D6`.
4. Завантаж цей код і відкрий монітор 115200.

Очікуваний результат

Картон на 2–15 см відкриває; у цьому кроці автоматичного закривання ще немає. Reset повертає початкову команду.

Що можна змінити

OPEN_CM = 15; не піднось предмет ближче 2 см.

Підключення справжнього комплекту



Два плюси розділені, GND спільний

На макетній платі верхня + — тільки джерело серво, нижня + — тільки Uno. Об’єднуємо мінусові шини. Перевір розриви шин; за потреби з’єднай половини тієї самої шини. У Wokwi верхня + живиться від Uno лише як спрощення. На справжній платі цей дріт заміни на +5 V окремого джерела; його мінус додай на верхню –.

Крок 3: код · 1

```
#include <Arduino.h>
const byte TRIG_PIN = 7;
const byte ECHO_PIN = 6;
#include <Servo.h>
const byte SERVO_PIN = 9;
const int CLOSED_ANGLE = 60; // Звірити без стріли
const int OPEN_ANGLE = 120; // Звірити напрямок на своєму серво
const float OPEN_CM = 15; // Відкрити при 2—15 см
Servo arm;

float readDistanceCm() {
  digitalWrite(TRIG_PIN, LOW); // Готуємо короткий імпульс
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH); // Запускаємо вимірювання
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  unsigned long echoUs = pulseIn(ECHO_PIN, HIGH, 25000UL);
  float cm = echoUs / 58.0; // Шлях туди й назад уже врахований

  if (echoUs == 0 || cm < 2 || cm > 400) {
    return -1; // Немає надійного вимірювання
  }

  return cm;
}

void setup() {
```

Крок 3: код · 2

```
  pinMode(TRIG_PIN, OUTPUT); // TRIG — команда датчику
  pinMode(ECHO_PIN, INPUT); // ECHO — відповідь датчика
  digitalWrite(TRIG_PIN, LOW);
  Serial.begin(115200); // Така сама швидкість у моніторі
  arm.write(CLOSED_ANGLE);
  arm.attach(SERVO_PIN);
}

void loop() {
  float cm = readDistanceCm();

  if (cm < 0) {
    Serial.println("No echo");
  } else {
    Serial.print("Distance: ");
    Serial.print(cm, 1);
    Serial.println(" cm");
  }


  if (cm >= 2 && cm <= OPEN_CM) {
    arm.write(OPEN_ANGLE); // Відкриваємо; назад лише після Reset
  }

  delay(100); // Нове вимірювання приблизно 10 разів/с
}
```

Дочекайся вільного інтервалу

Як це працює

Навіщо дві відстані

Біля одного порога показ може коливатися, наприклад 14, 16, 15 см. Тому відкривання дозволяємо до 15 см, а початок закривання — лише далі 25 см. Між ними є проміжна зона, яка не дає постійно перемикати рішення.

Як працює відлік

Коли зона стала вільною, програма запам'ятовує час millis(). Під час наступних вимірювань порівнює, скільки минуло. Лише три безперервні секунди з надійними показами далі 25 см дозволяють закрити. Наближення, проміжна зона або помилка скидають відлік.

Чому тут не delay(3000)

Під час звичайного delay програма не перевіряла б нову відстань. Відлік через millis дозволяє продовжувати вимірювання й помітити повернення предмета, поки три секунди ще не минули.

Зроби

1. Завантаж повний код.
2. Постав картон на 10 см.
3. Відсунь його на 40 см.
4. Поверни на 10 см через дві секунди.
5. Знову відсунь на 40 см на повні три секунди.

Очікуваний результат

Повернення предмета перериває відлік. Потрібен новий безперервний інтервал.

Що можна змінити

CLEAR_MS: 3000–5000; CLEAR_CM має бути більшим за OPEN_CM.

Підключення справжнього комплекту



Два плюси розділені, GND спільний

На макетній платі верхня + — тільки джерело серво, нижня + — тільки Uno. Об’єднуємо мінусові шини. Перевір розриви шин; за потреби з’єднай половини тієї самої шини. У Wokwi верхня + живиться від Uno лише як спрощення. На справжній платі цей дріт заміни на +5 V окремого джерела; його мінус додай на верхню –.

Крок 4: код · 1

```
#include <Arduino.h>
const byte TRIG_PIN = 7;
const byte ECHO_PIN = 6;
#include <Servo.h>
const byte SERVO_PIN = 9;
const int CLOSED_ANGLE = 60; // Звірити без тяги
const int OPEN_ANGLE = 120; // Звірити напрямок без тяги на своєму серво
const float OPEN_CM = 15; // Відкрити при 2—15 см
Servo arm;
const float CLEAR_CM = 25; // Далі цієї межі зона вільна
const unsigned long CLEAR_MS = 3000; // Безперервний вільний інтервал
bool isOpen = false;
bool clearing = false;
unsigned long clearAt = 0;

float readDistanceCm() {
  digitalWrite(TRIG_PIN, LOW); // Готуємо короткий імпульс
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH); // Запускаємо вимірювання
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  unsigned long echoUs = pulseIn(ECHO_PIN, HIGH, 25000UL);
  float cm = echoUs / 58.0; // Шлях туди й назад уже врахований

  if (echoUs == 0 || cm < 2 || cm > 400) {
    return -1; // Немає надійного вимірювання
  }

  return cm;
}

void setup() {
  pinMode(TRIG_PIN, OUTPUT); // TRIG — команда датчику
  pinMode(ECHO_PIN, INPUT); // ECHO — відповідь датчика
  digitalWrite(TRIG_PIN, LOW);
  Serial.begin(115200); // Така сама швидкість у моніторі
  arm.write(CLOSED_ANGLE);
  arm.attach(SERVO_PIN);
}
```

Крок 4: код · 2

```
void loop() {
  float cm = readDistanceCm();

  if (cm < 0) {
    Serial.println("No echo");
  } else {
    Serial.print("Distance: ");
    Serial.print(cm, 1);
    Serial.println(" cm");
  }

  unsigned long now = millis(); // Час після вимірювання

  if (cm < 0) {
    clearing = false; // Помилка не означає вільну зону
  } else if (cm <= OPEN_CM) {
    arm.write(OPEN_ANGLE);
    isOpen = true;
    clearing = false;
  } else if (isOpen && cm > CLEAR_CM) {

    if (!clearing) {
      clearAt = now; // Початок нового вільного інтервалу
      clearing = true;
    }

    if (now - clearAt >= CLEAR_MS) {
      arm.write(CLOSED_ANGLE);
      isOpen = false;
      clearing = false;
    }

  } else {
    clearing = false; // Проміжна зона скасовує відлік
  }

  delay(100); // Нове вимірювання приблизно 10 разів/с
}
```

Проведи перевірку перед механікою

Як це працює

Перевіряємо різні ситуації

10 см перевіряє відкривання. 20 см — проміжну зону без закривання. 40 см — відлік вільного інтервалу. Ненадійний показ перевіряє, чи програма не плутає помилку з порожнім простором.

Межа навчальної моделі

Один датчик дивиться лише у своєму напрямку. Він не бачить усю траєкторію кришки й не вимірює силу затискання. Тому перевірка програми не замінює перевірку петлі, тяги та вільного руху. До цього етапу серво працює без кришки.

Зроби

1. Перевір 10, 20 і 40 см.
2. У Wokwi постав 1 см: має бути No echo без команди закривання.
3. Запиши результати у зошит.
4. Збережи скетч як Auto_lid.
5. Попроси вчителя перевірити механізм без тяги, а потім із легкою кришкою.

Очікуваний результат

Програма реагує на всі три зони та помилку. Руки прибрані з траєкторії; живлення можна одразу вимкнути.

Що можна змінити

При гудінні або упорі негайно вимкни обидва джерела. Не збільшуй силу та не розширюй кути навмання.

Підключення справжнього комплекту



Два плюси розділені, GND спільний

На макетній платі верхня + — тільки джерело серво, нижня + — тільки Uno. Об’єднуємо мінусові шини. Перевір розриви шин; за потреби з’єднай половини тієї самої шини. У Wokwi верхня + живиться від Uno лише як спрощення. На справжній платі цей дріт заміни на +5 V окремого джерела; його мінус додай на верхню –.

Крок 5: код · 1

```
#include <Arduino.h>
const byte TRIG_PIN = 7;
const byte ECHO_PIN = 6;
#include <Servo.h>
const byte SERVO_PIN = 9;
const int CLOSED_ANGLE = 60; // Звірити без тяги
const int OPEN_ANGLE = 120; // Звірити напрямок без тяги на своєму серво
const float OPEN_CM = 15; // Відкрити при 2—15 см
Servo arm;
const float CLEAR_CM = 25; // Далі цієї межі зона вільна
const unsigned long CLEAR_MS = 3000; // Безперервний вільний інтервал
bool isOpen = false;
bool clearing = false;
unsigned long clearAt = 0;

float readDistanceCm() {
  digitalWrite(TRIG_PIN, LOW); // Готуємо короткий імпульс
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH); // Запускаємо вимірювання
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  unsigned long echoUs = pulseIn(ECHO_PIN, HIGH, 25000UL);
  float cm = echoUs / 58.0; // Шлях туди й назад уже врахований

  if (echoUs == 0 || cm < 2 || cm > 400) {
    return -1; // Немає надійного вимірювання
  }

  return cm;
}

void setup() {
  pinMode(TRIG_PIN, OUTPUT); // TRIG — команда датчику
  pinMode(ECHO_PIN, INPUT); // ECHO — відповідь датчика
  digitalWrite(TRIG_PIN, LOW);
  Serial.begin(115200); // Така сама швидкість у моніторі
  arm.write(CLOSED_ANGLE);
  arm.attach(SERVO_PIN);
}
```

Крок 5: код · 2

```
void loop() {
  float cm = readDistanceCm();

  if (cm < 0) {
    Serial.println("No echo");
  } else {
    Serial.print("Distance: ");
    Serial.print(cm, 1);
    Serial.println(" cm");
  }

  unsigned long now = millis(); // Час після вимірювання

  if (cm < 0) {
    clearing = false; // Помилка не означає вільну зону
  } else if (cm <= OPEN_CM) {
    arm.write(OPEN_ANGLE);
    isOpen = true;
    clearing = false;
  } else if (isOpen && cm > CLEAR_CM) {

    if (!clearing) {
      clearAt = now; // Початок нового вільного інтервалу
      clearing = true;
    }

    if (now - clearAt >= CLEAR_MS) {
      arm.write(CLOSED_ANGLE);
      isOpen = false;
      clearing = false;
    }

  } else {
    clearing = false; // Проміжна зона скасовує відлік
  }

  delay(100); // Нове вимірювання приблизно 10 разів/с
}
```

Джерела

Стаття й ілюстрації — DroneBot Workshop · Відео автора. Практичні завдання та програми адаптовано для цього курсу.

Далі: Ультразвуковий сканер →